HITACHI
Inspire the Next

OpenTP1 Version 7
分散トランザクション処理機能

OpenTP1 プロトコル TP1/NET/TCP/IP 編

解説・手引・文法・操作書

3000-3-D70-41

前書き

■ 対象製品

・適用 OS：Red Hat Enterprise Linux Server 6 (32-bit x86)，Red Hat Enterprise Linux Server 6 (64-bit x86_64)，Red Hat Enterprise Linux Server 7 (64-bit x86_64)，Red Hat Enterprise Linux Server 8 (64-bit x86_64)（32 ビット用）

P-8164-3111 uCosminexus TP1/Message Control 07-51

P-8164-3211 uCosminexus TP1/NET/Library 07-51

P-F8164-3211C uCosminexus TP1/NET/TCP/IP 07-50

P-F8164-3211D uCosminexus TP1/NET/High Availability 07-50

・適用 OS：Red Hat Enterprise Linux Server 6 (64-bit x86_64)，Red Hat Enterprise Linux Server 7 (64-bit x86_64)，Red Hat Enterprise Linux Server 8 (64-bit x86_64)（64 ビット用）

P-8264-3111 uCosminexus TP1/Message Control(64) 07-51

P-8264-3211 uCosminexus TP1/NET/Library(64) 07-51

P-F8264-3211C uCosminexus TP1/NET/TCP/IP(64) 07-50

P-F8264-3211D uCosminexus TP1/NET/High Availability(64) 07-50

・適用 OS：AIX V6.1，AIX V7.1，AIX V7.2（32 ビット用）

P-1M64-3141 uCosminexus TP1/Message Control 07-51

P-1M64-3241 uCosminexus TP1/NET/Library 07-51

P-F1M64-3241C uCosminexus TP1/NET/TCP/IP 07-50

P-F1M64-3241D uCosminexus TP1/NET/High Availability 07-50

・適用 OS：AIX V6.1，AIX V7.1，AIX V7.2（64 ビット用）

P-1M64-4121 uCosminexus TP1/Message Control(64) 07-51

P-1M64-4221 uCosminexus TP1/NET/Library(64) 07-51

P-F1M64-4221C uCosminexus TP1/NET/TCP/IP(64) 07-50

P-F1M64-4221D uCosminexus TP1/NET/High Availability(64) 07-50

・適用 OS：HP-UX 11i V3 (IPF)（32 ビット用）

P-1J64-3171 uCosminexus TP1/Message Control 07-51

P-1J64-3271 uCosminexus TP1/NET/Library 07-51

P-F1J64-3271C uCosminexus TP1/NET/TCP/IP 07-51

P-F1J64-3271D uCosminexus TP1/NET/High Availability 07-50

・適用 OS：HP-UX 11i V3 (IPF)（64 ビット用）

P-1J64-4171 uCosminexus TP1/Message Control(64) 07-51
P-1J64-4271 uCosminexus TP1/NET/Library(64) 07-51
P-F1J64-4271C uCosminexus TP1/NET/TCP/IP(64) 07-51
P-F1J64-4271D uCosminexus TP1/NET/High Availability(64) 07-50

・適用 OS：Windows 7，Windows 7 x64 Edition，Windows 8，Windows 8 x64 Edition，Windows 8.1，Windows 8.1 x64 Edition，Windows 10，Windows 10 x64 Edition，Windows Server 2008 R2，Windows Server 2012，Windows Server 2012 R2，Windows Server 2016，Windows Server 2019（32 ビット用）

P-2464-3164 uCosminexus TP1/Message Control 07-51
P-2464-3264 uCosminexus TP1/NET/Library 07-50
P-F2464-3264C uCosminexus TP1/NET/TCP/IP 07-51
P-F2464-3264D uCosminexus TP1/NET/High Availability 07-51

・適用 OS：Windows 7 x64 Edition，Windows 8 x64 Edition，Windows 8.1 x64 Edition，Windows 10 x64 Edition，Windows Server 2008 R2，Windows Server 2012，Windows Server 2012 R2，Windows Server 2016，Windows Server 2019（64 ビット用）

P-2964-3124 uCosminexus TP1/Message Control(64) 07-51
P-2964-3224 uCosminexus TP1/NET/Library(64) 07-50
P-F2964-3224C uCosminexus TP1/NET/TCP/IP(64) 07-51
P-F2964-3224D uCosminexus TP1/NET/High Availability(64) 07-50

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

■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。
なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

■ 発行

2021 年 4 月　3000-3-D70-41

■ 著作権

変更内容

変更内容（3000-3-D70-41） uCosminexus TP1/Message Control 07-51，uCosminexus TP1/NET/Library 07-51，uCosminexus TP1/NET/TCP/IP 07-51

追加・変更内容	変更個所
マニュアル訂正の内容を反映した。	－

単なる誤字・脱字などはお断りなく訂正しました。

変更内容（3000-3-D70-40） uCosminexus TP1/Message Control 07-50，uCosminexus TP1/NET/Library 07-50，uCosminexus TP1/NET/TCP/IP 07-50

追加・変更内容
マニュアル訂正の内容を反映した。

変更内容（3000-3-D70-30） uCosminexus TP1/Message Control 07-50，uCosminexus TP1/NET/Library 07-50，uCosminexus TP1/NET/TCP/IP 07-50

追加・変更内容
一つの MCF 通信プロセス当たりで使用できるコネクション数を拡張した。 これに伴い，次の定義コマンドの指定数の上限を拡張した。 • コネクション定義の開始（mcftalccn） • コネクション定義の終了（mcftalced） • 論理端末定義（mcftalcle）
次のオペランドのデフォルト値の説明を変更した。 • コネクション定義の開始（mcftalccn）の-f オプション releaselog オペランド cnerrlog オペランド
受信バッファの見積もり式の説明を変更した。
次のオペランドの指定値の上限を拡張した。 • システムサービス共通情報定義 max_socket_descriptors max_open_fds
MCF 定義オブジェクト解析コマンドの解析結果の省略値の扱いに説明を追加した。
次に示すバージョンの変更点を記載した。 TP1/NET/TCP/IP 07-50

変更内容（3000-3-D70-20） uCosminexus TP1/Message Control 07-05，uCosminexus TP1/NET/Library 07-05，uCosminexus TP1/NET/TCP/IP 07-04

追加・変更内容
問い合わせ応答形態をできるようにした。これに伴い，コネクション定義の開始（mcftalccn）に-l オプションを追加した。
同一ホスト内の他のプロセスとの通信に関する説明を追加した。
TP1/NET/TCP/IP で使用するポート番号の説明を追加した。
ディスクキューを使用した場合のシステム全体で使用できるコネクション数の上限に関する説明を追加した。
障害などによるコネクション解放が発生したときのコネクション解放形態を指定できるようにした。 これに伴い，コネクション定義の開始（mcftalccn -f）に cnrelease オペランドを追加した。
交代用コネクションと MCF 通信構成定義との関係の説明を追加した。
出力キューまたは入力キューの割り当て先としてディスクキューを使用する場合の説明を追加した。
受信バッファの空き待ちの説明を追加した。 また，受信バッファ数不足が発生したときの説明を変更した。
メッセージ送信完了監視時間を指定している場合の説明を追加した。
NULL またはヌル文字列設定時のコーディング例を追加した。また，ヌル文字列を設定できる引数に説明を追加した。
リターン値 DCMCFRTN_73001 およびステータスコード 73001 の説明を変更した。
リターン値 DCMCFRTN_NOMSG およびステータスコード 70904 の説明を変更した。
入力セグメント判定 UOC の標準提供 UOC を使用する場合の UAP および UOC に通知するメッセージの形式の説明を追加した。
ERREVT3 に設定するトランザクションブランチ ID の形式の説明を追加した。
コネクションを解放したとき，および障害によるコネクション切断が発生したときに出力するログメッセージの形式を指定できるようにした。 これに伴い，コネクション定義の開始（mcftalccn）に次のオペランドを追加した。 • -f オプションの releaselog オペランド • -f オプションの cnerrlog オペランド
受信バッファの見積もり式に注意事項を追加した。
MCF 性能検証用トレースの説明を追加した。
Linux 版または Solaris 版でファイル記述子数の最大数の条件を満たさない場合の注意事項を変更した。
定義オブジェクトファイルの出力先に読み取り権限を持つファイルがすでに存在する場合，定義オブジェクトファイルを上書きするかどうかを指定できるようにした。
UAP とシステム環境設定の関係について説明を追加した。
トレース情報量の見積もり式を変更した。
ホスト名または IP アドレスを変更する場合に，見直しが必要な定義と，変更手順について説明を追加した。

追加・変更内容
Windows 版で提供している定義例の説明を追加した。
運用コマンドを多数入力する場合の説明を追加した。
-s オプションを省略した場合の説明を追加した。
アプリケーション起動サービスに関する説明を追加した。
スタート関数を呼び出した後の注意事項を追加した。
UOC を使用しない場合の注意事項を追加した。
KFCA14830-E の出力に関する注意事項を変更した。
バージョン 5 からの移行に関する注意事項に次の項目を追加した。 • OpenTP1 終了中の相手システムからのコネクション確立 • 受信バッファ数不足発生時のメッセージログ • 受信バッファの空き待ち
メッセージ送達確認機能を使用する場合の一方送信メッセージの送信時の処理の流れを追加した。
Windows 版で提供するサンプルコーディングの格納場所について説明を変更した。

uCosminexus TP1/Message Control 07-05，uCosminexus TP1/NET/Library 07-05，uCosminexus TP1/NET/TCP/IP 07-03

追加・変更内容
コネクションの再確立時にメッセージの送信を抑止できるようにした。 これに伴い，論理端末定義（mcftalcle）に-d オプションを追加した。

変更内容（3000-3-D70-10） uCosminexus TP1/Message Control 07-02，uCosminexus TP1/NET/Library 07-03，uCosminexus TP1/NET/TCP/IP 07-02

追加・変更内容
論理端末の状態表示，閉塞，閉塞解除を，ライブラリ関数および COBOL-UAP 作成用プログラムインタフェースでできるようにした。これに伴い，次の関数を追加した。 • dc_mcf_tactle • dc_mcf_tdctle • dc_mcf_tlsle • CBLDCMCF('TACTLE△△') • CBLDCMCF('TDCTLE△△') • CBLDCMCF('TLSLE△△△')
同期型のメッセージを受信できるようにした。これに伴い，次の関数を追加した。 • dc_mcf_recvsync • CBLDCMCF('RECVSYNC')

追加・変更内容
次のオプションおよびオペランドを追加した。 • UAP 共通定義（mcfmuap）の-t オプションの recvtim オペランド • コネクション定義の開始（mcftalccn）の-C オプション 同期型メッセージの受信に伴う理由コードを追加した。 また，受信メッセージを保留できるようにした。 これに伴い，コネクション定義の開始（mcftalccn）に次のオペランドを追加した。 • -u オプションの msghold オペランド • -u オプションの holdlimit オペランド さらに，相手システムからメッセージを受信した場合，受信メッセージを保留して同期受信要求を待ち合わせるかどうかを指定できるようにした。 これに伴い，受信メッセージの保留判定 UOC を追加した。
コネクションの確立，解放，および状態表示を，ライブラリ関数および COBOL-UAP 作成用プログラムインタフェースでできるようにした。 これに伴い，次の関数を追加した。 • dc_mcf_tactcn • dc_mcf_tdctcn • dc_mcf_tlscn • CBLDCMCF('TACTCN△△') • CBLDCMCF('TDCTCN△△') • CBLDCMCF('TLSCN△△△')
相手アドレスを指定したコネクションの確立をできるようにした。
ネットワーク情報を表示できるようにした。これに伴い，mcftlsln コマンドに-t オプションを追加した。
サーバ型コネクションの確立要求の受付開始・終了を，ライブラリ関数および COBOL-UAP 作成用プログラムインタフェースでできるようにした。 これに伴い，次の関数を追加した。 • dc_mcf_tofln • dc_mcf_tonln • CBLDCMCF('TOFLN△△△') • CBLDCMCF('TONLN△△△')
キープアライブでコネクション障害を検出した場合に出力されるメッセージを記載した。
無通信状態の監視時間がタイムアウトした場合に出力されるメッセージを記載した。
同期型メッセージの送受信関数を発行したときに，指定した監視タイマのタイムアウトが発生した場合，コネクションの切断を抑止できるようにした。 これに伴い，コネクション定義の開始（mcftalccn）に-w オプションを追加した。
入力セグメント判定 UOC の説明に，次の内容を追加した。 • 未完成の受信メッセージが TP1/NET/TCP/IP に蓄積されている場合，受信バッファには未完成の受信メッセージと今回受信したメッセージが連結して格納されること • メッセージの全体長が不明な場合に，残りサイズに指定する情報

追加・変更内容
• 入力セグメント判定 UOC の処理の流れ
コネクションの確立要求の受付状態を，ライブラリ関数および COBOL-UAP 作成用プログラムインタフェースで表示できるようにした。 これに伴い，次の関数を追加した。 • dc_mcf_tlsln • CBLDCMCF('TLSLN△△△')
RPC の送信データ長の上限値オーバーが原因で，UAP への連絡に失敗した場合，エラーリターンするようにした。
リターン値（数値）の説明を追加した。
送受信できる一つのセグメントの最大長について記述を追加した。
入力セグメント判定 UOC インタフェースに，受信バッファサイズを通知する領域を追加した。
UOC で使用できる時間取得関数を明記した。
MCF 定義オブジェクトの解析コマンドについて説明を追加した。
MCF トレースファイルの見積もり式について説明を追加した。
トラブル発生時の調査手順を追加した。
バージョンアップ時の，定義，関数，コマンドおよびデフォルト値の変更を記載した。
バージョン 7 で変更された関数のデータ型について，変更一覧を記載した。
バージョン 5 以前のソケット関数の処理の流れを記載した。
ソケット関数の処理の流れ図に，setsockopt，getsockname を追加した。

uCosminexus TP1/Message Control 07-01，uCosminexus TP1/NET/Library 07-01，uCosminexus TP1/NET/TCP/IP 07-01

追加・変更内容
サーバ型コネクションの確立要求の受付開始・終了を，手動でできるようにした。これに伴い，コネクション定義の開始（mcftalccn）の-h オプションに listen オペランドを追加した。また，次のコマンドを追加した。 • mcftofln • mcftonln
サーバ型コネクションのアドレス割り当てに失敗した場合，MCF 通信プロセスの起動を中断しないようにした。
MCF 通信プロセス識別子を表示できるようにした。 これに伴い，コネクション定義の開始（mcftalccn）を追加した。
モデルとする定義の指定内容を流用できる，modelname オペランドを追加した。

はじめに

このマニュアルは，TP1/NET/TCP/IP の概要，機能，操作，および運用について説明したものです。

本文中に記載されている製品のうち，このマニュアルの対象製品ではない製品については，OpenTP1 Version 7 対応製品の発行時期をご確認ください。

■ 対象読者

OpenTP1 システムの通信に TCP/IP プロトコルを使用するシステム管理者，システム設計者，およびプログラマを対象としています。また，オンラインや OpenTP1 システムの基礎的な知識を持っていて，次のマニュアルを理解されていることを前提としています。

- OpenTP1 解説 （3000-3-D50）
- OpenTP1 プログラム作成の手引 （3000-3-D51）
- OpenTP1 システム定義 （3000-3-D52）
- OpenTP1 運用と操作 （3000-3-D53）
- OpenTP1 プログラム作成リファレンス C 言語編 （3000-3-D54）
- OpenTP1 プログラム作成リファレンス COBOL 言語編 （3000-3-D55）

■ 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/NET/TCP/IP を使用したシステム間の通信（AP 間通信）の概要について説明しています。

第 2 章　機能

TP1/NET/TCP/IP で使用できるコネクションに関する機能，論理端末に関する機能，およびメッセージ送受信に関する機能について説明しています。

第 3 章　C 言語のライブラリ関数

TP1/NET/TCP/IP で使用できる，C 言語のライブラリ関数について説明しています。

第 4 章　COBOL-UAP 作成用プログラムインタフェース

TP1/NET/TCP/IP で使用できる，COBOL-UAP 作成用プログラムインタフェースについて説明しています。

第５章　ユーザオウンコーディング，MCF イベントインタフェース

TP1/NET/TCP/IP のユーザオウンコーディングのインタフェース，および MCF イベントインタフェースについて説明しています。

第６章　システム定義

TCP/IP プロトコルを使用するために必要な，OpenTP1 のシステム定義の中での TP1/NET/TCP/IP 固有のシステム定義，および定義例について説明しています。

第７章　運用コマンド

TP1/NET/TCP/IP で使用できる運用コマンドについて説明しています。

第８章　組み込み方法

TP1/NET/TCP/IP を OpenTP1 システムへ組み込む方法について説明しています。

第９章　障害対策

TP1/NET/TCP/IP 運用中に発生する障害と，TP1/NET/TCP/IP の対応処理，およびメッセージの処理について説明しています。

第 10 章　トラブル発生時の調査手順

障害が発生したときに取得する情報および調査手順について説明しています。

付録 A　バージョンアップ時の変更点

各バージョンでの関数，定義およびコマンドの変更点について説明しています。

付録 B　旧製品からの移行に関する注意事項

バージョン６からバージョン７へ移行する場合，およびバージョン５以前からバージョン７へ移行する場合の注意事項について説明しています。また，バージョン５でのソケット関数の処理の流れについて説明しています。

付録 C　インタフェースの変更一覧（バージョン６以前から移行する場合）

バージョン６以前からバージョン７に移行する場合のインタフェースの変更一覧について説明しています。

付録 D　メッセージ送受信の処理の流れ

メッセージを送受信するときのデータの流れ，ジャーナル取得のタイミングについて説明しています。

付録 E　障害発生時の処理の流れ

障害が発生した場合の処理の流れについて説明しています。

付録 F　送受信メッセージの衝突時の処理の流れ（メッセージ送達確認機能使用時）

送受信メッセージが衝突した場合の処理の流れについて説明しています。

付録 G　ソケット関数の処理の流れ

ソケット関数の処理の流れについて説明しています。

付録 H　MCF 性能検証用トレースの取得

MCF 性能検証用トレースの取得について説明しています。

付録 I　ユーザアプリケーションプログラムの作成例

TP1/NET/TCP/IP のユーザアプリケーションプログラムの作成例について説明しています。

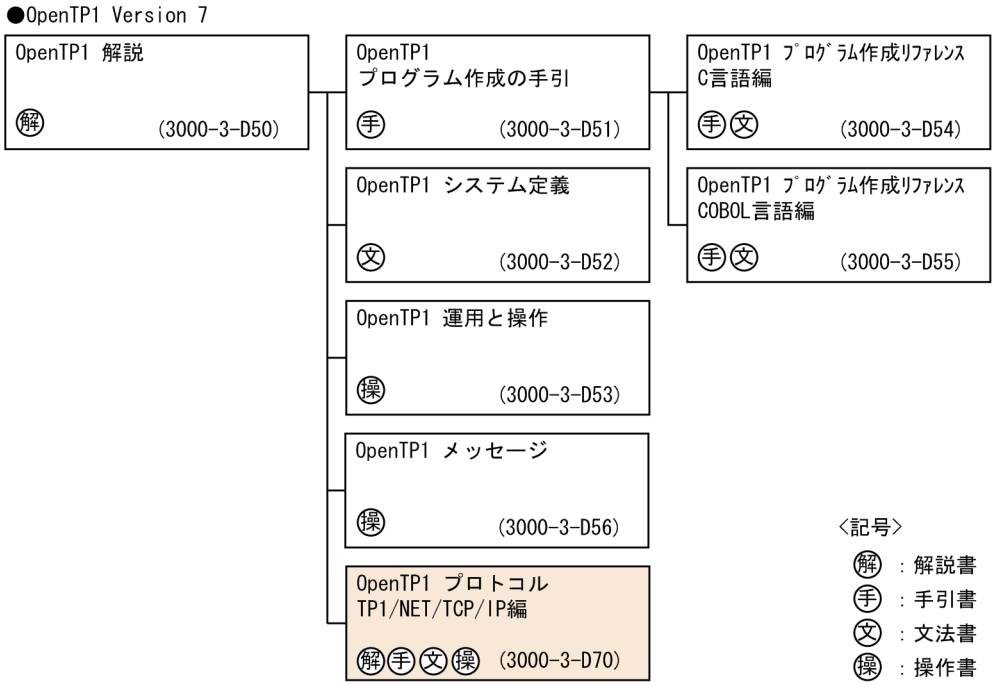
付録 J　理由コード一覧

障害通知イベントが発生した場合の理由コードについて説明しています。

付録 K　用語解説

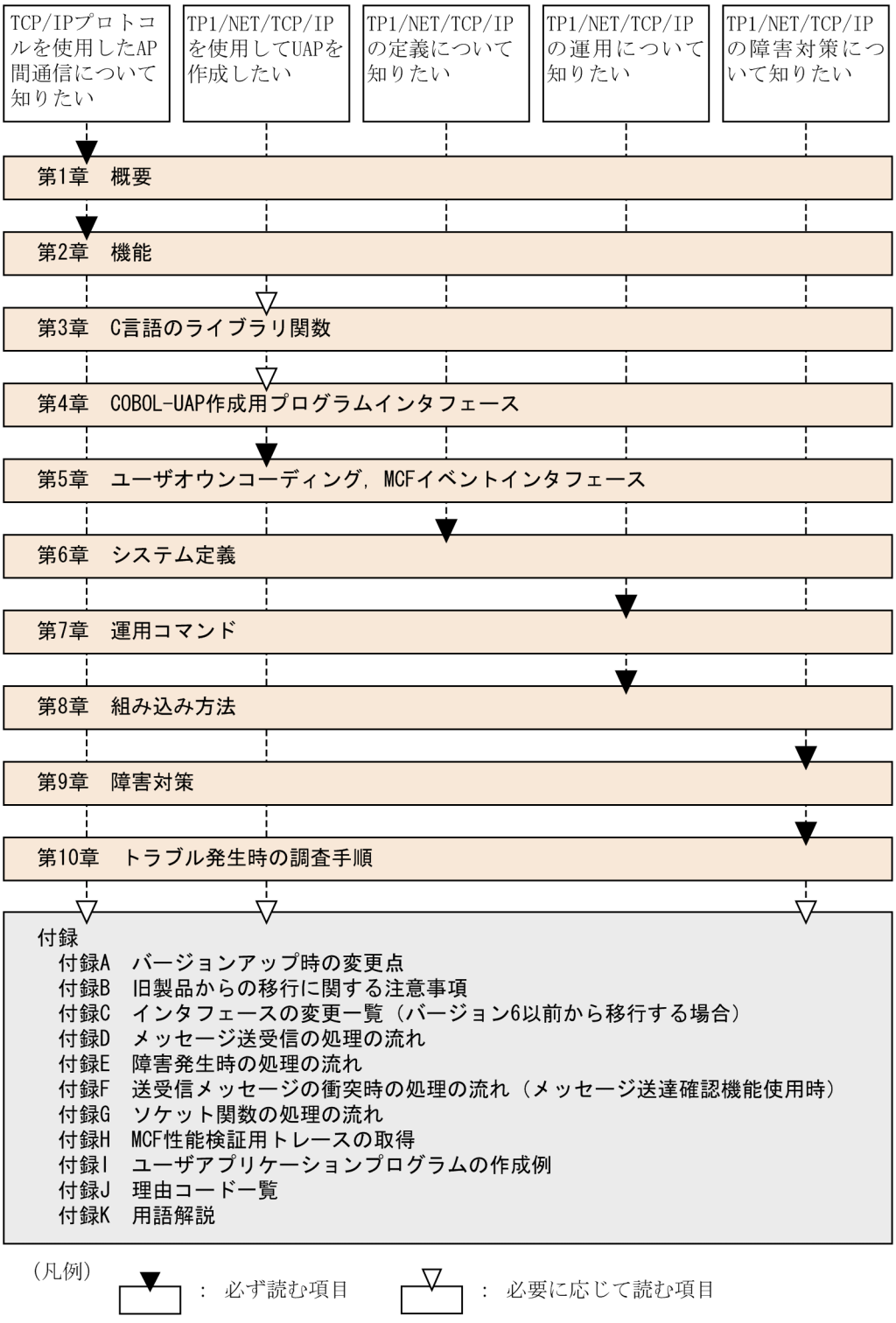
TP1/NET/TCP/IP で使用する用語について説明しています。

■ 関連マニュアル



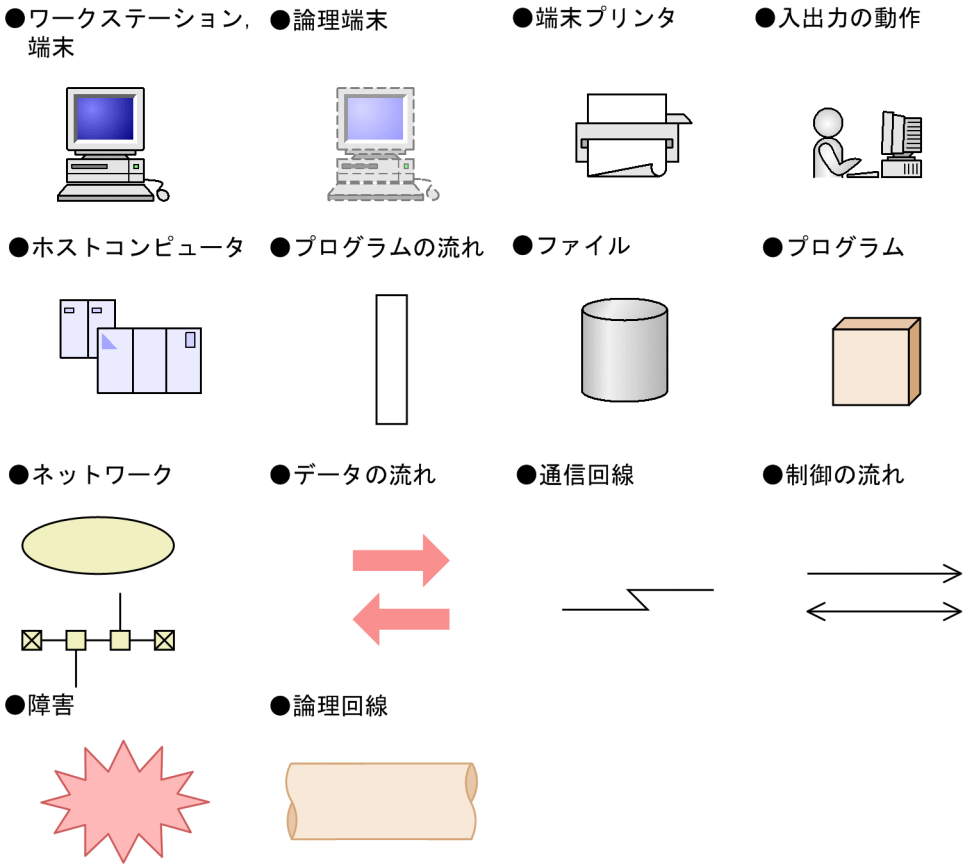
■ 読書手順

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



■ 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。



■ 文法の記号

このマニュアルで使用する各種の記号を説明します。

(1) 文法記述記号

文法の記述形式について説明する記号です。

文法記述記号	意味
〔 〕	この記号で囲まれている項目は省略できることを示します。 (例)〔-s MCF 通信プロセス識別子〕 　-s オプションとそのオペランドを指定するか，何も指定しないことを示します。
| (ストローク)	この記号で区切られた項目は選択できることを示します。 (例) -t reply | request 　-t オプションに reply または request を指定できることを示します。 ただし，C 言語のインタフェースの説明でこの記号を使用した場合は，C 言語の文法規則に従います。
{ }	この記号で囲まれている複数の項目のうちから一つを選択できることを示します。 (例) {DCMCFESI | DCMCFEMI} 　DCMCFESI と DCMCFEMI のうち，どちらかを指定できることを示します。

文法記述記号	意味
＿ （下線）	この記号で示す項目は，オペランド，オプションまたはコマンド引数を省略した場合の省略時解釈値を示します。 （例）-i auto ｜ manual -i オプションを省略した場合，manual を省略時解釈値とすることを示します。 ただし，データ操作言語の説明の場合，この下線記号で示す予約語は，必要語なので省略できないことを示します。下線がない予約語は，補助語なので書いても書かなくてもかまいません。
…	この記号で示す直前の一つの項目を繰り返し指定できることを示します。 ただし，項目が括弧で囲まれている場合，括弧全体が一つの項目となります。
△ （白三角）	空白を示します。 （例）コネクション ID1△コネクション ID2 コネクション ID1 とコネクション ID2 の間に，空白を 1 個入力することを示します。

(2) 属性表示記号

ユーザ指定値の範囲などを説明する記号です。

属性表示記号	意味
～	この記号のあとにユーザ指定値の属性を示します。
《　》	ユーザが指定を省略したときの省略時解釈値を示します。
＜　＞	ユーザ指定値の構文要素を示します。
((　))	ユーザ指定値の指定範囲を示します。

(3) 構文要素記号

ユーザ指定値の内容を説明する記号です。

構文要素記号	意味
＜英字＞	アルファベット（A～Z，a～z）と_（アンダスコア）
＜英字記号＞	アルファベット（A～Z，a～z）と#，@，¥
＜英数字＞	英字と数字（0～9）
＜英数字記号＞	英字記号と数字（0～9）
＜符号なし整数＞	数字列（0～9）
＜ 10 進数字＞	数字（0～9）
＜ 16 進数字＞	数字（0～9）とアルファベット（A～F，a～f）
＜識別子＞	先頭がアルファベットの英数字列

構文要素記号	意味
＜記号名称＞	先頭が英字記号の英数字記号列
＜文字列＞	任意の文字の配列
＜パス名＞	記号名称，／，または.（ピリオド） （ただし，パス名は使用する OS に依存）
＜ホスト名＞	先頭が英数字または-（ハイフン）で， 先頭以外が英数字，-（ハイフン），または.（ピリオド）

■ このマニュアルでの表記

（1）製品名

このマニュアルで使用する製品名称の略称を次に示します。

製品名称	略称	
AIX V6.1	AIX	
AIX V7.1		
AIX V7.2		
HP-UX 11i V2 (IPF)	HP-UX (IPF) (32 ビット用)， HP-UX (IPF) (64 ビット用)	HP-UX
HP-UX 11i V3 (IPF)		
Itanium Processor Family	IPF	
Windows 7 Enterprise	Windows 7	Windows 7
Windows 7 Professional		
Windows 7 Ultimate		
Windows 7 Enterprise (x64)	Windows 7 x64 Edition	
Windows 7 Professional (x64)		
Windows 7 Ultimate (x64)		
Windows 8 Enterprise	Windows 8	Windows 8
Windows 8 Pro		
Windows 8 Enterprise(x64)	Windows 8 x64 Edition	
Windows 8 Pro(x64)		
Windows 8.1 Enterprise	Windows 8.1	Windows 8.1

製品名称	略称	
Windows 8.1 Pro	Windows 8.1	Windows 8.1
Windows 8.1 Enterprise(x64)	Windows 8.1 x64 Edition	
Windows 8.1 Pro(x64)		
Windows 10 Enterprise	Windows 10	Windows 10
Windows 10 Pro		
Windows 10 Enterprise(x64)	Windows 10 x64 Edition	
Windows 10 Pro(x64)		
Windows Server 2008 R2, Datacenter Edition	Windows Server 2008 R2	
Windows Server 2008 R2, Enterprise Edition		
Windows Server 2008 R2, Standard Edition		
Windows Server 2012 Datacenter	Windows Server 2012	
Windows Server 2012 Standard		
Windows Server 2012 R2 Datacenter	Windows Server 2012 R2	
Windows Server 2012 R2 Standard		
Windows Server 2016 Datacenter	Windows Server 2016	
Windows Server 2016 Standard		
Windows Server 2019 Datacenter	Windows Server 2019	
Windows Server 2019 Standard		
uCosminexus TP1/Message Control	TP1/Message Control	
uCosminexus TP1/Message Control(64)		
uCosminexus TP1/NET/High Availability	TP1/NET/High Availability	
uCosminexus TP1/NET/High Availability(64)		
uCosminexus TP1/NET/Library	TP1/NET/Library	
uCosminexus TP1/NET/Library(64)		
uCosminexus TP1/NET/NCSB	TP1/NET/NCSB	
uCosminexus TP1/NET/OSI-TP	TP1/NET/OSI-TP	
uCosminexus TP1/Server Base	TP1/Server Base	
uCosminexus TP1/Server Base(64)		
uCosminexus TP1/NET/TCP/IP	TP1/NET/TCP/IP	

製品名称	略称	
uCosminexus TP1/NET/TCP/IP(64)	TP1/NET/TCP/IP	
uCosminexus TP1/NET/X25-Extended	TP1/NET/X25-Extended	
Linux	Linux	Linux
Red Hat Enterprise Linux Server 6 (32-bit x86)	Linux（x86，AMD/Intel 64）（32 ビット用）， Linux（AMD/Intel 64）（64 ビット用）	
Red Hat Enterprise Linux Server 6 (64-bit x86_64)		
Red Hat Enterprise Linux Server 7 (64-bit x86_64)		
Red Hat Enterprise Linux Server 8 (64-bit x86_64)		
Solaris 8	Solaris	
Solaris 9		
Solaris 10		

- Windows Server 2008 R2，Windows 7，Windows 8，Windows 8.1，Windows 10，Windows Server 2012，Windows Server 2012 R2 および Windows Server 2016 で機能差がない場合，Windows と表記しています。
- AIX，HP-UX，Linux，および Solaris を総称して UNIX と表記しています。

(2) 適用 OS による違いについて

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

項目	マニュアルの表記	読み換え
環境変数の表記	$aaaaaa （例） $DCDIR	%aaaaaa% （例） %DCDIR%
パス名の区切り文字	:	;
ディレクトリの区切り文字	/	¥
完全パス名	ルートディレクトリから指定します。 （例） /tmp	先頭にドライブ文字を付加して，ルートディレクトリから指定します。 （例） C:¥tmp
実行形式ファイル名	ファイル名だけを指定します。 （例） mcfmngrd	ファイル名に拡張子を付加して指定します。 （例） mcfmngrd.exe
make コマンド	make	nmake

(3) アーキテクチャによる違いについて

このマニュアルでは，32 ビットアーキテクチャ対応 OS と 64 ビットアーキテクチャ対応 OS で記述を書き分けている個所があります。ご使用の OS をご確認の上，アーキテクチャに応じた記載個所をお読みください。

次の表に，このマニュアルでのアーキテクチャの違いによる表記と対応 OS を示します。

マニュアルの表記	OS
32 ビットアーキテクチャの場合	• AIX（32 ビット用） • HP-UX （IPF）（32 ビット用） • Solaris • Windows（32 ビット用） • Linux （x86，AMD/Intel 64）（32 ビット用）
64 ビットアーキテクチャの場合	• AIX （64 ビット用） • HP-UX （IPF）（64 ビット用） • Windows（64 ビット用） • Linux （AMD/Intel 64）（64 ビット用）

(4) JIS コード配列のキーボードと ASCII コード配列のキーボードとの違いについて

JIS コード配列と ASCII コード配列では，次に示すコードで入力文字の違いがあります。このマニュアルの文字入力例（コーディング例）の表記は，JIS コード配列（日本語のキーボード）に従った文字に統一しています。

コード	JIS コード配列	ASCII コード配列
(5c)16	'¥'（円記号）	'＼'（バックスラッシュ）
(7e)16	'‾'（オーバライン）	'~'（チルド）

■ 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
ACK	Acknowledgement Flag
AP	Application Program
API	Application Programming Interface
DNS	Domain Name System
FIN	Fin Flag

英略語	英字での表記
LAN	Local Area Network
MCF	Message Control Facility
MHP	Message Handling Program
OS	Operating System
PSH	Push Flag
RST	Reset Flag
SPP	Service Providing Program
SYN	Synchronize Flag
TCP/IP	Transmission Control Protocol/Internet Protocol
UAP	User Application Program
UOC	User Own Coding

■ 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。

■ 謝辞

COBOL 言語仕様は，CODASYL（the Conference on Data Systems Languages：データシステムズ言語協議会）によって，開発された。OpenTP1 のユーザアプリケーションプログラムのインタフェース仕様のうち，データ操作言語（DML　Data Manipulation Language）の仕様は，CODASYL　COBOL（1981）の通信節，RECEIVE 文，SEND 文，COMMIT 文，及び ROLLBACK 文を参考にし，それに日立製作所独自の解釈と仕様を追加して開発した。原開発者に対し謝意を表すとともに，CODASYL の要求に従って以下の謝辞を掲げる。なお，この文章は，COBOL の原仕様書「CODASYL COBOL JOURNAL OF DEVELOPMENT 1984」の謝辞の一部を再掲するものである。

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

COBOL は産業界の言語であり，特定の団体や組織の所有物ではない。

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

次に示す著作権表示付資料の著作者及び著作権者

FLOW-MATIC（Sperry Rand Corporation の商標），

Programming for the Univac（R）I and II，Data Automation Systems，

Sperry Rand Corporation　著作権表示 1958 年，1959 年；

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

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

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

目次

1 概要

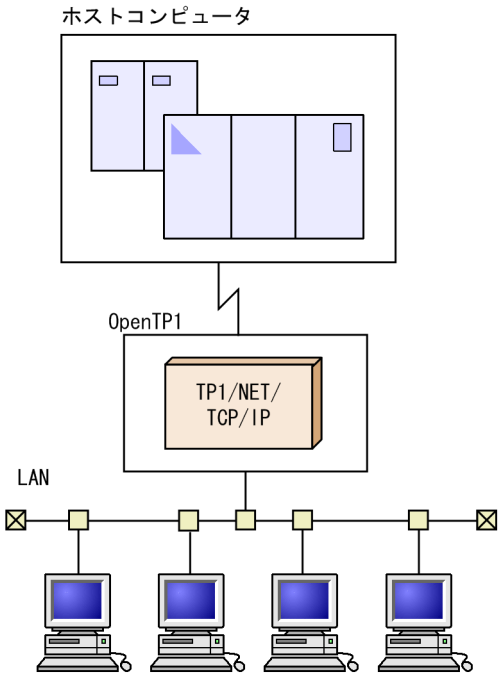
TP1/NET/TCP/IP は，OpenTP1 システムを構成するプログラムの一つです。ホストコンピュータ，端末などを TCP/IP プロトコルによって論理的に接続し，メッセージを送受信します。この章では，TP1/NET/TCP/IP を使用したシステム間の通信（AP 間通信）の概要について説明します。

1.1　AP 間通信の概要

AP 間通信とは，異なるシステムにあるアプリケーションプログラム間でのメッセージ送受信のことです。TP1/NET/TCP/IP は，OS が提供する TCP/IP のソケットインタフェースを利用して AP 間通信をするプログラムです。TP1/NET/TCP/IP を使用した AP 間通信では，相手システムで発生したトランザクションを自システムで処理したり，その結果を送信したりできます。

TP1/NET/TCP/IP を使用したネットワーク構成の例を次の図に示します。

図 1-1　TP1/NET/TCP/IP を使用したネットワーク構成の例



1.2　AP 間通信の形態

ここでは，コネクションと論理端末の関係，およびメッセージ送受信の形態について説明します。

1.2.1　コネクションと論理端末の関係

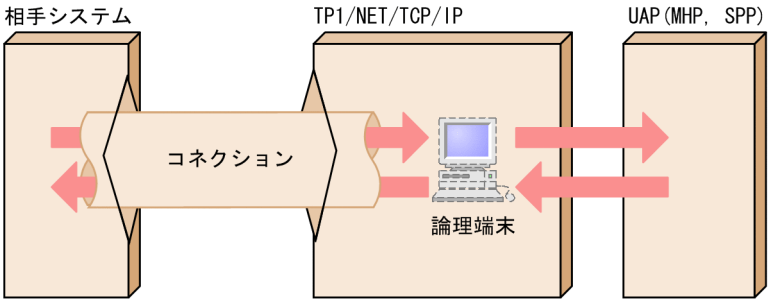
TP1/NET/TCP/IP は，AP 間通信をするために，自システムと相手システムとの間に論理的な通信路（**コネクション**）を確立します。コネクションの確立方法には，クライアント型とサーバ型の二つがあります。

一方，**論理端末**は，TP1/NET/TCP/IP と UAP との通信接点に当たります。TP1/NET/TCP/IP は，論理端末を通して，自システムの UAP とメッセージを送受信します。

コネクションと論理端末の指定を対応させると，自システムと相手システムとの論理的な通信路が確立でき，AP 間通信ができるようになります。コネクションと論理端末は，システム定義で対応させます。なお，コネクションと論理端末の数は，1 対 1 になります。

コネクションと論理端末の関係について，次の図に示します。

図 1-2　コネクションと論理端末の関係



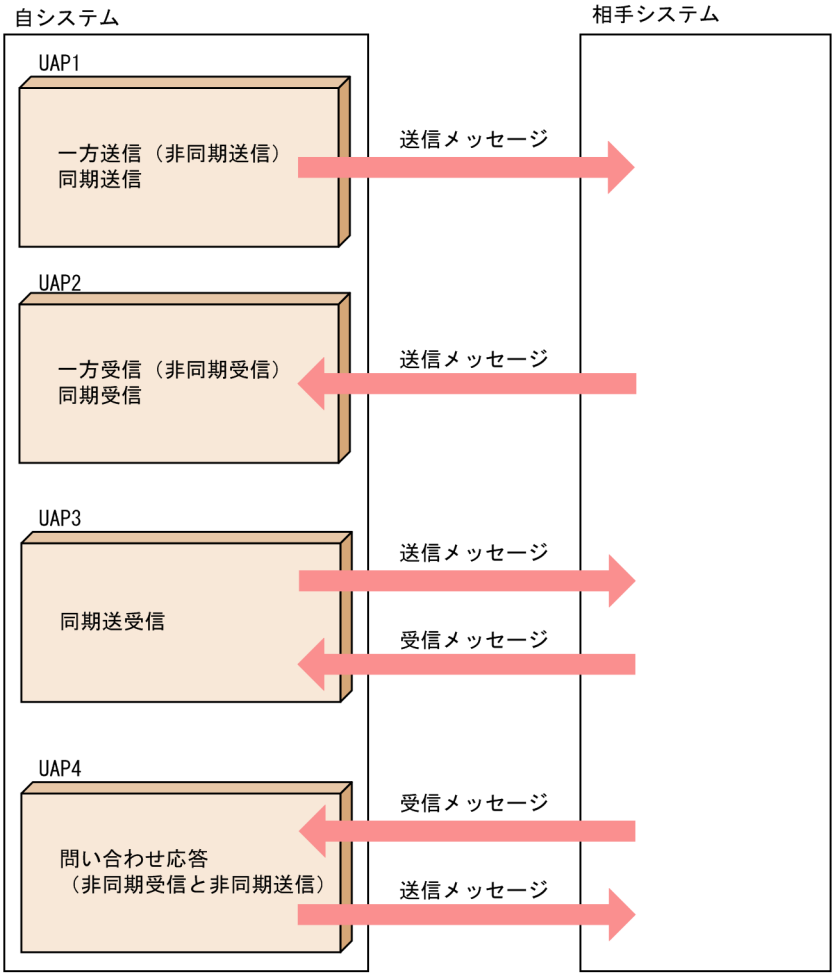
1.2.2　メッセージ送受信の形態

TP1/NET/TCP/IP を使用したメッセージ送受信の形態を次に示します。

- 一方送信（非同期送信）
- 一方受信（非同期受信）
- 同期送信
- 同期受信
- 同期送受信
- 問い合わせ応答（非同期受信と非同期送信）

TP1/NET/TCP/IP を使用したメッセージ送受信の例を次の図に示します。

図 1-3　TP1/NET/TCP/IP を使用したメッセージ送受信の例



相手システムとのメッセージ送受信に使用する UAP インタフェースを，**メッセージ送受信関数**と呼びます。

(1)　一方送信（非同期送信）

自システムから，一方送信メッセージを相手システムに送信する形態です。トランザクションと連動して相手システムにメッセージを送信する場合に使用します。

一つのトランザクションで複数のメッセージを一方送信した場合でも，トランザクションが決着するまで相手システムへメッセージは送信されません。

(2)　一方受信（非同期受信）

相手システムからの一方送信メッセージを，自システムで受信する形態です。トランザクションと連動してメッセージを受信する場合に使用します。

(3) 同期送信

相手システムに対して，メッセージを送信する形態です。トランザクションと連動しないで相手システムにメッセージを送信する場合，つまりメッセージの送信要求と相手システムへのメッセージ送信処理を同期させる場合に使用します。

一つのトランザクションで複数のメッセージを同期送信した場合は，メッセージの送信要求のたびに相手システムへメッセージが送信されます。

相手システムからの応答は要求しません。メッセージ送信処理が完了すると，送信を要求した UAP にリターンします。

(4) 同期受信

相手システムから，メッセージを受信する形態です。トランザクションと連動しないで相手システムからのメッセージを受信する場合に使用します。

相手システムからのメッセージの受信を待ちます。メッセージを受信すると，受信を要求した UAP にリターンします。

(5) 同期送受信

自システムからメッセージを送信し，相手システムからの応答を受信する形態です。トランザクションと連動しないでメッセージの送受信をする場合に使用します。

相手システムからのメッセージを受信したときに，同期送受信を要求した UAP にリターンします。

(6) 問い合わせ応答（非同期受信と非同期送信）

相手システムからの問い合わせメッセージを，自システムで受信したあと，自システムから応答メッセージを相手システムに送信する形態です。トランザクションと連動してメッセージを送受信する場合に使用します。

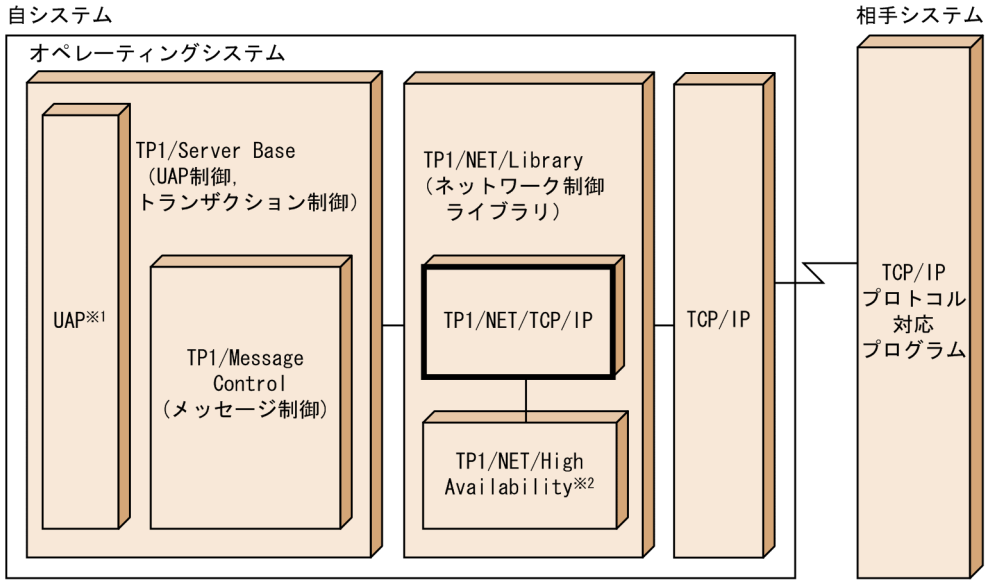
一つのトランザクションで複数のメッセージを応答送信した場合でも，トランザクションが決着するまで相手システムへメッセージは送信されません。

1.3　ソフトウェア構成の例

TP1/NET/TCP/IP は，OpenTP1 システムに組み込まれて動作するプログラムです。OpenTP1 のメッセージ送受信機能（TP1/Message Control，TP1/NET/Library）と連携して，メッセージ制御機能（MCF）を実現します。

TP1/NET/TCP/IP を組み込んだソフトウェア構成の例を次の図に示します。

図 1-4　TP1/NET/TCP/IP を組み込んだソフトウェア構成の例



注※1

TP1/NET/TCP/IP で扱う UAP は，MHP および SPP です。UAP については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

注※2

コネクションの切り替えをする場合に必要となるプログラムです。コネクションの切り替えについては，「2.1.12　コネクションの切り替え（クライアント型コネクション）」を参照してください。

2 機能

この章では，TP1/NET/TCP/IP で使用できるコネクションに関する機能，論理端末に関する機能，およびメッセージ送受信に関する機能について説明します。

2.1　コネクションに関する機能

ここでは，コネクションに関する機能について説明します。

2.1.1　TP1/NET/TCP/IP が使用する TCP/IP 資源

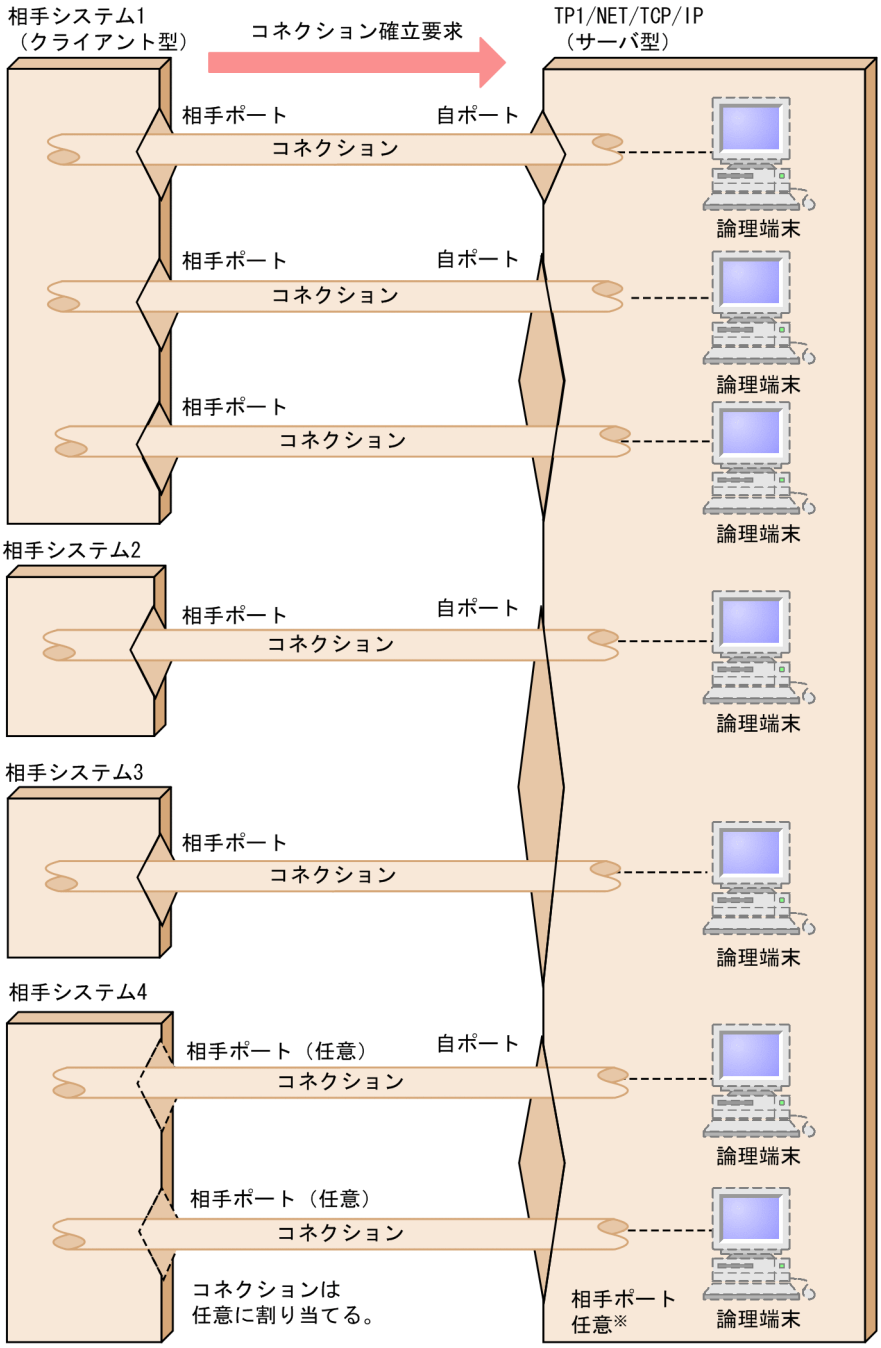
(1)　コネクションとポートの関係

ポートとは，TCP/IP プロトコルを使用してメッセージ送受信をするときに，相手システムと自システムとのサービスの窓口となるものです。ポート番号はシステム内で一意な値にする必要があるため，TP1/NET/TCP/IP が使用するポート番号は重複しないように注意してください。また，ほかのアプリケーションとの重複を避けるため，OS が任意に割り当てるポート番号（動的ポートまたは短命ポートと呼ばれるポート番号）を使用しないでください。

サーバ型コネクションの場合は，一つのポートに一つ以上のコネクションを持つことができます。

コネクションとポートの関係を，サーバ型，およびクライアント型の場合についてそれぞれ次の図に示します。なお，サーバ型とクライアント型のコネクションを混在させて確立することもできます。

図 2-1　コネクションとポートの関係（自システムがサーバ型）



注※

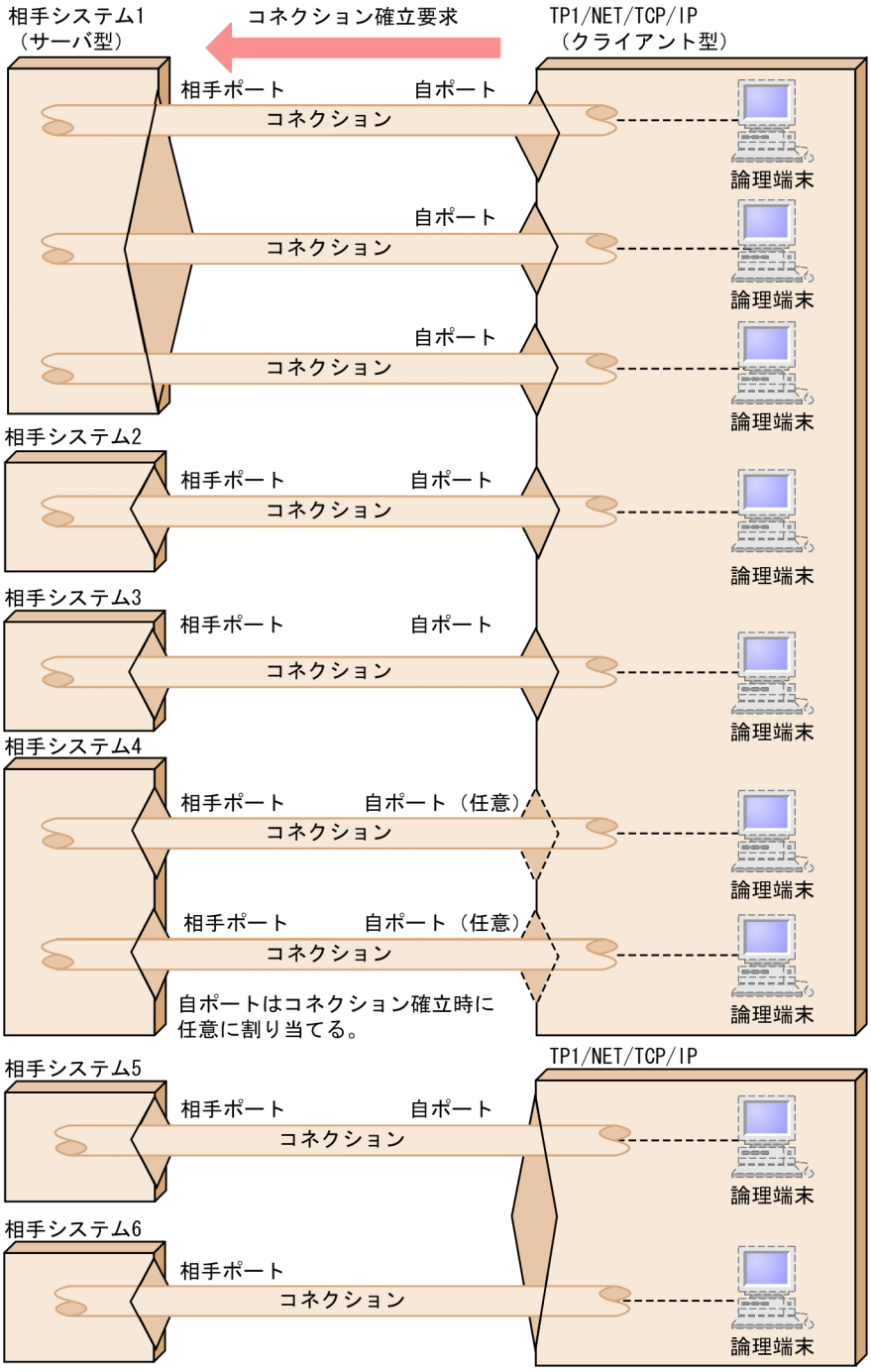
相手ポートが任意（コネクション定義（mcftalccn -o）の oprtno オペランドに free を指定）のサーバ型コネクションが複数存在する場合，相手システムから受け付けたコネクション確立要求をどのサーバ型コネクションに割り当てるかは任意となります。したがって，現在 TP1/NET/TCP/IP から割り当てられたサーバ型コネクションが，次回の確立要求時にも割り当てられるとは限りません。

このため，サーバ型コネクションの解放時に未送信メッセージが残っていた場合，意図しない相手システムのプログラムに未送信メッセージが送信されることがあります。

ユーザが未送信メッセージを破棄するか，システム間でメッセージの整合性をチェックするなどの処置をしてください。

問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用する場合，継続問い合わせ応答中の論理端末に対応するコネクションを割り当て対象とするかどうかをコネクション定義（mcftalccn -l）の cnassign オペランドで指定できます。ただし，継続問い合わせ応答中の論理端末に対応するコネクションを割り当て対象とする指定（all）をしても，継続問い合わせ応答が終了するまでは，該当するコネクションで確立できません。

図 2-2　コネクションとポートの関係（自システムがクライアント型）



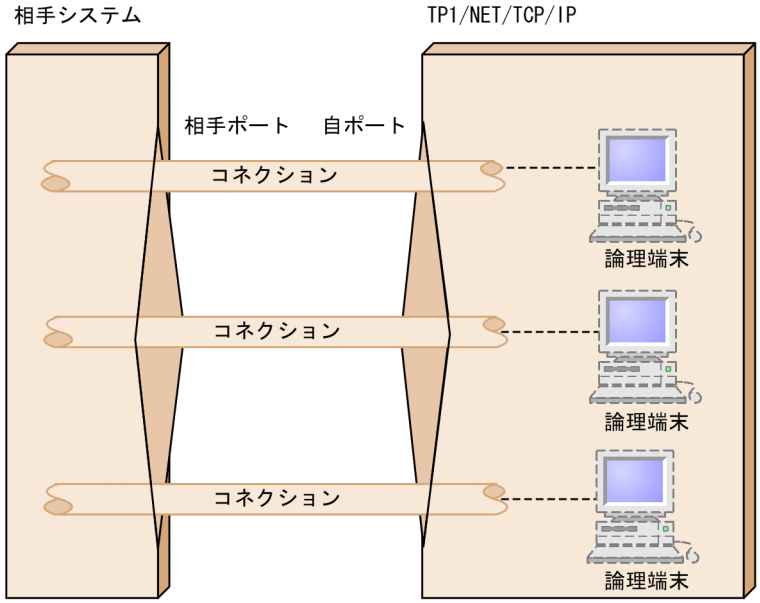
相手ポートは，相手システムごとにユニークな番号でも，すべて同じ番号でも問題ありません。

ポート番号の指定誤りなどによって次に示す状態になった場合は，TP1/NET/TCP/IP の起動やコネクション確立ができません。

1. 同一の相手ポートと自ポートの中で複数のコネクションを確立しようとしている。
2. 自システム内でプロセス間にわたって同一の自ポート番号を使用している。

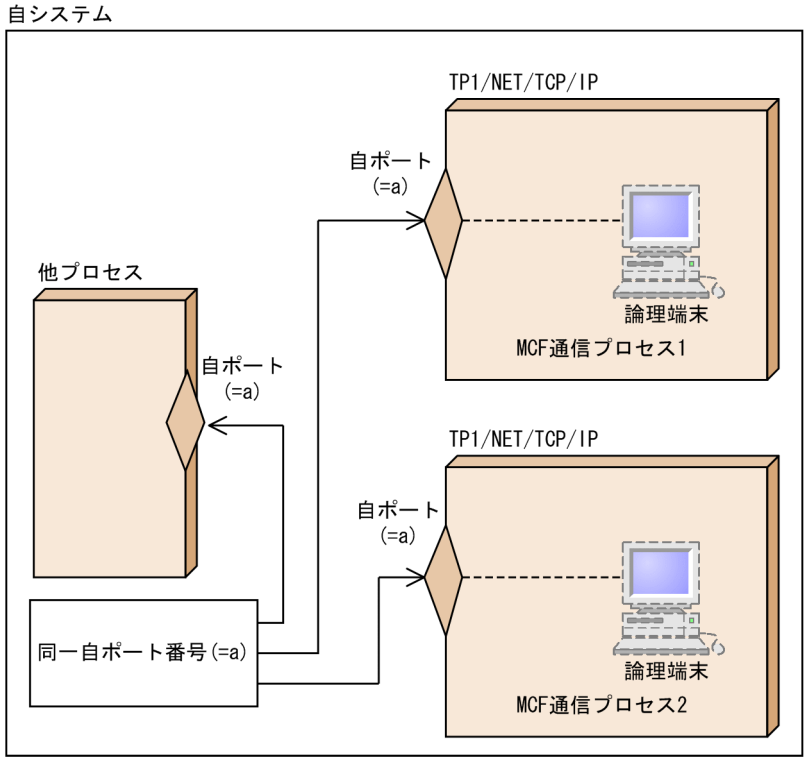
それぞれのパターンについて，次の図に示します。

図 2-3　コネクションとポートの不正なパターン例 1



同一の相手ポートと自ポートの中で複数のコネクションは持てません。

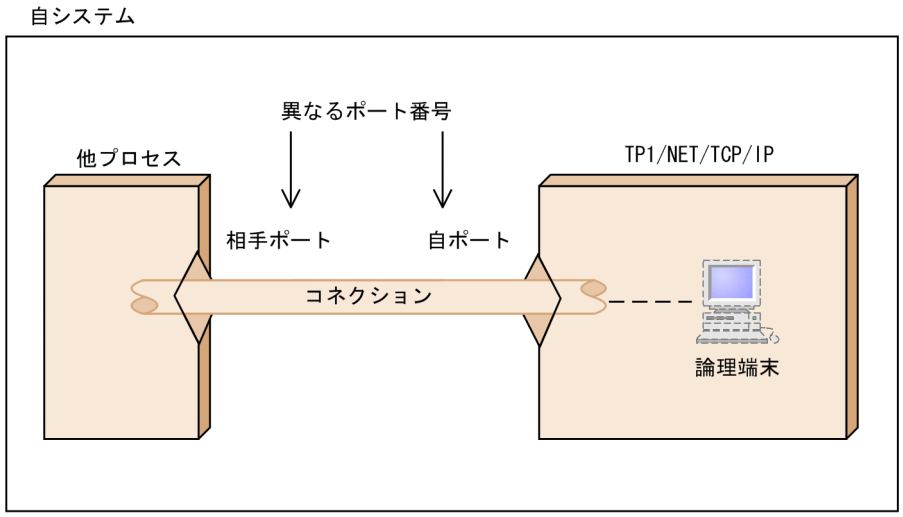
図 2-4　コネクションとポートの不正なパターン例 2



自システム内でプロセス間にわたって同一の自ポート番号は使用できません。

同一ホスト内の他のプロセス（TP1/NET/TCP/IP の MCF 通信プロセスを含みます）とコネクションを確立することもできます。この場合，自ポートと相手ポートは異なるポート番号を使用してください。

図 2-5　同一ホスト内のコネクションとポートの関係



(2)　TP1/NET/TCP/IP で使用しているポート番号

ポート番号を指定できるシステム定義のオペランドを，次の表に示します。

表 2-1　ポート番号を指定できるシステム定義のオペランド

定義名	定義	指定できる範囲	デフォルト値	対象システムサービス
MCF 通信構成定義	mcftalccn -r portno	1024～65535	OS の自動割り当てポートの範囲※	MCF 通信サービス

注※

サーバ型コネクションの場合，省略できません。

2.1.2　使用できるコネクション数

一つの MCF 通信プロセス当たりで使用できるコネクション数は 1～2048 です。

2048 より多くのコネクションを使用する場合は，MCF 通信プロセスを分割してください。

注意事項

- 使用するコネクション数に応じて各 MCF 通信プロセスで使用するファイル記述子数やアクセスするファイル数を見積もって，OpenTP1 のシステムサービス共通情報定義を指定する必要があります。詳細については，「6.　システム定義」の「システムサービス共通情報定義」を参照してください。

  また，使用するコネクション数を変更する場合は，メモリ所要量およびディスク占有量の再見積もりが必要になります。使用するコネクション数に応じた資源の見積もりと設定については，マニュアル「OpenTP1 システム定義」，および TP1/Server Base，TP1/Message Control，TP1/NET/TCP/IP のリリースノートを参照してください。資源不足が発生すると，コネクションの確立に失敗したり，OpenTP1 システムがダウンしたりする場合があります。
- 接続するコネクション数が増加すると，コネクションの検索や select システムコールによる事象監視のオーバヘッドも増加します。使用するコネクション数は，システムの性能を十分に検討した上で決定してください。
- ディスクキューを使用する場合，システム全体で使用できる論理端末数の上限に制約があります。これに伴ってシステム全体で使用できるコネクション数の上限も制限されます。詳細については，マニュアル「OpenTP1 システム定義」の拡張予約定義（mcfmexp）を参照してください。

2.1.3　クライアント型コネクションの確立

クライアント型コネクションは，自システムからコネクションの確立を要求する方法です。

クライアント型コネクションの場合，コネクション定義（mcftalccn）に相手システムのアドレス（ホスト名称），およびポート番号を指定する必要があります。また，必要に応じて自システムのアドレス（ホスト名称），およびポート番号を指定することもできます。

クライアント型コネクションと MCF 通信構成定義との関係については，「6.　システム定義」の「コネクションの形態と MCF 通信構成定義との関係」を参照してください。

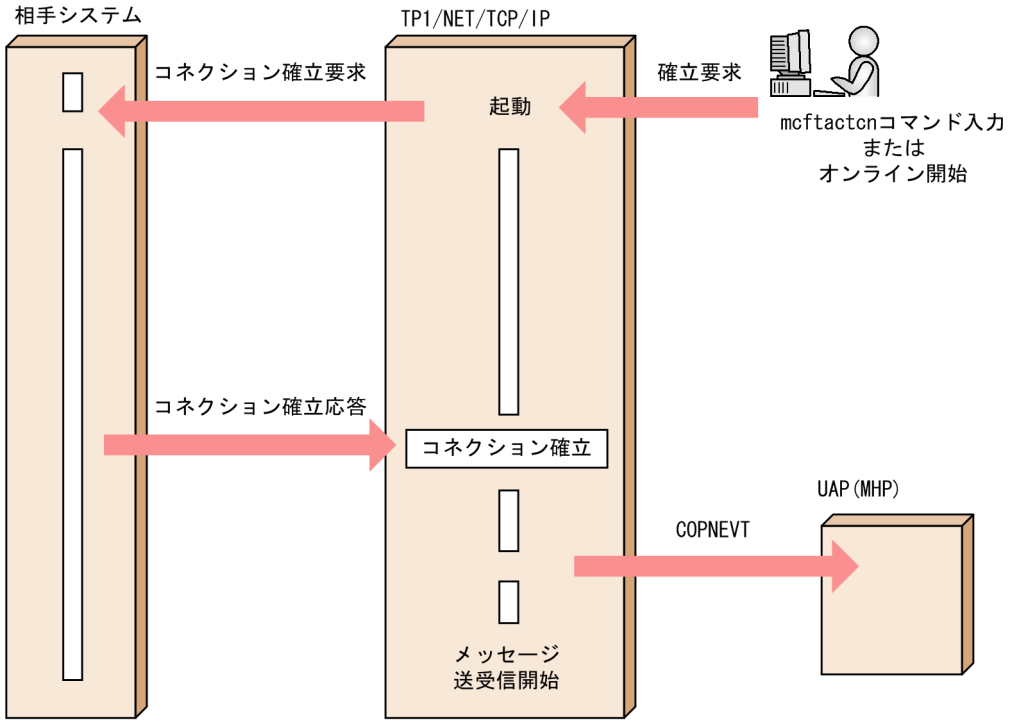
運用コマンド（mcftactcn）の入力，または API（dc_mcf_tactcn 関数もしくは CBLDCMCF('TACTCN△△')）の発行によって，TP1/NET/TCP/IP は相手システムにコネクションの確立を要求します。相手システムからの確立応答を受信した時点でコネクションが確立します。TP1/NET/TCP/IP は状態通知イベント（COPNEVT）によってコネクションの確立を通知します。

また，相手システムからの確立応答を長時間待ち合わせないように，コネクション確立時の監視時間をコネクション定義（mcftalccn -b）の concmptim オペランドで指定できます。

なお，コネクション定義（mcftalccn -i）に auto を指定することで，オンライン開始時に自動的にコネクションを確立することもできます。

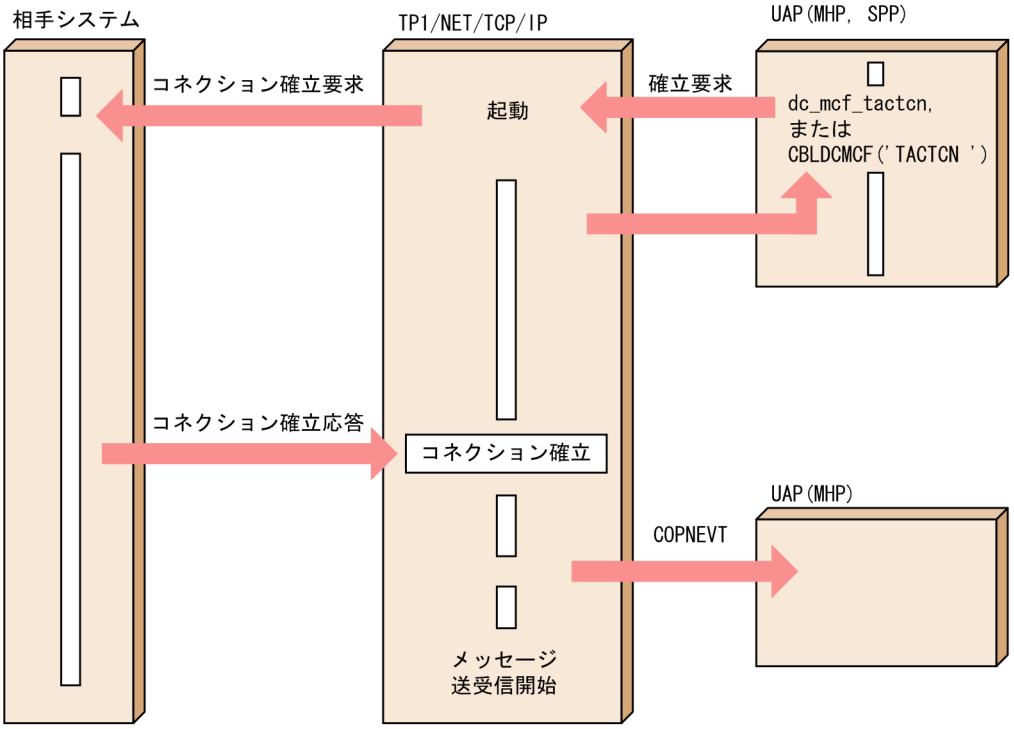
運用コマンド（mcftactcn）入力時またはオンライン開始時の，クライアント型コネクションの確立方法を次の図に示します。

図 2-6　クライアント型コネクションの確立（運用コマンド入力時またはオンライン開始時）



API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）発行時のクライアント型コネクションの確立方法を次の図に示します。

図 2-7　クライアント型コネクションの確立（API 発行時）



2.1.4　相手アドレスを指定したコネクションの確立（クライアント型コネクション）

クライアント型コネクションの確立の場合，相手アドレスを指定してコネクションを確立できます。

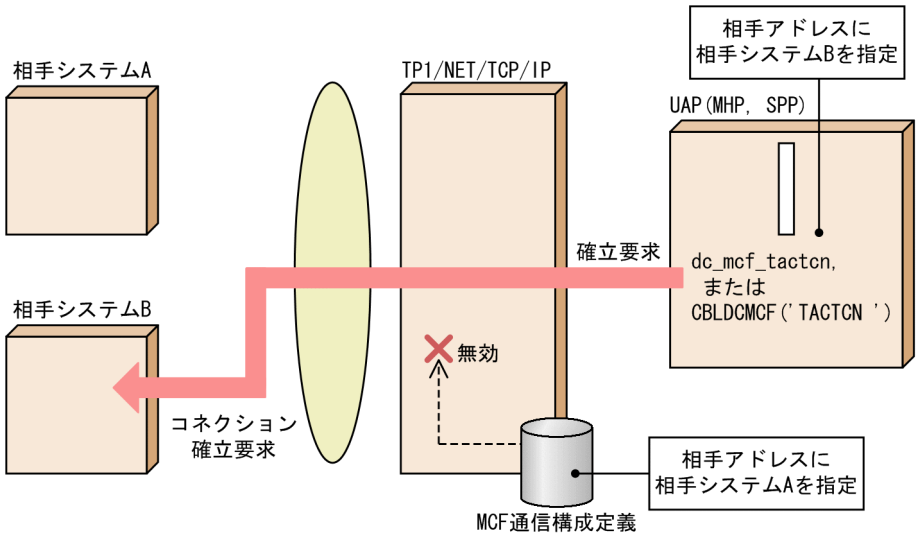
相手アドレスを指定したコネクション確立は，次に示すような場合に有効です。

- 接続相手の数が一つの MCF 通信プロセスで定義できる最大数を超える場合
- OpenTP1 がオンライン状態で接続相手を追加する場合

相手アドレスの指定は，dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')で行います。指定できるのは，自システムの IP アドレス，ポート番号，相手システムの IP アドレス，およびポート番号です。このとき，コネクション定義（mcftalccn）に指定したこれらの項目は無効となります。

相手アドレスを指定したコネクションの確立を次の図に示します。

図 2-8　相手アドレスを指定したコネクションの確立



2.1.5　サーバ型コネクションの確立

サーバ型コネクションは，相手システムからのコネクションの確立要求を受け入れる方法です。

サーバ型コネクションの場合，コネクション定義（mcftalccn）に自システムのポート番号，相手システムのアドレス（ホスト名称），およびポート番号を指定する必要があります。また，必要に応じて自システムのアドレス（ホスト名称）を指定することもできます。

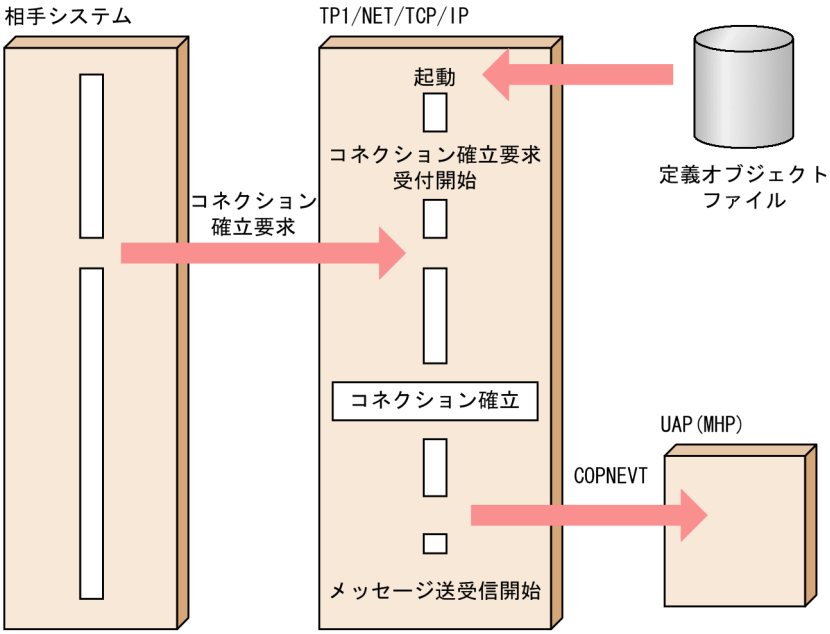
サーバ型コネクションと MCF 通信構成定義との関係については，「6.　システム定義」の「コネクションの形態と MCF 通信構成定義との関係」を参照してください。

確立要求を受け付けた相手システムのアドレスおよびポート番号がコネクション定義（mcftalccn）に定義されていない場合，コネクションを解放します。相手システムの任意のポート番号からの確立要求を受け入れる場合は，ポートフリーを指定してください。また，任意の相手システムからの確立要求を受け入れる場合は，相手アドレスのチェックを抑止してください。詳細は，「2.1.1(1)　コネクションとポートの関係」，「2.1.7　相手アドレスチェックの抑止（サーバ型コネクション）」，および「6.　システム定義」を参照してください。

TP1/NET/TCP/IP は，定義オブジェクトファイルの内容によって起動し，確立要求の受付を開始して，相手システムからのコネクション確立要求を待ちます。確立要求を受けると，コネクションが確立します。コネクションが確立すると，TP1/NET/TCP/IP は状態通知イベント（COPNEVT）によってコネクションの確立を通知します。

サーバ型コネクションの確立方法を，次の図に示します。

図 2-9　サーバ型コネクションの確立



2.1.6　コネクションの確立要求の受付開始と終了（サーバ型コネクション）

サーバ型コネクションの場合，コネクション定義（mcftalccn -h）の listen オペランドに manual を指定すると，OpenTP1 のオンライン開始時のコネクション確立要求の受付開始を手動にできます。

手動にした場合は，サーバ側の準備が完了し，相手システムからのコネクション確立要求を受け付け可能となった時点で，運用コマンド（mcftonln）を入力するか，または API（dc_mcf_tonln 関数もしくは CBLDCMCF('TONLN△△△')）を発行し，コネクションの確立要求の受け付けを開始します。これによって，相手システムからのデータ送受信を待ち合わせたい場合に，サービスグループをスケジュール閉塞したり，コネクション確立 UOC でコネクションの確立を拒否したりする必要がなくなります。

コネクション確立要求の受付を終了する場合は，運用コマンド（mcftofln）を入力するか，または API（dc_mcf_tofln 関数もしくは CBLDCMCF('TOFLN△△△')）を発行します。

一方，自動的にコネクション確立要求を受け付ける場合は，コネクション定義（mcftalccn -h）の listen オペランドで auto を指定してください。

なお，運用コマンド（mcftlsln）の入力，または API（dc_mcf_tlsln 関数もしくは CBLDCMCF('TLSLN△△△')）の発行によって，確立要求の受付状態を表示することもできます。

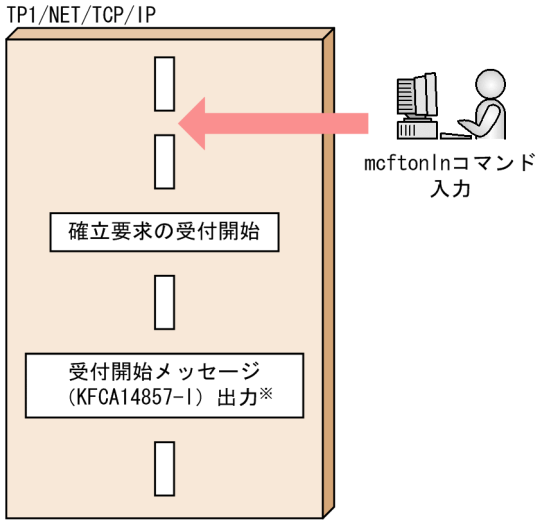
コネクション確立要求の受付開始・終了について，それぞれ説明します。

(1)　コネクション確立要求の受付開始

運用コマンド（mcftonln）の入力，または API（dc_mcf_tonln 関数もしくは CBLDCMCF('TONLN△△△')）の発行によって，TP1/NET/TCP/IP は相手システムからのコネクション確立要求の受付を開始します。

コネクション確立要求の受付開始について，運用コマンド（mcftonln）を入力する場合を例に次の図に示します。

図 2-10　コネクション確立要求の受付開始（運用コマンド入力時）

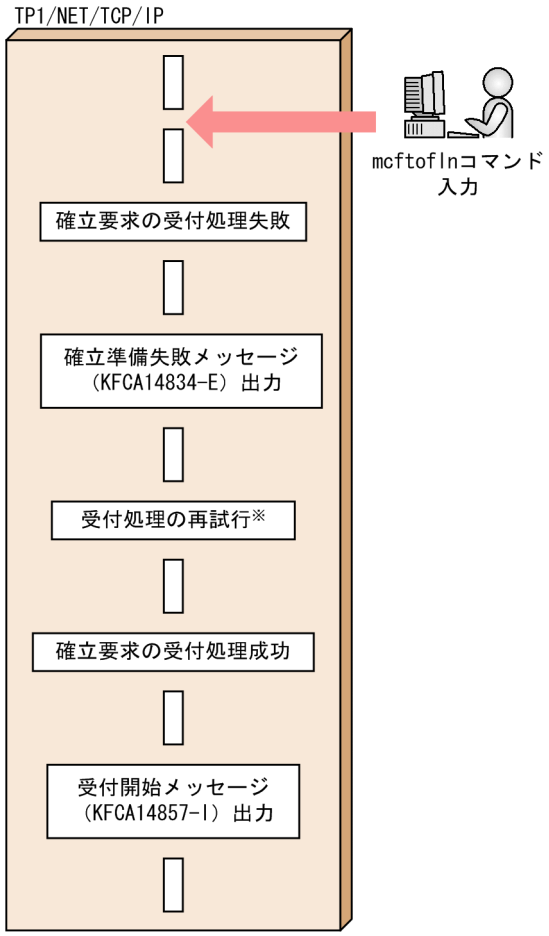


注※　オンライン開始時の自動受付開始では出力されません。

確立要求の受付処理に失敗した場合は，コネクション定義（mcftalccn -b）の bretry オペランド，bretryint オペランド，および bretrycnt オペランドの指定に従って再試行します。再試行回数が指定した上限を超えた場合は，受付処理を終了します。この場合は，障害を取り除いたあとに再度運用コマンドを入力してください。

コネクション確立要求の受付処理の失敗について，運用コマンド（mcftonln）を入力する場合を例に次の図に示します。

図 2-11　コネクション確立要求の受付処理の失敗（運用コマンド入力時）



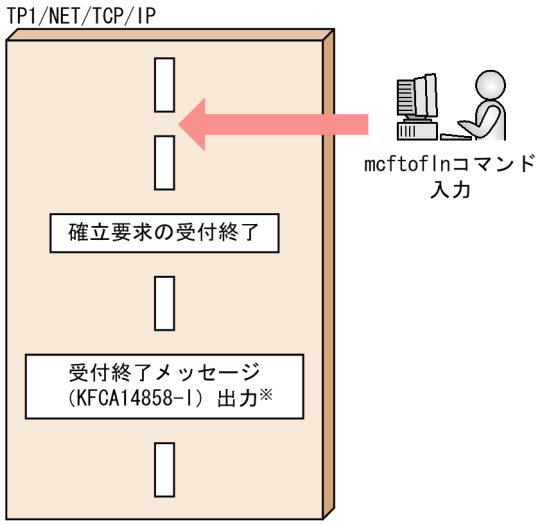
注※　MCF通信構成定義で指定した回数だけ試行します。

(2)　コネクション確立要求の受付終了

運用コマンド（mcftofln）の入力，または API（dc_mcf_tofln 関数または CBLDCMCF('TOFLN△△△')）の発行によって，TP1/NET/TCP/IP は相手システムからのコネクション確立要求の受付を終了します。

コネクション確立要求の受付終了について，運用コマンド（mcftofln）を入力する場合を例に次の図に示します。

図 2-12　コネクション確立要求の受付終了（運用コマンド入力時）



注※　オンライン終了時の自動受付終了では出力されません。

2.1.7　相手アドレスチェックの抑止（サーバ型コネクション）

相手アドレスチェックの抑止は，TP1/NET/TCP/IP による相手システムのアドレス情報のチェックを抑止する機能です。この機能を使用すると，業務運用に合わせた相手システムのアドレスチェックと，コネクション確立の制御ができるようになります。

相手アドレスチェックの抑止機能を使用すると，TP1/NET/TCP/IP は，定義されたコネクションの中から**コネクション選択ルール**に従って未確立コネクションを選び出します。そのあと，コネクション確立 UOC を呼び出し，コネクション確立を受け入れるかどうかを判定します。一方，未確立コネクションがない場合は，コネクション確立要求を拒否します。

コネクション選択ルールについては，「(2)　コネクション選択ルール」を参照してください。また，コネクション確立 UOC については，「(3)　コネクション確立 UOC」を参照してください。

このようにすることで，コネクション確立要求受信時の相手アドレスチェックを抑止し，任意の相手システムからのコネクション確立要求を受け入れられるようになります。相手システムのアドレス情報をコネクション定義に指定しておく必要がありません。

なお，相手アドレスチェックを抑止した場合でも，運用コマンド（mcftlsln -t）を入力することで接続相手を確認できます。

相手アドレスチェックを抑止している場合にコネクション障害が発生したときは対処が必要です。詳細については，「2.1.9(2)　相手アドレス指定時，または相手アドレスチェックの抑止時のコネクション障害」を参照してください。

(1) 相手アドレスチェックを抑止する場合の定義方法

相手アドレスチェックを抑止する場合は，コネクション定義（mcftalccn -h）の addrchk オペランドで no を指定します。このとき，-o オプションの次に示すオペランドに指定する相手システムのアドレス情報の定義は不要です。定義した場合は，無効となります。

- oipaddr オペランド
- ohostname オペランド
- oportno オペランド

また，同時に通信する相手システム数以上のコネクションを指定してください。

注意事項

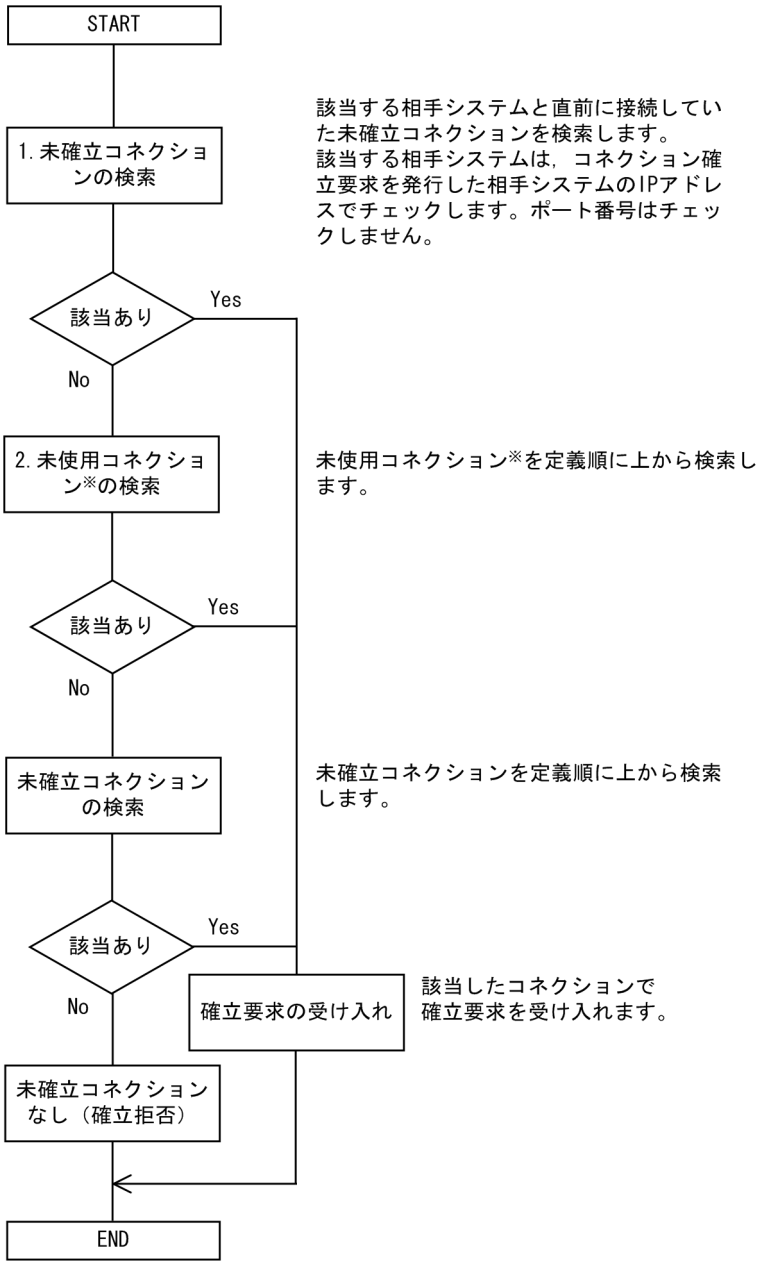
一つの通信プロセスで，相手アドレスチェックを抑止するコネクションと，相手アドレスをチェックするコネクションとの混在はできません。ただし，MCF 通信プロセスを分ければ混在できます。

(2) コネクション選択ルール

TP1/NET/TCP/IP は，相手アドレスチェックの抑止機能を使用してコネクション確立要求を受け入れる場合は，MCF 通信構成定義による構成管理をしません。コネクション選択ルールに従って未確立コネクションを選び出し，コネクション確立要求を受け入れます。未確立コネクションがなければ，コネクション確立要求を拒否します。

TP1/NET/TCP/IP が未確立コネクションを選び出すときの，コネクション選択ルールを次の図に示します。

図 2-13　コネクション選択ルール



注※
未使用コネクションとは，システム起動後，コネクションの確立に一度も使用されたことがないコネクションのことです。

相手システム数以上のコネクションを定義していて，相手システムとの間に 1 本だけ TCP コネクションを確立する場合は，コネクション選択ルールの 1.および 2.の検索で未確立コネクションを選べます。つまり，同じ相手システムとの再接続時には，前回接続時と同じコネクションで確立します。

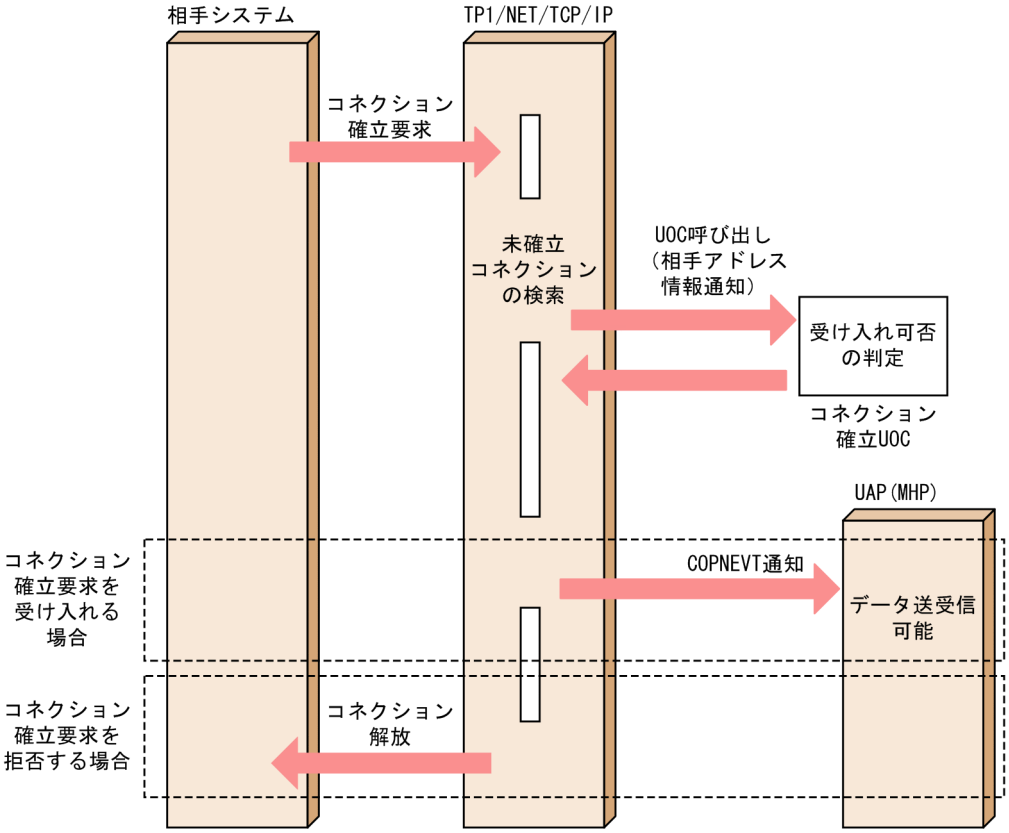
問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用する場合，継続問い合わせ応答中の論理端末に対応するコネクションを割り当て対象とするかどうかをコネクション定義（mcftalccn -l）の cnassign オペランドで指定できます。ただし，継続問い合わせ応答中の論理端末に対応するコネクションを割り当て対象とする指定（all）をしても，継続問い合わせ応答が終了するまでは，該当するコネクションで確立できません。

(3)　コネクション確立 UOC

コネクション確立 UOC を使用して，コネクション確立要求を受け入れてよい相手システムかどうかを判断できます。TP1/NET/TCP/IP はコネクション選択ルールに基づいて未確立コネクションを選択したあと，コネクション確立 UOC を呼び出します。コネクション確立 UOC は，TP1/NET/TCP/IP から渡された相手アドレス情報を基に，受け入れるかどうかを判定します。

コネクション確立要求受信時の流れを次の図に示します。

図 2-14　コネクション確立要求受信時の流れ



2.1.8　コネクションの解放

運用コマンド（mcftdctcn）の入力，または API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）の発行によって，TP1/NET/TCP/IP はコネクションを解放します。自システムからコネクションの解放要求を受けるとメッセージの送受信を終了し，コネクションを解放します。相手システムから解放要求を受信した場合も同様に，解放要求を受信した時点でコネクションを解放します。相手システムからの解放が，正常解放なのか，相手システムの異常による強制解放なのかの区別はできません。さらに，オンライン終了時にも，コネクションは解放されます。

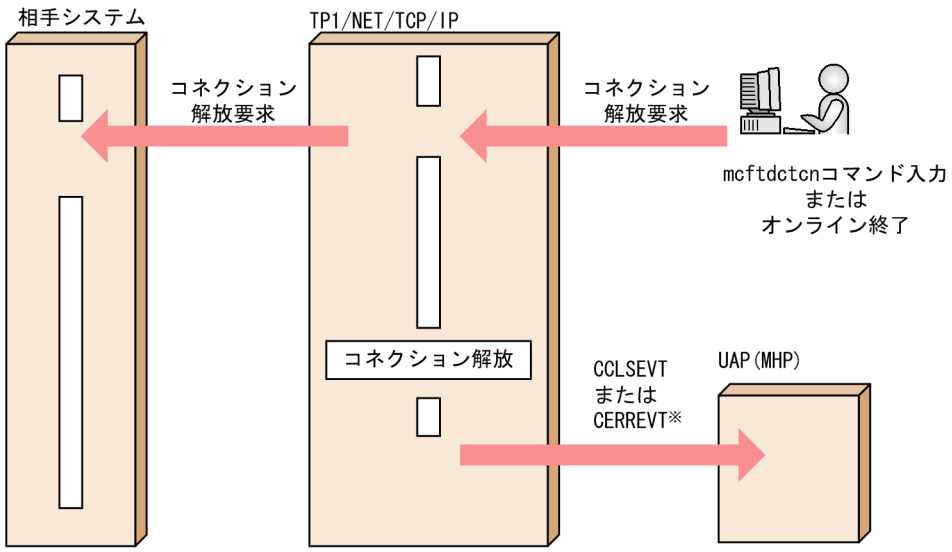
自システムからコネクションを解放した場合，TP1/NET/TCP/IP は，コネクション定義（mcftalccn -f）の cnrelease オペランドの指定によって，状態通知イベント（CCLSEVT），または障害通知イベント（CERREVT）のどちらかを通知します。また，相手システムからのコネクション解放の場合はコネクショ

ン定義（mcftalccn -f）の kind オペランドの指定によって，状態通知イベント（CCLSEVT），または障害通知イベント（CERREVT）のどちらかを通知します。コネクションは，クライアント，およびサーバのどちらからでも解放できます。

メッセージの欠落を防ぐため，コネクションの解放はユーザ間で，データ転送の終了の同期合わせ後に行ってください。

運用コマンド（mcftdctcn）を入力時またはオンライン終了時の，自システムからのコネクションの解放を次の図に示します。

図 2-15　自システムからのコネクションの解放（運用コマンド入力時またはオンライン終了時）



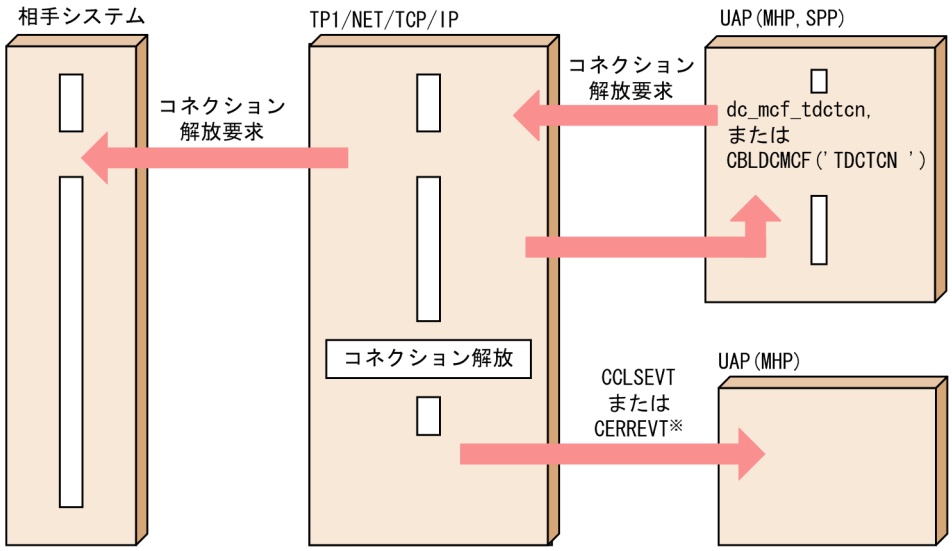
注※

オンライン終了時は通知されません。

運用コマンド（mcftdctcn）の-f オプションの指定の有無，およびコネクション定義（mcftalccn -f）の cnrelease オペランドの指定によって，CCLSEVT または CERREVT のどちらかを通知します。

API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）発行時の，自システムからのコネクションの解放を次の図に示します。

図 2-16　自システムからのコネクションの解放（API 発行時）

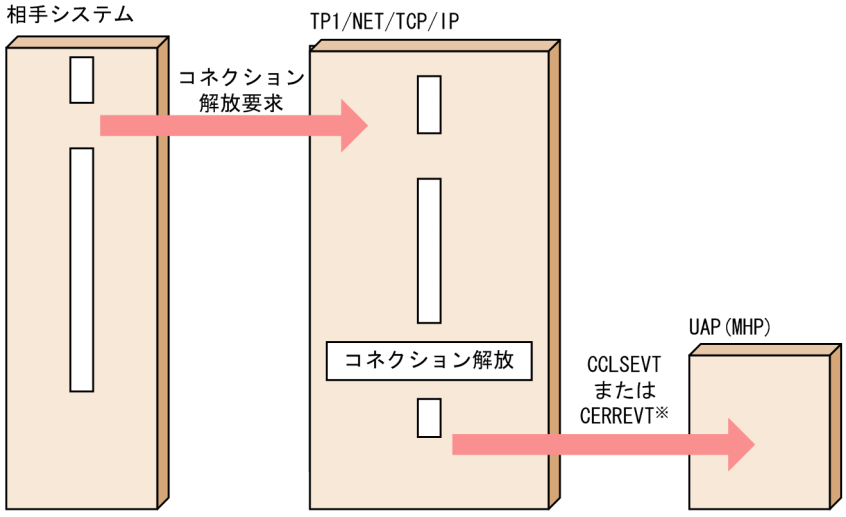


注※

API およびコネクション定義（mcftalccn -f）の cnrelease オペランドの指定によって，CCLSEVT または CERREVT のどちらかを通知します。

相手システムからのコネクションの解放を次の図に示します。

図 2-17　相手システムからのコネクションの解放



注※

オンライン終了時は通知されません。

コネクション定義（mcftalccn -f）の kind オペランドの指定によって，CCLSEVT または CERREVT のどちらかを通知します。

2.1.9　コネクション障害

ネットワーク障害または自システム内の異常によって，メッセージ送受信中にコネクション障害を検出すると，コネクションが解放され，メッセージを送受信できなくなります。このように，オンライン終了時を除いて，運用コマンドや API を使用しないでコネクションが解放されることを，**コネクションの切断**といいます。

ここでは，コネクションの切断と，相手アドレス指定時，または相手アドレスチェックの抑止時のコネクション障害について説明します。

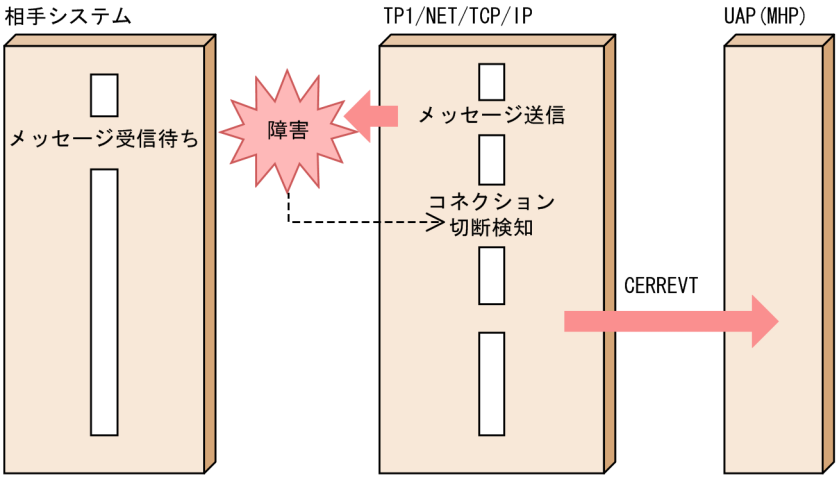
(1)　コネクションの切断

TP1/NET/TCP/IP は障害通知イベント（CERREVT）によって，障害の発生とコネクションの切断を通知します。送信中の場合，該当メッセージはコネクション再確立後に再送されますが，受信中の場合，該当メッセージは破棄されます。

コネクション障害時の対策については，「9.2　コネクション障害時の処理」を参照してください。

コネクションの切断を，次の図に示します。

図 2-18　コネクションの切断



主に障害の検出によって自システムからコネクションを解放する場合，コネクション定義（mcftalccn -f）の cnrelease オペランドを指定することで，コネクションの解放形態を指定できます。解放形態には，FIN パケットを送信してコネクションを解放する形態（fin を指定）と，RST パケットを送信して強制的にコネクションを解放する形態（rst を指定）があります。コネクションを解放する要因と送信するパケットを次の表に示します。

表 2-2　コネクションの解放要因と送信するパケット

コネクションの解放要因		コネクション定義(mcftalccn -f)の cnrelease オペランドの指定値	送信するパケット
mcftdctcn コマンド	-f オプションなし	任意	FIN パケット

コネクションの解放要因		コネクション定義(mcftalccn -f)の cnrelease オペランドの指定値	送信するパケット
mcftdctcn コマンド	-f オプションあり	fin	FIN パケット
		rst	RST パケット
API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）	強制解放オプションなし※1	任意	FIN パケット
	強制解放オプションあり※2	fin	FIN パケット
		rst	RST パケット
相手システムからの解放		任意	FIN パケット
OpenTP1 終了	• 正常終了 • 強制正常終了 • 計画停止 A • 計画停止 B • MCF 構成変更準備停止	任意	FIN パケット
	強制停止	fin	FIN パケット※3
		rst	RST パケット
障害の検出	サーバ型コネクションの確立拒否	任意	FIN パケット
	上記以外（受信バッファ数不足，後続セグメント受信監視タイムアウトなど）	fin	FIN パケット
		rst	RST パケット

注※1

dc_mcf_tdctcn 関数の場合，action 引数に DCMCFFRC を指定しません。

CBLDCMCF('TDCTCN△△')の場合，データ名 D1 に空白または'0'を指定します。

注※2

dc_mcf_tdctcn 関数の場合，action 引数に DCMCFFRC を指定します。

CBLDCMCF('TDCTCN△△')の場合，データ名 D1 に'1'を指定します。

注※3

Windows の場合，RST パケットを送信します。

注意事項

- TP1/NET/TCP/IP は OS が提供する TCP/IP のソケットインタフェースを利用しています。そのため，FIN パケットを送信する指定をしていても，ネットワーク上のデータのすれ違いなどによって，TP1/NET/TCP/IP の指示と関係なく OS が RST パケットを送信する場合があります。

- RST パケットによるコネクション解放の場合，TCP/IP ソケットの送信バッファに蓄積されている送信メッセージがコネクション解放時に破棄されます。送信メッセージが相手システムに到達してからコネクションを解放する場合は，メッセージの送達を確認してからコネクションを解放してください。送達確認については，「2.3.10　メッセージ送達の確認」を参照してください。

(2)　相手アドレス指定時，または相手アドレスチェックの抑止時のコネクション障害

次に示す場合，TP1/NET/TCP/IP は，コネクションの切断後にどの相手システムとコネクションを確立するか特定できません。

- 相手アドレスを指定してコネクションを確立する場合
- 相手アドレスチェックの抑止機能の使用時で次の場合
  - コネクション定義数が相手システム数より小さい
  - 相手システムとの間に複数本の TCP コネクションを確立している

このような場合に出力キューまたは入力キューにメッセージがあるときは，次のことに注意する必要があります。

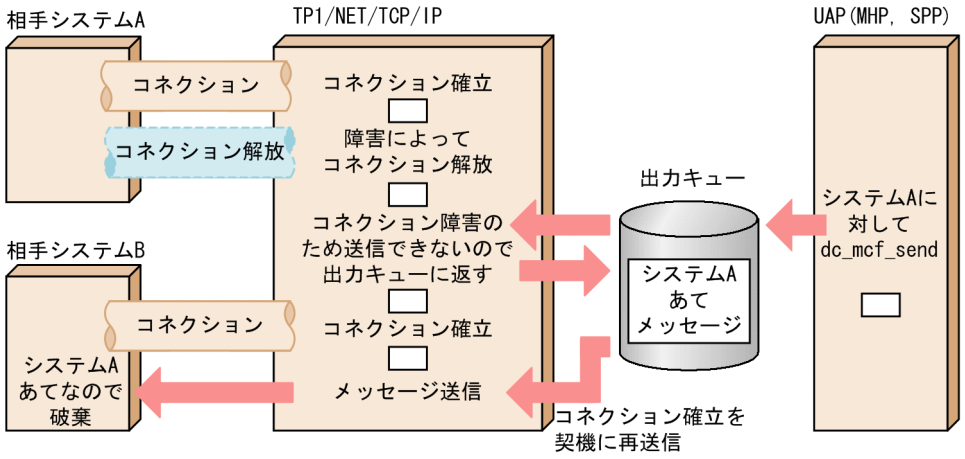
(a)　コネクション障害時に出力キューにメッセージがある場合

いったんコネクションが解放されると，そのコネクションは次にどの相手システムと接続するのか特定できないため，現在出力キューにあるメッセージが本来のあて先でない相手に送信される場合があります。

このため，コネクション確立前に，出力キューに残っている不要なメッセージを運用コマンド（mcftdlqle）の入力，または API（dc_mcf_tdlqle 関数もしくは CBLDCMCF('TDLQLE△△')）の発行によって削除するようにしてください。ただし，出力キューに残っている不要なメッセージを削除するタイミングよりも早く，別の相手システムと接続されてメッセージが送信されてしまう場合があります。相手システムが不要なメッセージを受信してしまった場合は，メッセージ受信後にそのメッセージを破棄してください。

コネクション障害時に出力キューにメッセージがあるときの流れを，dc_mcf_send 関数を使用する場合を例に次の図に示します。

図 2-19　コネクション障害時に出力キューにメッセージがある場合の流れ



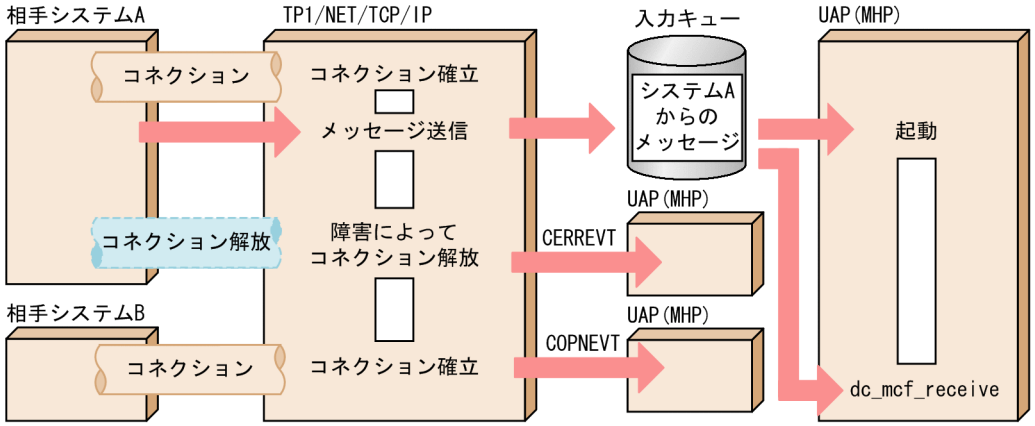
なお，MHP からメッセージを送信する場合は，コネクション再確立時に未送信メッセージの送信を抑止できます。詳細については，「2.3.12　コネクション再確立時の未送信メッセージの送信抑止」を参照してください。

(b)　コネクション障害時に入力キューにメッセージがある場合

いったんコネクションが解放されると，そのコネクションは次にどの相手システムと接続するのかを特定できないため，現在入力キューにあるメッセージが，現在接続している相手システムから送信されたものではない場合があります。したがって，UAP 側で，メッセージ内容などからどの相手システムから送信されたメッセージなのかを判断する必要があります。

コネクション障害後に入力キューからメッセージを取り出したときの流れを，dc_mcf_receive 関数を使用する場合を例に次の図に示します。

図 2-20　コネクション障害後に入力キューからメッセージを取り出した場合の流れ



2.1.10　コネクションリプレース（サーバ型コネクション）

コネクションリプレースは，相手システムからコネクション確立要求を受け付けた場合，TP1/NET/TCP/IP に割り当て可能な未確立コネクションがなければ，確立状態にある既存のコネクションを切断し，新た

なコネクション確立要求を受け付ける機能です。コネクションリプレースを使用しない場合に比べ，相手システムでの障害検出を契機としたコネクションの再確立を迅速に行えます。

コネクションリプレースは，コネクション定義（mcftalccn -h）の chgconn オペランドに replace を指定したサーバ型コネクションに対して使用できます。コネクションリプレースを使用する場合の注意事項については，「(4)　コネクションリプレースを使用するときの注意事項」を参照してください。

コネクションの切断については，「2.1.9　コネクション障害」を参照してください。

(1)　コネクションリプレースの概要

コネクションリプレースを使用しない場合と使用する場合に分け，相手システムが障害を検出したあとの処理について説明します。

(a)　コネクションリプレースを使用しない場合

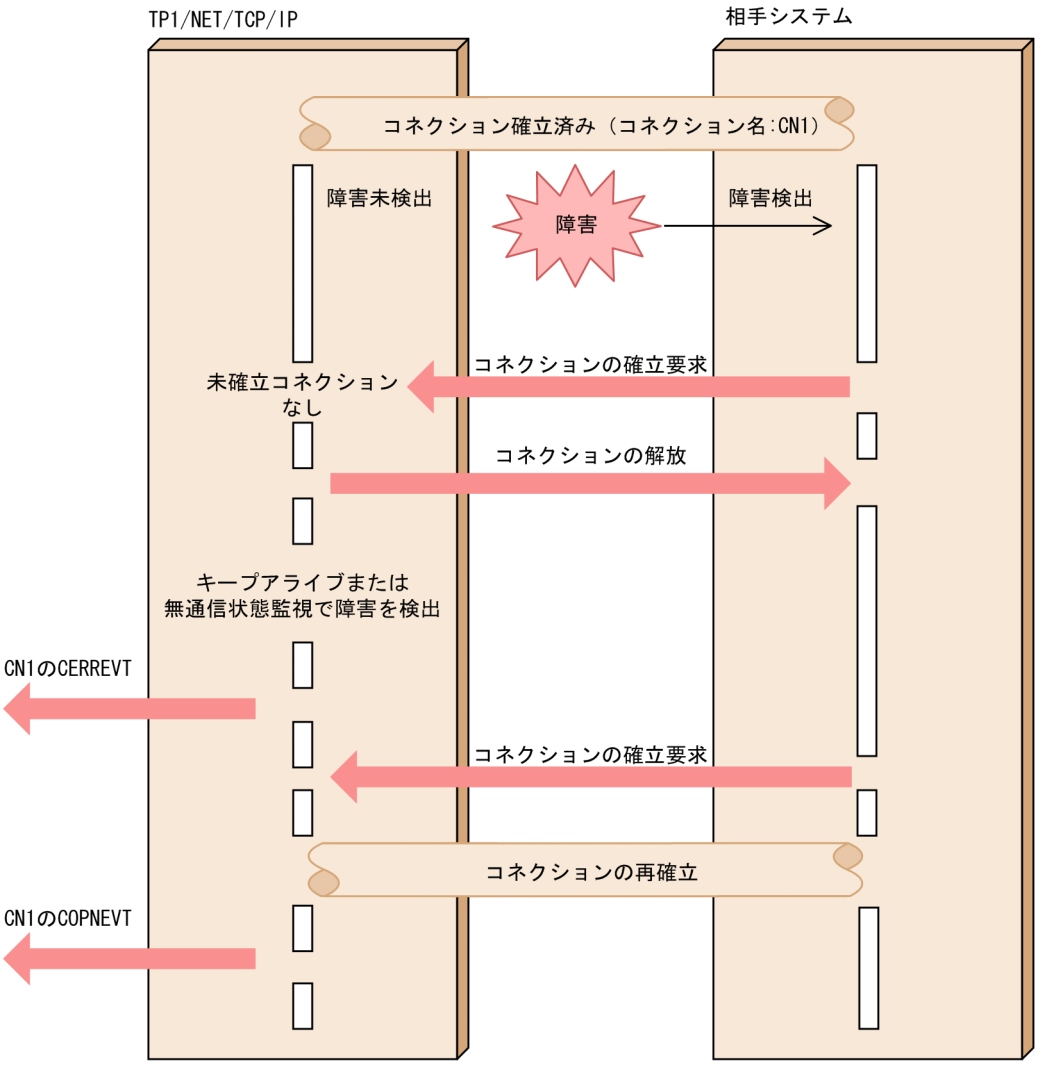
相手システムが TP1/NET/TCP/IP よりも先に障害を検出し，コネクションの再確立を要求した場合，割り当て可能な未確立コネクションがなければ，コネクション確立要求によって一時的に受け入れた新しいコネクションを解放します。

なお，次の状態になるまで，該当のコネクションは再確立できません。

- ユーザが障害を検出して運用コマンド（mcftdctcn）を入力するか，または API（dc_mcf_tdctcn 関数もしくは CBLDCMCF('TDCTCN△△')）を発行する。
- TP1/NET/TCP/IP がキープアライブ，または無通信状態監視で障害を検出する。

コネクションリプレースを使用しない場合の処理の流れを次に示します。

図 2-21　相手システムが障害を検出したあとの処理の流れ（コネクションリプレース未使用時）

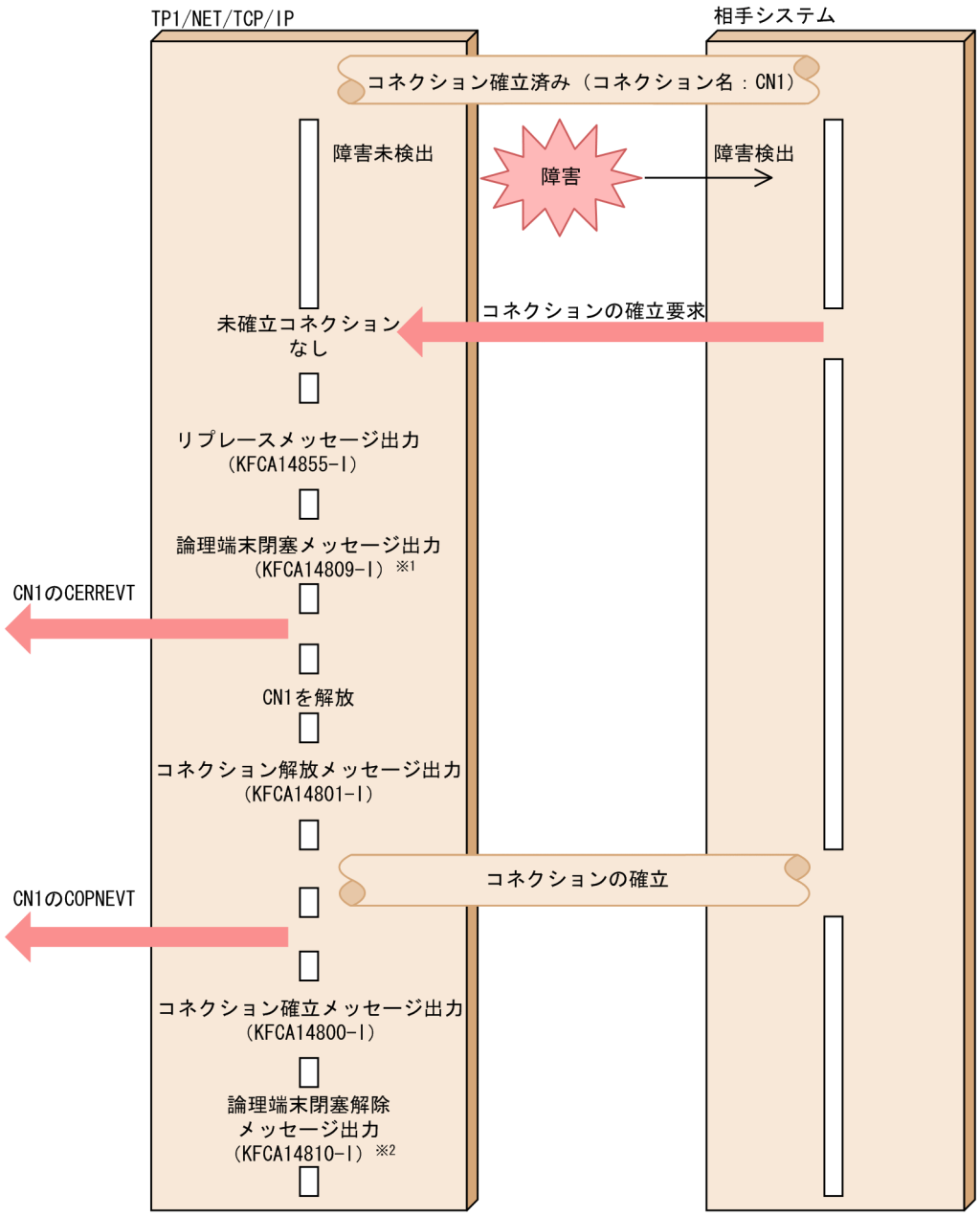


(b)　コネクションリプレースを使用する場合

相手システムが TP1/NET/TCP/IP よりも先に障害を検出し，コネクションの再確立を要求した場合，割り当て可能な未確立コネクションがなければ，確立状態の既存のコネクションは切断されます。その後，TP1/NET/TCP/IP は新たなコネクションの確立要求を受け付け，切断したコネクションを再確立します。

コネクションリプレースを使用した場合の処理の流れを次に示します。

図 2-22　相手システムが障害を検出したあとの処理の流れ（コネクションリプレース使用時）



注※1

論理端末が閉塞解除状態の場合に出力されます。

注※2

論理端末定義（mcftalcle -i）の auto オペランドを指定し，自動的に論理端末を閉塞解除するよう設定している場合に出力されます。

(2)　複数のコネクションを定義した場合のコネクション選択ルール

次の場合，自システムの IP アドレスおよびポート番号が同じコネクションを複数定義できます。

- ポートフリーのコネクション

コネクション定義（mcftalccn -o）の oportno オペランドに free を指定したコネクション。

- 相手アドレスのチェックを抑止したコネクション
  コネクション定義（mcftalccn -h）の addrchk オペランドに no を指定を指定したコネクション。

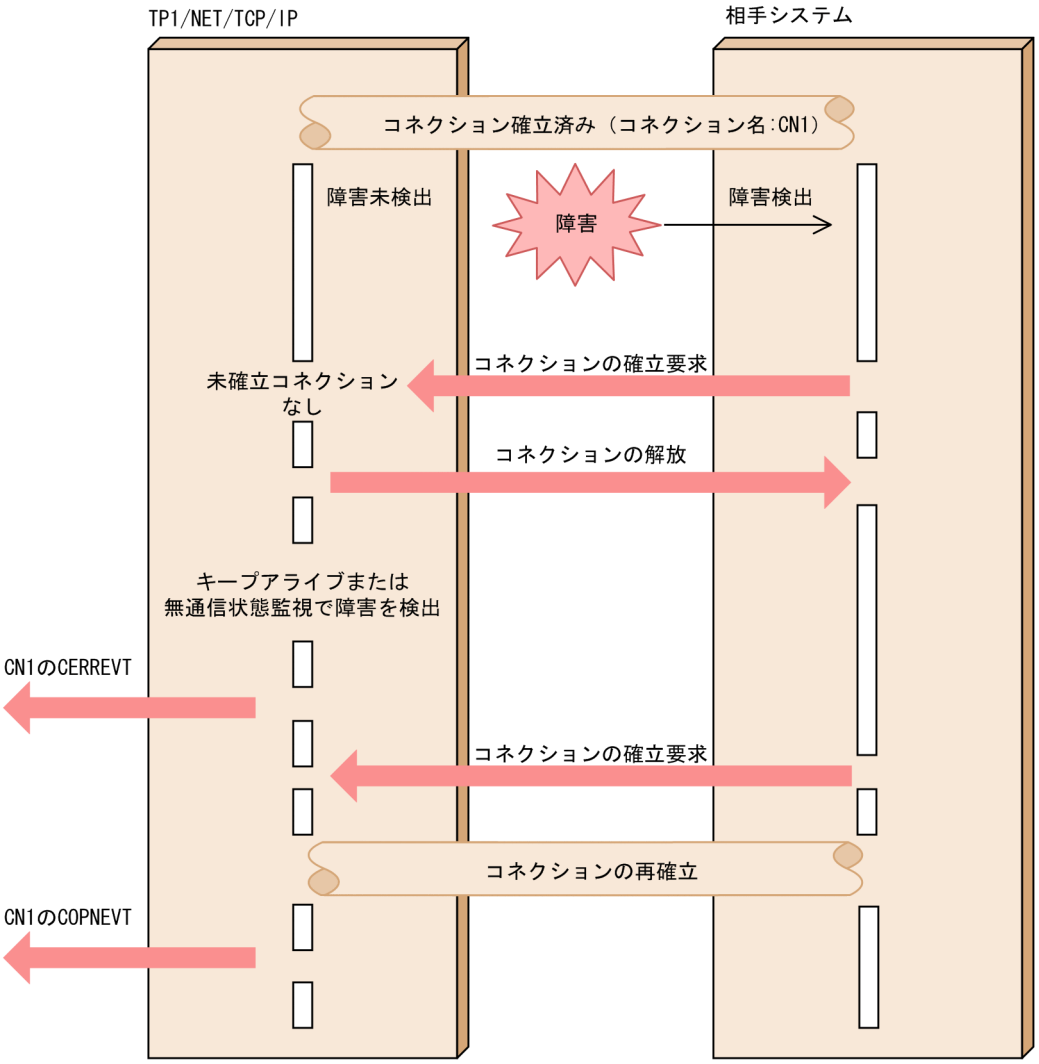
このような複数コネクションに対して，コネクションリプレースを使用できるのは，自システムの IP アドレスおよび自システムのポート番号が同じで，かつそれらすべてのコネクションが確立状態にあるときです。

ポートフリーのコネクションに対してコネクションリプレースを使用する場合，TP1/NET/TCP/IP は自システムの IP アドレスおよびポート番号が同じコネクションの中から，確立状態にあるいちばん古いコネクションをコネクションリプレースの対象として選択します。

(a) ポートフリーのコネクションの場合

ポートフリーのコネクションに対して，コネクションリプレースを使用する場合のコネクション選択ルールを次の図に示します。

図 2-23 コネクション選択ルール（ポートフリーのコネクションの場合）



(b)　相手アドレスチェックを抑止したコネクションの場合

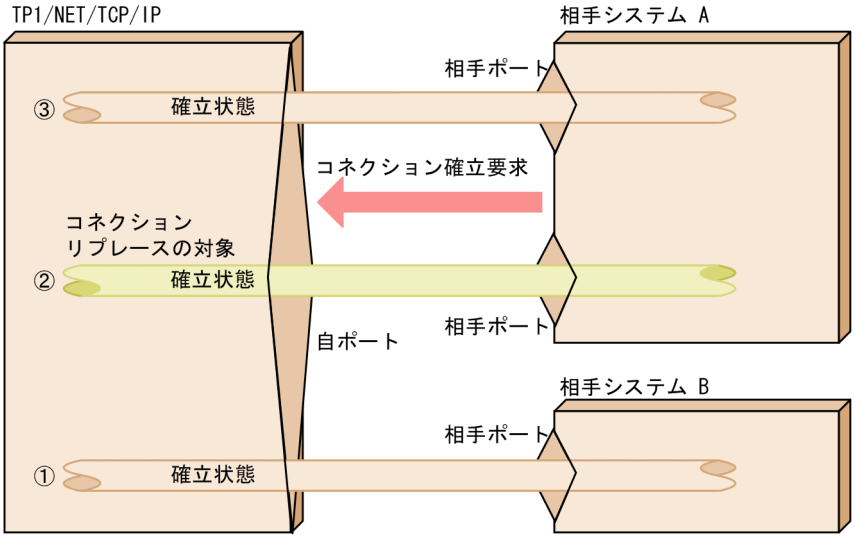
相手アドレスチェックを抑止したコネクションに対してコネクションリプレースを使用する場合には，TP1/NET/TCP/IP は次の両方の条件と一致する，確立状態にあるいちばん古いコネクションをコネクションリプレースの対象として選択します。

- 接続している既存コネクションの相手アドレスと，確立要求した相手のアドレスが一致
- 自システムの IP アドレスおよびポート番号が同じ

なお，該当するコネクションがない場合，ポートフリーのコネクションの場合と同様に，自システムの IP アドレスおよびポート番号が同じコネクションの中からいちばん古いコネクションをコネクションリプレース対象として選択します。

相手アドレスチェックを抑止したコネクションに対して，コネクションリプレースを使用する場合のコネクション選択ルールを次の図に示します。

図 2-24　コネクション選択ルール（相手アドレスチェックを抑止したコネクションの場合）

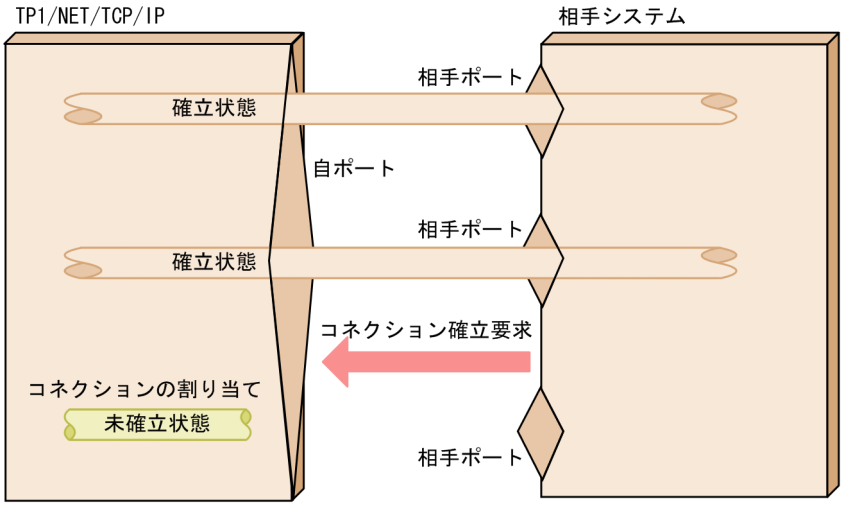


（凡例）
①，②，③：コネクションを確立した順番

(c)　未確立コネクションが存在する場合

未確立コネクションが存在する場合は，コネクションリプレースを行わないで，未確立コネクションを割り当てます。未確立コネクションが存在する場合のコネクション選択ルールを次に示します。

図 2-25　コネクション選択ルール（未確立コネクションが存在する場合）



(3)　コネクション確立 UOC を併用する場合

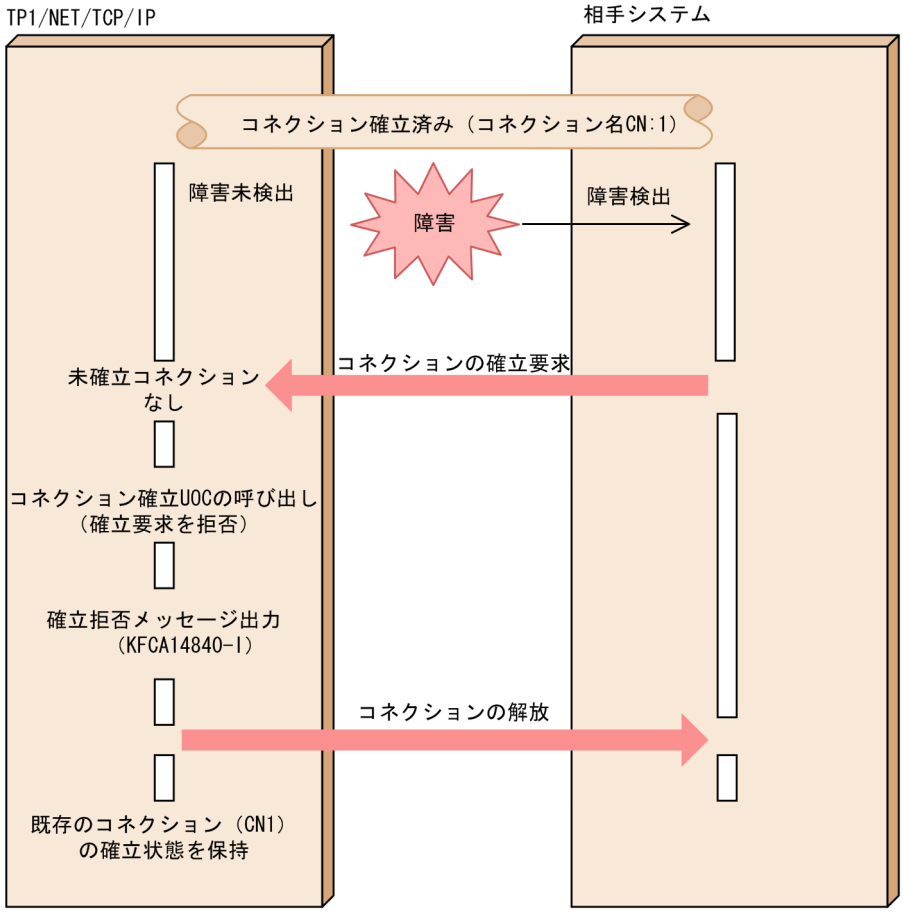
コネクションリプレースとコネクション確立 UOC を併用する場合，TP1/NET/TCP/IP がコネクションリプレースを行うコネクションを選択したあと，コネクション確立 UOC を呼び出します。コネクション確立 UOC には，相手システムのアドレス，ポート番号および自システムのポート番号を通知するので，これらの情報を基に相手システムからの確立要求を受け入れるか，または拒否するかを選択できます。

相手システムからの確立要求を受け入れた場合，コネクション確立 UOC を併用しないときと同様に，コネクションリプレースを行います。詳細は，「(1)　コネクションリプレースの概要」を参照してください。

コネクション確立要求を受け入れない場合，コネクションリプレースは行いません。また，コネクション確立 UOC がエラーリターンした場合，コネクション確立 UOC で設定したパラメタが誤っていた場合もコネクションリプレースは行いません。

コネクション確立要求を受け入れない場合の処理の流れを次の図に示します。

図 2-26　コネクション確立 UOC との併用（コネクションの確立要求を拒否した場合）



(4)　コネクションリプレースを使用するときの注意事項

コネクションリプレースを使用するときには，次の点に注意してください。

- コネクションリプレースは，相手システムのポート番号が固定（コネクション定義（mcftalccn -o）の oportno オペランドに 1～65535 を指定）のサーバ型コネクションには適用されません。ただし，相手システムのポート番号が固定のサーバ型コネクションの場合は，RFC 793 の規定によって，相手システムからのコネクション確立要求を契機にして，OS がコネクション障害を検出します。そのため，相手システムのポート番号が固定のサーバ型コネクションはコネクションリプレースを適用しなくても障害回復を行えます。
- コネクションリプレースを使用する場合，TP1/NET/TCP/IP は選択したコネクションの状態に関係なく，確立状態にある既存のコネクションを切断します。受信処理中のメッセージは破棄し，送信処理中の一方送信メッセージは出力キューに戻ります。また，同期型メッセージの送信関数および同期型メッセージの送受信関数を発行していた場合は，出力先論理端末の停止を示すリターン値またはステータスコードでエラーリターンします。
- 問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用する場合，継続問い合わせ応答中の論理端末に対応するコネクションをコネクションリプレースの対象とするかどうかをコネクション定義（mcftalccn -l）の cnassign オペランドで指定できます。ただし，継続問い合わせ応

答中の論理端末に対応するコネクションをコネクションリプレースの対象とする指定（all）をしても，継続問い合わせ応答が終了するまで該当するコネクションで再確立できません。

2.1.11　コネクションの状態表示

コネクションが確立状態か解放状態かといったコネクションの状態や，コネクション ID，プロトコル種別などを表示することができます。コネクションの状態を表示するには，運用コマンド（mcftlscn）を入力するか，または API（dc_mcf_tlscn 関数もしくは CBLDCMCF('TLSCN△△△')）を発行します。

運用コマンド（mcftlscn）を使用した場合，コネクションの状態は，標準出力で表示されます。一方，dc_mcf_tlscn 関数または CBLDCMCF('TLSCN△△△')を使用した場合，コネクションの状態は UAP 中で指定した領域に格納されます。

運用コマンド（mcftlscn）のオプションの指定によっては，コネクションに対応する論理端末の情報も表示できます。

2.1.12　コネクションの切り替え（クライアント型コネクション）

クライアント型コネクションの場合，TP1/NET/TCP/IP では，コネクションの切り替えができます。その場合，前提となるプログラムプロダクトとして，TP1/NET/High Availability が必要です。

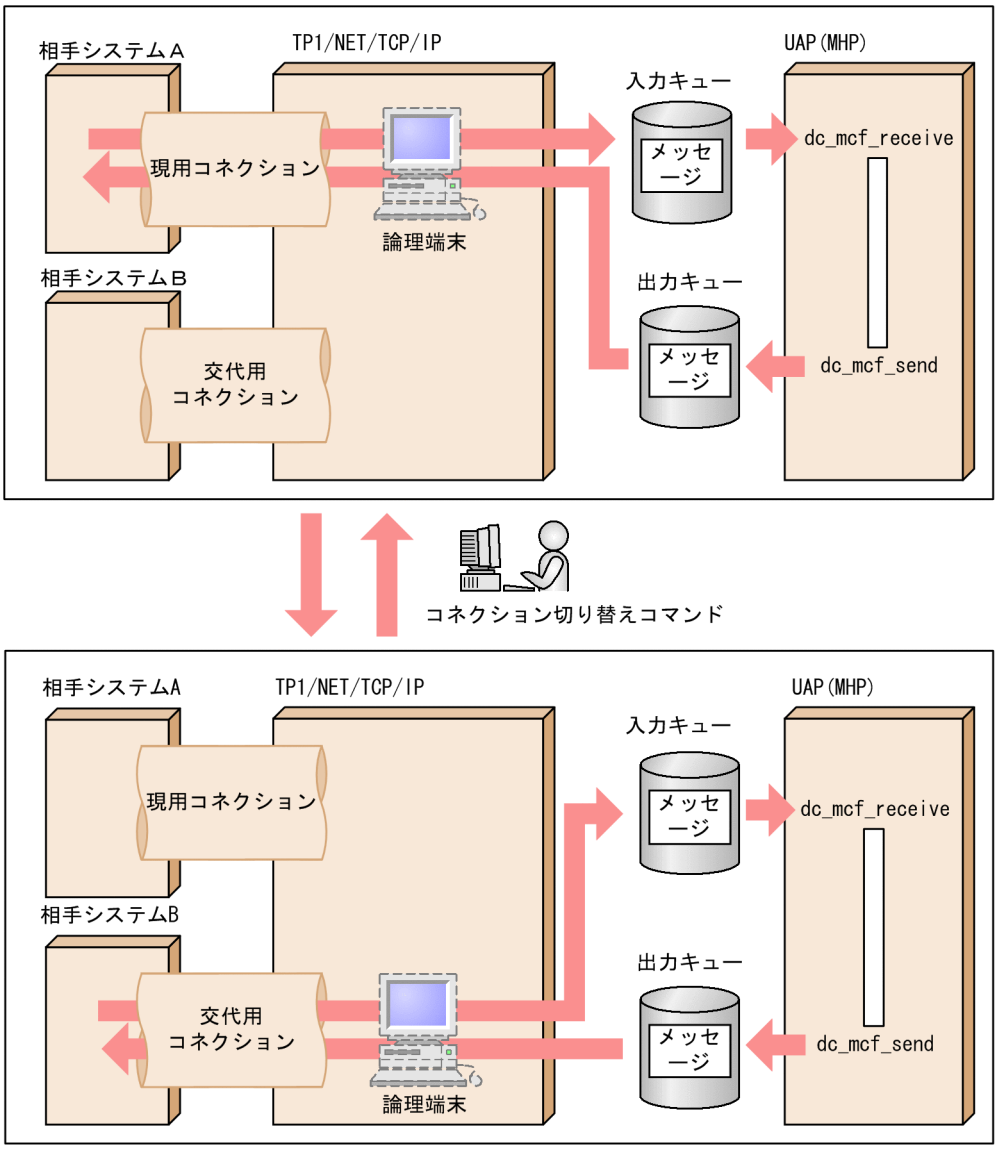
TP1/NET/TCP/IP は，システム定義で，現用コネクションと交代用コネクションに対して同一の論理端末を指定することで，コネクションと論理端末との接続関係を変更できます。UAP は，同一の論理端末名称を使用したまま，相手システムの切り替えができます。

また，現用コネクションと交代用コネクションは，1 対 1 の関係でなければなりません。出力キューに未送信メッセージがある状態でコネクションの切り替えをすると，未送信メッセージは交代用コネクションに対して送信されます。

交代用コネクションと MCF 通信構成定義との関係については，「6.　システム定義」の「コネクションの形態と MCF 通信構成定義との関係」を参照してください。

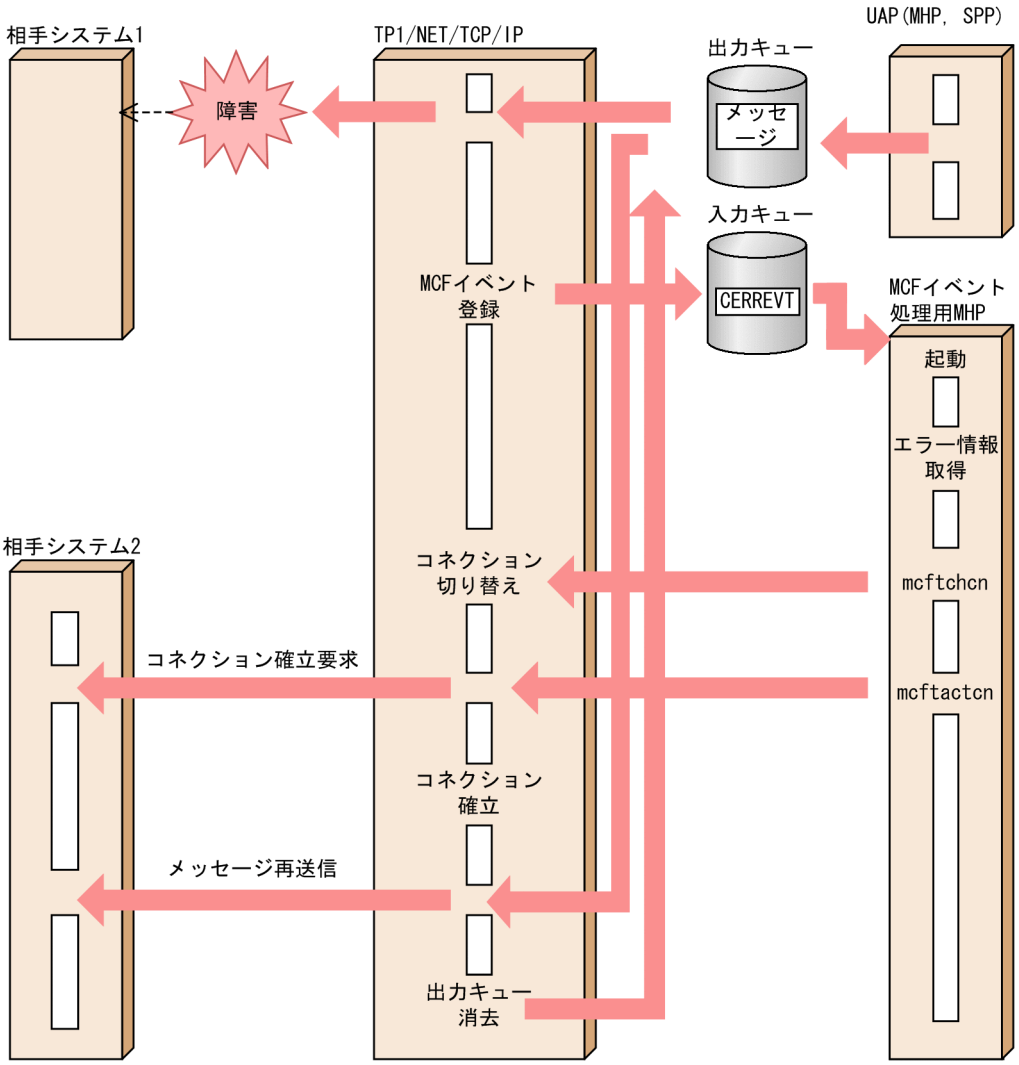
コネクション切り替え機能の概要を，dc_mcf_send 関数および dc_mcf_receive 関数を使用する場合を例に，次の図に示します。

図 2-27　コネクション切り替え機能の概要



次に，ホットスタンバイを行うような，同一の構成を持つシステムでコネクションの切り替えを適用した場合の例を次の図に示します。

図 2-28　コネクションの切り替えの流れ



メッセージ送受信中のコネクションに障害が発生すると，TP1/NET/TCP/IP は MCF イベント処理用 MHP を起動します。起動された MCF イベント処理用 MHP では，相手システムを交代させる場合，運用コマンド（mcftchcn）を入力して，通信先の相手システムを変更します。メッセージは，コネクションを切り替えたあとで，交代用コネクション確立時に再送信されます。コネクション切り替え後，元に戻す場合は運用コマンド（mcftchcn）を再び入力します。

2.1.13　キープアライブ

障害は TCP/IP の特性上，確実に検出できません。障害を確実に検出するために，TP1/NET/TCP/IP では OS が提供しているソケットオプション「SO_KEEPALIVE」を使用して，キープアライブを行います。**キープアライブ**とは，コネクションがメッセージ送受信中でない場合に，相手システムに対して，一定間隔でパケットを送信し，その間に応答がなければ障害と見なしてコネクションを切断する機能です。キープアライブを使用するかどうかはコネクション定義（mcftalccn -k）の keepalive オペランドで指定します。

TCP/IP のキープアライブの仕様は OS の実装によって異なり，ネットワークの負荷を高める場合があります。コネクション障害を確実に検出したい場合だけ使用してください。なお，キープアライブによって障害が検出された場合は，KFCA14802-E が出力されます。

2.1.14 TCP_NODELAY

TCP_NODELAY は，送信済みデータの応答待ちの状態でも遅延させることなくデータ送信できるようにする機能です。この機能では，OS が提供するソケットオプション「TCP_NODELAY」を使用します。この TCP_NODELAY の仕様は，OS によって異なります。

データ送信の際に TCP_NODELAY を使用するかどうかは，コネクション定義（mcftalccn -k）の nodelay オペランドで指定します。なお，TCP_NODELAY を使用するとネットワークの負荷が大きくなるため，この機能の必要性を十分に検討した上で使用するようにしてください。

2.1.15 無通信状態監視

無通信状態とは，任意のコネクションにおいて，相手システムとの間で，メッセージの送信または受信がない状態のことです。TP1/NET/TCP/IP は，この無通信状態を監視します。無通信状態の監視時間は，コネクション定義（mcftalccn -k）の notrftime オペランドで指定します。指定した時間を超過した場合，何らかの障害が発生しているものとして，コネクションを強制解放します。

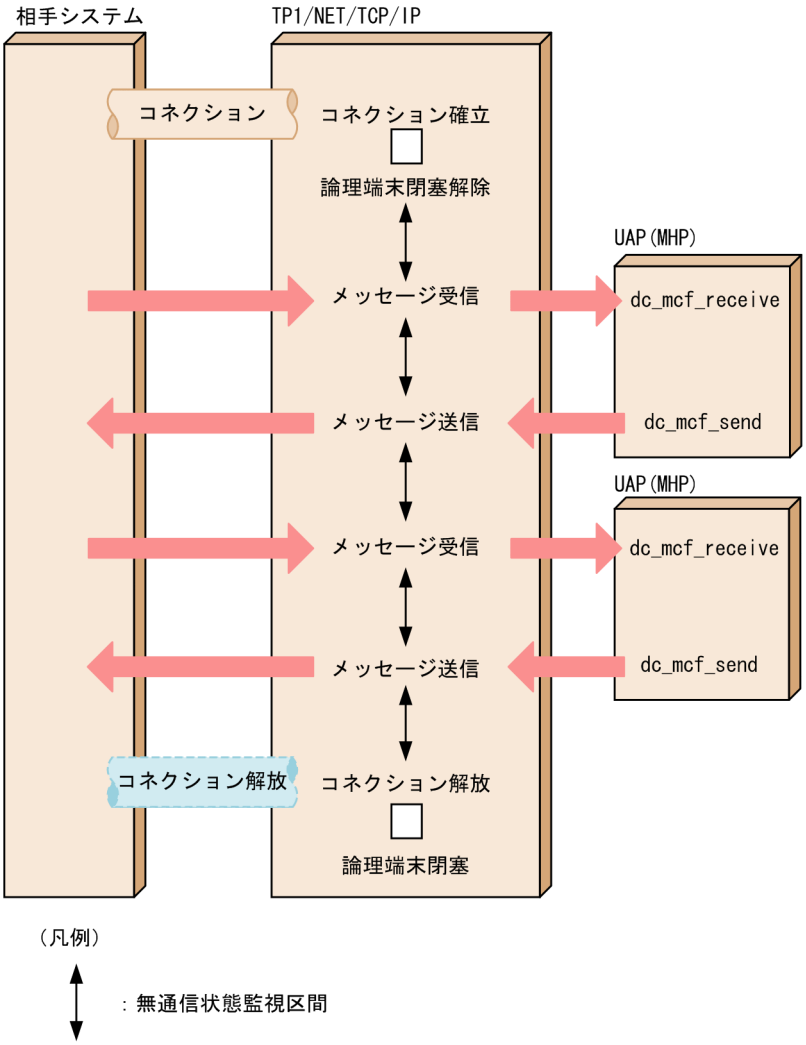
コネクション障害を検出する機能としてキープアライブがあります。キープアライブがソケットオプション「SO_KEEPALIVE」を使用して一定間隔でパケットを送信しながらコネクションの状態を監視するのに対し，無通信状態監視機能はパケットを送信しないでコネクションの状態を監視します。

なお，無通信状態の監視時間がタイムアウトした場合は，KFCA16520-E が出力されます。

TP1/NET/TCP/IP は，各コネクションに対して該当の論理端末が閉塞解除された時点から，論理端末の閉塞または OpenTP1 の終了（正常終了，計画停止 A，計画停止 B）までを監視します。また，メッセージを受信，または送信すると，状態監視をリセットし，改めて次のメッセージの受信または送信までの監視を開始します。

TP1/NET/TCP/IP が無通信状態を監視する区間を，dc_mcf_send 関数および dc_mcf_receive 関数を使用する場合を例に，次の図に示します。

図 2-29　無通信状態を監視する区間



2.2　論理端末に関する機能

ここでは，論理端末に関する機能について説明します。

2.2.1　論理端末とアプリケーションの型の関係

TP1/NET/TCP/IP で扱う論理端末の端末タイプは any（任意型）です。この端末タイプを指定することで，TP1/NET/TCP/IP で使用するすべての通信形態に対応できます。

アプリケーションは，ユーザが送受信データの中に指定したアプリケーション名をキーとして，一つの UAP（MHP）プロセスで実行されます。アプリケーションは，サービスの方式によって型が異なります。この型を TP1/Message Control のアプリケーション属性の一つとして，システム定義時に指定します。TP1/NET/TCP/IP のアプリケーションの型を次に示します。

- 非応答型（noans）
- 応答型（ans）
- 継続問い合わせ応答型（cont）

論理端末の端末タイプ，アプリケーションの型，メッセージの種類，UAP インタフェース，および通信形態の関係を次の表に示します。

表 2-3　論理端末の端末タイプ，アプリケーションの型，メッセージの種類，UAP インタフェース，および通信形態の関係

論理端末の端末タイプ	アプリケーションの型	メッセージの種類	UAP インタフェース	通信形態
any（任意型論理端末）	非応答型（noans）	一方送信メッセージ	dc_mcf_send dc_mcf_reply	一方送信
			dc_mcf_receive	一方受信
		同期型メッセージ	dc_mcf_sendsync	同期送信
			dc_mcf_recvsync	同期受信
			dc_mcf_sendrecv	同期送受信
	応答型（ans）	問い合わせメッセージ	dc_mcf_receive	問い合わせ応答
		応答メッセージ	dc_mcf_reply	
		一方送信メッセージ	dc_mcf_send	一方送信
	継続問い合わせ応答型（cont）	問い合わせメッセージ	dc_mcf_receive	問い合わせ応答
		応答メッセージ	dc_mcf_reply	
		一方送信メッセージ	dc_mcf_send	一方送信

それぞれのアプリケーションの型での，dc_mcf_reply，dc_mcf_send 呼び出しの可否を次の表に示します。

表 2-4　dc_mcf_reply，dc_mcf_send 呼び出しの可否

アプリケーションの型	UAP インタフェース	送信先の論理端末	
		入力元	入力元以外
非応答型（noans）	dc_mcf_reply	○※	×
	dc_mcf_send	○	○
応答型（ans）	dc_mcf_reply	◎	×
	dc_mcf_send	○	○
継続問い合わせ応答型（cont）	dc_mcf_reply	◎	×
	dc_mcf_send	○	○

（凡例）

◎：次の場合は，必ず呼び出してください。

・dc_mcf_execap 関数を呼び出してほかのアプリケーションに応答の権利を譲渡しない場合

・応答メッセージを送信したかどうかをチェックする（アプリケーション属性定義（mcfaalcap -n）の replychk オペランドを省略，または yes を指定）応答型のアプリケーションの場合

dc_mcf_execap 関数を呼び出して，応答の権利をほかのアプリケーションに譲渡した場合は呼び出せません。

○：呼び出せます。

×：呼び出せません。

注※

非応答型のアプリケーションからの問い合わせ応答をする（UAP 共通定義（mcfmuap）の-c オプションの noansreply オペランドに yes を指定）場合に呼び出せます。

一方送信メッセージとして送信します。

2.2.2　論理端末の状態表示

論理端末の状態は，運用コマンド（mcftlsle）の入力，または API（dc_mcf_tlsle 関数もしくは CBLDCMCF('TLSLE△△△')）の発行によって表示できます。表示できる内容は，MCF 識別子，論理端末名称，論理端末状態（閉塞状態，または閉塞解除状態）などです。

運用コマンド（mcftlsle）を使用した場合，論理端末の状態は，標準出力で表示されます。一方，dc_mcf_tlsle 関数または CBLDCMCF('TLSLE△△△')を使用した場合，論理端末の状態は UAP 中で指定した領域に格納されます。

2.2.3　論理端末の閉塞と閉塞解除

TP1/NET/TCP/IP は，コネクション確立後，または確立と同時に論理端末の閉塞を解除してメッセージの送受信をします。論理端末の状態について次に示します。

(1) 閉塞状態

UAP が送信要求したメッセージを，相手システムに送信できない状態です。この状態で UAP が送信要求を行った場合，送信要求は正常に受け付けられますが，送信メッセージは出力キューに滞留します。

また，この状態のとき，相手システムからの受信メッセージのスケジュールは，正常に行われます。

論理端末を閉塞するには，運用コマンド（mcftdctle）を入力するか，または API（dc_mcf_tdctle 関数もしくは CBLDCMCF('TDCTLE△△')）を発行します。閉塞中の一方送信メッセージの送信要求は，出力キューに滞留します。

なお，論理端末は障害によって閉塞することもあります。

(2) 閉塞解除状態

論理端末が持つ機能を使用できる状態です。

論理端末を閉塞解除するには，運用コマンド（mcftactle）を入力するか，または API（dc_mcf_tactle 関数もしくは CBLDCMCF('TACTLE△△')）を発行します。閉塞が解除されると，出力キューに残っているメッセージが送信されます。

なお，コネクションが未確立の場合は，論理端末の閉塞解除はできません。

2.3　メッセージ送受信に関する機能

TP1/NET/TCP/IP を使用すると，上位プロトコル（ユーザプロトコル）を意識しないでメッセージを送受信できます。TP1/NET/TCP/IP では，一方送信メッセージ，および同期型メッセージを使用してメッセージの送受信をします。

2.3.1　一方送信メッセージ

相手システムから一方送信メッセージを受信したり，自システムから一方送信メッセージを送信したりする形態です。UAP（SPP，MHP）のトランザクションの決着と連動してメッセージを送受信します。非トランザクションの MHP の場合は，サービスの終了と連動してメッセージを送受信します。論理端末の端末タイプは any，アプリケーションの型は非応答型です。なお，一方送信メッセージの送信は，応答型または継続問い合わせ応答型のアプリケーションでもできます。

UAP が API（dc_mcf_send 関数または CBLDCMCF('SEND△△△△')）で送信要求すると，メッセージが出力キューに登録されます。TP1/NET/TCP/IP は出力キューからメッセージを読み出し，出力メッセージ編集 UOC で編集されたメッセージを相手システムに送信します。一方送信メッセージを TCP/IP の送信バッファに書き込み完了した時点で，送信完了として送信完了通知イベント（SCMPEVT）を通知します。ただし，SCMPEVT を通知するのは，MCF アプリケーション定義で SCMPEVT を定義して，かつ，UAP が送信完了通知イベントを要求していた場合だけです。その後，TP1/NET/TCP/IP は出力キューにある送信済みメッセージのデータを消去します。

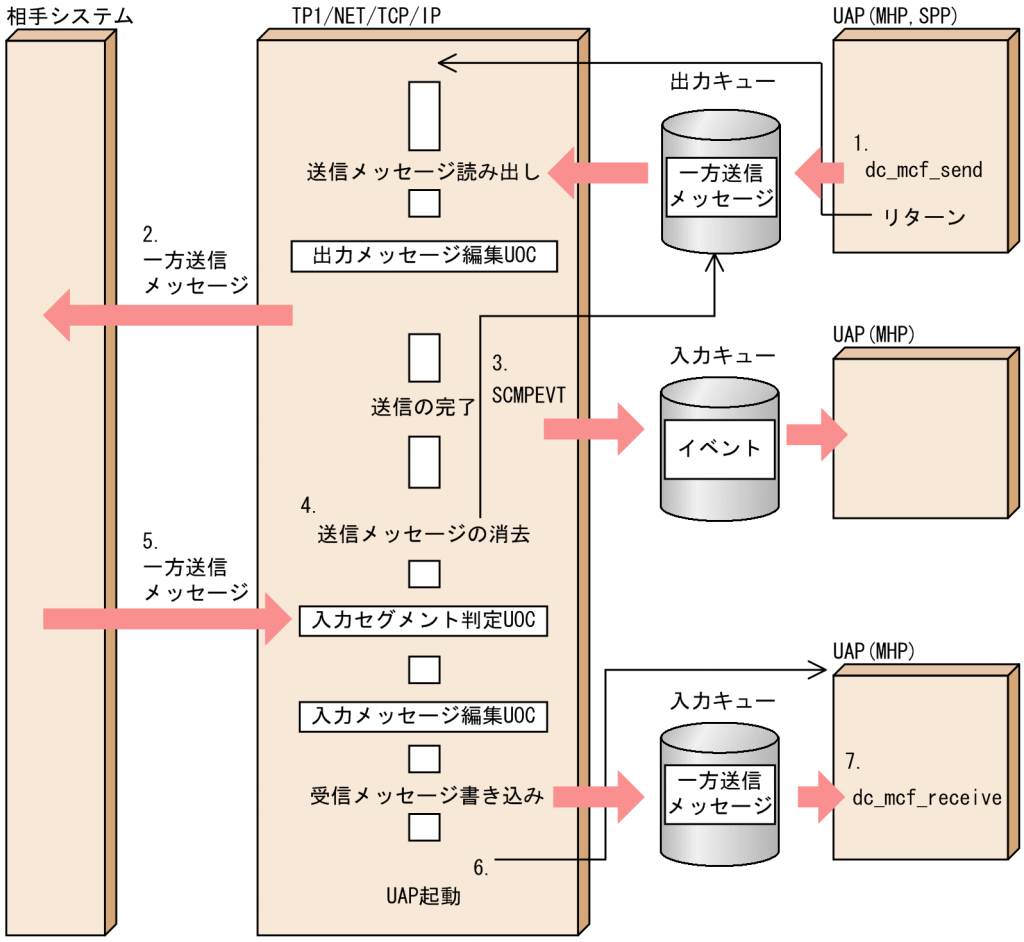
また，相手システムからのメッセージを受信すると，TP1/NET/TCP/IP は入力メッセージ編集 UOC で編集されたメッセージを入力キューに登録します。同時に，非応答型のアプリケーションに対応する UAP（MHP）を起動して，メッセージを引き渡します。

非応答型のアプリケーションからの問い合わせ応答をする（UAP 共通定義（mcfmuap）の-c オプションの noansreply オペランドに yes を指定）場合，非応答型のアプリケーションから dc_mcf_reply 関数または CBLDCMCF('REPLY△△△')を発行して一方送信メッセージを送信できます。この一方送信メッセージは，出力通番を付けない一般の一方送信メッセージとして扱います。

出力キューまたは入力キューの割り当て先としてディスクキューを使用する場合，入力キューおよび出力キューに蓄えられた一方送信メッセージは，オンラインシステムが異常終了した場合などの再開始時に，引き継ぐことができます。

一方送信メッセージの受信と送信を，dc_mcf_send 関数および dc_mcf_receive 関数を使用する場合を例に，次の図に示します。

図 2-30　一方送信メッセージの送信と受信



1. UAP は一方送信メッセージの送信を要求します。
2. TP1/NET/TCP/IP は一方送信メッセージを送信します。
3. TP1/NET/TCP/IP は送信完了通知イベント（SCMPEVT）を MHP に通知します。
4. 出力キューにある一方送信メッセージのデータを消去します。
5. 相手システムからメッセージを受信します。
6. TP1/NET/TCP/IP は相手システムからのメッセージを入力キューに書き込み，MHP を起動します。
7. MHP はメッセージを受け取ります。

メッセージを送受信するときのデータの流れ，およびジャーナルの取得タイミングについては，「付録 D メッセージ送受信の処理の流れ」を参照してください。

なお，TP1/NET/TCP/IP の通信プロセスが，一方送信メッセージの送信完了を認識する前に運用コマンド（mcftdctcn）を入力，または API（dc_mcf_tdctcn 関数もしくは CBLDCMCF('TDCTCN△△')）を発行した場合，TP1/NET/TCP/IP はコネクションを解放し，KFCA10608-W を出力することがあります。このとき，論理端末は閉塞され，状態通知イベント（CCLSEVT）を通知します。

2.3.2　同期型メッセージ

TP1/NET/TCP/IP ではトランザクションと連動しないで，任意のタイミングで相手システムとメッセージの送受信を行う形態である，同期型メッセージの送受信ができます。同期型メッセージの送受信時に，TP1/NET/TCP/IP では，次のような機能が使えます。

- 受信メッセージの保留
  相手システムから受信したメッセージをいったん保留して，任意のタイミングで受信できます。
- 監視タイマの設定
  メッセージの送受信処理を無制限に待つことがないよう，待ち時間を指定できます。監視タイマがタイムアウトした場合，コネクションは切断されますが，コネクション定義（mcftalccn -w）の srtimout オペランドに yes を指定することで切断を抑止できます。

ここでは，同期型メッセージの送受信に関する機能について説明します。

なお，メッセージを送受信するときのデータの流れ，およびジャーナルの取得タイミングについては，「付録 D　メッセージ送受信の処理の流れ」を参照してください。

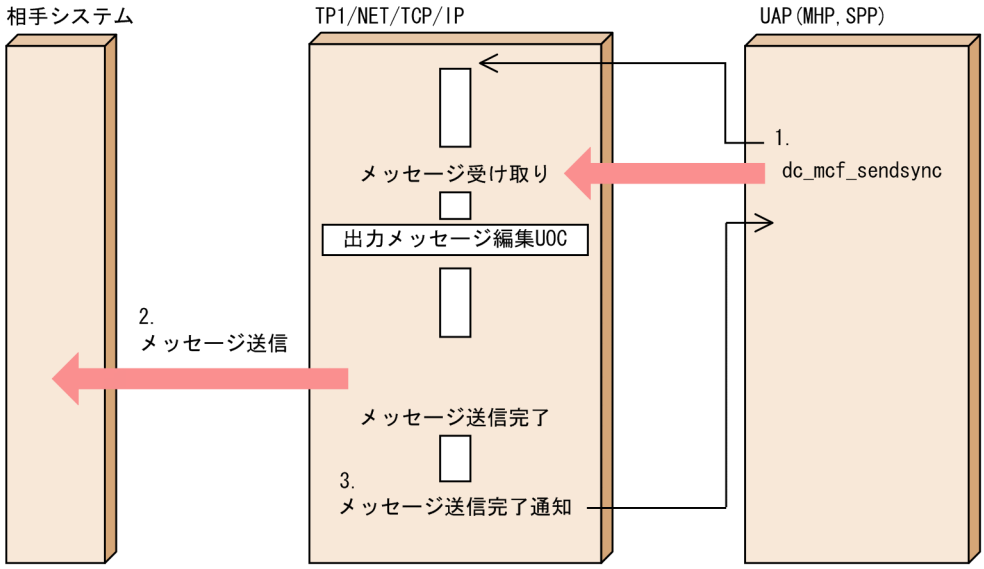
(1)　同期型メッセージの送信

この形態は，自システムの UAP とメッセージの送信完了を同期する場合に使用します。この場合の「送信完了」とは，TP1/NET/TCP/IP が行う TCP/IP の送信バッファへのメッセージ書き込みが完了したことを表します。したがって，API（dc_mcf_sendsync 関数または CBLDCMCF('SENDSYNC')）が UAP にリターンしたことは，送信メッセージが相手システムに到達したことを示しているわけではありません。

UAP は，API（dc_mcf_sendsync 関数または CBLDCMCF('SENDSYNC')）で TP1/NET/TCP/IP にメッセージの送信を要求します。TP1/NET/TCP/IP は受け取ったメッセージを編集したあと，相手システムに送信処理をします。送信処理が完了した時点で，UAP にリターンします。

dc_mcf_sendsync 関数を使用した場合の同期型メッセージの送信を次の図に示します。

図 2-31　同期型メッセージの送信



1. MHP，または SPP から同期型メッセージを送信します。
2. TP1/NET/TCP/IP は相手システムにメッセージを送信します。
3. 同期型メッセージ送信を要求した UAP に制御が渡されます。

(2)　同期型メッセージの受信

この形態は，自システムの UAP と相手システムからのメッセージ受信を同期する場合に使用します。

UAP は，API（dc_mcf_recvsync 関数または CBLDCMCF('RECVSYNC')）を発行して TP1/NET/TCP/IP にメッセージの受信を要求します。このとき，TP1/NET/TCP/IP が相手システムから受信したメッセージを保留しているかどうかによって，次のように処理が異なります。

受信メッセージを保留している場合

TP1/NET/TCP/IP はメッセージを編集して，UAP に引き渡します。

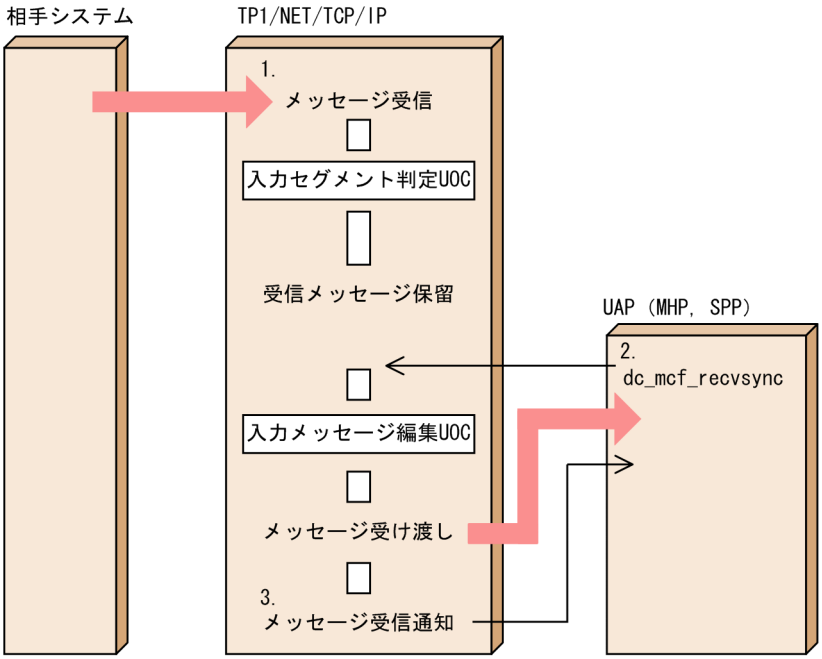
受信メッセージを保留していない場合

相手システムからのメッセージを受信するまで，TP1/NET/TCP/IP 内で待ち続けます。相手システムからのメッセージを受信した時点で，メッセージの受信を要求した UAP に渡します。

受信メッセージの保留については，「(4)　受信メッセージの保留」を参照してください。

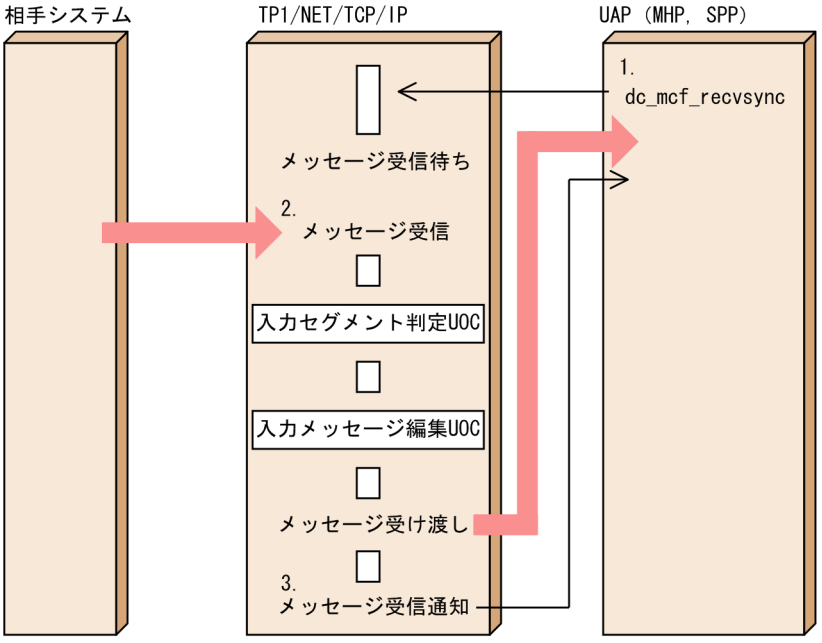
受信メッセージを保留している場合，および受信メッセージを保留していない場合の同期型メッセージの受信を以降の図に示します。

図 2-32　同期型メッセージの受信（受信メッセージを保留している場合）



1. 相手システムからメッセージを受信します。
2. MHP，または SPP から同期型メッセージの受信を要求します。
3. メッセージの同期受信を要求した UAP に制御が渡され，保留した受信メッセージを受信します。

図 2-33　同期型メッセージの受信（受信メッセージを保留していない場合）



1. MHP または，SPP から同期型メッセージの受信を要求します。
2. 相手システムからメッセージを受信します。
3. メッセージの同期受信を要求した UAP に制御が渡され，メッセージを受信します。

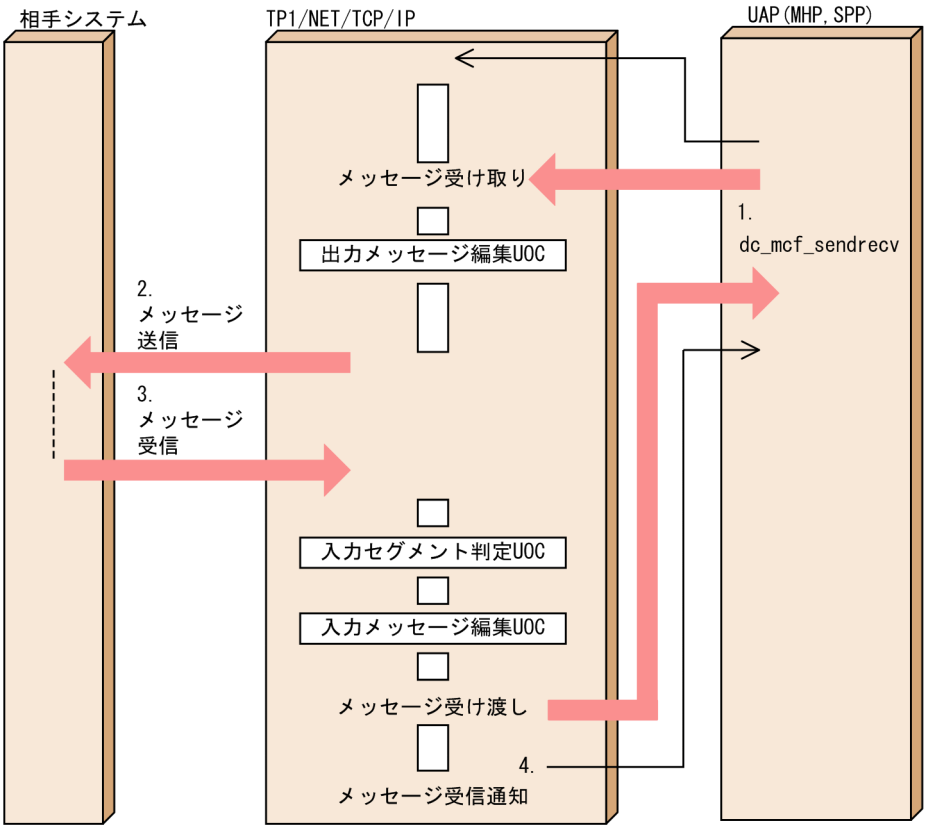
(3)　同期型メッセージの送受信

自システムからのメッセージ送信と，相手システムからの応答メッセージの受信を連続して行う形態です。この形態は，自システムの UAP と相手システムとの間でメッセージの同期合わせ，順序管理をする場合に使用します。

UAP は，API（dc_mcf_sendrecv 関数または CBLDCMCF('SENDRECV')）を発行して TP1/NET/TCP/IP にメッセージの送受信を要求します。TP1/NET/TCP/IP は受け取ったメッセージを編集したあと，相手システムに送信処理をします。送信処理が完了したあとは，UAP にリターンしないで，引き続き受信待ち状態となります。相手システムからのメッセージを受信すると，TP1/NET/TCP/IP はメッセージを編集して，UAP に引き渡します。

dc_mcf_sendrecv 関数を使用した場合の同期型メッセージの送受信を次の図に示します。

図 2-34　同期型メッセージの送受信

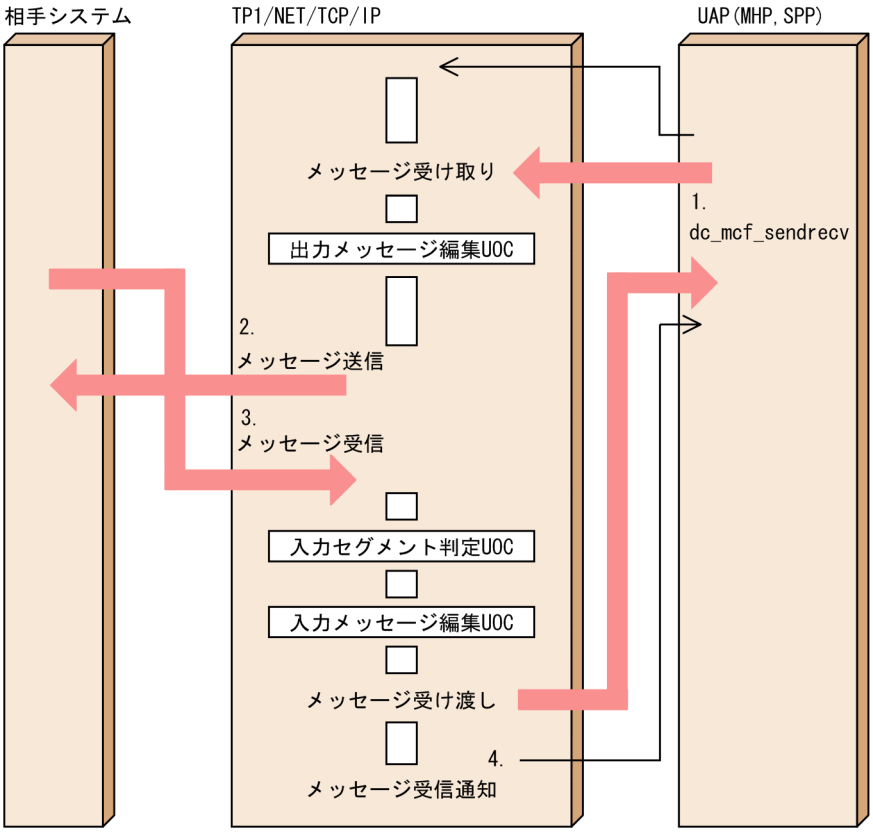


1. MHP，または SPP から同期型メッセージを送受信します。
2. TP1/NET/TCP/IP は相手システムにメッセージを送信します。
3. 相手システムからメッセージを受信します。
4. 同期型メッセージ送受信を要求した UAP に制御が渡され，メッセージを受信します。

同一コネクションでの自システムからのメッセージ送信と，相手システムからの一方送信メッセージがネットワーク上ですれ違うことがあります。メッセージがネットワーク上ですれ違った場合，TP1/NET/TCP/IP は相手システムからの一方送信メッセージを応答メッセージと見なして受信します。

dc_mcf_sendrecv 関数の使用時にメッセージがネットワーク上ですれ違った場合の例を次の図に示します。

図 2-35　ネットワーク上でメッセージがすれ違った場合の同期型メッセージの送受信



1. MHP，または SPP から同期型メッセージを送受信します。
2. TP1/NET/TCP/IP は相手システムにメッセージを送信します。
3. ネットワーク上ですれ違った相手システムからのメッセージを受信します。
4. 同期型メッセージ送受信を要求した UAP に制御が渡され，ネットワーク上ですれ違ったメッセージを受信します。

自システムからメッセージを送信した場合に，すでに TP1/NET/TCP/IP が相手システムからの一方送信メッセージを受信し，論理メッセージを組み立てていたときは，dc_mcf_sendrecv 関数または CBLDCMCF('SENDRECV')が，リターン値 DCMCFRTN_73003 またはステータスコード 73003 でエラーリターンします。

(4)　受信メッセージの保留

相手システムから受信したメッセージを，MHP にスケジュールしないで，TP1/NET/TCP/IP 内に保留しておくことができます。これによって，ユーザは任意のタイミングでメッセージを受信できます。

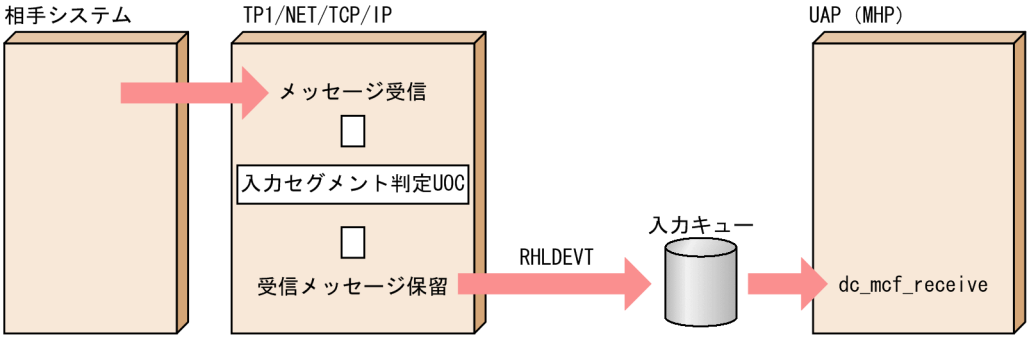
相手システムからメッセージを受信した場合，TP1/NET/TCP/IP 内に受信メッセージを保留して同期受信要求を待ち合わせるかどうかをコネクション定義（mcftalccn -u）の msghold オペランドで指定します。また，受信メッセージが無制限に保留されることを避けるため，コネクション定義（mcftalccn -u）

の holdlimit オペランドで，メッセージの最大保留数を指定できます。指定した最大保留数を超過した場合は，コネクションを切断します。

相手システムから受信したメッセージを保留すると，TP1/NET/TCP/IP は受信メッセージ保留通知イベント（RHLDEVT）を通知します。RHLDEVT を通知するには，MCF アプリケーション定義で定義しておく必要があります。

受信メッセージの保留について，dc_mcf_receive 関数を使用する場合を例に，次の図に示します。

図 2-36　受信メッセージの保留



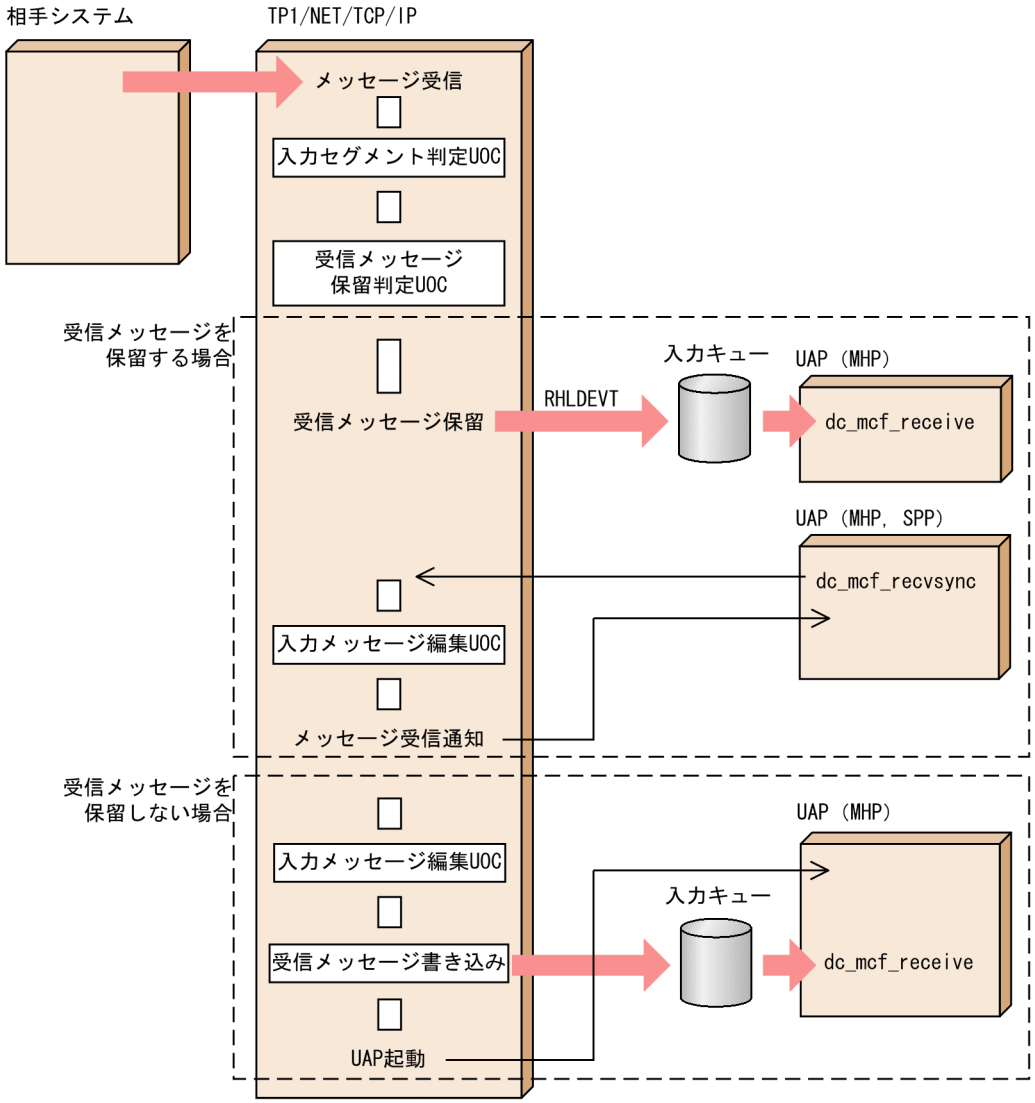
(a)　受信メッセージの保留判定

相手システムから受信したメッセージを保留するかどうか一意に決定できない場合は，コネクション定義（mcftalccn -u）の msghold オペランドに uoc を指定し，受信メッセージ保留判定 UOC を使用します。これによって，メッセージ単位に保留するかどうかを判定できます。

コネクション定義（mcftalccn -u）の msghold オペランドに uoc を指定した場合，TP1/NET/TCP/IP は相手システムからメッセージを受信したあと，受信メッセージ保留判定 UOC を呼び出します。受信メッセージ保留判定 UOC は，TP1/NET/TCP/IP から渡された受信メッセージの内容を基に，受信したメッセージを保留するかどうかを判定します。受信メッセージ保留判定 UOC については，「5.1.13　受信メッセージの保留判定」を参照してください。

受信メッセージの保留判定の流れを，dc_mcf_receive 関数および dc_mcf_recvsync 関数を使用する場合を例に，次の図に示します。

図 2-37　受信メッセージの保留判定の流れ



(b)　受信メッセージを保留する場合の注意事項

- TP1/NET/TCP/IP が受信メッセージを保留している間，dc_mcf_sendrecv 関数または CBLDCMCF('SENDRECV')は使用できません（dc_mcf_sendsync 関数または CBLDCMCF('SENDSYNC')は使用できます）。dc_mcf_sendrecv 関数または CBLDCMCF('SENDRECV')を使用した場合は，リターン値 DCMCFRTN_73003 またはステータスコード 73003 でエラーリターンします。
- TP1/NET/TCP/IP が dc_mcf_sendrecv 関数または dc_mcf_recvsync 関数（CBLDCMCF('RECVSYNC')または CBLDCMCF('SENDRECV')）の発行待ちの場合は，TP1/NET/TCP/IP は受信メッセージを保留しないで，仕掛り中の UAP に受信メッセージを引き渡します。このとき，受信メッセージ保留通知イベントの通知，および受信メッセージ保留判定 UOC の呼び出しは行いません。
- ネットワーク上のすれ違いなどにより，dc_mcf_sendrecv 関数または CBLDCMCF('SENDRECV')の送信処理仕掛り中（TCP/IP の送信バッファへのメッセージ書き込み中）に相手システムからメッセージを受信することがあります。この場合，TP1/NET/TCP/IP はコネクション定義の指定内容に従っ

て保留判定（RHLDEVT の通知，受信メッセージ保留判定 UOC の呼び出しを含む）を行います。保留した受信メッセージは，dc_mcf_sendrecv 関数または CBLDCMCF('SENDRECV')の送信処理完了後，応答メッセージと見なして UAP に引き渡されます。dc_mcf_sendrecv 関数または CBLDCMCF('SENDRECV')から渡された応答メッセージが意図しないメッセージであった場合は，受信したメッセージを破棄して dc_mcf_recvsync 関数または CBLDCMCF('RECVSYNC')を再度発行するような処置を行ってください。

(5)　コネクションの切断抑止

通常，同期型メッセージの送受信関数の発行時に指定した監視タイマがタイムアウトすると，コネクションは切断されます。しかし，システム定義の指定によっては，コネクションの切断を抑止することができます。

ここでは，コネクションの切断抑止の概要，およびタイムアウト発生時の動作について説明します。

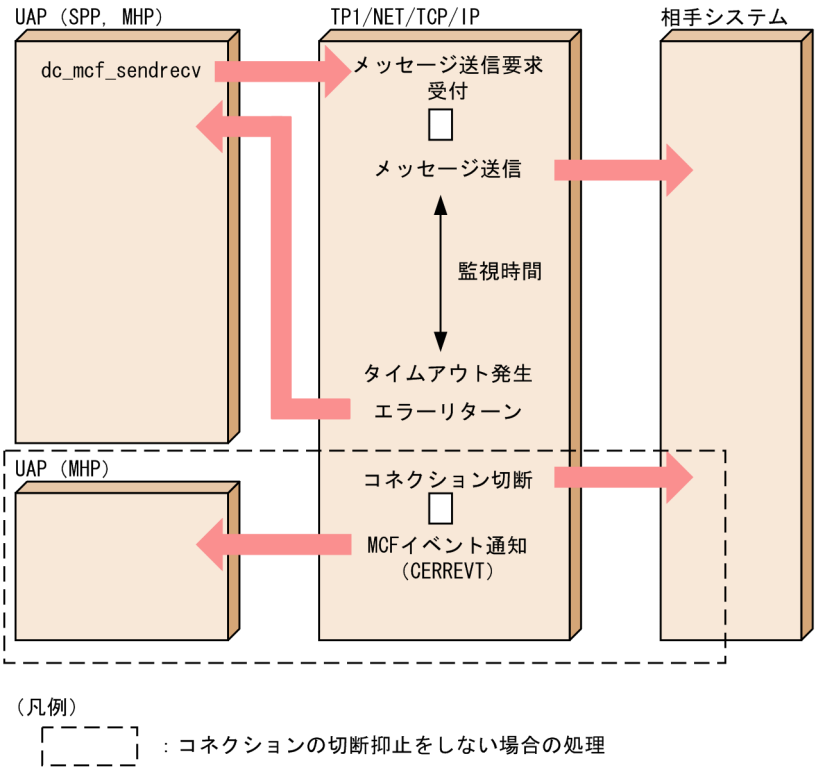
(a)　コネクションの切断抑止の概要

コネクションの切断抑止では，送受信関数がタイムアウトした場合のコネクション切断，および CERREVT の起動を行いません。コネクションの切断を抑止するかどうかは，コネクション定義（mcftalccn -w）の srtimout オペランドで指定します。

コネクションの切断を抑止する場合は，入力セグメント判定 UOC での後続メッセージ監視を行わないでください。

dc_mcf_sendrecv 関数がタイムアウトした場合の，コネクションの切断抑止の概要を，次の図に示します。

図 2-38　コネクションの切断抑止の概要



(b) タイムアウト発生時の動作

メッセージの送受信関数のタイムアウト発生時の動作を，次の表に示します。

表 2-5 メッセージの送受信関数タイムアウト発生時の動作

項番	項目	コネクション定義（mcftalccn -w）の指定による動作	
		srtimout オペランドに yes を指定 （コネクションの切断を抑止する）	srtimout オペランドに no を指定 （コネクションの切断を抑止しない）
1	コネクション	コネクションを切断しません。	コネクションを切断します。
2	CERREVT	起動しません。	起動します。 次に示す内容を通知します。 • 障害要因コード：0x30 • 理由コード 1： DCMTCP_RSN1_MCF • 理由コード 2： DCMTCP_RSN2_SRTO
3	リターン値	DCMCFRTN_73005（監視タイマタイムアウト）が返されます。	
4	ステータスコード	73005（監視タイマタイムアウト）が返されます。	

2.3.3 同一論理端末上での send 関数の併用に関する注意事項

同一論理端末上での，一方送信によるメッセージ送信（dc_mcf_send 関数または CBLDCMCF('SEND△△△')）と，次に示すメッセージ送受信との併用は避けてください。

- 同期型メッセージの送受信（dc_mcf_sendrecv 関数または CBLDCMCF('SENDRECV')）
- 同期型メッセージの送信（dc_mcf_sendsync 関数または CBLDCMCF('SENDSYNC')）

併用する場合は，UAP が行うメッセージの送信要求の優先順位に注意が必要です。MCF は，次の優先順位に従って，UAP の送信要求を TP1/NET/TCP/IP にスケジュールします。

1. 同期型メッセージの送受信および同期型メッセージの送信
2. 一方送信メッセージの送信（優先）
3. 一方送信メッセージの送信（一般）

併用する場合は，同期送受信，または同期送信によって送信されたメッセージが，一方送信によって送信されたメッセージを追い越したときでも，この優先順位に矛盾しない運用をしてください。優先順位に従った運用をしなかった場合に，同一論理端末上で一方送信によるメッセージ送信，または同期送受信もしくは同期送信によるメッセージ送信を併用すると，送信メッセージの追い越しが発生するおそれがあります。

2.3.4　問い合わせメッセージと応答メッセージ（問い合わせ応答形態）

相手システムから問い合わせメッセージを受信したあと，自システムから応答メッセージを送信する形態です。UAP（MHP）のトランザクションの決着と連動してメッセージを送受信します。非トランザクションの MHP の場合は，サービスの終了と連動してメッセージを送受信します。論理端末の端末タイプは any，アプリケーションの型は応答型です。

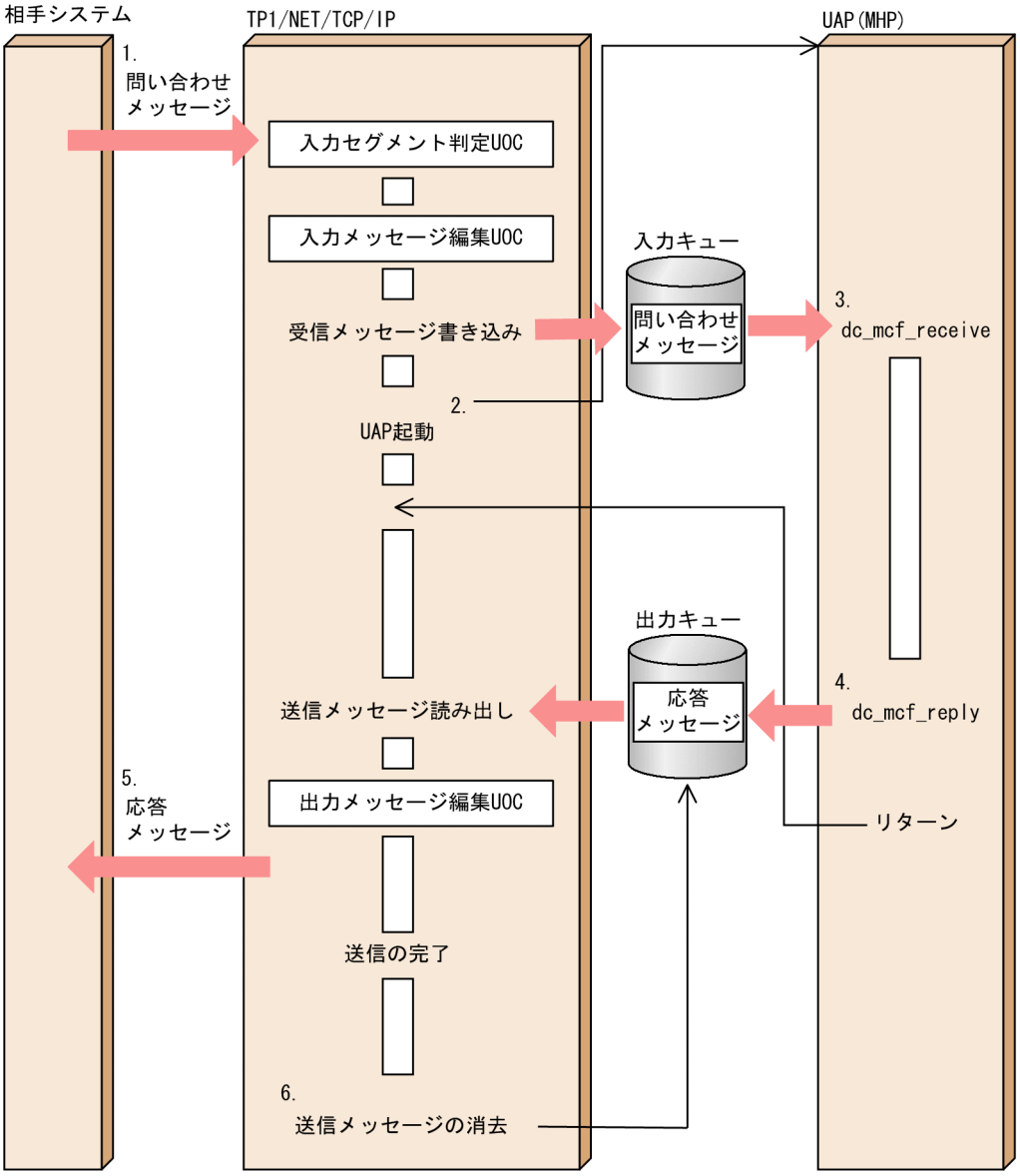
相手システムからのメッセージを受信すると，TP1/NET/TCP/IP は入力メッセージ編集 UOC で編集されたメッセージを入力キューに登録します。同時に，応答型のアプリケーションに対応する UAP を起動して，メッセージを引き渡します。

UAP が API（dc_mcf_reply 関数または CBLDCMCF('REPLY△△△')）で送信要求すると，メッセージが出力キューに登録されます。TP1/NET/TCP/IP は出力キューからメッセージを読み出し，出力メッセージ編集 UOC で編集されたメッセージを相手システムに送信します。応答メッセージを TCP/IP の送信バッファに書き込み完了した時点で，送信完了として TP1/NET/TCP/IP は出力キューにある送信済みメッセージのデータを消去します。

出力キューまたは入力キューの割り当て先の指定に関係なく，入力キューに蓄えられた問い合わせメッセージおよび出力キューに蓄えられた応答メッセージは，オンラインシステムが異常終了した場合などの再開始時に引き継がれません。

問い合わせメッセージの受信と応答メッセージの送信を，dc_mcf_receive 関数および dc_mcf_reply 関数を使用する場合を例に，次の図に示します。

図 2-39　問い合わせメッセージの受信と応答メッセージの送信



1. 相手システムからメッセージを受信します。
2. TP1/NET/TCP/IP は相手システムからのメッセージを入力キューに書き込み，MHP を起動します。
3. MHP はメッセージを受け取ります。
4. MHP は応答メッセージの送信を要求します。
5. TP1/NET/TCP/IP は応答メッセージを送信します。
6. 出力キューにある一方送信メッセージのデータを消去します。

以降，問い合わせ応答形態の詳細を説明します。

(1) 問い合わせ応答の開始と終了

問い合わせ応答は，相手システムから問い合わせメッセージを受信し，応答型のアプリケーションを起動することで開始します。そして，UAP から応答メッセージを受け付け，相手システムへの応答メッセージの送信が完了すると問い合わせ応答を終了します。また，障害などによるコネクションの解放および論理端末の閉塞となった場合にも問い合わせ応答を終了します。

問い合わせ応答の開始から問い合わせ応答の終了までを**問い合わせ応答中**と呼びます。

(2) 問い合わせ応答中のアプリケーション起動

問い合わせ応答中にアプリケーション起動要求をする場合，非応答型または応答型のアプリケーション名を指定できます。ただし，応答型のアプリケーションを起動できるのは一つのサービスで 1 回だけです。

応答型のアプリケーションを起動した場合，そのサービスでは応答メッセージを送信できません。起動先の応答型アプリケーションで応答メッセージを送信してください。

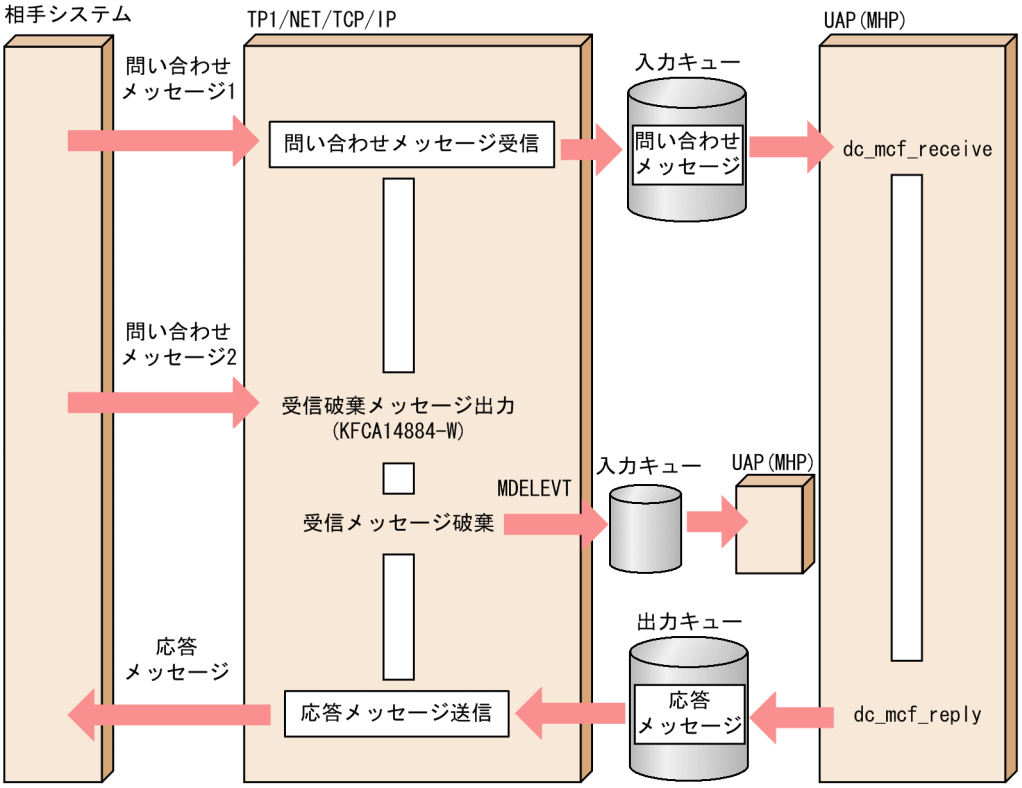
使用方法の詳細については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

(3) 問い合わせ応答中の論理端末へのメッセージ受信

問い合わせ応答中の論理端末が相手システムからメッセージを受信した場合，TP1/NET/TCP/IP はメッセージログ（KFCA14884-W）を出力し，MDELEVT を起動します。

UAP 応答待ちの論理端末へのメッセージ受信を次の図に示します。

図 2-40　UAP 応答待ちの論理端末へのメッセージ受信



(4)　問い合わせ応答中の論理端末へのメッセージ送信

問い合わせ応答中の論理端末に対してメッセージを送信した場合，メッセージの種類によって動作が異なります。

(a)　一方送信メッセージの場合

一方送信メッセージは出力キューに登録され，API は正常にリターンします。登録された一方送信メッセージは，TP1/NET/TCP/IP の取り出しを待ちます。応答メッセージの送信完了後に出力キューから読み出し，相手システムに送信します。

(b)　同期型メッセージの場合

API がリターン値 DCMCFRTN_72001 またはステータスコード 72001 でエラーリターンします。

(5)　閉塞中の論理端末への問い合わせメッセージ受信

閉塞中の論理端末が相手システムからメッセージを受信した場合，TP1/NET/TCP/IP は UAP を起動しますが，サービスが終了しても出力キューから応答メッセージを読み出さないで，問い合わせ応答中のままになります。相手システムに応答メッセージを送信する場合，運用コマンド（mcftactle）を入力し，論理端末を閉塞解除してください。

なお，出力キューに滞留した応答メッセージを運用コマンド（mcftdlqle）などで削除した場合，論理端末を閉塞解除しても問い合わせ応答は終了しません。運用コマンド（mcftdctcn）を入力し，コネクションを解放してください。

2.3.5　問い合わせメッセージと応答メッセージ（継続問い合わせ応答形態）

相手システムからの問い合わせメッセージの受信と，自システムからの応答メッセージの送信を繰り返す形態です。UAP（MHP）のトランザクションの決着と連動してメッセージを送受信します。非トランザクションの MHP の場合は，サービスの終了と連動してメッセージを送受信します。論理端末の端末タイプは any，アプリケーションの型は継続問い合わせ応答型です。

継続問い合わせ応答は継続問い合わせ応答型のアプリケーションの起動によって開始され，UAP から API（dc_mcf_contend 関数または CBLDCMCF('CONTEND△')）を発行したり，運用コマンド（mcftendct）を入力したりすることで終了します。

相手システムからのメッセージを受信すると，TP1/NET/TCP/IP は入力メッセージ編集 UOC で編集されたメッセージを入力キューに登録します。同時に，継続問い合わせ応答型のアプリケーションに対応する UAP を起動して，メッセージを引き渡します。

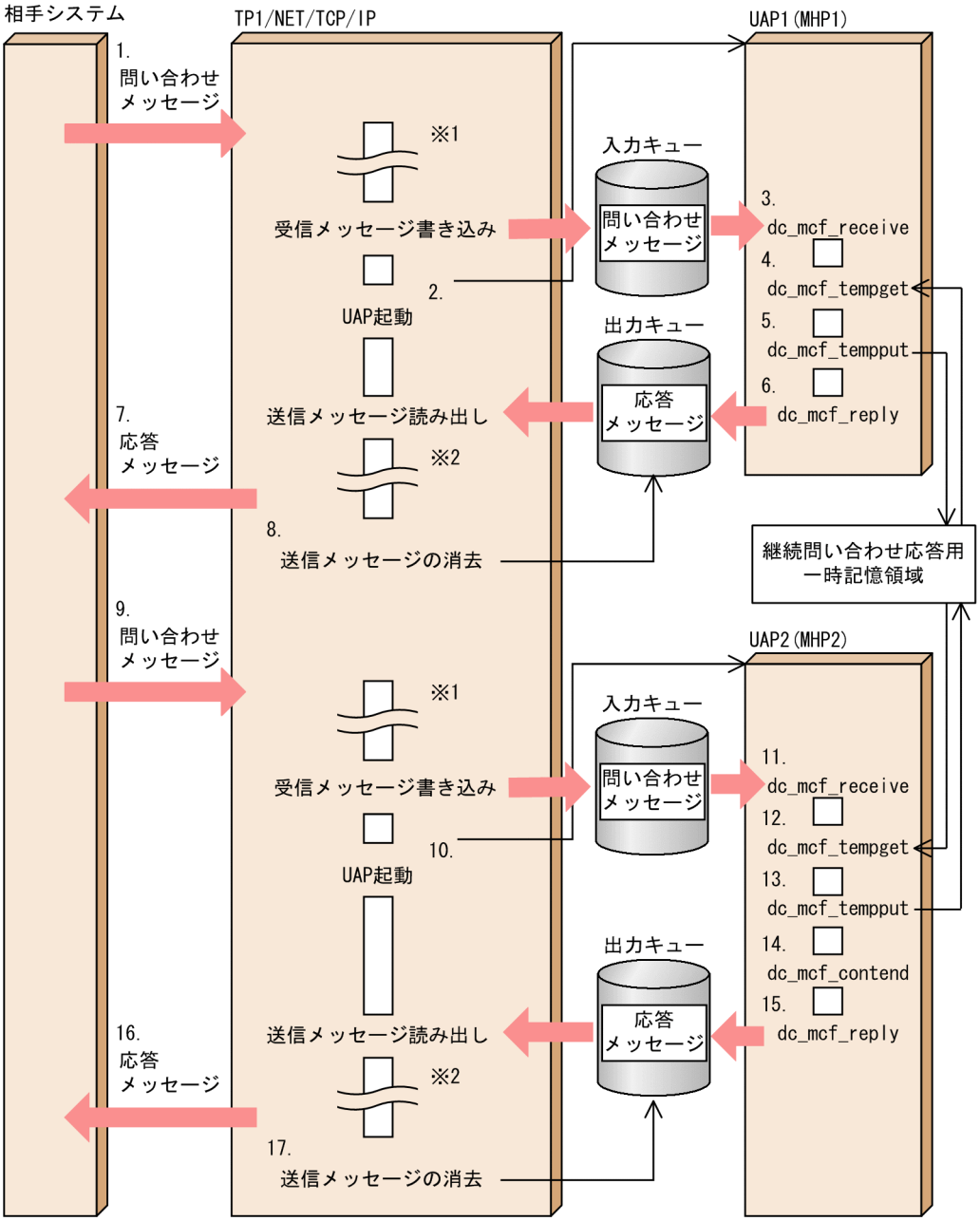
UAP が API（dc_mcf_reply 関数または CBLDCMCF('REPLY△△△')）で送信要求すると，メッセージが出力キューに登録されます。継続問い合わせ応答中の UAP は，dc_mcf_reply 関数または CBLDCMCF('REPLY△△△')発行時に次に起動するアプリケーションを指定できます。また，継続問い合わせ応答用一時記憶領域を使って次に起動するアプリケーションにデータを引き継ぐことができます。

TP1/NET/TCP/IP は出力キューからメッセージを読み出し，出力メッセージ編集 UOC で編集されたメッセージを相手システムに送信します。応答メッセージを TCP/IP の送信バッファに書き込み完了した時点で，送信完了として TP1/NET/TCP/IP は出力キューにある送信済みメッセージのデータを消去します。

出力キューまたは入力キューの割り当て先の指定に関係なく，入力キューに蓄えられた問い合わせメッセージおよび出力キューに蓄えられた応答メッセージはオンラインシステムが異常終了した場合などの再開始時に引き継がれません。

継続問い合わせ応答を，dc_mcf_receive 関数および dc_mcf_reply 関数を使用する場合を例に，次の図に示します。

図 2-41　継続問い合わせ応答



注※1　問い合わせメッセージの受信から受信メッセージ書き込みまでは，応答型の問い合わせメッセージの受信と同じです。
注※2　送信メッセージ読み出しから応答メッセージの送信までは，応答型の応答メッセージの送信と同じです。

1. 相手システムからメッセージを受信します。
2. TP1/NET/TCP/IP は相手システムからのメッセージを入力キューに書き込み，MHP1 を起動します。
3. MHP1 は問い合わせメッセージを受け取ります。
4. MHP1 は一時記憶データの受け取り要求を行い，一時記憶データを受け取ります。
5. MHP1 は一時記憶データの更新要求を行い，一時記憶データを更新します。
6. MHP1 は応答メッセージの送信を要求します。

7. TP1/NET/TCP/IP は応答メッセージを送信します。
8. 出力キューにある応答メッセージのデータを消去します。
9. 相手システムからメッセージを受信します。
10. TP1/NET/TCP/IP は相手システムからのメッセージを入力キューに書き込み，MHP2 を起動します。
11. MHP2 は問い合わせメッセージを受け取ります。
12. MHP2 は一時記憶データの受け取り要求を行い，一時記憶データを受け取ります。
13. MHP2 は一時記憶データの更新要求を行い，一時記憶データを更新します。
14. MHP2 は継続問い合わせ応答の終了を要求します。
15. MHP2 は応答メッセージの送信を要求します。
16. TP1/NET/TCP/IP は応答メッセージを送信します。
17. 出力キューにある応答メッセージのデータを消去します。

以降，継続問い合わせ応答形態の詳細を説明します。

(1)　継続問い合わせ応答の開始

継続問い合わせ応答は，継続問い合わせ応答中でないときに相手システムから問い合わせメッセージを受信し，継続問い合わせ応答型のアプリケーションを起動することで開始します。

問い合わせ応答と異なり，継続問い合わせ応答では相手システムへの応答メッセージの送信が完了しても継続問い合わせ応答を終了しません。

継続問い合わせ応答の開始から継続問い合わせ応答の終了までを**継続問い合わせ応答中**と呼びます。また，継続問い合わせ応答中に UAP を起動してから送信メッセージを読み出すまでを**継続問い合わせ応答の UAP 実行中**と呼びます。

(2)　次起動アプリケーションの予約

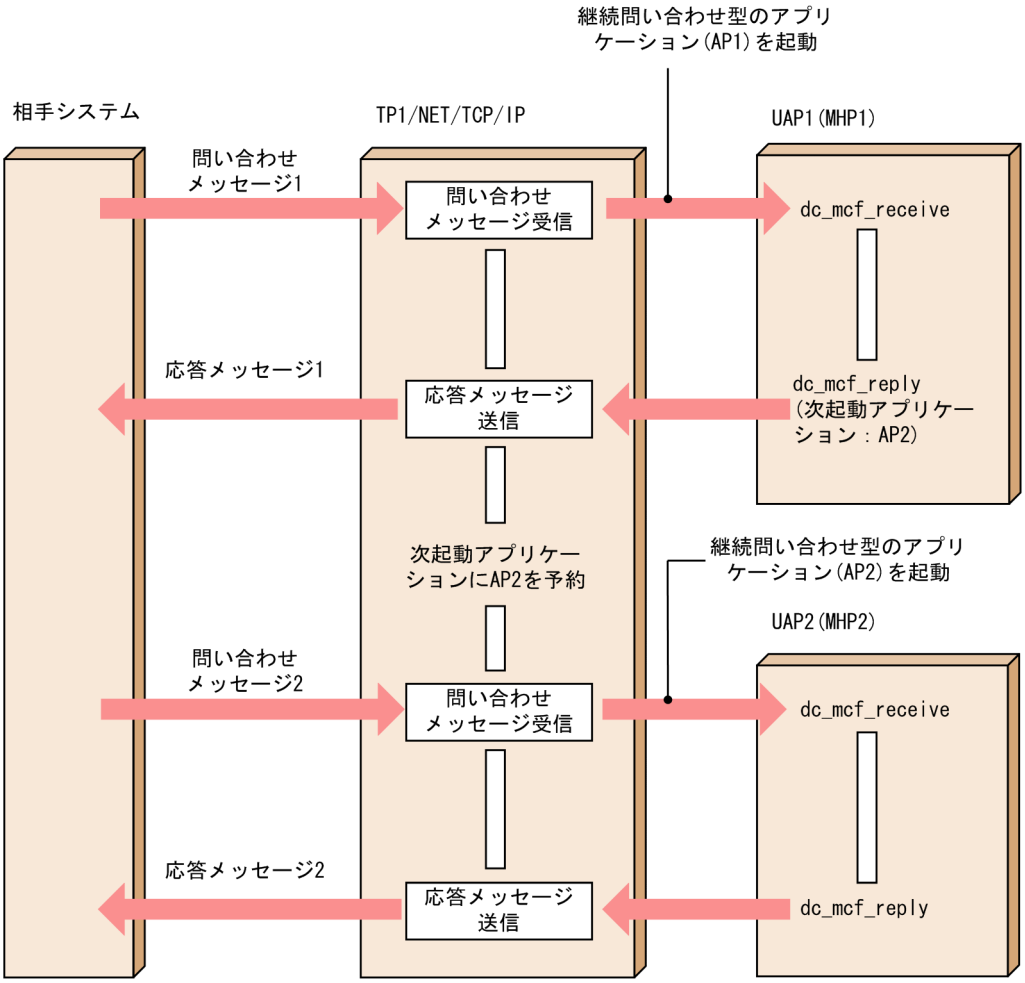
応答メッセージ送信要求時，次に起動する継続問い合わせ応答型のアプリケーション（次起動アプリケーション）を予約できます。

次起動アプリケーションの指定を省略した場合，実行中のアプリケーション名が指定されたものとします。ただし，エラーイベントで次起動アプリケーションの指定を省略した場合は，継続問い合わせ応答を終了します。詳細については，「2.3.5(6)　エラーイベント」を参照してください。

次起動アプリケーションの予約を次の図に示します。

使用方法の詳細については，「3.　C 言語のライブラリ関数」または「4.　COBOL-UAP 作成用プログラムインタフェース」を参照してください。

図 2-42　次起動アプリケーションの予約



(3)　継続問い合わせ応答中のアプリケーション起動

継続問い合わせ応答中にアプリケーション起動要求をする場合，非応答型または継続問い合わせ応答型のアプリケーション名を指定できます。ただし，継続問い合わせ応答型のアプリケーションを起動できるのは一つのサービスで 1 回だけです。

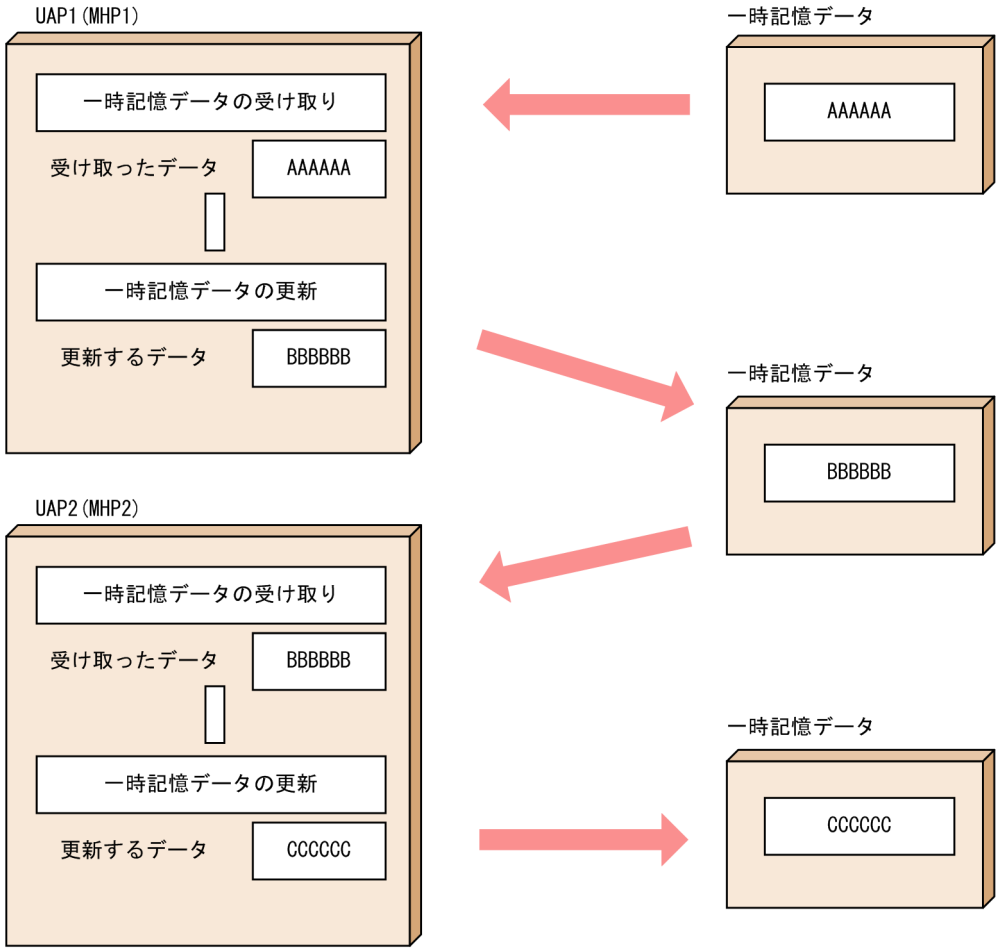
継続問い合わせ応答型のアプリケーションを起動した場合，そのサービスでは応答メッセージを送信できません。また，継続問い合わせ応答の終了要求もできません。起動先の継続問い合わせ応答型アプリケーションで応答メッセージを送信したり，継続問い合わせ応答の終了要求をしてください。

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

(4)　継続問い合わせ応答用一時記憶データ

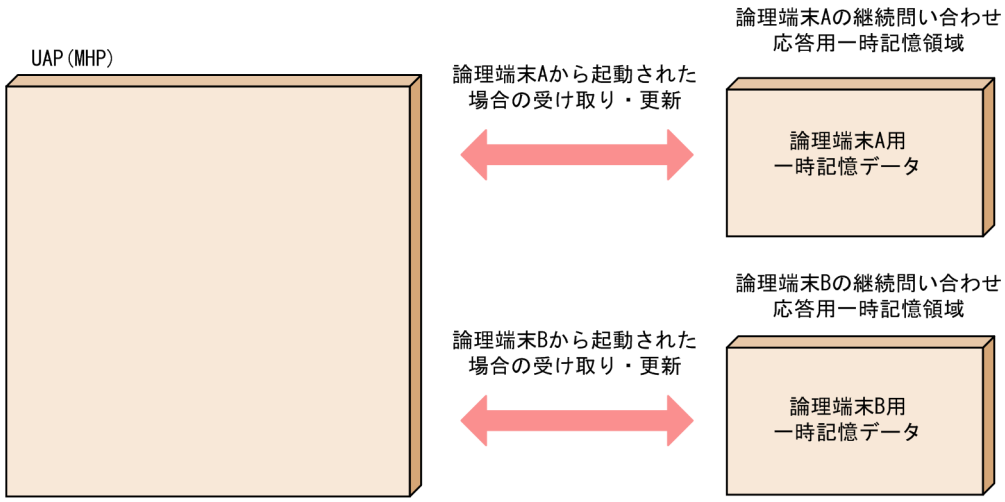
継続問い合わせ応答用一時記憶データ（以降，一時記憶データと呼びます）を使用して，次起動アプリケーションへデータの引き継ぎができます。データの引き継ぎを，次の図に示します。

図 2-43　一時記憶データの引き継ぎ



一時記憶データを格納する継続問い合わせ応答用一時記憶領域は論理端末ごとに確保するため，複数の論理端末で同時に一つのアプリケーションを使用して継続問い合わせ応答ができます。論理端末と一時記憶データの関係を次の図に示します。

図 2-44　論理端末と一時記憶データの関係



(a)　継続問い合わせ応答用一時記憶領域の確保

一時記憶データを格納する継続問い合わせ応答用一時記憶領域は，更新用領域および回復用領域の 2 面を共有メモリに確保します。

アプリケーション属性定義（mcfaalcap）でアプリケーションごとに継続問い合わせ応答用一時記憶領域のサイズを定義できます。

ただし，実際に確保する継続問い合わせ応答用一時記憶領域のサイズは，継続問い合わせ応答を開始したあと最初に一時記憶データの更新要求をしたアプリケーションのアプリケーション属性定義（mcfaalcap -n）の tempsize オペランドの指定値となります。確保済みの継続問い合わせ応答用一時記憶領域のサイズが，一時記憶データの更新要求を行う場合のデータ長よりも小さいとき，実行中のアプリケーションのアプリケーション属性定義（mcfaalcap -n）の tempsize オペランドの指定値のサイズで，継続問い合わせ応答用一時記憶領域を再確保します。したがって，継続問い合わせ応答で使用する各アプリケーションの更新する一時記憶データの領域長が異なる場合，最初に起動するアプリケーションのアプリケーション属性定義（mcfaalcap -n）の tempsize オペランドにデータ長の最大値を指定すると，途中で再確保をしないため，性能が向上します。

(b)　一時記憶データの受け渡し

一時記憶データの受け渡しには「受け取り要求」と「更新要求」があります。

- 一時記憶データの受け取り要求

  指定したデータ長の一時記憶データを受け取れます。初期値は，アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドで指定したサイズ分の$(00)_{16}$が渡されます。指定した受け取り領域長が一時記憶データより小さい場合，指定した領域長の長さだけ受け取り，残りは切り捨てられます。
- 一時記憶データの更新要求

  指定したデータ長のデータを用いて一時記憶データを更新します。最大でアプリケーション属性定義（mcfaalcap -n）の tempsize オペランドで指定したデータ長の更新ができます。

使用方法の詳細については，「3.　C 言語のライブラリ関数」または「4.　COBOL-UAP 作成用プログラムインタフェース」を参照してください。

(c)　継続問い合わせ応答用一時記憶領域の解放

継続問い合わせ応答を終了すると継続問い合わせ応答用一時記憶領域を解放します。

(d)　障害発生時の一時記憶データ

障害発生時の一時記憶データの扱いについて次に示します。

- OpenTP1 システム障害による全面回復の場合

  一時記憶データは失われます。
- UAP 障害による部分回復の場合

異常終了した UAP での一時記憶データの更新要求が取り消され，異常終了した UAP が一時記憶データの受け取り要求を行ったときの状態に戻ります。

(5) 継続問い合わせ応答の終了

継続問い合わせ応答は次のどれかで終了します。

(a) MHP からの終了要求（dc_mcf_contend 関数または CBLDCMCF('CONTEND△')）

MHP から API（dc_mcf_contend 関数または CBLDCMCF('CONTEND△')）を発行して継続問い合わせ応答の終了を要求します。

ただし，次に示す条件に該当する場合はエラーリターンします。

- 次起動アプリケーションを予約して，応答メッセージの送信を要求している。
- 継続問い合わせ応答型のアプリケーションの起動を要求している。

使用方法の詳細については，「3. C 言語のライブラリ関数」または「4. COBOL-UAP 作成用プログラムインタフェース」を参照してください。

(b) 運用コマンドによる終了要求（mcftendct）

運用コマンド（mcftendct）で論理端末名称を指定して，その論理端末に対する継続問い合わせ応答を強制終了できます。また，継続問い合わせ応答を行っていない UAP からも運用コマンドを発行できます。詳細については，「7. 運用コマンド」を参照してください。

継続問い合わせ応答の UAP 実行中に継続問い合わせ応答を即時強制終了する場合，-f オプションの指定が必要です。

なお，-f オプションを指定した mcftendct コマンドの実行後に MHP が次に示す処理をしたとき，MHP が異常終了します。このとき，エラーイベントは起動しません。

- TP1/Message Control 以外のリソースマネジャにもアクセスする MHP のサービス終了時
- 継続問い合わせ用一時記憶領域アクセス時（dc_mcf_tempget 関数，dc_mcf_tempput 関数発行時）
- ロールバック要求時（dc_mcf_rollback 関数発行時（action：DCMCFRTRY，または DCMCFNRTN の場合））

(c) コネクション解放および論理端末の閉塞

障害などによるコネクションの解放および論理端末の閉塞となった場合に，継続問い合わせ応答を終了します。

相手からのコネクション解放，または運用コマンド（mcftdctcn）や障害によってコネクションを解放する場合，継続問い合わせ応答を処理している MHP が実行中であれば MHP のサービス完了を待ち合わせます。MHP のサービスが完了した後に継続問い合わせ応答を終了し，コネクションは解放状態となります。

なお，継続問い合わせ応答中の論理端末は，運用コマンド（mcftdctle）による閉塞はできません。

(6) エラーイベント

MHP の処理中に障害が発生した場合，MCF イベント処理用 MHP にエラーイベントを通知します。継続問い合わせ応答中に障害が発生した場合，エラーイベントを継続問い合わせ応答型として起動します。

(a) エラーイベントを定義した場合

MCF イベント処理用 MHP で次のどちらかを行うことで継続問い合わせを続行できます。

- 次起動アプリケーションを指定して，応答メッセージの送信要求をする。
- 継続問い合わせ応答型のアプリケーションの起動要求をする。

MCF イベント処理用 MHP で継続問い合わせ応答型のアプリケーションの起動要求を行わない，かつ，次起動アプリケーションを指定しないで応答メッセージの送信要求をした場合，継続問い合わせ応答処理は終了します。

ただし，MCF イベント処理用 MHP で継続問い合わせ応答型のアプリケーションを起動し，起動先のアプリケーションで次に起動するアプリケーション名を指定しないで応答メッセージの送信要求をした場合，継続問い合わせ応答を終了することなく，MCF イベント処理用 MHP から起動した継続問い合わせ応答型のアプリケーションを次のメッセージ受信時に再び起動します。

(b) エラーイベントを定義しない場合

継続問い合わせ応答処理は終了します。

(7) 継続問い合わせ応答中の論理端末へのメッセージ受信

継続問い合わせ応答中の論理端末が相手システムからメッセージを受信した場合，応答型と同様に TP1/NET/TCP/IP はメッセージログ（KFCA14884-W）を出力し，MDELEVT を起動します。

(8) 継続問い合わせ応答中の論理端末へのメッセージ送信

継続問い合わせ応答中の論理端末に対してメッセージを送信した場合，メッセージの種類によって動作が異なります。

(a) 一方送信メッセージの場合

一方送信メッセージは出力キューに登録され，API は正常にリターンします。登録された一方送信メッセージは，TP1/NET/TCP/IP の取り出しを待ちます。継続問い合わせ応答の終了後に出力キューから読み出し，相手システムに送信します。

(b) 同期型メッセージの場合

API がリターン値 DCMCFRTN_72001 またはステータスコード 72001 でエラーリターンします。

(9)　閉塞中の論理端末への問い合わせメッセージ受信

閉塞中の論理端末が相手システムからメッセージを受信した場合，TP1/NET/TCP/IP は UAP を起動しますが，サービスが終了しても出力キューから応答メッセージを読み出さないで，継続問い合わせ応答の UAP 実行中のままになります。相手システムに応答メッセージを送信する場合，運用コマンド（mcftactle）を入力し，論理端末を閉塞解除してください。

なお，出力キューに滞留した応答メッセージを運用コマンド（mcftdlqle）などで削除した場合，論理端末を閉塞解除したり，コネクションを解放しても継続問い合わせ応答は終了しません。-f オプションを指定した mcftendct コマンドを入力し，継続問い合わせ応答を即時強制終了してください。

2.3.6　メッセージとセグメントの関係

UAP の 1 回の関数で入出力できるメッセージの分割単位を，**セグメント**といいます。TP1/NET/TCP/IP を使用して相手システムと送受信する場合，一つのメッセージは一つのセグメントで構成されます。つまり，1 回の関数の呼び出しで，一つのメッセージを送受信できます。ただし，MCF イベント（MCF が通知する特殊なメッセージ）は，二つのセグメントで構成されることがあります。これを受信する場合は，メッセージを受信する関数を 2 回呼び出してください。MCF イベントのセグメント構成については，「5.2.2　MCF イベント通知時のセグメント構成」を参照してください。

メッセージ送受信の関数で処理するセグメントの先頭には，MCF で使用されるヘッダ領域があります。C 言語および COBOL 言語のメッセージ送受信関数を使用している場合，ヘッダ領域の長さは，バッファ形式 1 とバッファ形式 2 で異なります。バッファ形式を省略した場合，バッファ形式 1 が仮定されます。

2.3.7　メッセージの分割と組み立て

TP1/NET/TCP/IP がメッセージを受け取る場合，相手システムから送信されたメッセージがネットワーク内で分割されたり，前後に送信されたメッセージと結合してしまうことがあります。また，メッセージを受け取る側のシステムでは，送信されたメッセージのセグメントがどこまでで終わっているのかを正確に認識できません。ただし，メッセージの送信順序は保証されます。

分割，または結合されたメッセージは，受け取る側のシステムが再度分割，組み立てをすることで，正常な論理メッセージとなります。メッセージの分割，および組み立て時には，該当するメッセージが論理メッセージのセグメントの終わりかどうかを判定する必要があります。TP1/NET/TCP/IP では，受信メッセージ組み立て機能，または入力セグメント判定 UOC で入力セグメントの終わりを判定し，メッセージの分割・組み立てをします。受信メッセージ組み立て機能，または入力セグメント判定 UOC を使用しない場合は，TP1/NET/TCP/IP は TCP/IP ソケットの受信バッファから受け取ったメッセージをそのまま UAP に通知します。この場合，UAP でメッセージの分割・組み立てを行ってください。

(1) 受信メッセージの組み立て機能

コネクション定義（mcftalccn -u）の masm オペランドに yes を指定した場合，受信メッセージの組み立て機能を使用できます。

受信メッセージ組み立て機能を使用する場合，メッセージ送信時に，TP1/NET/TCP/IP はメッセージ中の先頭に自動的にメッセージ長エリアを付けます。そのエリアにバイト順序がビッグエンディアン※の整数型のメッセージ長を設定し，特定のフォーマットにして送信します。このとき，メッセージが分割されるかどうかは，システム（ソケット）の送受信バッファ長に依存します。

メッセージ受信時には，メッセージ中に設定したメッセージ長エリアのバイト順序がビッグエンディアンの整数型であることを前提に論理メッセージを組み立てます。このため，相手システムはメッセージ長エリアにバイト順序がビッグエンディアンの整数型のメッセージ長を設定する必要があります。

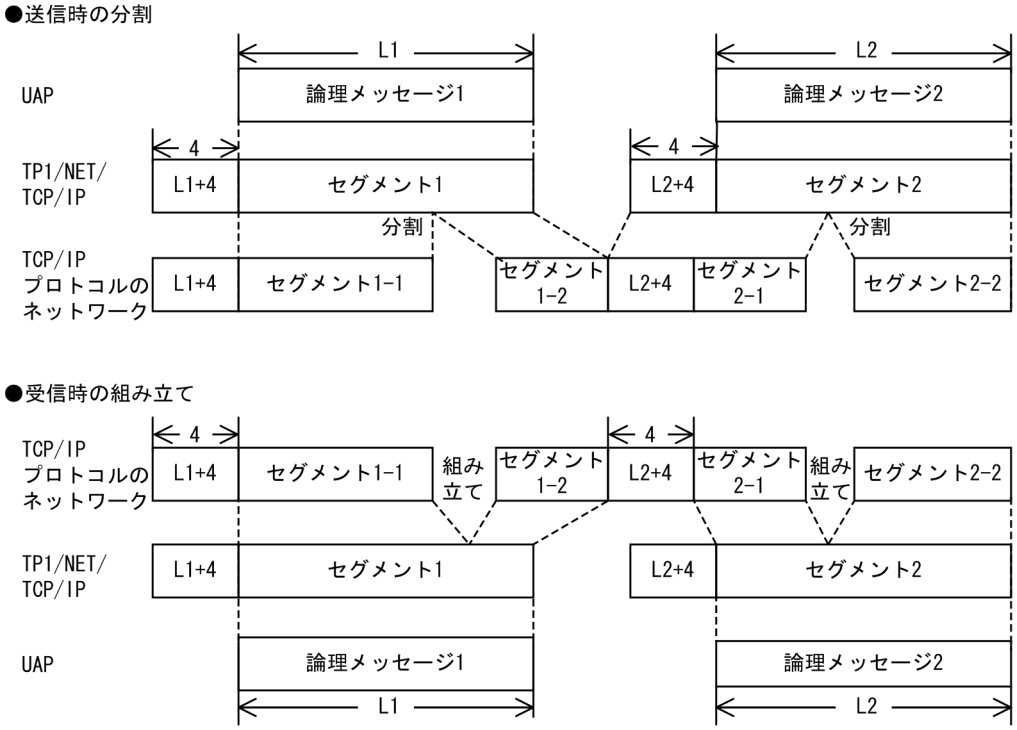
注※

ビッグエンディアンとはバイトを低い方から順番に並べて，最下位のビットを最上位のバイトに置く方式のことです。

バイト0								バイト1								バイト2								バイト3							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

メッセージ受信時，メッセージ中の TP1/NET/TCP/IP が送信時に付けたメッセージ長エリアを自動的に削除します。受信メッセージ組み立て機能での分割・組み立ての概要を次の図に示します。

図 2-45　受信メッセージ組み立て機能での分割・組み立て



（凡例）

L1，L2：メッセージ長

メッセージ長エリアにバッファグループ定義（mcftbuf -g）の length オペランドで定義した受信バッファのサイズを超える値が設定されていた場合，TP1/NET/TCP/IP は受信バッファオーバフローと見なし，コネクションを解放します。

(2) 入力セグメント判定 UOC

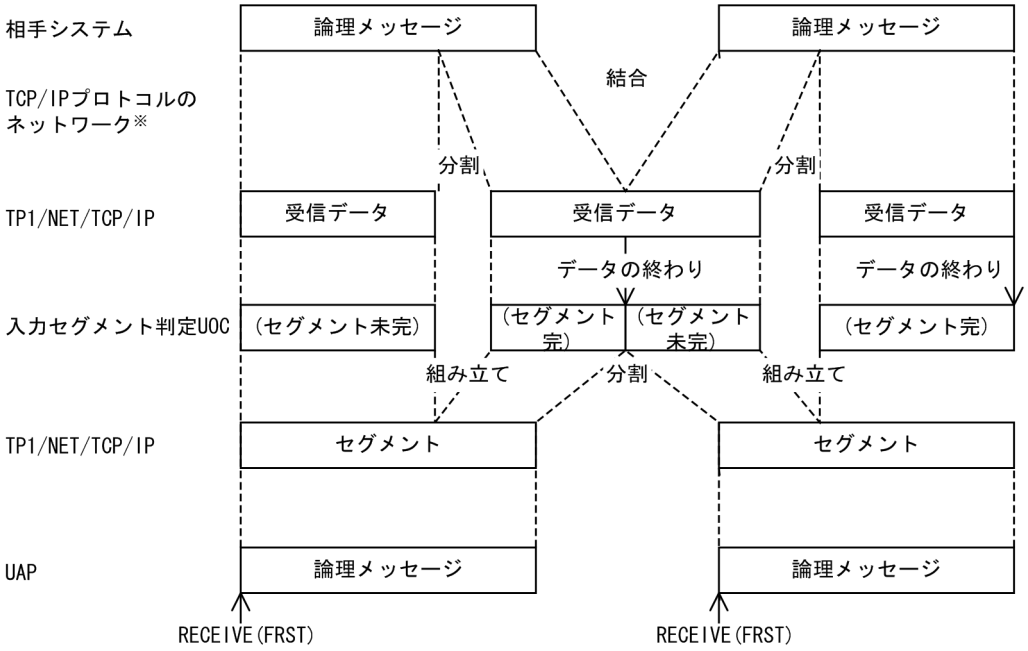
入力セグメント判定 UOC は，TP1/NET/TCP/IP が蓄積している受信メッセージがセグメントとして完成しているかどうか，完成している場合は，完成したセグメントと後続のセグメントとの境界がどこであるかを判断するための UOC です。

入力セグメント判定 UOC を使用する場合，セグメントの終わりを判定するための目印が必要となります。例えば，メッセージにメッセージ長を含んだヘッダを付けることによって，入力セグメント判定 UOC で，セグメントの終わりを判定できます。

送信されたメッセージが後続のメッセージと結合している場合は，該当するメッセージのセグメントの終わりを判定すると同時に，次のメッセージの長さを計算します。メッセージの形式は，システム間であらかじめ取り決めておきます。

入力セグメント判定 UOC での分割・組み立てを次の図に示します。

図 2-46 入力セグメント判定 UOC での分割・組み立て



注
※ TCP/IPプロトコルでは，一つの論理メッセージを複数のTCPパケットに分けたり，
複数の論理メッセージを一つのTCPパケットにまとめたりすることがあります。

入力セグメント判定 UOC がセグメント未完を指示したとき，UOC に渡したメッセージのサイズと該当メッセージの残りのサイズの和がバッファグループ定義で定義した受信バッファのサイズを超えている場合，TP1/NET/TCP/IP は受信バッファオーバフローと見なし，コネクションを解放します。

入力セグメント判定 UOC の詳細については，「5.1.1 入力セグメントの判定」を参照してください。

(3)　受信バッファの空き待ち

論理メッセージの組み立てと論理メッセージの入力キューへの格納は，非同期に行われます。

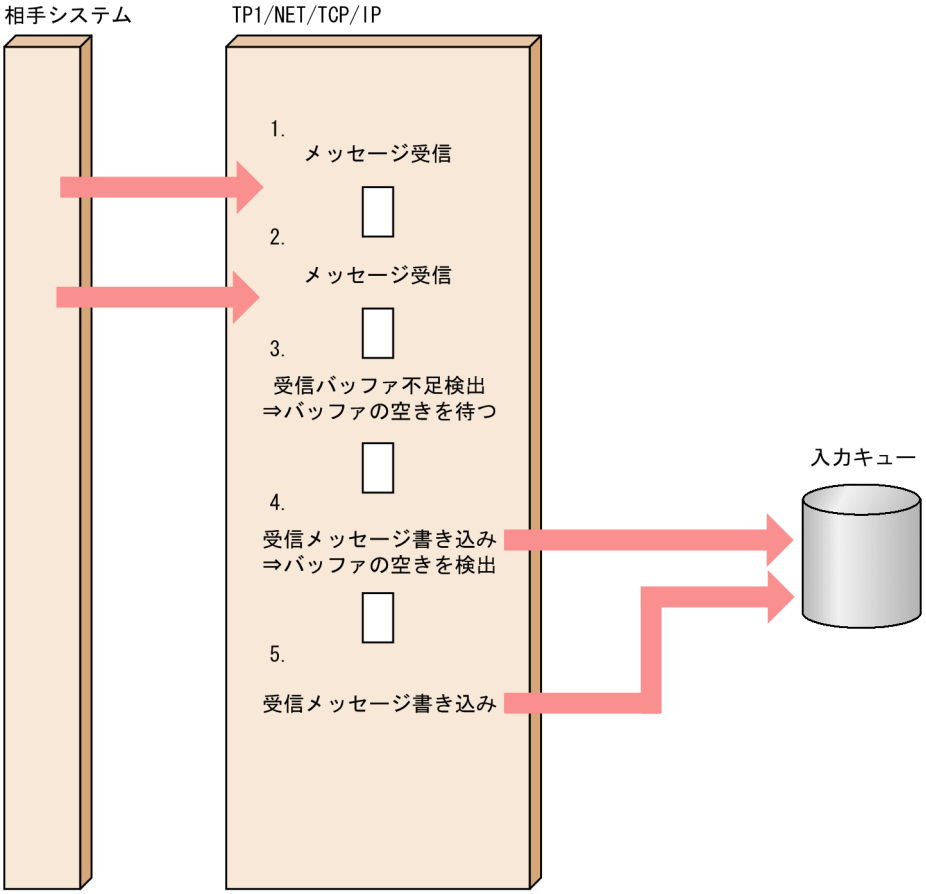
TCP/IP ソケットの受信バッファから受け取ったメッセージは，入力キューに格納するまでコネクション定義（mcftalccn -g）の rcvbuf オペランドで指定した受信バッファに保持されます。

相手システムから一方的に連続してメッセージを受信するなどによって受信バッファの空きが不足した場合，TP1/NET/TCP/IP は受信バッファの空きを最大 3 分間待ち合わせます。受信バッファの空きを待ち合わせている間は，相手システムからメッセージを受信したり，相手システムからのコネクション解放を検出できなくなります。一定時間が経過したあとも受信バッファの空きがない場合は，メッセージログ（KFCA14847-E）を出力し，コネクションを解放します。

入力キューへの格納が完了するなどによって，TP1/NET/TCP/IP が受信バッファの空きを検出すると，再び相手システムからメッセージを受信したり，相手システムからのコネクション解放を検出できるようになります。なお，受信バッファの空きチェックはタイマ定義（mcfttim -b）の btim オペランドの指定値の間隔で行います。

受信バッファの空き待ちを次の図に示します。

図 2-47　受信バッファの空き待ち



1. 相手システムからメッセージを受信します。

2. 1.で受信したメッセージを入力キューに格納する前に相手システムからメッセージを受信します。
3. 受信バッファ不足を検出すると，メッセージログ（KFCA14877-I）を出力し，受信バッファの空きを待ち合わせます。
4. 1.で受信したメッセージを入力キューに格納すると，受信バッファの空きを検出します。
5. 2.で受信したメッセージを入力キューに格納します。

受信バッファの空き待ちによって受信処理が遅延し，性能に影響を与えるおそれがあるため，受信バッファ面数には余裕を持った値を見積もってください。詳細については，「6. システム定義」の「mcftalccn（コネクション定義の開始）」の注意事項を参照してください。

2.3.8 送受信時のデータサイズ

TP1/NET/TCP/IP が相手システムにメッセージを送信するときは，ネットワークに対して，UAP から渡された論理メッセージのサイズで送信要求します。また，TP1/NET/TCP/IP が相手システムからメッセージを受信し，UAP に受信メッセージを渡すときは，受信メッセージの組み立て機能または入力セグメント判定 UOC で組み立てたメッセージを UAP に通知します。

出力メッセージ編集 UOC または入力メッセージ編集 UOC でヘッダを付ける場合，データサイズは送受信するメッセージの長さとヘッダの長さとの合計になります。このため，送受信バッファのサイズには，上記の合計値，またはそれ以上の値を設定してください。

2.3.9 メッセージの重複，および欠落のチェック

TP1/NET/TCP/IP では，送信したメッセージの重複および欠落をチェックしません。メッセージの重複，および欠落をチェックする場合は，相手システムと自システムの UAP との間でメッセージの形式を決めます。例えば，メッセージの内部を制御部とデータ部に分け，制御部にメッセージの通番を格納し，データ部にはユーザの送信したいデータを格納します。メッセージを受信する側が通番をチェックすることで，メッセージの重複，および欠落を検出できます。メッセージの形式の例を次の図に示します。

図 2-48 メッセージの形式の例

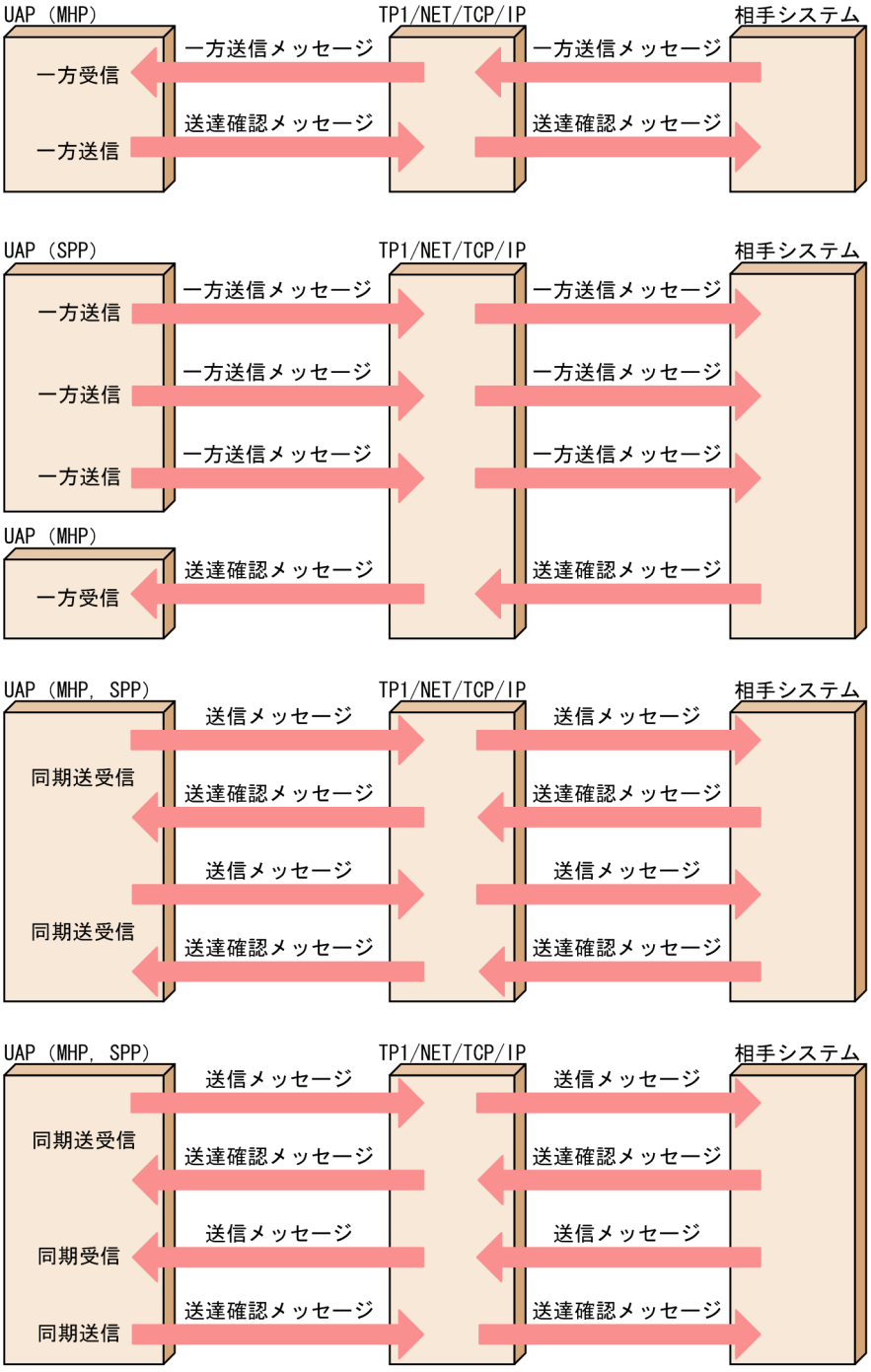
2.3.10 メッセージ送達の確認

メッセージの送達確認には，UAP で実装する方法と TP1/NET/TCP/IP が提供する送達確認機能を使用する方法があります。

(1) UAP で実装する場合

UAP で実装する場合，メッセージ送達の確認は，あらかじめ相手システムと自システムの UAP との間で送達確認のためのメッセージ形式を取り決めておき，送達確認メッセージの送受信によって行います。メッセージ送達確認の例を次の図に示します。

図 2-49　メッセージ送達確認の例



(2)　TP1/NET/TCP/IP が提供する送達確認機能を使用する場合

TP1/NET/TCP/IP が提供する送達確認機能には，DCCMII/TCP および DCCM3/TCP（以降，DCCMII/TCP および DCCM3/TCP のメッセージ送達確認機能をサポートした相手システムを DCCM と呼びます）のメッセージ送達確認のプロトコルに従ったメッセージ送達確認と，任意の相手システムとのメッセージ送達確認があります。

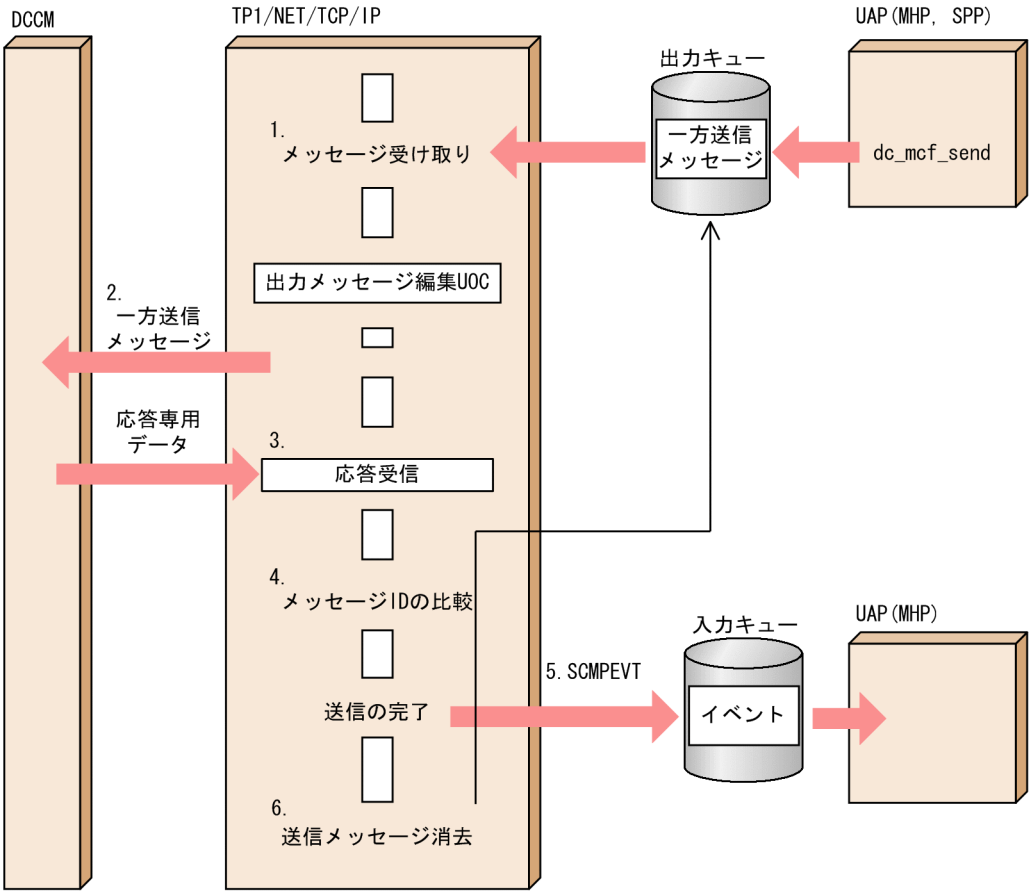
(a)　DCCM とのメッセージ送達確認

DCCM のメッセージ送達確認のプロトコルに従って，メッセージ送受信を行います。UAP は応答専用データ（DCCM のメッセージ送達確認機能で使用される送達確認メッセージ）を意識する必要はありません。

DCCM と接続し，メッセージ送達確認機能を使用する場合，TP1/NET/TCP/IP の定義と DCCM の定義を対応させる必要があります。対応させる内容の詳細については，「6.　システム定義」の「DCCMII/TCP および DCCM3/TCP の通信定義との関係」を参照してください。

DCCM への一方送信時の処理の流れを，dc_mcf_send 関数を使用する場合を例に，次の図に示します。

図 2-50　DCCM への一方送信時の処理の流れ



1. MHP，または SPP から一方送信メッセージを受信します。
2. DCCM へ一方送信メッセージを送信します。

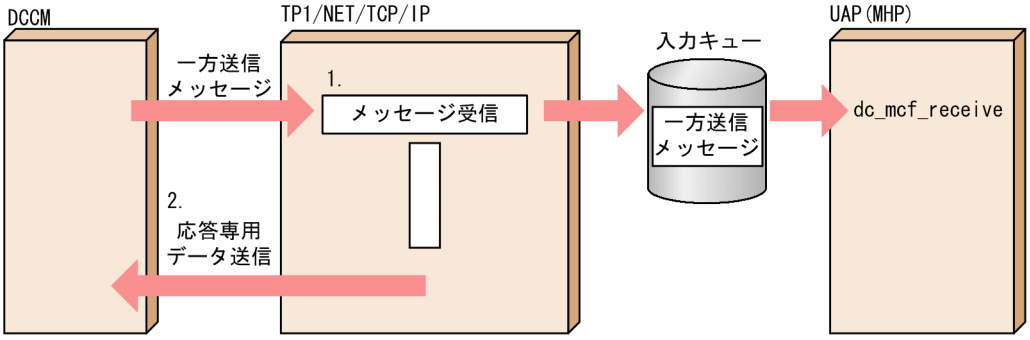
3. DCCM から応答専用データを受信します。
4. TP1/NET/TCP/IP が付加した送信メッセージのメッセージ ID と，DCCM が付加した応答専用データのメッセージ ID を比較します。その結果，メッセージ ID が一致した場合，送信処理を完了します。不一致の場合は，受信した応答専用データを破棄します。
5. 送信を完了した場合，送信完了通知イベント（SCMPEVT）を MHP に通知します。
6. 送信を完了した場合，出力キューにある一方送信メッセージのデータを消去します。

TP1/NET/TCP/IP は，受信した応答専用データを UAP に通知しません。

コネクション定義のメッセージ送信完了監視時間（mcftalccn -b sndcmptim）を指定している場合，メッセージ送信から応答専用データの受信までを監視できます。応答専用データの受信監視時間がタイムアウトした場合，CERREVT を起動します。このとき，送信メッセージは出力キューに戻されます。

DCCM からの一方受信時の処理の流れを，dc_mcf_receive 関数を使用する場合を例に，次の図に示します。

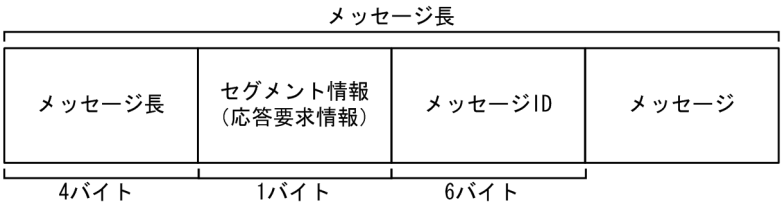
図 2-51　DCCM からの一方受信時の処理の流れ



1. DCCM から一方送信メッセージを受信します。
2. 入力キューへの書き込みが完了した時点で応答専用データを送信します。
   DCCM からの受信メッセージ内のセグメント情報に応答不要が指定されている場合は，応答専用データを送信しません。

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

図 2-52　TP1/NET/TCP/IP と DCCM 間での送受信メッセージの形式



メッセージ送達確認機能を使用する場合，メッセージ長，セグメント情報，メッセージ ID は TP1/NET/TCP/IP が自動的に付加・削除します。

メッセージ長

メッセージの組み立て・分解をするための全体長です。

セグメント情報

セグメントの種類と応答要求の種類が設定されます。セグメントの種類と応答要求の種類を次の表に示します。なお，次の表に示す格納値以外の値は不正データとし，コネクションを解放します。

表 2-6　セグメントの種類と応答要求の種類

セグメントの種類	応答要求の種類	格納値※	
		TP1/NET/TCP/IP からの送信	DCCM からの送信
単一セグメント	応答不要	－	$(10)_{16}$
	自動応答要求	$(18)_{16}$	$(18)_{16}$

（凡例）

－：該当しません（TP1/NET/TCP/IP はこのメッセージを送信しません）

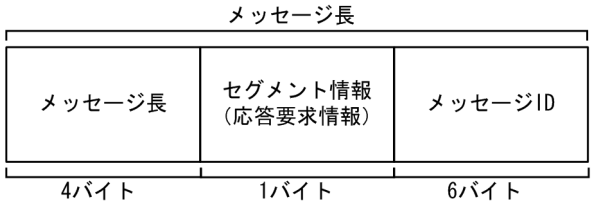
DCCM のプロトコルではセグメント情報に単一セグメントと複数セグメントを設定できますが，TP1/NET/TCP/IP は単一セグメントだけをサポートします。DCCM から複数セグメントのメッセージを受信した場合，TP1/NET/TCP/IP は不正データと見なしてコネクションを解放します。

メッセージ ID

メッセージと応答専用データの組を確認するための情報です。TP1/NET/TCP/IP はメッセージ送信ごとにユニークな値を採番します。

TP1/NET/TCP/IP と DCCM 間で送受信する応答専用データの形式を次の図に示します。

図 2-53　応答専用データの形式



メッセージ長

メッセージの組み立て・分解をするための全体長として，11 を設定します。

セグメント情報

TP1/NET/TCP/IP は，単一セグメント，応答専用データを示す，0x10 を設定します。

メッセージ ID

DCCM から受信したメッセージのメッセージ ID を設定します。

TP1/NET/TCP/IP のメッセージ送信と，DCCM のメッセージ送信が衝突した場合の処理の流れについては，「付録 F　送受信メッセージの衝突時の処理の流れ（メッセージ送達確認機能使用時）」を参照してください。

(b)　任意の相手システムとのメッセージ送達確認

受信メッセージ判定 UOC を使用し，TP1/NET/TCP/IP が受信したメッセージが送達確認メッセージかどうかを判定することで，任意の相手システムとのメッセージ送達確認を行うことができます。

受信メッセージ判定 UOC を使用した任意の相手システムとのメッセージ送達確認では，受信メッセージ判定 UOC の msg_type に指定したメッセージの種別によって，処理が異なります。

メッセージの種別には，次の種別があります。

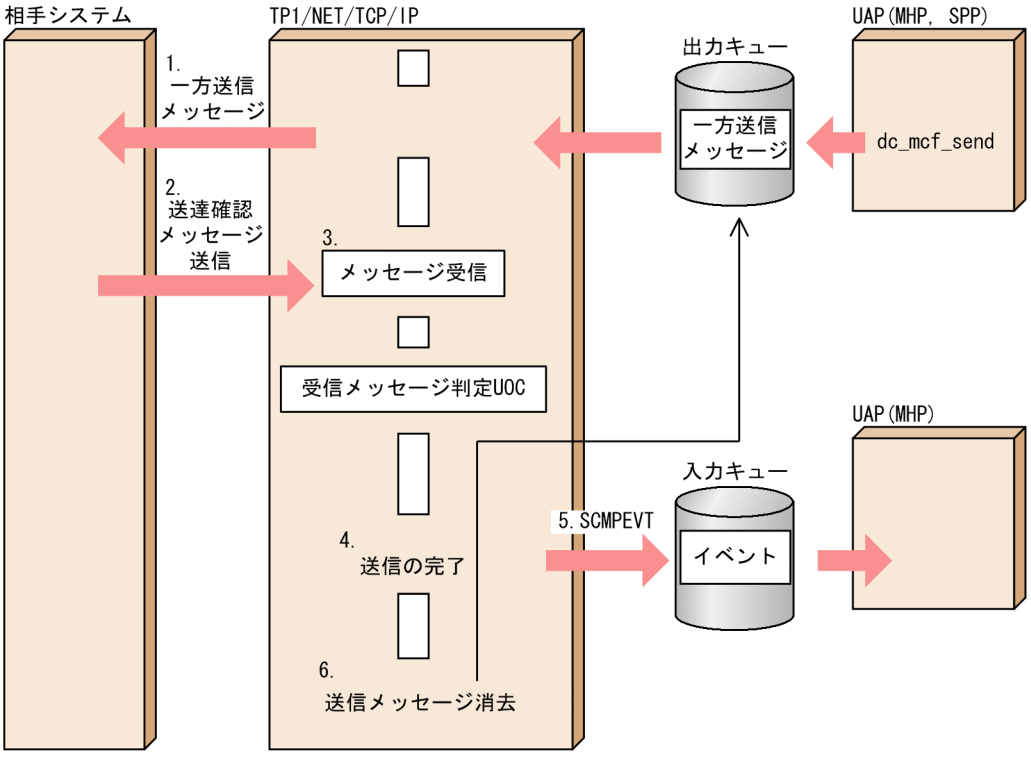
- DCTCP_UOC_MR_MT_ACK（送達確認メッセージ）
- DCTCP_UOC_MR_MT_INFO（一方送信メッセージ）
- DCTCP_UOC_MR_MT_CLS（コネクション解放）
- DCTCP_UOC_MR_MT_IGN（破棄メッセージ）

種別ごとに，処理の流れを説明します。

■ 相手システムへの一方送信（DCTCP_UOC_MR_MT_ACK（送達確認メッセージ）の場合）

相手システムへの一方送信時の処理の流れを次の図に示します。

図 2-54　相手システムへの一方送信時の処理の流れ（DCTCP_UOC_MR_MT_ACK）



1. UAP は相手システムへの一方送信メッセージを送信します。
2. 相手システムは，送達確認メッセージを送信します。
3. 相手システムからの送達確認メッセージを待ち合わせ，受信します。

4. メッセージ受信時に起動する受信メッセージ判定 UOC で送達確認メッセージを指定した場合，送信処理を完了します。
5. 送信を完了した場合，送信完了通知イベント（SCMPEVT）を MHP に通知します。
6. 送信を完了した場合，出力キューから送信済みメッセージのデータを消去します。

注

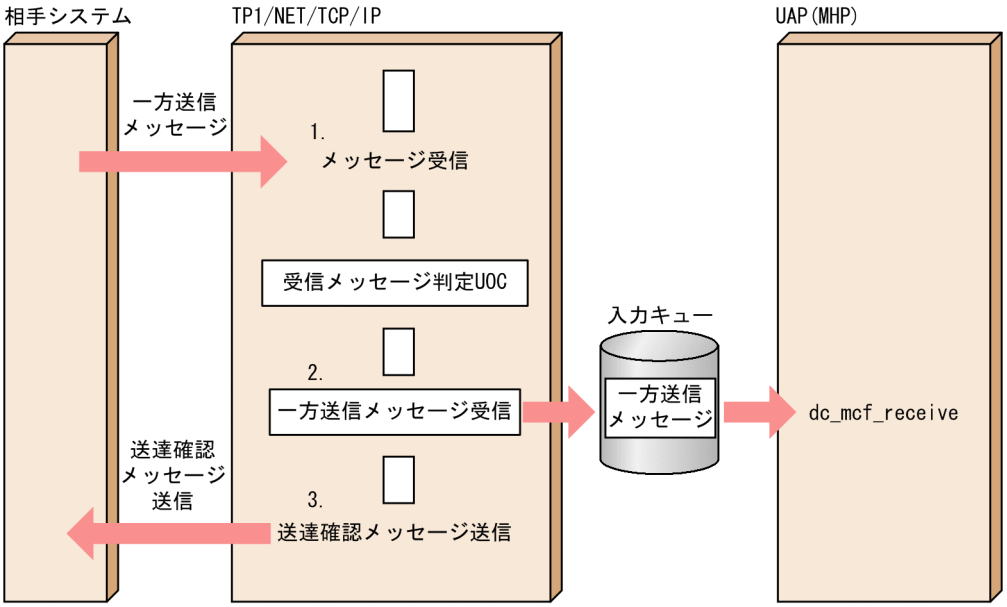
送達確認メッセージを不要とする一方送信メッセージは送信できません。

コネクション定義のメッセージ送信完了監視時間（mcftalccn -b sndcmptim）を指定している場合，メッセージ送信から送達確認メッセージの受信までを監視できます。メッセージ送信完了監視時間がタイムアウトした場合，CERREVT を起動します。このとき，送信メッセージは出力キューに戻されます。

■ 相手システムからの一方受信（DCTCP_UOC_MR_MT_INFO（一方送信メッセージ）の場合）

相手システムからの一方受信時の処理の流れを次の図に示します。

図 2-55　相手システムからの一方受信時の処理の流れ（DCTCP_UOC_MR_MT_INFO）



1. 相手システムから一方送信メッセージを受信します。
2. 受信メッセージ判定 UOC で一方送信メッセージを設定した場合，一方送信メッセージを受信し，UAP にメッセージ受信を通知します。
3. 受信メッセージ判定 UOC で，送達確認メッセージの送信要否に応答要を設定した場合，TP1/NET/TCP/IP は入力キューへの書き込みが完了した時点で受信メッセージ判定 UOC で編集したメッセージを送信します。

■ 相手システムからの一方受信（DCTCP_UOC_MR_MT_CLS（コネクション解放）の場合）

受信メッセージ判定 UOC で，メッセージ種別にコネクション解放を設定した場合，TP1/NET/TCP/IP は受信したメッセージを UAP に通知しないで破棄し，メッセージログ(KFCA14850-I)を出力してコネクションを解放し，CERREVT を起動します。このとき，送信処理中のメッセージは出力キューに残ります。

■ 相手システムからの一方受信（DCTCP_UOC_MR_MT_IGN（破棄メッセージ）の場合）

受信メッセージ判定 UOC で，メッセージ種別に破棄メッセージを設定した場合，TP1/NET/TCP/IP はメッセージログ（KFCA14849-I）を出力して，受信したメッセージを UAP に通知しないで破棄します。コネクション確立状態を維持したまま受信したメッセージを UAP に通知させない場合に指定してください。

TP1/NET/TCP/IP のメッセージ送信と，相手システムのメッセージ送信が衝突した場合の処理の流れについては，「付録 F　送受信メッセージの衝突時の処理の流れ（メッセージ送達確認機能使用時）」を参照してください。

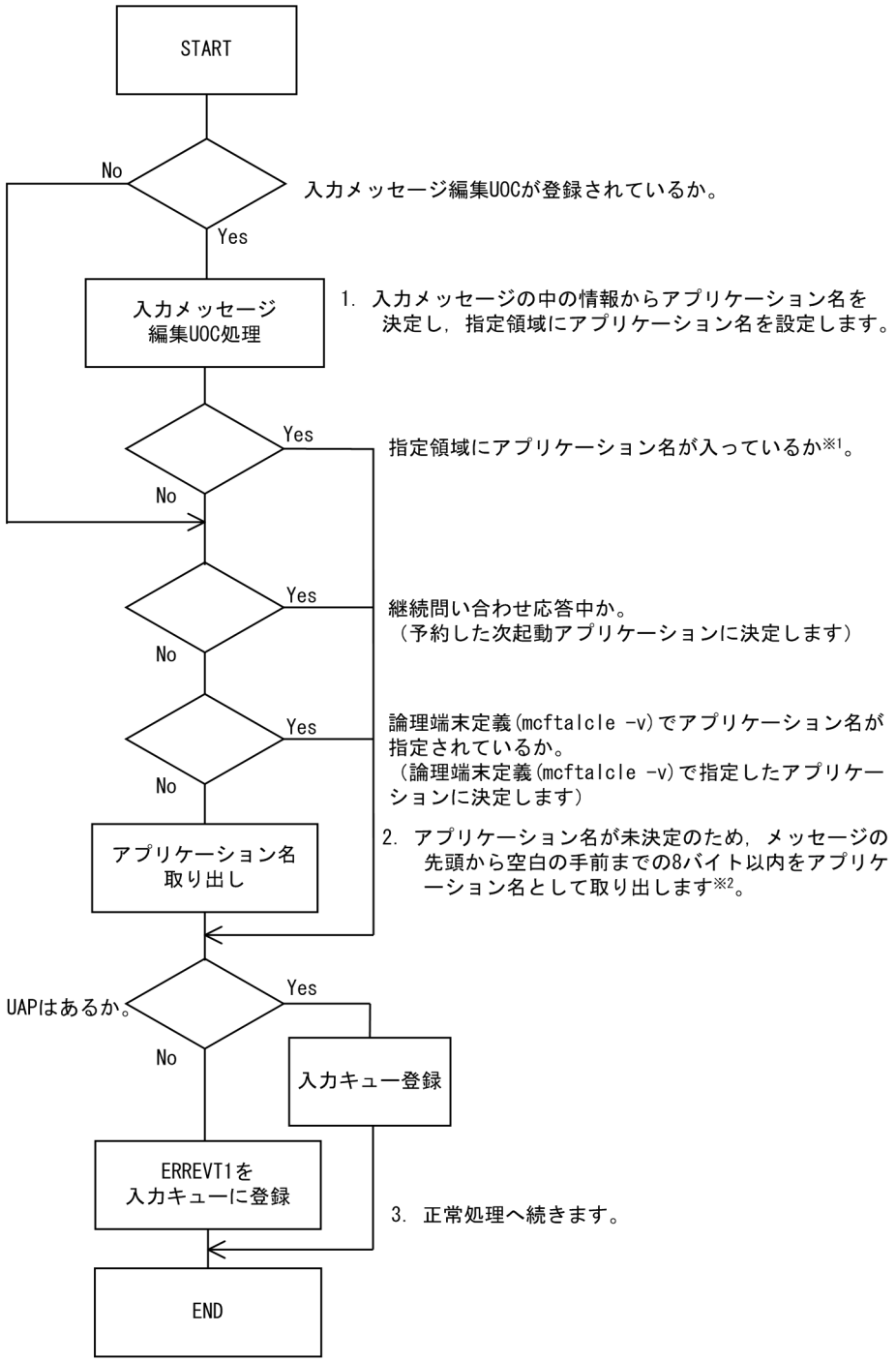
2.3.11　アプリケーション名の決定

TP1/NET/TCP/IP は，メッセージを受信するときに，アプリケーション名を決定します。アプリケーション名を決定する方法を，優先順位の高い順に次に示します。

1. 入力メッセージ編集 UOC でアプリケーション名を決定する。
2. 予約した次起動アプリケーション（継続問い合わせ応答中の場合）でアプリケーション名を決定する。
3. 論理端末定義（mcftalcle -v）でアプリケーション名を決定する。
4. 入力メッセージの先頭から空白の手前までの 8 バイト以内でアプリケーション名を決定する。

アプリケーション名を決定できなかった場合，エラーイベント（ERREVT1）を起動します。アプリケーション名の決定の処理手順を次の図に示します。

図 2-56　アプリケーション名の決定の処理手順



注※1

指定領域(dcmcf_uoc_min_nのaplname)が「¥0」で埋められている場合，アプリケーション名が入っていないと見なします。

注※2

入力セグメント判定UOCを使用している場合は，ヘッダの先頭に必ずアプリケーション名を設定してください。標準提供の入力セグメント判定UOCを使用する場合は，取り出すことはできません。

先頭から9バイト目までに空白がない，または先頭に空白がある場合は，アプリケーション名を不正とし，ERREVT1を起動します。

2.3.12　コネクション再確立時の未送信メッセージの送信抑止

(1)　コネクション再確立時の未送信メッセージの送信抑止の概要

コネクション再確立時の未送信メッセージの送信抑止は，MHP からメッセージ送受信関数を発行したときに，TP1/NET/TCP/IP がコネクションの再確立の有無をチェックし，再確立前の古いコネクションで受信したメッセージに対する一方送信メッセージの送信や同期型の関数の発行を抑止する機能です。

この機能を使用すると，UAP や相手システムで不要なメッセージを削除する必要がなくなるため，TP1/NET/TCP/IP を使用したシステムの開発が容易になります。

一方送信メッセージの送信を抑止した場合，送受信メッセージ廃棄通知イベント（MDELEVT）を通知します。MDELEVT を通知するには，MCF アプリケーション定義で MDELEVT を定義しておく必要があります。

同期型の関数の発行を抑止した場合，MDELEVT は起動しません。

なお，応答メッセージについては，この機能の使用に関係なく，コネクションが再確立すると応答メッセージの送信が抑止されます。このとき，MDELEVT は起動しません。

注意事項

一方送信メッセージの送信を抑止した場合，TP1/NET/TCP/IP は相手システムにメッセージを送信したと見なしません。そのため，次の点に注意してください。

- AJ（メッセージ送信完了ジャーナル）を取得する（論理端末定義（mcftalcle -o）の aj オペランドを省略，または yes を指定）場合，AJ を取得しません。
- コネクションの無通信状態を監視している場合（コネクション定義（mcftalccn -k）の notrftime オペランドに 0 以外の値を指定），状態監視の経過時間をリセットしません。

(2)　送信抑止ルール

この機能が有効となるか無効となるかは，メッセージ送受信関数を発行した UAP の種類，UAP の起動要因，入力元論理端末とメッセージ送受信関数を発行した論理端末の差異に依存します。各々の条件で機能が有効となるか無効となるかを次の表に示します。

表 2-7　送信抑止の実行条件

UAP の種類	UAP の起動要因	入力元論理端末と関数発行先論理端末の差異	送信の抑止
MHP	mcfuevt コマンド	−	×
	上記以外	同じ	○
		異なる	×

UAP の種類	UAP の起動要因	入力元論理端末と関数発行先論理端末の差異	送信の抑止
SPP	−	−	×

（凡例）

○：有効です（送信は抑止されます）。

×：無効です（送信は抑止されません）。

−：該当しません。

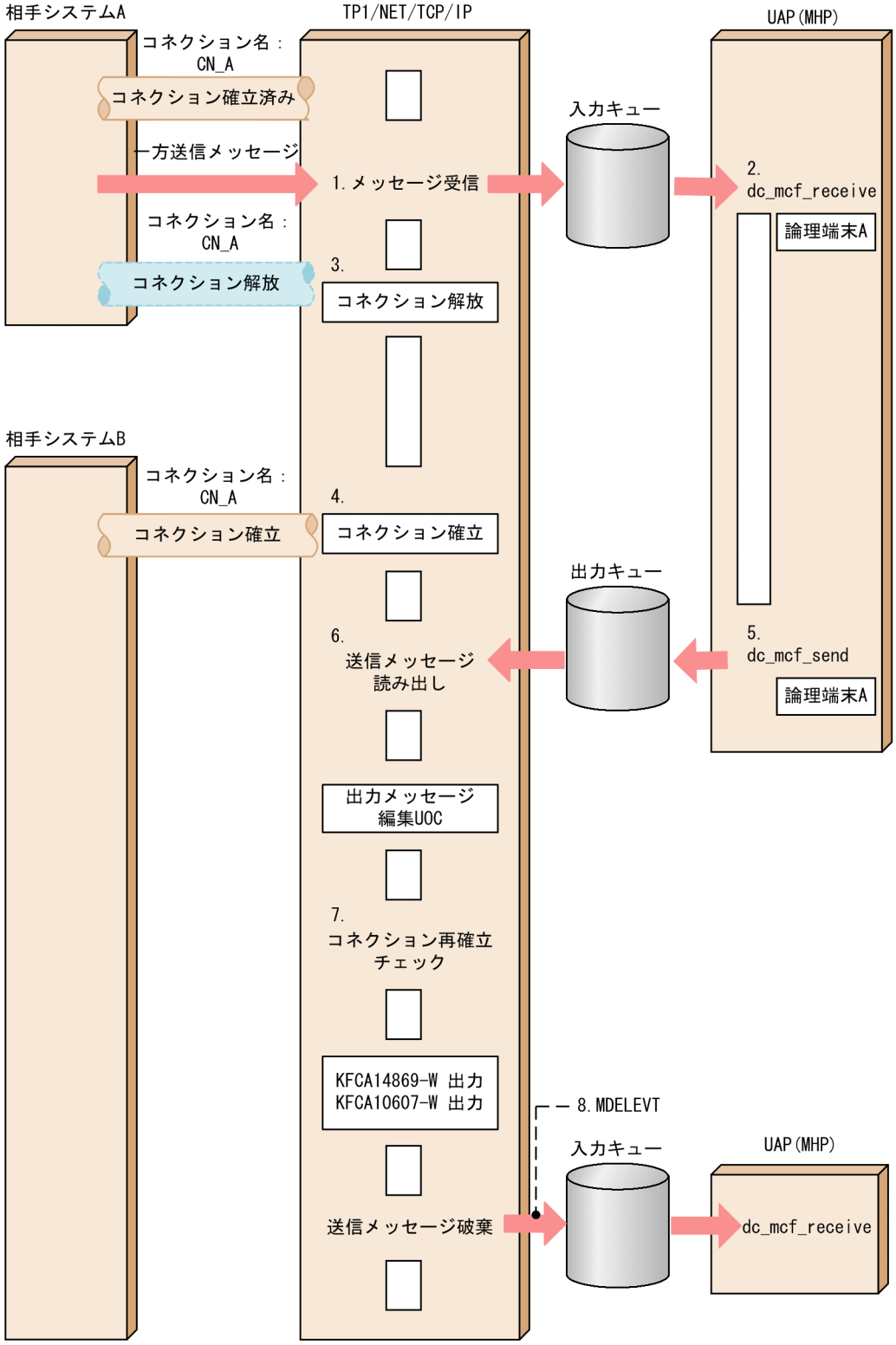
(3) 入力元論理端末と同じ論理端末にメッセージ送受信関数を発行した場合

(a) 一方送信メッセージの送信の場合

TP1/NET/TCP/IP が一方送信メッセージを受信してから，一方送信メッセージを受信した MHP で，入力元論理端末と同じ論理端末に一方送信メッセージを送信するまでの間に，コネクションの解放と再確立が発生した場合，TP1/NET/TCP/IP はメッセージを破棄し，MDELEVT を通知します。

入力元論理端末と同じ論理端末への一方送信メッセージの送信関数発行を，dc_mcf_send 関数を使用する場合を例に，次の図に示します。

図 2-57　入力元論理端末と同じ論理端末で一方送信メッセージの送信関数を発行したときの処理の流れ



1. 相手システム A（コネクション名：CN_A）からメッセージを受信します。
2. dc_mcf_receive 関数を発行してメッセージを受け取ります。
3. コネクション CN_A を解放します。

4. CN_A が相手システム B とコネクションを再確立します。

5. dc_mcf_send 関数を発行して出力キューにメッセージを格納します。

6. 論理端末の閉塞解除を契機に，出力キューからメッセージを読み出します。

7. TP1/NET/TCP/IP がコネクションの再確立を検出したため，出力キューのメッセージを破棄します。

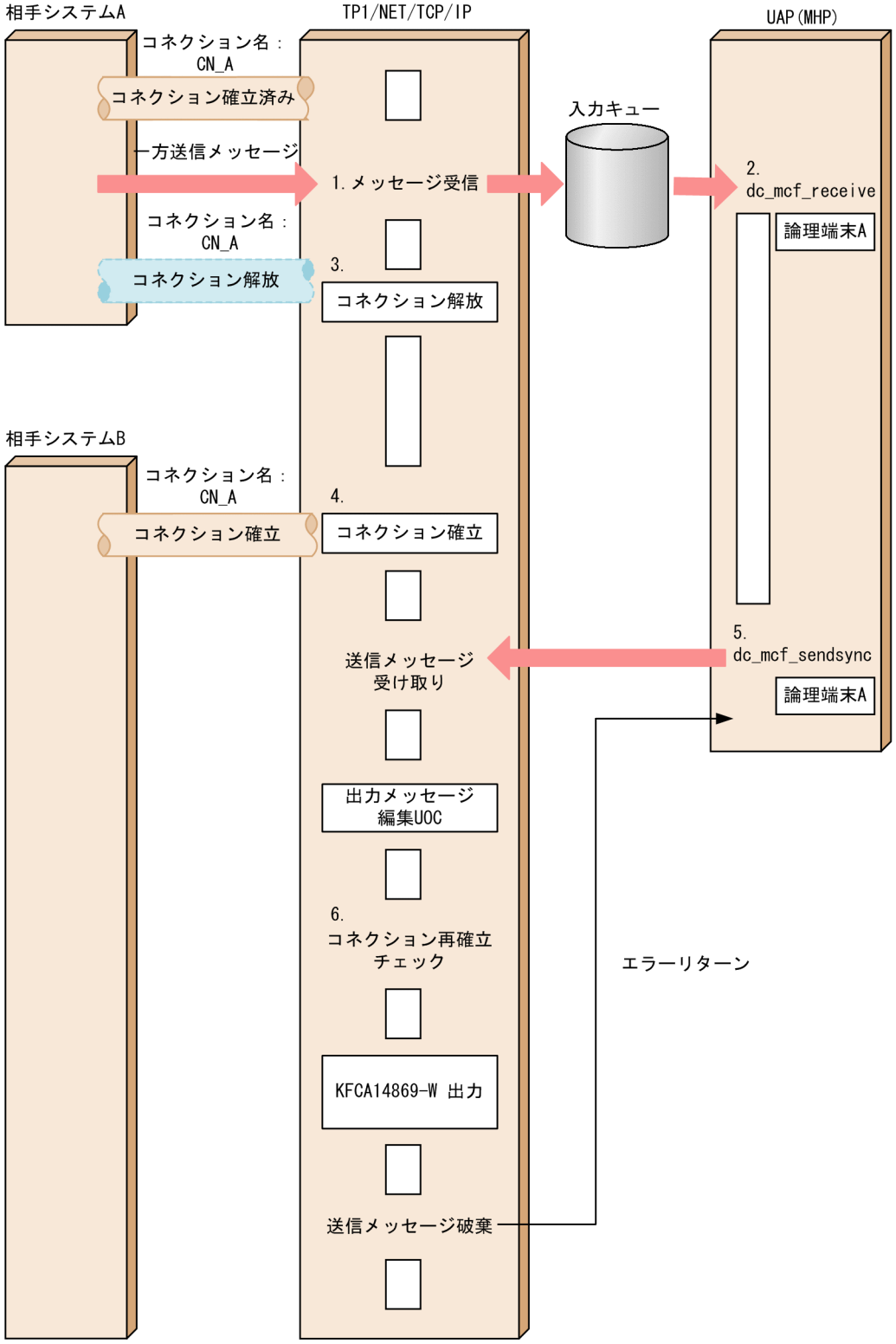
8. MDELEVT を起動します。

(b) 同期型メッセージの送信または同期型メッセージの送受信の場合

TP1/NET/TCP/IP が一方送信メッセージを受信してから，一方送信メッセージを受信した MHP で，入力元論理端末と同じ論理端末に，同期型メッセージの送信関数または同期型メッセージの送受信関数を発行するまでの間に，コネクションの解放と再確立が発生した場合，発行した関数はリターン値 DCMCFRTN_73002 またはステータスコード 73002 でエラーリターンし，送信メッセージを破棄します。

入力元論理端末と同じ論理端末への同期型メッセージの送信関数または同期型メッセージの送受信関数発行を，dc_mcf_sendsync 関数を使用する場合を例に，次の図に示します。

図 2-58　入力元論理端末と同じ論理端末で同期型メッセージの送信関数を発行したときの処理の流れ



1. 相手システム A（コネクション名：CN_A）からメッセージを受信します。
2. dc_mcf_receive 関数を発行してメッセージを受け取ります。
3. コネクション CN_A を解放します。

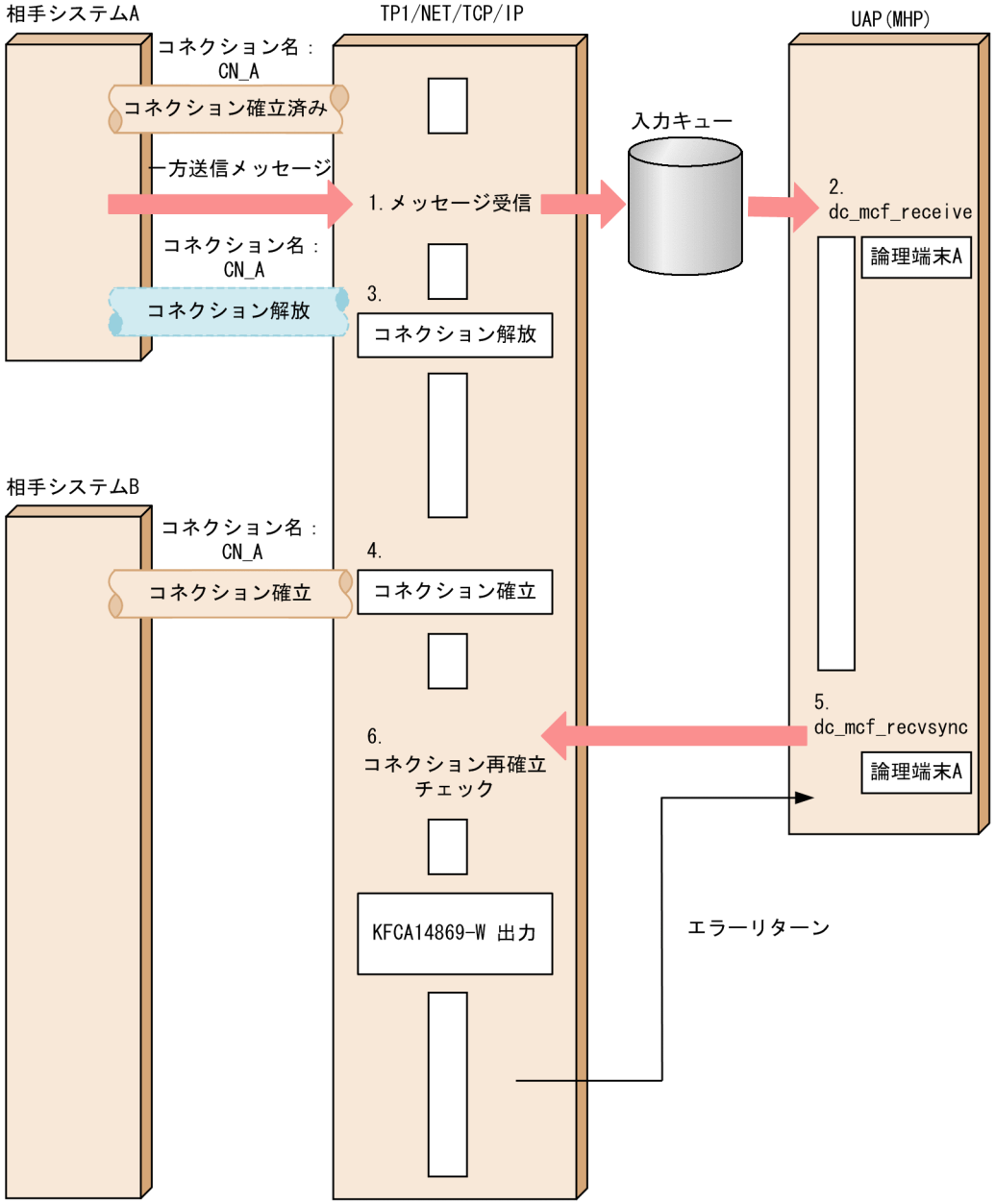
4. CN_A が相手システム B とコネクションを再確立します。
5. dc_mcf_sendsync 関数を発行して同期型メッセージを送信します。
6. TP1/NET/TCP/IP がコネクションの再確立を検出したため，メッセージを破棄して MHP にリターン値 DCMCFRTN_73002 でエラーリターンします。

(c) 同期型メッセージの受信の場合

TP1/NET/TCP/IP が一方送信メッセージを受信してから，一方送信メッセージを受信した MHP で，入力元論理端末と同じ論理端末に同期型メッセージの受信関数を発行するまでの間に，コネクションの解放と再確立が発生した場合，発行した関数はリターン値 DCMCFRTN_73002 またはステータスコード 73002 でエラーリターンします。

入力元論理端末と同じ論理端末への同期型メッセージの受信関数発行を，dc_mcf_recvsync 関数を使用する場合を例に，次の図に示します。

図 2-59　入力元論理端末と同じ論理端末で同期型メッセージの受信関数を発行したときの処理の流れ



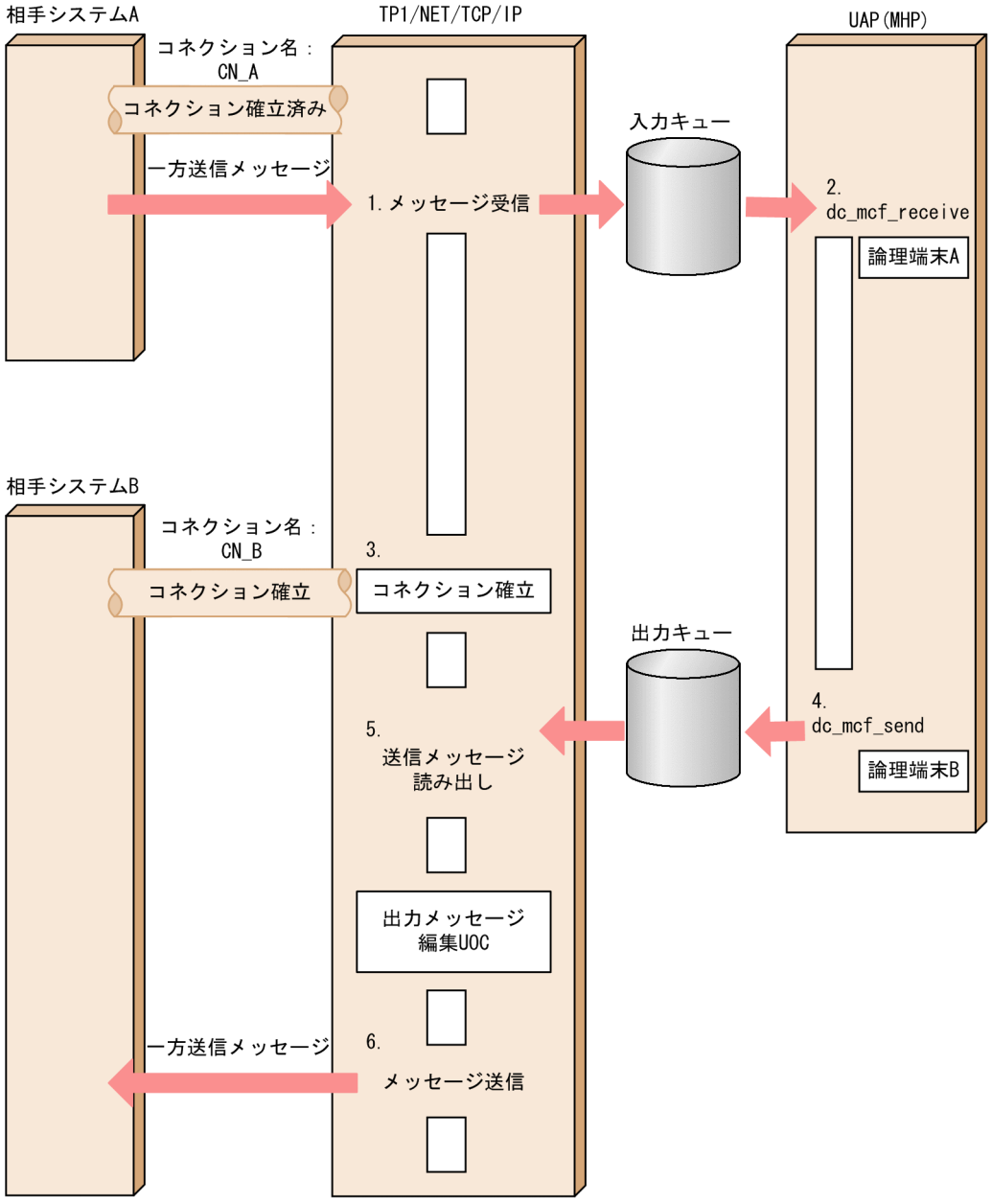
1. 相手システム A（コネクション名：CN_A）からメッセージを受信します。
2. dc_mcf_receive 関数を発行してメッセージを受け取ります。
3. コネクション CN_A を解放します。
4. CN_A が相手システム B とコネクションを再確立します。
5. dc_mcf_recvsync 関数を発行して同期型メッセージの受信を要求します。
6. TP1/NET/TCP/IP がコネクションの再確立を検出したため，MHP にリターン値 DCMCFRTN_73002 でエラーリターンします。

(4)　入力元論理端末と異なる論理端末でメッセージ送受信関数を発行した場合

TP1/NET/TCP/IP が一方送信メッセージを受信してから，一方送信メッセージを受信した MHP で，入力元論理端末と異なる論理端末にメッセージ送受信関数を発行した場合，TP1/NET/TCP/IP は相手システムとのコネクションが再確立したかどうかをチェックしません。

入力元論理端末と異なる論理端末へのメッセージ送受信関数発行を，dc_mcf_send 関数を使用する場合を例に，次の図に示します。

図 2-60　入力元論理端末と異なる論理端末でメッセージ送受信関数を発行したときの処理の流れ



1. 相手システム A（コネクション名：CN_A）からメッセージを受信します。
2. dc_mcf_receive 関数を発行してメッセージを受け取ります。
3. 相手システム B（コネクション名：CN_B）とコネクションを確立します。

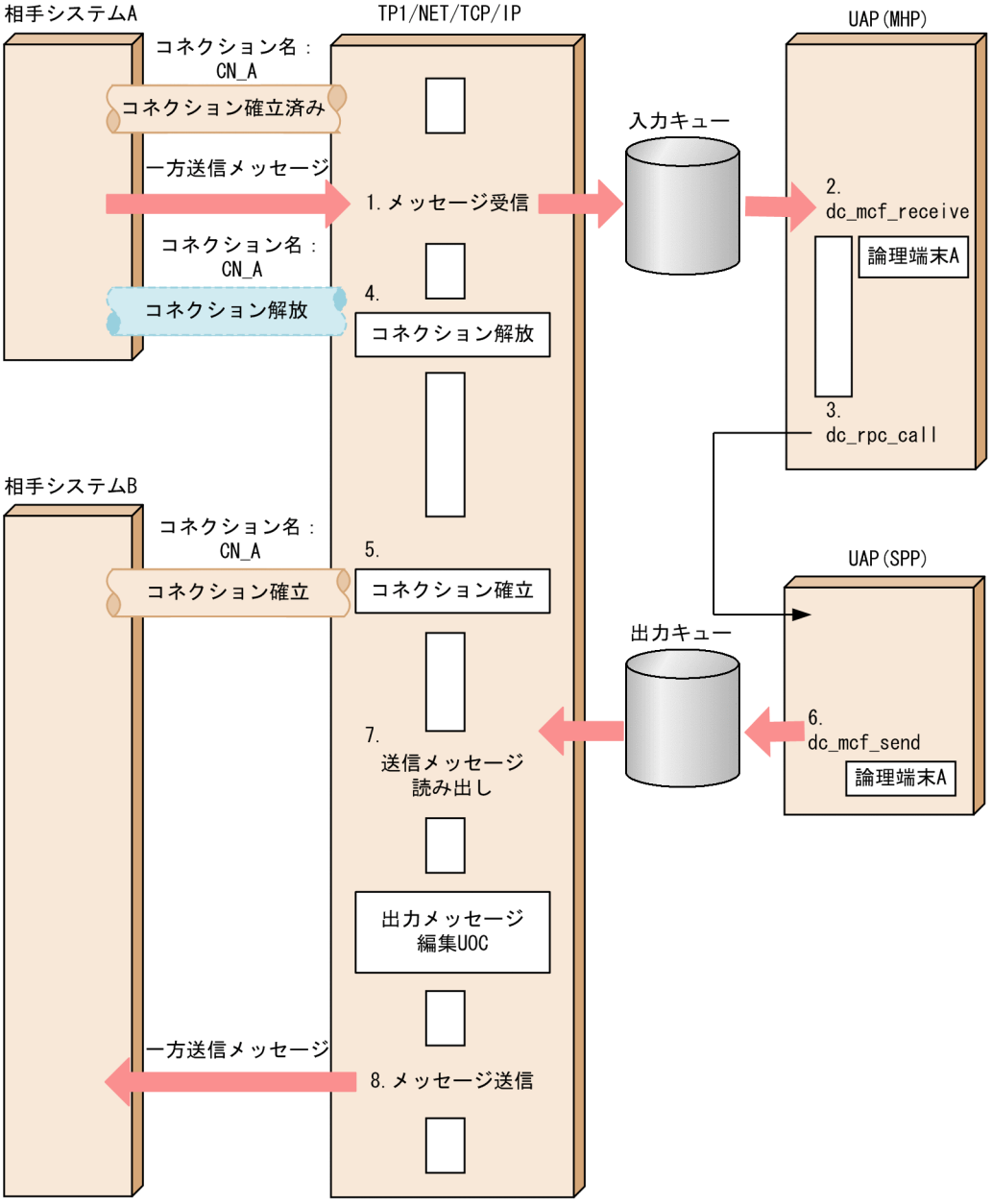
4. dc_mcf_send 関数を発行して CN_B に対応する論理端末 B の出力キューにメッセージを格納します。
5. 論理端末の閉塞解除を契機に出力キューからメッセージを読み出します。
6. 入力元の論理端末と異なるため，コネクションが再確立したかどうかをチェックしないで，相手システム B にメッセージを送信します。

(5) SPP からメッセージ送受信関数を発行した場合

SPP からメッセージ送受信関数を発行した場合，TP1/NET/TCP/IP はコネクションが再確立したかどうかをチェックしません。

SPP からのメッセージ送受信関数発行を，dc_mcf_send 関数を使用する場合を例に次の図に示します。

図 2-61 SPP からメッセージ送受信関数を発行したときの処理の流れ



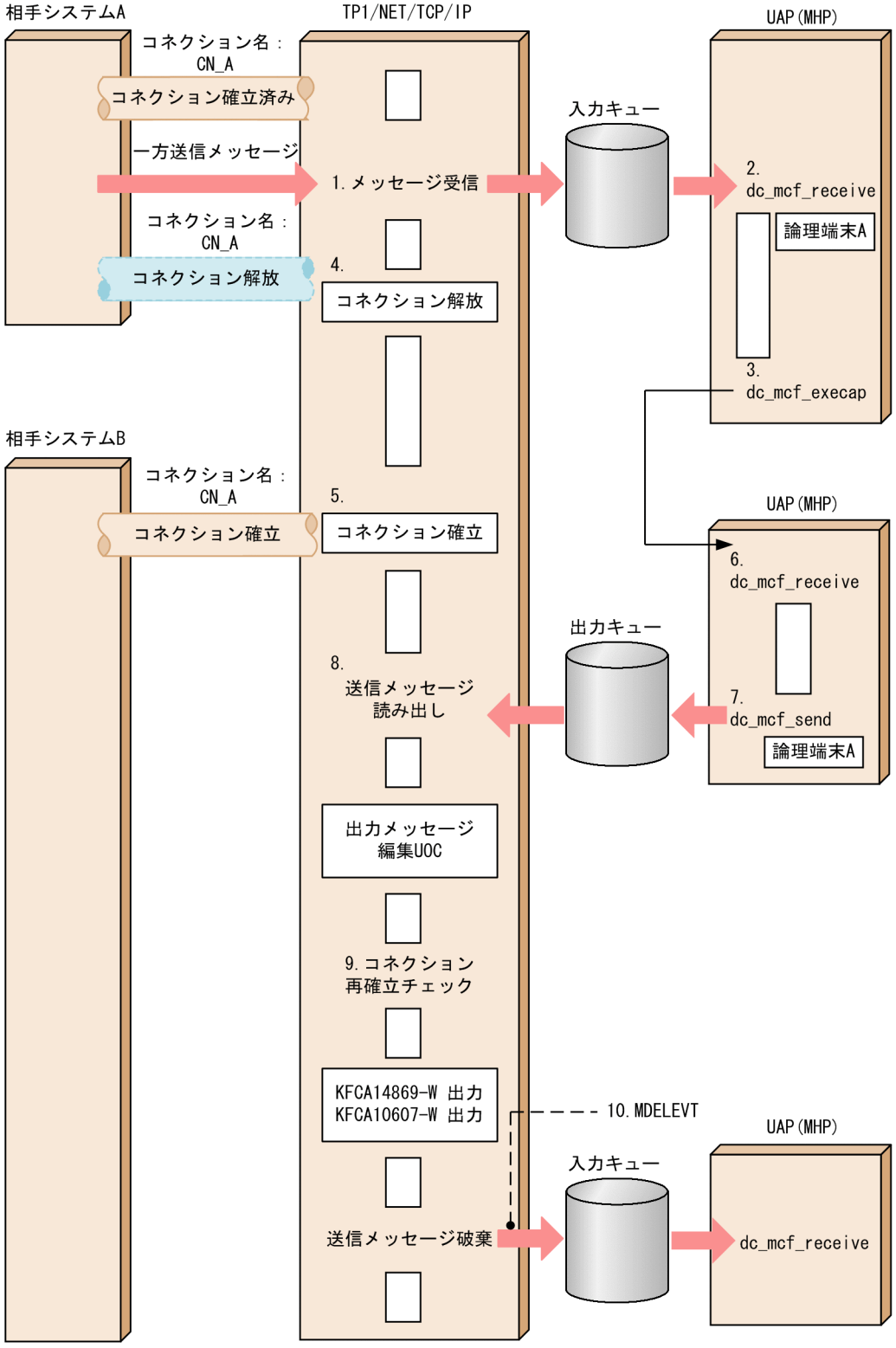
1. 相手システム A（コネクション名：CN_A）からメッセージを受信します。
2. dc_mcf_receive 関数を発行してメッセージを受け取ります。
3. dc_rpc_call 関数を発行して SPP を起動します。
4. コネクション CN_A を解放します。
5. CN_A が相手システム B とコネクションを再確立します。
6. dc_mcf_send 関数を発行して出力キューにメッセージを格納します。
7. 論理端末の閉塞解除を契機に，出力キューからメッセージを読み出します。
8. メッセージ送受信関数を発行した UAP の種類が SPP のため，コネクションが再確立したかどうかをチェックしないで，相手システム B にメッセージを送信します。

(6) アプリケーションプログラムの起動によって起動された MHP からメッセージ送受信関数を発行した場合

TP1/NET/TCP/IP が一方送信メッセージを受信してから，一方送信メッセージを受信した MHP からアプリケーションプログラムの起動によって起動された MHP で，入力元論理端末と同じ論理端末でメッセージ送受信関数を発行するまでの間に，コネクションの解放と再確立が発生した場合，一方送信メッセージを受信した MHP からメッセージ送受信関数を発行したときと同様に送信を抑止します。

アプリケーションプログラムの起動によって起動された MHP からのメッセージ送受信関数発行を，dc_mcf_send 関数を使用する場合を例に，次の図に示します。

図 2-62　アプリケーションプログラムの起動によって起動された MHP からメッセージ送受信関数を発行したときの処理の流れ



1. 相手システム A（コネクション名：CN_A）からメッセージを受信します。
2. dc_mcf_receive 関数を発行してメッセージを受け取ります。
3. dc_mcf_execap 関数を発行して MHP を起動します。

4. コネクション CN_A を解放します。
5. CN_A が相手システム B とコネクションを再確立します。
6. dc_mcf_receive 関数を発行してメッセージを受け取ります。
7. dc_mcf_send 関数を発行して出力キューにメッセージを格納します。
8. 論理端末の閉塞解除を契機に，出力キューからメッセージを読み出します。
9. TP1/NET/TCP/IP がコネクションの再確立を検出したため，出力キューのメッセージを破棄します。
10. MDELEVT を起動します。

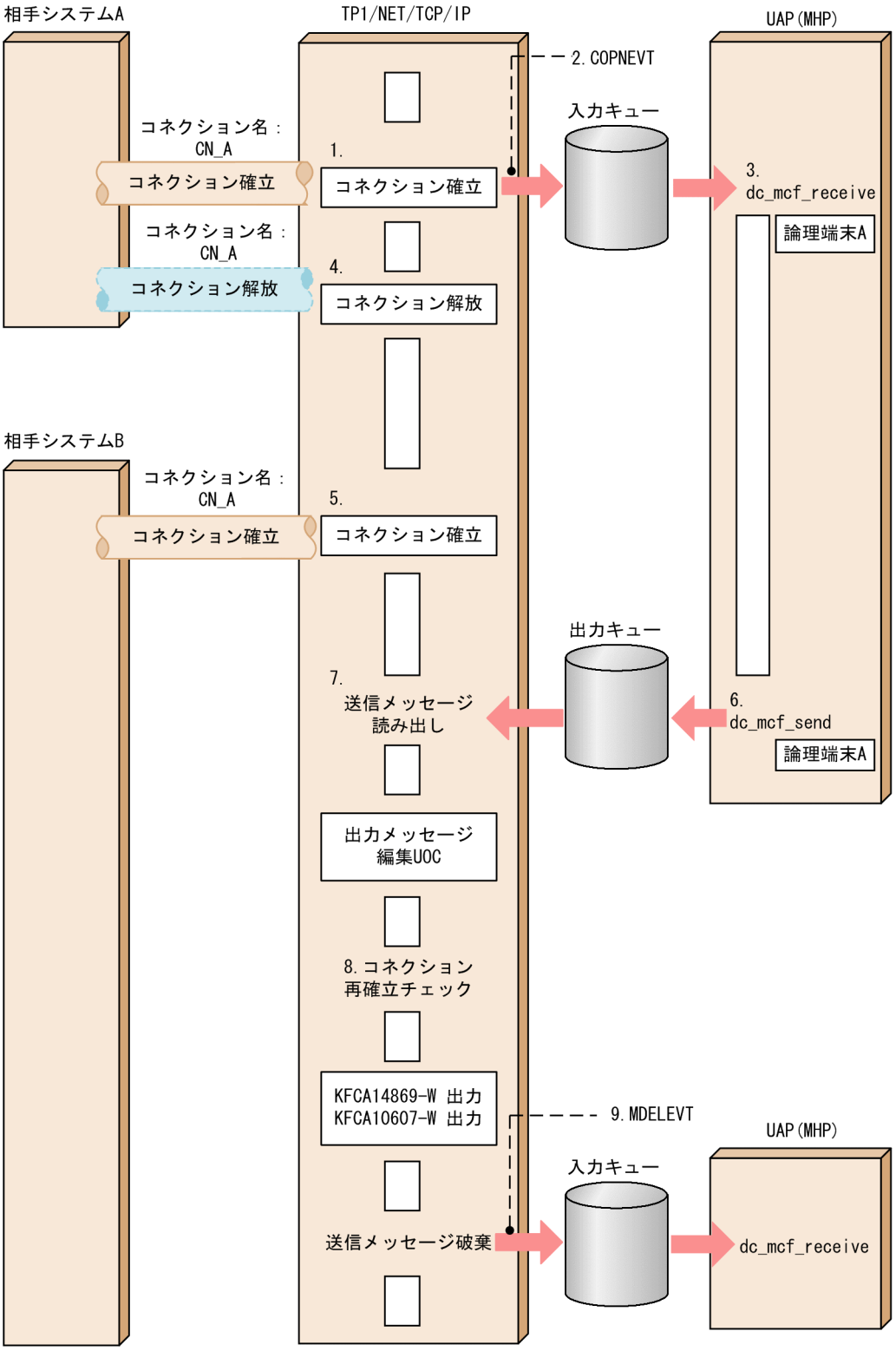
(7) MCF イベント処理用 MHP からメッセージ送受信関数を発行した場合

TP1/NET/TCP/IP で MCF イベントを起動するイベントが発生してから，MCF イベント処理用 MHP で，イベントが発生した論理端末と同じ論理端末でメッセージ送受信関数を発行するまでの間に，コネクションの解放と再確立が発生した場合，一方送信メッセージを受信した MHP からメッセージ送受信関数を発行したときと同様に送信を抑止します。

MCF イベント処理用 MHP からのメッセージ送受信関数発行を，COPNEVT および ERREVT1 から dc_mcf_send 関数を使用する場合を例に，次の図に示します。

(a)　COPNEVT の場合

図 2-63　COPNEVT 処理用 MHP からメッセージ送受信関数を発行したときの処理の流れ



1. 相手システム A（コネクション名：CN_A）とコネクションを確立します。
2. COPNEVT を起動します。
3. dc_mcf_receive 関数を発行してメッセージを受け取ります。

4. コネクション CN_A を解放します。

5. CN_A が相手システム B とコネクションを再確立します。

6. dc_mcf_send 関数を発行して出力キューにメッセージを格納します。

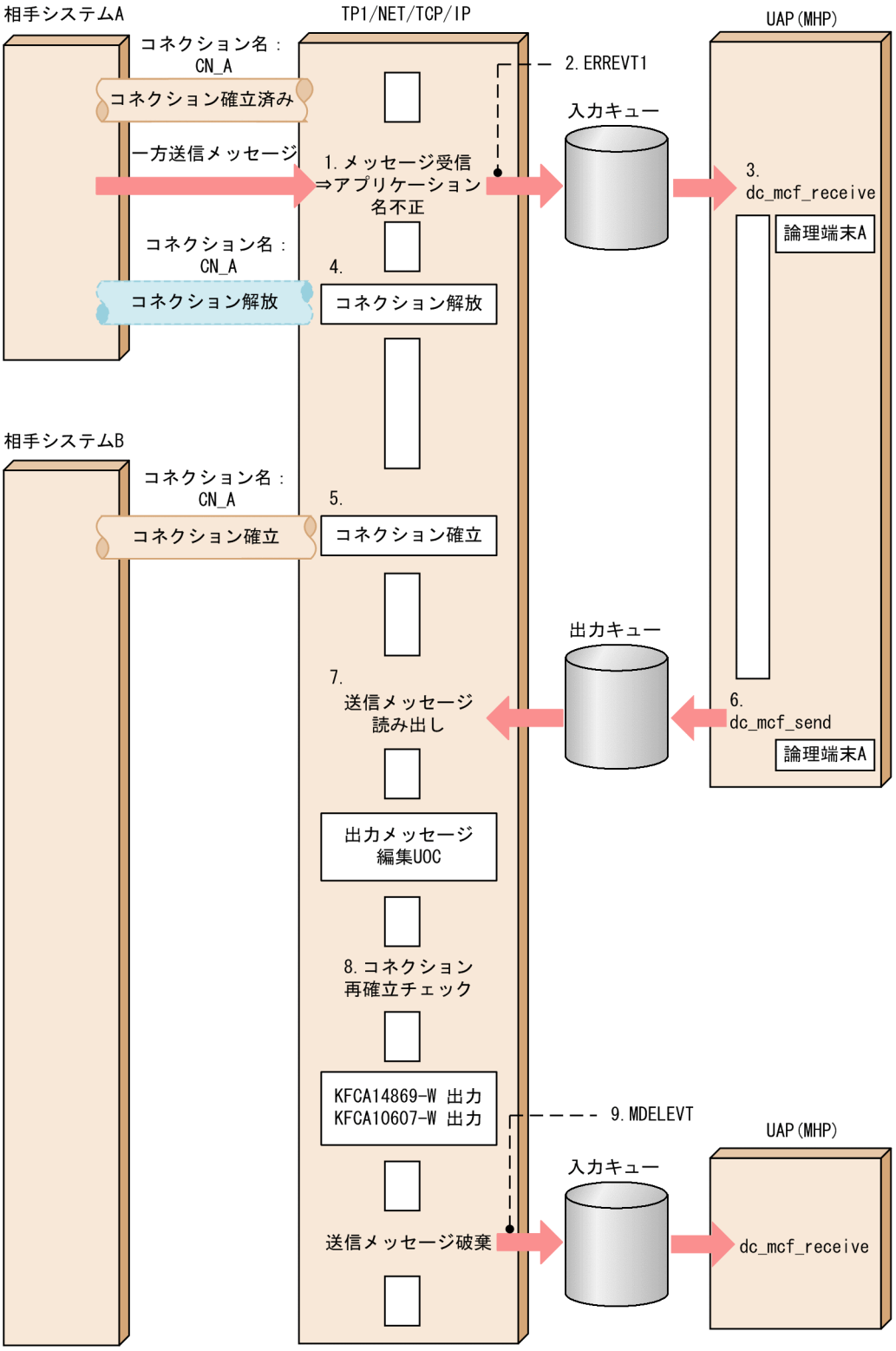
7. 論理端末の閉塞解除を契機に，出力キューからメッセージを読み出します。

8. TP1/NET/TCP/IP がコネクションの再確立を検出したため，出力キューのメッセージを破棄します。

9. MDELEVT を起動します。

(b)　ERREVT1 の場合

図 2-64　ERREVT1 処理用 MHP からメッセージ送受信関数を発行したときの処理の流れ



1. 相手システム A（コネクション名：CN_A）からメッセージを受信し，アプリケーション名不正を検出します。
2. ERREVT1 を起動します。

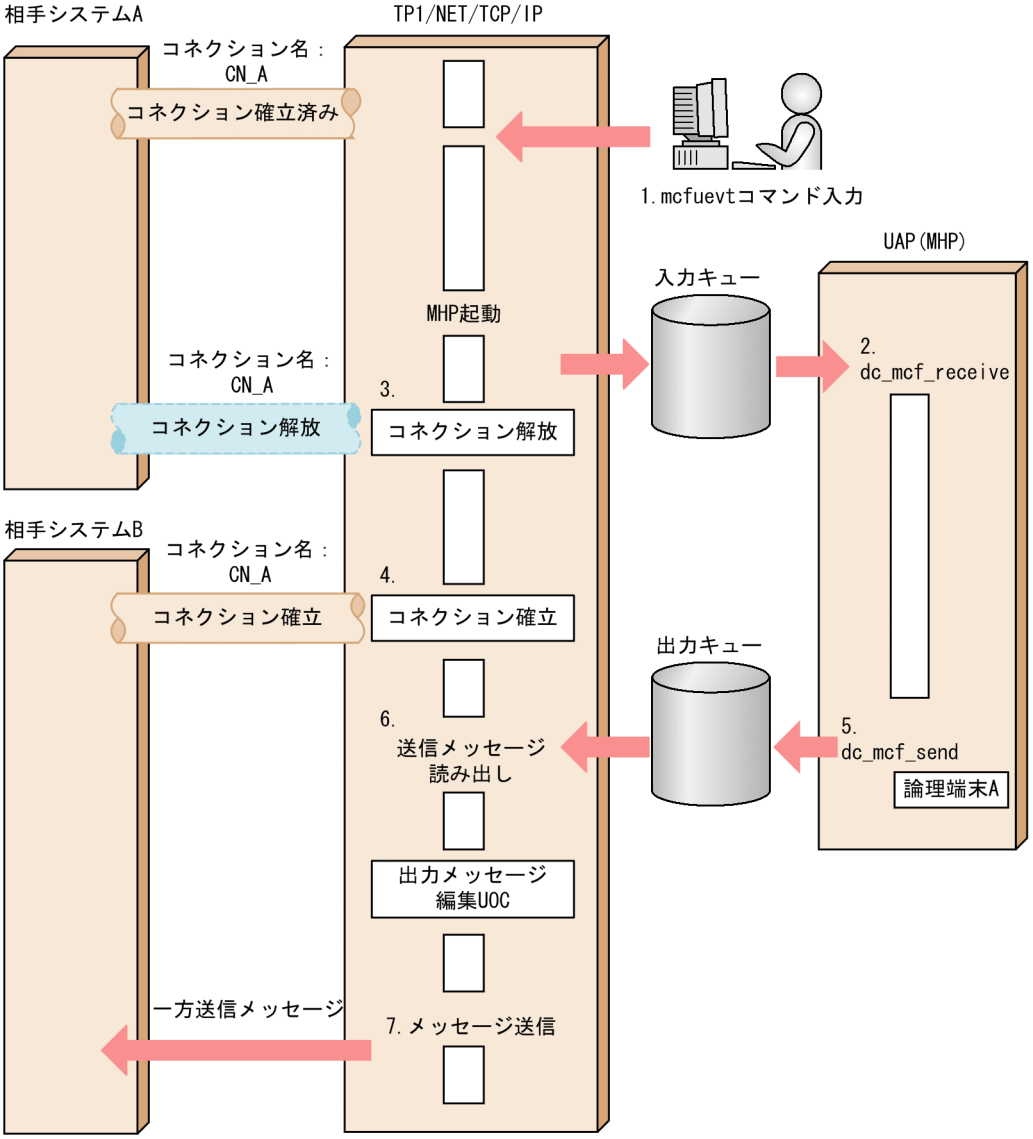
3. dc_mcf_receive 関数を発行してメッセージを受け取ります。
4. コネクション CN_A を解放します。
5. CN_A が相手システム B とコネクションを再確立します。
6. dc_mcf_send 関数を発行して出力キューにメッセージを格納します。
7. 論理端末の閉塞解除を契機に，出力キューからメッセージを読み出します。
8. TP1/NET/TCP/IP がコネクションの再確立を検出したため，出力キューのメッセージを破棄します。
9. MDELEVT を起動します。

(8) mcfuevt コマンドによって起動した MHP からメッセージ送受信関数を発行した場合

mcfuevt コマンドで起動された MHP からメッセージ送受信関数を発行した場合，TP1/NET/TCP/IP は相手システムとのコネクションが再確立したかどうかをチェックしません。

mcfuevt コマンドで起動した MHP からのメッセージ送受信関数発行を，dc_mcf_send 関数を使用する場合を例に，次の図に示します。

図 2-65　mcfuevt コマンドより起動した MHP からメッセージ送受信関数を発行したときの処理の流れ



1. mcfuevt コマンドを入力して MHP を起動します。
2. dc_mcf_receive 関数を発行してメッセージを受け取ります。
3. コネクション CN_A を解放します。
4. CN_A が相手システム B とコネクションを再確立します。
5. dc_mcf_send 関数を発行して出力キューにメッセージを格納します。
6. 論理端末の閉塞解除を契機に，出力キューからメッセージを読み出します。
7. メッセージ送受信関数を発行した MHP が mcfuevt コマンドによって起動したため，コネクションが再確立したかどうかをチェックしないで，相手システム B にメッセージを送信します。

3 C 言語のライブラリ関数

この章では，TP1/NET/TCP/IP で使用できる，C 言語のライブラリ関数について説明します。

C 言語のライブラリ関数の一覧

TP1/NET/TCP/IP で使用する C 言語のライブラリ関数の一覧について説明します。

C 言語のライブラリ関数に対応する機能の一覧を次の表に示します。

表 3-1　C 言語のライブラリ関数に対応する機能の一覧

関数名	機能
dc_mcf_contend	継続問い合わせ応答の終了
dc_mcf_receive	メッセージの受信
dc_mcf_recvsync	同期型メッセージの受信
dc_mcf_reply	応答メッセージの送信
dc_mcf_resend	メッセージの再送
dc_mcf_send	一方送信メッセージの送信
dc_mcf_sendrecv	同期型メッセージの送受信
dc_mcf_sendsync	同期型メッセージの送信
dc_mcf_tactcn	コネクションの確立
dc_mcf_tactle	論理端末の閉塞解除
dc_mcf_tdctcn	コネクションの解放
dc_mcf_tdctle	論理端末の閉塞
dc_mcf_tempget	一時記憶データの受け取り
dc_mcf_tempput	一時記憶データの更新
dc_mcf_tlscn	コネクションの状態取得
dc_mcf_tlsle	論理端末の状態取得
dc_mcf_tlsln	サーバ型コネクションの確立要求の受付状態取得
dc_mcf_tofln	サーバ型コネクションの確立要求の受付終了
dc_mcf_tonln	サーバ型コネクションの確立要求の受付開始

なお，UAP 作成の詳細については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。その他の関数については，マニュアル「OpenTP1 プログラム作成リファレンス C 言語編」を参照してください。

NULL またはヌル文字列設定時のコーディング例

C 言語のライブラリ関数の引数に NULL またはヌル文字列を設定する場合のコーディング例を示します。

NULL を設定する場合

```
char *resv01=NULL;
dc_mcf_receive(…, resv01, …);
```

ヌル文字列を設定する場合

```
char resv01[1]="¥0";
dc_mcf_receive(…, resv01, …);
```

注

resv01 以外の dc_mcf_receive 関数の引数は省略しています。

dc_mcf_contend － 継続問い合わせ応答の終了（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_contend(DCLONG action, char *resv01)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int  dc_mcf_contend(action, resv01)
DCLONG    action;
char      *resv01;
```

機能

継続問い合わせ応答を終了します。

UAP で値を設定する引数

●action

DCNOFLAGS を設定します。

●resv01

ヌル文字列を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_72000	-13000	＜ MHP の実行でリターンした場合＞ 先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_contend 関数を呼び出しています。
		＜ SPP の実行でリターンした場合＞ SPP では dc_mcf_contend 関数を呼び出せません。
DCMCFRTN_72016	-13016	action に設定した値が間違っています。
		resv01 に設定した値が間違っています。
DCMCFRTN_72101	-13101	継続問い合わせ応答型でないアプリケーションで，dc_mcf_contend 関数を呼び出しています。
DCMCFRTN_72107	-13107	dc_mcf_contend 関数を 2 回以上呼び出しています。

リターン値	リターン値（数値）	意味
DCMCFRTN_72111	-13111	次起動アプリケーションを設定して dc_mcf_reply 関数を呼び出したあと，dc_mcf_contend 関数を呼び出しています。
		継続問い合わせ応答型のアプリケーション名を設定して dc_mcf_execap 関数を呼び出したあと，dc_mcf_contend 関数を呼び出しています。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_receive － メッセージの受信（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_receive(DCLONG action, DCLONG commform,
                    char *termnam, char *resv01,
                    char *recvdata, DCLONG *rdataleng,
                    DCLONG inbufleng, DCLONG *time)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int  dc_mcf_receive(action, commform, termnam, resv01, recvdata,
                    rdataleng, inbufleng, time)
DCLONG    action;
DCLONG    commform;
char      *termnam;
char      *resv01;
char      *recvdata;
DCLONG    *rdataleng;
DCLONG    inbufleng;
DCLONG    *time;
```

機能

論理端末に届いたメッセージのうち，一つのセグメントを受信します。セグメントの数だけ dc_mcf_receive 関数を呼び出すと，一つの論理メッセージを受信できます。

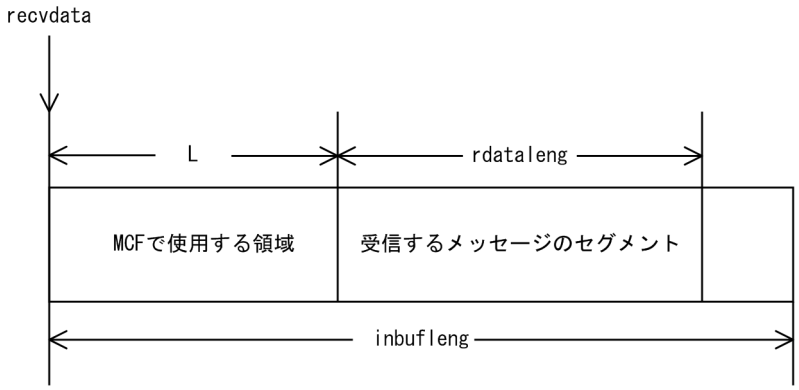
受信できるメッセージの一つのセグメントの最大長は，1 メガバイトです。

dc_mcf_receive 関数で受信できるメッセージの種類を次に示します。

- 相手システムから送信されたメッセージ
- MCF イベント
- アプリケーション起動で渡されたメッセージ

TP1/NET/TCP/IP を使用して通信する場合，相手システムから送信されるメッセージは，常に単一セグメントで構成されます。

セグメントを受信する領域の形式を次に示します。L は，バッファ形式 1 の場合は 8 バイト，バッファ形式 2 の場合は 4 バイトです。



UAP で値を設定する引数

●action

メッセージの先頭セグメントを受信するかどうか，および使用するバッファ形式を，次の形式で設定します。

```
{DCMCFFRST | DCMCFSEG}  〔|  {DCMCFBUF1 | DCMCFBUF2} 〕
```

DCMCFFRST

先頭セグメントを受信する場合や，メッセージが単一セグメントの場合に設定します。

DCMCFSEG

中間セグメントまたは最終セグメントを受信する場合に設定します。

DCMCFBUF1

バッファ形式 1 を使用する場合に設定します。

DCMCFBUF2

バッファ形式 2 を使用する場合に設定します。

●commform

DCNOFLAGS を設定します。

●termnam

先頭セグメントまたは単一セグメントを受信する場合は，メッセージ入力元の論理端末名称を受け取る領域を設定します。

処理終了後，termnam には OpenTP1 から値が返されます。

中間セグメントまたは最終セグメントを受信する場合は，先頭セグメントの受信時に返されたメッセージ入力元の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。論理端末名称の最後にはヌル文字を付けてください。

中間セグメントまたは最終セグメントを受信した場合，値は返されません。

●resv01

NULL またはヌル文字列を設定します。

●recvdata

セグメントを受信する領域を設定します。

dc_mcf_receive 関数が終了すると，メッセージのセグメントの一つが返されます。

処理終了後，recvdata には OpenTP1 から値が返されます。

●inbufleng

セグメントを受信する領域の長さを設定します。

OpenTP1 から値が返される引数

●termnam

先頭セグメントまたは単一セグメントを受信する場合，入力元の論理端末名称が返されます。論理端末名称は最大 8 バイトの長さです。論理端末名称の最後にはヌル文字が付けられます。

中間セグメントまたは最終セグメントの受信する場合は，ここで返された論理端末名称を termnam に設定します。

●recvdata

受信したセグメントの内容が返されます。

●rdataleng

受信したセグメントの長さが返されます。

●time

メッセージを受信した時刻が，1970 年 1 月 1 日 0 時 0 分 0 秒からの通算の秒数で返されます。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71000	-12000	先頭セグメントを受信する dc_mcf_receive 関数を 2 回以上呼び出しています。中間セグメントまたは最終セグメントを受信する場合は，action に DCMCFSEG を設定して dc_mcf_receive 関数を呼び出してください。
DCMCFRTN_71001	-12001	メッセージの最終セグメントを受信したあとで，次のセグメントを受信する dc_mcf_receive 関数を呼び出しています。直前に呼び出した dc_mcf_receive 関数でメッセージはすべて受信しました。

リターン値	リターン値（数値）	意味
DCMCFRTN_71001	-12001	このリターン値が返されたあとに，再び dc_mcf_receive 関数を呼び出した場合は，リターン値 DCMCFRTN_72000 が返されます。
DCMCFRTN_71002	-12002	メッセージキューからの入力処理中に障害が発生しました。
		メッセージキューが閉塞されています。
DCMCFRTN_72000	-13000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，中間セグメントまたは最終セグメントを受信する dc_mcf_receive 関数を呼び出しています。先頭セグメントを受信する場合は，action に DCMCFFRST を設定して dc_mcf_receive 関数を呼び出してください。 • リターン値 DCMCFRTN_71001 が返されたあとに，再び dc_mcf_receive 関数を呼び出しています。
		＜ SPP の実行でリターンした場合＞ SPP では dc_mcf_receive 関数を呼び出せません。
DCMCFRTN_72001	-13001	termnam に設定した論理端末名称が間違っています。
DCMCFRTN_72013	-13013	inbufleng の指定値を超えるセグメントを受信しました。inbufleng の指定値を超えた部分は切り捨てられました。
DCMCFRTN_72016	-13016	action に設定した値が間違っています。
		resv01 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。
DCMCFRTN_72025	-13025	action に設定したセグメント種別（DCMCFFRST または DCMCFSEG）の値が間違っています。
DCMCFRTN_72036	-13036	inbufleng の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_recvsync － 同期型メッセージの受信（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_recvsync(DCLONG action, DCLONG commform,
                     char *termnam, char *resv01,
                     char *recvdata, DCLONG *rdataleng,
                     DCLONG inbufleng, DCLONG *time,
                     DCLONG watchtime)
```

K&R 版 C の形式

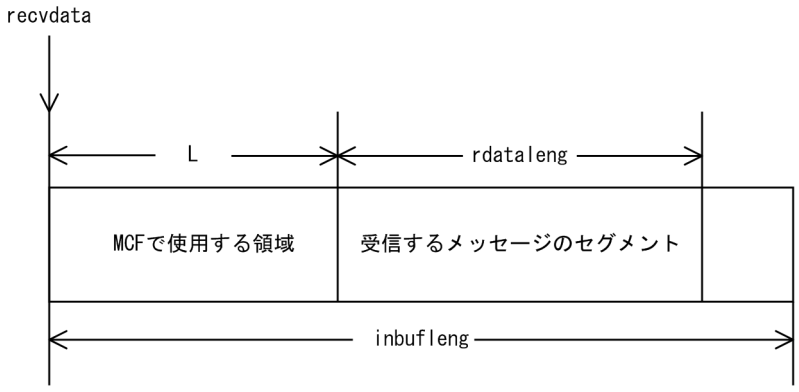
```
#include <dcmcf.h>
int  dc_mcf_recvsync(action, commform, *termnam,
                     resv01, recvdata, rdataleng,
                     inbufleng, time, watchtime)
DCLONG     action;
DCLONG     commform;
char       *termnam;
char       *resv01;
char       *recvdata;
DCLONG     *rdataleng;
DCLONG     inbufleng;
DCLONG     *time;
DCLONG     watchtime;
```

機能

相手システムから同期型でメッセージを受信します。相手システムから送信されるメッセージは，常に一つのセグメントで構成されます。

受信できるメッセージの一つのセグメントの最大長は，1 メガバイトです。

セグメントを受信する領域の形式を次に示します。L は，バッファ形式 1 の場合は 8 バイト，バッファ形式 2 の場合は 4 バイトです。



UAP で値を設定する引数

●action

受信する論理メッセージのセグメントと，使用するバッファ形式を次の形式で指定します。

```
DCMCFFRST〔| {DCMCFBUF1|DCMCFBUF2} 〕
```

DCMCFFRST

単一セグメントを示す DCMCFFRST を設定します。

DCMCFBUF1

バッファ形式 1 を使用する場合に設定します。

DCMCFBUF2

バッファ形式 2 を使用する場合に設定します。

●commform

DCNOFLAGS を設定します。

●termnam

入力元の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。論理端末名称の最後にはヌル文字を付けてください。

●resv01

NULL またはヌル文字列を設定します。

●recvdata

セグメントを受信する領域を設定します。dc_mcf_recvsync 関数が終了すると，受信したセグメントが返されます。

処理終了後，recvdata には OpenTP1 から値が返ります。

●inbufleng

セグメントを受信する領域の長さを設定します。

●watchtime

dc_mcf_recvsync 関数を呼び出してから終了するまでの最大時間を設定します。0 を設定した場合，MCF マネジャ定義の UAP 共通定義で指定した同期型受信監視時間（mcfmuap -t recvtim）が設定されます。

負の値を設定した場合は，時間を監視しません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，設定した監視時間と実際にタイ

ムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，設定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

OpenTP1 から値が返される引数

●recvdata

受信したメッセージの先頭セグメントの内容が返されます。

●rdataleng

受信したメッセージの先頭セグメントの長さが返されます。

●time

メッセージを受信した時刻が，1970 年 1 月 1 日 0 時 0 分 0 秒からの通算の秒数で返されます。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	メッセージの最終セグメントを受信したあとで，次のセグメントを受信する dc_mcf_recvsync 関数を呼び出しています。直前に呼び出した dc_mcf_recvsync 関数でメッセージはすべて受信しました。 このリターン値が返されたあとに，再び dc_mcf_recvsync 関数を呼び出した場合は，リターン値 DCMCFRTN_72000 が返されます。
DCMCFRTN_71002	-12002	MCF が終了処理中のため，メッセージの受信を受け付けられません。
DCMCFRTN_71108	-12108	メッセージ受信に必要な管理テーブルが確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_recvsync 関数を呼び出しています。
DCMCFRTN_72001	-13001	termnam に設定した論理端末名称が間違っています。
		termnam に設定した入力元の論理端末名称は，MCF で定義していません。
		dc_mcf_recvsync 関数を呼び出せない論理端末を設定しています。
		＜問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用している（コネクション定義（mcftalccn -l）の replymsg オペランドに yes を指定）場合＞ 問い合わせ応答中または継続問い合わせ応答中のため受け付けられません。
DCMCFRTN_72012	-13012	MCF バッファグループ定義のバッファ長が不足しました。
		MCF 通信プロセスは相手システムからメッセージを受信しましたが，UAP への応答連絡で RPC 通信の送信可能な上限値を超えました。

リターン値	リターン値（数値）	意味
DCMCFRTN_72013	-13013	inbufleng の指定値を超えるセグメントを受信しました。inbufleng の指定値を超えた部分は切り捨てられました。
DCMCFRTN_72016	-13016	action に設定した値が間違っています。
		resv01 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。
DCMCFRTN_72025	-13025	action に設定したセグメント種別（DCMCFFRST）の値が間違っています。
DCMCFRTN_72036	-13036	inbufleng の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
DCMCFRTN_73001	-14001	watchtime に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
		前回の通信から無通信監視時間（コネクション定義（mcftalccn -k）の notrftime オペランド指定値）が経過しましたが，相手システムからの通信がありません。
		相手システムからメッセージを受信しましたが，受信バッファを確保できませんでした。
		入力元の論理端末で MCF 通信プロセスの内部障害が発生しました。
DCMCFRTN_73002	-14002	コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指定）場合に，MHP でメッセージ受信後にコネクションが再確立されたため，受信を取り消しました。
DCMCFRTN_73005	-14005	watchtime に設定した時間が経過しましたが，論理端末からの応答がありません。
DCMCFRTN_73010	-14010	入力メッセージ編集 UOC で障害が発生しました。
		メッセージの読み込み時に障害が発生しました。
DCMCFRTN_73015	-14015	指定した論理端末は，ほかの UAP で仕掛り中です。
DCMCFRTN_73018	-14018	watchtime に設定した値が間違っています。
DCMCFRTN_73020	-14020	入力元の論理端末は停止しています。
上記以外	−	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

−：該当しません。

dc_mcf_reply － 応答メッセージの送信（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_reply(DCLONG action, DCLONG commform, char *resv01,
                  char *resv02, char *senddata, DCLONG sdataleng,
                  char *nextap, DCLONG opcd)
```

K&R 版 C の形式

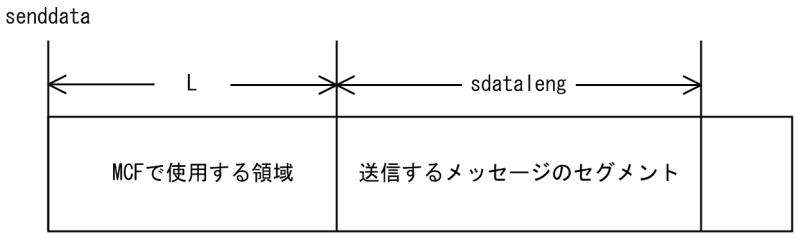
```
#include <dcmcf.h>
int  dc_mcf_reply(action, commform, resv01, resv02, senddata,
                  sdataleng, nextap, opcd)
DCLONG    action;
DCLONG    commform;
char      *resv01;
char      *resv02;
char      *senddata;
DCLONG    sdataleng;
char      *nextap;
DCLONG    opcd;
```

機能

メッセージを入力した論理端末に対して応答メッセージを送信します。応答メッセージは，一つのセグメントで構成されます。

送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

セグメントを送信する領域の形式を次に示します。L は，バッファ形式 1 の場合は 8 バイト，バッファ形式 2 の場合は 4 バイトです。



UAP で値を設定する引数

●action

単一セグメントを送信すること，出力通番を付けるかどうか，および使用するバッファ形式を，次の形式で設定します。

```
DCMCFEMI〔| {DCMCFSEQ|DCMCFNSEQ}〕〔| {DCMCFBUF1|DCMCFBUF2}〕
```

DCMCFEMI

単一セグメントを示す DCMCFEMI を設定します。

DCMCFSEQ

出力通番が必要な場合に設定します。

ただし，非応答型のアプリケーションの場合，出力通番を付けません。

DCMCFNSEQ

出力通番が必要ない場合に設定します。

DCMCFBUF1

バッファ形式 1 を使用する場合に設定します。

DCMCFBUF2

バッファ形式 2 を使用する場合に設定します。

●commform

DCNOFLAGS を設定します。

●resv01

ヌル文字列を設定します。

●resv02

NULL またはヌル文字列を設定します。

●senddata

送信するセグメントの内容を設定した領域を設定します。一つのセグメントで 32000 バイトまで送信できます。

●sdataleng

送信するセグメントの長さを設定します。

●nextap

＜継続問い合わせ応答形態の場合＞

次起動アプリケーションを設定します。次起動アプリケーションは最大 8 バイトの長さです。次起動アプリケーションの最後にはヌル文字を付けてください。

ヌル文字列を設定した場合，実行中のアプリケーションを次のメッセージ受信時に再び起動します。

dc_mcf_reply 関数を発行するサービスで dc_mcf_contend 関数を呼び出す場合，nextap にはヌル文字列を設定してください。

継続問い合わせ応答を引き継いだエラーイベントで，dc_mcf_reply 関数を発行する際に，nextap にヌル文字列を設定した場合，継続問い合わせ応答を終了します。ただし，継続問い合わせ応答を引き継

いだエラーイベントで，dc_mcf_execap 関数を発行して継続問い合わせ応答型のアプリケーションを起動し，起動先のアプリケーションで dc_mcf_reply 関数を発行する際に，nextap にヌル文字列を設定した場合，継続問い合わせ応答を終了しないで，dc_mcf_execap 関数を発行したときに設定した継続問い合わせ応答型のアプリケーションを次のメッセージ受信時に再び起動します。

＜継続問い合わせ応答形態以外の場合＞

ヌル文字列を設定します。

●opcd

DCNOFLAGS を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71002	-12002	メッセージキューへの出力処理中に障害が発生しました。
		メッセージキューが閉塞されています。
		メッセージキューが割り当てられていません。
		sdataleng に 32000 バイトを超える値を設定しています。
		MCF が終了処理中のため，メッセージの送信を受け付けられません。
DCMCFRTN_71003	-12003	メッセージキューが満杯です。
DCMCFRTN_71004	-12004	メッセージを格納するバッファをメモリ上に確保できませんでした。
DCMCFRTN_71108	-12108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_reply 関数を呼び出しています。 • 非応答型のアプリケーションからの問い合わせ応答をしない（UAP 共通定義（mcfmuap -c）の noansreply オペランドに no を指定）場合に，非応答型のアプリケーションから dc_mcf_reply 関数を呼び出しています。
		＜ SPP の実行でリターンした場合＞ SPP では dc_mcf_reply 関数を呼び出せません。
DCMCFRTN_72001	-13001	入力元論理端末は，応答メッセージの送信をサポートしていません。
		入力元論理端末は，問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用していません（コネクション定義（mcftalccn -l）の replymsg オペランドを省略，または no を指定）。

リターン値	リターン値（数値）	意味
DCMCFRTN_72008	-13008	応答型のアプリケーションから dc_mcf_execap 関数を呼び出して応答型のアプリケーションを起動したあとで，dc_mcf_reply 関数を呼び出しています。
		継続問い合わせ応答型のアプリケーションから dc_mcf_execap 関数を呼び出して継続問い合わせ応答型のアプリケーションを起動したあとで，dc_mcf_reply 関数を呼び出しています。
DCMCFRTN_72011	-13011	継続問い合わせ応答型でないアプリケーションが，次起動アプリケーションを設定して，dc_mcf_reply 関数を呼び出しています。
DCMCFRTN_72016	-13016	opcd に設定した値が間違っています。
		resv01，または resv02 に設定した値が間違っています。
		nextap に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72026	-13026	action に設定したセグメント種別（DCMCFEMI）の値が間違っています。
DCMCFRTN_72041	-13041	sdataleng に 0 バイト，またはマイナス値を設定しています。
DCMCFRTN_72044	-13044	dc_mcf_contend 関数を呼び出したあとで，次起動アプリケーションを設定して dc_mcf_reply 関数を呼び出しています。
DCMCFRTN_72045	-13045	nextap に，継続問い合わせ応答型でないアプリケーションのアプリケーション名を設定して dc_mcf_reply 関数を呼び出しています。
DCMCFRTN_72046	-13046	nextap に設定したアプリケーション名が異なる dc_mcf_reply 関数を，2 回以上呼び出しています。
DCMCFRTN_72047	-13047	nextap に，アプリケーション属性定義（mcfaalcap）に定義されていないアプリケーション名を設定して dc_mcf_reply 関数を呼び出しています。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_resend － メッセージの再送（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_resend(DCLONG action, DCLONG commform, char *rtermnam,
                   char *resv01, DCLONG oseqid, DCLONG orgseq,
                   char *otermnam, char *resv02, char *resv03,
                   char *resv04, DCLONG opcd)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int  dc_mcf_resend(action, commform, rtermnam, resv01, oseqid,
                   orgseq, otermnam, resv02, resv03, resv04,
                   opcd)
DCLONG    action;
DCLONG    commform;
char      *rtermnam;
char      *resv01;
DCLONG    oseqid;
DCLONG    orgseq;
char      *otermnam;
char      *resv02;
char      *resv03;
char      *resv04;
DCLONG    opcd;
```

機能

以前に送信したメッセージを，再び送信します。再送するメッセージは，以前に送信したメッセージとは別の，新しいメッセージとして扱います。どのメッセージを再送するかは，次に示す送信済みメッセージの情報で選択できます。

- 出力先の論理端末名称
- メッセージ出力通番
- メッセージ種別（一般の一方送信，優先の一方送信）

対象としたメッセージが以前に送信されていない場合は，dc_mcf_resend 関数はリターン値 DCMCFRTN_NOMSG を返します。また，メッセージキュー（ディスクキュー）内に対象のメッセージがない場合も，リターン値 DCMCFRTN_NOMSG を返します。このため，使用するメッセージキューの種別ではディスクキューを指定して，メッセージキューの大きさの定義は余裕を持った値を指定してください。

UAP で値を設定する引数

●action

再送するメッセージに出力通番を付け直すかどうか，一般か優先か，および最終出力通番のメッセージを再送するかどうかを，次の形式で設定します。

```
{DCMCFSEQ｜DCMCFNSEQ} 〔｜ {DCMCFNORM｜DCMCFPRIO} 〕 〔｜DCMCFLAST〕
```

DCMCFSEQ

再送するメッセージに出力通番を付け直す場合に設定します。

DCMCFNSEQ

再送するメッセージに出力通番を付け直さない場合に設定します。

DCMCFNORM

一般の一方送信メッセージとして再送する場合に設定します。

DCMCFPRIO

優先の一方送信メッセージとして再送する場合に設定します。

DCMCFLAST

最終出力通番を持つメッセージを再送する場合に設定します。この値を設定した場合は，orgseq に設定した値は無効となります。

●commform

一方送信を示す，DCMCFOUT を設定します。

●rtermnam

出力先の論理端末名称を設定します。論理端末名称は最大８バイトの長さです。論理端末名称の最後にはヌル文字を付けてください。

●resv01

NULL を設定します。

●oseqid

再送するメッセージを検索するキーとして，以前に送信したメッセージの送信種別を設定します。

DCMCFRID_NORM

一般の一方送信メッセージを対象とする場合に設定します。

DCMCFRID_PRIO

優先の一方送信メッセージを対象とする場合に設定します。

省略した場合は，DCMCFRID_NORM（一般の一方送信メッセージを対象）が設定されます。

●orgseq

再送するメッセージを検索するキーとして，以前に送信したメッセージの出力通番を設定します。action で DCMCFLAST を設定した場合は，ここに設定した値は無効となります。

●otermnam

再送するメッセージを検索するキーとして，以前に送信したメッセージの出力先の論理端末名称を設定します。論理端末名称は最大８バイトの長さです。論理端末名称の最後にはヌル文字を付けてください。

●resv02，resv03，resv04

NULL を設定します。

●opcd

DCNOFLAGS を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_NOMSG	-11904	出力通番使用論理端末数（MCF マネジャ共通定義（mcfmcomn）の-n オプション）を省略，または 0 を指定しています。
		otermnam，oseqid，または orgseq に設定した値が間違っています。
		再送するメッセージは次に示す理由によって，再送できるメッセージの条件を満たしていません。 • 再送対象とするメッセージの送信時に出力通番を付けていません。 • 出力メッセージの割り当て先にメモリキューを使用しています（論理端末定義（mcftalcle -k）の quekind オペランドを省略，または memory を指定）。 • 論理端末が閉塞しているなどの要因によって，送信メッセージが送信済みになっていません。 • 送信済みのメッセージがディスクキューに保持されていません（保持メッセージ数は，メッセージキューサービス定義の quegrp コマンドの-m オプションで指定します）。
		otermnam で指定した論理端末に割り当てられているメッセージキューは，システム開始時に障害が発生したため，メモリキューを代用して縮退運転をしています。
DCMCFRTN_BUF_SHORT	-11905	再送するメッセージのセグメントの長さが，UAP 共通定義（mcfmuap -e）で指定した値を超えています。
DCMCFRTN_71002	-12002	メッセージキューへの出力処理中に障害が発生しました。
		メッセージキューが閉塞されています。
		メッセージキューが割り当てられていません。

リターン値	リターン値（数値）	意味
DCMCFRTN_71002	-12002	MCF が終了処理中のため，メッセージの再送を受け付けられません。
DCMCFRTN_71003	-12003	メッセージキューが満杯です。
DCMCFRTN_71004	-12004	メッセージキューから取り出したメッセージを格納するバッファ（作業領域）を，メモリ上に確保できませんでした。
DCMCFRTN_71108	-12108	メッセージを再送しようとしましたが，再送先の管理テーブルが確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_resend 関数を呼び出しています。 • 非トランザクション属性の MHP から，dc_mcf_resend 関数を呼び出しています。
		＜ SPP の実行でリターンした場合＞ トランザクションでない SPP の処理から，dc_mcf_resend 関数を呼び出しています。
DCMCFRTN_72001	-13001	rtermnam または otermnam に設定した論理端末名称が間違っています。
		dc_mcf_resend 関数を呼び出せない論理端末を設定しています。
DCMCFRTN_72016	-13016	action に設定したメッセージ種別（DCMCFNORM または DCMCFPRIO）の値が間違っています。
		action に設定した値が間違っています。
		oseqid に設定した値が間違っています。
		opcd に設定した値が間違っています。
		resv01，resv02，resv03，または resv04 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72017	-13017	action に設定した出力通番の要否（DCMCFSEQ または DCMCFNSEQ）の値が間違っています。
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

注意事項

メッセージの再送時には，MCF マネジャ定義の UAP 共通定義（mcfmuap）の-e オプションと-l オプションの指定値に注意してください。

-e オプション

-e オプションでは，dc_mcf_resend 関数で使用する作業領域の大きさを指定します。再送するメッセージのセグメントがこの作業領域より大きい場合，dc_mcf_resend 関数はメッセージを再送しないで，リターン値 DCMCFRTN_BUF_SHORT を返します。このため，-e オプションでは，セグメントの最大長よりも大きな値を設定しておいてください。

-l オプション

-l オプションでは，出力通番に関して指定します。この内容によっては，メッセージキューファイル内に同じ出力通番を持つメッセージが同時に存在する場合があります。この場合は，どのメッセージを再送するか保証できません。

dc_mcf_send － 一方送信メッセージの送信（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_send(DCLONG action, DCLONG commform, char *termnam,
                 char *resv01, char *senddata,
                 DCLONG sdataleng, char *resv02, DCLONG opcd)
```

K&R 版 C の形式

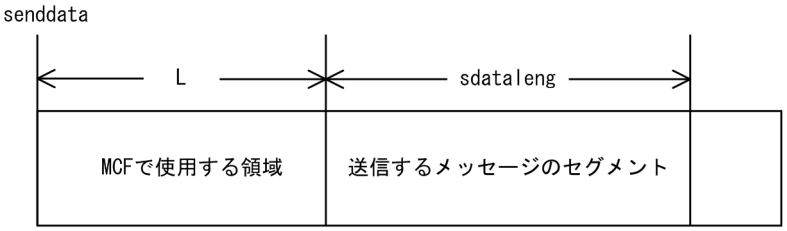
```
#include <dcmcf.h>
int  dc_mcf_send(action, commform, termnam, resv01, senddata,
                 sdataleng, resv02, opcd)
DCLONG    action;
DCLONG    commform;
char      *termnam;
char      *resv01;
char      *senddata;
DCLONG    sdataleng;
char      *resv02;
DCLONG    opcd;
```

機能

相手システムへ一方送信メッセージを送信します。一方送信メッセージは，一つのセグメントで構成されます。

送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

セグメントを送信する領域の形式を次に示します。L は，バッファ形式 1 の場合は 8 バイト，バッファ形式 2 の場合は 4 バイトです。



UAP で値を設定する引数

●action

単一セグメントを送信すること，優先か一般か，出力通番を付けるかどうか，使用するバッファ形式，および送信完了通知イベントを通知させるかどうかを，次の形式で設定します。

```
DCMCFEMI 〔|  {DCMCFNORM | DCMCFPRIO} 〕
 〔|  {DCMCFSEQ | DCMCFNSEQ} 〕 〔|  {DCMCFBUF1 | DCMCFBUF2} 〕
 〔| DCMCFSEVT〕
```

DCMCFEMI

単一セグメントを示す DCMCFEMI を設定します。

DCMCFNORM

一般の一方送信メッセージとして送信する場合に設定します。

DCMCFPRIO

優先の一方送信メッセージとして送信する場合に設定します。

DCMCFSEQ

出力通番が必要な場合に設定します。

DCMCFNSEQ

出力通番が必要ない場合に設定します。

DCMCFBUF1

バッファ形式 1 を使用する場合に設定します。

DCMCFBUF2

バッファ形式 2 を使用する場合に設定します。

DCMCFSEVT

送信完了通知イベントを通知させるときに設定します。ただし，送信完了通知イベント処理用の MHP を MCF アプリケーション定義で指定していない場合は無効です。

●commform

一方送信を示す，DCMCFOUT を設定します。

●termnam

出力先の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。論理端末名称の最後にはヌル文字を付けてください。

●resv01

NULL またはヌル文字列を設定します。

●senddata

送信するセグメントの内容を設定した領域を設定します。一つのセグメントで 32000 バイトまで送信できます。

●sdataleng

送信するセグメントの長さを設定します。

●resv02

NULL またはヌル文字列を設定します。

●opcd

DCNOFLAGS を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71002	-12002	メッセージキューへの出力処理中に障害が発生しました。
		メッセージキューが閉塞されています。
		メッセージキューが割り当てられていません。
		sdataleng に 32000 バイトを超える値を設定しています。
		MCF が終了処理中のため，メッセージの送信を受け付けられません。
DCMCFRTN_71003	-12003	メッセージキューが満杯です。
DCMCFRTN_71004	-12004	メッセージを格納するバッファをメモリ上に確保できませんでした。
DCMCFRTN_71108	-12108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	＜ MHP の実行でリターンした場合＞ 先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_send 関数を呼び出しています。
		＜ SPP の実行でリターンした場合＞ トランザクションでない SPP の処理から，dc_mcf_send 関数を呼び出しています。
DCMCFRTN_72001	-13001	dc_mcf_send 関数を呼び出せない論理端末を設定しています。
		termnam に設定した論理端末名称が間違っています。
DCMCFRTN_72016	-13016	action に設定したメッセージ種別（DCMCFNORM または DCMCFPRIO）の値が間違っています。
		action に設定した値が間違っています。
		opcd に設定した値が間違っています。
		resv01 または resv02 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72017	-13017	action に設定した出力通番の要否（DCMCFSEQ または DCMCFNSEQ）の値が間違っています。

リターン値	リターン値（数値）	意味
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。
DCMCFRTN_72026	-13026	action に設定したセグメント種別（DCMCFEMI）の値が間違っています。
DCMCFRTN_72041	-13041	sdataleng に 0 バイト，またはマイナス値を設定しています。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_sendrecv － 同期型メッセージの送受信（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_sendrecv(DCLONG action, DCLONG commform,
                     char *termnam, char *resv01,
                     char *senddata, DCLONG sdataleng,
                     char *recvdata, DCLONG *rdataleng,
                     DCLONG inbufleng, DCLONG *time,
                     DCLONG watchtime)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int  dc_mcf_sendrecv(action, commform, termnam, resv01,
                     senddata, sdataleng, recvdata, rdataleng,
                     inbufleng, time, watchtime)
DCLONG    action;
DCLONG    commform;
char      *termnam;
char      *resv01;
char      *senddata;
DCLONG    sdataleng;
char      *recvdata;
DCLONG    *rdataleng;
DCLONG    inbufleng;
DCLONG    *time;
DCLONG    watchtime;
```

機能

同期型でメッセージを送信したあと，同期型でメッセージを受信します。

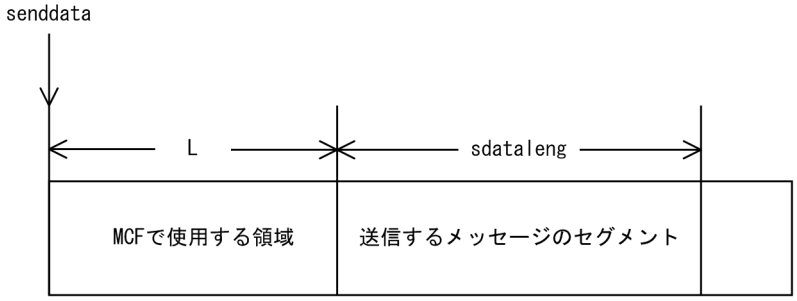
dc_mcf_sendrecv 関数では，相手システムへ送るメッセージのセグメントを送信できます。

メッセージのセグメントを送信すると，dc_mcf_sendrecv 関数は相手システムからの応答を待ちます。応答が届くと，そのメッセージのセグメントを受信します。

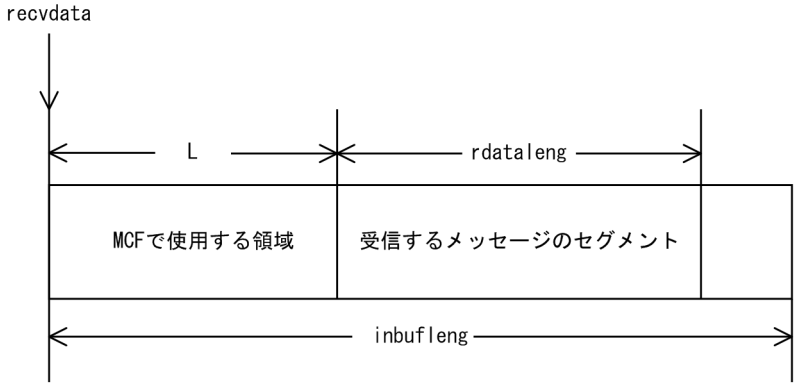
受信できるメッセージの一つのセグメントの最大長は，1 メガバイトです。また，送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

セグメントを送信する領域と受信する領域の形式をそれぞれ示します。L は，バッファ形式 1 の場合は 8 バイト，バッファ形式 2 の場合は 4 バイトです。

●セグメントを送信する領域



●セグメントを受信する領域



UAP で値を設定する引数

●action

メッセージの最終セグメントを送信するかどうか，および使用するバッファ形式を，次の形式で設定します。

```
DCMCFEMI〔| {DCMCFBUF1|DCMCFBUF2}〕
```

DCMCFEMI

単一セグメントを示す DCMCFEMI を設定します。

DCMCFBUF1

バッファ形式 1 を使用する場合に設定します。

DCMCFBUF2

バッファ形式 2 を使用する場合に設定します。

●commform

同期型のメッセージの送受信を示す，DCMCFIO を設定します。

●termnam

メッセージを出力して応答を入力する論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。論理端末名称の最後にはヌル文字を付けてください。

●resv01

NULL またはヌル文字列を設定します。

●senddata

送信するセグメントの内容を設定した領域を設定します。一つのセグメントで 32000 バイトまで送信できます。

●sdataleng

送信するセグメントの長さを設定します。

●recvdata

セグメントを受信する領域を設定します。

単一セグメントまたは最終セグメントを送信する dc_mcf_sendrecv 関数が終了すると，受信したメッセージの先頭セグメントが返されます。

処理終了後，recvdata には OpenTP1 から値が返されます。

●inbufleng

セグメントを受信する領域の長さを設定します。

●watchtime

dc_mcf_sendrecv 関数を呼び出してから終了するまでの最大時間を設定します。0 を設定した場合，MCF マネジャ定義の UAP 共通定義で指定した同期型送受信監視時間（mcfmuap -t sndrcvtim）が設定されます。

負の値を設定した場合は，時間を監視しません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，設定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，設定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

OpenTP1 から値が返される引数

●recvdata

受信したメッセージの先頭セグメントの内容が返されます。

●rdataleng

受信したメッセージの先頭セグメントの長さが返されます。

●time

メッセージを受信した時刻が，1970 年 1 月 1 日 0 時 0 分 0 秒からの通算の秒数で返されます。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71002	-12002	メッセージキューへの入出力処理時に障害が発生しました。
		メッセージキューが閉塞されています。
		メッセージキューが割り当てられていません。
		sdataleng に 32000 バイトを超える値を設定しています。
		MCF が終了処理中のため，メッセージの送受信を受け付けられません。
DCMCFRTN_71003	-12003	メッセージキューが満杯です。
DCMCFRTN_71004	-12004	メッセージを格納するバッファをメモリ上に確保できませんでした。
DCMCFRTN_71108	-12108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_sendrecv 関数を呼び出しています。
DCMCFRTN_72001	-13001	termnam に設定した論理端末名称が間違っています。
		termnam に設定した出力先の論理端末名称は，MCF で定義していません。
		dc_mcf_sendrecv 関数を呼び出せない論理端末を設定しています。
		＜問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用している（コネクション定義（mcftalccn -l）の replymsg オペランドに yes を指定）場合＞ 問い合わせ応答中または継続問い合わせ応答中のため受け付けられません。
DCMCFRTN_72012	-13012	MCF バッファグループ定義のバッファ長が不足しました。
		MCF 通信プロセスは相手システムからメッセージを受信しましたが，UAP への応答連絡で RPC 通信の送信可能な上限値を超えました。
DCMCFRTN_72013	-13013	inbufleng の指定値を超えるセグメントを受信しました。inbufleng の指定値を超えた部分は切り捨てられました。
DCMCFRTN_72016	-13016	action に設定した値が間違っています。
		resv01 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。

リターン値	リターン値（数値）	意味
DCMCFRTN_72026	-13026	action に設定したセグメント種別（DCMCFESI または DCMCFEMI）の値が違っています。
DCMCFRTN_72036	-13036	inbufleng の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
DCMCFRTN_72041	-13041	sdataleng に 0 バイト，またはマイナス値を設定しています。
DCMCFRTN_72073	-13073	非同期メッセージを送信処理中です。
DCMCFRTN_73001	-14001	watchtime に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
		無通信監視時間（コネクション定義（mcftalccn -k）の notrftime オペランド指定値）が経過しましたが，相手システムからの通信がありません。
		相手システムからメッセージを受信しましたが，受信バッファを確保できませんでした。
		メッセージの送信処理中に相手システムからメッセージを受信したため，非同期受信処理を行いましたが，障害（UOC 障害など）が発生しました。
		出力先の論理端末で MCF 通信プロセスの内部障害が発生しました。
DCMCFRTN_73002	-14002	コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指定）場合に，MHP でメッセージ受信後にコネクションが再確立されたため，送信を抑止しました。
DCMCFRTN_73003	-14003	メッセージ受信が仕掛り中です。
DCMCFRTN_73005	-14005	watchtime に設定した時間が経過しましたが，論理端末からの応答がありません。
DCMCFRTN_73010	-14010	入力または出力メッセージ編集 UOC で障害が発生しました。
		メッセージの読み込み時に障害が発生しました。
		メッセージの編集エラーが発生しました。
DCMCFRTN_73015	-14015	出力先の論理端末は，ほかの UAP で仕掛り中です。
DCMCFRTN_73018	-14018	watchtime に設定した値が間違っています。
DCMCFTRN_73019	-14019	メッセージ送信完了監視タイマのタイムアウトが発生しました。
DCMCFRTN_73020	-14020	出力先の論理端末は停止しています。
上記以外	−	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

−：該当しません。

dc_mcf_sendsync － 同期型メッセージの送信（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_sendsync(DCLONG action, DCLONG commform,
                     char *termnam, char *resv01,
                     char *senddata, DCLONG sdataleng,
                     char *resv02, DCLONG opcd,
                     DCLONG watchtime)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int  dc_mcf_sendsync(action, commform, termnam, resv01, senddata,
                     sdataleng, resv02, opcd, watchtime)
DCLONG          action;
DCLONG          commform;
char            *termnam;
char            *resv01;
char            *senddata;
DCLONG          sdataleng;
char            *resv02;
DCLONG          opcd;
DCLONG          watchtime;
```

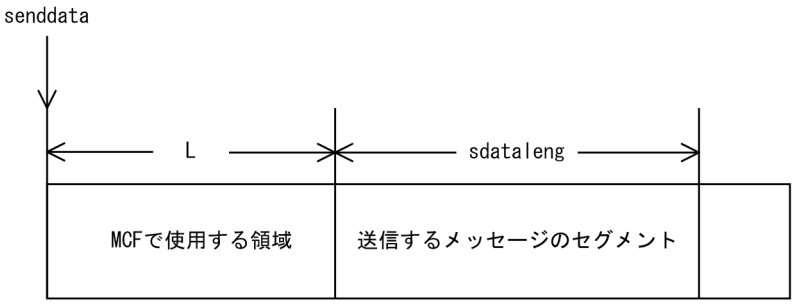
機能

相手システムへ同期型でメッセージを送信します。メッセージは，一つのセグメントで構成されます。

送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

セグメントを送信する領域の形式を次に示します。L は，バッファ形式 1 の場合は 8 バイト，バッファ形式 2 の場合は 4 バイトです。

●セグメントを送信する領域



UAP で値を設定する引数

●action

送信する論理メッセージのセグメントと，使用するバッファ形式を次の形式で指定します。

```
DCMCFEMI〔| {DCMCFBUF1|DCMCFBUF2}〕
```

DCMCFEMI

単一セグメントを示す DCMCFEMI を設定します。

DCMCFBUF1

バッファ形式 1 を使用する場合に設定します。

DCMCFBUF2

バッファ形式 2 を使用する場合に設定します。

●commform

DCNOFLAGS を設定します。

●termnam

メッセージを出力する論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。論理端末名称の最後にはヌル文字を付けてください。

●resv01

NULL またはヌル文字列を設定します。

●senddata

送信するセグメントの内容を設定した領域を設定します。一つのセグメントで 32000 バイトまで送信できます。

●sdataleng

送信するセグメントの長さを設定します。

●resv02

NULL またはヌル文字列を設定します。

●opcd

DCNOFLAGS を設定します。

●watchtime

dc_mcf_sendsync 関数を呼び出してから終了するまでの最大時間を設定します。0 を設定した場合，MCF マネジャ定義の UAP 共通定義で指定した同期型送信監視時間（mcfmuap -t sndtim）が仮定されます。

負の値を設定した場合は，時間を監視しません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，設定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，設定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71002	-12002	sdataleng に 32000 バイトを超える値を設定しています。
		MCF が終了処理中のため，メッセージの送信を受け付けられません。
DCMCFRTN_71003	-12003	メッセージキューが満杯です。
DCMCFRTN_71004	-12004	メッセージを格納するバッファをメモリ上に確保できませんでした。
DCMCFRTN_71108	-12108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_sendsync 関数を呼び出しています。
DCMCFRTN_72001	-13001	termnam に設定した論理端末名称が間違っています。
		termnam に設定した出力先の論理端末名称は，MCF で定義していません。
		dc_mcf_sendsync 関数を呼び出せない論理端末を設定しています。
		＜問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用している（コネクション定義（mcftalccn -l）の replymsg オペランドに yes を指定）場合＞ 問い合わせ応答中または継続問い合わせ応答中のため受け付けられません。
DCMCFRTN_72016	-13016	action に設定できない値を設定しています。
		opcd に設定した値が間違っています。
		resv01，resv02 の指す領域の値が NULL になっていません。
		引数に設定した値に間違いがあります。
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。
DCMCFRTN_72026	-13026	action に設定したセグメント種別（DCMCFEMI）の値が間違っています。
DCMCFRTN_72041	-13041	sdataleng に 0 バイト，またはマイナス値を設定しています。

リターン値	リターン値（数値）	意味
DCMCFRTN_72073	-13073	非同期メッセージを送信処理中です。
DCMCFRTN_73001	-14001	watchtime に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
		無通信監視時間（コネクション定義（mcftalccn -k）の notrftime オペランド指定値）が経過しましたが，相手システムからの通信がありません。
		相手システムからメッセージを受信しましたが，受信バッファを確保できませんでした。
		メッセージの送信処理中に相手システムからメッセージを受信したため，非同期受信処理を行いましたが，障害（UOC 障害など）が発生しました。
		出力先の論理端末で MCF 通信プロセスの内部障害が発生しました。
DCMCFRTN_73002	-14002	コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指定）場合に，MHP でメッセージ受信後にコネクションが再確立されたため，送信を抑止しました。
DCMCFRTN_73005	-14005	watchtime に設定した時間が経過しましたが，論理端末からの応答がありません。
DCMCFRTN_73010	-14010	出力メッセージ編集 UOC で障害が発生しました。
		メッセージの読み込み時に障害が発生しました。
DCMCFRTN_73015	-14015	出力先の論理端末は，ほかの UAP で仕掛り中です。
DCMCFRTN_73018	-14018	watchtime に設定した値が間違っています。
DCMCFRTN_73020	-14020	出力先の論理端末は停止しています。
上記以外	−	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

−：該当しません。

dc_mcf_tactcn － コネクションの確立（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_tactcn (DCLONG action, dcmcf_tactcnopt *cnopt,
                   char *proinf, DCLONG *resv02, char *resv03,
                   char *resv04)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_tactcn (action, cnopt, proinf, resv02, resv03, resv04)
DCLONG             action;
dcmcf_tactcnopt    *cnopt;
char               *proinf;
DCLONG             *resv02;
char               *resv03;
char               *resv04;
```

機能

コネクションを確立します。

なお，dc_mcf_tactcn 関数の正常終了は，コネクション確立要求を TP1/NET/TCP/IP が正常に受け付けたことを意味します。このため，相手システムとのコネクションの確立が正常に完了したことを示すものではありません。

dc_mcf_tactcn 関数の呼び出し後にコネクションに関する何らかの処理をする場合は，dc_mcf_tlscn 関数を用いてコネクションの状態を確認してください。

UAP で値を設定する引数

●action

確立するコネクションの指定方法を次の形式で設定します。

```
{DCMCFLE | DCMCFCN}  〔 | DCMCFPRO〕
```

DCMCFLE

確立するコネクションを論理端末名称で指定するときに設定します。

DCMCFCN

確立するコネクションをコネクション ID で指定するときに設定します。

DCMCFPRO

TP1/NET/TCP/IP 固有の機能を使用するときに設定します。

●cnopt

この関数の対象となったコネクションの情報を，構造体 dcmcf_tactcnopt に設定します。

構造体の形式を次に示します。

```
typedef struct {
                DCLONG        mcfid;          …MCF通信プロセス識別子
                char          resv01[4];      …予備領域
                char          idnam[9];       …論理端末名称,
                                                コネクションID
                char          resv02[7];      …予備領域
                char          resv03[112];    …予備領域
                char          scnnam[9];      …MCF使用領域
                char          resv04[7];      …予備領域
                char          yournam[9];     …MCF使用領域
                char          resv05[7];      …予備領域
                char          hostnam[143];   …MCF使用領域
                char          resv06[17];     …予備領域
                char          resv07[184];    …予備領域
                } dcmcf_tactcnopt;
```

- mcfid

  処理対象のコネクションを持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

  論理端末名称を使用してコネクションの確立を要求する場合は，無効となります。

  0 を指定すると，該当するコネクション ID が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの関数を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

  注※

  MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。

  例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

- resv01

  領域をヌル文字で埋めます。

- idnam

  確立するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は最大 8 バイトの長さです。論理端末名称の最後にはヌル文字を付けてください。

- resv02，resv03，scnnam，resv04，yournam，resv05，hostnam，resv06，resv07

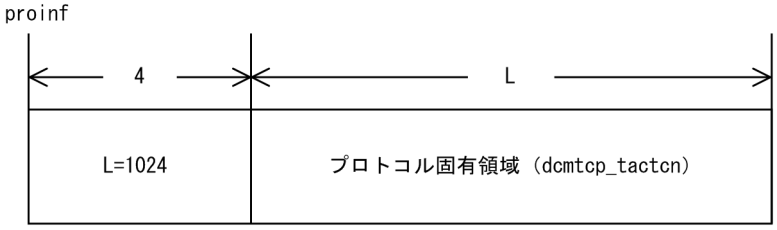
  領域をヌル文字で埋めます。

●proinf

プロトコル固有領域として，構造体 dcmtcp_tactcn を設定します。

TP1/NET/TCP/IP 固有の機能を使用しない場合は，NULL を設定します。

プロトコル固有領域の形式を次に示します。



構造体の形式を次に示します。

```
typedef struct {
                char            pnam[4];        …プロトコルの名称
                char            resv01[4];      …予備領域
                DCULONG         i_ipaddr;       …自システムの
                                                  IPアドレス
                unsigned short  i_portno;       …自システムの
                                                  ポート番号
                char            resv02[2];      …予備領域
                DCULONG         o_ipaddr;       …相手システムの
                                                  IPアドレス
                unsigned short  o_portno;       …相手システムの
                                                  ポート番号
                char            resv03[2];      …予備領域
                char            i_pathname[128]; …予備領域
                char            o_pathname[128]; …予備領域
                char            resv04[744];    …予備領域
                } dcmtcp_tactcn;
```

- pnam

  プロトコル固有領域のプロトコル名称を設定します。

  "TCP△"

  TCP/IP プロトコル

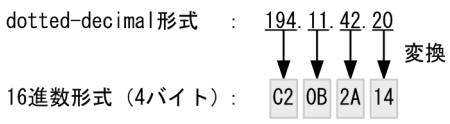
- resv01

  領域をヌル文字で埋めます。

- i_ipaddr

  自システムの IP アドレスを 16 進形式（ビッグエンディアン）で設定します。

  ＜IPアドレスの設定例＞

  dotted-decimal 形式では指定できません。

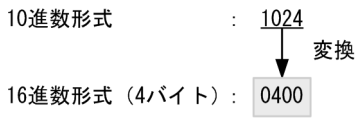
  自システムの IP アドレスの割り当てを OS に任せる場合は，0 を設定してください。

  IP アドレス 255.255.255.255 は設定できません。

- i_portno

自システムのポート番号を 16 進形式（ビッグエンディアン）で設定します。

＜ポート番号の設定例＞



設定できる範囲は 0，または 1024～65535 です。

自システムのポート番号の割り当てを OS に任せる場合は，0 を設定してください。

- resv02

  領域をヌル文字で埋めます。

- o_ipaddr

  相手システムの IP アドレスを 16 進形式（ビッグエンディアン）で設定します。

  dotted-decimal 形式では指定できません。

  IP アドレス 0.0.0.0，および 255.255.255.255 は設定できません。

- o_portno

  相手システムのポート番号を 16 進形式（ビッグエンディアン）で設定します。

  設定できる範囲は 1～65535 です。

- resv03，i_pathname，o_pathname，resv04

  領域をヌル文字で埋めます。

●resv02，resv03，resv04

NULL を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tactcn 関数が受け付けられません。
DCMCFRTN_71002	-12002	MCF が終了処理中のため，dc_mcf_tactcn 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tactcn 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71007	-12007	指定されたコネクション名は登録されていません。
DCMCFRTN_71008	-12008	指定された論理端末名称は登録されていません。

リターン値	リターン値(数値)	意味
DCMCFRTN_71009	-12009	dc_mcf_tactcn 関数が，該当する MCF 通信プロセスではサポートされていません。
DCMCFRTN_71010	-12010	MCF 通信プロセスにコネクションの確立を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71011	-12011	コネクションが削除されているため，dc_mcf_tactcn 関数が受け付けられません。
DCMCFRTN_72050	-13050	action に未サポートのフラグを設定しています。
DCMCFRTN_72051	-13051	cnopt に NULL が設定されています。
DCMCFRTN_72052	-13052	＜ action に DCMCFPRO を設定していない場合＞ proinf に NULL が設定されていません。
		＜ action に DCMCFPRO を設定した場合＞ proinf が示すプロトコル固有領域長 L に 0 未満，または 1025 以上の値を設定しています。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72054	-13054	resv03 に NULL が設定されていません。
DCMCFRTN_72055	-13055	resv04 に NULL が設定されていません。
DCMCFRTN_72060	-13060	action には DCMCFLE と DCMCFCN を同時に設定できません。
DCMCFRTN_72061	-13061	dcmcf_tactcnopt の mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72062	-13062	dcmcf_tactcnopt の resv01 がヌル文字で埋められていません。
DCMCFRTN_72063	-13063	dcmcf_tactcnopt の idnam の先頭がヌル文字です。
DCMCFRTN_72064	-13064	dcmcf_tactcnopt の resv02 がヌル文字で埋められていません。
DCMCFRTN_72065	-13065	dcmcf_tactcnopt の resv03 がヌル文字で埋められていません。
DCMCFRTN_72066	-13066	dcmcf_tactcnopt の scnnam がヌル文字で埋められていません。
DCMCFRTN_72067	-13067	dcmcf_tactcnopt の resv04 がヌル文字で埋められていません。
DCMCFRTN_72068	-13068	dcmcf_tactcnopt の yournam がヌル文字で埋められていません。
DCMCFRTN_72069	-13069	dcmcf_tactcnopt の resv05 がヌル文字で埋められていません。
DCMCFRTN_72070	-13070	dcmcf_tactcnopt の hostnam がヌル文字で埋められていません。
DCMCFRTN_72071	-13071	dcmcf_tactcnopt の resv06 がヌル文字で埋められていません。
DCMCFRTN_72072	-13072	dcmcf_tactcnopt の resv07 がヌル文字で埋められていません。
DCMCFRTN_72073	-13073	dcmcf_tactcnopt の idnam に 9 以上の文字が設定されています。
DCMCFRTN_72074	-13074	dcmcf_tactcnopt の idnam に設定された文字列中に不正な文字があります。

リターン値	リターン値(数値)	意味
DCMCFRTN_73100	-14100	コネクションが確立済みのため受け付けられません。
DCMCFRTN_73101	-14101	コネクションが確立処理中のため受け付けられません。
DCMCFRTN_73102	-14102	コネクションに対する論理端末が接続されていないため受け付けられません。
DCMCFRTN_73103	-14103	サーバ型コネクションのため受け付けられません。
DCMCFRTN_73104	-14104	コネクションが解放処理中のため受け付けられません。
DCMCFRTN_73130	-14130	プロトコル固有領域長 L に設定した値が不足しています。
DCMCFRTN_73131	-14131	dcmtcp_tactcn の pnam の値が間違っています。
DCMCFRTN_73132	-14132	dcmtcp_tactcn の resv01 がヌル文字で埋められていません。
DCMCFRTN_73133	-14133	dcmtcp_tactcn の i_ipaddr に 255.255.255.255 の IP アドレスが設定されています。
DCMCFRTN_73134	-14134	dcmtcp_tactcn の i_portno に 1 以上 1023 以下の値が設定されています。
DCMCFRTN_73135	-14135	dcmtcp_tactcn の resv02 がヌル文字で埋められていません。
DCMCFRTN_73136	-14136	dcmtcp_tactcn の o_ipaddr に 0.0.0.0，または 255.255.255.255 の IP アドレスが設定されています。
DCMCFRTN_73137	-14137	dcmtcp_tactcn の o_portno に 0 が設定されています。
DCMCFRTN_73138	-14138	dcmtcp_tactcn の resv03 がヌル文字で埋められていません。
DCMCFRTN_73139	-14139	dcmtcp_tactcn の i_pathname がヌル文字で埋められていません。
DCMCFRTN_73140	-14140	dcmtcp_tactcn の o_pathname がヌル文字で埋められていません。
DCMCFRTN_73141	-14141	dcmtcp_tactcn の resv04 がヌル文字で埋められていません。

dc_mcf_tactle － 論理端末の閉塞解除（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_tactle (DCLONG action, dcmcf_tactleopt *leopt,
                   char *proinf, DCLONG *resv02,
                   char *resv03, char *resv04)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_tactle (action, leopt, proinf, resv02, resv03, resv04)
DCLONG             action;
dcmcf_tactleopt    *leopt;
char               *proinf;
DCLONG             *resv02;
char               *resv03;
char               *resv04;
```

機能

論理端末の閉塞を解除します。

なお，dc_mcf_tactle 関数の正常終了は，論理端末の閉塞解除要求を TP1/NET/TCP/IP が正常に受け付けたことを意味します。このため，論理端末の閉塞解除が正常に完了したことを示すものではありません。

dc_mcf_tactle 関数の呼び出し後に論理端末に関する何らかの処理をする場合は，dc_mcf_tlsle 関数を用いて論理端末の状態を確認してください。

UAP で値を設定する引数

●action

閉塞解除する論理端末の指定方法を次の形式で設定します。

```
DCMCFLE
```

DCMCFLE

閉塞解除する論理端末を論理端末名称で指定するときに設定します。

●leopt

この関数の対象となった論理端末の情報を，構造体 dcmcf_tactleopt に設定します。

構造体の形式を次に示します。

```
typedef struct {
                DCLONG    mcfid;          …MCF通信プロセス識別子
                char      resv01[4];      …予備領域
                char      idnam[9];       …論理端末名称
                char      resv02[7];      …予備領域
                char      resv03[112];    …予備領域
                char      resv04[376];    …予備領域
                } dcmcf_tactleopt;
```

- mcfid

  処理対象の論理端末を持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

  0 を指定すると，該当する論理端末名称が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの関数を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

  注※

  MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

- resv01

  領域をヌル文字で埋めます。

- idnam

  閉塞解除する論理端末の名称を設定します。論理端末名称は 8 バイト以内で設定し，文字列の最後にヌル文字を付けます。

- resv02，resv03，resv04

  領域をヌル文字で埋めます。

●proinf，resv02，resv03，resv04

NULL を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tactle 関数が受け付けられません。
DCMCFRTN_71002	-12002	MCF が終了処理中のため，dc_mcf_tactle 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tactle 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。

リターン値	リターン値（数値）	意味
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71008	-12008	指定された論理端末名称は登録されていません。
DCMCFRTN_71009	-12009	dc_mcf_tactle 関数が，該当する MCF 通信プロセスではサポートされていません。または，サポートしていない DCMCFPRO を指定しています。
DCMCFRTN_71010	-12010	MCF 通信プロセスに論理端末の閉塞の解除を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71011	-12011	論理端末が削除されているため，dc_mcf_tactle 関数が受け付けられません。
DCMCFRTN_72050	-13050	action に未サポートのフラグを設定しています。
		action に DCMCFLE が指定されていません。
DCMCFRTN_72051	-13051	leopt に NULL が設定されています。
DCMCFRTN_72052	-13052	proinf に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72054	-13054	resv03 に NULL が設定されていません。
DCMCFRTN_72055	-13055	resv04 に NULL が設定されていません。
DCMCFRTN_72061	-13061	dcmcf_tactleopt の mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72062	-13062	dcmcf_tactleopt の resv01 がヌル文字で埋められていません。
DCMCFRTN_72063	-13063	dcmcf_tactleopt の idnam の先頭がヌル文字です。
DCMCFRTN_72064	-13064	dcmcf_tactleopt の resv02 がヌル文字で埋められていません。
DCMCFRTN_72065	-13065	dcmcf_tactleopt の resv03 がヌル文字で埋められていません。
DCMCFRTN_72067	-13067	dcmcf_tactleopt の resv04 がヌル文字で埋められていません。
DCMCFRTN_72073	-13073	dcmcf_tactleopt の idnam に設定された文字数が 9 以上です。
DCMCFRTN_72074	-13074	dcmcf_tactleopt の idnam に設定された文字列中に不正な文字があります。

dc_mcf_tdctcn － コネクションの解放（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_tdctcn (DCLONG action, dcmcf_tdctcnopt *cnopt,
                   char *proinf, DCLONG *resv02, char *resv03,
                   char *resv04)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_tdctcn (action, cnopt, proinf, resv02, resv03, resv04)
DCLONG               action;
dcmcf_tdctcnopt      *cnopt;
char                 *proinf;
DCLONG               *resv02;
char                 *resv03;
char                 *resv04;
```

機能

コネクションを解放します。

なお，dc_mcf_tdctcn 関数の正常終了は，コネクション解放要求を TP1/NET/TCP/IP が正常に受け付けたことを意味します。このため，相手システムとのコネクションの解放が正常に完了したことを示すものではありません。

dc_mcf_tdctcn 関数の呼び出し後にコネクションに関する何らかの処理をする場合は，dc_mcf_tlscn 関数を用いてコネクションの状態を確認してください。

コネクション解放後，CCLSEVT または CERREVT を通知します。コネクション解放後に通知する MCF イベントを次の表に示します。

action 引数の DCMCFFRC 指定	コネクション定義（mcftalccn -f）の cnrelease オペランドの指定値	通知する MCF イベント
なし	任意	CCLSEVT
あり	fin	CCLSEVT
	rst	CERREVT

UAP で値を設定する引数

●action

解放するコネクションの指定方法を次の形式で設定します。

```
{DCMCFLE | DCMCFCN}  〔 | DCMCFFRC〕
```

DCMCFLE

解放するコネクションを論理端末名称で指定するときに設定します。

DCMCFCN

解放するコネクションをコネクション ID で指定するときに設定します。

DCMCFFRC

コネクションを強制的に解放するときに設定します。

コネクション定義（mcftalccn -f）の cnrelease オペランドに rst を指定している場合，RST パケットを送信してコネクションを強制解放します。

コネクション定義（mcftalccn -f）の cnrelease オペランドを省略，または fin を指定している場合，指定しても無効となり，FIN パケットを送信してコネクションを正常に解放します。

●cnopt

この関数の対象となったコネクションの情報を，構造体 dcmcf_tdctcnopt に設定します。

構造体の形式を次に示します。

```
typedef struct {
                DCLONG    mcfid;          …MCF通信プロセス識別子
                char      resv01[4];      …予備領域
                char      idnam[9];       …論理端末名称,
                                            コネクションID
                char      resv02[7];      …予備領域
                char      resv03[112];    …予備領域
                char      scnnam[9];      …MCF使用領域
                char      resv04[7];      …予備領域
                char      resv05[360];    …予備領域
                } dcmcf_tdctcnopt;
```

- mcfid

  処理対象のコネクションを持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

  論理端末名称を使用してコネクションの解放を要求する場合は，無効となります。

  0 を指定すると，該当するコネクション ID が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの関数を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

  注※

  MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

- resv01

  領域をヌル文字で埋めます。

- idnam
  解放するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は８バイト以内で設定し，文字列の最後にヌル文字を付けます。
- resv02，resv03，scnnam，resv04，resv05
  領域をヌル文字で埋めます。

●proinf，resv02，resv03，resv04

NULL を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tdctcn 関数が受け付けられません。
DCMCFRTN_71002	-12002	MCF が終了処理中のため，dc_mcf_tdctcn 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tdctcn 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71007	-12007	指定されたコネクション名は登録されていません。
DCMCFRTN_71008	-12008	指定された論理端末名称は登録されていません。
DCMCFRTN_71009	-12009	dc_mcf_tdctcn 関数が，該当する MCF 通信プロセスではサポートされていません。または，サポートしていない DCMCFPRO を指定しています。
DCMCFRTN_71010	-12010	MCF 通信プロセスにコネクションの解放を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71011	-12011	コネクションが削除されているため，dc_mcf_tdctcn 関数が受け付けられません。
DCMCFRTN_71014	-12014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループ名を指定しています。
DCMCFRTN_72050	-13050	action に未サポートのフラグを設定しています。
DCMCFRTN_72051	-13051	cnopt に NULL が設定されています。
DCMCFRTN_72052	-13052	proinf に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72054	-13054	resv03 に NULL が設定されていません。

リターン値	リターン値(数値)	意味
DCMCFRTN_72055	-13055	resv04 に NULL が設定されていません。
DCMCFRTN_72060	-13060	action には DCMCFLE と DCMCFCN を同時に設定できません。
DCMCFRTN_72061	-13061	dcmcf_tdctcnopt の mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72062	-13062	dcmcf_tdctcnopt の resv01 がヌル文字で埋められていません。
DCMCFRTN_72063	-13063	dcmcf_tdctcnopt の idnam の先頭がヌル文字です。
DCMCFRTN_72064	-13064	dcmcf_tdctcnopt の resv02 がヌル文字で埋められていません。
DCMCFRTN_72065	-13065	dcmcf_tdctcnopt の resv03 がヌル文字で埋められていません。
DCMCFRTN_72066	-13066	dcmcf_tdctcnopt の scnnam がヌル文字で埋められていません。
DCMCFRTN_72067	-13067	dcmcf_tdctcnopt の resv04 がヌル文字で埋められていません。
DCMCFRTN_72069	-13069	dcmcf_tdctcnopt の resv05 がヌル文字で埋められていません。
DCMCFRTN_72073	-13073	dcmcf_tdctcnopt の idnam に設定された文字数が 9 以上です。
DCMCFRTN_72074	-13074	dcmcf_tdctcnopt の idnam に設定された文字列中に不正な文字があります。

dc_mcf_tdctle － 論理端末の閉塞（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_tdctle (DCLONG action, dcmcf_tdctleopt *leopt,
                   char *proinf, DCLONG *resv02,
                   char *resv03, char *resv04)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_tdctle (action, leopt, proinf, resv02, resv03, resv04)
DCLONG             action;
dcmcf_tdctleopt    *leopt;
char               *proinf;
DCLONG             *resv02;
char               *resv03;
char               *resv04;
```

機能

論理端末を閉塞します。

なお，dc_mcf_tdctle 関数の正常終了は，論理端末の閉塞要求を TP1/NET/TCP/IP が正常に受け付けたことを意味します。このため，論理端末の閉塞が正常に完了したことを示すものではありません。

dc_mcf_tdctle 関数の呼び出し後に論理端末に関する何らかの処理をする場合は，dc_mcf_tlsle 関数を用いて論理端末の状態を確認してください。

UAP で値を設定する引数

●action

閉塞する論理端末の指定方法を次の形式で設定します。

```
DCMCFLE
```

DCMCFLE

閉塞する論理端末を論理端末名称で指定するときに設定します。

●leopt

この関数の対象となったコネクションの情報を，構造体 dcmcf_tdctleopt に設定します。

構造体の形式を次に示します。

```
typedef struct {
                DCLONG    mcfid;          …MCF通信プロセス識別子
                char      resv01[4];      …予備領域
                char      idnam[9];       …論理端末名称
                char      resv02[7];      …予備領域
                char      resv03[112];    …予備領域
                char      resv04[376];    …予備領域
                } dcmcf_tdctleopt;
```

- mcfid

  処理対象の論理端末を持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

  0 を指定すると，該当する論理端末名称が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの関数を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

  注※

  MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

- resv01

  領域をヌル文字で埋めます。

- idnam

  閉塞する論理端末の名称を設定します。論理端末名称は 8 バイト以内で設定し，文字列の最後にヌル文字を付けます。

- resv02，resv03，resv04

  領域をヌル文字で埋めます。

●proinf，resv02，resv03，resv04

NULL を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tdctle 関数が受け付けられません。
DCMCFRTN_71002	-12002	MCF が終了処理中のため，dc_mcf_tdctle 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tdctle 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。

リターン値	リターン値（数値）	意味
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71008	-12008	指定された論理端末名称は登録されていません。
DCMCFRTN_71009	-12009	dc_mcf_tdctle 関数が，該当する MCF 通信プロセスではサポートされていません。または，サポートしていない DCMCFPRO を指定しています。
DCMCFRTN_71010	-12010	MCF 通信プロセスに論理端末の閉塞を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71011	-12011	論理端末が削除されているため，dc_mcf_tdctle 関数が受け付けられません。
DCMCFRTN_72050	-13050	action に未サポートのフラグを設定しています。
		action に DCMCFLE が指定されていません。
DCMCFRTN_72051	-13051	leopt に NULL が設定されています。
DCMCFRTN_72052	-13052	proinf に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72054	-13054	resv03 に NULL が設定されていません。
DCMCFRTN_72055	-13055	resv04 に NULL が設定されていません。
DCMCFRTN_72061	-13061	dcmcf_tdctleopt の mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72062	-13062	dcmcf_tdctleopt の resv01 がヌル文字で埋められていません。
DCMCFRTN_72063	-13063	dcmcf_tdctleopt の idnam の先頭がヌル文字です。
DCMCFRTN_72064	-13064	dcmcf_tdctleopt の resv02 がヌル文字で埋められていません。
DCMCFRTN_72065	-13065	dcmcf_tdctleopt の resv03 がヌル文字で埋められていません。
DCMCFRTN_72067	-13067	dcmcf_tdctleopt の resv04 がヌル文字で埋められていません。
DCMCFRTN_72073	-13073	dcmcf_tdctleopt の idnam に設定された文字数が 9 以上です。
DCMCFRTN_72074	-13074	dcmcf_tdctleopt の idnam に設定された文字列中に不正な文字があります。

dc_mcf_tempget － 一時記憶データの受け取り（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_tempget(DCLONG action, char *getdata, DCLONG gtempleng,
                    DCLONG *gdataleng, char *resv01)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int  dc_mcf_tempget(action, getdata, gtempleng, gdataleng, resv01)
DCLONG    action;
char      *getdata;
DCLONG    gtempleng;
DCLONG    *gdataleng;
char      *resv01;
```

機能

継続問い合わせ応答用一時記憶領域に格納されている一時記憶データを受け取ります。

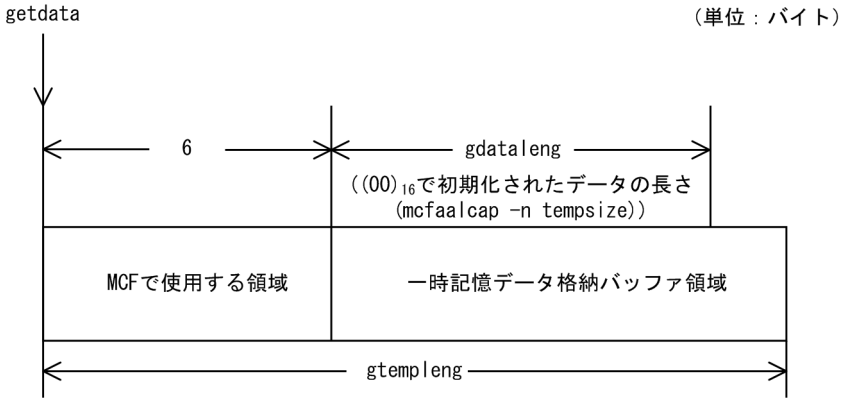
gtempleng の長さ（7～32006 バイト）を超える一時記憶データがある場合，超えた部分については切り捨てます。

dc_mcf_tempget 実行時，gtempleng から 6 を減算した値と比べて一時記憶データ長が短い場合，一時記憶データを受け取り領域に設定します。残りの受け取り領域については何も設定しません。

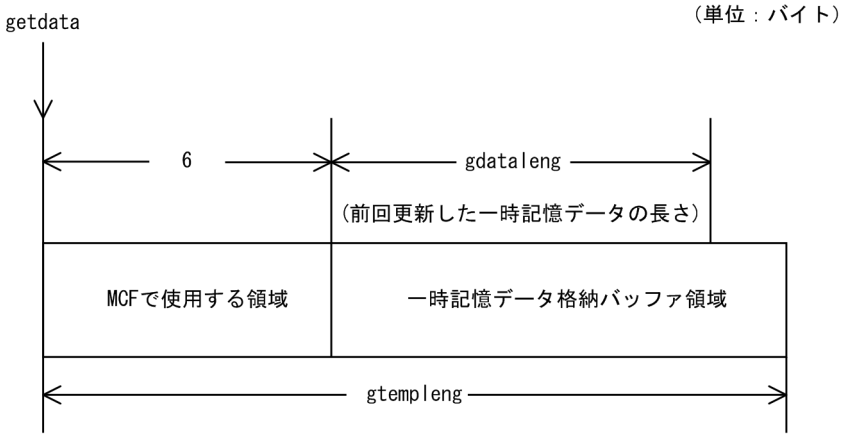
dc_mcf_tempget 実行時，初期状態（継続問い合わせ応答開始後，dc_mcf_tempput を 1 回も実行していない状態）の場合，アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドで指定した長さの$(00)_{16}$の一時記憶データがあるものとして，dc_mcf_tempget を実行します。

受け取り領域の形式を次に示します。

●dc_mcf_tempput未実行（継続問い合わせ応答開始後，dc_mcf_tempputを実行していない（初期状態））



●dc_mcf_tempput実行済み（継続問い合わせ応答開始後，dc_mcf_tempputを1回以上実行）



UAP で値を設定する引数

●action

DCMCFBUF2 を設定します。

●getdata

受け取り領域を設定します。

処理終了後，getdata には OpenTP1 から値が返されます。

●gtempleng

受け取り領域長を 7～32006 バイトで設定します。

●resv01

ヌル文字列を設定します。

OpenTP1 から値が返される引数

●getdata

受け取った一時記憶データが返されます。初期状態の場合，継続問い合わせ応答用一時記憶領域の長さ（アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドの指定値）分だけ$(00)_{16}$が埋められます。

●gdataleng

前回更新した一時記憶データの長さが返されます。初期状態の場合，継続問い合わせ応答用一時記憶領域の長さ（アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドの指定値）が返されます。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_72000	-13000	SPP では dc_mcf_tempget 関数を呼び出せません。
DCMCFRTN_72013	-13013	gtempleng の設定値を超える一時記憶データを受け取りました。gtempleng の設定値を超えた部分は切り捨てられました。
DCMCFRTN_72016	-13016	resv01 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72036	-13036	gtempleng の設定値が不足しています。7 バイト以上の領域を確保してください。
DCMCFRTN_72101	-13101	継続問い合わせ応答型でないアプリケーションで，dc_mcf_tempget 関数を呼び出しています。
DCMCFRTN_72106	-13106	先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_tempget 関数を呼び出しています。
DCMCFRTN_72107	-13107	dc_mcf_contend 関数を呼び出したあとで，dc_mcf_tempget 関数を呼び出しています。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_tempput － 一時記憶データの更新（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int  dc_mcf_tempput(DCLONG action, char *putdata, DCLONG pdataleng,
                    char *resv01)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int  dc_mcf_tempput(action, putdata, pdataleng, resv01)
DCLONG    action;
char      *putdata;
DCLONG    pdataleng;
char      *resv01;
```

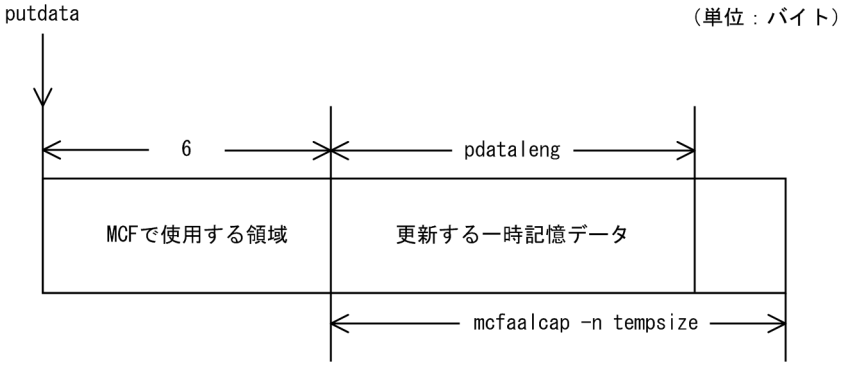
機能

継続問い合わせ応答用一時記憶領域に格納されている一時記憶データを更新します。

更新要求をする前に，必ず受け取り要求をしてください。

アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドには，更新する一時記憶データ長（pdataleng）以上の値を指定してください。

更新する領域の形式を次に示します。



UAP で値を設定する引数

●action

DCMCFBUF2 を設定します。

●putdata

一時記憶データの更新データが格納されている領域を設定します。

●pdataleng

一時記憶データの更新データ長を設定します。

●resv01

ヌル文字列を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71103	-12103	一時記憶データを更新するための領域をメモリ上に確保できませんでした。
DCMCFRTN_72000	-13000	SPP では dc_mcf_tempput 関数を呼び出せません。
DCMCFRTN_72016	-13016	resv01 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72035	-13035	pdataleng に設定した更新データの長さが，アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドで定義した長さを超えています。
		pdataleng に 0 バイト，またはマイナス値を設定しています。
DCMCFRTN_72101	-13101	継続問い合わせ応答型でないアプリケーションで，dc_mcf_tempput 関数を呼び出しています。
DCMCFRTN_72105	-13105	dc_mcf_tempget 関数を呼び出す前に，dc_mcf_tempput 関数を呼び出しています。
DCMCFRTN_72106	-13106	先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_tempput 関数を呼び出しています。
DCMCFRTN_72107	-13107	dc_mcf_contend 関数を呼び出したあとで，dc_mcf_tempput 関数を呼び出しています。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_tlscn － コネクションの状態取得（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_tlscn (DCLONG action, dcmcf_tlscnopt *cnopt,
                  char *resv01, DCLONG *resv02,
                  char *resv03, DCLONG *infcnt,
                  dcmcf_cninf *inf, char *resv04)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_tlscn (action, cnopt, resv01, resv02, resv03, infcnt,
                  inf, resv04)
DCLONG            action;
dcmcf_tlscnopt    *cnopt;
char              *resv01;
DCLONG            *resv02;
char              *resv03;
DCLONG            *infcnt;
dcmcf_cninf       *inf;
char              *resv04;
```

機能

コネクションの状態を取得します。

UAP で値を設定する引数

●action

状態を取得するコネクションの指定方法を次の形式で設定します。

```
{DCMCFLE | DCMCFCN}
```

DCMCFLE

状態を取得するコネクションを論理端末名称で指定するときに設定します。

DCMCFCN

状態を取得するコネクションをコネクション ID で指定するときに設定します。

●cnopt

この関数の対象となったコネクションの情報を，構造体 dcmcf_tlscnopt に設定します。

構造体の形式を次に示します。

```
typedef struct {
                DCLONG    mcfid;        …MCF通信プロセス識別子
                char      resv01[4];    …予備領域
                char      idnam[9];     …論理端末名称，コネクションID
                char      resv02[7];    …予備領域
                char      resv03[112];  …予備領域
                char      resv04[376];  …予備領域
                } dcmcf_tlscnopt;
```

- mcfid

  処理対象のコネクションを持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

  論理端末名称を使用してコネクションの状態取得を要求する場合は，無効となります。

  0 を指定すると，該当するコネクション ID が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの関数を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

  注※

  MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。

  例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

- resv01

  領域をヌル文字で埋めます。

- idnam

  状態を取得するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は 8 バイト以内で設定し，文字列の最後にヌル文字を付けます。

- resv02，resv03，resv04

  領域をヌル文字で埋めます。

●resv01，resv02，resv03

NULL を設定します。

●infcnt

コネクション状態を格納する領域 dcmcf_cninf の個数として，1 を設定します。

処理終了後は，該当するコネクションの個数が返されます。

●inf

コネクション状態を格納する領域 dcmcf_cninf を設定します。

「構造体 dcmcf_cninf のサイズ×infcnt」バイト数分の領域が必要です。

●resv04

NULL を設定します。

OpenTP1 から値が返される引数

●infcnt

この関数の対象となったコネクションの個数が返されます。

●inf

この関数の対象となったコネクションの情報が，構造体 dcmcf_cninf で返されます。

構造体の形式を次に示します。

```
typedef struct {
                char      idnam[9];     …コネクションID
                char      resv01[7];    …予備領域
                char      pnam[4];      …プロトコル名
                DCLONG    status;       …コネクション状態
                char      resv02[40];   …予備領域
                } dcmcf_cninf;
```

- idnam

  要求したコネクションのコネクション ID が設定されます。

- resv01

  領域をヌル文字で埋めます。

- pnam

  要求したコネクションのプロトコル種別が設定されます。

  TCP

  TCP/IP プロトコル

- status

  要求したコネクションの状態として，次の値が設定されます。

  DCMCF_CNST_ACT

  確立状態

  DCMCF_CNST_ACT_B

  確立処理中状態

  DCMCF_CNST_DCT

  解放状態

  DCMCF_CNST_DCT_B

  解放処理中状態

- resv02

  領域をヌル文字で埋めます。

リターン値

リターン値	リターン値(数値)	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tlscn 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tlscn 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71007	-12007	指定されたコネクション名は登録されていません。
DCMCFRTN_71008	-12008	指定された論理端末名称は登録されていません。
DCMCFRTN_71009	-12009	dc_mcf_tlscn 関数が，該当する MCF 通信プロセスではサポートされていません。
DCMCFRTN_71010	-12010	MCF 通信プロセスにコネクションの状態取得を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71011	-12011	コネクションが削除されているため，dc_mcf_tlscn 関数が受け付けられません。
DCMCFRTN_71014	-12014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループ名を指定しています。
DCMCFRTN_72050	-13050	action に未サポートのフラグを設定しています。
DCMCFRTN_72051	-13051	cnopt に NULL が設定されています。
DCMCFRTN_72052	-13052	resv01 に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72054	-13054	resv03 に NULL が設定されていません。
DCMCFRTN_72055	-13055	resv04 に NULL が設定されていません。
DCMCFRTN_72056	-13056	infcnt に NULL が設定されています。
DCMCFRTN_72057	-13057	inf に NULL が設定されています。
DCMCFRTN_72060	-13060	action には DCMCFLE と DCMCFCN を同時に設定できません。
DCMCFRTN_72061	-13061	dcmcf_tlscnopt の mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72062	-13062	dcmcf_tlscnopt の resv01 がヌル文字で埋められていません。
DCMCFRTN_72063	-13063	dcmcf_tlscnopt の idnam の先頭がヌル文字です。

リターン値	リターン値(数値)	意味
DCMCFRTN_72064	-13064	dcmcf_tlscnopt の resv02 がヌル文字で埋められていません。
DCMCFRTN_72065	-13065	dcmcf_tlscnopt の resv03 がヌル文字で埋められていません。
DCMCFRTN_72067	-13067	dcmcf_tlscnopt の resv04 がヌル文字で埋められていません。
DCMCFRTN_72073	-13073	dcmcf_tlscnopt の idnam に設定された文字数が 9 以上です。
DCMCFRTN_72074	-13074	dcmcf_tlscnopt の idnam に設定された文字列中に不正な文字があります。
DCMCFRTN_72076	-13076	infcnt に 1 が設定されていません。

dc_mcf_tlsle － 論理端末の状態取得（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_tlsle (DCLONG action, dcmcf_tlsleopt *leopt,
                  char *resv01, DCLONG *resv02,
                  char *resv03, DCLONG *infcnt,
                  dcmcf_leinf2 *inf, char *resv04)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_tlsle (action, leopt, resv01, resv02, resv03, infcnt,
                  inf, resv04)
DCLONG            action;
dcmcf_tlsleopt    *leopt;
char              *resv01;
DCLONG            *resv02;
char              *resv03;
DCLONG            *infcnt;
dcmcf_leinf2      *inf;
char              *resv04;
```

機能

論理端末の状態を取得します。

UAP で値を設定する引数

●action

状態を取得する論理端末の指定方法を次の形式で設定します。

```
DCMCFLE
```

DCMCFLE

状態を取得する論理端末を論理端末名称で指定するときに設定します。

●leopt

この関数の対象となったコネクションの情報を，構造体 dcmcf_tlsleopt に設定します。

構造体の形式を次に示します。

```
typedef struct {
                DCLONG    mcfid;        …MCF通信プロセス識別子
                char      resv01[4];    …予備領域
```

```
char      idnam[9];     …論理端末名称
char      resv02[7];    …予備領域
char      resv03[112];  …予備領域
char      resv04[376];  …予備領域
} dcmcf_tlsleopt;
```

- mcfid

  処理対象の論理端末を持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

  0 を指定すると，該当する論理端末名称が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの関数を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

  注※

  MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

- resv01

  領域をヌル文字で埋めます。

- idnam

  状態を取得する論理端末の名称を設定します。論理端末名称は 8 バイト以内で設定し，文字列の最後にヌル文字を付けます。

- resv02，resv03，resv04

  領域をヌル文字で埋めます。

●resv01，resv02，resv03

NULL を設定します。

●infcnt

論理端末の状態を格納する領域 dcmcf_leinf2 の個数として，1 を設定します。

処理終了後は，該当する論理端末の個数が返されます。

●inf

論理端末の状態を格納する領域 dcmcf_leinf2 を設定します。

「構造体 dcmcf_leinf2 のサイズ×infcnt」バイト数分の領域が必要です。

●resv04

NULL を設定します。

OpenTP1 から値が返される引数

●infcnt

この関数の対象となった論理端末の個数が返されます。

●inf

この関数の対象となった論理端末の情報が構造体 dcmcf_leinf2 で返されます。

構造体の形式を次に示します。

```
typedef struct {
                char      idnam[9];     …論理端末名称
                char      resv01[7];    …予備領域
                char      resv02[4];    …予備領域
                DCLONG    status;       …論理端末状態
                char      resv03[40];   …予備領域
                } dcmcf_leinf2;
```

- idnam

  要求した論理端末の名称が設定されます。

- resv01，resv02

  領域をヌル文字で埋めます。

- status

  要求した論理端末の状態として，次の値が設定されます。

  DCMCF_LEST_ACT

  　閉塞解除状態

  DCMCF_LEST_DCT

  　閉塞状態

- resv03

  領域をヌル文字で埋めます。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tlsle 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tlsle 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。

リターン値	リターン値（数値）	意味
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71008	-12008	指定された論理端末名称は登録されていません。
DCMCFRTN_71009	-12009	dc_mcf_tlsle 関数が，該当する MCF 通信プロセスではサポートされていません。
DCMCFRTN_71010	-12010	MCF 通信プロセスに論理端末の状態取得を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71011	-12011	論理端末が削除されているため，dc_mcf_tlsle 関数が受け付けられません。
DCMCFRTN_72050	-13050	action に未サポートのフラグを設定しています。
		action に DCMCFLE が指定されていません。
DCMCFRTN_72051	-13051	leopt に NULL が設定されています。
DCMCFRTN_72052	-13052	resv01 に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72054	-13054	resv03 に NULL が設定されていません。
DCMCFRTN_72055	-13055	resv04 に NULL が設定されていません。
DCMCFRTN_72056	-13056	infcnt に NULL が設定されています。
DCMCFRTN_72057	-13057	inf に NULL が設定されています。
DCMCFRTN_72061	-13061	dcmcf_tlsleopt の mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72062	-13062	dcmcf_tlsleopt の resv01 がヌル文字で埋められていません。
DCMCFRTN_72063	-13063	dcmcf_tlsleopt の idnam の先頭がヌル文字です。
DCMCFRTN_72064	-13064	dcmcf_tlsleopt の resv02 がヌル文字で埋められていません。
DCMCFRTN_72065	-13065	dcmcf_tlsleopt の resv03 がヌル文字で埋められていません。
DCMCFRTN_72067	-13067	dcmcf_tlsleopt の resv04 がヌル文字で埋められていません。
DCMCFRTN_72073	-13073	dcmcf_tlsleopt の idnam に設定された文字数が 9 以上です。
DCMCFRTN_72074	-13074	dcmcf_tlsleopt の idnam に設定された文字列中に不正な文字があります。
DCMCFRTN_72076	-13076	infcnt に 1 が設定されていません。

dc_mcf_tlsln － サーバ型コネクションの確立要求の受付状態取得（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_tlsln (DCLONG action, DCLONG mcfid, char *resv01,
                  DCLONG *infcnt, dcmcf_lninf *inf, char *resv02)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_tlsln (action, mcfid, resv01, infcnt, inf, resv02)
DCLONG          action;
DCLONG          mcfid;
char            *resv01;
DCLONG          *infcnt;
dcmcf_lninf     *inf;
char            *resv02;
```

機能

サーバ型コネクションの確立要求の受付状態を取得します。

UAP で値を設定する引数

●action

DCNOFLAGS を設定します。

●mcfid

処理対象の MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 1～239 です。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。

例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●resv01

NULL を設定します。

●infcnt

サーバ型コネクションの確立要求の受付状態を格納する領域 dcmcf_lninf の個数として，1 を設定します。

処理終了後は，該当する MCF 通信サービスの個数が返されます。

●inf

サーバ型コネクションの確立要求の受付状態を格納する領域 dcmcf_lninf を設定します。

「構造体 dcmcf_lninf のサイズ×infcnt」バイト数分の領域が必要です。

●resv02

NULL を設定します。

OpenTP1 から値が返される引数

●infcnt

この関数の対象となった MCF 通信サービスの個数が返されます。

●inf

この関数の対象となった MCF 通信サービスのサーバ型コネクションの確立要求の受付状態が構造体 dcmcf_lninf で返されます。

構造体の形式を次に示します。

```
typedef struct {
                DCLONG    status;       …受付状態
                char      resv01[60];   …予備領域
                } dcmcf_lninf;
```

- status

  サーバ型コネクションの確立要求の受付状態として，次の値が設定されます。

  DCMCF_LNST_LISTEN

  受付開始状態

  DCMCF_LNST_RETRY

  受付開始処理中状態

  DCMCF_LNST_ONLN_W

  受付開始要求待ち状態

  DCMCF_LNST_INIT

  受付終了状態

  それぞれの状態のときに使用できるライブラリ関数を，次の表に示します。

status の設定値	使用できるライブラリ関数	
	dc_mcf_tonln	dc_mcf_tofln
DCMCF_LNST_LISTEN	×	○
DCMCF_LNST_RETRY	×	○

status の設定値	使用できるライブラリ関数	
	dc_mcf_tonln	dc_mcf_tofln
DCMCF_LNST_ONLN_W	○	○
DCMCF_LNST_INIT	○	×

（凡例）

○：使用できます。

×：使用できません。

- resv01

  領域をヌル文字で埋めます。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tlsln 関数が受け付けられません。
DCMCFRTN_71002	-12002	MCF が終了処理中のため，dc_mcf_tlsln 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tlsln 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71009	-12009	dc_mcf_tlsln 関数が，該当する MCF 通信プロセスではサポートされていません。
DCMCFRTN_71010	-12010	MCF 通信プロセスにサーバ型コネクションの確立要求の状態取得を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_72050	-13050	action に DCNOFLAGS が設定されていません。
DCMCFRTN_72052	-13052	resv01 に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72056	-13056	infcnt に NULL が設定されています。
DCMCFRTN_72057	-13057	inf に NULL が設定されています。
DCMCFRTN_72061	-13061	mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72076	-13076	infcnt に 1 が設定されていません。

dc_mcf_tofln － サーバ型コネクションの確立要求の受付終了（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_tofln (DCLONG action, DCLONG mcfid, char *resv01,
                  char *resv02)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_tofln (action, mcfid, resv01, resv02)
DCLONG    action;
DCLONG    mcfid;
char      *resv01;
char      *resv02;
```

機能

サーバ型コネクションの確立要求の受付を終了します。

UAP で値を設定する引数

●action

DCNOFLAGS を設定します。

●mcfid

処理対象の MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 1～239 です。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●resv01，resv02

NULL を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tofln 関数が受け付けられません。

リターン値	リターン値（数値）	意味
DCMCFRTN_71002	-12002	MCF が終了処理中のため，dc_mcf_tofln 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tofln 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71009	-12009	dc_mcf_tofln 関数が，該当する MCF 通信プロセスではサポートされていません。
DCMCFRTN_71010	-12010	MCF 通信プロセスにサーバ型コネクションの確立要求の受付終了を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_72050	-13050	action に DCNOFLAGS が設定されていません。
DCMCFRTN_72052	-13052	resv01 に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72061	-13061	mcfid に 0 未満または 240 以上の値が設定されています。

dc_mcf_tonln － サーバ型コネクションの確立要求の受付開始（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_tonln (DCLONG action, DCLONG mcfid, char *resv01,
                  char *resv02)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_tonln (action, mcfid, resv01, resv02)
DCLONG    action;
DCLONG    mcfid;
char      *resv01;
char      *resv02;
```

機能

サーバ型コネクションの確立要求の受付を開始します。

UAP で値を設定する引数

●action

DCNOFLAGS を設定します。

●mcfid

処理対象の MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 1～239 です。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●resv01，resv02

NULL を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tonln 関数が受け付けられません。

リターン値	リターン値（数値）	意味
DCMCFRTN_71002	-12002	MCF が終了処理中のため，dc_mcf_tonln 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tonln 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71009	-12009	dc_mcf_tonln 関数が，該当する MCF 通信プロセスではサポートされていません。
DCMCFRTN_71010	-12010	MCF 通信プロセスにサーバ型コネクションの確立要求の受付開始を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_72050	-13050	action に DCNOFLAGS が設定されていません。
DCMCFRTN_72052	-13052	resv01 に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72061	-13061	mcfid に 0 未満または 240 以上の値が設定されています。

4 COBOL-UAP 作成用プログラムインタフェース

この章では，TP1/NET/TCP/IP で使用できる，COBOL-UAP 作成用プログラムインタフェースについて説明します。

COBOL-UAP 作成用プログラムインタフェースの一覧

TP1/NET/TCP/IP で使用する COBOL-UAP 作成用プログラムインタフェースについて，COBOL 言語，およびデータ操作言語に分けて説明します。

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

COBOL 言語のプログラムインタフェース

COBOL 言語で UAP を作成する場合，OpenTP1 システムの関数に対応しているプログラムを，CALL 文で呼び出して UAP を作成します。

COBOL 言語のプログラムインタフェースの一覧を次の表に示します。

表 4-1　COBOL 言語のプログラムインタフェースの一覧

プログラム名	データ名 A に設定する要求コード	機能
CBLDCMCF	'CONTEND△'	継続問い合わせ応答の終了
	'RECEIVE△'	メッセージの受信
	'RECVSYNC'	同期型メッセージの受信
	'REPLY△△△'	応答メッセージの送信
	'RESEND△△'	メッセージの再送
	'SEND△△△△'	一方送信メッセージの送信
	'SENDRECV'	同期型メッセージの送受信
	'SENDSYNC'	同期型メッセージの送信
	'TACTCN△△'	コネクションの確立
	'TACTLE△△'	論理端末の閉塞解除
	'TDCTCN△△'	コネクションの解放
	'TDCTLE△△'	論理端末の閉塞
	'TEMPGET△'	一時記憶データの受け取り
	'TEMPPUT△'	一時記憶データの更新
	'TLSCN△△△'	コネクションの状態取得
	'TLSLE△△△'	論理端末の状態取得
	'TLSLN△△△'	サーバ型コネクションの確立要求の受付状態取得
	'TOFLN△△△'	サーバ型コネクションの確立要求の受付終了

プログラム名	データ名 A に設定する要求コード	機能
CBLDCMCF	'TONLN△△△'	サーバ型コネクションの確立要求の受付開始

その他のプログラムについては，マニュアル「OpenTP1 プログラム作成リファレンス COBOL 言語編」を参照してください。

データ操作言語のプログラムインタフェース

データ操作言語（DML）を使用した，通信文について説明します。データ操作言語の形式の詳細については，マニュアル「OpenTP1 プログラム作成リファレンス COBOL 言語編」を参照してください。

データ操作言語のプログラムインタフェースの一覧を，次の表に示します。

表 4-2　データ操作言語のプログラムインタフェースの一覧

通信文		機能	対応する C 言語のライブラリ関数
データコミュニケーション機能	RECEIVE	メッセージの受信	「表 4-4　RECEIVE 文（データコミュニケーション機能）の指定と C 言語のライブラリ関数との対応」，および「表 4-5　SEND 文（データコミュニケーション機能）の指定と C 言語のライブラリ関数との対応」を参照してください。
	SEND	メッセージの送信	
サービス機能	DISABLE	継続問い合わせ応答の終了	dc_mcf_contend
	RECEIVE	一時記憶データの受け取り	dc_mcf_tempget
	SEND	一時記憶データの更新	dc_mcf_tempput

注

dc_mcf_resend（メッセージの再送）に対応するデータ操作言語のインタフェースはありません。

そのほかの通信文については，マニュアル「OpenTP1 プログラム作成リファレンス COBOL 言語編」を参照してください。

通信記述項について

TP1/NET/TCP/IP のメッセージ送受信の通信文で，通信記述項に指定できる句の指定要否を，次の表に示します。

表 4-3　通信記述項に指定できる句の指定要否

データ名を指定する句	データ領域の値の設定元									
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
STATUS KEY	B	B	B	B	B	B	B	B	B	B
SYMBOLIC TERMINAL	B	U_1※	U_1	U_1	−	U_1	U_1	×	×	×

データ名を指定する句	データ領域の値の設定元									
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
MESSAGE DATE	B	B	B	−	−	−	−	−	−	−
MESSAGE TIME	B	B	B	−	−	−	−	−	−	−
SYNCHRONOUS MODE	U_2	U_2	U_1	U_2	U_2	U_1	U_1	−	−	−
SWITCHING MODE	−	−	−	U_2	−	−	−	−	−	−
NEXT TRANSACTION	−	−	−	−	U_2	−	−	−	−	−
DETAIL MODE	−	−	−	U_2	U_2	−	−	−	−	−
WAITING TIME	−	−	U_2	−	−	U_2	U_2	−	−	−

（凡例）

1.：先頭セグメントの非同期受信（RECEIVE）

2.：中間，最終セグメントの非同期受信（RECEIVE）

3.：単一セグメントの同期受信（RECEIVE）

4.：一方送信メッセージの単一セグメントの非同期送信（SEND）

5.：応答メッセージの単一セグメントの非同期送信（SEND）

6.：単一セグメントの同期送信（SEND）

7.：単一セグメントの同期送受信（SEND）

8.：継続問い合わせ応答の終了（DISABLE）

9.：一時記憶データの受け取り（RECEIVE）

10.：一時記憶データの更新（SEND）

B：OpenTP1 から値が返されます。省略できます。

U_1：UAP で値を設定します。省略できません。

U_2：UAP で値を設定します。省略できます。

−：該当しません。設定しても無効です。

×：空白以外の値を設定するとエラーリターンします。

注※

先頭メッセージ受信時の RECEIVE 文と同一の CD 句を用いた場合は省略できます。

データコミュニケーション機能と C 言語のライブラリ関数の対応

RECEIVE 文（データコミュニケーション機能）の指定と C 言語のライブラリ関数との対応を，次の表に示します。

表 4-4　RECEIVE 文（データコミュニケーション機能）の指定と C 言語のライブラリ関数との対応

FOR 句		SYNCHRONOUS MODE 句		対応する C 言語のライブラリ関数
INPUT	I-O	SYNC，または'1'	ASYNC，'0'，'△'，または省略	
○	－	－	○	dc_mcf_receive
－	○	－	○	
－	○	○	－	dc_mcf_recvsync

（凡例）

○：指定あり

－：指定なし

SEND 文（データコミュニケーション機能）の指定と C 言語のライブラリ関数との対応を，次の表に示します。

表 4-5　SEND 文（データコミュニケーション機能）の指定と C 言語のライブラリ関数との対応

FOR 句		SYNCHRONOUS MODE 句		BEFORE 句	対応する C 言語のライブラリ関数
OUTPUT	I-O	SYNC，または'1'	ASYNC，'0'，'△'，または省略		
○	－	－	○	－	dc_mcf_send
－	○	－	○	－	dc_mcf_reply
－	○	○	－	－	dc_mcf_sendsync
－	○	○	－	○	dc_mcf_sendrecv

（凡例）

○：指定あり

－：指定なし

CBLDCMCF('CONTEND△') － 継続問い合わせ応答の終了（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE ’CONTEND ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(16) VALUE LOW-VALUE.
```

機能

継続問い合わせ応答を終了します。

UAP で値を設定するデータ領域

●データ名 A

継続問い合わせ応答の終了を示す要求コード「VALUE 'CONTEND△'」を設定します。

●データ名 C

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，５けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
72000	＜ MHP の実行でリターンした場合＞ 先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('CONTEND△')を呼び出しています。
	＜ SPP の実行でリターンした場合＞ SPP では CBLDCMCF('CONTEND△')を呼び出せません。
72016	データ名 C に設定した値が間違っています。

ステータスコード	意味
72028	データ名 A に設定した値が間違っています。
72101	継続問い合わせ応答型でないアプリケーションで，CBLDCMCF('CONTEND△')を呼び出しています。
72107	CBLDCMCF('CONTEND△')を 2 回以上呼び出しています。
72111	次起動アプリケーションを設定して CBLDCMCF('REPLY△△△')を呼び出したあと，CBLDCMCF('CONTEND△')を呼び出しています。
	継続問い合わせ応答型のアプリケーション名を設定して CBLDCMCF('EXECAP△△')を呼び出したあと，CBLDCMCF('CONTEND△')を呼び出しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('RECEIVE△') － メッセージの受信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE ’RECEIVE ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP.
   02 データ名H   PIC X(4)  VALUE SPACE.
   02 データ名I   PIC X(4)  VALUE SPACE.
   02 データ名J   PIC X(4)  VALUE SPACE.
   02 データ名K   PIC X(4)  VALUE SPACE.
   02 データ名L   PIC X(8)  VALUE SPACE.
   02 データ名M1  PIC X(4)  VALUE SPACE.
   02 データ名M2  PIC X(8)  VALUE SPACE.
   02 データ名M3  PIC X(4)  VALUE SPACE.
   02 データ名M4  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M6  PIC X(1)  VALUE SPACE.
   02 データ名M7  PIC X(1).
   02 データ名N   PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O   PIC X(4)  VALUE SPACE.
   02 データ名P   PIC X(8).
   02 データ名Q   PIC X(8).
   02 データ名R   PIC X(8)  VALUE SPACE.
   02 データ名T   PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名U   PIC 9(x)  COMP.
   02 データ名V   PIC X(x).
   02 データ名Y   PIC X(n).
```

機能

論理端末に届いたメッセージのうち，一つのセグメントを受信します。セグメントの数だけCBLDCMCF('RECEIVE△')を呼び出すと，一つの論理メッセージを受信できます。

受信できるメッセージの一つのセグメントの最大長は，1 メガバイトです。

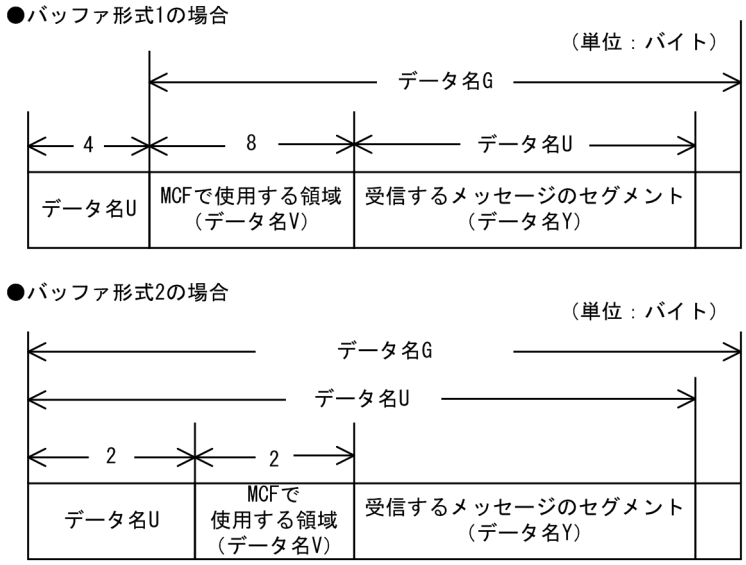
CBLDCMCF('RECEIVE△')で受信できるメッセージの種類を次に示します。

- 相手システムから送信されたメッセージ

- MCF イベント
- アプリケーション起動で渡されたメッセージ

TP1/NET/TCP/IP を使用して通信する場合，相手システムから送信されるメッセージは，常に単一セグメントで構成されます。

セグメントを受信する領域（一意名 3 で示す領域）の形式を次に示します。



UAP で値を設定するデータ領域

●データ名 A

メッセージの受信を示す要求コード「VALUE 'RECEIVE△'」を設定します。

●データ名 C

メッセージの先頭セグメントを受信するかどうかを設定します。次のどちらかを設定します。

VALUE 'FRST'

先頭セグメントを受信する場合や，メッセージが単一セグメントの場合に設定します。

VALUE 'SEG△'

中間セグメントまたは最終セグメントを受信する場合に設定します。

●データ名 D

空白を設定します。

●データ名 G

セグメントを受信する領域の長さを設定します。

●データ名 H，データ名 I，データ名 J，データ名 K，データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'
　バッファ形式 1 を使用する場合に設定します。

VALUE '2'
　バッファ形式 2 を使用する場合に設定します。

空白
　省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O

空白を設定します。

●データ名 P

中間セグメントまたは最終セグメントを受信する場合は，入力元の論理端末名称を設定します。先頭セグメントの受信時に返された論理端末名称を設定してください。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

先頭セグメントの受信処理終了後，データ名 P には OpenTP1 から値が返されます。

●データ名 Q

MCF で使用する領域です。

●データ名 R

空白を設定します。

●データ名 T

MCF で使用する領域です。

●データ名 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 E

メッセージを受信した日付が YYYYMMDD（YYYY：西暦年　MM：月　DD：日）の形式で返されます。

●データ名 F

メッセージを受信した時刻が HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で返されます。

●データ名 P

先頭セグメントまたは単一セグメントを受信する場合，入力元の論理端末名称が返されます。

論理端末名称は最大 8 バイトの長さです。8 バイトに満たない場合，論理端末名称の後ろが空白で埋められます。

中間セグメントまたは最終セグメントを受信する場合は，ここで返された論理端末名称をデータ名 P に設定します。

●データ名 U

【バッファ形式 1 の場合】 PIC 9(9)

受信したセグメントの長さが返されます。

【バッファ形式 2 の場合】 PIC 9(4)

受信したセグメントの長さ＋ 4 が返されます。

●データ名 Y

受信したセグメントの内容が返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71000	先頭セグメントを受信する CBLDCMCF('RECEIVE△')を 2 回以上呼び出しています。

ステータスコード	意味
71000	中間セグメントまたは最終セグメントを受信する場合は，データ名 C に「VALUE 'SEG△'」を設定して CBLDCMCF('RECEIVE△')を呼び出してください。
71001	メッセージの最終セグメントを受信したあとで，次のセグメントを受信する CBLDCMCF('RECEIVE△')を呼び出しています。直前に呼び出した CBLDCMCF('RECEIVE△')でメッセージはすべて受信しました。このステータスコードが返されたあとに，再び CBLDCMCF('RECEIVE△')を呼び出した場合は，ステータスコード 72000 が返されます。
71002	メッセージキューからの入力処理中に障害が発生しました。
	メッセージキューが閉塞されています。
71108	プロセスのローカルメモリが不足しています。
72000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，中間セグメントまたは最終セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出しています。先頭セグメントを受信する場合は，データ名 C に「VALUE 'FRST'」を設定して CBLDCMCF('RECEIVE△')を呼び出してください。 • ステータスコード 71001 が返されたあとに，再び CBLDCMCF('RECEIVE△')を呼び出しています。
	＜ SPP の実行でリターンした場合＞ SPP では CBLDCMCF('RECEIVE△')を呼び出せません。
72001	データ名 P に設定した論理端末名称が間違っています。
72013	データ名 G の指定値を超えるセグメントを受信しました。データ名 G の指定値を超えた部分は切り捨てられました。
	＜バッファ形式 2 の場合＞ 32763 バイトを超えるセグメントを受信しました。32763 バイトを超えた部分は切り捨てられました。
72016	データ名 D に設定した値が間違っています。
	データ名 N またはデータ名 T に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72025	データ名 C に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72036	データ名 G の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('RECVSYNC') － 同期型メッセージの受信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01  一意名1.
    02  データ名A    PIC X(8)  VALUE ’RECVSYNC’.
    02  データ名B    PIC X(5).
    02  FILLER       PIC X(3).
    02  データ名C    PIC X(4).
    02  データ名D    PIC X(4)  VALUE SPACE.
    02  データ名E    PIC 9(8).
    02  データ名F    PIC 9(8).
    02  データ名G    PIC 9(9)  COMP.
    02  データ名H    PIC X(4)  VALUE SPACE.
    02  データ名I    PIC X(4)  VALUE SPACE.
    02  データ名J    PIC X(4)  VALUE SPACE.
    02  データ名K    PIC X(4)  VALUE SPACE.
    02  データ名L    PIC X(8)  VALUE SPACE.
    02  データ名M1   PIC X(4)  VALUE SPACE.
    02  データ名M2   PIC X(8)  VALUE SPACE.
    02  データ名M3   PIC X(4)  VALUE SPACE.
    02  データ名M4   PIC 9(9)  COMP VALUE ZERO.
    02  データ名M5   PIC 9(9)  COMP※.
    02  データ名M6   PIC X(1)  VALUE SPACE.
    02  データ名M7   PIC X(1).
    02  データ名N    PIC X(14) VALUE LOW-VALUE.
01  一意名2.
    02  データ名O    PIC X(4)  VALUE SPACE.
    02  データ名P    PIC X(8).
    02  データ名Q    PIC X(8)  VALUE SPACE.
    02  データ名R    PIC X(8)  VALUE SPACE.
    02  データ名T    PIC X(28) VALUE LOW-VALUE.
01  一意名3.
    02  データ名U    PIC 9(x)  COMP.
    02  データ名V    PIC X(x).
    02  データ名Y    PIC X(n).
```

注※

負の値を設定する場合，「PIC S9(9) COMP」としてください。

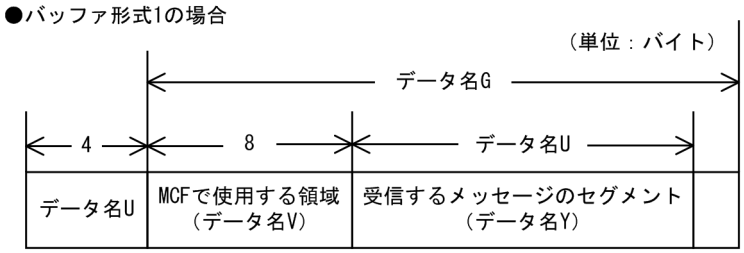
機能

相手システムから同期型でメッセージを受信します。相手システムから送信されるメッセージは，常に一つのセグメントで構成されます。

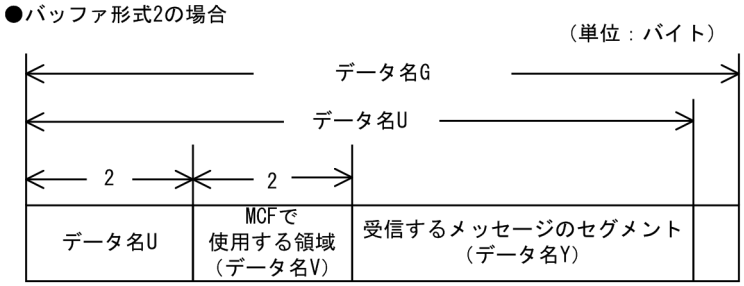
受信できるメッセージの一つのセグメントの最大長は，1 メガバイトです。

セグメントを受信する領域（一意名 3 で示す領域）の形式を次に示します。

●バッファ形式1の場合



●バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

同期型のメッセージの受信を示す要求コード「VALUE 'RECVSYNC'」を設定します。

●データ名 C

「VALUE 'FRST'」を設定します。

●データ名 D

空白を設定します。

●データ名 G

セグメントを受信する領域の長さを設定します。

●データ名 H，データ名 I，データ名 J，データ名 K，データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4

0 を設定します。

●データ名 M5

CBLDCMCF('RECVSYNC')を呼び出してから終了するまでの，監視時間を設定します。0 を設定した場合は，MCF マネジャ定義の UAP 共通定義で指定した同期型受信監視時間（mcfmuap -t recvtim）が設定されます。

負の値を設定した場合は，時間を監視しません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，設定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，設定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を指定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O

空白を設定します。

●データ名 P

入力元の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 Q，データ名 R

空白を設定します。

●データ名 T

MCF で使用する領域です。

●データ名 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 E

メッセージを受信した日付が YYYYMMDD（YYYY：西暦年　MM：月　DD：日）の形式で返されます。

●データ名 F

メッセージを受信した時刻が HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で返されます。

●データ名 U

【バッファ形式 1 の場合】 PIC 9(9)

受信したセグメントの長さが返されます。

【バッファ形式 2 の場合】 PIC 9(4)

受信したセグメントの長さ＋ 4 が返されます。

●データ名 Y

受信したセグメントの内容が返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	メッセージの最終セグメントを受け取ったあとで，次のセグメントを受信する CBLDCMCF('RECVSYNC')を呼び出しています。直前に呼び出した CBLDCMCF('RECVSYNC')でメッセージはすべて受信しました。このステータスコードが返されたあとに，再び CBLDCMCF('RECVSYNC')を呼び出した場合は，ステータスコード 72000 が返されます。
71002	MCF が終了処理中のため，メッセージの受信を受け付けられません。
71108	メッセージ受信に必要な管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('RECVSYNC')を呼び出しています。
72001	データ名 P に設定した論理端末名称が間違っています。

ステータスコード	意味
72001	データ名 P に設定した入力元の論理端末名称は，MCF で定義していません。
	CBLDCMCF('RECVSYNC')を呼び出せない論理端末を設定しています。
	＜問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用している（コネクション定義（mcftalccn -l）の replymsg オペランドに yes を指定）場合＞ 問い合わせ応答中または継続問い合わせ応答中のため受け付けられません。
72012	MCF バッファグループ定義のバッファ長が不足しました。
	MCF 通信プロセスは相手システムからメッセージを受信しましたが，UAP への応答連絡で RPC 通信の送信可能な上限値を超えました。
72013	データ名 G の指定値を超えるセグメントを受信しました。データ名 G の指定値を超えた部分は切り捨てられました。
	＜バッファ形式 2 の場合＞ 32763 バイトを超えるセグメントを受信しました。32763 バイトを超えた部分は切り捨てられました。
72016	データ名 N またはデータ名 T に設定した値が間違っています。
	データ名 Q に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72025	データ名 C に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72036	データ名 G の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
73001	データ名 M5 に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
	前回の通信から無通信監視時間（コネクション定義（mcftalccn -k）の notrftime オペランド指定値）が経過しましたが，相手システムからの通信がありません。
	相手システムからメッセージを受信しましたが，受信バッファを確保できませんでした。
	入力元の論理端末で MCF 通信プロセスの内部障害が発生しました。
73002	コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指定）場合に，MHP でメッセージ受信後にコネクションが再確立されたため，受信を取り消しました。
73005	データ名 M5 に設定した時間が経過しましたが，論理端末からの応答がありません。
73010	入力メッセージ編集 UOC で障害が発生しました。
	メッセージの読み込み時に障害が発生しました。
73015	指定した論理端末は，ほかの UAP で仕掛り中です。
73018	データ名 M5 に設定した値が間違っています。

ステータスコード	意味
730200	入力元の論理端末は停止しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('REPLY△△△') － 応答メッセージの送信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE ’REPLY   ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)  VALUE SPACE.
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP VALUE ZERO.
   02 データ名H   PIC X(4).
   02 データ名I   PIC X(4)  VALUE SPACE.
   02 データ名J   PIC X(4)  VALUE SPACE.
   02 データ名K   PIC X(4)  VALUE SPACE.
   02 データ名L   PIC X(8)  VALUE SPACE.
   02 データ名M1  PIC X(4)  VALUE SPACE.
   02 データ名M2  PIC X(8)  VALUE SPACE.
   02 データ名M3  PIC X(4)  VALUE SPACE.
   02 データ名M4  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M6  PIC X(1)  VALUE SPACE.
   02 データ名M7  PIC X(1).
   02 データ名N   PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O   PIC X(4)  VALUE SPACE.
   02 データ名P   PIC X(8)  VALUE SPACE.
   02 データ名Q   PIC X(8)  VALUE SPACE.
   02 データ名R   PIC X(8).
   02 データ名T   PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名U  PIC 9(x)  COMP.
   02 データ名V  PIC X(x)  VALUE LOW-VALUE.
   02 データ名W  PIC X(n).
```

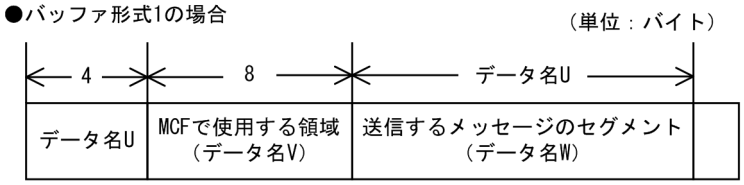
機能

メッセージを入力した論理端末に対して応答メッセージを送信します。応答メッセージは，一つのセグメントで構成されます。

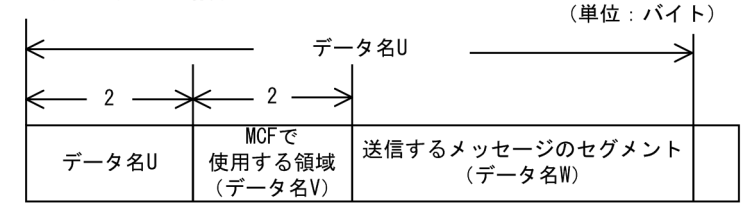
送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

セグメントを送信する領域（一意名 3 で示す領域）の形式を次に示します。

●バッファ形式1の場合



●バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

応答メッセージの送信を示す要求コード「VALUE 'REPLY△△△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E，データ名 F

MCF で使用する領域です。

●データ名 G

0 を設定します。

●データ名 H

単一セグメントの送信を示す「VALUE 'EMI△'」を設定します。

●データ名 I，データ名 J

空白を設定します。

●データ名 K

出力通番を付けるかどうかを設定します。

VALUE 'SEQ△'

出力通番を付ける場合に設定します。

ただし，非応答型のアプリケーションの場合，出力通番を付けません。

VALUE 'NSEQ'

出力通番を付けない場合に設定します。

空白

省略されたものとして，「VALUE 'NSEQ'」（出力通番は付けない）が設定されます。

●データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O，データ名 P，データ名 Q

空白を設定します。

●データ名 R

＜継続問い合わせ応答形態の場合＞

次起動アプリケーションを設定します。次起動アプリケーションは最大 8 バイトの長さです。8 バイトに満たない場合，次起動アプリケーションの後ろを空白で埋めてください。

空白を設定した場合，実行中のアプリケーションを次のメッセージ受信時に再び起動します。

CBLDCMCF('REPLY△△△')を呼び出すサービスで CBLDCMCF('CONTEND△')を呼び出す場合はデータ名 R に空白を設定してください。

継続問い合わせ応答を引き継いだエラーイベントで CBLDCMCF('REPLY△△△')を発行する際に，データ名 R に空白を設定した場合，継続問い合わせ応答を終了します。ただし，継続問い合わせ応答を引き継いだエラーイベントで，CBLDCMCF('EXECAP△△')を発行して継続問い合わせ応答型のアプリケーションを起動し，起動先のアプリケーションで CBLDCMCF('REPLY△△△')を発行する際に，データ名 R に空白を設定した場合，継続問い合わせ応答を終了しないで，CBLDCMCF('EXECAP△△')を発行したときに設定した継続問い合わせ応答型のアプリケーションを次のメッセージ受信時に再び起動します。

＜継続問い合わせ応答形態以外の場合＞

空白を設定します。

●データ名 T

MCF で使用する領域です。

●データ名 U

【バッファ形式 1 の場合】 PIC 9(9)

送信するセグメントの長さを設定します。

【バッファ形式 2 の場合】 PIC 9(4)

送信するセグメントの長さ＋ 4 を設定します。

●データ名 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

●データ名 W

送信する応答メッセージのセグメントの内容を設定します。一つのセグメントで 32000 バイトまで送信できます。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71002	メッセージキューへの出力処理中に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	バッファ形式 1 の場合はデータ名 U に 32000 バイトを超える値を設定しています。 バッファ形式 2 の場合はデータ名 U に 32004 バイトを超える値を設定しています。
	MCF が終了処理中のため，メッセージの送信を受け付けられません。
71003	メッセージキューが満杯です。
71004	メッセージを格納するバッファをメモリ上に確保できませんでした。

ステータスコード	意味
71108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('REPLY△△△')を呼び出しています。 • 非応答型のアプリケーションからの問い合わせ応答をしない（UAP 共通定義（mcfmuap -c）の noansreply オペランドに no を指定）場合に，非応答型のアプリケーションから CBLDCMCF('REPLY△△△')を呼び出しています。
	＜ SPP の実行でリターンした場合＞ SPP では CBLDCMCF('REPLY△△△')を呼び出せません。
72001	入力元論理端末は，応答メッセージの送信をサポートしていません。
	入力元論理端末は，問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用していません（コネクション定義（mcftalccn -l）の replymsg オペランドを省略，または no を指定）。
72008	応答型のアプリケーションから，CBLDCMCF('EXECAP△△')を呼び出して応答型のアプリケーションを起動したあとで，CBLDCMCF('REPLY△△△')を呼び出しています。
	継続問い合わせ応答型のアプリケーションから，CBLDCMCF('EXECAP△△')で継続問い合わせ応答型のアプリケーションを起動したあとで，CBLDCMCF('REPLY△△△')を呼び出しています。
72011	継続問い合わせ応答型でないアプリケーションが，次起動アプリケーションを設定して，CBLDCMCF('REPLY△△△')を呼び出しています。
72016	データ名 N またはデータ名 T に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72017	データ名 K に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72020	データ名 I に設定した値が間違っています。
72026	データ名 H に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72041	バッファ形式 1 の場合はデータ名 U に 0 バイト，またはマイナス値を設定しています。バッファ形式 2 の場合はデータ名 U に 0 から 4 バイト，またはマイナス値を設定しています。
72044	CBLDCMCF('CONTEND△')を呼び出したあとで，次起動アプリケーションを設定して CBLDCMCF('REPLY△△△')を呼び出しています。
72045	データ名 R に，継続問い合わせ応答型でないアプリケーションのアプリケーション名を設定して CBLDCMCF('REPLY△△△')を呼び出しています。

ステータスコード	意味
72046	データ名 R に設定したアプリケーション名が異なる CBLDCMCF('REPLY△△△')を，2 回以上呼び出しています。
72047	データ名 R に，アプリケーション属性定義（mcfaalcap）に定義されていないアプリケーション名を設定して CBLDCMCF('REPLY△△△')を呼び出しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('RESEND△△') － メッセージの再送（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE ’RESEND ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)  VALUE SPACE.
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP VALUE ZERO.
   02 データ名H   PIC X(4)  VALUE SPACE.
   02 データ名I   PIC X(4)  VALUE SPACE.
   02 データ名J   PIC X(4).
   02 データ名K   PIC X(4).
   02 データ名L   PIC X(8)  VALUE SPACE.
   02 データ名M1  PIC X(4)  VALUE SPACE.
   02 データ名M2  PIC X(8)  VALUE SPACE.
   02 データ名M3  PIC X(4)  VALUE SPACE.
   02 データ名M4  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M6  PIC X(1)  VALUE SPACE.
   02 データ名M7  PIC X(1)  VALUE SPACE.
   02 データ名N   PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O   PIC X(4)  VALUE ’OUT ’.
   02 データ名P   PIC X(8).
   02 データ名Q   PIC X(8)  VALUE SPACE.
   02 データ名R   PIC X(8)  VALUE SPACE.
   02 データ名S   PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名T   PIC X(8).
   02 データ名U   PIC X(4).
   02 データ名V   PIC 9(9)  COMP.
   02 データ名W   PIC X(4).
   02 データ名X   PIC X(12) VALUE LOW-VALUE.
```

機能

以前に送信したメッセージを，再び送信します。再送するメッセージは，以前に送信したメッセージとは別の，新しいメッセージとして扱います。どのメッセージを再送するかは，次に示す送信済みメッセージの情報で選択できます。

- 出力先の論理端末名称

- メッセージ出力通番
- メッセージ種別（一般の一方送信，優先の一方送信）

対象としたメッセージが以前に送信されていない場合は，CBLDCMCF('RESEND△△')はステータスコード 70904 を返します。また，メッセージキュー（ディスクキュー）内に対象のメッセージがない場合も，ステータスコード 70904 を返します。このため，使用するメッセージキューの種別ではディスクキューを指定するとともに，メッセージキューの大きさの定義では余裕を持った値を指定してください。

UAP で値を設定するデータ領域

●データ名 A

メッセージの再送を示す要求コード「VALUE 'RESEND△△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E，データ名 F

MCF で使用する領域です。

●データ名 G

0 を設定します。

●データ名 H，データ名 I

空白を設定します。

●データ名 J

一般として再送するか優先として再送するかを設定します。

VALUE 'NORM'
　一般の一方送信メッセージとして再送する場合に設定します。

VALUE 'PRIO'
　優先の一方送信メッセージとして再送する場合に設定します。

空白
　省略されたものとして，「VALUE 'NORM'」（一般の一方送信メッセージとして再送）が設定されます。

●データ名 K

再送するメッセージに出力通番を付け直すかどうかを設定します。

VALUE 'SEQ△'
　再送するメッセージに出力通番を付け直す場合に設定します。

VALUE 'NSEQ'

再送するメッセージに出力通番を付け直さない場合に設定します。

空白

省略されたものとして，「VALUE 'NSEQ'」（出力通番を付け直さない）が設定されます。

●データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 6，データ名 M7

空白を設定します。

●データ名 N

MCF で使用する領域です。

●データ名 O

一方送信を示す「VALUE 'OUT△'」を設定します。

●データ名 P

出力先の論理端末名称を設定します。論理端末名称は最大８バイトの長さです。８バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 Q，データ名 R

空白を設定します。

●データ名 S

MCF で使用する領域です。

●データ名 T

再送するメッセージを検索するキーとして，以前に送信したメッセージの出力先の論理端末名称を設定します。論理端末名称は最大８バイトの長さです。８バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 U

再送するメッセージを検索するキーとして，以前に送信したメッセージの送信種別を設定します。

VALUE 'NORM'

一般の一方送信メッセージを対象とする場合に設定します。

VALUE 'PRIO'

優先の一方送信メッセージを対象とする場合に設定します。

空白

省略されたものとして，「VALUE 'NORM'」（一般の一方送信メッセージを対象）が設定されます。

「VALUE 'PRIO'」を設定した場合は，データ名 T に出力先の論理端末名称を設定できません。

●データ名 V

再送するメッセージを検索するキーとして，以前に送信したメッセージの出力通番を設定します。データ名 W に「VALUE 'LAST'」を設定した場合は，ここに設定した値は無効となります。

●データ名 W

最終出力通番を持つメッセージを再送するかどうかを設定します。

VALUE 'LAST'

最終出力通番を持つメッセージを再送する場合に設定します。この値を設定した場合は，データ名 V に設定した値は無効となります。

空白

データ名 V で設定した出力通番を持つメッセージを再送する場合に設定します。

●データ名 X

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
70904	出力通番使用論理端末数（MCF マネジャ共通定義（mcfmcomn）の-n オプション）を省略，または 0 を指定しています。
	データ名 T，データ名 U，またはデータ名 V に設定した値が間違っています。
	再送するメッセージは次に示す理由によって，再送できるメッセージの条件を満たしていません。 • 再送対象とするメッセージの送信時に出力通番を付けていません。 • 出力メッセージの割り当て先にメモリキューを使用しています（論理端末定義（mcftalcle -k）の quekind オペランドを省略，または memory を指定）。 • 論理端末が閉塞しているなどの要因によって，送信メッセージが送信済みになっていません。

ステータスコード	意味
70904	• 送信済みのメッセージがディスクキューに保持されていません（保持メッセージ数は，メッセージキューサービス定義の quegrp コマンドの-m オプションで指定します）。
	データ名 T で指定した論理端末に割り当てられているメッセージキューは，システム開始時に障害が発生したため，メモリキューを代用して縮退運転をしています。
70905	再送するメッセージのセグメントの長さが，UAP 共通定義（mcfmuap -e）で指定した値を超えています。
71002	メッセージキューへの出力処理中に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	MCF が終了処理中のため，メッセージの再送を受け付けられません。
71003	メッセージキューが満杯です。
71004	メッセージキューから取り出したメッセージを格納するバッファ（作業領域）を，メモリ上に確保できませんでした。
71108	メッセージを再送しようとしましたが，再送先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('RESEND△△')を呼び出しています。 • 非トランザクション属性の MHP から，CBLDCMCF('RESEND△△')を呼び出しています。
	＜ SPP の実行でリターンした場合＞ トランザクションでない SPP の処理から，CBLDCMCF('RESEND△△')を呼び出しています。
72001	データ名 P またはデータ名 T に設定した論理端末名称が間違っています。
	CBLDCMCF('RESEND△△')を呼び出せない論理端末を設定しています。
72016	データ名 J に設定した値が間違っています。
	データ名 M4 に設定した値が間違っています。
	データ名 U に設定した値が間違っています。
	データ名 N，データ名 S，またはデータ名 X に設定した値が間違っています。
72017	データ名 K に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

注意事項

メッセージの再送時には，MCF マネジャ定義の UAP 共通定義（mcfmuap）の-e オプションと-l オプションの指定値に注意してください。

-e オプション

-e オプションでは，CBLDCMCF('RESEND△△')で使用する作業領域の大きさを指定します。再送するメッセージのセグメントがこの作業領域より大きい場合，CBLDCMCF('RESEND△△')はメッセージを再送しないで，ステータスコード 70905 を返します。このため，-e オプションでは，セグメントの最大長よりも大きな値を設定しておいてください。

-l オプション

-l オプションでは，出力通番に関して指定します。この内容によっては，メッセージキューファイル内に同じ出力通番を持つメッセージが同時に存在する場合があります。この場合は，どのメッセージを再送するか保証できません。

CBLDCMCF('SEND△△△△') － 一方送信メッセージの送信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL  'CBLDCMCF'  USING  一意名1  一意名2  一意名3
```

DATA DIVISION の指定

```
01  一意名1.
    02  データ名A   PIC X(8)  VALUE 'SEND    '.
    02  データ名B   PIC X(5).
    02  FILLER      PIC X(3).
    02  データ名C   PIC X(4)  VALUE SPACE.
    02  データ名D   PIC X(4)  VALUE SPACE.
    02  データ名E   PIC 9(8).
    02  データ名F   PIC 9(8).
    02  データ名G   PIC 9(9)  COMP VALUE ZERO.
    02  データ名H   PIC X(4).
    02  データ名I   PIC X(4)  VALUE SPACE.
    02  データ名J   PIC X(4).
    02  データ名K   PIC X(4).
    02  データ名L   PIC X(8)  VALUE SPACE.
    02  データ名M1  PIC X(4)  VALUE SPACE.
    02  データ名M2  PIC X(8)  VALUE SPACE.
    02  データ名M3  PIC X(4)  VALUE SPACE.
    02  データ名M4  PIC 9(9)  COMP VALUE ZERO.
    02  データ名M5  PIC 9(9)  COMP VALUE ZERO.
    02  データ名M6  PIC X(1)  VALUE SPACE.
    02  データ名M7  PIC X(1).
    02  データ名N   PIC X(14) VALUE LOW-VALUE.
01  一意名2.
    02  データ名O   PIC X(4)  VALUE 'OUT '.
    02  データ名P   PIC X(8).
    02  データ名Q   PIC X(8)  VALUE SPACE.
    02  データ名R   PIC X(8)  VALUE SPACE.
    02  データ名T   PIC X(28) VALUE LOW-VALUE.
01  一意名3.
    02  データ名U  PIC 9(x)  COMP.
    02  データ名V  PIC X(x).
    02  データ名W  PIC X(n).
```

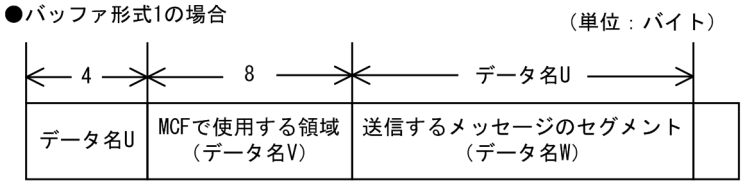
機能

相手システムへ一方送信メッセージを送信します。一方送信メッセージは，一つのセグメントで構成されます。

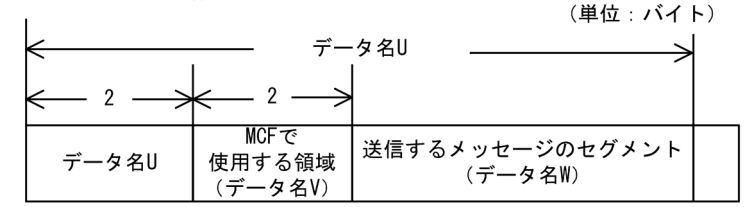
送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

セグメントを送信する領域（一意名 3 で示す領域）の形式を示します。

●バッファ形式1の場合



●バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

一方送信メッセージの送信を示す要求コード「VALUE 'SEND△△△△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E，データ名 F

MCF で使用する領域です。

●データ名 G

0 を設定します。

●データ名 H

単一セグメントの送信を示す「VALUE 'EMI△'」を設定します。

●データ名 I

空白を設定します。

●データ名 J

一般として送信するか優先として送信するかを設定します。

VALUE 'NORM'

一般の一方送信メッセージとして送信する場合に設定します。

VALUE 'PRIO'

優先の一方送信メッセージとして送信する場合に設定します。

空白

省略されたものとして，「VALUE 'NORM'」（一般の一方送信メッセージとして送信）が設定されます。

●データ名 K

出力通番を付けるかどうかを設定します。

VALUE 'SEQ△'

出力通番を付ける場合に設定します。

VALUE 'NSEQ'

出力通番を付けない場合に設定します。

空白

省略されたものとして，「VALUE 'NSEQ'」（出力通番は付けない）が設定されます。

●データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O

一方送信を示す「VALUE 'OUT△'」を設定します。

●データ名 P

出力先の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 Q，データ名 R

空白を設定します。

●データ名 T

MCF で使用する領域です。

●データ名 U

【バッファ形式 1 の場合】 PIC 9(9)

送信するセグメントの長さを設定します。

【バッファ形式 2 の場合】 PIC 9(4)

送信するセグメントの長さ＋ 4 を設定します。

●データ名 V

送信完了通知イベントを設定します。データ名 V とビットの設定値の関係を次に示します。

AIX，HP-UX の場合

【バッファ形式 1 のとき】 PIC X(8)

【バッファ形式 2 のとき】 PIC X(2)

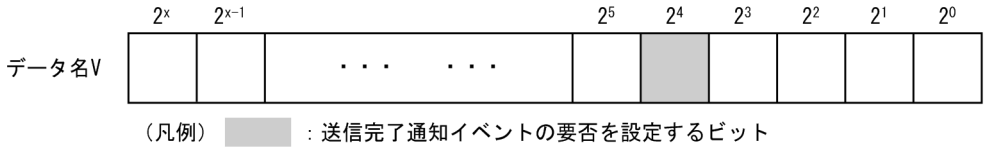
領域の 2^4 ビットに 1 を設定

送信完了通知イベントを通知させます。ただし，送信完了通知イベント処理用の MHP を MCF アプリケーション定義で指定していない場合は無効です。

領域の 2^4 ビットに 0 を設定

送信完了通知イベントを通知させません。

データ名 V とビットの設定値の関係を次に示します。



注

X の値は次のとおりです。

バッファ形式 1 の場合　X=63

バッファ形式 2 の場合　X=15

Linux，Windows の場合

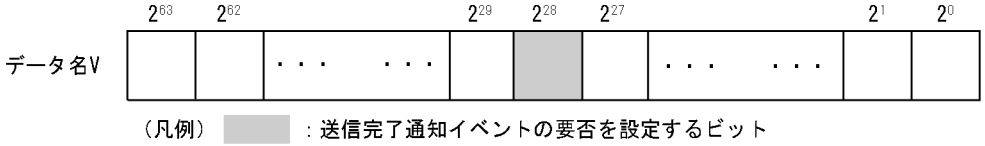
【バッファ形式 1 のとき】 PIC X(8)

領域の 2^{28} ビットに 1 を設定

送信完了通知イベントを通知させます。ただし，送信完了通知イベント処理用の MHP を MCF アプリケーション定義で指定していない場合は無効です。

領域の 2^{28} ビットに 0 を設定

送信完了通知イベントを通知させません。



【バッファ形式 2 のとき】 PIC X(2)

領域の 2^{12} ビットに 1 を設定

送信完了通知イベントを通知させます。ただし，送信完了通知イベント処理用の MHP を MCF アプリケーション定義で指定していない場合は無効です。

領域の 2^{12} ビットに 0 を設定

送信完了通知イベントを通知させません。

●データ名 W

送信するセグメントの内容を設定します。一つのセグメントで 32000 バイトまで送信できます。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71002	メッセージキューへの出力処理中に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	バッファ形式 1 の場合はデータ名 U に 32000 バイトを超える値を設定しています。バッファ形式 2 の場合はデータ名 U に 32004 バイトを超える値を設定しています。

ステータスコード	意味
71002	MCF が終了処理中のため，メッセージの送信を受け付けられません。
71003	メッセージキューが満杯です。
71004	メッセージを格納するバッファをメモリ上に確保できませんでした。
71108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	＜ MHP の実行でリターンした場合＞ 先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('SEND△△△△')を呼び出しています。
	＜ SPP の実行でリターンした場合＞ トランザクションでない SPP の処理から，CBLDCMCF('SEND△△△△')を呼び出しています。
72001	データ名 P に設定した論理端末名称が間違っています。
	CBLDCMCF('SEND△△△△')を呼び出せない論理端末を設定しています。
72016	データ名 N またはデータ名 T に設定した値が間違っています。
	データ名 J に設定した値が間違っています。
	データ名 M1 に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72017	データ名 K に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72020	データ名 I に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72026	データ名 H に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72041	バッファ形式 1 の場合はデータ名 U に 0 バイト，またはマイナス値を設定しています。バッファ形式 2 の場合はデータ名 U に 0 から 4 バイト，またはマイナス値を設定しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('SENDRECV') － 同期型メッセージの送受信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3 一意名4
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE ’SENDRECV’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)  VALUE SPACE.
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP.
   02 データ名H   PIC X(4).
   02 データ名I   PIC X(4)  VALUE SPACE.
   02 データ名J   PIC X(4)  VALUE SPACE.
   02 データ名K   PIC X(4)  VALUE SPACE.
   02 データ名L   PIC X(8)  VALUE SPACE.
   02 データ名M1  PIC X(4)  VALUE SPACE.
   02 データ名M2  PIC X(8)  VALUE SPACE.
   02 データ名M3  PIC X(4)  VALUE SPACE.
   02 データ名M4  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5  PIC 9(9)  COMP※.
   02 データ名M6  PIC X(1)  VALUE SPACE.
   02 データ名M7  PIC X(1).
   02 データ名N   PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O   PIC X(4)  VALUE ’IO  ’.
   02 データ名P   PIC X(8).
   02 データ名Q   PIC X(8)  VALUE SPACE.
   02 データ名R   PIC X(8)  VALUE SPACE.
   02 データ名T   PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名U   PIC 9(x)  COMP.
   02 データ名V   PIC X(x).
   02 データ名W   PIC X(n).
01 一意名4.
   02 データ名X   PIC 9(x)  COMP.
   02 データ名Y1  PIC X(x)  VALUE SPACE.
   02 データ名Y2  PIC X(1).
   02 データ名Z   PIC X(n).
```

注※

負の値を設定する場合，「PIC S9(9) COMP」としてください。

機能

同期型でメッセージを送信したあと，同期型でメッセージを受信します。

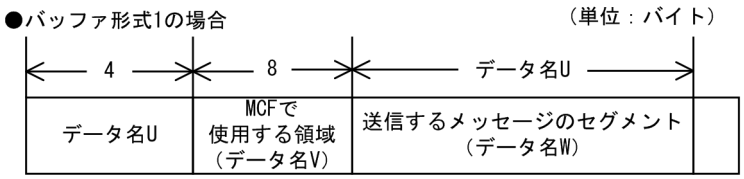
CBLDCMCF('SENDRECV')では，相手システムへ送るメッセージのセグメントを送信できます。

メッセージのセグメントを送信すると，CBLDCMCF('SENDRECV')は相手システムからの応答を待ちます。応答が届くと，そのメッセージのセグメントを受信します。

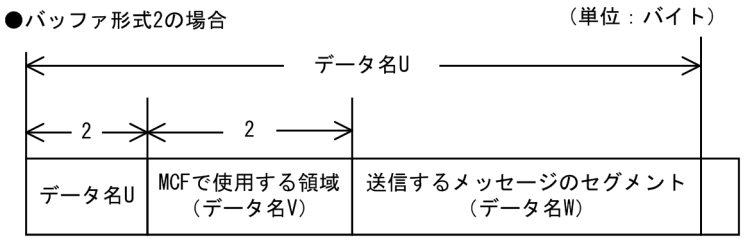
受信できるメッセージの一つのセグメントの最大長は，1 メガバイトです。また，送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

セグメントを送信する領域（一意名 3 で示す領域）の形式を次に示します。

●バッファ形式1の場合

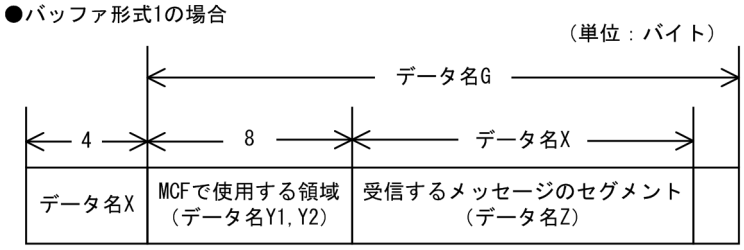


●バッファ形式2の場合

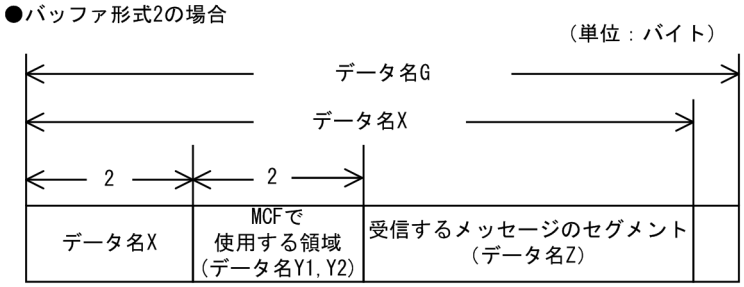


セグメントを受信する領域（一意名 4 で示す領域）の形式を次に示します。

●バッファ形式1の場合



●バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

同期型のメッセージの送受信を示す要求コード「VALUE 'SENDRECV'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 G

セグメントを受信する領域の長さを設定します。

●データ名 H

単一セグメントを示す「VALUE 'EMI△'」を設定します。

●データ名 I，データ名 J，データ名 K，データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4

0 を設定します。

●データ名 M5

CBLDCMCF('SENDRECV')を呼び出してから終了するまでの最大時間を設定します。0 を設定した場合，MCF マネジャ定義の UAP 共通定義で指定した同期型送受信監視時間（mcfmuap -t sndrcvtim）が設定されます。

負の値を設定した場合は，時間を監視しません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，設定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，設定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O

同期型のメッセージの送受信を示す「VALUE 'IO△△'」を設定します。

●データ名 P

メッセージを出力して応答を入力する論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 Q，データ名 R

空白を設定します。

●データ名 T

MCF で使用する領域です。

●データ名 U

【バッファ形式 1 の場合】 PIC 9(9)

送信するセグメントの長さを設定します。

【バッファ形式 2 の場合】 PIC 9(4)

送信するセグメントの長さ＋ 4 を設定します。

●データ名 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

●データ名 W

送信するセグメントの内容を設定します。一つのセグメントで 32000 バイトまで送信できます。

●データ名 Y1

【バッファ形式 1 の場合】 PIC X(7)

【バッファ形式 2 の場合】 PIC X(1)

空白を設定します。

●データ名 Y2

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 E

メッセージを受信した日付が YYYYMMDD（YYYY：西暦年　MM：月　DD：日）の形式で返されます。

●データ名 F

メッセージを受信した時刻が HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で返されます。

●データ名 X

【バッファ形式 1 の場合】 PIC 9(9)

受信したセグメントの長さが返されます。

【バッファ形式 2 の場合】 PIC 9(4)

受信したセグメントの長さ＋ 4 が返されます。

●データ名 Z

受信したセグメントの内容が返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71002	メッセージキューへの入出力処理時に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	バッファ形式 1 の場合はデータ名 U に 32000 バイトを超える値を設定しています。バッファ形式 2 の場合はデータ名 U に 32004 バイトを超える値を設定しています。
	MCF が終了処理中のため，メッセージの送受信を受け付けられません。
71003	メッセージキューが満杯です。
71004	メッセージを格納するバッファをメモリ上に確保できませんでした。

ステータスコード	意味
71108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('SENDRECV')を呼び出しています。
72001	データ名 P に設定した論理端末名称が間違っています。
	データ名 P に設定した出力先の論理端末名称は，MCF で定義していません。
	CBLDCMCF('SENDRECV')を呼び出せない論理端末を設定しています。
	＜問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用している（コネクション定義（mcftalccn -l）の replymsg オペランドに yes を指定）場合＞ 問い合わせ応答中または継続問い合わせ応答中のため受け付けられません。
72012	MCF バッファグループ定義のバッファ長が不足しました。
	MCF 通信プロセスは相手システムからメッセージを受信しましたが，UAP への応答連絡で RPC 通信の送信可能な上限値を超えました。
72013	データ名 G の指定値を超えるセグメントを受信しました。データ名 G の指定値を超えた部分は切り捨てられました。
	＜バッファ形式 2 の場合＞ 32763 バイトを超えるセグメントを受信しました。32763 バイトを超えた部分は切り捨てられました。
72016	データ名 N またはデータ名 T に設定した値が間違っています。
	データ名 M4 に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72026	データ名 H に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72036	データ名 G の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
72041	バッファ形式 1 の場合はデータ名 U に 0 バイト，またはマイナス値を設定しています。バッファ形式 2 の場合はデータ名 U に 0 から 4 バイト，またはマイナス値を設定しています。
72073	非同期メッセージを送信処理中です。
73001	データ名 M5 に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
	無通信監視時間（コネクション定義（mcftalccn -k）の notrftime オペランド指定値）が経過しましたが，相手システムからの通信がありません。
	相手システムからメッセージを受信しましたが，受信バッファを確保できませんでした。

ステータスコード	意味
73001	メッセージの送信処理中に相手システムからメッセージを受信したため，非同期受信処理を行いましたが，障害（UOC 障害など）が発生しました。
	出力先の論理端末で MCF 通信プロセスの内部障害が発生しました。
73002	コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指定）場合に，MHP でメッセージ受信後にコネクションが再確立されたため，送信を抑止しました。
73003	メッセージ受信が仕掛り中です。
73005	データ名 M5 に設定した時間が経過しましたが，論理端末からの応答がありません。
73010	入力または出力メッセージ編集 UOC で障害が発生しました。
	メッセージの読み込み時に障害が発生しました。
	メッセージの編集エラーが発生しました。
73015	出力先の論理端末は，ほかの UAP で仕掛り中です。
73018	データ名 M5 に設定した値が間違っています。
73019	メッセージ送信完了監視タイマのタイムアウトが発生しました。
73020	出力先の論理端末は停止しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('SENDSYNC') － 同期型メッセージの送信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1  一意名2  一意名3
```

DATA DIVISION の指定

```
01  一意名1.
    02  データ名A    PIC X(8)  VALUE ’SENDSYNC’.
    02  データ名B    PIC X(5).
    02  FILLER        PIC X(3).
    02  データ名C    PIC X(4)  VALUE SPACE.
    02  データ名D    PIC X(4)  VALUE SPACE.
    02  データ名E    PIC 9(8).
    02  データ名F    PIC 9(8).
    02  データ名G    PIC 9(9)  COMP VALUE ZERO.
    02  データ名H    PIC X(4).
    02  データ名I    PIC X(4)  VALUE SPACE.
    02  データ名J    PIC X(4)  VALUE SPACE.
    02  データ名K    PIC X(4)  VALUE SPACE.
    02  データ名L    PIC X(8)  VALUE SPACE.
    02  データ名M1   PIC X(4)  VALUE SPACE.
    02  データ名M2   PIC X(8)  VALUE SPACE.
    02  データ名M3   PIC X(4)  VALUE SPACE.
    02  データ名M4   PIC 9(9)  COMP VALUE ZERO.
    02  データ名M5   PIC 9(9)  COMP※.
    02  データ名M6   PIC X(1)  VALUE SPACE.
    02  データ名M7   PIC X(1).
    02  データ名N    PIC X(14) VALUE LOW-VALUE.
01  一意名2.
    02  データ名O    PIC X(4)  VALUE SPACE.
    02  データ名P    PIC X(8).
    02  データ名Q    PIC X(8)  VALUE SPACE.
    02  データ名R    PIC X(8)  VALUE SPACE.
    02  データ名T    PIC X(28) VALUE LOW-VALUE.
01  一意名3.
    02  データ名U    PIC 9(x)  COMP.
    02  データ名V    PIC X(x).
    02  データ名W    PIC X(n).
```

注※

負の値を設定する場合，「PIC S9(9) COMP」としてください。

機能

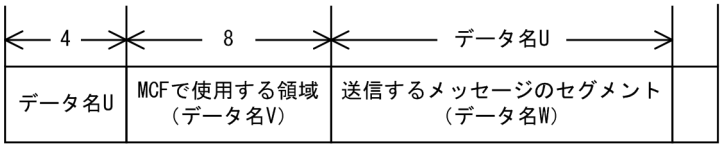
相手システムへ同期型でメッセージを送信します。メッセージは，一つのセグメントで構成されます。

送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

セグメントを送信する領域（一意名 3 で示す領域）の形式を次に示します。

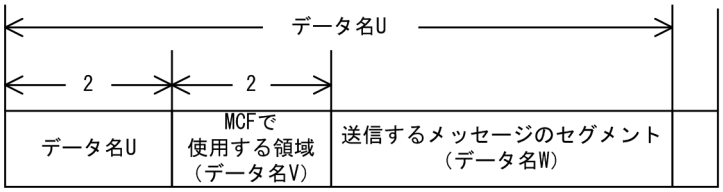
●バッファ形式 1 の場合

（単位：バイト）



●バッファ形式 2 の場合

（単位：バイト）



UAP で値を設定するデータ領域

●データ名 A

同期型のメッセージの送信を示す要求コード「VALUE 'SENDSYNC'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E，データ名 F

MCF で使用する領域です。

●データ名 G

0 を設定します。

●データ名 H

単一セグメントの送信を示す「VALUE 'EMI△'」を設定します。

●データ名 I，データ名 J，データ名 K，データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4

0 を設定します。

●データ名 M5

CBLDCMCF('SENDSYNC')を呼び出してから終了するまでの最大時間を設定します。0 を設定した場合，MCF マネジャ定義の UAP 共通定義で指定した同期型送信監視時間（mcfmuap -t sndtim）が仮定されます。

負の値を設定した場合は，時間を監視しません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，設定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，設定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O

空白を設定します。

●データ名 P

出力先の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 Q，データ名 R

空白を設定します。

●データ名 T

MCF で使用する領域です。

●データ名 U

【バッファ形式 1 の場合】 PIC 9(9)

送信するセグメントの長さを設定します。

【バッファ形式 2 の場合】 PIC 9(4)

送信するセグメントの長さ＋ 4 を設定します。

●データ名 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

●データ名 W

送信するセグメントの内容を設定します。一つのセグメントで 32000 バイトまで送信できます。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71002	バッファ形式 1 の場合はデータ名 U に 32000 バイトを超える値を設定しています。バッファ形式 2 の場合はデータ名 U に 32004 バイトを超える値を設定しています。
	MCF が終了処理中のため，メッセージの送信を受け付けられません。
71003	メッセージキューが満杯です。
71004	メッセージを格納するバッファをメモリ上に確保できませんでした。
71108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('SENDSYNC')を呼び出しています。
72001	データ名 P に設定した論理端末名称が間違っています。
	データ名 P に設定した出力先の論理端末名称は，MCF で定義していません。

ステータスコード	意味
72001	CBLDCMCF('SENDSYNC')を呼び出せない論理端末を設定しています。
	＜問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用している（コネクション定義（mcftalccn -l）の replymsg オペランドに yes を指定）場合＞ 問い合わせ応答中または継続問い合わせ応答中のため受け付けられません。
72016	データ名 M1 に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
	データ名 N またはデータ名 T に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72026	データ名 H に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72041	バッファ形式 1 の場合はデータ名 U に 0 バイト，またはマイナス値を設定しています。バッファ形式 2 の場合はデータ名 U に 0 から 4 バイト，またはマイナス値を設定しています。
72073	非同期メッセージを送信処理中です。
73001	データ名 M5 に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
	無通信監視時間（コネクション定義（mcftalccn -k）の notrftime オペランド指定値）が経過しましたが，相手システムからの通信がありません。
	相手システムからメッセージを受信しましたが，受信バッファを確保できませんでした。
	メッセージの送信処理中に相手システムからメッセージを受信したため，非同期受信処理を行いましたが，障害（UOC 障害など）が発生しました。
	出力先の論理端末で MCF 通信プロセスの内部障害が発生しました。
73002	コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指定）場合に，MHP でメッセージ受信後にコネクションが再確立されたため，送信を抑止しました。
73005	データ名 M5 に設定した時間が経過しましたが，論理端末からの応答がありません。
73010	出力メッセージ編集 UOC で障害が発生しました。
	メッセージの読み込み時に障害が発生しました。
73015	出力先の論理端末は，ほかの UAP で仕掛り中です。
73018	データ名 M5 に設定した値が間違っています。
73020	出力先の論理端末は停止しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('TACTCN△△') － コネクションの確立（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TACTCN ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D1  PIC X(1)   VALUE SPACE.
   02 データ名D2  PIC X(1).
   02 データ名D3  PIC X(26)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F1  PIC X(8).
   02 データ名F2  PIC X(56)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K1  PIC 9(9)   COMP.
   02 データ名K2  PIC X(4).
   02 データ名K3  PIC X(4)   VALUE LOW-VALUE.
   02 データ名K4  PIC X(4).
   02 データ名K5  PIC X(2).
   02 データ名K6  PIC X(2)   VALUE LOW-VALUE.
   02 データ名K7  PIC X(4).
   02 データ名K8  PIC X(2).
   02 データ名K9  PIC X(2)   VALUE LOW-VALUE.
   02 データ名K10 PIC X(128) VALUE LOW-VALUE.
   02 データ名K11 PIC X(128) VALUE LOW-VALUE.
   02 データ名K12 PIC X(744) VALUE LOW-VALUE.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.
```

機能

コネクションを確立します。

なお，CBLDCMCF('TACTCN△△')の正常終了は，コネクション確立要求を TP1/NET/TCP/IP が正常に受け付けたことを意味します。このため，相手システムとのコネクションの確立が正常に完了したことを示すものではありません。

CBLDCMCF('TACTCN△△')の呼び出し後にコネクションに関する何らかの処理をする場合は，CBLDCMCF('TLSCN△△△')を用いてコネクションの状態を確認してください。

UAP で値を設定するデータ領域

●データ名 A

コネクション確立を示す要求コード「VALUE 'TACTCN△△'」を設定します。

●データ名 C

確立するコネクションの指定方法を設定します。

'LE△△'

確立するコネクションを論理端末名称で指定するときに設定します。

'CN△△'

確立するコネクションをコネクション ID で指定するときに設定します。

空白

省略されたものとして，'LE△△'（論理端末名称指定）が仮定されます。

●データ名 D1

空白を設定します。

●データ名 D2

TP1/NET/TCP/IP 固有の機能を使用するかどうかを設定します。

'1'

TP1/NET/TCP/IP 固有の機能を使用するときに設定します。

'0'

TP1/NET/TCP/IP 固有の機能を使用しないときに設定します。

空白

省略されたものとして，'0'（使用しない）が仮定されます。

●データ名 D3

空白を設定します。

●データ名 E

処理対象のコネクションを持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は，0～239 です。

論理端末名称を使用してコネクションの確立を要求する場合は，無効となります。

0 を指定すると，該当するコネクション ID が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの命令文を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。

例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●データ名 F1

確立するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は 8 バイト以内で設定してください。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 F2，データ名 G，データ名 H，データ名 I，データ名 J

空白を設定します。

●データ名 K1

TP1/NET/TCP/IP 固有の機能を使用するときは 1024 を，使用しないときは 0 を設定してください。

●データ名 K2

プロトコル名称を設定します。

"TCP△"

TCP/IP プロトコル

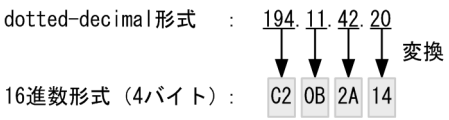
●データ名 K3，データ名 K6，データ名 K9，データ名 K10，データ名 K11，データ名 K12

0 を設定します。

●データ名 K4

自システムの IP アドレスを 16 進形式（ビッグエンディアン）で設定します。

＜IPアドレスの設定例＞



dotted-decimal 形式では指定できません。

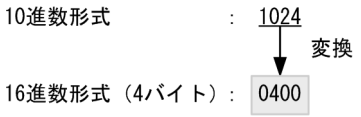
自システムの IP アドレスの割り当てを OS に任せる場合は，0 を設定してください。

IP アドレス 255.255.255.255 は設定できません。

●データ名 K5

自システムのポート番号を 16 進形式（ビッグエンディアン）で設定します。

＜ポート番号の設定例＞



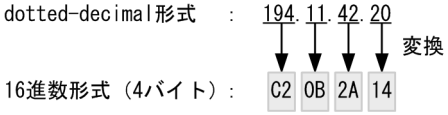
設定できる範囲は 0，または 1024～65535 です。

自システムのポート番号の割り当てを OS に任せる場合は，0 を設定してください。

●データ名 K7

相手システムの IP アドレスを 16 進形式（ビッグエンディアン）で設定します。

＜IPアドレスの設定例＞



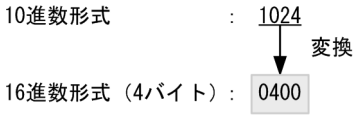
dotted-decimal 形式では指定できません。

IP アドレス 0.0.0.0，および，255.255.255.255 は設定できません。

●データ名 K8

相手システムのポート番号を 16 進形式（ビッグエンディアン）で設定します。

＜ポート番号の設定例＞



設定できる範囲は 1～65535 です。

●データ名 L

0 を設定します。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。

ステータスコード	意味
71001	MCF が開始処理中のため，CBLDCMCF('TACTCN△△')が受け付けられません。
71002	MCF が終了処理中のため，CBLDCMCF('TACTCN△△')が受け付けられません。
71004	CBLDCMCF('TACTCN△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71007	指定されたコネクション名は登録されていません。
71008	指定された論理端末名称は登録されていません。
71009	CBLDCMCF('TACTCN△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスにコネクションの確立を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
71011	コネクションが削除されているため，CBLDCMCF('TACTCN△△')が受け付けられません。
71014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループを指定しています。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K1 に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に'LE△△'，'CN△△'，または空白以外の値が設定されています。
72059	データ名 D1，またはデータ名 D3 に空白でない値が設定されています。
	データ名 D2 に 1，0，または空白以外の値が設定されています。
72061	データ名 E に 0 未満，または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。
73100	コネクション確立済みのため受け付けられません。
73101	コネクション確立処理中のため受け付けられません。
73102	コネクションに対する論理端末が接続されていないため受け付けられません。
73103	サーバ型のコネクションのため受け付けられません。

ステータスコード	意味
73104	コネクション解放処理中のため受け付けられません。
73130	データ名 K1 に 1 以上 1024 未満の値が設定されています。
73131	データ名 K2 に設定した値が間違っています。
73132	データ名 K3 に 0 でない値が設定されています。
73133	データ名 K4 に 255.255.255.255 の IP アドレスが設定されています。
73134	データ名 K5 に 1 以上 1023 以下の値が設定されています。
73135	データ名 K6 に 0 でない値が設定されています。
73136	データ名 K7 に 0.0.0.0，または 255.255.255.255 の IP アドレスが設定されています。
73137	データ名 K8 に 0 が設定されています。
73138	データ名 K9 に 0 でない値が設定されています。
73139	データ名 K10 に 0 でない値が設定されています。
73140	データ名 K11 に 0 でない値が設定されています。
73141	データ名 K12 に 0 でない値が設定されています。

CBLDCMCF('TACTLE△△') － 論理端末の閉塞解除（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TACTLE ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)   VALUE SPACE.
   02 データ名D1  PIC X(1)   VALUE SPACE.
   02 データ名D2  PIC X(1)   VALUE SPACE.
   02 データ名D3  PIC X(26)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F1  PIC X(8).
   02 データ名F2  PIC X(56)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.
```

機能

論理端末の閉塞を解除します。

なお，CBLDCMCF('TACTLE△△')の正常終了は，論理端末の閉塞解除要求を TP1/NET/TCP/IP が正常に受け付けたことを意味します。このため，論理端末の閉塞解除が正常に完了したことを示すものではありません。

CBLDCMCF('TACTLE△△')の呼び出し後に論理端末に関する何らかの処理をする場合は，CBLDCMCF('TLSLE△△△')を用いてコネクションの状態を確認してください。

UAP で値を設定するデータ領域

●データ名 A

論理端末の閉塞の解除を示す要求コード「VALUE 'TACTLE△△'」を設定します。

●データ名 C，データ名 D1，データ名 D2，データ名 D3

空白を設定します。

●データ名 E

処理対象の論理端末を持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

0 を指定すると，該当する論理端末名称が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの命令文を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●データ名 F1

閉塞解除する論理端末の名称を設定します。論理端末名称は 8 バイト以内で設定してください。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 F2，データ名 G，データ名 H，データ名 I，データ名 J

空白を設定します。

●データ名 K，データ名 L

0 を設定します。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	MCF が開始処理中のため，CBLDCMCF('TACTLE△△')が受け付けられません。
71002	MCF が終了処理中のため，CBLDCMCF('TACTLE△△')が受け付けられません。
71004	CBLDCMCF('TACTLE△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71008	指定された論理端末名称は登録されていません。
71009	CBLDCMCF('TACTLE△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスに論理端末の閉塞の解除を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。

ステータスコード	意味
71011	論理端末が削除されているため，CBLDCMCF('TACTLE△△')が受け付けられません。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D1，データ名 D2，またはデータ名 D3 に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。

CBLDCMCF('TDCTCN△△') － コネクションの解放（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TDCTCN ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D1  PIC X(1).
   02 データ名D2  PIC X(1)   VALUE SPACE.
   02 データ名D3  PIC X(26)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F1  PIC X(8).
   02 データ名F2  PIC X(56)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.
```

機能

コネクションを解放します。

なお，CBLDCMCF('TDCTCN△△')の正常終了は，コネクション解放要求を TP1/NET/TCP/IP が正常に受け付けたことを意味します。このため，相手システムとのコネクションの解放が正常に完了したことを示すものではありません。

CBLDCMCF('TDCTCN△△')の呼び出し後にコネクションに関する何らかの処理をする場合は，CBLDCMCF('TLSCN△△△')を用いてコネクションの状態を確認してください。

コネクション解放後，CCLSEVT または CERREVT を通知します。コネクション解放後に通知する MCF イベントを次の表に示します。

データ名 D1	コネクション定義（mcftalccn -f）の cnrelease オペランドの指定値	通知する MCF イベント
空白または'0'	任意	CCLSEVT
'1'	fin	CCLSEVT

データ名 D1	コネクション定義（mcftalccn -f）の cnrelease オペランドの指定値	通知する MCF イベント
'1'	rst	CERREVT

UAP で値を設定するデータ領域

●データ名 A

コネクション解放を示す要求コード「VALUE 'TDCTCN△△'」を設定します。

●データ名 C

解放するコネクションの指定方法を設定します。

'LE△△'

解放するコネクションを論理端末名称で指定するときに設定します。

'CN△△'

解放するコネクションをコネクション ID で指定するときに設定します。

空白

省略されたものとして，'LE△△'（論理端末名称指定）が仮定されます。

●データ名 D1

コネクションを強制的に解放するかどうかを設定します。

'1'

コネクションを強制的に解放します。

コネクション定義（mcftalccn -f）の cnrelease オペランドに rst を指定している場合，RST パケットを送信してコネクションを強制解放します。

コネクション定義（mcftalccn -f）の cnrelease オペランドを省略，または fin を指定している場合，1 を指定しても無効となり，FIN パケットを送信してコネクションを正常に解放します。

'0'

FIN パケットを送信してコネクションを正常に解放します。

空白

省略されたものとして，'0'（正常解放）が仮定されます。

●データ名 D2，データ名 D3

空白を設定します。

●データ名 E

処理対象のコネクションを持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

論理端末名称を使用してコネクションの解放を要求する場合は，無効となります。

0 を指定すると，該当するコネクション ID が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの命令文を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●データ名 F1

解放するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は 8 バイト以内で設定してください。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 F2，データ名 G，データ名 H，データ名 I，データ名 J

空白を設定します。

●データ名 K，データ名 L

0 を設定します。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	MCF が開始処理中のため，CBLDCMCF('TDCTCN△△')が受け付けられません。
71002	MCF が終了処理中のため，CBLDCMCF('TDCTCN△△')が受け付けられません。
71004	CBLDCMCF('TDCTCN△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71007	指定されたコネクション名は登録されていません。
71008	指定された論理端末名称は登録されていません。
71009	CBLDCMCF('TDCTCN△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスにコネクションの解放を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。

ステータスコード	意味
71011	コネクションが削除されているため，CBLDCMCF('TDCTCN△△')が受け付けられません。
71014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループ名を指定しています。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に'LE△△'，'CN△△'，または空白以外の値が設定されています。
72059	データ名 D1，データ名 D2，またはデータ名 D3 に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。
72075	データ名 D1 に 1，0，または空白以外の値が設定されています。

CBLDCMCF('TDCTLE△△') － 論理端末の閉塞（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TDCTLE  ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)   VALUE SPACE.
   02 データ名D1  PIC X(1)   VALUE SPACE.
   02 データ名D2  PIC X(1)   VALUE SPACE.
   02 データ名D3  PIC X(26)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F1  PIC X(8).
   02 データ名F2  PIC X(56)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.
```

機能

論理端末を閉塞します。

なお，CBLDCMCF('TDCTLE△△')の正常終了は，論理端末の閉塞要求を TP1/NET/TCP/IP が正常に受け付けたことを意味します。このため，論理端末の閉塞が正常に完了したことを示すものではありません。

CBLDCMCF('TDCTLE△△')の呼び出し後に論理端末に関する何らかの処理をする場合は，CBLDCMCF('TLSLE△△△')を用いてコネクションの状態を確認してください。

UAP で値を設定するデータ領域

●データ名 A

論理端末の閉塞を示す要求コード「VALUE 'TDCTLE△△'」を設定します。

●データ名 C，データ名 D1，データ名 D2，データ名 D3

空白を設定します。

●データ名 E

処理対象の論理端末を持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

0 を指定すると，該当する論理端末名称が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの命令文を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●データ名 F1

閉塞する論理端末の名称を設定します。論理端末名称は 8 バイト以内で設定してください。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 F2，データ名 G，データ名 H，データ名 I，データ名 J

空白を設定します。

●データ名 K，データ名 L

0 を設定します。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	MCF が開始処理中のため，CBLDCMCF('TDCTLE△△')が受け付けられません。
71002	MCF が終了処理中のため，CBLDCMCF('TDCTLE△△')が受け付けられません。
71004	CBLDCMCF('TDCTLE△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71008	指定された論理端末名称は登録されていません。
71009	CBLDCMCF('TDCTLE△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスに論理端末の閉塞を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。

ステータスコード	意味
71011	論理端末が削除されているため，CBLDCMCF('TDCTLE△△')が受け付けられません。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D2，またはデータ名 D3 に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。
72075	データ名 D1 に空白以外の値が設定されています。

CBLDCMCF('TEMPGET△') － 一時記憶データの受け取り（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A  PIC X(8)  VALUE ’TEMPGET ’.
   02 データ名B  PIC X(5).
   02 FILLER     PIC X(3).
   02 データ名C  PIC X(4)  VALUE SPACE.
   02 データ名D  PIC X(4)  VALUE SPACE.
   02 データ名E  PIC 9(8).
   02 データ名F  PIC 9(8).
   02 データ名G  PIC 9(9)  COMP.
   02 データ名H  PIC X(4)  VALUE SPACE.
   02 データ名I  PIC X(4)  VALUE SPACE.
   02 データ名J  PIC X(4)  VALUE SPACE.
   02 データ名K  PIC X(4)  VALUE SPACE.
   02 データ名L  PIC X(8)  VALUE SPACE.
   02 データ名M1 PIC X(4)  VALUE SPACE.
   02 データ名M2 PIC X(8)  VALUE SPACE.
   02 データ名M3 PIC X(4)  VALUE SPACE.
   02 データ名M4 PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5 PIC 9(9)  COMP VALUE ZERO.
   02 データ名M6 PIC X(1)  VALUE SPACE.
   02 データ名M7 PIC X(1)  VALUE ’2’.
   02 データ名N  PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O  PIC X(4)  VALUE SPACE.
   02 データ名P  PIC X(8)  VALUE SPACE.
   02 データ名Q  PIC X(8)  VALUE SPACE.
   02 データ名R  PIC X(8)  VALUE SPACE.
   02 データ名S  PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名T  PIC 9(4)  COMP.
   02 データ名U  PIC X(4).
   02 データ名V  PIC X(n).
```

機能

継続問い合わせ応答用一時記憶領域に格納されている一時記憶データを受け取ります。

データ名 G の長さ（7～32006 バイト）を超える一時記憶データがある場合，超えた分については切り捨てます。

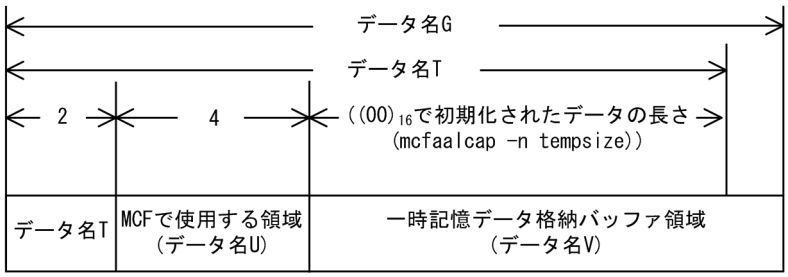
データ名 G から 6 を減算した値と比べて一時記憶データ長の方が短い場合，データ名 V に一時記憶データを設定します。データ名 V の残りの領域については何も設定しません。

受け取り要求実行時，初期状態（継続問い合わせ応答開始後，CBLDCMCF('TEMPPUT△')を 1 回も実行していない状態）の場合，アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドで指定した長さの$(00)_{16}$の一時記憶データがあるものとして CBLDCMCF('TEMPGET△')を実行します。

受け取り領域（一意名 3 で示す領域）の形式を次に示します。

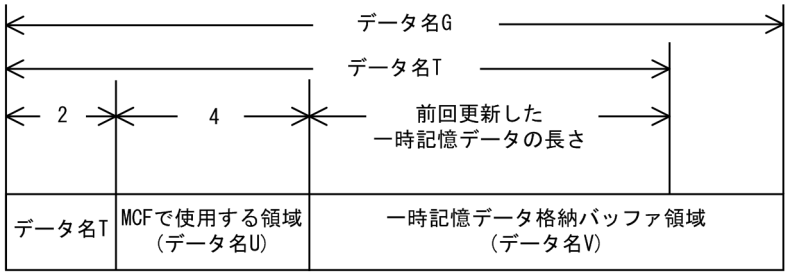
●CBLDCMCF('TEMPPUT△')未実行（継続問い合わせ応答開始後，CBLDCMCF('TEMPPUT△')を実行していない（初期状態））

（単位：バイト）



●CBLDCMCF('TEMPPUT△')実行済み（継続問い合わせ応答開始後，CBLDCMCF('TEMPPUT△')を1回以上実行）

（単位：バイト）



UAP で値を設定するデータ領域

●データ名 A

一時記憶データの受け取りを示す要求コード「VALUE 'TEMPGET△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E，データ名 F

MCF で使用する領域です。

●データ名 G

一時記憶データを受け取る領域の長さを 7～32006 バイトで設定します。

●データ名 H，データ名 I，データ名 J，データ名 K，データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファの形式を「VALUE '2'」と設定します。

●データ名 N

MCF で使用する領域です。

●データ名 O，データ名 P，データ名 Q，データ名 R

空白を設定します。

●データ名 S，データ名 U

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 T

前回更新した一時記憶データの長さ＋ 6 が返されます。初期状態の場合，継続問い合わせ応答用一時記憶領域の長さ（アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドの指定値）＋ 6 が返されます。

●データ名 V

受け取った一時記憶データが返されます。初期状態の場合，継続問い合わせ応答用一時記憶領域の長さ（アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドの指定値）分だけ$(00)_{16}$が埋められます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
72000	SPP では CBLDCMCF('TEMPGET△')を呼び出せません。
72013	一時記憶データ格納バッファ領域の長さ（データ名 G から 6 を減算した値）を超える一時記憶データを受け取りました。一時記憶データ格納バッファ領域の長さを超える一時記憶データを切り捨てました。

ステータスコード	意味
72016	データ名 N またはデータ名 S に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72036	データ名 G の設定値が不足しています。7 バイト以上の領域を確保してください。
72101	継続問い合わせ応答型でないアプリケーションで，CBLDCMCF('TEMPGET△')を呼び出しました。
72106	先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('TEMPGET△')を呼び出しました。
72107	CBLDCMCF('CONTEND△')を呼び出したあとで，CBLDCMCF('TEMPGET△')を呼び出しました。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('TEMPPUT△') － 一時記憶データの更新（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE ’TEMPPUT ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)  VALUE SPACE.
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP VALUE ZERO.
   02 データ名H   PIC X(4)  VALUE SPACE.
   02 データ名I   PIC X(4)  VALUE SPACE.
   02 データ名J   PIC X(4)  VALUE SPACE.
   02 データ名K   PIC X(4)  VALUE SPACE.
   02 データ名L   PIC X(8)  VALUE SPACE.
   02 データ名M1  PIC X(4)  VALUE SPACE.
   02 データ名M2  PIC X(8)  VALUE SPACE.
   02 データ名M3  PIC X(4)  VALUE SPACE.
   02 データ名M4  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M6  PIC X(1)  VALUE SPACE.
   02 データ名M7  PIC X(1)  VALUE ’2’.
   02 データ名N   PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O   PIC X(4)  VALUE SPACE.
   02 データ名P   PIC X(8)  VALUE SPACE.
   02 データ名Q   PIC X(8)  VALUE SPACE.
   02 データ名R   PIC X(8)  VALUE SPACE.
   02 データ名S   PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名T   PIC 9(4)  COMP.
   02 データ名U   PIC X(4).
   02 データ名V   PIC X(n).
```

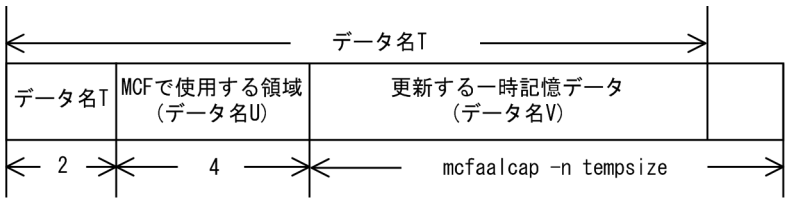
機能

継続問い合わせ応答用一時記憶領域に格納されている一時記憶データを更新します。

アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドには，更新する一時記憶データ長（データ名 T から 6 を減算した値）以上の値を指定してください。

更新する領域（一意名 3 で示す領域）の形式を次に示します。

（単位：バイト）



UAP で値を設定するデータ領域

●データ名 A

一時記憶データの更新を示す要求コード「VALUE 'TEMPPUT△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E，データ名 F

MCF で使用する領域です。

●データ名 G

0 を設定します。

●データ名 H，データ名 I，データ名 J，データ名 K，データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファの形式を「VALUE '2'」と設定します。

●データ名 N

MCF で使用する領域です。

●データ名 O，データ名 P，データ名 Q，データ名 R

空白を設定します。

●データ名 S

MCF で使用する領域です。

●データ名 T

一時記憶データの更新データ長を設定します。

●データ名 U

MCF で使用する領域です。

●データ名 V

一時記憶データが格納されている領域を設定します。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71103	一時記憶データを更新するための領域をメモリ上に確保できませんでした。
72000	SPP では CBLDCMCF('TEMPPUT△')を呼び出せません。
72016	データ名 N またはデータ名 S に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72035	データ名 T に設定した更新データの長さが，アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドで定義した長さを超えています。
	データ名 T に 0 から 6 バイト，またはマイナス値を設定しています。
72101	継続問い合わせ応答型でないアプリケーションで，CBLDCMCF('TEMPPUT△')を呼び出しました。
72105	CBLDCMCF('TEMPGET△')を呼び出す前に，CBLDCMCF('TEMPPUT△')を呼び出しました。
72106	先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('TEMPPUT△')を呼び出しました。
72107	CBLDCMCF('CONTEND△')を呼び出したあとで，CBLDCMCF('TEMPPUT△')を呼び出しました。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('TLSCN△△△') － コネクションの状態取得（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TLSCN   ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D   PIC X(28)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F1  PIC X(8).
   02 データ名F2  PIC X(56)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.

01 一意名3.
   02 データ名M   PIC 9(9)   COMP.
   02 一意名4.
      03 データ名N   PIC X(8).
      03 データ名O   PIC X(4).
      03 データ名P   PIC X(4).
      03 データ名Q   PIC X(40)   VALUE LOW-VALUE.
```

機能

コネクションの状態を取得します。

UAP で値を設定するデータ領域

●データ名 A

コネクション状態取得を示す要求コード「VALUE 'TLSCN△△△'」を設定します。

●データ名 C

状態を取得するコネクションの指定方法を設定します。

'LE△△'

状態を取得するコネクションを論理端末名称で指定するときに設定します。

'CN△△'

状態を取得するコネクションをコネクション ID で指定するときに設定します。

空白

省略されたものとして，'LE△△'（論理端末名称指定）が仮定されます。

●データ名 D

空白を設定します。

●データ名 E

処理対象のコネクションを持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

論理端末名称を使用してコネクションの状態取得を要求する場合は，無効となります。

0 を指定すると，該当するコネクション ID が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの命令文を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。

例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●データ名 F1

状態を取得するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は 8 バイト以内で設定してください。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 F2，データ名 G，データ名 H，データ名 I，データ名 J

空白を設定します。

●データ名 K，データ名 L

0 を設定します。

●データ名 M

一意名 4 から一意名 n の数（データ名 N，データ名 O，データ名 P，とデータ名 Q の組の数）として，1 を設定します。

処理終了後は，該当するコネクションの個数が返されます。

●データ名 Q

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 M

この命令文の対象となったコネクションの個数が返されます。

●データ名 N

要求したコネクションのコネクション ID が設定されます。

●データ名 O

要求したコネクションのプロトコル種別が設定されます。

'TCP△'
　TCP/IP プロトコル

●データ名 P

要求したコネクションの状態として，次の値が設定されます。

'ACT '
　確立状態

'ACTB'
　確立処理中状態

'DCT '
　解放状態

'DCTB'
　解放処理中状態

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	MCF が開始処理中のため，CBLDCMCF('TLSCN△△△')が受け付けられません。
71004	CBLDCMCF('TLSCN△△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71007	指定されたコネクション名は登録されていません。
71008	指定された論理端末名称は登録されていません。

ステータスコード	意味
71009	CBLDCMCF('TLSCN△△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスにコネクションの状態取得を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
71011	コネクションが削除されているため，CBLDCMCF('TLSCN△△△')が受け付けられません。
71014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループ名を指定しています。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に'LE△△'，'CN△△'，または空白以外の値が設定されています。
72059	データ名 D に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。
72076	データ名 M に 1 でない値が設定されています。

CBLDCMCF('TLSLE△△△') － 論理端末の状態取得（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TLSLE   ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)   VALUE SPACE.
   02 データ名D   PIC X(28)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F1  PIC X(8).
   02 データ名F2  PIC X(56)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.

01 一意名3.
   02 データ名M   PIC 9(9)   COMP.
   02 一意名4.
      03 データ名N   PIC X(8).
      03 データ名O   PIC X(4)  VALUE LOW-VALUE.
      03 データ名P   PIC X(4).
      03 データ名Q   PIC X(40) VALUE LOW-VALUE.
```

機能

論理端末の状態を取得します。

UAP で値を設定するデータ領域

●データ名 A

論理端末の状態取得を示す要求コード「VALUE 'TLSLE△△△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E

処理対象の論理端末を持つ MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 0～239 です。

0 を指定すると，該当する論理端末名称が属する MCF 通信サービスを検索します。MCF 通信サービスが多い構成や UAP からこの命令文を多数発行する場合は，MCF 通信プロセス識別子の指定をお勧めします。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●データ名 F1

状態を取得する論理端末の名称を設定します。論理端末名称は 8 バイト以内で設定してください。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●データ名 F2，データ名 G，データ名 H，データ名 I，データ名 J

空白を設定します。

●データ名 K，データ名 L

0 を設定します。

●データ名 M

一意名 4 から一意名 n の数（データ名 N，データ名 O，データ名 P，とデータ名 Q の組の数）として，1 を設定します。

処理終了後は，該当する論理端末の個数が返されます。

●データ名 O，データ名 Q

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 M

この命令文の対象となった論理端末の個数が返されます。

●データ名 N

要求した論理端末の名称が設定されます。

●データ名 P

要求した論理端末の状態として，次の値が設定されます。

'ACT△'

閉塞解除状態

'DCT△'

閉塞状態

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	MCF が開始処理中のため，CBLDCMCF('TLSLE△△△')が受け付けられません。
71004	CBLDCMCF('TLSLE△△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71008	指定された論理端末名称は登録されていません。
71009	CBLDCMCF('TLSLE△△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスに論理端末の状態取得を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
71011	論理端末が削除されているため，CBLDCMCF('TLSLE△△△')が受け付けられません。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。
72076	データ名 M に 1 でない値が設定されています。

CBLDCMCF('TLSLN△△△') － サーバ型コネクションの確立要求の受付状態取得（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TLSLN   ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)   VALUE SPACE.
   02 データ名D   PIC X(28)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F   PIC X(64)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.

01 一意名3.
   02 データ名M   PIC 9(9)   COMP VALUE 1.
   02 一意名4.
      03 データ名N   PIC X(4).
      03 データ名O   PIC X(60) VALUE LOW-VALUE.
```

機能

サーバ型コネクションの確立要求の受付状態を取得します。

UAP で値を設定するデータ領域

●データ名 A

コネクションの確立要求の受付状態取得を示す要求コード「VALUE 'TLSLN△△△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E

処理対象の MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 1～239 です。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。

例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●データ名 F，データ名 G，データ名 H，データ名 I，データ名 J

空白を設定します。

●データ名 K，データ名 L

0 を設定します。

●データ名 M

1 を設定します。

●データ名 O

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 N

サーバ型コネクションの確立要求の受付状態として，次の値が設定されます。

'LSTN'

受付開始状態

'RTRY'

受付開始処理中状態

'ON_W'

受付開始要求待ち状態

'INIT'

受付終了状態

それぞれの状態のときに使用できる命令文を，次の表に示します。

データ名 N の設定値	使用できる COBOL-UAP 作成用プログラム	
	CBLDCMCF('TONLN△△△')	CBLDCMCF('TOFLN△△△')
LSTN	×	○
RTRY	×	○
ON_W	○	○
INIT	○	×

（凡例）

○：使用できます。

×：使用できません。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	MCF が開始処理中のため，CBLDCMCF('TLSLN△△△')が受け付けられません。
71002	MCF が終了処理中のため，CBLDCMCF('TLSLN△△△')が受け付けられません。
71004	CBLDCMCF('TLSLN△△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71009	CBLDCMCF('TLSLN△△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスにサーバ型コネクションの確立要求の受付状態取得を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D に空白でない値が設定されています。
72061	データ名 E に 0 以下または 240 以上の値が設定されています。
72065	データ名 F に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72076	データ名 M に 1 でない値が設定されています。

CBLDCMCF('TOFLN△△△') － サーバ型コネクションの確立要求の受付終了（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TOFLN   ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)   VALUE SPACE.
   02 データ名D   PIC X(28)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F   PIC X(64)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.
```

機能

サーバ型コネクションの確立要求の受付を終了します。

UAP で値を設定するデータ領域

●データ名 A

コネクションの確立要求の受付終了を示す要求コード「VALUE 'TOFLN△△△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E

処理対象の MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 1～239 です。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●データ名 F，データ名 G，データ名 H，データ名 I，データ名 J

空白を設定します。

●データ名 K，データ名 L

0 を設定します。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	MCF が開始処理中のため，CBLDCMCF('TOFLN△△△')が受け付けられません。
71002	MCF が終了処理中のため，CBLDCMCF('TOFLN△△△')が受け付けられません。
71004	CBLDCMCF('TOFLN△△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71009	CBLDCMCF('TOFLN△△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスにサーバ型コネクションの確立要求の受付終了を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D に空白でない値が設定されています。
72061	データ名 E に 0 以下または 240 以上の値が設定されています。
72065	データ名 F に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。

CBLDCMCF('TONLN△△△') － サーバ型コネクションの確立要求の受付開始（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TONLN   ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4)   VALUE SPACE.
   02 データ名D   PIC X(28)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F   PIC X(64)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.
```

機能

サーバ型コネクションの確立要求の受付を開始します。

UAP で値を設定するデータ領域

●データ名 A

コネクションの確立要求の受付開始を示す要求コード「VALUE 'TONLN△△△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E

処理対象の MCF 通信サービスの MCF 通信プロセス識別子※を設定します。設定できる範囲は 1～239 です。

注※

MCF 環境定義（mcftenv -s）で指定する MCF 通信プロセス識別子は 16 進数と見なしてください。例えば，MCF 通信プロセス識別子が 10 の場合，16 を設定してください。

●データ名 F，データ名 G，データ名 H，データ名 I，データ名 J

空白を設定します。

●データ名 K，データ名 L

0 を設定します。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	MCF が開始処理中のため，CBLDCMCF('TONLN△△△')が受け付けられません。
71002	MCF が終了処理中のため，CBLDCMCF('TONLN△△△')が受け付けられません。
71004	CBLDCMCF('TONLN△△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71009	CBLDCMCF('TONLN△△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスにサーバ型コネクションの確立要求の受付開始を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D に空白でない値が設定されています。
72061	データ名 E に 0 以下または 240 以上の値が設定されています。
72065	データ名 F に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。

DISABLE － 継続問い合わせ応答の終了（データ操作言語）

形式

DATA DIVISION（通信記述項）の指定

```
CD  通信記述名
    FOR  I-O  STORAGE
   〔STATUS  KEY  IS  データ名1〕.
```

PROCEDURE DIVISION（通信文）の指定

```
DISABLE  通信記述名.
```

機能

次に示す CALL インタフェースの機能を実現します。

- 継続問い合わせ応答の終了　CBLDCMCF('CONTEND△')

UAP で値を設定する項目

●FOR 句

継続問い合わせ応答の終了を示す I-O STORAGE を設定します。

OpenTP1 から値が返される項目

●STATUS KEY 句

ステータスコードを受け取りたい場合に設定します。省略した場合は，ステータスコードを受け取れません。データ名 1 にステータスコードが返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
72000	＜ MHP の実行でリターンした場合＞ 先頭セグメントを受信する RECEIVE 文を呼び出す前に，DISABLE 文を呼び出しました。
	＜ SPP の実行でリターンした場合＞ SPP では DISABLE 文を呼び出せません。
72042	通信記述項に SYMBOLIC TERMINAL 句の設定があり，空白以外が設定されています。
72101	継続問い合わせ応答型でないアプリケーションで，DISABLE 文を呼び出しました。
72107	DISABLE 文を 2 回以上呼び出しました。

ステータスコード	意味
72111	次起動アプリケーションを設定して SEND（応答メッセージの送信）文を呼び出したあと，DISABLE 文を呼び出しました。
	継続問い合わせ応答型のアプリケーション名を設定して SEND（アプリケーションプログラムの起動）文を呼び出したあと，DISABLE 文を呼び出しました。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

RECEIVE － メッセージの受信（データ操作言語）

形式

DATA DIVISION（通信記述項）の指定

```
CD  通信記述名
    FOR   {INPUT|I-O}
    〔STATUS  KEY  IS  データ名1〕
    〔SYMBOLIC  TERMINAL  IS  データ名2〕
    〔MESSAGE  DATE  IS  データ名3〕
    〔MESSAGE  TIME  IS  データ名4〕
    〔SYNCHRONOUS  MODE  IS  {SYNC|ASYNC|データ名6}〕
    〔WAITING  TIME  IS  データ名11〕.
```

DATA DIVISION（データ記述項）の指定

```
01  一意名1.
    02  データ名A   PIC 9(4)  COMP.
    02  データ名B   PIC X(2).
    02  データ名C   PIC X(n).
```

PROCEDURE DIVISION（通信文）の指定

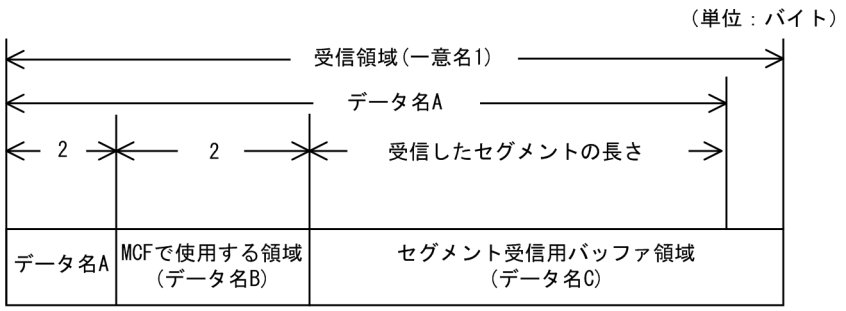
```
RECEIVE  通信記述名
    〔FIRST〕  SEGMENT
       INTO  一意名1.
```

機能

次に示す CALL インタフェースの機能を実現します。ただし，受信できるメッセージの一つのセグメントの最大長は，32763 バイトです。

- メッセージの受信　CBLDCMCF('RECEIVE△')
- 同期型メッセージの受信　CBLDCMCF('RECVSYNC')

セグメントを受信する領域（一意名 1 で示す領域）の形式を次に示します。



UAP で値を設定する項目

●FOR 句

次のどちらかの値を設定します。

INPUT

一方送信メッセージの受信

I-O

問い合わせメッセージまたは同期型メッセージの受信

●SYMBOLIC TERMINAL 句

非同期型のメッセージ（中間セグメントまたは最終セグメント）の受信，または，同期型のメッセージの受信の場合，論理端末名称を設定したデータ項目を設定します。データ名 2 にメッセージ入力元の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

非同期型のメッセージ（中間セグメントまたは最終セグメント）の受信の場合，先頭セグメントの受信時に返された論理端末名称を設定してください。

非同期型のメッセージ（先頭セグメント）の受信処理終了後，SYMBOLIC TERMINAL 句には OpenTP1 から値が返されます。

●SYNCHRONOUS MODE 句

非同期型でメッセージを受信するか，同期型でメッセージを受信するかを設定します。

SYNC

同期型のメッセージの受信

ASYNC

非同期型のメッセージの受信

データ名 6

次の値を設定したデータ項目

'0'または'△'：非同期型のメッセージの受信

'1'：同期型のメッセージの受信

省略した場合は，ASYNC（非同期型のメッセージの受信）が設定されます。

●WAITING TIME 句

RECEIVE 文を実行してから終了するまでの，最大時間を設定したデータ項目を設定します。同期型のメッセージの受信の場合に設定します。

データ名 11

監視時間を HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で設定したデータ項目を設定します。

省略した場合，またはデータ名 11 に'00000000'を設定した場合は，MCF マネジャ定義の UAP 共通定義で設定した同期型受信監視時間（mcfmuap -t recvtim）が設定されます。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，設定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，設定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3 秒以上の値の設定を推奨します。

●データ名 B

MCF で使用する領域です。

●FIRST

先頭セグメントを受信する場合に設定します。

OpenTP1 から値が返される項目

●STATUS KEY 句

ステータスコードを受け取りたい場合に設定します。省略した場合は，ステータスコードを受け取れません。データ名 1 にステータスコードが返されます。

●SYMBOLIC TERMINAL 句

非同期型のメッセージ（先頭セグメント）の受信の場合，入力元の論理端末名称を受け取りたいときに設定します。省略した場合は，論理端末名称を受け取れません。データ名 2 にメッセージ入力元の論理端末名称が返されます。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない場合，論理端末名称の後ろが空白で埋められます。

中間セグメントまたは最終セグメントを受信する場合は，ここで返された論理端末名称を SYMBOLIC TERMINAL 句に設定します。

●MESSAGE DATE 句

メッセージを受信した日付を受け取りたい場合に設定します。省略した場合は，メッセージを受信した日付を受け取れません。データ名 3 にメッセージを受信した日付が YYMMDD（YY：西暦下 2 けた　MM：月　DD：日）の形式で返されます。

●MESSAGE TIME 句

メッセージを受信した時刻を受け取りたい場合に設定します。省略した場合は，メッセージを受信した時刻を受け取れません。データ名４にメッセージを受信した時刻が HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で返されます。

●データ名 A

受信したセグメントの長さ＋４が返されます。

●データ名 C

受信したセグメントの内容が返されます。一つのセグメントで 32763 バイトまで受信できます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71000	先頭セグメントを受信する RECEIVE 文を 2 回以上実行しています。中間セグメントまたは最終セグメントを受信する場合は，FIRST を設定しないで RECEIVE 文を実行してください。
71001	メッセージの最終セグメントを受信したあとで，次のセグメントを受信する RECEIVE 文を実行しています。直前に実行した RECEIVE 文でメッセージはすべて受信しました。 このステータスコードが返されたあとに，再び RECEIVE 文を実行した場合は，ステータスコード 72000 が返されます。
71002	メッセージキューからの入力処理中に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	MCF が終了処理中のため，メッセージの受信を受け付けられません。
71108	メッセージ受信に必要な管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する RECEIVE 文を実行する前に，中間セグメントまたは最終セグメントを受信する RECEIVE 文を実行しています。先頭セグメントを受信する場合は，FIRST を設定して RECEIVE 文を実行してください。 • ステータスコード 71001 が返されたあとで，RECEIVE 文を実行しています。
	＜ SPP の実行でリターンした場合＞ SPP では RECEIVE 文を実行できません。
72001	SYMBOLIC TERMINAL 句に設定した論理端末名称が間違っています。
	SYMBOLIC TERMINAL 句に設定した入力元の論理端末名称は，MCF で定義していません。
	RECEIVE 文を実行できない論理端末を設定しています。

ステータスコード	意味
72001	＜問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用している（コネクション定義（mcftalccn -l）の replymsg オペランドに yes を指定）場合＞ 問い合わせ応答中または継続問い合わせ応答中のため受け付けられません。
72012	MCF バッファグループ定義のバッファ長が不足しました。
	MCF 通信プロセスは相手システムからメッセージを受信しましたが，UAP への応答連絡で RPC 通信の送信可能な上限値を超えました。
72013	データ名 C のサイズを超えるセグメントを受信しました。データ名 C のサイズを超えた部分は切り捨てられました。
	32763 バイトを超えるセグメントを受信しました。32763 バイトを超えた部分は切り捨てられました。
72016	WAITING TIME 句に設定した値が間違っています。
72020	SYNCHRONOUS MODE 句に設定した値が間違っています。
72024	FOR 句に設定した値が間違っています。
72036	データ名 C のサイズが不足しています。5 バイト以上の領域を確保してください。
73001	データ名 11 に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
	前回の通信から無通信監視時間（コネクション定義（mcftalccn -k）の notrftime オペランド指定値）が経過しましたが，相手システムからの通信がありません。
	相手システムからメッセージを受信しましたが，受信バッファを確保できませんでした。
	入力元の論理端末で MCF 通信プロセスの内部障害が発生しました。
73002	＜同期型メッセージの受信を行う（FOR 句に I-O を設定，かつ，SYNCHRONOUS MODE 句に SYNC を設定または 1 を設定したデータ名 6 を設定）場合＞ コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指定）場合に，MHP でメッセージ受信後にコネクションが再確立されたため，受信を取り消しました。
73005	データ名 11 に設定した時間が経過しましたが，論理端末からの応答がありません。
73010	入力メッセージ編集 UOC で障害が発生しました。
	メッセージの読み込み時に障害が発生しました。
73015	設定した論理端末は，ほかの UAP で仕掛り中です。
73018	データ名 11 に設定した監視時間が間違っています。
73020	入力元の論理端末は停止しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

RECEIVE － 一時記憶データの受け取り（データ操作言語）

形式

DATA DIVISION（通信記述項）の指定

```
CD  通信記述名
    FOR   {INPUT|I-O}   STORAGE
   〔STATUS  KEY  IS  データ名1〕.
```

DATA DIVISION（データ記述項）の指定

```
01  一意名1.
    02  データ名A   PIC 9(4)  COMP.
    02  データ名B   PIC X(4).
    02  データ名C   PIC X(n).
```

PROCEDURE DIVISION（通信文）の指定

```
RECEIVE  通信記述名    {MESSAGE|SEGMENT}
   INTO  一意名1.
```

機能

次に示す CALL インタフェースの機能を実現します。

- 一時記憶データの受け取り　CBLDCMCF('TEMPGET△')

一時記憶データ格納バッファ領域（データ名 C）は，1～32000 バイトの長さを確保してください。

データ名 C の長さを超える一時記憶データがある場合，超えた分については切り捨てます。

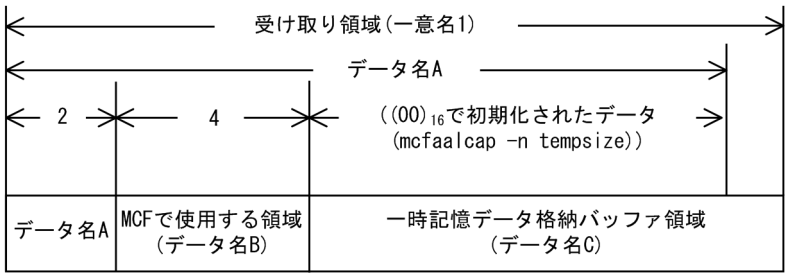
データ名 C の長さと比べて一時記憶データの長さの方が短い場合，データ名 C に一時記憶データを設定します。残りの領域については何も設定しません。

受け取り要求実行時，初期状態（継続問い合わせ応答開始後，SEND（一時記憶データの更新）文を 1 回も実行していない状態）の場合，アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドで指定した長さの$(00)_{16}$の一時記憶データがあるものとして実行します。

受け取り後の一意名 1 の形式を次に示します。

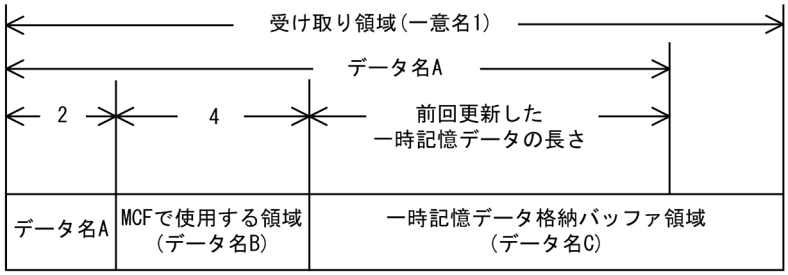
●SEND（一時記憶データの更新）文未実行（継続問い合わせ応答開始後，SEND（一時記憶データの更新）文を実行していない（初期状態））

（単位：バイト）



●SEND（一時記憶データの更新）文実行済み（継続問い合わせ応答開始後，SEND（一時記憶データの更新）文を1回以上実行）

（単位：バイト）



UAP で値を設定する項目

●FOR 句

一時記憶データの受け取りであることを示す，INPUT STORAGE または I-O STORAGE を設定します。

●データ名 B

MCF で使用する領域です。

●MESSAGE，SEGMENT

どちらかを設定します。

OpenTP1 から値が返される項目

●STATUS KEY 句

ステータスコードを受け取りたい場合に設定します。省略した場合は，ステータスコードを受け取れません。データ名 1 にステータスコードが返されます。

●データ名 A

前回更新した一時記憶データの長さ＋ 6 が返されます。初期状態の場合，継続問い合わせ応答用一時記憶領域の長さ（アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドの指定値）＋ 6 が返されます。

●データ名 C

受け取った一時記憶データが返されます。初期状態の場合，継続問い合わせ応答用一時記憶領域の長さ（アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドの指定値）分だけ$(00)_{16}$が埋められます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
72000	SPP では RECEIVE（一時記憶データの受け取り）文を呼び出せません。
72013	一時記憶データ格納バッファ領域の長さ（データ名 C の長さ）を超える一時記憶データを受け取りました。一時記憶データ格納バッファ領域の長さを超える一時記憶データを切り捨てました。
72016	通信文に BEFORE ERASING が設定されています。
72036	受け取り領域の長さ（一意名 1 の長さ）が不足しています。7 バイト以上の領域を確保してください。
72042	通信記述項に SYMBOLIC TERMINAL 句の設定があり，空白以外が設定されています。
72101	継続問い合わせ応答型でないアプリケーションで，RECEIVE（一時記憶データの受け取り）文を呼び出しました。
72106	先頭セグメントを受信する RECEIVE 文を呼び出す前に，RECEIVE（一時記憶データの受け取り）文を呼び出しました。
72107	DISABLE 文を呼び出したあとで，RECEIVE（一時記憶データの受け取り）文を呼び出しました。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

SEND － メッセージの送信（データ操作言語）

形式

DATA DIVISION（通信記述項）の指定

```
CD  通信記述名
    FOR   {OUTPUT｜I-O}
   〔STATUS  KEY  IS  データ名1〕
   〔SYMBOLIC  TERMINAL  IS  データ名2〕
   〔SYNCHRONOUS  MODE  IS   {SYNC｜ASYNC｜データ名6} 〕
   〔SWITCHING  MODE  IS   {NORMAL｜PRIOR｜データ名7} 〕
   〔NEXT  TRANSACTION  IS データ名8〕
   〔DETAIL  MODE  IS  データ名10〕
   〔WAITING  TIME  IS  データ名11〕.
```

DATA DIVISION（データ記述項）の指定

```
01  一意名1.
    02  データ名A   PIC 9(4)  COMP.
    02  データ名B   PIC X(2).
    02  データ名C   PIC X(n).
01  一意名3.
    02  データ名D   PIC 9(4)  COMP.
    02  データ名E   PIC X(2).
    02  データ名F   PIC X(n).
```

PROCEDURE DIVISION（通信文）の指定

```
SEND  通信記述名  FROM  一意名1
     〔WITH   {EMI｜一意名2} 〕
     〔BEFORE  RECEIVING  MESSAGE  INTO  一意名3〕.
```

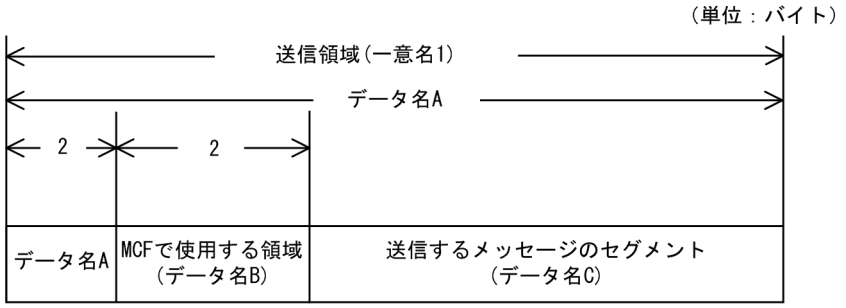
機能

次に示す CALL インタフェースの機能を実現します。

- 応答メッセージの送信　CBLDCMCF('REPLY△△△')
- 一方送信メッセージの送信　CBLDCMCF('SEND△△△△')
- 同期型メッセージの送受信　CBLDCMCF('SENDRECV')
- 同期型メッセージの送信　CBLDCMCF('SENDSYNC')

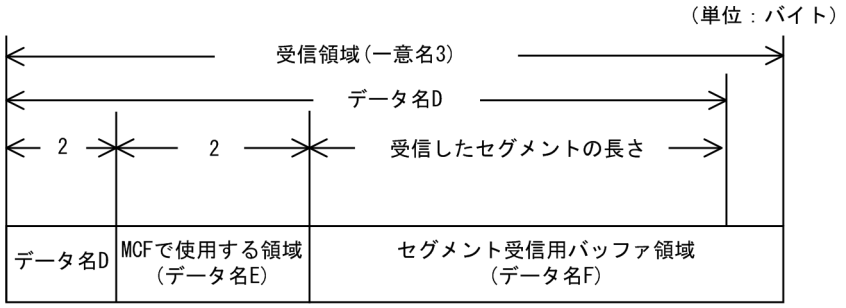
送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。単一セグメントだけ扱えます。

セグメントを送信する領域（一意名 1 で示す領域）の形式を次に示します。



同期型メッセージの送受信の場合，セグメントを受信する領域（一意名 3 で示す領域）も設定します。受信できるメッセージの一つのセグメントの最大長は，32763 バイトです。

セグメントを受信する領域の形式を次に示します。



UAP で値を設定する項目

●FOR 句

次のどちらかの値を設定します。

OUTPUT

一方送信メッセージの送信

I-O

応答メッセージの送信，同期型のメッセージの送受信，または，同期型のメッセージの送信。

●SYMBOLIC TERMINAL 句

論理端末名称を設定したデータ項目を設定します。データ名 2 にメッセージ出力先の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

●SYNCHRONOUS MODE 句

非同期型でメッセージを送信するか，同期型でメッセージを送信するかを設定します。

SYNC

同期型のメッセージの送信または送受信

同期型のメッセージの送信または同期型のメッセージの送受信のとき設定します。

ASYNC

非同期型のメッセージの送信

一方送信メッセージの送信または応答メッセージの送信のとき設定します。

データ名 6

次の値を設定したデータ項目

'0'または'△'：非同期型のメッセージの送信

'1'：同期型のメッセージの送信

省略した場合は，ASYNC（非同期型のメッセージの送信）が設定されます。

●SWITCHING MODE 句

一方送信メッセージの場合に，一般か優先かを設定します。

NORMAL

一般の一方送信メッセージ

PRIOR

優先の一方送信メッセージ

データ名 7

次の値を設定したデータ項目

'0'または'△'：一般の一方送信メッセージ

'1'：優先の一方送信メッセージ

省略した場合，および非応答型のアプリケーションから応答メッセージを送信した場合は，NORMAL（一般の一方送信メッセージ）が設定されます。

●NEXT TRANSACTION 句

＜継続問い合わせ応答形態の場合＞

次のメッセージ受信時に起動するアプリケーションを設定します。データ名 8 に起動する MHP のアプリケーション名を設定します。アプリケーション名は最大 8 バイトの長さです。8 バイトに満たない名称を設定する場合は，後ろを空白で埋めてください。

NEXT TRANSACTION 句を省略，または，データ名 8 に空白を設定した場合，実行中のアプリケーションを次のメッセージ受信時に再び起動します。

応答メッセージの送信を行う SEND（メッセージの送信）文を呼び出すサービスで DISABLE（継続問い合わせ応答の終了）文を呼び出す場合は NEXT TRANSACTION 句を省略，または，データ名 8 に空白を設定してください。

継続問い合わせ応答を引き継いだエラーイベントで応答メッセージの送信を行う SEND（メッセージの送信）文を発行する際に，NEXT TRANSACTION 句を省略，または，データ名 8 に空白を設定した場合，継続問い合わせ応答を終了します。ただし，継続問い合わせ応答を引き継いだエラーイベントで，SEND（アプリケーションプログラムの起動）文を発行して継続問い合わせ応答型のアプリケーションを起動し，起動先のアプリケーションで応答メッセージの送信を行う SEND（メッセージの送

信）文を発行する際に，NEXT TRANSACTION 句を省略，または，データ名 8 に空白を設定した場合，継続問い合わせ応答を終了しないで，SEND（アプリケーションプログラムの起動）文を発行したときに設定した継続問い合わせ応答型のアプリケーションを次のメッセージ受信時に再び起動します。

＜継続問い合わせ応答形態以外の場合＞

NEXT TRANSACTION 句を省略，または，データ名 8 に空白を設定します。

●DETAIL MODE 句

非同期型のメッセージの送信の場合に，出力通番を付けるかどうかを設定します。データ名 10 に次のどちらかを設定します。

データ名 10

次の値を設定したデータ項目

'0'または'△'：出力通番を付けます。

'1'：出力通番を付けません。

省略した場合，および非応答型のアプリケーションから応答メッセージを送信した場合は，出力通番を付けません。

●WAITING TIME 句

SEND 文を実行してから終了するまでの，最大時間を設定したデータ項目を設定します。同期型メッセージの送受信，または同期型メッセージの送信の場合に設定します。

データ名 11

監視時間の値を HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で設定したデータ項目を設定します。

省略した場合，またはデータ名 11 に'00000000'を設定した場合，同期型メッセージの送受信のときは，MCF マネジャ定義の UAP 共通定義で指定した同期型送受信監視時間（mcfmuap -t sndrcvtim）が設定されます。同期型メッセージの送信のときは，MCF マネジャ定義の UAP 共通定義で指定した同期型送信監視時間（mcfmuap -t sndtim）が設定されます。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，設定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，設定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3 秒以上の値の設定を推奨します

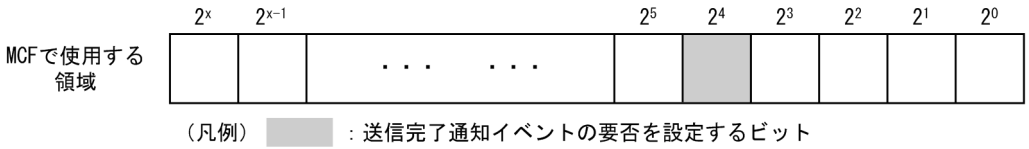
●データ名 A

送信するセグメントの長さ＋ 4 を設定します。

●データ名 B

送信完了通知イベントを設定します。データ名 B とビットの設定値の関係を次に示します。

AIX，HP-UX の場合



送信完了通知イベントの要否は，データ名 B の領域の 2^4 ビットに設定します。

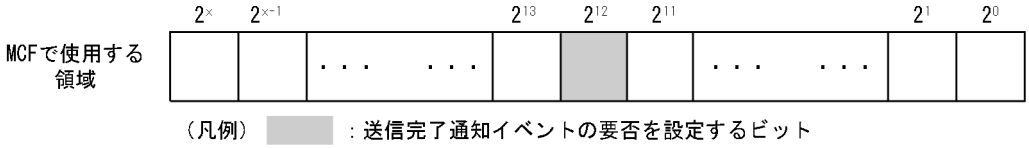
領域の 2^4 ビットに 1 を設定

送信完了通知イベントを通知させます。ただし，送信完了通知イベント処理用の MHP を MCF アプリケーション定義で指定していない場合は無効です。

領域の 2^4 ビットに 0 を設定

送信完了通知イベントを通知させません。

Linux，Windows の場合



送信完了通知イベントの要否は，データ名 B の領域の 2^{12} ビットに設定します。

領域の 2^{12} ビットに 1 を設定

送信完了通知イベントを通知させます。ただし，送信完了通知イベント処理用の MHP を MCF アプリケーション定義で指定していない場合は無効です。

領域の 2^{12} ビットに 0 を設定

送信完了通知イベントを通知させません。

●データ名 C

送信するセグメントの内容を設定します。一つのセグメントで 32000 バイトまで送信できます。

●データ名 E

MCF で使用する領域です。

●WITH 句

送信するセグメントが単一の論理メッセージであることを設定します。

EMI

単一セグメントを設定します。

一意名 2

次の値を設定したデータ項目

'2'：EMI（単一セグメント）

指定を省略した場合は，EMI（単一セグメント）を設定します。

●BEFORE 句

同期型のメッセージの送受信の場合に，セグメントを受信するデータ項目（一意名 3）を設定します。一方送信メッセージの送信，応答メッセージの送信，または，同期型のメッセージの送信の場合，BEFORE 句を省略します。

OpenTP1 から値が返される項目

●STATUS KEY 句

ステータスコードを受け取りたい場合に設定します。省略した場合は，ステータスコードを受け取れません。データ名 1 にステータスコードが返されます。

●データ名 D

受信したセグメントの長さ＋ 4 が返されます。

●データ名 F

受信したセグメントの内容が返されます。一つのセグメントで 32763 バイトまで受信できます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71002	メッセージキューへの出力処理中に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	MCF が終了処理中のため，メッセージの送信を受け付けられません。
	データ名 A に 32004 バイトを超える値を設定しています。
71003	メッセージキューが満杯です。
71004	メッセージを格納するバッファをメモリ上に確保できませんでした。
71108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。

ステータスコード	意味
72000	＜MHP の実行でリターンした場合＞ • 先頭セグメントを受信する RECEIVE 文を実行する前に，SEND 文を実行しています。 • 非応答型のアプリケーションからの問い合わせ応答をしない（UAP 共通定義（mcfmuap -c）の noansreply オペランドに no を指定）場合に，非応答型のアプリケーションから応答メッセージを送信しています。
	＜SPP の実行でリターンした場合＞ • トランザクションでない SPP の処理から，一方送信メッセージを送信する SEND 文を実行しています。 • SPP では，応答メッセージを送信する SEND 文を実行できません。
72001	SYMBOLIC TERMINAL 句に設定した論理端末名称が間違っています。
	SYMBOLIC TERMINAL 句に設定した出力先の論理端末名称は，MCF で定義していません。
	SEND 文を実行できない論理端末を設定しています。
	＜応答メッセージの送信を行う（FOR 句に I-O を設定し，かつ，SYNCHRONOUS MODE 句を省略または SYNCHRONOUS MODE 句に ASYNC を設定または SYNCHRONOUS MODE 句に 0 もしくは空白を設定したデータ名 6 を設定）場合＞ • 入力元論理端末は，応答メッセージの送信をサポートしていません。 • 入力元論理端末は，問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用していません（コネクション定義（mcftalccn -l）の replymsg オペランドを省略，または no を指定）。
	＜同期型メッセージの送信または同期型メッセージの送受信を行う（FOR 句に I-O を設定し，かつ，SYNCHRONOUS MODE 句に SYNC を設定または 1 を設定したデータ名 6 を設定）場合＞ 問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用している（コネクション定義（mcftalccn -l）の replymsg オペランドに yes を指定）場合，問い合わせ応答中または継続問い合わせ応答中のため受け付けられません。
72008	応答型のアプリケーションから SEND（アプリケーションプログラムの起動）文を呼び出して応答型のアプリケーションを起動したあとで，応答メッセージを送信する SEND 文を呼び出しました。
	継続問い合わせ応答型のアプリケーションから SEND（アプリケーションプログラムの起動）文を呼び出して継続問い合わせ応答型のアプリケーションを起動したあとで，応答メッセージを送信する SEND 文を呼び出しました。
72011	継続問い合わせ応答型でないアプリケーションが，次起動アプリケーションを設定して，応答メッセージを送信する SEND 文を呼び出しました。
72012	MCF バッファグループ定義のバッファ長が不足しました。
	MCF 通信プロセスは相手システムからメッセージを受信しましたが，UAP への応答連絡で RPC 通信の送信可能な上限値を超えました。
72013	データ名 F のサイズを超えるセグメントを受信しました。データ名 F のサイズを超えた部分は切り捨てられました。
	32763 バイトを超えるセグメントを受信しました。32763 バイトを超えた部分は切り捨てられました。
72017	DETAIL MODE 句に設定した値が間違っています。

ステータスコード	意味
72018	SWITCHING MODE 句に設定した値が間違っています。
72020	SYNCHRONOUS MODE 句に設定した値が間違っています。
72024	FOR 句に設定した値が間違っています。
72026	WITH 句に設定した値が間違っています。
72036	データ名 F のサイズが不足しています。5 バイト以上の領域を確保してください。
72037	非同期型メッセージの送信（SYNCHRONOUS MODE 句を省略，または SYNCHRONOUS MODE 句に ASYNC を設定，または SYNCHRONOUS MODE 句に 0 もしくは空白を設定したデータ名 6 を設定）の場合，BEFORE 句は設定できません。
72041	データ名 A に 0 から 4 バイト，またはマイナス値を設定しています。
72044	DISABLE 文を呼び出したあとで，次起動アプリケーションを設定して応答メッセージを送信する SEND 文を呼び出しました。
72045	NEXT TRANSACTION 句に継続問い合わせ応答型でないアプリケーションのアプリケーション名を設定して応答メッセージを送信する SEND 文を呼び出しました。
72046	NEXT TRANSACTION 句に設定したアプリケーション名が異なる応答メッセージの送信を行う SEND 文を 2 回以上呼び出しました。
72047	NEXT TRANSACTION 句にアプリケーション属性定義（mcfaalcap）に定義されていないアプリケーション名を設定して応答メッセージを送信する SEND 文を呼び出しました。
72073	非同期メッセージを送信処理中です。
73001	データ名 11 に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
	無通信監視時間（コネクション定義（mcftalccn -k）の notrftime オペランド指定値）が経過しましたが，相手システムからの通信がありません。
	相手システムからメッセージを受信しましたが，受信バッファを確保できませんでした。
	メッセージの送信処理中に相手システムからメッセージを受信したため，非同期受信処理を行いましたが，障害（UOC 障害など）が発生しました。
	出力先の論理端末で MCF 通信プロセスの内部障害が発生しました。
73002	＜同期型メッセージの送信または同期型メッセージの送受信を行う（FOR 句に I-O を設定し，かつ，SYNCHRONOUS MODE 句に SYNC を設定または 1 を設定したデータ名 6 を設定）場合＞ コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指定）場合に，MHP でメッセージ受信後にコネクションが再確立されたため，送信を抑止しました。
73003	メッセージ受信仕掛り中です。
73005	WAITING TIME 句に設定した時間が経過しましたが，論理端末からの応答がありません。
73010	入力または出力メッセージ編集 UOC で障害が発生しました。
	メッセージの読み込み時に障害が発生しました。
73015	出力先の論理端末は，ほかの UAP で仕掛り中です。

ステータスコード	意味
73018	WAITING TIME 句に設定した値が間違っています。
73019	メッセージ送信完了監視タイマのタイムアウトが発生しました。
73020	出力先の論理端末は停止しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

SEND － 一時記憶データの更新（データ操作言語）

形式

DATA DIVISION（通信記述項）の指定

```
CD  通信記述名
    FOR  I-O  STORAGE
   〔STATUS  KEY  IS  データ名1〕.
```

DATA DIVISION（データ記述項）の指定

```
01  一意名1.
    02  データ名A   PIC 9(4)  COMP.
    02  データ名B   PIC X(4).
    02  データ名C   PIC X(n).
```

PROCEDURE DIVISION（通信文）の指定

```
SEND  通信記述名  FROM  一意名1.
```

機能

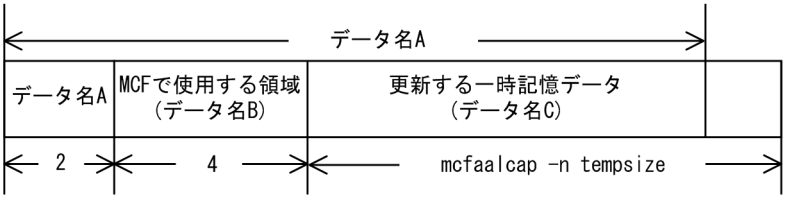
次に示す CALL インタフェースの機能を実現します。

- 一時記憶データの更新　CBLDCMCF('TEMPPUT△')

アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドには，更新する一時記憶データ長（データ名 A から 6 を減算した値）以上の値を指定してください。

更新する領域（一意名 1 で示す領域）の形式を次に示します。

（単位：バイト）



UAP で値を設定する項目

●FOR 句

一時記憶データの更新であることを示す，I-O STORAGE を設定します。

●データ名 A

一意名 1 の長さ（データ名 A の長さ（2）＋データ名 B の長さ（4）＋データ名 C の長さ）を設定します。

●データ名 B

MCF で使用する領域です。

●データ名 C

一時記憶データが格納されている領域を設定します。

OpenTP1 から値が返される項目

●STATUS KEY 句

ステータスコードを受け取りたい場合に設定します。省略した場合は，ステータスコードを受け取れません。データ名 1 にステータスコードが返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71103	一時記憶データを更新するための領域をメモリ上に確保できませんでした。
72000	SPP では SEND（一時記憶データの更新）文を呼び出せません。
72024	FOR 句に設定した値が間違っています。
72035	データ名 A に設定した更新データの長さが，アプリケーション属性定義（mcfaalcap -n）の tempsize オペランドで定義した長さを超えています。
	データ名 A に 0 から 6 バイト，またはマイナス値を設定しています。
72042	通信記述項に SYMBOLIC TERMINAL 句の設定があり，空白以外が設定されています。
72101	継続問い合わせ応答型でないアプリケーションで，SEND（一時記憶データの更新）文を呼び出しました。
72105	RECEIVE（一時記憶データの受け取り）文を呼び出す前に，SEND（一時記憶データの更新）文を呼び出しました。
72106	先頭セグメントを受信する RECEIVE 文を呼び出す前に，SEND（一時記憶データの更新）文を呼び出しました。
72107	DISABLE 文を呼び出したあとで，SEND（一時記憶データの更新）文を呼び出しました。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

5 ユーザオウンコーディング，MCF イベントインタフェース

この章では，TP1/NET/TCP/IP のユーザオウンコーディングのインタフェース，および MCF イベントのインタフェースについて説明します。

5.1　ユーザオウンコーディングインタフェース

メッセージ送受信の UAP を，より多様な業務に対応させるために補助するプログラムを**ユーザオウンコーディング**（以降，**UOC** と略します）といいます。

TP1/NET/TCP/IP で使用する UOC を次に示します。

- 入力セグメント判定 UOC
- 入力メッセージ編集 UOC
- 出力メッセージ編集 UOC
- 送信メッセージ通番編集 UOC
- コネクション確立 UOC
- 受信メッセージ判定 UOC
- 受信メッセージの保留判定 UOC

UOC を使用する場合は，あらかじめ MCF メイン関数または UAP のメイン関数に UOC 関数のアドレスを登録し，UOC 関数のオブジェクトファイルを MCF 通信プロセスまたは UAP の実行形式プログラムに結合（リンケージ）しておく必要があります。また，UOC は C 言語で作成します。

5.1.1　入力セグメントの判定

入力セグメント判定 UOC は，メッセージを受信したとき，そのメッセージでセグメントが完成しているかどうか，完成している場合は，完成したセグメントと後続のセグメントの境界がどこであるかを判定します。UOC は，相手システムからのメッセージを受信するたびに起動します。セグメントが未完成のとき（後続メッセージがある場合）は，後続メッセージに対して，監視タイマを設定できます。

ただし，コネクションの切断を抑止している場合は，監視タイマを設定しないでください。コネクションの切断抑止については，「2.3.2(5)　コネクションの切断抑止」を参照してください。

なお，入力セグメント判定 UOC は，次に示す条件に該当する場合は呼び出されません。

- MCF イベントの発生
- UAP からのアプリケーションプログラムの起動
- 受信メッセージの組み立て機能の使用

メッセージのデータ形式の詳細については，「2.3.7　メッセージの分割と組み立て」を参照してください。

(1)　入力セグメントの判定

TP1/NET/TCP/IP は UOC を呼び出すと，受信したメッセージのセグメントが格納されている受信バッファを引き渡します。未完成の受信メッセージが TP1/NET/TCP/IP に蓄積されている場合，受信バッ

ファには未完成の受信メッセージと今回受信したメッセージが連結して格納されます。UOC では，受信バッファに格納された受信メッセージの内容を基に，セグメントが完成しているかどうかを判定し，リターン値を設定します。UOC での判定結果別に，設定する必要がある情報について説明します。

セグメントが未完成の場合

該当メッセージの残りサイズを設定する必要があります。UOC が呼び出された時点で残りサイズを一意に決定できないときは，一意に決定できる情報（セグメントの全体長など）が格納された位置までの残りサイズ，または，受信バッファの残りサイズを設定してください。

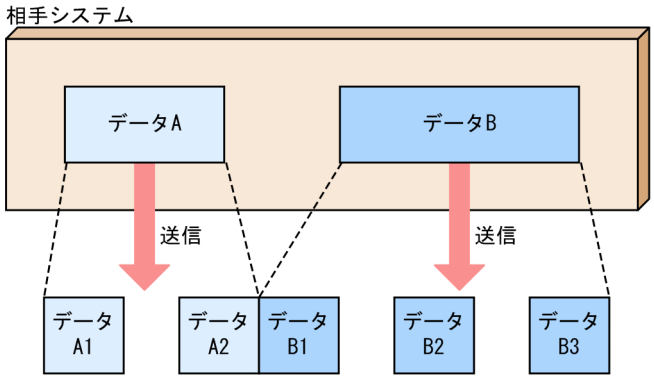
セグメントが完成した場合

該当メッセージのサイズと次メッセージについての情報を設定する必要があります。次メッセージがある場合，TP1/NET/TCP/IP は次メッセージの先頭を受信バッファの先頭に再配置して，再度 UOC を呼び出します。

入力セグメントの判定処理の流れを次の図に示します。ここでは，データ A とデータ B を相手システムから送信する場合を例に説明します。

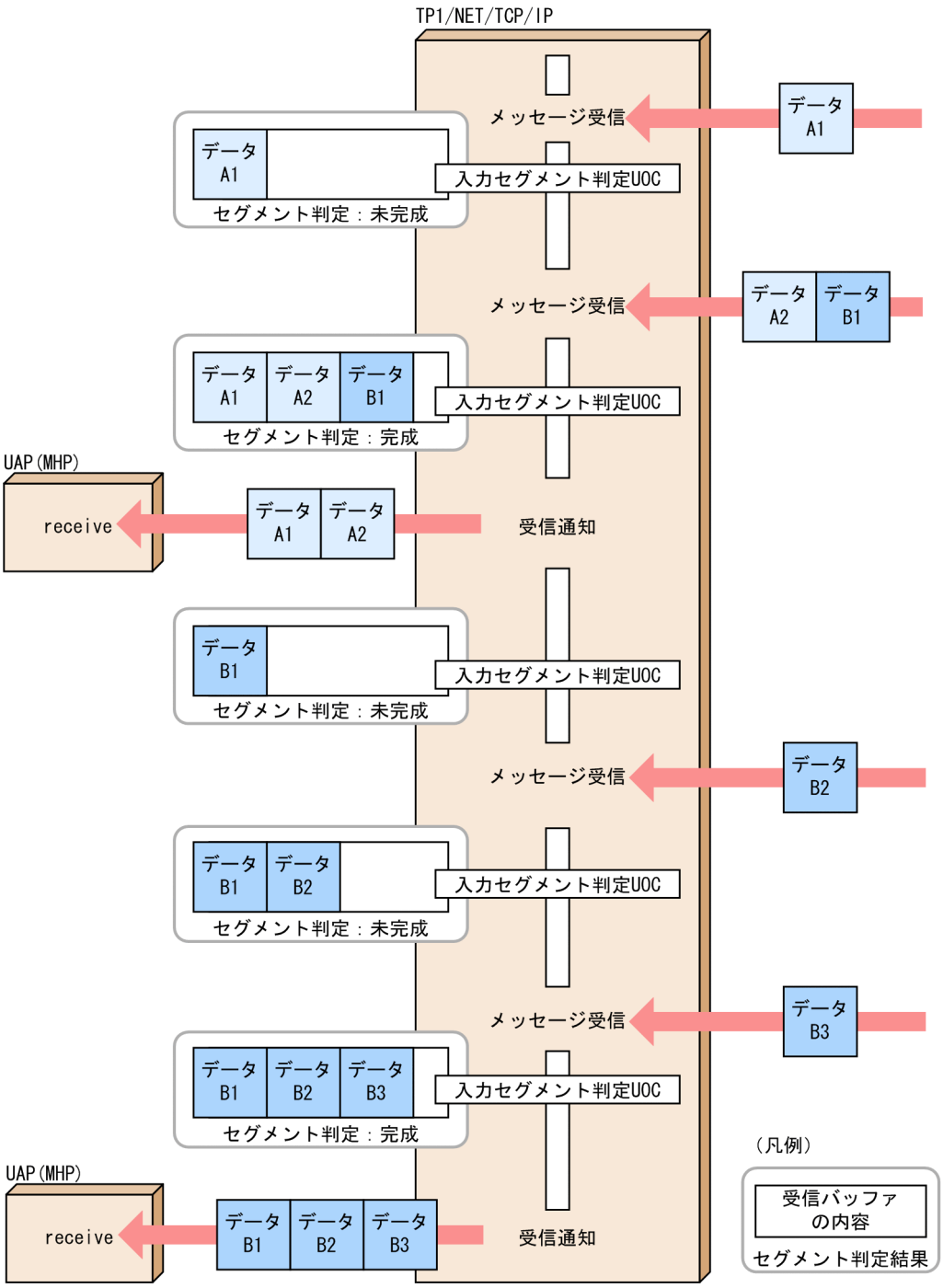
図 5-1　入力セグメントの判定処理の流れ

●相手システムからのメッセージの送信



ここでは，メッセージの送信に伴い，ネットワーク上でデータがセグメント単位に分割される場合を例に説明します。
TP1/NET/TCP/IPは，データA1から順に，セグメント単位でメッセージを受信します。

●相手システムからメッセージを受信した入力セグメント判定UOCの処理の流れ



各パラメタに設定した値が不正の場合，TP1/NET/TCP/IP は MCF イベントを通知して，コネクションを解放します。

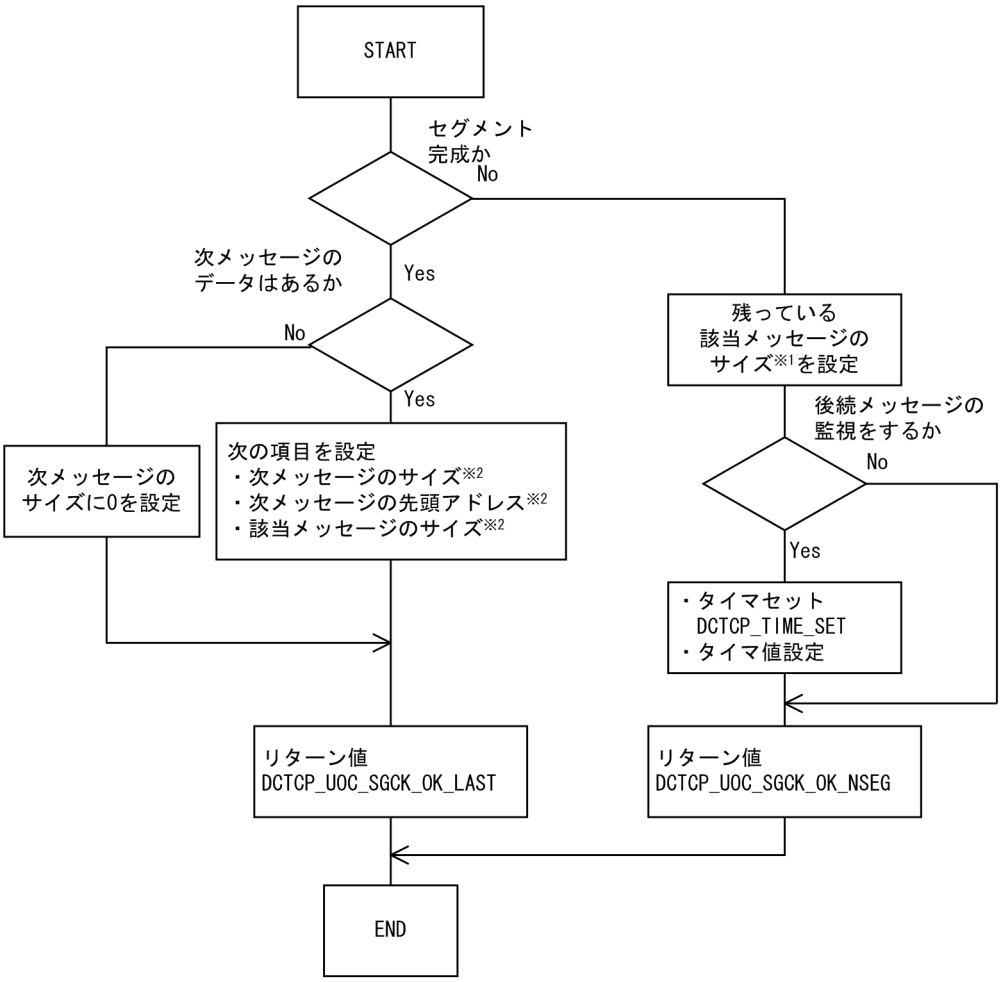
(2)　後続メッセージ監視タイマの設定

セグメントが未完成のとき，必要に応じて後続メッセージ監視タイマを設定できます。後続メッセージ監視タイマのセットを指示したあとで後続メッセージのタイムアウトが発生したときは，コネクションを解放して MCF イベント処理用 MHP を起動します。このとき，該当メッセージは破棄されます。また，各

パラメタに設定した値が不正の場合，TP1/NET/TCP/IP は MCF イベントを起動して，コネクションを解放します。

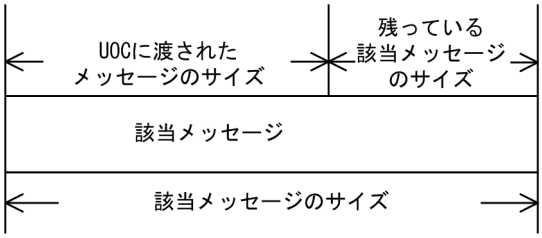
ユーザが作成する入力セグメント判定 UOC の処理の流れを次の図に示します。

図 5-2　入力セグメント判定の処理



注※1

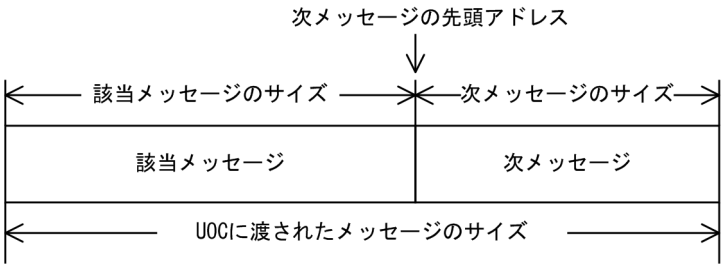
残っている該当メッセージのサイズを示します。



UOC を呼び出した時点で残っている該当メッセージのサイズを一意に決定できない場合は，一意に決定できる情報が格納された位置までの残りサイズ，または受信バッファの残りサイズを設定してください。

注※2

次メッセージのサイズ，次メッセージの先頭アドレス，該当メッセージのサイズを示します。



(3) UOC エラーリターン処理

UOC から DCTCP_UOC_SGCK_NG でリターンした場合，TP1/NET/TCP/IP は障害通知イベント（CERREVT）を通知します。

(4) UOC パラメタ不正の場合の処理

UOC で設定した値に不正があった場合，MCF はメッセージログを出力し，障害通知イベント（CERREVT）を通知してコネクションを解放します。該当するメッセージは破棄します。

(5) OpenTP1 への組み込み方法

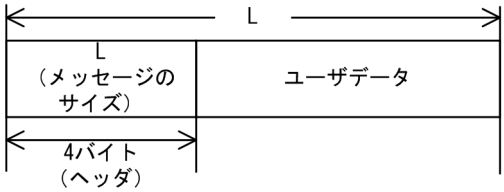
スタート関数（dc_mcf_svstart）を発行する MCF メイン関数に，作成した UOC の関数アドレスを設定します。入力セグメント判定 UOC の関数アドレスは任意に決められます。UOC 関数をコンパイルして生成した UOC オブジェクトファイルを，UOC 関数を登録した MCF メイン関数と結合して，TP1/NET/TCP/IP の実行形式プログラムを生成します。MCF メイン関数の詳細については，「8.2 MCF メイン関数の作成」を参照してください。

(6) 標準提供 UOC

入力セグメント判定 UOC には，TP1/NET/TCP/IP が標準提供する UOC（標準提供 UOC）があります。標準提供 UOC を使用する場合は，MCF メイン関数に標準提供 UOC 関数（dc_mcf_stduoc_tcp_segchk）の関数アドレスを登録します。

標準提供する入力セグメント判定 UOC が正常に動作するためには，受信するメッセージの先頭 4 バイト（ヘッダ）にメッセージ全体の長さ（メッセージのサイズ）が設定されていなければなりません。メッセージのサイズはバイト順序がビッグエンディアン※である整数型でなければなりません。また，標準提供の入力セグメント判定 UOC を使用する場合，メッセージの先頭にアプリケーション名を設定できなくなります。このため，アプリケーション名は入力メッセージ編集 UOC など，別の方法で決定してください。また，セグメント未完のときの後続メッセージに対する監視タイマ値を 30 秒で設定します。

メッセージの形式を次に示します。



注※

ビッグエンディアンとはバイトを低い方から順番に並べて，最下位のビットを最上位のバイトに置く方式のことです。

バイト0								バイト1								バイト2								バイト3							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

(a) 妥当性のチェック

入力セグメント判定 UOC は，ヘッダ（メッセージのサイズ）の妥当性をチェックします。

受信メッセージの先頭 4 バイトに格納されている値（メッセージのサイズ）が 4 バイト未満の場合は，エラーリターンします。

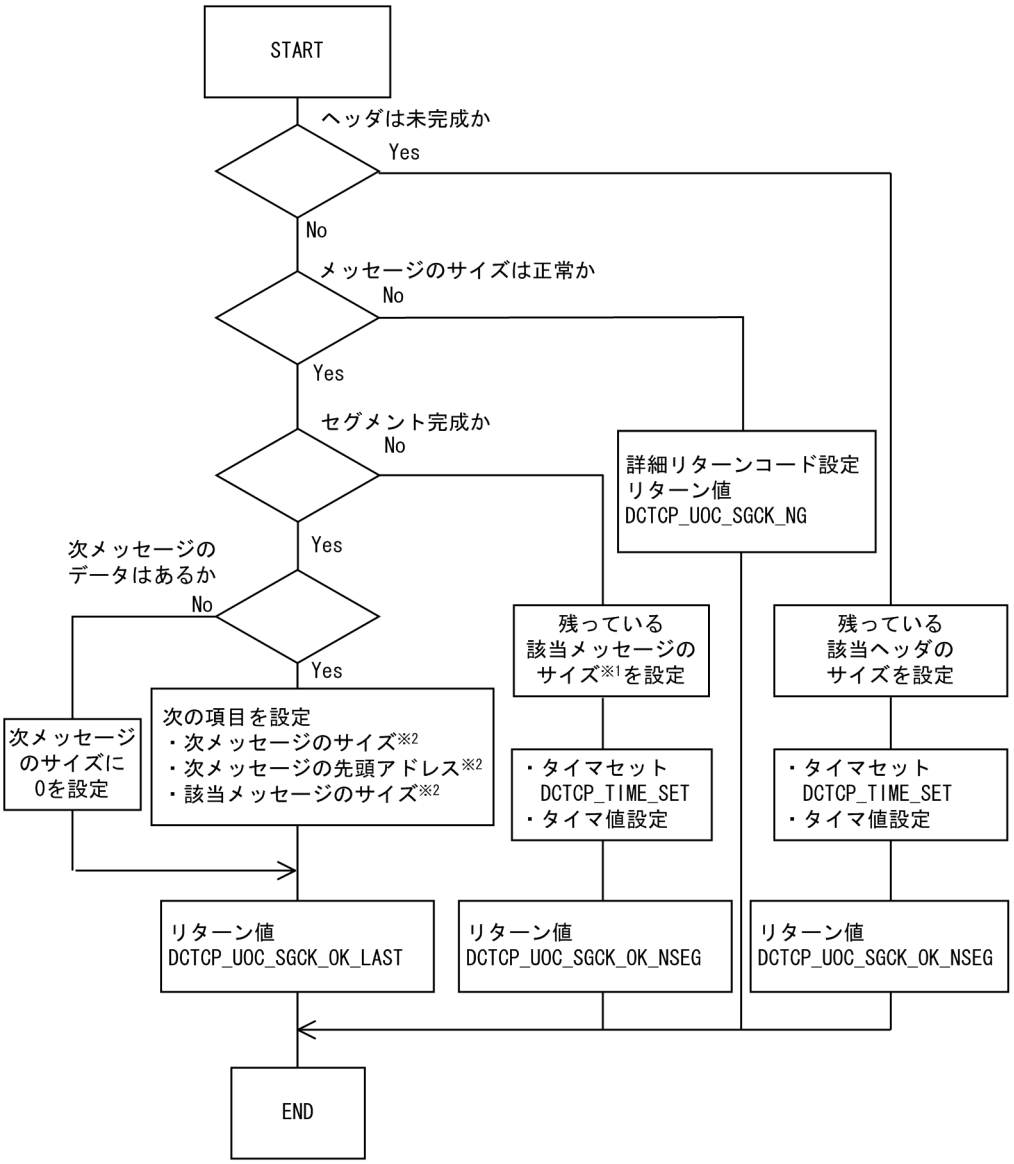
(b) エラー時の設定内容

入力セグメント判定 UOC でエラーを検出すると，リターンコードに DCTCP_UOC_SGCK_NG を設定します。その場合，詳細リターンコード（rtn_detail）に-19001（メッセージのサイズ不正）を設定します。

(c) 処理の流れ

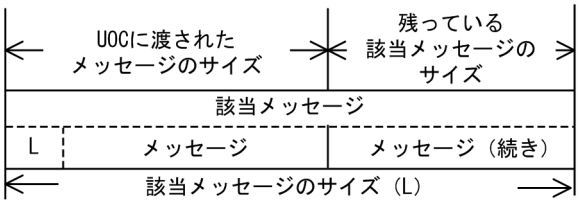
標準提供 UOC の処理の流れを次の図に示します。

図 5-3　標準提供 UOC の処理の流れ



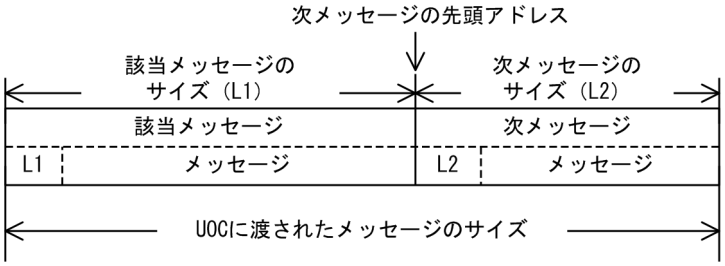
注※1

残っている該当メッセージのサイズを示します。



注※2

次メッセージのサイズ，次メッセージの先頭アドレス，該当メッセージのサイズを示します。



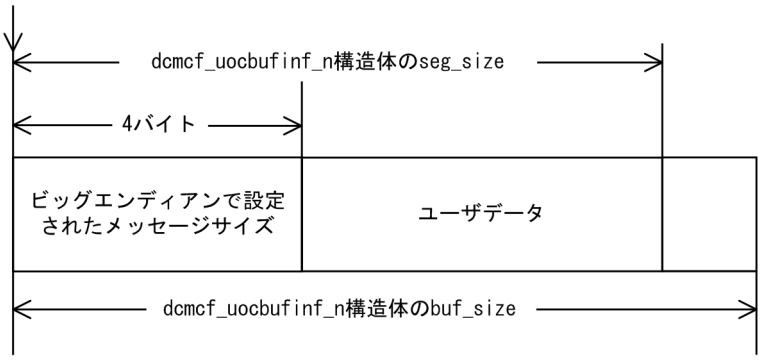
(d) UAP や UOC に通知するメッセージの形式

UAP や UOC に通知するメッセージの形式を次に示します。

ホストバイトオーダがリトルエンディアンの OS（Linux または Windows）では，ビックエンディアンで設定されたメッセージサイズを参照する場合，UAP または UOC でバイトオーダを変換してください。

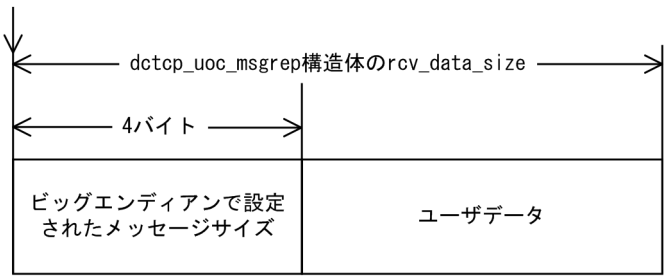
入力メッセージ編集UOCの場合

dcmcf_uocbufinf_n構造体のbuf_adr　　（単位：バイト）



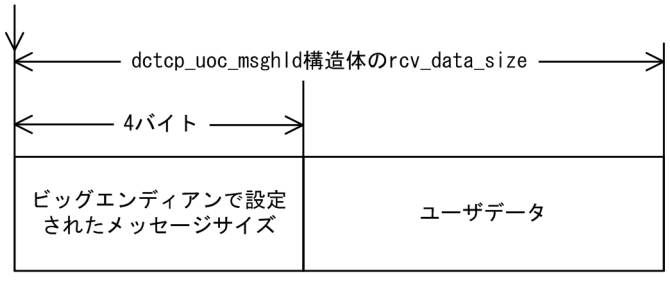
受信メッセージ判定UOCの場合

dctcp_uoc_msgrep構造体のrcv_data_adr　　（単位：バイト）



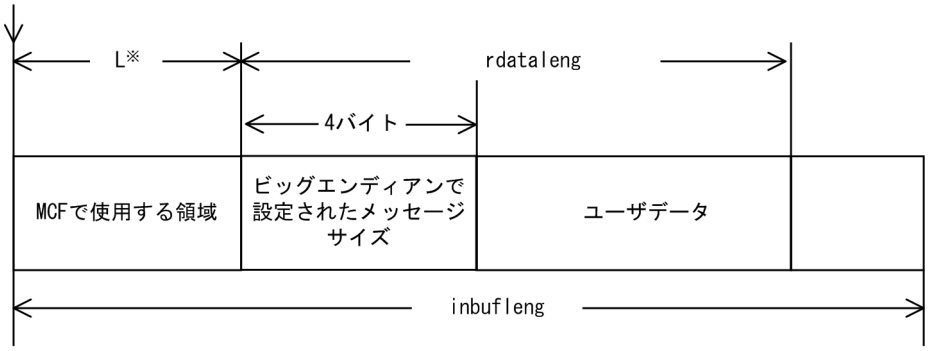
受信メッセージ保留判定UOCの場合

dctcp_uoc_msghld構造体のrcv_data_adr　　（単位：バイト）



UAP(C言語)の場合

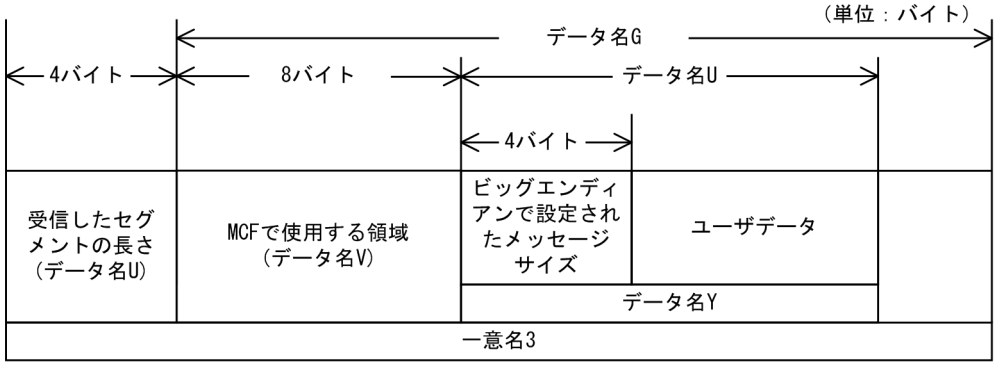
recvdata　　（単位：バイト）



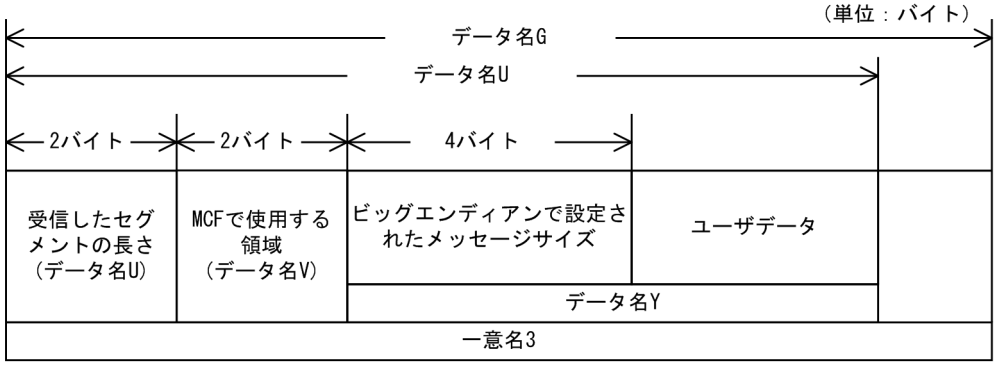
注※
　バッファ形式1の場合は，8バイト
　バッファ形式2の場合は，4バイト

UAP(CALL形式のCOBOL言語)の場合

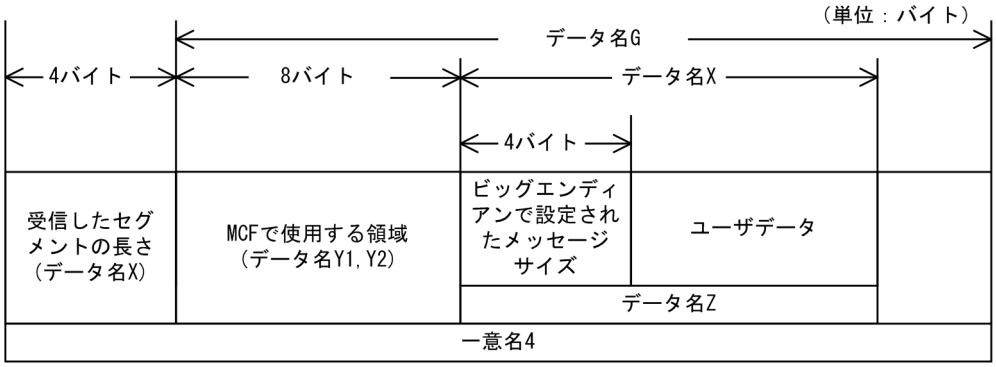
・CBLDCMCF('RECEIVE')，CBLDCMCF('RECVSYNC')のバッファ形式1の場合



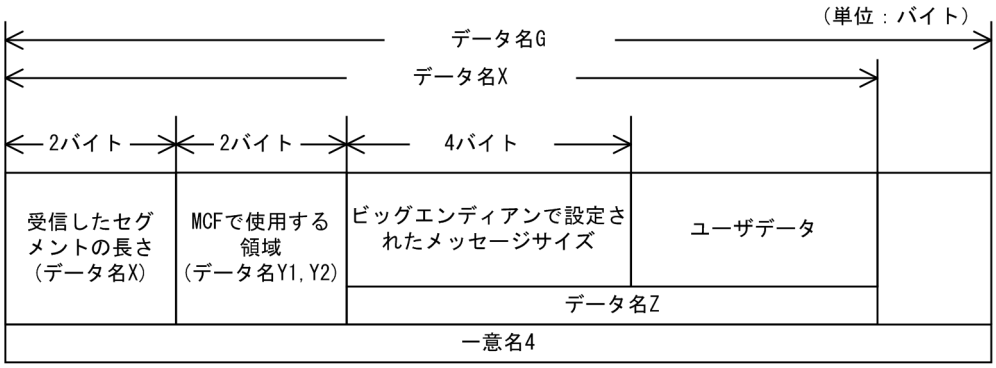
・CBLDCMCF('RECEIVE')，CBLDCMCF('RECVSYNC')のバッファ形式2の場合



・CBLDCMCF('SENDRECV')のバッファ形式1の場合

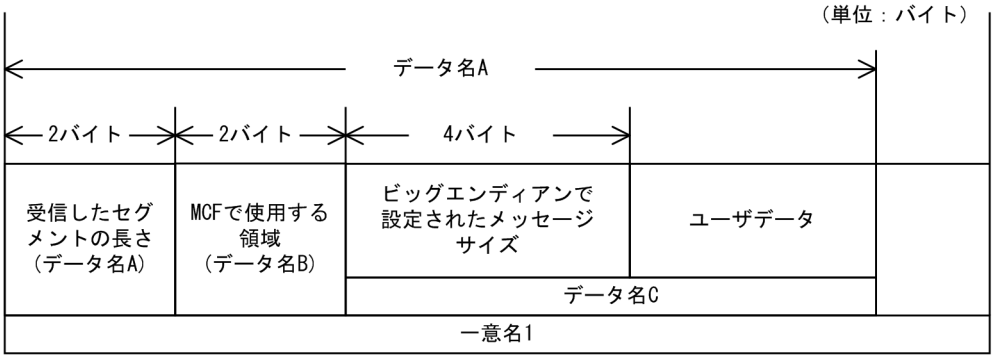


・CBLDCMCF('SENDRECV')のバッファ形式2の場合

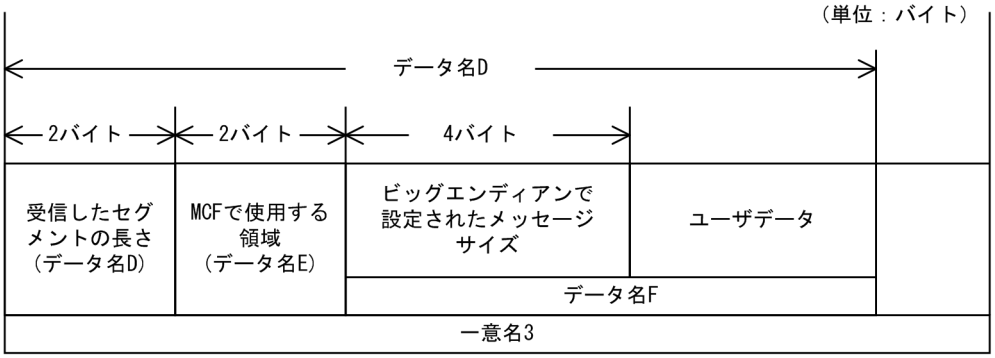


UAP(DML形式のCOBOL言語)の場合

・RECEIVEの場合



・SENDの場合



5.1.2　入力セグメント判定 UOC インタフェース

入力セグメント判定 UOC は，次に示す形式で呼び出します。

(1)　形式

ANSI C，C++の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>
DCLONG uoc_func(dctcp_uoc_sgck *parm)
```

K&R 版 C の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>
DCLONG    uoc_func(parm)

dctcp_uoc_sgck *parm ;
```

(2)　説明

uoc_func（入力セグメント判定 UOC）を呼び出すとき，MCF は次に示す所定のパラメタを parm に設定します。

(3)　パラメタの内容

(a)　dctcp_uoc_sgck の内容

```
typedef struct {
    DCLONG pro_kind;            …プロトコル種別
    char le_name[9];            …論理端末名称
    char reserve1[7];           …予備
    DCLONG rcv_prim;            …受信サービスプリミティブ
    char *rcv_data_adr;         …受信データ先頭アドレス
    DCLONG rcv_data_size;       …受信データ有効サイズ
    char *uoc_inf_adr;          …MCF使用領域
    DCLONG uoc_inf_size;        …MCF使用領域
    dctcp_sguoc_prot *pro_indv_ifa;
                                …プロトコル個別インタフェース
                                  領域アドレス
    dctcp_uoctimer_inf *ptimerinf_adr;
                                …プロトコルタイマ情報アドレス
    DCLONG rtn_detail;          …詳細リターンコード
    DCLONG rcv_buf_size;        …受信バッファサイズ
    DCLONG reserve2[1];         …予備
} dctcp_uoc_sgck;
```

(b) dctcp_uoctimer_inf（タイマ情報），dctcp_sguoc_prot（領域アドレス）の内容

```
typedef struct {
    DCLONG timer_code;          …タイマセット指示
    DCLONG timer_value;         …後続メッセージ監視タイマ値
    DCLONG timer_result;        …MCF使用領域
    DCLONG reserve1[2];         …予備
} dctcp_uoctimer_inf;

typedef struct {
    DCLONG rest_data_size;      …残っている該当メッセージのサイズ
    DCLONG next_data_size;      …次メッセージの有効長
    char *next_data_adr;        …次メッセージの先頭アドレス
    DCLONG now_data_size;       …該当メッセージの有効長
    DCLONG reserve[2];          …予備
} dctcp_sguoc_prot;
```

(4) MCF が値を設定する項目

(a) dctcp_uoc_sgck

- pro_kind

  プロトコル種別として，次の値が設定されます。

  DCMCF_UOC_PRO_TCP

  TCP/IP プロトコル

- le_name

  メッセージを入力した論理端末の名称が設定されます。

- rcv_prim

  受信サービスプリミティブとして，次のどちらかの値が設定されます。

  DCMCF_UOC_RCV_BRD

  一方送信メッセージの受信，問い合わせメッセージの受信または同期型メッセージの受信

  DCMCF_UOC_RCV_REP_SR

  同期型メッセージの送受信関数によるメッセージの受信

- rcv_data_adr

  受信したメッセージの先頭アドレスが設定されます。

- rcv_data_size

  受信したメッセージの有効長が設定されます。

- uoc_inf_adr

  MCF で使用するパラメタです。

- uoc_inf_size

  MCF で使用するパラメタです。

- pro_indv_ifa
  プロトコル個別インタフェース領域アドレスが設定されます。
- ptimerinf_adr
  プロトコルタイマ情報アドレスが設定されます。
- rcv_buf_size
  受信したメッセージを格納しているバッファ長が設定されます。

(b) dctcp_uoctimer_inf（タイマ情報）

- timer_result
  MCF で使用するパラメタです。

(5) ユーザが値を設定する項目

(a) dctcp_uoc_sgck

- rtn_detail
  詳細リターンコードを設定できます。このコードは，UOC が DCTCP_UOC_SGCK_NG をリターンしたときに，MCF に返されます。MCF は，詳細リターンコードをメッセージログファイルに出力します。
  詳細リターンコードは，-19999～-19000 の範囲で指定してください。
  なお，標準提供の入力セグメント判定 UOC は，詳細リターンコードとして-19001 を返します。

(b) dctcp_sguoc_prot（領域アドレス）

- rest_data_size
  残っている該当メッセージのサイズを設定します。
  受信したメッセージのセグメントが未完成のときは，不足分のメッセージのサイズを設定します。
  DCTCP_UOC_SGCK_OK_NSEG でリターンした場合だけ有効です。
- next_data_size
  次のメッセージのサイズを設定します。
  受信したメッセージのセグメントが完成しているとき，次のメッセージが含まれていれば，次のメッセージのサイズを設定します。含まれていない場合は 0 を設定します。
  DCTCP_UOC_SGCK_OK_LAST でリターンした場合だけ有効です。
- next_data_adr
  次のメッセージの先頭アドレスを設定します。next_data_size が 0 でない場合に設定します。
- now_data_size
  該当するメッセージのサイズを設定します。next_data_size が 0 でない場合に設定します。

(c) dctcp_uoctimer_inf（タイマ情報）

- timer_code

  後続メッセージ監視タイマを設定します。

  受信したメッセージのセグメントが未完成のとき，後続メッセージの受信に対して監視タイマをセットするかどうかを設定します。DCTCP_UOC_SGCK_OK_NSEG でリターンした場合だけ有効です。

  DCTCP_TIME_SET

  監視タイマをセットします。

  DCTCP_TIME_NO_SET

  監視タイマをセットしません。

- timer_value　　～〈10 進数字〉((1～2550))（単位：秒）

  後続メッセージの監視タイマの値を設定します。

  この値は，監視タイマをセットする場合に設定します。timer_code が DCTCP_TIME_SET の場合だけ有効です。

  注意事項

  監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，設定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，設定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

(6) リターン値

uoc_func()は次のコードでリターンしてください。

リターン値	意味
DCTCP_UOC_SGCK_OK_NSEG	正常リターン（セグメント未完成）
DCTCP_UOC_SGCK_OK_LAST	正常リターン（最終セグメント受信完了）
DCTCP_UOC_SGCK_NG	セグメント判定エラー

DCTCP_UOC_SGCK_OK_NSEG でリターンした場合は，残っている該当メッセージのサイズの値が有効になります。DCTCP_UOC_SGCK_OK_LAST でリターンした場合は，次のメッセージのサイズの値が有効になります。

(7) コーディング例

入力セグメント判定 UOC のコーディング例（K&R 版）を次に示します。また，このコーディング例を次のファイルで提供しています。

適用 OS が Windows の場合

%DCDIR%¥examples¥mcf¥tcpip¥cmlib¥c¥uoc.c

適用 OS が Linux の場合

/opt/OpenTP1/examples/mcf/TCPIP/cmlib/c/uoc.c

その他の適用 OS の場合

/BeTRAN/examples/mcf/TCPIP/cmlib/c/uoc.c

```
/************************************************************/
/*                                                          */
/* name = 入力セグメント判定UOC名                              */
/*                                                          */
/* func = 入力セグメント判定UOC（TP1/NET/TCP/IP用）             */
/*  （メッセージの先頭4バイトにメッセージ                        */
/*    サイズがint型で設定されている場合）                       */
/*                                                          */
/************************************************************/
/************************************************************/
/*    ヘッダファイル                                        */
/************************************************************/
#include <stdio.h>          /* UNIXシステムヘッダ             */
#include <sys/types.h>
#include <netinet/in.h>
#include <dcmcf.h>          /* TP1/Message Control 提供ヘッダ */
#include <dcmcfuoc.h>       /* TP1/NET/TCP/IP UOC 用ヘッダ    */
#include <dcmtcpu.h>        /* TP1/NET/TCP/IP ヘッダ          */

/************************************************************/
/*    定数宣言                                              */
/************************************************************/
#define ERR_RCVMSG_LEN  -19001
                    /* エラー詳細（受信メッセージサイズ不正）    */
#define DCTCP_TIME_VALUE  30
                    /* 後続メッセージ監視タイマ設定値（秒）      */
/************************************************************/
/* dc_mcf_stduoc_tcp_segchk                                 */
/*                                                          */
/* name = 入力セグメントを判定する。                           */
/*                                                          */
/* return = DCTCP_UOC_SGCK_OK_NSEG  ：正常（セグメント未完成）  */
/*          DCTCP_UOC_SGCK_OK_LAST  ：正常（最終セグメント受信完了）*/
/*          DCTCP_UOC_SGCK_NG       ：セグメント判定エラー      */
/*          <error detail>                                  */
/*             1  ：受信メッセージサイズ不正                    */
/* memo =1. メッセージの先頭4バイトにメッセージサイズが          */
/*          int型で設定されていることを前提とする。            */
/*                                                          */
/************************************************************/

DCLONG dc_mcf_stduoc_tcp_segchk(parm)
dctcp_uoc_sgck *parm;
    /* parm <parm=UOCインタフェース情報テーブルのアドレス>      */
    /* 属性 <type=dctcp_uoc_sgck *>                         */
{
/***** define valiable (dc_mcf_stduoc_tcp_segchk) **********/
```

```
  DCLONG   rcode ;
  int      *bufadr ;              /* 受信データ格納領域ポインタ   */
  DCLONG   msglen ;              /* メッセージサイズ */
  DCLONG  next_msglen ;          /* 次メッセージサイズ */
  dctcp_sguoc_prot *proadr ;   /* プロトコル個別インタフェース*/
                               /* 領域ポインタ */

/***** procedure start (dc_mcf_stduoc_tcp_segchk) **********/
  bufadr = (long *)parm->rcv_data_adr;
        /* CAST変換で受信データ格納領域のポインタ取得           */
  proadr = (dctcp_sguoc_prot *)parm->pro_indv_ifa;
        /* CAST変換でプロトコル個別インタフェース領域のポインタ取得 */
  if (parm->rcv_data_size < sizeof(msglen)) {
        /* ヘッダは未完成か？                                   */
    proadr->rest_data_size =
      sizeof(msglen) - parm->rcv_data_size;
        /* 残り該当メッセージサイズに残りヘッダサイズを設定       */
    parm->ptimerinf_adr->timer_code = DCTCP_TIME_SET;
                               /* タイマセット設定 */
    parm->ptimerinf_adr->timer_value = DCTCP_TIME_VALUE;
                               /* タイマ値設定 */
    return(DCTCP_UOC_SGCK_OK_NSEG);
                               /* セグメント未完成でリターン */
  }
  msglen = ntohl(*bufadr);
        /* メッセージの先頭4バイトよりメッセージサイズを取得 */
  if (msglen < sizeof(msglen)) {
        /* 受信メッセージサイズは不正か？ */
    parm->rtn_detail = ERR_RCVMSG_LEN;
        /* 詳細リターンコード設定 */
    return(DCTCP_UOC_SGCK_NG);
        /* セグメント判定エラー */
  }

  next_msglen = parm->rcv_data_size - msglen;
        /* 次メッセージサイズの算出 */

  if (next_msglen >=0) {          /* セグメント完成か？ */

  /******** セグメント完成時の処理 ********/
    if (next_msglen > 0) {        /* 次メッセージはあるか？ */
      /******** 次メッセージありの場合 ********/
      proadr->next_data_size = next_msglen;
                               /* 次メッセージサイズ設定 */
      proadr->next_data_adr = parm->rcv_data_adr + msglen;
                              /* 次メッセージ先頭アドレス設定 */
      proadr->now_data_size = msglen;
                              /* 該当メッセージサイズ設定 */
    }else{
      /******** 次メッセージなしの場合 ********/
      proadr->next_data_size = NULL;
                              /* 次メッセージサイズにヌルを設定 */
    }
    proadr->rest_data_size = NULL;
                              /* 残り該当メッセージサイズクリア */
    rcode = DCTCP_UOC_SGCK_OK_LAST;
                              /* 最終セグメント受信完了 */
  }else{
```

```
  /******** セグメント未完成時の処理 ********/
    proadr->rest_data_size = -next_msglen;
                              /* 残り該当メッセージサイズ設定 */
    parm->ptimerinf_adr->timer_code = DCTCP_TIME_SET;
                              /* タイマセット設定 */
    parm->ptimerinf_adr->timer_value = DCTCP_TIME_VALUE;
                              /* タイマ値設定 */
    rcode = DCTCP_UOC_SGCK_OK_NSEG;
                              /* セグメント未完成 */
  }
  return(rcode);              /* リターン */
}
```

5.1.3 入力メッセージの編集とアプリケーション名の決定

入力メッセージ編集 UOC は，受信した論理メッセージをユーザ任意の形式に変換します。次に，受信した論理メッセージを基に，ユーザは任意のアプリケーション名を決定できます。標準提供の入力セグメント判定 UOC を使用する場合，受信メッセージでアプリケーション名を決定できなくなります。ユーザが入力メッセージ編集 UOC，または論理端末定義（mcftalcle）の-v オプションで，アプリケーション名を決定してください。UOC は，MHP を起動するメッセージのセグメントを受信すると起動します。

なお，入力メッセージ編集 UOC は，次に示す条件に該当する場合は呼び出されません。

- MCF イベントの発生時
- UAP からのアプリケーションプログラムの起動

ユーザは，MCF メイン関数で UOC 関数アドレスを設定します。また，必要に応じてメッセージ編集用バッファグループ番号（コネクション定義（mcftalccn -e）の msgbuf オペランド）を定義します。

(1) 入力メッセージの編集

受信したメッセージが格納されている受信バッファ，および定義で指定した編集バッファを引き渡します。UOC では，これらのバッファを使用して，入力メッセージの編集ができます。

また，UAP に通知するメッセージのセグメントは，受信バッファ，または編集バッファのどちらかに格納されたものを使用できます。どちらのセグメントを使用するかは，UOC から返されるリターンコードによって選択できます。

各パラメタに設定した値が不正の場合，TP1/NET/TCP/IP は MCF イベントを通知して論理端末を閉塞またはコネクションを解放します。

(2) アプリケーション名の決定

該当する MCF 通信プロセスに入力メッセージ編集 UOC が登録されている場合，論理メッセージの受信と同時にアプリケーション名を決定できます。

このUOCでアプリケーション名を決定する場合，アプリケーション名の形式は，アプリケーション名格納領域の先頭から，'¥0'の手前までの1～8バイトの英数字です。先頭から9バイト目までに'¥0'がないときは，アプリケーション名を不正とします。

アプリケーション名の決定の処理は，「2.3.11　アプリケーション名の決定」を参照してください。

(3)　UOCエラーリターン処理

UOCからDCMCF_UOC_MSG_NGでリターンした場合，MCFはメッセージログを出力し，障害通知イベント（CERREVT）を通知します。

UOCで障害を検出し，MCFイベント処理用MHPを起動したい場合は，ユーザ任意のMCFイベント処理用MHPのアプリケーション名を設定します。このとき，MCFにはDCMCF_UOC_MSG_OK（_RCV）でリターンします。

(4)　UOCパラメタ不正の場合の処理

UOCで設定したパラメタに不正があった場合，MCFはメッセージログを出力し，障害通知イベント（CERREVT）を通知して，論理端末を閉塞またはコネクションを解放します。

(5)　OpenTP1への組み込み方法

入力セグメント判定UOCの組み込み方法と同じです。「5.1.1(5)　OpenTP1への組み込み方法」を参照してください。

5.1.4　入力メッセージ編集UOCインタフェース

入力メッセージ編集UOCは，次に示す形式で呼び出します。

(1)　形式

ANSI C，C++の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>※
DCLONG  uoc_func(dcmcf_uoc_min_n *parm)
```

K&R版Cの場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>※
DCLONG    uoc_func(parm)

dcmcf_uoc_min_n   *parm ;
```

注※

dcmtcp_min_ifa（プロトコル個別インタフェース）を参照する場合に指定が必要です。

(2) 説明

uoc_func（入力メッセージ編集 UOC）を呼び出すとき，MCF は次に示す所定のパラメタを parm に設定します。

(3) パラメタの内容

(a) dcmcf_uoc_min_n の内容

```
typedef struct {
    DCLONG pro_kind;              …プロトコル種別
    char le_name[9];              …論理端末名称
    char reserve1[7];             …予備
    DCLONG rcv_prim;              …受信サービスプリミティブ
    dcmcf_uocbuff_list_n *buflist_adr;
                                  …受信バッファリストアドレス
    dcmcf_uocbuff_list_n *ebuflist_adr;
                                  …編集バッファリストアドレス
    char aplname[9];              …アプリケーション名
    char reserve2[7];             …予備
    char *pro_indv_ifa;           …プロトコル個別インタフェース領域アドレス
    DCLONG rtn_detail;            …詳細リターンコード
    char reserve3[8];             …予備
} dcmcf_uoc_min_n;
```

(b) dcmcf_uocbuff_list_n（バッファリスト）の内容

```
typedef struct {
    DCLONG buf_num;               …バッファ情報数
    DCLONG used_buf_num;          …使用バッファ情報数
    char reserve1[8];             …予備
    dcmcf_uocbufinf_n buf_array[DCMCF_UOC_BUFF_MAX];
                                  …バッファ情報
} dcmcf_uocbuff_list_n;
```

(c) dcmcf_uocbufinf_n（バッファ情報）の内容

```
typedef struct {
    char *buf_adr;               …バッファアドレス
    DCULONG buf_size;            …バッファ最大長
    DCULONG seg_size;            …バッファ使用長
    char reserve1[4];            …予備
    dcmcfuoc_w_type buff_id;     …MCF内部情報1
    DCMLONG buff_addr;           …MCF内部情報2
    char reserve2[4];            …予備
} dcmcf_uocbufinf_n;
```

(d) dcmtcp_min_ifa（プロトコル個別インタフェース）の内容

```
typedef struct {
    char cont_status;           …継続問い合わせ応答ステータス
    char reserve1[63];          …予備
} dcmtcp_min_ifa;
```

(4) MCF が値を設定する項目

(a) dcmcf_uoc_min_n

- pro_kind

  プロトコル種別として，次の値が設定されます。

  DCMCF_UOC_PRO_TCP

  TCP/IP プロトコル

- le_name

  メッセージを入力した論理端末の名称が設定されます。

- rcv_prim

  受信サービスプリミティブとして，次の値が設定されます。

  DCMCF_UOC_RCV_BRD

  一方送信メッセージの受信および問い合わせメッセージの受信

  DCMCF_UOC_RCV_REP_RE

  同期型メッセージの受信

  DCMCF_UOC_RCV_REP_SR

  同期型メッセージの送受信関数によるメッセージの受信

- buflist_adr

  受信用バッファリストのアドレスが設定されます。

- ebuflist_adr

  編集用バッファリストのアドレスが設定されます。

  メッセージ編集バッファが未定義の場合，つまり，コネクション定義（mcftalccn）の-e オプションを省略した場合，ebuflist_adr には NULL が設定されます。

- aplname

  アプリケーション名が設定されます。設定するアプリケーション名を次の表に示します。

受信サービスプリミティブ	継続問い合わせ応答ステータス	論理端末定義（mcftalcle）の-v オプションの指定	設定するアプリケーション名
DCMCF_UOC_RCV_BRD	継続問い合わせ応答中	任意	応答メッセージの送信時に指定した次起動アプリケーション名

受信サービスプリミティブ	継続問い合わせ応答ステータス	論理端末定義（mcftalcle）の-v オプションの指定	設定するアプリケーション名
DCMCF_UOC_RCV_BRD	継続問い合わせ応答中	任意	（次起動アプリケーション名を省略した場合，応答メッセージの送信を行ったアプリケーション名）
	継続問い合わせ応答中以外（問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能未使用の場合を含みます）	あり	論理端末定義（mcftalcle）の-v オプションの指定値
		なし	なし（ヌル文字で埋められます）
DCMCF_UOC_RCV_REP_RE DCMCF_UOC_RCV_REP_SR	任意	任意	なし（ヌル文字で埋められます）

- pro_indv_ifa

  プロトコル個別インタフェース領域のアドレスが設定されます。

  設定する領域の内容を次の表に示します。

問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能（mcftalccn -l replymsg 指定値）	受信サービスプリミティブ	設定する領域の内容
使用（yes）	DCMCF_UOC_RCV_BRD	dcmtcp_min_ifa の内容を設定します。
	上記以外	MCF 使用領域です。
未使用（no）	任意	MCF 使用領域です。

(b) dcmcf_uocbuff_list_n（バッファリスト）

- buf_num

  バッファ情報の数として，1 が設定されます。

- buf_array

  バッファ情報の配列が設定されます。バッファ情報は，buf_num の数だけ設定されます。

(c) dcmcf_uocbufinf_n（バッファ情報）

- buf_adr

  バッファのアドレスが設定されます。

- buf_size

  バッファの最大長が設定されます。

- seg_size

  送信，または受信用バッファリストの場合だけ，バッファの使用長が設定されます。

- buff_id，buff_addr
  MCF で使用するパラメタです。

(d) dcmtcp_min_ifa（プロトコル個別インタフェース）

- cont_status
  継続問い合わせ応答ステータスが設定されます。
  'c'
  　継続問い合わせ応答中
  'n'
  　継続問い合わせ応答中以外

(5) ユーザが値を設定する項目

(a) dcmcf_uoc_min_n

- aplname
  一方送信メッセージの受信および問い合わせメッセージの受信の場合，UOC で決定したアプリケーション名を設定します。
  同期型メッセージの受信または同期型メッセージの送受信関数によるメッセージの受信の場合は，無効となります。
- rtn_detail
  詳細リターンコードを設定します。
  このコードは，UOC が DCMCF_UOC_MSG_NG でリターンしたときに，MCF に渡されます。MCF は，詳細リターンコードをメッセージログファイルに出力します。
  詳細リターンコードは，-19999～-19000 の範囲で指定してください。

(b) dcmcf_uocbuff_list_n（バッファリスト）

- used_buf_num
  使用したバッファ情報の数として，1 を設定します。

(c) dcmcf_uocbufinf_n（バッファ情報）

- seg_size
  バッファの使用長を設定します。

(6) リターン値

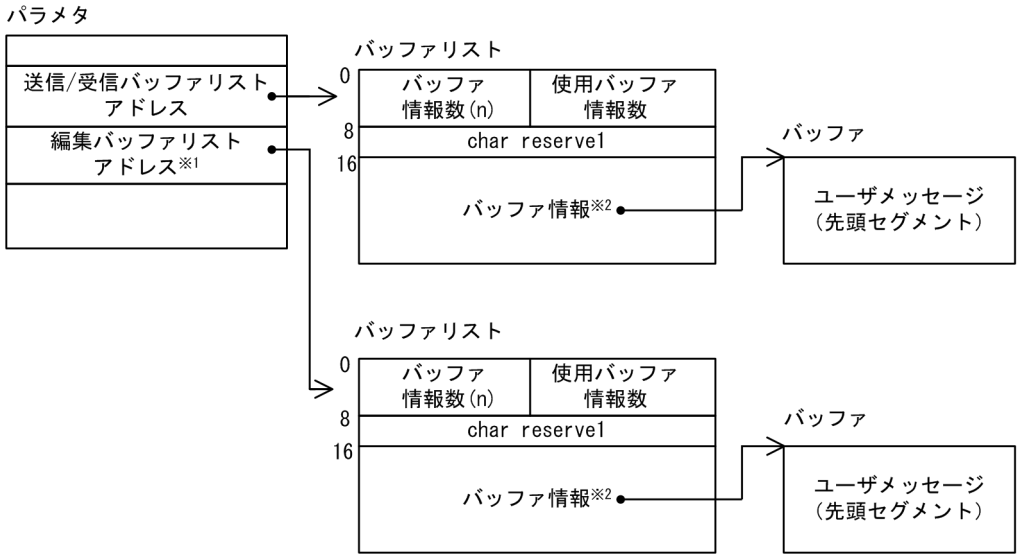
uoc_func()は次のコードでリターンしてください。

リターン値	意味
DCMCF_UOC_MSG_OK	正常リターン（編集バッファでスケジューリング）
DCMCF_UOC_MSG_OK_RCV	正常リターン（受信バッファでスケジューリング）
DCMCF_UOC_MSG_NG	メッセージ編集エラー

(7) パラメタとバッファの関係

UOC インタフェース用のパラメタとバッファの関係を次の図に示します。

図 5-4 UOC インタフェース用のパラメタとバッファの関係

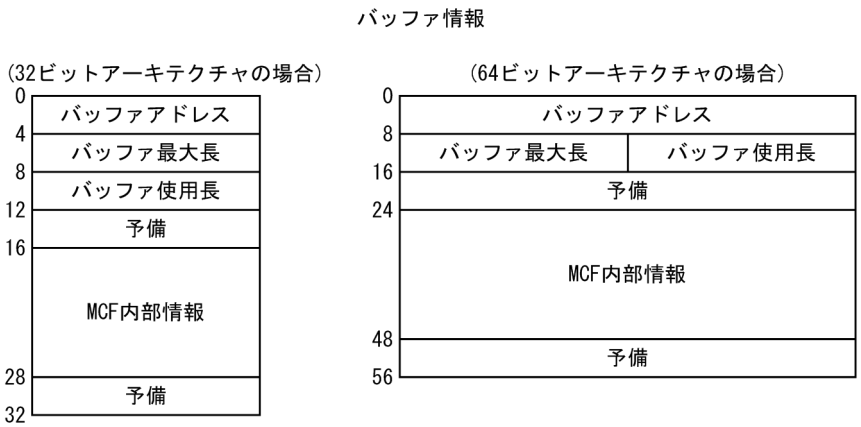


注※1

mcftalccn コマンドの-e オプションを指定しない場合，NULL となり，バッファリストとバッファは確保されません。

注※2

バッファ情報は次の形式をしています。



5.1.5　出力メッセージの編集

出力メッセージ編集 UOC は，送信メッセージの編集をする UOC です。出力メッセージの編集 UOC は，UAP が発行した送信メッセージを相手システムに実際に送信する前に処理するように位置させます。出力キューからセグメントを読み出すと起動します。

ユーザは，MCF メイン関数で UOC 関数アドレスを設定します。また，必要に応じてメッセージ編集用バッファグループ番号（コネクション定義（mcftalccn -e）の msgbuf オペランド）を定義します。

(1)　出力メッセージの編集

送信するメッセージが格納されている送信バッファ，および定義で指定した編集バッファを引き渡します。UOC では，これらのバッファを使用して，出力メッセージの編集処理ができます。

また，UOC からのリターンコードによって UAP に通知するメッセージとして送信バッファに格納されたものを使用するか，編集バッファに格納されたものを使用するかを選択できます。

各パラメタに設定した値が不正の場合，TP1/NET/TCP/IP は MCF イベントを起動して論理端末を閉塞します。

(2)　UOC エラーリターン処理

UOC から DCMCF_UOC_MSG_NG でリターンした場合，MCF はメッセージログを出力し，障害通知イベント（CERREVT）を通知します。該当するメッセージは破棄します。

(3)　UOC パラメタ不正の場合の処理

UOC で設定したパラメタに不正があった場合，MCF はメッセージログを出力し，障害通知イベント（CERREVT）を通知して論理端末を閉塞します。該当するメッセージは破棄します。

(4)　OpenTP1 への組み込み方法

入力セグメント判定 UOC の組み込み方法と同じです。「5.1.1(5)　OpenTP1 への組み込み方法」を参照してください。

5.1.6　出力メッセージ編集 UOC インタフェース

出力メッセージ編集 UOC は，次に示す形式で呼び出します。

(1) 形式

ANSI C，C++の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
DCLONG  uoc_func(dcmcf_uoc_mout_n *parm)
```

K&R 版 C の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
DCLONG  uoc_func(parm)

dcmcf_uoc_mout_n *parm ;
```

(2) 説明

uoc_func（出力メッセージ編集 UOC）を呼び出すとき，MCF は次に示す所定のパラメタを parm に設定します。

(3) パラメタの内容

(a) dcmcf_uoc_mout_n の内容

```
typedef struct {
    DCLONG pro_kind;              …プロトコル種別
    char le_name[9];              …論理端末名称
    char reserve1[7];             …予備
    dcmcf_uocbuff_list_n *buflist_adr;
                                  …送信バッファリストアドレス
    dcmcf_uocbuff_list_n *ebuflist_adr;
                                  …編集バッファリストアドレス
    DCLONG output_no;             …メッセージ出力通番
    char msg_type;                …メッセージ種別
    char outputno_flag;           …メッセージ出力通番有効フラグ
    char resend_flag;             …再送メッセージフラグ
    char reserve2[1];             …予備
    char *pro_indv_ifa;           …MCF使用領域
    DCLONG rtn_detail;            …詳細リターンコード
    char reserve3[20];            …予備
} dcmcf_uoc_mout_n;
```

(b) dcmcf_uocbuff_list_n（バッファリスト），dcmcf_uocbufinf_n（バッファ情報）の内容

「5.1.4　入力メッセージ編集 UOC インタフェース」を参照してください。

(4) MCF が値を設定する項目

(a) dcmcf_uoc_mout_n

- pro_kind
  プロトコル種別として，次の値が設定されます。

  DCMCF_UOC_PRO_TCP
  　TCP/IP プロトコル
- le_name
  メッセージを出力する論理端末の名称が設定されます。
- buflist_adr
  送信用バッファリストのアドレスが設定されます。
- ebuflist_adr
  編集用バッファリストのアドレスが設定されます。
  メッセージ編集バッファが未定義の場合（コネクション定義（mcftalccn）の-e オプションを省略した場合），ebuflist_adr には NULL が設定されます。
- output_no
  メッセージ出力通番が設定されます。ただし，outputno_flag が DCMCF_UOC_OUTPUTNO_OK のときだけ有効です。
- msg_type
  メッセージ種別として，次の値が設定されます。

  'o'
  　応答メッセージ

  'n'
  　一般一方送信メッセージ

  'p'
  　優先一方送信メッセージ

  's'
  　同期送信メッセージ
- outputno_flag
  メッセージ出力通番の有効フラグとして，次のどちらかの値が設定されます。

  DCMCF_UOC_OUTPUTNO_OK
  　メッセージ出力通番を有効にします。

  DCMCF_UOC_OUTPUTNO_NG
  　メッセージ出力通番を無効とします。

- resend_flag
  再送メッセージフラグとして，次のどちらかの値が設定されます。
  'r'
  　再送メッセージです。
  'n'
  　再送メッセージではありません。
- pro_indv_ifa
  MCF で使用するパラメタです。

(b)　dcmcf_uocbuff_list_n（バッファリスト），dcmcf_uocbufinf_n（バッファ情報）

「5.1.4　入力メッセージ編集 UOC インタフェース」を参照してください。

(5)　ユーザが値を設定する項目

(a)　dcmcf_uoc_mout_n

- rtn_detail
  詳細リターンコードを設定します。
  このコードは，UOC が DCMCF_UOC_MSG_NG でリターンしたときに，MCF に渡されます。MCF は，詳細リターンコードをメッセージログファイルに出力します。
  詳細リターンコードは，-19999～-19000 の範囲で指定してください。

(b)　dcmcf_uocbuff_list_n（バッファリスト），dcmcf_uocbufinf_n（バッファ情報）

「5.1.4　入力メッセージ編集 UOC インタフェース」を参照してください。

(6)　リターン値

uoc_func()は次のコードでリターンしてください。

リターン値	意味
DCMCF_UOC_MSG_OK	正常リターン（編集バッファでスケジューリング）
DCMCF_UOC_MSG_OK_SND	正常リターン（送信バッファでスケジューリング）
DCMCF_UOC_MSG_NG	メッセージ編集エラー

(7)　パラメタとバッファの関係

UOC インタフェース用のパラメタとバッファの関係は，入力メッセージ編集 UOC と同じです。
「5.1.4(7)　パラメタとバッファの関係」を参照してください。

5.1.7　送信メッセージの通番編集

(1)　出力通番編集

送信メッセージの通番編集 UOC では，受け取った出力通番を基に，ユーザ独自の処理ができます。例えば，出力通番を使用して編集した送信メッセージを再送要求します。

送信メッセージの通番編集 UOC を起動する場合，メッセージ送信の関数で，出力通番を付けるように設定してください。UOC は，UAP が dc_mcf_send 関数，dc_mcf_reply 関数または dc_mcf_resend 関数を発行したときに，MCF によって起動されます。

(2)　OpenTP1 への組み込み方法

UAP のメイン関数の中に，UOC の関数アドレスを登録しておきます。UAP のメイン関数に登録する dc_mcf_regster 関数の形式を次に示します。

(a)　形式

ANSI C，C++の場合

```
#include <dcmcf.h>
int dc_mcf_regster(DCLONG flags, DCLONG (*uoc_addr)
                   (DCLONG flags, char *termname,
                   DCLONG sendno, DCLONG sendid,
                   DCLONG dataleng, char *senddata))
```

K&R 版 C の場合

```
#include <dcmcf.h>
int dc_mcf_regster(flags, uoc_addr)
DCLONG flags;
DCLONG (*uoc_addr)();
```

(b)　ユーザが値を設定する引数

- flags

  DCMCF_SEND_UOC を指定します。

- uoc_addr

  flags に対応する UOC のアドレスを指定します。

(c)　リターン値

リターン値	意味
DC_OK	正常に終了しました。
DCMCFER_INVALID_ARGS	引数の指定が間違っています。
DCMCFER_NOMEM	ローカルメモリが不足しました。

メイン関数への登録例を次に示します。

- MHP の場合

```
main()
{
extern DCLONG send_uoc();

dc_rpc_open();
dc_mcf_open();
dc_mcf_regster(DCMCF_SEND_UOC,send_uoc);
dc_mcf_mainloop();
dc_mcf_close();
dc_rpc_close();
}
```

- SPP の場合

```
main()
{
extern DCLONG send_uoc();

dc_rpc_open();
dc_mcf_open();
dc_mcf_regster(DCMCF_SEND_UOC,send_uoc);
dc_rpc_mainloop();
dc_mcf_close();
dc_rpc_close();
}
```

メイン関数，サービス関数，スタブ関数に UOC 関数をリンケージして UAP の実行形式プログラムを生成します。

5.1.8　送信メッセージの通番編集 UOC インタフェース

送信メッセージの通番編集 UOC は，次に示す形式で，send_uoc 関数として作成します。なお，UOC の関数名称はユーザの任意です。

(1)　形式

ANSI C，C++の場合

```
#include <dcmcf.h>
DCLONG send_uoc(DCLONG flags, char *termname, DCLONG sendno,
                DCLONG sendid, DCLONG dataleng, char *senddata)
```

K&R 版 C の場合

```
#include <dcmcf.h>
DCLONG send_uoc(flags, termname, sendno, sendid, dataleng,
              senddata)
DCLONG        flags;
```

```
char          *termname;
DCLONG        sendno;
DCLONG        sendid;
DCLONG        dataleng;
char          *senddata;
```

(2) MCF が値を設定する項目

- flags

  送信メッセージの通番編集 UOC がいつ呼ばれたかが設定されます。

  DCMCF_SEND_DML

  メッセージを送信する関数または命令文が呼び出されたときを意味します。

  DCMCF_RESEND_DML

  メッセージを再送する関数または命令文が呼び出されたときを意味します。

- termname

  送信先の論理端末名称が設定されます。

- sendno

  送信メッセージの出力通番が設定されます。

- sendid

  送信するメッセージ種別が設定されます。

  DCMCF_SEND_IO

  応答メッセージ

  DCMCF_SEND_PRIO

  優先の一方送信メッセージ

  DCMCF_SEND_NORM

  一般の一方送信メッセージ

- dataleng

  送信メッセージ長が設定されます。

- senddata

  セグメントの内容が設定されます。

(3) リターン値

リターン値	意味
DC_OK	正常に終了しました。

5.1.9　コネクション確立要求の判定

コネクション確立 UOC は，相手システムからコネクション確立要求を受信するたびに起動し，その確立要求を受け入れるかどうかを判定できます。

(1)　コネクション確立要求の判定

TP1/NET/TCP/IP はコネクション確立 UOC を呼び出すとき，コネクション確立要求を発行した相手システムのアドレス情報を通知します。コネクション確立 UOC は，通知された相手システムのアドレス情報を基に，コネクション確立要求を受け入れるかどうかを判定します。

コネクション確立要求を受け入れる場合は，TP1/NET/TCP/IP が状態通知イベント（COPNEVT）によってコネクションの確立を通知すると，MHP は相手システムとのデータ送受信をできるようになります。

コネクション確立要求を拒否する場合は，相手アドレス識別のために一時的に受け入れたコネクションを解放します。この場合，相手システムにはコネクション確立後，TP1/NET/TCP/IP から即座にコネクションが解放されたように見えます。また，相手システムから送信されたデータは破棄されます。

注意事項

相手アドレスチェック抑止機能を使用し，コネクション確立 UOC を使用しない場合，未確立コネクションがあれば必ずコネクション確立要求を受け入れます。この場合，TP1/NET/TCP/IP および UOC による相手アドレスチェックが行われないため，誤接続のおそれがあります。相手アドレスチェック抑止機能を使用する際は十分注意してください。

(2)　UOC エラーリターン処理

UOC から DCMTCP_UOC_CON_NG でリターンした場合，TP1/NET/TCP/IP はメッセージログを出力し，コネクションの確立を拒否します。

(3)　UOC パラメタ不正の場合の処理

UOC で設定したパラメタに不正があった場合，TP1/NET/TCP/IP はメッセージログを出力し，コネクションの確立を拒否します。

(4)　OpenTP1 への組み込み方法

入力セグメント判定 UOC の組み込み方法と同じです。「5.1.1(5)　OpenTP1 への組み込み方法」を参照してください。また，MCF メイン関数のコーディング例は「5.1.10(7)　MCF メイン関数のコーディング例」を参照してください。

5.1.10　コネクション確立 UOC インタフェース

コネクション確立 UOC は，次に示す形式で呼び出します。

(1) 形式

ANSI C，C++の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>
DCLONG con_uoc(dcmtcp_uoc_con_n *parm)
```

K&R 版 C の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>
DCLONG con_uoc(parm)

dcmtcp_uoc_con_n *parm;
```

(2) 説明

con_uoc（コネクション確立 UOC）を呼び出すとき，MCF は次に示す所定のパラメタを parm に設定します。

(3) パラメタの内容

(a) dcmtcp_uoc_con_n の内容

```
typedef struct {
    char  cn_name[9];            …コネクションID
    char  reserve[7];            …予備
    dcmtcp_cnuoc_oaddr *oaddr;   …相手システムのアドレス情報
    unsigned short  oportno;     …相手システムのポート番号
    unsigned short  iportno;     …自システムのポート番号
    DCLONG  connect_permit;      …コネクション確立要求受け入れ可否
    DCLONG  reject_reason;       …コネクション確立拒否理由
    DCLONG  rtn_detail;          …詳細リターンコード
} dcmtcp_uoc_con_n;
```

(b) dcmtcp_cnuoc_oaddr（相手システムのアドレス情報）の内容

```
typedef struct {
    unsigned short o_type;       …アドレス通知形式
    unsigned short o_len;        …アドレス長
    char  o_addr[256];           …相手システムのアドレス
} dcmtcp_cnuoc_oaddr;
```

(4) MCF が値を設定する項目

(a) dcmtcp_uoc_con_n

- cn_name
  確立要求を受け付けたコネクション ID が設定されます。
- oaddr
  確立要求を行った相手システムのアドレス情報として，次の値が設定されます。
- oportno
  確立要求を行った相手システムのポート番号が設定されます。
- iportno
  確立要求を受け付けた自システムのポート番号が設定されます。

(b) dcmtcp_cnuoc_oaddr（相手システムのアドレス情報）

- o_type
  相手システムアドレスの通知形式が設定されます。
  0x01　ドット 10 進数字（dotted-decimal 形式）表記
- o_len
  相手システムアドレスの格納長が設定されます。
- o_addr
  相手システムアドレスを格納した領域が設定されます。
  相手システムの IP アドレスはドット 10 進数字（dotted-decimal 形式）表記の文字列で設定されます（例："172.18.151.40"）。

(5) ユーザが値を設定する項目

(a) dcmtcp_uoc_con_n

- connect_permit
  コネクション確立要求を受け入れるかどうかを設定します。

  DCMTCP_UOC_CON_ACCEPT
  　コネクション確立要求を受け入れます。

  DCMTCP_UOC_CON_REJECT
  　コネクション確立要求を拒否します。
- reject_reason
  コネクション確立要求を拒否する場合の拒否理由を設定します。値は任意です。この拒否理由はログメッセージ（KFCA14840-I）に出力されます。

- rtn_detail

  コネクション確立 UOC が DCMTCP_UOC_CON_NG（UOC 異常終了）でリターンするときの詳細コードを設定します。この詳細リターンコードはログメッセージ（KFCA14808-E）に出力されます。値は-19999～-19000 の範囲内で設定します。TP1/NET/TCP/IP は，値の範囲をチェックしません。

(6) リターン値

con_uoc()は次のコードでリターンしてください。

リターン値	意味
DCMTCP_UOC_CON_OK	正常リターン（コネクション確立要求の受信）
DCMTCP_UOC_CON_NG	コネクション確立要求判定エラー

(7) MCF メイン関数のコーディング例

コネクション確立 UOC をメイン関数に登録する場合のコーディング例を次に示します。

ANSI C，C++の場合

```
#include <dcmtcp.h>
#include <dcmtcpu.h>

extern DCLONG con_uoc(dcmtcp_uoc_con_n *);/* コネクション確立UOC*/
extern dcmcf_uoc_t  dcmcf_uoctbl;     /* UOCテーブルextern宣言 */

int main()
{
     dcmcf_uoctbl.assctn                    /* コネクション確立UOC */
               = (dcmcf_uocfunc)con_uoc;    /* アドレス設定 */
     dc_mcf_svstart();
     return 0;
}
```

K&R 版 C の場合

```
#include <dcmtcp.h>

extern DCLONG con_uoc();                    /* コネクション確立UOC */
extern dcmcf_uoc_t dcmcf_uoctbl;      /* UOCテーブルextern宣言 */

main()
{
     dcmcf_uoctbl.assctn = con_uoc;         /* コネクション確立UOC */
                                            /* アドレス設定 */
     dc_mcf_svstart();
}
```

5.1.11　受信メッセージ判定

受信メッセージ判定 UOC は，メッセージ受信時に呼び出され，受信したメッセージの種別と送達確認メッセージの送信要否を決定します。受信メッセージ判定 UOC は，任意の相手システムとのメッセージ送達確認機能を使用する場合に必要となります。

(1)　受信メッセージの判定

TP1/NET/TCP/IP は受信メッセージ判定 UOC を呼び出す際に，受信したメッセージを通知します。受信メッセージ判定 UOC は，通知されたメッセージを基に，メッセージの種別と送達確認メッセージの送信要否を決定します。

(2)　UOC エラーリターン処理

UOC から DCTCP_UOC_MR_NG でリターンした場合，TP1/NET/TCP/IP はメッセージログを出力し，障害通知イベント（CERREVT）を通知します。該当するメッセージは破棄します。

(3)　UOC パラメタ不正の場合の処理

UOC で設定した値に不正があった場合，TP1/NET/TCP/IP はメッセージログを出力し，障害通知イベント（CERREVT）を通知します。該当するメッセージは破棄します。

(4)　OpenTP1 への組み込み方法

入力セグメント判定 UOC の組み込み方法と同じです。「5.1.1(5)　OpenTP1 への組み込み方法」を参照してください。また，MCF メイン関数のコーディング例は「5.1.12(7)　MCF メイン関数のコーディング例」を参照してください。

5.1.12　受信メッセージ判定 UOC インタフェース

受信メッセージ判定 UOC は，次に示す形式で呼び出します。なお，UOC の関数名称はユーザの任意です。

(1)　形式

ANSI C，C++の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>
DCLONG msgrep_uoc(dctcp_uoc_msgrep *parm)
```

K&R 版 C の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>
```

```
DCLONG    msgrep_uoc(parm)

dctcp_uoc_msgrep *parm;
```

(2) 説明

msgrep_uoc（受信メッセージ判定 UOC）を呼び出すとき，MCF は次に示す所定のパラメタを parm に設定します。

(3) パラメタの内容

(a) dctcp_uoc_msgrep の内容

```
typedef struct {
    DCLONG pro_kind;            …プロトコル種別
    char cn_name[9];            …コネクションID
    char reserve1[7];           …予備
    char le_name[9];            …論理端末名称
    char reserve2[7];           …予備
    DCLONG snd_status;          …送信状態
    char *rcv_data_adr;         …受信メッセージ格納アドレス
    DCLONG rcv_data_size;       …受信メッセージ長
    DCLONG msg_type;            …メッセージ種別
    DCLONG reason;              …理由コード
    DCLONG msg_reply;           …送達確認メッセージの送信要否
    char *snd_data_adr;         …送達確認メッセージ格納アドレス
    DCLONG snd_buff_size;       …送達確認メッセージ格納バッファサイズ
    DCLONG snd_data_size;       …送達確認メッセージ長
    DCLONG snd_cont;            …送信処理の継続可否
    DCLONG rtn_detail;          …詳細リターンコード
    char reserve3[16];          …予備
}dctcp_uoc_msgrep;
```

(4) MCF が値を設定する項目

(a) dctcp_uoc_msgrep

- pro_kind

  プロトコル種別として，次の値が設定されます。

  DCMCF_UOC_PRO_TCP

  TCP/IP プロトコル

- cn_name

  メッセージを受信したコネクション名が設定されます。

- le_name

  cn_name に対応する論理端末の名称が設定されます。

- snd_status

送信状態が設定されます。DCTCP_UOC_MR_ST_SND が設定されている場合，送受信メッセージの衝突が発生しています。

DCTCP_UOC_MR_ST_IDL

未送信

DCTCP_UOC_MR_ST_SND

送信処理中

- rcv_data_adr

受信メッセージを格納したバッファのアドレスが設定されます。

ユーザは，このパラメタを使用して受信メッセージを参照・編集できます。ただし，受信メッセージのメッセージ長は変更できません。

- rcv_data_size

受信メッセージ長が設定されます。

UOC でこのパラメタを変更しても無効となります。

- snd_data_adr

送達確認メッセージを格納するバッファのアドレスが設定されます。

ユーザは，このパラメタを使用して送達確認メッセージを編集します。

- snd_buff_size

送達確認メッセージを格納するバッファサイズが設定されます。

UOC でこのパラメタを変更しても無効となります。

(5) ユーザが値を設定する項目

(a) dctcp_uoc_msgrep

- msg_type

受信したメッセージの種別を設定します。

DCTCP_UOC_MR_MT_INFO

一方送信メッセージ

DCTCP_UOC_MR_MT_ACK

送達確認メッセージ

DCTCP_UOC_MR_MT_IGN

破棄メッセージ

DCTCP_UOC_MR_MT_CLS

コネクション解放

- reason

破棄メッセージおよびコネクション解放を指定するときの理由を設定します。この理由コードはログメッセージ（KFCA14849-I，KFCA14850-I）に出力されます。値はユーザの任意に設定します。msg_type が，DCTCP_UOC_MR_MT_IGN，DCTCP_UOC_MR_MT_CLS の場合だけ有効です。

- msg_reply

  送達確認メッセージの送信要否を設定します。msg_type が DCTCP_UOC_MR_MT_INFO の場合だけ有効です。

  DCTCP_UOC_MR_REP_ON

  送達確認メッセージの送信要

  DCTCP_UOC_MR_REP_OFF

  送達確認メッセージの送信不要

- snd_data_size

  送達確認メッセージ長を設定します。msg_type が DCTCP_UOC_MR_MT_INFO であり，かつ msg_reply が DCTCP_UOC_MR_REP_ON の場合だけ有効です。

- snd_cont

  仕掛り中の送信処理を継続するかどうかを設定します。snd_status が DCTCP_UOC_MR_ST_SND の場合で，msg_type が DCTCP_UOC_MR_MT_INFO または DCTCP_UOC_MR_MT_IGN のときだけ有効です。

  DCTCP_UOC_MR_SC_CONT

  送信処理を継続します。

  DCTCP_UOC_MR_SC_RSND

  送信処理を中断します。送信中のメッセージは，受信処理および送達確認メッセージの送信処理完了後，再送します。

  DCTCP_UOC_MR_SC_STOP

  論理端末を閉塞し，送信処理を停止します。

- rtn_detail

  UOC エラーリターン時の詳細リターンコードを設定します。

(6) リターン値

msgrep_uoc()は次のコードでリターンしてください。

リターン値	意味
DCTCP_UOC_MR_OK	正常リターン
DCTCP_UOC_MR_NG	エラーリターン

(7) MCF メイン関数のコーディング例

受信メッセージ判定 UOC をメイン関数に登録する場合のコーディング例を次に示します。

ANSI C，C++の場合

```
#include  <dcmtcp.h>
#include  <dcmtcpu.h>

extern DCLONG   msgrep_uoc(dctcp_uoc_msgrep *);  /* 受信メッセージ判定UOC */
extern dcmcf_uoc_t dcmcf_uoctbl;   /* UOCテーブルextern宣言 */

int main()
{
    dcmcf_uoctbl.msgrep            /* 受信メッセージ判定 UOC */
   = (dcmcf_uocfunc)msgrep_uoc;    /*           アドレス設定  */
    dc_mcf_svstart();
    return 0;
}
```

K&R 版 C の場合

```
#include  <dcmtcp.h>
extern DCLONG        msgrep_uoc(); /* 受信メッセージ判定 UOC    */
extern dcmcf_uoc_t dcmcf_uoctbl;   /* UOC テーブルextern宣言   */

main()
{
    dcmcf_uoctbl.msgrep = msgrep_uoc; /* 受信メッセージ判定 UOC  */
                                  /*         アドレス設定    */
    dc_mcf_svstart();
}
```

5.1.13　受信メッセージの保留判定

受信メッセージ保留判定 UOC は，メッセージ受信時に呼び出され，受信したメッセージ単位で保留するかどうかを決定します。この UOC は，同期型メッセージの受信時に使用できます。

(1)　受信メッセージの保留判定

TP1/NET/TCP/IP は，受信メッセージ保留判定 UOC を呼び出すときに，受信したメッセージをユーザに通知します。受信メッセージ保留判定 UOC は，通知されたメッセージを基に，受信したメッセージを保留するかどうか決定します。

(2)　UOC エラーリターン処理

UOC から DCTCP_UOC_MH_NG でリターンした場合，TP1/NET/TCP/IP はメッセージログを出力し，障害通知イベント（CERREVT）を通知します。該当するメッセージは破棄します。

(3)　UOC パラメタ不正の場合の処理

UOC で設定した値に不正があった場合，TP1/NET/TCP/IP はメッセージログを出力し，障害通知イベント（CERREVT）を通知します。該当するメッセージは破棄します。

(4) OpenTP1 への組み込み方法

入力セグメント判定 UOC の組み込み方法と同じです。「5.1.1(5) OpenTP1 への組み込み方法」を参照してください。また，MCF メイン関数のコーディング例は「5.1.14(7) MCF メイン関数のコーディング例」を参照してください。

5.1.14 受信メッセージの保留判定 UOC インタフェース

受信メッセージ保留判定 UOC は，次に示す形式で呼び出します。

(1) 形式

ANSI C，C++の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>
DCLONG msghld_uoc(dctcp_uoc_msghld *parm)
```

K&R 版 C の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
#include <dcmtcpu.h>
DCLONG    msghld_uoc(parm)

dctcp_uoc_msghld *parm;
```

(2) 説明

msghld_uoc（受信メッセージ保留判定 UOC）を呼び出すとき，MCF は次に示す所定のパラメタを parm に設定します。

(3) パラメタの内容

(a) dctcp_uoc_msghld の内容

```
typedef struct {
    DCLONG pro_kind;          …プロトコル種別
    char cn_name[9];          …コネクションID
    char reserve1[7];         …予備
    char le_name[9];          …論理端末名称
    char reserve2[7];         …予備
    char reserve3[4];         …予備
    char *rcv_data_adr;       …受信メッセージ格納アドレス
    DCLONG rcv_data_size;     …受信メッセージ長
    DCLONG msg_hld;           …受信メッセージの保留要否
    DCLONG rtn_detail;        …詳細リターンコード
```

```
    char reserve4[20];          …予備
}dctcp_uoc_msghld;
```

(4) MCF が値を設定する項目

(a) dctcp_uoc_msghld

- pro_kind

  プロトコル種別として，次の値が設定されます。

  DCMCF_UOC_PRO_TCP

  TCP/IP プロトコル

- cn_name

  メッセージを受信したコネクション ID が設定されます。

- le_name

  cn_name に対応する論理端末の名称が設定されます。

- rcv_data_adr

  受信メッセージを格納したバッファのアドレスが設定されます。

  ユーザは，このパラメタを使用して受信メッセージを参照・編集できます。ただし，受信メッセージのメッセージ長は変更できません。

- rcv_data_size

  受信メッセージ長が設定されます。

  UOC でこのパラメタを変更しても無効となります。

(5) ユーザが値を設定する項目

(a) dctcp_uoc_msghld

- msg_hld

  受信メッセージを保留するかどうかを設定します。

  DCTCP_UOC_MH_HLD_ON

  受信したメッセージを保留します。

  保留したメッセージは同期型メッセージの受信関数を使用して受信してください。

  DCTCP_UOC_MH_HLD_OFF

  受信したメッセージを保留しません。

  TP1/NET/TCP/IP は受信メッセージを入力キューに書き込み，MHP を起動します。

- rtn_detail

  UOC エラーリターン時の詳細リターンコードを設定します。

(6) リターン値

msghld_uoc()は次のコードでリターンしてください。

リターン値	意味
DCTCP_UOC_MH_OK	正常リターン
DCTCP_UOC_MH_NG	エラーリターン

(7) MCF メイン関数のコーディング例

受信メッセージ保留判定 UOC をメイン関数に登録する場合のコーディング例を次に示します。

ANSI C，C++の場合

```
#include  <dcmtcp.h>
#include  <dcmtcpu.h>

extern DCLONG   msghld_uoc(dctcp_uoc_msghld *);
                                        /* 受信メッセージ保留判定UOC */
extern dcmcf_uoc_t dcmcf_uoctbl;
                                           /* UOCテーブルextern宣言 */

int main()
{
    dcmcf_uoctbl.msghld                 /* 受信メッセージ保留判定UOC */
                  = (dcmcf_uocfunc)msghld_uoc;     /*アドレス設定 */
    dc_mcf_svstart();
    return 0;

}
```

K&R 版 C の場合

```
#include  <dcmtcp.h>
extern DCLONG        msghld_uoc();     /* 受信メッセージ保留判定UOC */
extern dcmcf_uoc_t dcmcf_uoctbl;         /* UOC テーブルextern宣言*/

main()
{
    dcmcf_uoctbl.msghld               /* 受信メッセージ保留判定UOC */
                  = msghld_uoc;                    /*アドレス設定 */
    dc_mcf_svstart();
}
```

5.1.15 UOC 作成上の注意事項

UOC 作成上の注意事項を次に示します。

(1) UOC の構造

UOC で使用するローカル変数のサイズの合計は，各 UOC で 1024 バイト以内になるよう作成してください。また，UOC の中で関数の再帰呼び出しはしないでください。

(2) UOC で使用できる関数

UOC を作成する場合，UOC では次に示す関数だけが使用できます。ほかの関数を使用した場合，正常に動作しないことがあるためご注意ください。

- メモリ操作をする関数
  - データ領域管理（例：malloc，free）
  - 共有メモリ管理関数（例：shmctl，shmget，shmop）
  - メモリ操作（例：memcpy）
  - 文字列操作（例：strcpy）
- 時間取得関数（例：time，gettimeofday）

(3) UOC の異常処理

TP1/NET/TCP/IP の UOC で異常を検知した場合，MCF の所定のリターンコードを使用して MCF に異常の発生を通知してください。UOC でプロセス終了となるシグナル，または abort()を発行すると，MCF が異常終了します。

(4) UOC の実行タイミング

MCF が起動する UOC の実行タイミングは，OpenTP1 システム，および UAP の開始，終了シーケンスと同期しない場合があります。UAP より先に UOC が実行されたり，UAP がすべて終了してから UOC が呼び出されたりしてもよいように作ってください。

5.2 MCF イベントインタフェース

TP1/NET/TCP/IP でメッセージ送受信をすると，OpenTP1 の各種システム情報が MHP に通知されます。これを **MCF イベント**といいます。メッセージ送受信処理でエラーや障害が発生した場合，システム内で何が起こっているのかが MCF イベントの内容でわかります。MCF イベントに対応する MHP を **MCF イベント処理用 MHP** といいます。

MCF イベントは入力キューに渡されて，MCF イベント処理用 MHP で回復処理をします。なお，MCF イベントの発生時は入力メッセージ編集 UOC は呼び出しません。詳細についてはマニュアル「OpenTP1 プログラム作成の手引」を参照してください。

5.2.1 MCF イベントの種類

TP1/NET/TCP/IP が通知する MCF イベントの種類を次の表に示します。

表 5-1 TP1/NET/TCP/IP が通知する MCF イベントの種類

MCF イベント名	MCF イベントコード	発生した原因	MCF イベント処理用 MHP での処理の例
不正アプリケーション名検出通知イベント	ERREVT1	メッセージのアプリケーション名が MCF アプリケーション定義にありません。	該当するアプリケーションがないことを報告します。
メッセージ廃棄通知イベント	ERREVT2	次の理由で，受信メッセージを破棄しました。 • 入力キューに障害が発生しました。 • 入力メッセージ最大格納数を超過しました。 • 動的共用メモリが不足しました。 • キューファイルが満杯になりました。 • MHP のサービス，サービスグループ，アプリケーションが閉塞しています。 • スケジュール閉塞されているサービスグループの入力キューに未処理受信メッセージが残った状態で，OpenTP1 を正常終了または計画停止 A で終了しました。 • MHP のサービスグループまたはアプリケーションがセキュア状態です。 • MHP で呼び出す receive 関数にセグメントを渡す前に，MHP が異常終了しました。 • アプリケーション名に相当する MHP のサービスがありません。 • ユーザサーバ未起動などによって，MHP の起動に失敗しました。	メッセージを廃棄したことを報告します。

MCF イベント名	MCF イベントコード	発生した原因	MCF イベント処理用 MHP での処理の例
メッセージ廃棄通知イベント	ERREVT2	• DBMS の障害などによって，トランザクションの開始に失敗しました。	メッセージを廃棄したことを報告します。
UAP 異常終了通知イベント	ERREVT3	MHP に受信メッセージを渡したあとに MHP の異常終了※が発生しました。	UAP 異常終了時の対処障害メッセージを送信します。
タイマ起動メッセージ廃棄通知イベント	ERREVT4	アプリケーションのタイマ起動時に障害が発生しました。	メッセージを廃棄したことを報告します。
未処理送信メッセージ廃棄通知イベント	ERREVTA	次の理由で，未処理送信メッセージを破棄しました。 • MCF の正常終了処理時に，未処理送信メッセージの滞留時間監視の時間切れ（タイムアウト）が発生しました。 • 運用コマンド（mcftdlqle）の入力，または API（dc_mcf_tdlqle 関数もしくは CBLDCMCF('TDLQLE△△')）の発行によって，出力キューが削除されました。 • 閉塞されている論理端末の出力キューに未処理送信メッセージが残った状態で，dcstop コマンドが実行されました。	未処理送信メッセージを廃棄したことを報告します。
送信完了通知イベント	SCMPEVT	メッセージの送信を正常に完了しました。	送信の完了を確認して，任意の処理ができます。
障害通知イベント	CERREVT	コネクション障害，または論理端末障害が発生しました。	コネクション，または論理端末に障害が発生したことを報告します。
状態通知イベント	COPNEVT	コネクションが確立しました。	コネクションが確立したことを報告します。
	CCLSEVT	コネクションが正常に解放されました。	コネクションが解放されたことを報告します。
受信メッセージ保留通知イベント	RHLDEVT	受信したメッセージを保留しました。	同期型メッセージの受信関数を使用してメッセージを受信します。
送受信メッセージ廃棄通知イベント	MDELEVT	次の理由で，受信メッセージまたは送信メッセージを破棄しました。 • 任意の相手システムとのメッセージ送達確認を使用している（コネクション定義（mcftalcle -u）の delichk オペランドに use を指定）場合に，受信メッセージ判定 UOC で受信したメッセージの種別に破棄メッセージを指定しました。 • コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指	メッセージを廃棄したことを報告します。

MCF イベント名	MCF イベントコード	発生した原因	MCF イベント処理用 MHP での処理の例
送受信メッセージ廃棄通知イベント	MDELEVT	定）場合に，MHP でメッセージ受信後にコネクションが再確立されました。 • 問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信機能を使用している（コネクション定義（mcftalccn -l）の replymsg オペランドに yes を指定）場合に，問い合わせ応答中の論理端末がメッセージを受信しました。	メッセージを廃棄したことを報告します。

注※

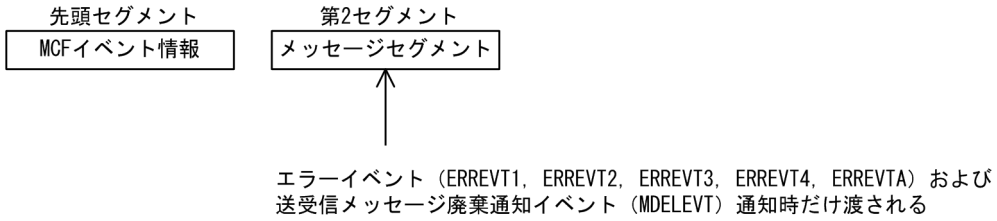
アプリケーション属性定義（mcfaalcap -g）の recvmsg オペランドに r を指定した場合，または dc_mcf_rollback の action に DCMCFRTRY または DCMCFRRTN を指定した場合は除きます。

5.2.2 MCF イベント通知時のセグメント構成

MCF イベントを MHP に通知する場合，先頭セグメントに MCF イベント情報を設定します。エラーイベント（ERREVT1，ERREVT2，ERREVT3，ERREVT4，ERREVTA）および送受信メッセージ廃棄通知イベント（MDELEVT）の場合は，第 2 セグメントに処理できなかったメッセージセグメントを設定します。

MCF イベント通知時のセグメント構成を次の図に示します。

図 5-5 MCF イベント通知時のセグメント構成



MCF イベントは，UAP を作成した言語によって，UAP に通知されるデータの形式が異なります。

エラーイベントの場合はバッファ形式 1 とバッファ形式 2 で先頭の内容が異なります。このため，それ以降の項目の位置にずれがあります。「5.2.4 MCF イベント情報の形式（COBOL 言語）」のエラーイベントの表では ERREVT1，ERREVT2，ERREVT3，および ERREVTA のバッファ形式別に位置（バイト）を分けて説明しています。なお，ERREVT4 については，マニュアル「OpenTP1 プログラム作成リファレンス」の該当する言語編を参照してください。

5.2.3 MCF イベント情報の形式（C 言語）

MCF イベント情報は構造体で，MCF イベント処理用 MHP に渡されます。MHP に渡される構造体の形式は，MCF イベントの種類によって異なります。ただし，MCF イベント情報の先頭部分の形式は，各イベントに共通です。

エラーイベント（ERREVT1，ERREVT2，ERREVT3，ERREVT4，ERREVTA）および送信完了通知イベント（SCMPEVT）で使用する構造体は，<dcmcf.h>で定義してあります。なお，dc_mcf_evtheaderは，<dcmcf.h>で定義されています。<dcmtcpu.h>の前に<dcmcf.h>を取り込んでおいてください。

MCF イベントとして設定される項目を次に示します。

(1) MCF イベントの共通ヘッダ

(a) 形式

```
struct dc_mcf_evtheader {
  char mcfevt_name[9] ; …  MCFイベントコード
  char le_name[16] ;    …  入力元論理端末名称
                           (ERREVT1, ERREVT2, ERREVT3, CERREVT,
                            COPNEVT, CCLSEVT, RHLDEVT, MDELEVTの場合)
                           出力先論理端末名称
                           (ERREVTA, SCMPEVTの場合)
  char cn_name[9] ;     …  コネクション名
  unsigned char format_kind;
                        …  MCF使用領域
  char reserve01;       …  予備
  DCLONG time ;         …  メッセージ入力時刻
};
```

(b) MCF イベントとして設定される項目

- le_name

  ERREVT1，ERREVT2，ERREVT3，または RHLDEVT では，メッセージを入力した論理端末名称が設定されます。

  ただし，ERREVT2 または ERREVT3 で，次に示す場合は，'*'が設定されます。

  - SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合
  - 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合

  ERREVTA，SCMPEVT では，メッセージを出力する論理端末名称が設定されます。

  CERREVT では，障害が発生した論理端末名称，または障害が発生したコネクションに対応する論理端末名称が設定されます。

  COPNEVT または CCLSEVT では，確立または解放したコネクションに対応する論理端末名称が設定されます。

MDELEVT では，破棄したメッセージを入力した論理端末名称，または破棄したメッセージを出力しようとした論理端末名称が設定されます。

- cn_name

  コネクション名が設定されます。

  ただし，ERREVT2 または ERREVT3 で，次に示す場合は，'*'が設定されます。

  - SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合
  - 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合

- time

  メッセージを入力した時刻が，1970 年 1 月 1 日 0 時 0 分 0 秒からの通算の秒数で設定されます。

(2) ERREVT1

(a) 形式

```
struct dc_mcf_evt1_type {
  struct dc_mcf_evtheader  evtheader ;
                        … MCFイベント共通ヘッダ
  char reserve01[12] ;  … 予備
  char reserve02[10] ;  … 予備
  char reserve03[2] ;   … 予備
  char ap_name[10] ;    … アプリケーション名
                          （メッセージに対応する
                           アプリケーション名）
  char reserve04[2] ;   … 予備
};
```

(b) MCF イベントとして設定される項目

- ap_name

  次に示すどちらかが設定されます。

  - 形式不正の場合…不正となったアプリケーション名
  - 定義されていない場合…定義されていないアプリケーション名

  アプリケーション名は，MHP から送信されたメッセージの場合に設定されます。ただし，MHP 以外から送信された場合はヌル文字が設定されます。

(3) ERREVT2

(a) 形式

```
struct dc_mcf_evt2_type {
  struct dc_mcf_evtheader evtheader ;
                        … MCFイベント共通ヘッダ
  char reserve01[12] ;  … 予備
```

```
  char reserve02[10] ;  … 予備
  char reserve03[2] ;   … 予備
  char ap_name[10] ;    … アプリケーション名
                          （メッセージに対応する
                           アプリケーション名）
  short reason_code ;   … 理由コード
};
```

(b) MCF イベントとして設定される項目

- ap_name

  エラーになった UAP のアプリケーション名が設定されます。

  アプリケーション名は，MHP から送信されたメッセージの場合に設定されます。ただし，MHP 以外から送信された場合はヌル文字が設定されます。

- reason_code

  ERREVT2 の理由コードが設定されます。理由コードの詳細については，「付録 J 理由コード一覧」を参照してください。

(4) ERREVT3

(a) 形式

```
struct dc_mcf_evt3_type {
  struct dc_mcf_evtheader evtheader ;
                              … MCFイベント共通ヘッダ
  char reserve01[12] ;        … 予備
  char map_name[10] ;         … MCF使用領域
  char reserve03[2] ;         … 予備
  char ap_name[10] ;          … アプリケーション名
                                （異常が発生したメッセージの
                                 アプリケーション名）
  char reserve04[2] ;         … 予備
  char service_name[32] ;     … サービス名
  char serv_grp_name[32] ;    … サービスグループ名
  char bid[36] ;              … トランザクションブランチ
                                 ID領域
} ;
```

(b) MCF イベントとして設定される項目

- ap_name

  異常が発生した MHP のアプリケーション名が設定されます。

  アプリケーション名は，MHP から送信されたメッセージの場合に設定されます。ただし，MHP 以外から送信された場合はヌル文字が設定されます。

- service_name

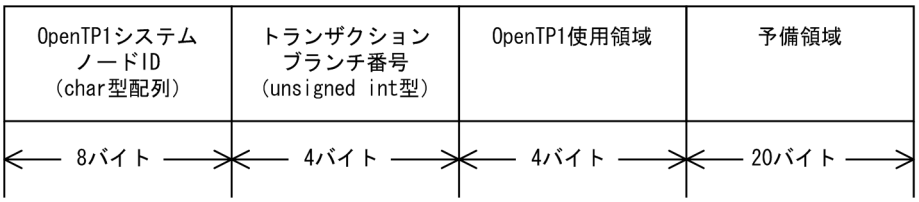
  異常が発生した MHP のアプリケーション名に対応するサービス名が設定されます。

- serv_grp_name

異常が発生したMHPのサービスが属するサービスグループ名が設定されます。

- bid

  トランザクションブランチIDが次の形式で設定されます。



(5) ERREVTA

(a) 形式

```
struct dc_mcf_evta_type {
  struct dc_mcf_evtheader  evtheader ;
                           … MCFイベント共通ヘッダ
  char reserve01[12] ;     … 予備
  char reserve02[10] ;     … 予備
  char reserve03[2] ;      … 予備
  char ap_name[10] ;       … アプリケーション名
                            （正常終了したメッセージの
                              アプリケーション名）
  char reserve04[2] ;      … 予備
  char reserve05[32] ;     … 予備
  char reserve06[32] ;     … 予備
  DCLONG user_leng ;       … 他プロトコルの場合の使用領域
  char user_data[16] ;     … 他プロトコルの場合の使用領域
  char reserve07[16] ;     … 予備
} ;
```

(b) MCFイベントとして設定される項目

- ap_name

  正常終了したメッセージのアプリケーション名が設定されます。

  アプリケーション名は，MHPから送信されたメッセージの場合に設定されます。ただし，MHP以外から送信された場合はヌル文字が設定されます。

(6) SCMPEVT

(a) 形式

```
struct dc_mcf_scmpevt_type {
  struct dc_mcf_evtheader  evtheader ; … MCFイベント共通ヘッダ
  DCLONG output_no ;                   … 出力通番
  char map_name[9] ;                   … MCFが使用
  char msg_type ;                      … メッセージ種別
  char reason_code ;                   … 起動理由
  char reserve01[5] ;                  … 予備
```

```
  DCLONG user_leng ;                    … MCF使用領域
  char pro_indv_inf[16] ;               … MCF使用領域
  char user_data[16] ;                  … MCF使用領域
} ;
```

(b) MCF イベントとして設定される項目

- output_no

  出力通番が設定されます。出力通番がない場合は，「-1」が設定されます。

- msg_type

  メッセージ種別として，次のどちらかの値が設定されます。

  n：一般一方送信メッセージ

  p：優先一方送信メッセージ

- reason_code

  起動理由として次の値が設定されます。

  △：メッセージが正常に送信されました。

(7) CERREVT

(a) 形式

```
typedef struct {
  struct dc_mcf_evtheader  header ;
                           … MCFイベント共通ヘッダ
  DCLONG err_fact ;        … 障害要因コード（4バイト）
                             0x30：コネクション障害発生
                             0x31：論理端末障害発生
  DCLONG err_reason1 ;     … 理由コード1（4バイト）
  DCLONG err_reason2 ;     … 理由コード2（4バイト）
  char group_name[16];     … MCF使用領域
  char reserve1[32] ;      … 予備
} dcmtcp_cerrevt ;
```

(b) MCF イベントとして設定される項目

- err_reason1，err_reason2

  CERREVT の理由コード 1，理由コード 2 が設定されます。「付録 J　理由コード一覧」を参照してください。

(8) COPNEVT，CCLSEVT

(a) 形式

```
typedef struct {
  struct dc_mcf_evtheader  header ;
                           … MCFイベント共通ヘッダ
```

```
  char group_name[16];      … MCF使用領域
  char reserve1[40] ;       … 予備
} dcmtcp_statevt ;
```

(9) RHLDEVT

(a) 形式

```
typedef struct {
  struct dc_mcf_evtheader  header ;
                          … MCFイベント共通ヘッダ
  char group_name[16];    … MCF使用領域
  char reserve1[44] ;     … 予備
} dcmtcp_rhldevt ;
```

(10) MDELEVT

(a) 形式

```
typedef struct {
  struct dc_mcf_evtheader  header ;
                          … MCFイベント共通ヘッダ
  DCLONG del_reason;      … 理由コード
  char group_name[16];    … MCF使用領域
  char reserve1[40] ;     … 予備
} dcmtcp_mdelevt ;
```

(b) MCF イベントとして設定される項目

- del_reason

  MDELEVT の理由コードが設定されます。理由コードの詳細については，「付録 J　理由コード一覧」を参照してください。

5.2.4 MCF イベント情報の形式（COBOL 言語）

COBOL 言語の場合はセグメントの並びとして渡されます。

COBOL 言語の UAP の場合の MCF イベント情報の内容を，以降の表に示します。

表 5-2　COBOL 言語の MCF イベント情報の内容（ERREVT1）

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備（形式 1 のときだけ）	0	−	2	−	−
予備（形式 1 のときだけ）	2	−	2	−	−

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
エラーイベントコード	4	0	3	英数字	'ERR'が設定されます。
	7	3	3	−	−
	10	6	2	英数字	ERREVT1 を示す'1△'が設定されます。
入力元論理端末名称	12	8	8	英数字	メッセージを入力した論理端末名称です。
予備	20	16	20	−	−
アプリケーション名	40	36	8	英数字	次に示すどれかが設定されます。 • 形式不正となったアプリケーション名 • 定義されていないアプリケーション名
予備	48	44	8	−	−
予備	56	52	8	−	−
予備	64	60	8	−	−
コネクション名	72	68	8	英数字	コネクション名です。
予備	80	76	16	−	−
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。YYYYMMDD の形式です。 YYYY：西暦の年 MM：月 DD：日
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
予備	112	108	16	−	−

（凡例）

−：該当しません，または使用されません。

表 5-3　COBOL 言語の MCF イベント情報の内容（ERREVT2）

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備（形式 1 のときだけ）	0	−	2	−	−

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備（形式 1 のときだけ）	2	−	2	−	−
エラーイベントコード	4	0	3	英数字	'ERR'が設定されます。
	7	3	3	−	−
	10	6	2	英数字	ERREVT2 を示す'2△'が設定されます。
入力元論理端末名称	12	8	8	英数字	メッセージを入力した論理端末名称です。 次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	20	16	20	−	−
アプリケーション名	40	36	8	英数字	エラーになった UAP のアプリケーション名です。
予備	48	44	8	−	−
予備	56	52	8	−	−
予備	64	60	8	−	−
コネクション名	72	68	8	英数字	コネクション名です。 次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	80	76	16	−	−
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。YYYYMMDD の形式です。 YYYY：西暦の年 MM：月 DD：日

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
理由コード※	112	108	4	外部 10 進数字	理由コードが設定されます。
予備	116	112	12	−	−

（凡例）

−：該当しません，または使用されません。

注※

理由コードの詳細については，「付録 J　理由コード一覧」を参照してください。

表 5-4　COBOL 言語の MCF イベント情報の内容（ERREVT3）

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備（形式 1 のときだけ）	0	−	2	−	−
予備（形式 1 のときだけ）	2	−	2	−	−
エラーイベントコード	4	0	3	英数字	'ERR'が設定されます。
	7	3	3	−	−
	10	6	2	英数字	ERREVT3 を示す'3△'が設定されます。
入力元論理端末名称	12	8	8	英数字	メッセージを入力した論理端末名称です。 次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	20	16	20	−	−
予備	40	36	8	−	−
マップ名	48	44	8	−	MCF が使用します。

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
アプリケーション名	56	52	8	英数字	異常が発生したメッセージのアプリケーション名です。
予備	64	60	8	−	−
コネクション名	72	68	8	英数字	コネクション名です。 次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	80	76	16	−	−
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。YYYYMMDD の形式です。 YYYY：西暦の年 MM：月 DD：日
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
予備	112	108	16	−	−
サービス名	128	124	31	英数字	異常が発生した UAP のアプリケーション名に対応するサービス名です。
予備	159	155	1	−	−
サービスグループ名	160	156	31	英数字	異常が発生した UAP のサービスグループ名です。
予備	191	187	1	−	−
トランザクションブランチ ID（BID）	192	188	36	英数字	異常が発生したトランザクションの BID です。 トランザクションブランチ ID の形式については，表 5-5 を参照してください。
予備	228	224	28	−	−

（凡例）

－：該当しません，または使用されません。

表 5-5　トランザクションブランチ ID の形式

項目	位置（バイト）	長さ（バイト）	属性
OpenTP1 システムノード ID	0	8	英数字
トランザクションブランチ番号	8	4	2 進数字
OpenTP1 使用領域	12	4	－
予備	16	20	－

表 5-6　COBOL 言語の MCF イベント情報の内容（ERREVTA）

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備（形式 1 のときだけ）	0	－	2	－	－
予備（形式 1 のときだけ）	2	－	2	－	－
エラーイベントコード	4	0	3	英数字	'ERR'が設定されます。
	7	3	3	－	－
	10	6	2	英数字	ERREVTA を示す'A△'が設定されます。
出力先論理端末名称	12	8	8	英数字	メッセージを出力する論理端末名称です。
予備	20	16	20	－	－
予備	40	36	8	－	－
マップ名	48	44	8	－	MCF が使用します。
アプリケーション名	56	52	8	英数字	正常終了したメッセージのアプリケーション名です。MHP から送信されたメッセージの場合設定されます。MHP 以外から送信された場合は空白が設定されます。
予備	64	60	8	－	－
コネクション名	72	68	8	英数字	コネクション名です。
予備	80	76	16	－	－
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。YYYYMMDD の形式です。 YYYY：西暦の年

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
メッセージが入力された日付	96	92	8	外部 10 進数字	MM：月 DD：日
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
予備	112	108	16	－	－
予備	128	124	31	－	－
予備	159	155	1	－	－
予備	160	156	31	－	－
予備	191	187	1	－	－
予備	192	188	36	－	－
予備	228	224	28	－	－

（凡例）

－：該当しません，または使用されません。

表 5-7　COBOL 言語の MCF イベント情報の内容（SCMPEVT）

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
長さ （形式 1 のときだけ）	0	－	2	－	MCF が使用します。
予備 （形式 1 のときだけ）	2	－	2	－	－
イベントコード	4	0	8	英数字	「SCMPEVT」が設定されます。
出力先論理端末名称	12	8	8	英数字	一方送信メッセージの送信が完了した論理端末名称が設定されます。
予備	20	16	8	－	－
コネクション名	28	24	8	英数字	一方送信メッセージの送信が完了したコネクション名が設定されます。

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
一方送信メッセージの送信が完了した日付	36	32	8	外部 10 進	一方送信メッセージの送信が完了した日付が設定されます。 YYYYMMDD の形式です。 YYYY：西暦の年 MM：月 DD：日
一方送信メッセージの送信が完了した時刻	44	40	8	外部 10 進	一方送信メッセージの送信が完了した時刻が設定されます。HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
出力通番	52	48	4	2 進数	出力通番が設定されます。 出力通番がない場合は，$(FFFFFFFF)_{16}$ が設定されます。
出力マップ名	56	52	8	英数字	MCF が使用します。
メッセージ種別	64	60	1	英数字	メッセージ種別として次の値が設定されます。 n：一般一方送信メッセージ p：優先一方送信メッセージ
起動理由	65	61	1	英数字	起動理由として次の値が設定されます。 △：送信が正常に完了しました。
予備	66	62	9	－	－
プロトコル個別情報	75	71	16	英数字	MCF が使用します。
予備	91	87	19	－	－

（凡例）

－：該当しません，または使用されません。

表 5-8　COBOL 言語の MCF イベント情報の内容（CERREVT）

項目	位置（バイト）	長さ（バイト）	属性	内容
イベントコード	0	8	英数字	イベントコード「CERREVT」が設定されます。
入力元論理端末名称	8	8	英数字	障害が発生した論理端末名称，または障害が発生したコネクション名に対応する論理端末名称が設定されます。
予備	16	8	－	－
入力元コネクション名	24	8	英数字	コネクション名が設定されます。
メッセージ入力日付	32	8	外部 10 進数字	CERREVT を入力した日付です。
メッセージ入力時刻	40	8	外部 10 進数字	CERREVT を入力した時刻です。
障害要因コード	48	4	2 進数字	障害要因コードが設定されます。 $(00000030)_{16}$：コネクション障害 $(00000031)_{16}$：論理端末障害
理由コード 1※	52	4	2 進数字	理由コード 1 が設定されます。
理由コード 2※	56	4	2 進数字	理由コード 2 が設定されます。
予備	60	48	－	－

（凡例）

－：該当しません，または使用されません。

注※

理由コード 1，および理由コード 2 は，「付録 J　理由コード一覧」を参照してください。

表 5-9　COBOL 言語の MCF イベント情報の内容（COPNEVT，CCLSEVT）

項目	位置（バイト）	長さ（バイト）	属性	内容
イベントコード	0	8	英数字	イベントコード「COPNEVT」，または「CCLSEVT」が設定されます。
入力元論理端末名称	8	8	英数字	確立または解放したコネクション名に対応する論理端末名称が設定されます。
予備	16	8	－	－
入力元コネクション名	24	8	英数字	コネクション名が設定されます。
メッセージ入力日付	32	8	外部 10 進数字	COPNEVT，CCLSEVT を入力した日付です。

項目	位置（バイト）	長さ（バイト）	属性	内容
メッセージ入力時刻	40	8	外部 10 進数字	COPNEVT，CCLSEVT を入力した時刻です。
予備	48	60	－	－

（凡例）

－：該当しません，または使用されません。

表 5-10　COBOL 言語の MCF イベント情報の内容（RHLDEVT）

項目	位置（バイト）	長さ（バイト）	属性	内容
イベントコード	0	8	英数字	イベントコード「RHLDEVT」が設定されます。
入力元論理端末名称	8	8	英数字	受信メッセージを保留した論理端末名称が設定されます。
予備	16	8	－	－
入力元コネクション名	24	8	英数字	コネクション名が設定されます。
メッセージ入力日付	32	8	外部 10 進数字	RHLDEVT を入力した日付です。
メッセージ入力時刻	40	8	外部 10 進数字	RHLDEVT を入力した時刻です。
予備	48	16	－	－
予備	64	44	－	－

（凡例）

－：該当しません，または使用されません。

表 5-11　COBOL 言語の MCF イベント情報の内容（MDELEVT）

項目	位置（バイト）	長さ（バイト）	属性	内容
イベントコード	0	8	英数字	イベントコード「MDELEVT」が設定されます。
論理端末名称	8	8	英数字	破棄したメッセージを入力した論理端末名称，または破棄したメッセージを出力しようとした論理端末名称が設定されます。
予備	16	8	－	－
入力元コネクション名	24	8	英数字	コネクション名が設定されます。
メッセージ入力日付	32	8	外部 10 進数字	MDELEVT を入力した日付です。

項目	位置（バイト）	長さ（バイト）	属性	内容
メッセージ入力時刻	40	8	外部 10 進数字	MDELEVT を入力した時刻です。
理由コード※	48	4	2 進数字	理由コードが設定されます。
予備	52	56	－	－

（凡例）

－：該当しません，または使用されません。

注※

理由コードの詳細については，「付録 J　理由コード一覧」を参照してください。

6 システム定義

この章では，TCP/IP プロトコルを使用するために必要な，OpenTP1 のシステム定義の中での TP1/NET/TCP/IP 固有のシステム定義，および定義例について説明します。

TP1/NET/TCP/IP の定義の概要

TP1/NET/TCP/IP のシステム定義は，OpenTP1 のネットワークコミュニケーション定義の中で定義します。

OpenTP1 のネットワークコミュニケーション定義の中での定義

OpenTP1 のネットワークコミュニケーション定義のうち，TP1/NET/TCP/IP に固有の定義について説明します。

使用する定義ファイル

MCF および TP1/NET/TCP/IP を起動するには，定義ファイルに環境情報を設定する必要があります。MCF で使用する定義ファイルを次の表に示します。

表 6-1　MCF で使用する定義ファイル

定義の種類	定義のソースファイル	定義の内容
MCF マネジャ定義	MCF マネジャ定義ソースファイル	MCF 全体の実行環境
MCF 通信構成定義	共通定義ソースファイル	プロトコルごとの実行環境
	プロトコル固有定義ソースファイル	
MCF アプリケーション定義	MCF アプリケーション定義ソースファイル	アプリケーションの属性

定義のソースファイルは，定義コマンド，オプション，およびオペランドを指定して作成します。それらの中には，プロトコルで共通のものと，プロトコルに固有のものがあります。次の定義には，TP1/NET/TCP/IP に固有の定義があります。

- MCF マネジャ定義
- MCF 通信構成定義

この章では，TP1/NET/TCP/IP に固有の定義コマンド，オプション，およびオペランドについて説明します。プロトコルで共通の定義については，マニュアル「OpenTP1 システム定義」を参照してください。ただし，mcftbuf（バッファグループ定義）の length，count オペランドの指定値については，「6.　システム定義」の「mcftalccn（コネクション定義の開始）」の注意事項を参照してください。

TP1/NET/TCP/IP の組み込み時に必要なファイル

次に示すファイルは，TP1/NET/TCP/IP を OpenTP1 システムに組み込むときに必要なファイルです。

- システムサービス情報定義ファイル
- システムサービス共通情報定義ファイル
- MCF 定義オブジェクトファイル

この章では，システムサービス情報定義ファイルとシステムサービス共通情報定義ファイルの記述内容，およびMCF定義オブジェクトファイルを生成するユティリティの起動コマンドについて説明します。TP1/NET/TCP/IPを組み込む方法については，「8.　組み込み方法」を参照してください。

TP1/NET/TCP/IP 固有のシステム定義の種類

OpenTP1 のネットワークコミュニケーション定義のうち，TP1/NET/TCP/IP に固有の定義の一覧を次の表に示します。

表 6-2　TP1/NET/TCP/IP 固有の定義の一覧

定義名		コマンド	オプション・オペランド		定義内容	指定値((値範囲))《省略時解釈値》
MCF マネジャ定義		mcfmuap※ (UAP 共通定義)	-t	sndtim	同期型送信監視時間	符号なし整数((0～65535))《0》(単位：秒)
				sndrcvtim	同期型送受信監視時間	符号なし整数((0～65535))《0》(単位：秒)
				recvtim	同期型受信監視時間	符号なし整数((0～65535))《0》(単位：秒)
MCF 通信構成定義	共通定義	mcfttrc※ (トレース環境定義)	-t	msgsize	トレース取得するメッセージのサイズ	符号なし整数((0～1073741824))《128》(単位：バイト)
	プロトコル固有定義	mcftalccn (コネクション定義の開始) 指定数：1～2048	-c	－	コネクション ID	1～8 文字の識別子
			-N	modelname	モデルコネクション ID	1～8 文字の識別子
			-p	－	プロトコルの種別	tcp
			-g	sndbuf	メッセージ送信用バッファグループ番号	符号なし整数((1～512))
				rcvbuf	メッセージ受信用バッファグループ番号	符号なし整数((1～512))
			-e	msgbuf	メッセージ編集用バッファグループ番号	符号なし整数((1～512))
				count	メッセージ編集用バッファ数	符号なし整数((1～131070))
			-i	－	コネクションの確立方法	auto ｜《manual》
			-b	bretry	コネクション確立障害時の確立再試行するかどうかを指定	《yes》｜ no
				bretrycnt	コネクション確立障害時の確立再試行回数	符号なし整数((0～65535))《0》(単位：回)

定義名		コマンド	オプション・オペランド		定義内容	指定値((値範囲))《省略時解釈値》
MCF 通信構成定義	プロトコル固有定義	mcftalccn (コネクション定義の開始) 指定数：1～2048	-b	bretryint	コネクション確立障害時の確立再試行間隔	符号なし整数((0～2550))《60》(単位：秒)
				concmptim	コネクション確立時の監視時間	符号なし整数((0～65535))《0》(単位：秒)
				sndcmptim	メッセージ送信完了までの監視時間	符号なし整数((0～65535))《0》(単位：秒)
			-w	srtimout	コネクション切断抑止をするかどうかを指定	yes | 《no》
			-t	－	トランスポート層のプロトコルの種別	tcp
			-y	mode	クライアントとサーバの種別	client | server
			-r	ipaddr	自システムのホストの IP アドレス	符号なし整数((0～255))(nnn.nnn.nnn.nnn)
				hostname	自システムのホスト名	1～255 文字のホスト名
				portno	自システムのホストのポート番号	符号なし整数((1024～65535))
			-o	oipaddr	相手システムのホストの IP アドレス	符号なし整数((0～255))(nnn.nnn.nnn.nnn)
				ohostname	相手システムのホスト名	1～255 文字のホスト名
				oportno	相手システムのホストのポート番号	符号なし整数((1～65535))《free》
			-k	keepalive	ソケットオプション「SO_KEEPALIVE」を使用するかどうかを指定	yes | 《no》
				nodelay	ソケットオプション「TCP_NODELAY」を使用するかどうかを指定	yes | 《no》
				notrftime	無通信状態監視時間	符号なし整数((0～65535))《0》(単位：秒)

定義名		コマンド	オプション・オペランド		定義内容	指定値((値範囲))《省略時解釈値》
MCF 通信構成定義	プロトコル固有定義	mcftalccn (コネクション定義の開始) 指定数：1～2048	-f	kind	相手システムからのコネクション解放の通知	《ccls》｜cerr
				cnrelease	コネクション解放形態	《fin》｜rst
				releaselog	コネクション解放時のログメッセージの形式	1｜《2》
				cnerrlog	障害によるコネクション切断時のログメッセージの形式	1｜《2》
			-A	mastercn	現用コネクション ID	1～8 文字の識別子
			-u	masm	受信メッセージ組み立て機能を使用するかどうかを指定	yes｜《no》
				ntimer	後続セグメント受信の監視タイマ	《yes》｜no
				ntime	後続セグメント受信の監視タイマ値	符号なし整数((1～2550))《30》(単位：秒)
				delichk	メッセージ送達確認機能を使用するかどうかを指定	dccm2m｜dccm2s｜dccm3m｜dccm3s｜use｜《nouse》
				msghold	受信メッセージを保留するかどうかを指定	《nohold》｜hold｜uoc
				holdlimit	受信メッセージの最大保留数	符号なし整数((0～256))《1》
			-h	addrchk	相手アドレス情報のチェックをするかどうかを指定	《yes》｜no
				chgconn	コネクション確立要求を受け付けた場合，未確立コネクションがなかったときの動作（コネクションリプレース機能を使用するかどうかを指定）	replace｜《keep》
				listen	オンライン開始時から自動的にコネクション確立要求を受け付けるかどうかを指定	《auto》｜manual

定義名		コマンド	オプション・オペランド		定義内容	指定値((値範囲))《省略時解釈値》
MCF 通信構成定義	プロトコル固有定義	mcftalccn (コネクション定義の開始) 指定数：1～2048	-C	lscnfmt	mcftlscn コマンドの表示形式	《1》｜2
			-l	replymsg	問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信を行うかどうかを指定	yes｜《no》
				cnassign	相手からのコネクション確立要求時にコネクションを割り当てる対象の選択	《freeonly》｜all
		mcftalcle (論理端末定義) 指定数：1～2048	-l	－	論理端末名称	1～8 文字の識別子
			-N	modelname	モデル論理端末名称	1～8 文字の識別子
			-t	－	論理端末の端末タイプ	any
			-m	mmsgcnt	メモリ出力メッセージ最大格納数	符号なし整数((0～65535))《0》
				dmsgcnt	ディスク出力メッセージ最大格納数	符号なし整数((0～65535))《0》
			-i	－	論理端末の閉塞解除方法	《auto》｜manual
			-k	quekind	出力メッセージの割り当て先	《memory》｜disk
				quegrpid	キューグループ ID	1～8 文字の識別子
			-o	aj	メッセージ送信完了ジャーナルを取得するかどうかを指定	《yes》｜no
			-v	－	アプリケーション名	1～8 文字の識別子
			-d	replacemsg	コネクション再確立時に未送信メッセージを送信するかどうかを指定（コネクション再確立時の未送信メッセージの送信抑止機能を使用するかどうかを指定)	《send》｜discard
		mcftalced (コネクション定義の終了) 指定数：mcftalccn と同数	－	－	コネクション定義の終了	－

定義名	コマンド	オプション・オペランド		定義内容	指定値((値範囲))《省略時解釈値》
MCF アプリケーション定義	mcfaalcap※ (アプリケーション属性定義) 指定数：1～8192	-n	replychk	応答送信チェックをするかどうかを指定	《yes》｜no

(凡例)

－：該当する内容がないことを表します。

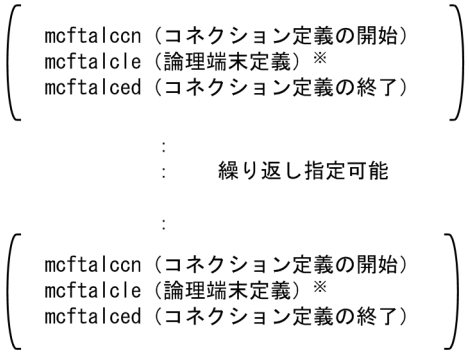
注※

TP1/NET/TCP/IP に固有の定義だけ記載してあります。このほかにも，プロトコルで共通の定義コマンド，オプション，オペランドがあります。それらについては，マニュアル「OpenTP1 システム定義」を参照してください。

定義の指定順序

TP1/NET/TCP/IP のプロトコル固有定義コマンドの指定順序を次の図に示します。MCF 通信構成定義コマンドを指定するときは，必ずこの順序に従ってください。

図 6-1　TP1/NET/TCP/IP のプロトコル固有定義コマンドの指定順序



注※

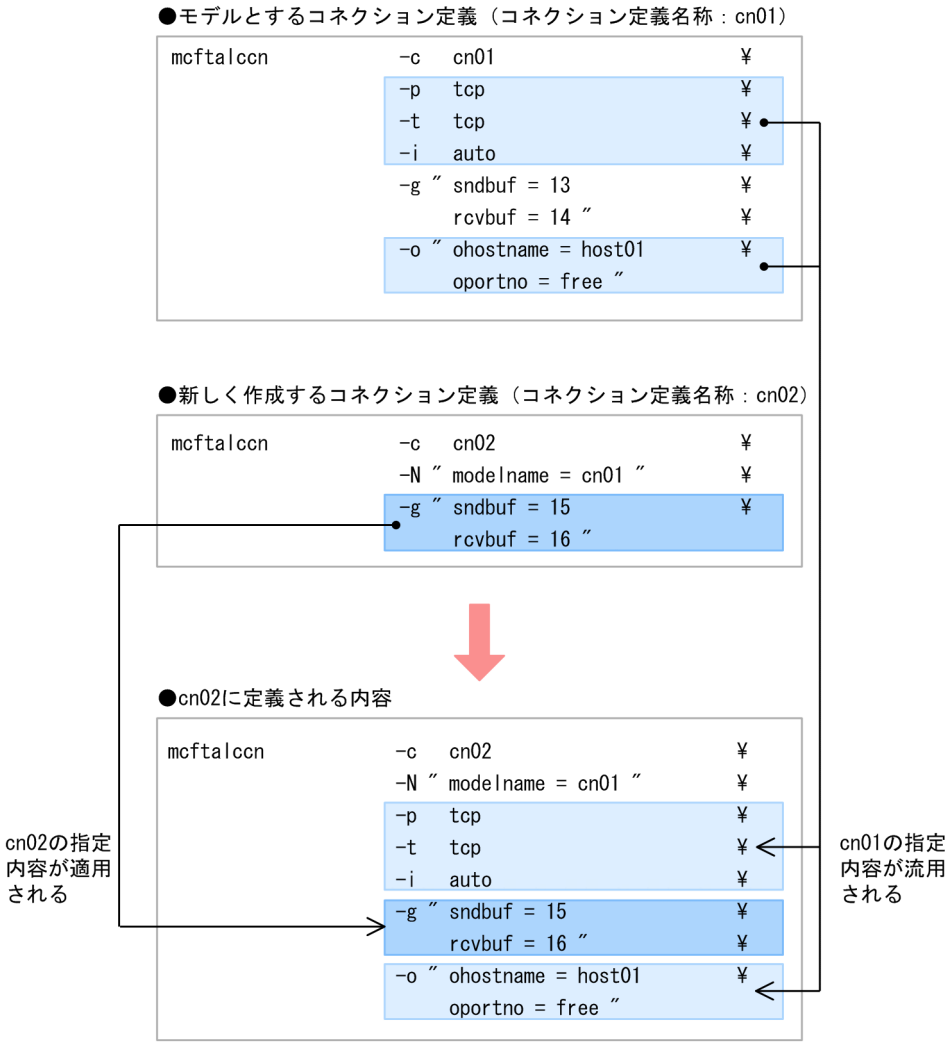
mcftalccn コマンドで-A オプションを指定する場合，mcftalcle は指定できません。

定義内容の流用

MCF 通信構成定義では，コネクション定義の開始（mcftalccn -N）または論理端末定義（mcftalcle -N）の modelname オペランドを使用すると，モデルとする定義の指定内容を流用できます。

このとき，モデルとする MCF 通信構成定義の一部だけを流用して，新しい定義を作成することもできます。例えば，コネクション名称「cn01」のコネクション定義の一部を流用してコネクション名称「cn02」を定義する場合，次の図に示すように定義できます。

図 6-2　コネクション定義の指定例



mcfaalcap（アプリケーション属性定義）

形式

```
mcfaalcap           :
      〔-n ”〔replychk=yes｜no〕”〕

                    :
```

機能

アプリケーションに関する属性を定義します。

オプション

この定義コマンドには，ほかにもオプションがあります。詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

●-n

このオプションには，ほかにもオペランドがあります。詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

（オペランド）

replychk=yes ｜ no　～《yes》

応答型アプリケーションが応答メッセージを送信したかどうか，MCF にチェックさせるかどうかを指定します。

yes

応答型アプリケーションが応答メッセージを送信していない場合，またはほかの応答型アプリケーションを起動していない場合に，トランザクションをロールバックし ERREVT3 を通知します。

no

応答型アプリケーションが応答メッセージを送信したかどうかをチェックしません。

replychk オペランドを指定した場合は，type オペランドに ans を，kind オペランドに user を指定する必要があります。

mcfmuap（UAP 共通定義）

形式

```
mcfmuap                 :
       〔-t ”〔sndtim=同期型送信監視時間〕
            〔sndrcvtim=同期型送受信監視時間〕
            〔recvtim=同期型受信監視時間〕”〕

                        :
```

機能

UAP に共通する環境を定義します。

オプション

この定義コマンドには，ほかにもオプションがあります。詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

●-t

（オペランド）

sndtim=同期型送信監視時間　　～〈符号なし整数〉((0～65535))《0》（単位：秒）

同期型のメッセージ送信の仕掛り開始から終了までの限界監視時間を指定します。監視時間の詳細については，「6.　システム定義」の「アプリケーションプログラムとシステム環境設定の関連」を参照してください。0 を指定した場合，監視を行いません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

sndrcvtim=同期型送受信監視時間　　～〈符号なし整数〉((0～65535))《0》（単位：秒）

同期型のメッセージ送受信の仕掛り開始から終了までの限界監視時間を指定します。監視時間の詳細については，「6.　システム定義」の「アプリケーションプログラムとシステム環境設定の関連」を参照してください。0 を指定した場合，監視を行いません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間

が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

recvtim=同期型受信監視時間　　～〈符号なし整数〉((0～65535))《0》(単位：秒)

同期型のメッセージ受信の仕掛り開始から終了までの限界監視時間を指定します。監視時間の詳細については，「6.　システム定義」の「アプリケーションプログラムとシステム環境設定の関連」を参照してください。0 を指定した場合，監視を行いません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

mcfttrc（トレース環境定義）

形式

```
mcfttrc                 :
      〔-t ”〔msgsize=MCFトレースに取得する送受信メッセージのサイズ〕”〕
                        :
```

機能

MCF のトレースに関する環境を定義します。

オプション

この定義コマンドには，ほかにもオプションがあります。詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

●-t

（オペランド）

msgsize=MCF トレースに取得する送受信メッセージのサイズ　～〈符号なし整数〉((0～1073741824))《128》（単位：バイト）

TP1/NET/TCP/IP が MCF トレースに取得する送受信メッセージの最大メッセージ長を指定します。0 を指定した場合，送受信メッセージの内容を MCF トレースに取得しません。送受信メッセージのサイズがこの指定値より小さい場合は，メッセージの全内容を MCF トレースに取得します。

このオペランドを指定する場合，トレースバッファの大きさ（mcfttrc -t size）はこのオペランドよりも十分に大きな値を指定してください。トレースバッファの大きさよりもこのオペランドの指定値が大きい場合，トレースバッファの大きさよりも大きなメッセージを送受信したとき，TP1/NET/TCP/IP は該当するメッセージの MCF トレースを取得しません。

MCF トレースファイルの見積もり式については，「6.　システム定義」の「MCF トレースファイルの見積もり式」を参照してください。

mcftalccn（コネクション定義の開始）

形式

```
mcftalccn -c コネクションID
         〔-N "modelname=モデルコネクションID"〕
          -p tcp
          -g "sndbuf=メッセージ送信用バッファグループ番号
              rcvbuf=メッセージ受信用バッファグループ番号"
         〔-e "msgbuf=メッセージ編集用バッファグループ番号
               count=メッセージ編集用バッファ数"〕
         〔-i auto｜manual〕
         〔-b "〔bretry=yes｜no〕
               〔bretrycnt=コネクション確立障害時の確立再試行回数〕
               〔bretryint=コネクション確立障害時の確立再試行間隔〕
               〔concmptim=コネクション確立時監視時間〕
               〔sndcmptim=メッセージ送信完了監視時間〕"〕
         〔-w "srtimout=yes｜no"〕
          -t tcp
          -y "mode=client｜server"
         〔-r "〔ipaddr=自システムのホストのIPアドレス〕
               〔hostname=自システムのホスト名〕
                portno=自システムのポート番号"〕
         〔-o "〔oipaddr=相手システムのホストのIPアドレス〕
               〔ohostname=相手システムのホスト名〕
                oportno=相手システムのホストのポート番号｜free"〕
         〔-k "〔keepalive=yes｜no〕
               〔nodelay=yes｜no〕
               〔notrftime=無通信状態監視時間〕"〕
         〔-f "〔kind=ccls｜cerr〕
               〔cnrelease=fin｜rst〕
               〔releaselog=1｜2〕
               〔cnerrlog=1｜2〕"〕
         〔-A "mastercn=現用コネクションID"〕
         〔-u "〔masm=yes｜no〕
               〔ntimer=yes｜no〕
               〔ntime=後続セグメント受信の監視タイマ値〕
               〔delichk=dccm2m｜dccm2s｜dccm3m｜dccm3s｜use｜nouse〕
               〔msghold=nohold｜hold｜uoc〕
               〔holdlimit=受信メッセージ最大保留数〕"〕
         〔-h "〔addrchk=yes｜no〕
               〔chgconn=replace｜keep〕
               〔listen=auto｜manual〕"〕
         〔-C "lscnfmt=1｜2"〕
         〔-l "〔replymsg=yes｜no〕
               〔cnassign=freeonly｜all〕"〕
```

機能

コネクションに関する環境を定義します。

オプション

●-c コネクション ID　～〈1～8 文字の識別子〉

OpenTP1 システム内で，一意となるコネクション ID を指定します。

●-N

（オペランド）

modelname=モデルコネクション ID　～〈1～8 文字の識別子〉

このコネクション定義で使用する定義情報を持つ（モデルとする），コネクション定義のコネクション ID を指定します。ただし，指定したコネクション ID に対するコネクション定義が，先に定義されていなければなりません。

このオペランドを指定したときは，-c オプション以外のオプション，またはオペランドを省略できます。

-c オプション以外のオプション，またはオペランドを省略した場合，modelname に指定したコネクション ID の定義指定値がすべて流用されます。

このコネクション定義に-N オプション以外のオプション，またはオペランドを指定した場合は，指定したオプション，またはオペランドの指定値が優先されます。

このオプションは 2 回以上指定できません。また，モデルとしたコネクションと，-N オプション以外に指定したオプション，またはオペランドの組み合わせによっては，相関チェックによってエラーになる場合があります。そのため，-i オプションには，モデル定義と異なる指定をしないでください。

●-p tcp

プロトコルの種別を指定します。

tcp

TCP/IP プロトコル

●-g

（オペランド）

sndbuf=メッセージ送信用バッファグループ番号　～〈符号なし整数〉((1～512))

メッセージ送信用バッファグループ番号を指定します。

mcftbuf コマンドの-g オプションの groupno オペランドで指定するバッファグループ番号を指定してください。

rcvbuf=メッセージ受信用バッファグループ番号　～〈符号なし整数〉((1～512))

メッセージ受信用バッファグループ番号を指定します。

mcftbuf コマンドの-g オプションの groupno オペランドで指定するバッファグループ番号を指定してください。

●-e

（オペランド）

msgbuf=メッセージ編集用バッファグループ番号　～〈符号なし整数〉((1～512))

入力，出力メッセージ編集 UOC 呼び出し時に，メッセージ編集用として使用するバッファグループ番号を指定します。

このオペランドを省略した場合は，メッセージ編集用バッファは確保されません。メッセージ編集用バッファグループ番号は，mcftbuf コマンドの-g オプションの groupno オペランドで指定するバッファグループ番号を指定します。

count=メッセージ編集用バッファ数　～〈符号なし整数〉((1～131070))

入力，出力メッセージ編集 UOC 呼び出し時に，メッセージ編集用として使用するバッファの数を指定します。

msgbuf オペランドで指定するメッセージ編集用バッファグループ番号に対応する mcftbuf コマンドの-g オプションの count，および extend オペランドで指定するバッファ数の中から，1 回の UOC 呼び出しごとに 1 面のバッファを使用します。このオペランドに 2 以上の値を指定しても，入力メッセージ編集 UOC および出力メッセージ編集 UOC で使用できる編集バッファ数は一つだけです。

メッセージ編集用バッファグループ番号に対応する mcftbuf の count オペランドおよび extend オペランドには，すべての論理端末で入力メッセージ編集 UOC および出力メッセージ編集 UOC が，同時に動作した場合を考慮した面数を指定してください。

また，この count オペランドで指定するメッセージ編集用バッファ数は，mcftbuf コマンドの-g オプションの count，および extend オペランドで指定するバッファ数の合計値を超える指定はできません。

msgbuf オペランドを省略した場合は，このオペランドの指定は無効です。

msgbuf オペランドを指定した場合，このオペランドを省略できません。省略した場合，定義オブジェクト生成時に KFCA11519-E メッセージを出力してエラーとなります。

●-i auto | manual　～《manual》

OpenTP1 システム開始時および再開始時の，コネクションの確立の方法を指定します。このオプションは，mcftalccn コマンドの-y オプションの mode オペランドで，client を指定した場合だけ有効です。また，-A オプションを指定した場合は manual 以外は指定できません。

auto

OpenTP1 システム開始時および再開始時にコネクションを自動的に確立します。

manual

MCF 起動後，コネクションを確立します。コネクションの確立は，運用コマンド（mcftactcn）の入力，または API（dc_mcf_tactcn 関数もしくは CBLDCMCF('TACTCN△△')）の発行で行います。

●-b

(オペランド)

bretry=yes | no　～《yes》

コネクション確立時に障害が発生した場合，コネクション確立の再試行をするかどうかを指定します。mcftalccn コマンドの-y オプションの mode オペランドで，server を指定した場合にはネットワークへのアドレス割り当ての再試行をします。

yes

コネクションの確立再試行をします。

no

コネクションの確立再試行をしません。

bretrycnt=コネクション確立障害時の確立再試行回数　～〈符号なし整数〉((0～65535))《0》(単位：回)

コネクションの確立再試行をする場合の確立再試行回数を指定します。bretry オペランドで no を指定した場合，このオペランドの指定は無効です。

bretry オペランドで yes を指定し，このオペランドに 0 を指定した場合，無限に確立再試行をします。

bretryint=コネクション確立障害時の確立再試行間隔　～〈符号なし整数〉((0～2550))《60》(単位：秒)

コネクションの確立再試行をする場合の確立再試行間隔を指定します。bretry オペランドで no を指定した場合，このオペランドの指定は無効です。

注意事項

再試行間隔の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔で再試行するかどうかをチェックします。このため，このオペランドで指定した再試行間隔と実際に再試行する時間には秒単位の誤差が生じます。

concmptim=コネクション確立時監視時間　～〈符号なし整数〉((0～65535))《0》(単位：秒)

コネクション確立時の監視時間を指定します。0 を指定するか，指定を省略した場合は，OS の監視時間に依存します。

コネクション確立時の監視時間がタイムアウトした場合，コネクション確立を再試行できれば，コネクション確立障害時の確立再試行間隔後，コネクション確立を再試行します。コネクション確立を再試行できなければ，CERREVT を起動します。コネクション確立障害時再試行との関係については，「付録 E　障害発生時の処理の流れ」を参照してください。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

sndcmptim=メッセージ送信完了監視時間　～〈符号なし整数〉((0～65535))《0》(単位：秒)

メッセージ送信完了までの監視時間を指定します。0 を指定した場合，時間監視をしません。また，同期型メッセージの送信関数の発行時，このオペランドによる時間監視を行いません。同期型メッセージの送信関数の時間監視は，関数発行時に指定する監視時間で行ってください。

同期型メッセージの送受信関数でメッセージ送信完了までの時間を監視する場合，このオペランドには，次の値より小さな値を指定してください。

- mcfmuap コマンドの-t オプションの sndrcvtim オペランドで指定する同期型送受信監視時間

- 同期型メッセージの送受信関数の発行時（dc_mcf_sendrecv 関数または CBLDCMCF('SENDRECV')）に指定する最大時間

監視対象区間やタイムアウトした場合の動作については，「6.　システム定義」の「アプリケーションプログラムとシステム環境設定の関連」を参照してください。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

●-w

（オペランド）

srtimout=yes | no　　～《no》

同期型メッセージの送受信関数がタイムアウトした場合に，コネクションの切断を抑止するかどうかを指定します。

yes

同期型メッセージの送受信関数がタイムアウトした場合，コネクションを切断しません。

no

同期型メッセージの送受信関数がタイムアウトした場合，コネクションを切断します。

●-t tcp

トランスポート層に適用するプロトコルを指定します。

tcp

TCP/IP プロトコル

●-y

（オペランド）

mode=client | server

自システムのコネクション確立モードを指定します。コネクション切り替え機能の対象となるコネクションの場合，このオペランドでは client しか指定できません。

client

クライアントとして，コネクションの確立要求をします。

server

サーバとして，コネクションの確立要求を受けます。

●-r

（オペランド）

ipaddr=自システムのホストの IP アドレス　～（nn.nn.nn.nn）〈符号なし整数〉((0～255))

自システムのホストの IP アドレスを指定します。

指定形式は nn.nn.nn.nn です。nn の値は，0～255 まで指定できます。

ただし，0.0.0.0 と 255.255.255.255 は指定できません。

このオペランドを指定した場合，指定したアドレスをバインドします。また，このオペランドを省略した場合，自システムの IP アドレスは OS によって割り振られます

hostname=自システムのホスト名　～〈1～255 文字のホスト名〉

自システムのホスト名を指定します。

自システムがサーバで，OpenTP1 の開始時に IP アドレスを取得できなかった場合，開始処理を中断します。

ipaddr オペランドを指定した場合，このオペランドの指定は無効です。

このオペランドを指定し，ipaddr オペランドの指定を省略した場合，このオペランドで指定したアドレスをバインドします。

portno=自システムのポート番号　～〈符号なし整数〉((1024～65535))

メッセージ送受信に使用する，自システムのポート番号を指定します。

コネクションとポートの関係については，「2.1.1(1)　コネクションとポートの関係」を参照してください。

ほかのプロセスが使用するポートと重複しないポート番号を指定してください。また，OS が任意に割り当てるポート番号（動的ポートまたは短命ポートと呼ばれるポート番号）を使用しないでください。

OS が任意に割り当てるポート番号は，OS の種別やバージョンによって異なります。詳細については，ご使用の OS のマニュアルを参照してください。

一方，同じ定義内（MCF 通信プロセス内）に複数のコネクションを指定する場合は，重複したポート番号を指定できます。ただし，図 2-3 に示すパターンで重複させることはできません。

クライアント型コネクション（コネクション定義（mcftalccn -y）の mode オペランドで client を指定）の場合は省略できます。省略した場合，自システムのポート番号はコネクション確立時に OS によって割り振られます。

サーバ型コネクション（コネクション定義（mcftalccn -y）の mode オペランドで server を指定）の場合は必ず指定してください。

-r オプションの portno オペランドが同じサーバ型コネクションを複数指定する場合，-r オプションの ipaddr オペランドおよび hostname オペランドの指定を省略したコネクションと，-r オプションの ipaddr オペランドまたは hostname オペランドを指定したコネクションを混在して使用しないでください。

●-o

（オペランド）

oipaddr=相手システムのホストの IP アドレス　～（nn.nn.nn.nn）〈符号なし整数〉((0～255))

相手システムのホストの IP アドレスを指定します。

指定形式は，nn.nn.nn.nn です。nn の値は，0～255 まで指定できます。

ただし，0.0.0.0 と 255.255.255.255 は指定できません。

ohostname=相手システムのホスト名　～〈1～255 文字のホスト名〉

相手システムのホスト名を指定します。

-o オプションを指定するときは，上に示す二つのオペランドのどちらかを必ず指定してください。二つのオペランドを指定すると，ohostname オペランドは無効となります。

oportno=相手システムのホストのポート番号｜free　～〈符号なし整数〉((1～65535))《free》

相手システムのホストのポート番号を指定します。free を指定できるのは，-y オプションの mode オペランドで server を指定した場合だけです。

free を指定した場合，oipaddr または ohostname オペランドで指定した相手システムと一致する任意のポート番号からのコネクション確立要求を受け入れます。このとき，一つの相手システムから複数のコネクションを受け入れるためには，受け入れるコネクション数の分だけコネクション定義を指定する必要があります。さらに，それぞれのコネクション定義には，同一の oipaddr または ohostname オペランドを指定してください。指定したコネクション定義の数以上のコネクション確立要求を受けた場合，要求を拒否します。

●-k

（オペランド）

keepalive=yes｜no　～《no》

TCP/IP が提供するソケットオプション「SO_KEEPALIVE」を使用するかどうかを指定します。

SO_KEEPALIVE を使用することで，コネクションの切断を検出することができるようになりますが，ネットワークの負荷を高める場合があります。そのため，この機能の必要性を十分に検討した上で使用するようにしてください。

yes

キープアライブを使用します。

no

キープアライブを使用しません。

nodelay=yes｜no　～《no》

TCP/IP が提供するソケットオプション「TCP_NODELAY」を使用するかどうかを指定します。

TCP_NODELAY を使用することで，送信済みデータの応答待ちの状態でも，遅延させることなくデータ送信ができるようになりますが，ネットワークの負荷は大きくなります。そのため，この機能の必要性を十分に検討した上で使用するようにしてください。

yes

TCP_NODELAY を使用します。

no

TCP_NODELAY を使用しません。

notrftime=無通信状態監視時間　～〈符号なし整数〉((0～65535))《0》(単位：秒)

コネクションの無通信状態の経過時間を監視するタイマ値を設定します。0 を指定した場合，時間監視をしません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

●-f

(オペランド)

kind=ccls | cerr　～《ccls》

相手システムからのコネクション解放の通知方法を指定します。

ccls

状態通知イベント（CCLSEVT）で通知します。

cerr

障害通知イベント（CERREVT）で通知します。

cnrelease=fin | rst　～《fin》

障害などによるコネクション解放が発生したときのコネクション解放形態を指定します。コネクションを解放する要因と送信するパケットの関係については，「2.1.9　コネクション障害」を参照してください。

fin

FIN パケットを送信してコネクションを解放します。

rst

RST パケットを送信してコネクションを強制的に解放します。

releaselog=1 | 2　～《2》

コネクションを解放したときのログメッセージの形式を指定します。

1

KFCA14801-I の形式で出力します。

2

KFCA14875-I の形式で出力します。

cnerrlog=1 | 2　～《2》

障害によるコネクション切断が発生したときのログメッセージの形式を指定します。

1

KFCA14802-E を出力します。

2

KFCA14802-E と KFCA14876-I を出力します。

●-A

（オペランド）

mastercn=現用コネクション ID　～〈1～8 文字の識別子〉

コネクション切り替え機能の使用時に，該当するコネクションを交代用として指定する場合に，その対となる現用コネクション ID を指定します。

このオペランドを指定した場合，現用コネクションで指定されている論理端末定義が反映されます。

このオペランドを指定するには，TP1/NET/High Availability が必要です。TP1/NET/High Availability がないと，コネクション切り替え機能を使用できません。

交代用コネクションと MCF 通信構成定義との関係については，「6.　システム定義」の「コネクションの形態と MCF 通信構成定義との関係」を参照してください。

●-u

（オペランド）

masm=yes | no　～《no》

受信メッセージの組み立て機能を使用するかどうかを指定します。

yes

受信メッセージ組み立て機能を使用します。

yes を指定した場合，入力セグメント判定 UOC は無効となるため，入力セグメント判定 UOC が組み込まれていても起動しません。

no

メッセージ組み立て機能を使用しません。

入力セグメント判定 UOC が組み込まれている場合，入力セグメント判定 UOC を起動します。入力セグメント判定 UOC が組み込まれていない場合，セグメントの組み立てをしないで，TP1/NET/TCP/IP が受信したデータを論理メッセージとします。

ntimer=yes | no　～《yes》

後続セグメント受信の監視タイマをセットするかどうかを指定します。このオペランドの指定値は，受信メッセージ組み立て機能を使用する場合，つまり，masm オペランドに yes を指定した場合だけ有効です。

yes

監視タイマをセットします。

no

監視タイマをセットしません。

ntime=後続セグメント受信の監視タイマ値　～〈符号なし整数〉((1～2550))《30》(単位：秒)

後続セグメント受信の監視タイマをセットする場合に，監視タイマ値を指定します。後続セグメント受信監視時間がタイムアウトした場合，コネクションを解放し，CERREVT を起動します。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。監視時間が小さくなるほど，誤差の影響を受けやすくなりますので，監視時間は 3（単位：秒）以上の値の設定を推奨します。

delichk=dccm2m | dccm2s | dccm3m | dccm3s | use | nouse　～《nouse》

メッセージ送達確認機能を使用するかどうかを指定します。メッセージ送達確認機能を使用する場合，同期型メッセージの送受信関数を使用できません。

dccm2m

メッセージ送達確認機能を使用し，TERMINAL 文の CONTENT オペランドが MASTER を指定した DCCMII/TCP として動作します。

dccm2s

メッセージ送達確認機能を使用し，TERMINAL 文の CONTENT オペランドが SLAVE を指定した DCCMII/TCP として動作します。

dccm3m

メッセージ送達確認機能を使用し，TERMINAL 文の CONTENT オペランドが MASTER を指定した DCCM3/TCP として動作します。

dccm3s

メッセージ送達確認機能を使用し，TERMINAL 文の CONTENT オペランドが SLAVE を指定した DCCM3/TCP として動作します。

use

任意の相手システム間とのメッセージ送達確認を行います。受信メッセージ判定 UOC が組み込まれている場合，メッセージ受信後に呼び出します。受信メッセージ判定 UOC が組み込まれていない場合，開始処理を中断します。

nouse

メッセージ送達確認機能を使用しません。

メッセージ送達確認機能を使用する場合のほかのオプション，オペランドとの関係

- 送信完了監視タイマ（-b オプションの sndcmptim オペランド）
  メッセージの送信から応答データの受信までを監視します。
- 受信メッセージの組み立て機能を使用するかどうか（-u オプションの masm オペランド）

dccm2m，dccm2s，dccm3m，dccm3s を指定した場合は，no を指定できません。省略した場合，yes を指定したものと見なします。

msghold=nohold | hold | uoc　～《nohold》

相手システムから受信したメッセージを TP1/NET/TCP/IP 内で保留するかどうかを指定します。

ただし，同期型メッセージの送受信関数または同期型メッセージの受信関数で受信を待ち合わせていた場合，このオペランドの指定値に関係なく，同期型メッセージの送受信関数または同期型メッセージの受信関数を発行していた UAP に受信メッセージを通知します。

nohold

受信したメッセージを保留しません。

TP1/NET/TCP/IP は受信メッセージを入力キューに書き込み，MHP を起動します。

hold

受信したメッセージを保留します。

保留したメッセージは同期型メッセージの受信関数を使用して受信してください。

uoc

受信したメッセージごとに，受信メッセージ保留判定 UOC で保留するかどうかを判定します。

受信メッセージ保留判定 UOC が組み込まれていない場合，メッセージログ（KFCA14844-E）を出力し，開始処理を中断します。

holdlimit=受信メッセージ最大保留数　～〈符号なし整数〉((0～256))《1》

TP1/NET/TCP/IP 内で保留する受信メッセージの最大数を指定します。

このオペランドの指定値を超えて受信メッセージを保留した場合，メッセージログ（KFCA14867-E）を出力し，コネクションを解放します。

0 を指定した場合，最大保留数は無制限になります。ただし，実際に保留できるメッセージ数は，バッファグループ定義で指定したバッファ数（mcftbuf -g count）に依存します。

msghold オペランドを省略した場合，または msghold オペランドに nohold を指定した場合，このオペランドの指定値は無効となります。

●-h

（オペランド）

addrchk=yes | no　～《yes》

コネクション確立要求受信時に相手アドレス情報をチェックするかどうかを指定します。

一つの MCF 通信構成定義内でのすべてのサーバ型コネクションで，このオペランドに同じ値を指定する必要があります。

クライアント型コネクションにこのオペランドを指定しても無効となります。

yes

コネクション確立要求受信時に相手アドレス情報をチェックします。

相手システムのアドレス情報を定義しておく必要があります。

コネクション定義で指定された相手アドレスと確立要求発行元アドレスが一致する場合は確立要求を受け入れますが，一致しない場合は確立要求を拒否します。

no

コネクション確立要求受信時に相手アドレス情報をチェックしません（相手アドレスチェックを抑止します）。

相手システムのアドレス情報の定義は不要です。定義した場合は，無効となります。

コネクション確立 UOC の判断によって，任意の相手システムと確立できます。未確立コネクションがない場合は，確立を拒否します。

no を指定した場合，ほかのオプションのオペランドとの関係は次のようになります。

- -y オプションの mode オペランドで client を指定した場合は，addrchk オペランドの指定は無効となります。
- -o オプションの oipaddr オペランド，ohostname オペランドおよび oportno オペランドで相手システムのアドレス情報を指定しても無効となります。

chgconn=replace | keep　～《keep》

相手システムからコネクションの確立要求を受け付けた場合に，割り当てできる未確立コネクションがないときの動作（コネクションリプレースの使用有無）を指定します。

一つの MCF 通信構成定義内で，同じ自システムのポート番号（コネクション定義（mcftalccn -r）の portno オペランドで指定）を持つすべてのサーバ型コネクション（コネクション定義（mcftalccn -y）の mode オペランドで server を指定）で，このオペランドに同じ値を指定する必要があります。

クライアント型コネクション（コネクション定義（mcftalccn -y）の mode オペランドで client を指定）および，相手ポート番号が固定（コネクション定義（mcftalccn -o）の oportno オペランドに 1～65535 を指定）のサーバ型コネクションにこのオペランドを指定した場合，無効となります。

replace

コネクションリプレース機能を使用します。相手システムと確立状態にある最も古いコネクションを切断して，新たなコネクションの確立要求を受け付けます。

keep

コネクションリプレース機能を使用しません。既存のコネクションの確立状態を保持し，相手システムからの新たなコネクション確立要求を拒否します。

このオペランドを省略した場合は keep を仮定します。

listen=auto | manual　～《auto》

オンライン開始時に自動的にコネクション確立要求を受け付けるかどうかを指定します。

一つの MCF 通信構成定義内で，同じ自システムのポート番号（mcftalccn -r portno）を持つすべてのサーバ型コネクションで，このオペランドに同じ値を指定する必要があります。

クライアント型コネクションにこのオペランドを指定した場合は無効となります。

auto

オンライン開始時に自動的にコネクション確立要求を受け付けます。

manual

MCF 起動後に，相手システムからのコネクション確立要求の受け付けを開始します。コネクション確立要求の受付開始は，運用コマンド（mcftonln）の入力，または API（dc_mcf_tonln 関数もしくは CBLDCMCF('TONLN△△△')）で行います。

●-C

（オペランド）

lscnfmt=1 | 2　～《1》

mcftlscn コマンドの出力形式を指定します。

1

出力形式 1（TP1/NET/TCP/IP 07-01 以前と互換性のある形式）で状態表示します。

2

出力形式 2（TP1/NET/TCP/IP 07-02 で追加した形式）で状態表示します。

-d オプションを指定したときに，受信メッセージの保留数が追加で表示されます。

●-l

（オペランド）

replymsg=yes | no　～《no》

問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信を行うかどうかを指定します。

サーバ型コネクションで指定する場合，一つの MCF 通信構成定義内で，同じ自システムのポート番号（コネクション定義（mcftalccn -r）の portno オペランド）を持つすべてのサーバ型コネクション（コネクション定義（mcftalccn -y）の mode オペランドで server を指定）で，このオペランドに同じ値を指定する必要があります。

yes

問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信を行います。

yes を指定した場合，メッセージ送達確認機能は使用できません。コネクション定義（mcftalccn -u）の delichk オペランドに nouse 以外を指定した場合，定義オブジェクト生成時に KFCA11513-E メッセージを出力してエラーとなります。

no

問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信を行いません。

cnassign=freeonly | all　～《freeonly》

問い合わせ応答形態および継続問い合わせ応答形態のメッセージ送受信を行う場合（コネクション定義（mcftalccn -l）の replymsg オペランドで yes を指定），相手からのコネクション確立要求時にコネクションを割り当てる対象を選択します。

一つの MCF 通信構成定義内で，同じ自システムのポート番号（コネクション定義（mcftalccn -r）の portno オペランド）を持つすべてのサーバ型コネクション（コネクション定義（mcftalccn -y）の mode オペランドで server を指定）で，このオペランドに同じ値を指定する必要があります。

クライアント型コネクションにこのオペランドを指定しても無効となります。

freeonly

未確立のコネクションのうち，継続問い合わせ応答中の論理端末に対応するコネクションを割り当て対象から除きます。また，コネクションリプレースと併用する場合，継続問い合わせ応答中の論理端末に対応するコネクションをコネクションリプレースの対象から除きます。

all

未確立のコネクションすべてを割り当て対象とします。

継続問い合わせ応答中の論理端末に対応するコネクションが割り当てられた場合，KFCA14883-Eを出力し，コネクションを解放します。

replymsg オペランドを省略した場合，または replymsg オペランドに no を指定した場合，このオペランドの指定値は無効となります。この場合，相手からのコネクション確立要求時には未確立のコネクションすべてを割り当て対象とします。

注意事項

-g オプション，および-e オプションで指定するバッファグループ番号は，バッファグループ定義の mcftbuf コマンドに対応しています。mcftbuf コマンドでは，1 コネクション単位に次の表に示す資源が必要です。

なお，負荷テストの実行後に mcftlsbuf コマンドに-m オプションを指定して実行すると，最大バッファ使用数を確認できます。計算式の計算結果によって大量のバッファ数が必要となった場合，最大バッファ使用数を基にバッファ数を調整してください。

バッファグループ定義については，マニュアル「OpenTP1 システム定義」を参照してください。

バッファ種別	length オペランド	count オペランド※1
sndbuf	mcftalccn -u delichk=use，nouse の場合 （最大論理メッセージ長）※2 以上 mcftalccn -u delichk=dccm2m，dccm2s，dccm3m，dccm3s の場合 （最大論理メッセージ長＋ 7）※2 以上	mcftalccn -u delichk=nouse の場合 1 以上 mcftalccn -u delichk=dccm2m，dccm2s，dccm3m，dccm3s，use の場合 （受信バッファ数＋ 1）以上
rcvbuf	sndbuf と同じ	2 または次の計算式のどちらか大きい方 （length オペランドの値／受信するメッセージの最小長）※3 ＋（保留する受信メッセージの数）以上 （小数点以下を切り上げる）
msgbuf	sndbuf と同じ	2 以上

注※1

length オペランドと受信メッセージの最小長の差が極端に大きく，大量のバッファ数が必要となる場合，extend オペランドを使用してください。これによって，MCF 通信プロセスのメモリ使用量を抑えられます。

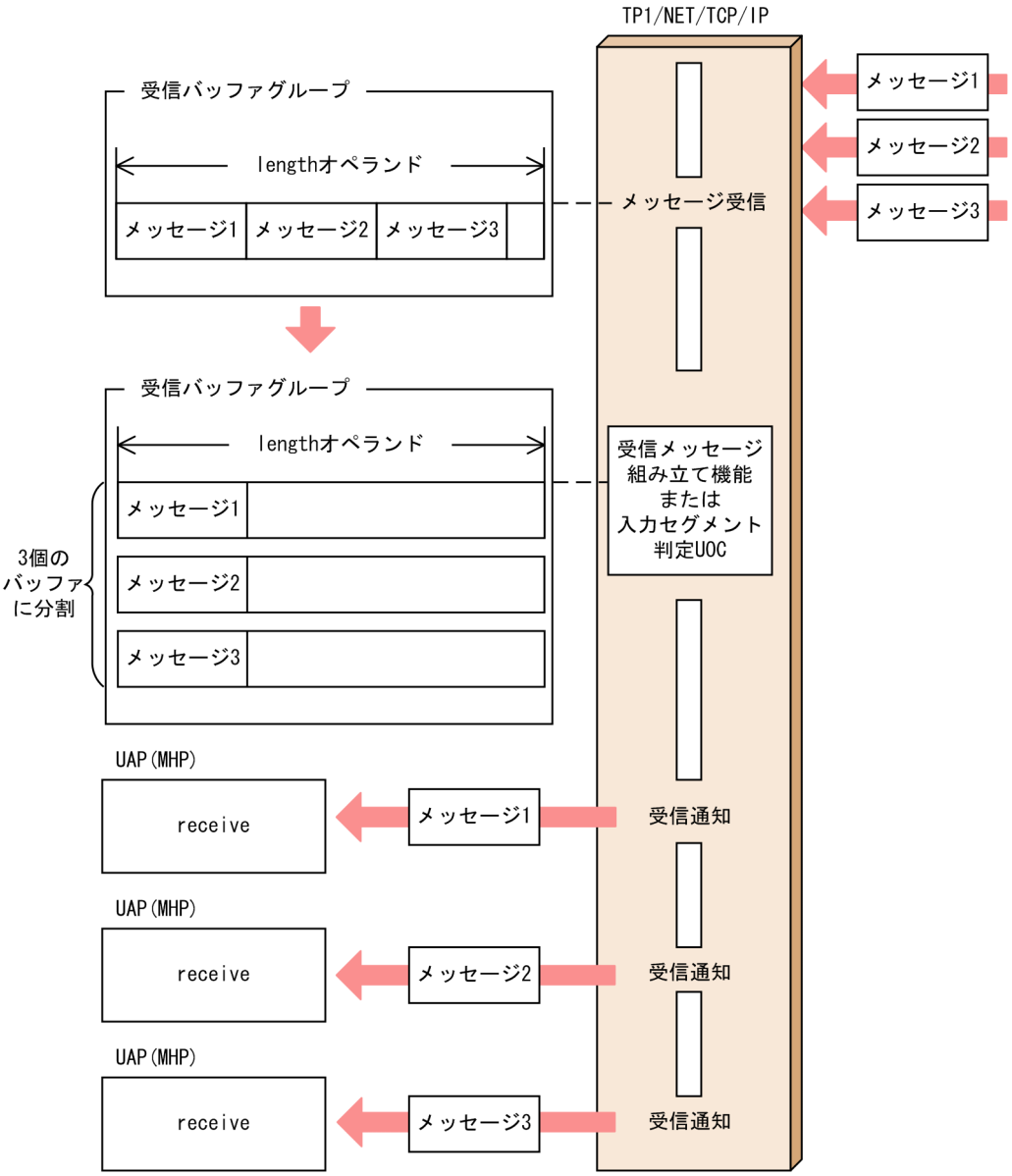
注※2

出力メッセージ編集 UOC でメッセージサイズを変更している場合，次の値を設定してください。

- mcftalccn -u delichk=use，nouse のとき
  出力メッセージ編集 UOC で変更したあとのメッセージサイズ，または最大論理メッセージ長
- mcftalccn -u delichk=dccm2m，dccm2s，dccm3m，dccm3s のとき
  出力メッセージ編集 UOC で変更したあとのメッセージサイズ，または（最大論理メッセージ長＋ 7）

注※3

- 相手システムから連続してメッセージを受信する場合，一つの受信バッファに複数のメッセージが格納されることがあります。受信メッセージ組み立て機能または入力セグメント判定 UOC で分割する場合，分割した数に応じた受信バッファが必要となります。
  一つの受信バッファに 3 個分のメッセージをまとめて受信した場合の例を次に示します。



- 相手システムから連続してメッセージを受信する形態では，受信メッセージの最大長よりも極端に大きい値を length オペランドに指定すると，一つの受信バッファに多数のメッセージが格納されることがあります。受信バッファに格納されたメッセージは，受信メッセージ組み立て機能または入力セグメント判定 UOC によって分割され，格納されているメッセージ数に応じた受信バッファを確保します。そのため，count オペランドにも大きい値を指定する必要がありますので，length オペランドには適切な値を設定するようにしてください。
- length オペランドよりも長い受信メッセージは受信できません。一つの受信メッセージを複数の受信バッファに分割して受信することもできません。

mcftalced（コネクション定義の終了）

形式

```
mcftalced
```

機能

コネクション定義の終了を示します。

オプション

ありません。

mcftalcle（論理端末定義）

形式

```
mcftalcle -l 論理端末名称
          〔-N ”modelname=モデル論理端末名称”〕
           -t any
          〔-m ”〔mmsgcnt=メモリ出力メッセージ最大格納数〕
               〔dmsgcnt=ディスク出力メッセージ最大格納数〕”〕
          〔-i auto｜manual〕
          〔-k ”〔quekind=memory｜disk〕
               〔quegrpid=キューグループID〕”〕
          〔-o ”aj=yes｜no”〕
          〔-v アプリケーション名〕
          〔-d ”replacemsg=send｜discard”〕
```

機能

論理端末に関する環境を定義します。

コネクション定義の開始（mcftalccn）で-A オプションを指定する場合，この定義は指定できません。

オプション

●-l 論理端末名称　　～〈1～8 文字の識別子〉

OpenTP1 システム内で，一意となる論理端末名称を指定します。

●-N

（オペランド）

modelname=モデル論理端末名称　　～〈1～8 文字の識別子〉

この論理端末定義で使用する定義情報を持つ（モデルとする），論理端末定義の論理端末名称を指定します。ただし，指定した論理端末名称に対する論理端末定義が，先に定義されていなければなりません。

このオペランドを指定したときは，-l オプション以外のオプション，またはオペランドを省略できます。

-l オプション以外のオプション，またはオペランドを省略した場合，modelname に指定した論理端末名称の定義指定値がすべて流用されます。

この論理端末定義に-N オプション以外のオプション，またはオペランドを指定した場合は，指定したオプション，またはオペランドの指定値が優先されます。

このオプションは 2 回以上指定できません。また，モデルとした論理端末と，-N オプション以外に指定したオプション，またはオペランドの組み合わせによっては，相関チェックによってエラーになる場合があります。そのため，-k オプションの quekind オペランドには，モデル定義と異なる指定をしないでください。

●-t any

この論理端末の端末タイプとして，次の値を指定します。

any

任意型論理端末

●-m

（オペランド）

mmsgcnt=メモリ出力メッセージ最大格納数　～〈符号なし整数〉((0～65535))《0》

メモリキューで待ち合わせをする出力メッセージの最大格納数を指定します。

出力メッセージの待ち合わせ数が指定した最大数になると，それ以後 UAP からの一方送信メッセージの送信要求（dc_mcf_send 関数または CBLDCMCF('SEND△△△△')）はエラーリターンとなります。

0 を指定した場合，または省略した場合，メモリキューで待ち合わせをする出力メッセージの数は指定可能な最大数（65535）になります。ただし，実際に待ち合わせをできる出力メッセージ数は動的共用メモリの容量に依存します。

dmsgcnt=ディスク出力メッセージ最大格納数　～〈符号なし整数〉((0～65535))《0》

ディスクキューで待ち合わせをする出力メッセージの最大格納数を指定します。

出力メッセージの待ち合わせ数が指定した最大数になると，それ以後 UAP からの一方送信メッセージの送信要求（dc_mcf_send 関数または CBLDCMCF('SEND△△△△')）はエラーリターンとなります。

0 を指定した場合，または省略した場合，ディスクキューで待ち合わせをする出力メッセージの数は指定可能な最大数（65535）になります。ただし，実際に待ち合わせをできる出力メッセージ数はメッセージキューファイルの容量に依存します。

●-i auto | manual　～《auto》

論理端末の閉塞解除方法を指定します。

auto

コネクション確立時，システムが自動的に論理端末を閉塞解除します。

manual

運用コマンド（mcftactle）の入力，または API（dc_mcf_tactle 関数もしくは CBLDCMCF('TACTLE△△')）の発行で論理端末を閉塞解除します。

●-k

（オペランド）

quekind=memory | disk　～《memory》

出力メッセージの割り当て先（メモリキューまたはディスクキュー）を指定します。

memory

メモリキューだけに割り当てます。

disk

ディスクキューおよびメモリキューに割り当てます。

disk を指定した場合，必ず quegrpid オペランドを指定してください。

quegrpid=キューグループ ID　～〈1～8 文字の識別子〉

ディスクで待ち合わせをする出力メッセージに使用するキューグループ ID を指定します。

MCF マネジャ定義の mcfmqgid コマンドで指定するキューグループ ID（キュー種別は otq）のどれかを指定してください。

このオペランドは，quekind オペランドで disk を指定した場合だけ指定します。

●-o

（オペランド）

aj=yes | no　～《yes》

メッセージ送信が完了した場合に，メッセージ送信完了ジャーナル（AJ）を取得するかどうかを指定します。

ただし，dc_mcf_sendsync 関数または dc_mcf_sendrecv 関数で送信したメッセージは，このオペランドの指定内容に関係なく，メッセージ送信完了ジャーナルを取得しません。

yes

メッセージ送信完了ジャーナルを取得します。

no

メッセージ送信完了ジャーナルを取得しません。

●-v アプリケーション名　～〈1～8 文字の識別子〉

入力メッセージを受信した場合に起動するアプリケーション名称（MHP）を指定します。アプリケーション属性定義（mcfaalcap -n name）で定義した名称を指定してください。アプリケーション属性定義の詳細については，マニュアル「OpenTP1 システム定義」を参照してください。このオプションを省略した場合，入力メッセージ編集 UOC で指定した値，または入力メッセージの先頭から空白の手前までの 8 バイト以内の値がアプリケーション名として仮定されます。アプリケーション名決定の優先順位は，「2.3.11 アプリケーション名の決定」を参照してください。

●-d

（オペランド）

replacemsg=send | discard　～《send》

MHP でメッセージを受信し，受信した論理端末と同一の論理端末に対してメッセージ送信をするまでの間にコネクションがいったん解放され再確立した場合，未送信となっていたメッセージを送信するかどうかを指定します。

send

コネクション再確立時に未送信となったメッセージを送信します。

discard

コネクション再確立時に未送信となったメッセージを送信しません。

ただし，次の場合は対象となりません。

- 入力元論理端末と異なる論理端末にメッセージを送信した場合
- mcfuevt で起動した MHP からメッセージを送信した場合
- SPP からメッセージを送信した場合

システムサービス情報定義

MCF サービスはユーザが作るシステムサービスで，OpenTP1 のシステムサービスと同じ位置づけになります。

システムサービス情報定義では，MCF 通信サービスを起動するための環境を定義します。ユーザが MCF サービスを作成するときに定義する必要があります。

システムサービス情報定義は，テキストエディタを使用して作成します。

システムサービス情報定義の完全パス名を次に示します。

```
$DCDIR/lib/sysconf/定義ファイル名
```

定義ファイル名には，システムサービス情報定義の module オペランドで指定する実行形式プログラム名を指定します。この定義ファイル名を MCF マネジャ定義の mcfmcname コマンドに指定します。

形式

set 形式

```
 set module=”TP1/NET/TCP/IPの実行形式プログラム名”
〔set mcf_prf_trace=Y|N〕
```

機能

プロセスサービスが MCF 通信サービスを起動するための環境を定義します。

各 MCF 通信サービスに対して一つ，システムサービス情報定義を作成できます。また，複数の MCF 通信サービスで一つのシステムサービス情報定義を共用することもできます。

説明

set 形式のオペランド

set 形式のオペランドについて次に示します。

●module="TP1/NET/TCP/IP の実行形式プログラム名"　～〈1～8 文字の識別子〉

MCF 通信サービスを起動するための実行形式プログラム名を指定します。

MCF 実行形式プログラムには，MCF 通信プロセスのためのものとアプリケーション起動プロセスのためのものがあります。

MCF 実行形式プログラムは，MCF 通信プロセス同士，アプリケーション起動プロセス同士で共有できます。

TP1/NET/TCP/IP の実行形式プログラム名には，先頭 4 文字が mcfu で始まる最大 8 文字の名称を指定します。

●mcf_prf_trace=Y|N　～《Y》

MCF 通信サービスごとに，MCF 性能検証用トレース情報を取得するかどうかを指定します。このオペランドの指定値を有効にするには，システムサービス共通情報定義の mcf_prf_trace_level オペランドに 00000001 を指定してください。

Y

MCF 性能検証用トレース情報を取得します。

N

MCF 性能検証用トレース情報を取得しません。

MCF 通信サービスでの MCF 性能検証用トレース情報取得有無とオペランドの指定値の関係を，次の表に示します。

システムサービス共通情報定義 mcf_prf_trace_level オペランドの指定値	システムサービス情報定義 mcf_prf_trace オペランドの指定値	
	Y	N
00000000	取得しない	取得しない
00000001	取得する	取得しない

このオペランドの使用は，TP1/Extension 1 をインストールしていることが前提です。TP1/Extension 1 をインストールしていない場合の動作は保証できません。

システムサービス共通情報定義

TP1/NET/TCP/IP で定義したシステム構成の内容（使用するコネクション数など）によっては，OpenTP1 のシステムサービス共通情報定義を指定する必要があります。

システムサービス共通情報定義の完全パス名を次に示します。

```
$DCDIR/lib/sysconf/mcf
```

形式

set 形式

```
  set max_socket_descriptors=ソケット用ファイル記述子の最大数
  set max_open_fds=MCF通信プロセスでアクセスするファイルの最大数
〔set mcf_prf_trace_level=MCF性能検証用トレース情報の取得レベル〕
```

機能

システムサービス共通情報定義では，複数の MCF 通信サービスに共通する情報を定義します。この定義ファイルは，標準値を定義した状態で製品に含まれています。次に示すオペランドについては，必要に応じて，テキストエディタを使用して定義値を変更してください。ほかのオペランドについては，変更しないでください。

説明

set 形式のオペランド

この定義には，ほかにもオペランドがあります。詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

●max_socket_descriptors=ソケット用ファイル記述子の最大数　～〈符号なし整数〉((64～3596))

各 MCF 通信プロセスでソケット用に使用するファイル記述子数の中の最大値を指定します。

OpenTP1 制御下のプロセスでは，システムサーバやユーザサーバとの間で，ソケットを使用した TCP/IP 通信でプロセス間の情報交換をしています。そのため，同時に稼働する UAP プロセスの数などによって，ソケット用のファイル記述子の最大数を変更する必要があります。

各 MCF 通信プロセスまたはアプリケーション起動プロセスが使用するソケット用ファイル記述子の最大数を求める計算式を次に示します。

```
↑（このMCF通信プロセスに対してメッセージ送信要求を行うUAPプロセス数※1
  ＋システムサービスプロセス数※2
  ＋このMCF通信プロセスまたはアプリケーション起動プロセスに対して同時に処理要求を行う運用コマンド数
  ）／0.8↑
```

（凡例）

↑↑：小数点以下を切り上げます。

注※1

アプリケーション起動プロセスに対するアプリケーション起動要求を行う UAP プロセス数も含みます。

注※2

システムサービスプロセス数とは，自 OpenTP1 内のシステムサービスプロセス数です。自 OpenTP1 内のシステムサービスプロセスは，rpcstat コマンドで表示されるサーバ名をカウントすることで求められます。rpcstat コマンドで表示されるサーバ名のうち，マニュアル「OpenTP1 解説」の OpenTP1 のプロセス構造に記載されているシステムサービスプロセスをカウントしてください。

自 OpenTP1 内の各 MCF 通信プロセスおよびアプリケーション起動プロセスごとに計算し，その結果の中で最大値が 64 より大きい場合は，その値を指定します。64 以下の場合は，64 を指定します。

このオペランドの指定値が小さいと，OpenTP1 制御下の他プロセスとのコネクションが設定できなくなるため，プロセスが KFCA00307-E メッセージを出力して異常終了します。

●max_open_fds=MCF 通信プロセスでアクセスするファイルの最大数　～〈符号なし整数〉((500～4032))

各 MCF 通信プロセスでアクセスするファイル数の中の最大値を指定します。

MCF 通信プロセスが行うメッセージの送受信にもファイル記述子が使われます。この数が不足すると，コネクションの確立ができないなどの障害が発生するため，事前に必要となるファイル記述子の数を設定しておく必要があります。

各 MCF 通信プロセスが使用するファイル記述子の最大数を求める計算式を次に示します。

```
（プロトコル制御で使用するファイル記述子数※1）
＋MCFメイン関数でユーザが使用するファイル記述子数
＋30※2
```

注※1

TP1/NET/TCP/IP の場合，MCF 通信構成定義に定義したコネクションの総数，サーバ型コネクションの自システムのホストの IP アドレス（ホスト名），およびポート番号の組み合わせ数を加算した値になります。実際に通信を行うコネクションの総数ではありませんので，注意してください。

TP1/NET/TCP/IP で使用するファイル記述子数を求める計算式を次に示します。

```
TP1/NET/TCP/IPで使用するファイル記述子数＝
MCF通信構成定義に定義したコネクションの総数＋
サーバ型コネクションの自システムのホストのIPアドレス（ホスト名）とポート番号の組み合わせ数
```

自システムのホストの IP アドレスとポート番号の組み合わせ数の数え方の例を次に示します。

例 1，3 の場合：2

例 2 の場合：1

例 1

```
mcftalccn -y ”mode=server” -r ”ipaddr=11.11.11.11 portno=10000”
mcftalccn -y ”mode=server” -r ”ipaddr=11.11.11.11 portno=10001”
```

例 2

```
mcftalccn -y ”mode=server” -r ”ipaddr=11.11.11.11 portno=10000”
mcftalccn -y ”mode=server” -r ”ipaddr=11.11.11.11 portno=10000”
```

例 3

```
mcftalccn -y ”mode=server” -r ”ipaddr=11.11.11.11 portno=10000”
mcftalccn -y ”mode=server” -r ”ipaddr=22.22.22.22 portno=10000”
```

注※2

MCF 通信プロセスが扱う定義ファイルなどの数の最大値です。

自 OpenTP1 内の MCF 通信プロセスごとに計算し，その結果の中で最大値が 500 より大きい場合は，その値を指定します。500 以下の場合は，500 を指定します。指定値を超えてファイルのアクセスが発生した場合，その超過分はソケット用ファイル記述子使用数として扱われます。この場合，「max_socket_descriptors オペランドの指定値－max_open_fds オペランドの指定値の超過分」が，実際のソケット用ファイル記述子の最大数になりますので，ご注意ください。

max_socket_descriptors オペランドと max_open_fds オペランドには次の条件を満たす値を指定してください。

```
（「max_socket_descriptorsオペランドの指定値」
＋「max_open_fdsオペランドの指定値」）
≦4096
```

TP1/NET/TCP/IP の MCF 通信プロセス以外のプロセスで，max_socket_descriptors オペランドと max_open_fds オペランドの和が 1 プロセスで使用できるファイル記述子の最大数を超えている場合，各プロセスで使用できるファイル記述子数は，4096 よりも小さい値に強制的に補正されます。

1 プロセスで使用できるファイル記述子の最大数を次の表に示します。

プロセス	OS	1 プロセスで使用できるファイル記述子の最大数
TP1/NET/TCP/IP の MCF 通信プロセス	すべて	4096
上記以外	AIX，HP-UX，Windows	2048
	Linux，Solaris	1024

max_socket_descriptors オペランドと max_open_fds オペランドの和が 1 プロセス当たりでオープンできるファイル数の物理限界値（ハードリミット）を超えていたとき，MCF の開始を中断します。

●mcf_prf_trace_level=MCF 性能検証用トレース情報の取得レベル　～((00000000～00000001))《00000001》

MCF 性能検証用トレース情報の取得レベルを指定します。MCF 性能検証用トレースを取得する場合は，システム共通定義の prf_trace オペランドに Y を指定するか，または省略してください。

00000000

MCF 性能検証用トレース情報を取得しません。

00000001

MCF 性能検証用トレース情報（イベント ID：0xa000～0xa0ff）を取得します。イベント ID の詳細については，マニュアル「OpenTP1 運用と操作」を参照してください。また，TP1/NET/TCP/IP 固有の出力情報や取得タイミングについては，「付録 H　MCF 性能検証用トレースの取得」を参照してください。

オペランドの指定に誤りがある場合は，OpenTP1 開始処理中に OpenTP1 が異常終了します。

このオペランドの使用は，TP1/Extension 1 をインストールしていることが前提です。TP1/Extension 1 をインストールしていない場合の動作は保証できません。

注意事項

max_socket_descriptors オペランドの指定値と max_open_fds オペランドの指定値の合計は，OS のシステムパラメタで指定する「1 プロセスでオープンできるファイル数」を超えないようにする必要があります。超える場合は，OS のシステムパラメタの指定を変更してください。

MCF 定義オブジェクトの生成

MCF 定義オブジェクト生成ユティリティでは，MCF の定義ファイルの構文のチェックと定義オブジェクトファイルへの変換をします。ここでは，MCF 定義オブジェクト生成ユティリティの起動コマンドについて説明します。

形式

```
mcftcp -i 〔パス名〕入力ファイル名
       -o 〔パス名〕出力オブジェクトファイル名
      〔-r {no | rep} 〕
```

機能

MCF 通信構成定義の TP1/NET/TCP/IP のプロトコル固有定義ファイルの構文をチェックし，定義オブジェクトファイルを作成します。

ただし，開始から再開始の間に定義オブジェクトファイルを変更しないでください。変更した場合，再開始時に正常に動作しないことがありますのでご注意ください。

TP1/NET/TCP/IP のプロトコル固有定義オブジェクトファイル以外の生成ユティリティについては，マニュアル「OpenTP1 システム定義」を参照してください。

オプション

●-i 〔パス名〕入力ファイル名　～〈パス名〉〈1～8 文字の識別子〉

定義ソースが格納されているファイル名を指定します。

●-o 〔パス名〕出力オブジェクトファイル名　～〈パス名〉〈1～8 文字の英数字〉

定義オブジェクトを格納するファイル名を指定します。

次に示す条件を満たした名称を指定してください。

- 先頭 3 文字が_mu で始まる最大 8 文字の名称
- 通信サービス定義（mcfmcname -s）の mcfsvname オペランドで指定する MCF 通信サーバ名

●-r {no | rep}　～《no》

定義オブジェクトファイルの出力先に読み取り権限を持つファイルがすでに存在する場合，定義オブジェクトファイルを上書きするかどうかを指定します。

no

定義オブジェクトファイルを上書きしないで，KFCA10332-E メッセージを出力します。

rep

定義オブジェクトファイルを上書きします。

MCF 定義オブジェクトの解析

対応する定義ソースが不明となった MCF 定義オブジェクトファイルの内容を知りたい場合に，MCF 定義オブジェクトの解析をします。ここでは，MCF 定義オブジェクト解析コマンドについて説明します。

形式

```
mcftcpr -i 〔パス名〕解析対象オブジェクトファイル名
```

機能

次の定義オブジェクトを解析し，定義ソースの形式で標準出力します。

- TP1/NET/TCP/IP のプロトコル固有定義オブジェクトファイル

オプション

●-i 〔パス名〕解析対象オブジェクトファイル名　～〈1～8 文字の英数字〉

定義オブジェクトが格納されているファイル名を指定します。

出力形式

```
#################################################################

        MCF communication configuration definition
                    TCP/IP definition

#################################################################

   OBJECT FILE NAME : xxxxxxxx
   VV-RR            : vv-rr
   DATE             : yyyy-mm-dd hh:mm:ss

#################################################################

mcftalccn
   -c            = cn01
   -p            = tcp
   -g sndbuf     = 1
   -g rcvbuf     = 2
   -e msgbuf     = 3
   -e count      = 1
   -i            = manual
   -t            = tcp
   -y mode       = client
   -b bretry     = yes
   -b bretrycnt  = 10
   -b bretryint  = 5
   -b dretry     = yes
```

```
   -b dretrycnt  = 0
   -b dretryint  = 10000
   -b concmptim  = 0
   -b sndcmptim  = 0
   -r portno     = 10001
   -r hostname   = host01
  *-r backlog    = 0

.
.
.

 mcftalcle
   -l            = le01
   -t            = any
   -m mmsgcnt    = 0
   -m dmsgcnt    = 0
   -i            = manual

.
.
.

 mcftalced
```

```
####################### End Of File ########################
```

解析結果

定義オブジェクト解析コマンドは，その解析結果を定義ソースの形式で出力します。出力される内容は解析結果であり，記述形式は元の定義ソースの記述形式とは一致しません。定義ソースと定義オブジェクト解析結果の差異を次に示します。

表 6-3　定義ソースと定義オブジェクト解析結果の差異

項目	定義ソース	定義オブジェクト解析結果
注釈文	書き込みできる。	出力しない。
省略値の扱い	省略できる。	• 限定公開部分も含めて，省略値を出力する。 • システム環境定義の default_value_option オペランドの指定値で省略値が変更となるオペランドの場合，行の先頭に"@"を付与する。
限定公開部分の表記方法	一般公開部分と差異なし。	バージョン 7 での限定公開機能の行の先頭に，"*"を付与する。
定義コマンド名とオプションの表記方法	1 行に表記できる。 (例) mcftalccn -c cn01	定義コマンド名を表記後，改行する。また，オプションに"="を付記する。 (例) mcftalccn -c = cn01

項目	定義ソース	定義オブジェクト解析結果
1 定義コマンドが複数の行にわたる場合	継続記号"￥"を付与する。 (例) mcftalccn -c cn01 ￥ -p tcp	継続記号は出力しない。 (例) mcftalccn -c = cn01 -p = tcp
1 定義オプションに複数のオペランドを指定する場合	複数のオペランドをまとめて二重引用符(")で囲む。 (例) mcftalccn -g "sndbuf=1 rcvbuf=2"	個々のオペランドに対してオプションを付記する。 (例) mcftalccn -g sndbuf = 1 -g rcvbuf = 2
その他	なし	• ファイル名などを記したタイトルが出力される。 • 定義オブジェクト生成時の補正によって，実際の指定値とは異なる内容が出力される場合がある。 • 定義ソースと該当コマンドのバージョンの差異によって，解析結果にサポート内容の過不足がある場合がある。

注意事項

解析対象が不正であった場合は，正常に動作しないことがあります。

コネクションの形態と MCF 通信構成定義との関係

コネクションの形態と MCF 通信構成定義との関係を次の表に示します。

表 6-4　コネクションの形態と MCF 通信構成定義との関係

コマンド	オプション	オペランド	クライアント型	サーバ型
mcftalccn（コネクション定義の開始）	-c	−	○	○
	-N	modelname	△	△
	-p	−	○	○
	-g	sndbuf	△	△
		rcvbuf	△	△
	-e	msgbuf	△	△
		count	△	△
	-i	−	△	×
	-b	bretry	△	△
		bretrycnt	△	△
		bretryint	△	△
		concmptim	△	×
		sndcmptim	△	△
	-w	srtimout	△	△
	-t	−	○	○
	-y	mode	○	○
	-r	ipaddr※1	△	△
		hostname※1	△	△
		portno※1	△	○
	-o	oipaddr※1，※2	○	○※3
		ohostname※1，※2	○	○※3
		oportno※1	○	○※3
	-k	keepalive	△	△
		nodelay	△	△
		notrftime	△	△
	-f	kind	△	△

コマンド	オプション	オペランド	クライアント型	サーバ型
mcftalccn（コネクション定義の開始）	-f	cnrelease	△	△
		releaselog	△	△
		cnerrlog	△	△
	-A	mastercn	△	×
	-u	masm	△	△
		ntimer	△	△
		ntime	△	△
		delichk	△	△
		msghold	△	△
		holdlimit	△	△
	-h	addrchk	×	△
		chgconn	×	△
		listen	×	△
	-C	lscnfmt	△	△
	-l	replymsg	△	△
		cnassign	×	△
mcftalcle（論理端末定義）	-l	−	○	○
	-N	modelname	△	△
	-t	−	○	○
	-m	mmsgcnt	△	△
		dmsgcnt	△	△
	-i	−	△	△
	-k	quekind	△	△
		quegrpid	△	△
	-o	aj	△	△
	-v	−	△	△
	-d	replacemsg	△	△

（凡例）

−：該当する内容がないことを表します。

○：指定が必要です。

△：指定は任意です。

×：指定できません，または，無効となります。

注※1

すべてのコネクションで，これらのオペランド指定値が完全に重複しないように設定してください。

複数のコネクションを使用する場合は，これらのオペランド指定値のうち，一つでも指定値が異なるように設定してください。

注※2

これらのオペランドは，どちらか一つだけを設定してください。

注※3

-h オプションの addrchk オペランドの指定値が no の場合，指定値は無効（×）になります。

交代用コネクションと MCF 通信構成定義との関係を次の表に示します。

表 6-5　交代用コネクションと MCF 通信構成定義との関係

コマンド	オプション	オペランド	交代用コネクションでの指定値
mcftalccn（コネクション定義の開始）	-c	−	◇
	-N	modelname	△
	-p	−	○
	-g	sndbuf	○
		rcvbuf	○
	-e	msgbuf	○
		count	○
	-i	−	○
	-b	bretry	○
		bretrycnt	○
		bretryint	○
		concmptim	○
		sndcmptim	○
	-w	srtimout	○
	-t	−	○
	-y	mode	○
	-r	ipaddr※	△
		hostname※	△
		portno※	△
	-o	oipaddr※	△
		ohostname※	△
		oportno※	△

コマンド	オプション	オペランド	交代用コネクションでの指定値
mcftalccn（コネクション定義の開始）	-k	keepalive	○
		nodelay	○
		notrftime	○
	-f	kind	○
		cnrelease	○
		releaselog	○
		cnerrlog	○
	-A	mastercn	◎
	-u	masm	○
		ntimer	○
		ntime	○
		delichk	○
		msghold	○
		holdlimit	○
	-h	addrchk	×
		chgconn	×
		listen	×
	-C	lscnfmt	○
	-l	replymsg	○
		cnassign	×

（凡例）

－：該当する内容がないことを表します。

◎：現用コネクション ID を指定する必要があります。

○：現用コネクションの指定と一致する必要があります。

◇：OpenTP1 システムで一意にする必要があります。

△：任意の値を指定できます。

×：指定できません，または，無効となります。

注※

現用コネクションを除いたすべてのコネクションで，これらのオペランド指定値が完全に重複しないように設定してください。複数のコネクションを使用する場合は，これらのオペランド指定値のうち，一つでも指定値が異なるように設定してください。

アプリケーションプログラムとシステム環境設定の関連

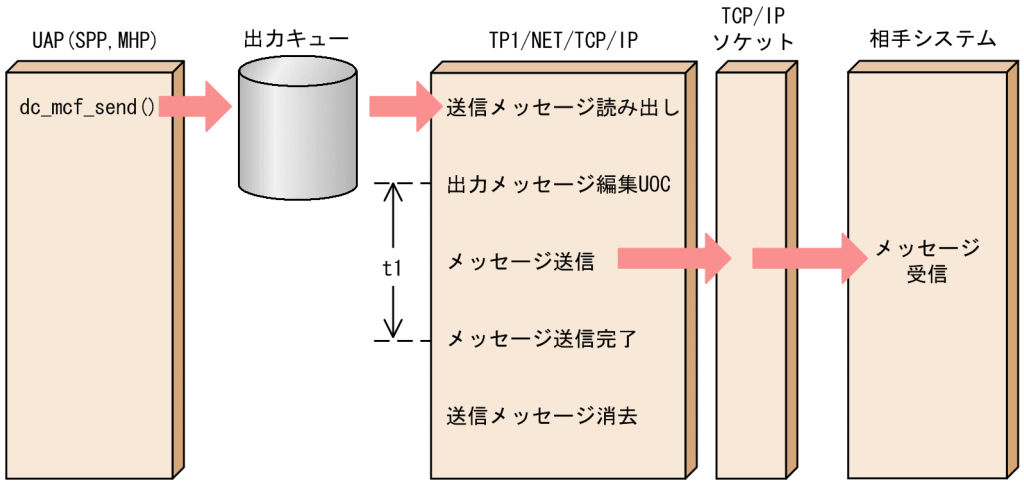
UAP とシステム環境設定の関係について説明します。

メッセージ送受信の監視時間

一方送信メッセージの送信と各種タイマ監視値の関係

メッセージ送達確認機能を使用しない場合の一方送信メッセージの送信と，各種タイマ監視値の関係を次の図に示します。

図 6-3　一方送信メッセージの送信と各種タイマ監視値の関係（メッセージ送達確認機能を使用しない場合）



図中の t1 について次に説明します。

t1：メッセージ送信完了監視時間

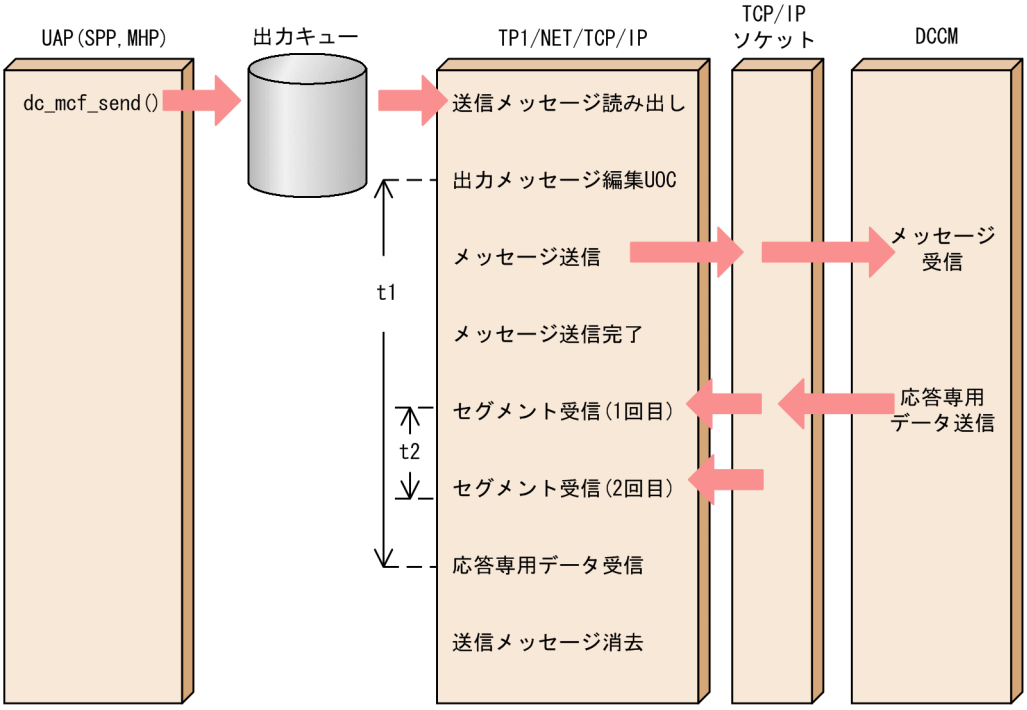
出力メッセージ編集 UOC のリターン直後からメッセージの送信完了（TCP/IP ソケットの送信バッファへのメッセージ書き込み完了）までの時間を監視します。

コネクション定義（mcftalccn -b）の sndcmptim オペランドの指定値が監視時間の最大値となります（省略または 0 を指定した場合は時間監視をしません）。

タイムアウト発生時には，KFCA14815-E メッセージが出力され，論理端末を閉塞し，CERREVT を起動します。

DCCM とのメッセージ送達確認機能を使用する場合の一方送信メッセージの送信と各種タイマ監視値の関係を次の図に示します。

図 6-4　一方送信メッセージの送信と各種タイマ監視値の関係（DCCM とのメッセージ送達確認機能を使用する場合）



図中の t1，t2 について次に説明します。

t1：メッセージ送信完了監視時間

出力メッセージ編集 UOC のリターン直後から応答専用データを受信するまでの時間を監視します。

コネクション定義（mcftalccn -b）の sndcmptim オペランドの指定値が監視時間の最大値となります（省略または 0 を指定した場合は時間監視をしません）。

メッセージ送信完了前にタイムアウトが発生した場合は，KFCA14815-E メッセージが出力され，論理端末を閉塞し，CERREVT を起動します。メッセージ送信完了から応答専用データ受信前にタイムアウトが発生した場合は，KFCA14815-E メッセージが出力され，コネクションを解放し，CERREVT を起動します。

t2：後続セグメント受信監視時間

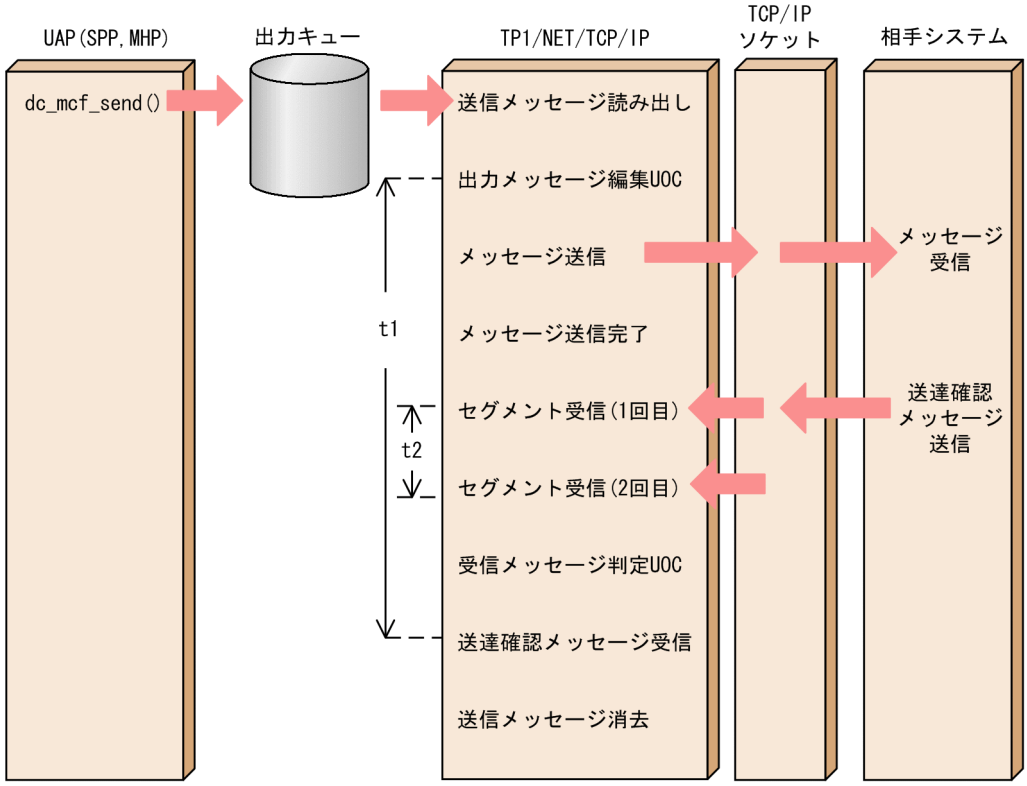
応答専用データがネットワーク上で分割された場合，未完成のセグメントを受信してから次のセグメントを受信するまでの時間を監視します。

コネクション定義（mcftalccn -u）の ntime オペランドの指定値が監視時間の最大値となります（コネクション定義（mcftalccn -u）の ntimer オペランドに no を指定した場合は時間監視をしません）。

タイムアウト発生時には，KFCA14823-W メッセージが出力され，コネクションを解放し，CERREVT を起動します。

任意の相手システムとのメッセージ送達確認機能を使用する場合の一方送信メッセージの送信と各種タイマ監視値の関係を次の図に示します。

図 6-5　一方送信メッセージの送信と各種タイマ監視値の関係（任意の相手システムとのメッセージ送達確認機能を使用する場合）



図中の t1，t2 について次に説明します。

t1：メッセージ送信完了監視時間

出力メッセージ編集 UOC のリターン直後から送達確認メッセージを受信するまでの時間を監視します。

コネクション定義（mcftalccn -b）の sndcmptim オペランドの指定値が監視時間の最大値となります（省略または 0 を指定した場合は時間監視をしません）。

メッセージ送信完了前にタイムアウトが発生した場合は，KFCA14815-E メッセージが出力され，論理端末を閉塞し，CERREVT を起動します。メッセージ送信完了から送達確認メッセージ受信前にタイムアウト発生した場合は，KFCA14815-E メッセージが出力され，コネクションを解放し，CERREVT を起動します。

t2：後続セグメント受信監視時間

送達確認メッセージがネットワーク上で分割された場合，未完成のセグメントを受信してから次のセグメントを受信するまでの時間を監視します。

コネクション定義（mcftalccn -u）の masm オペランドの指定によって，監視時間の最大値が異なります。監視時間を次の表に示します。

表 6-6　後続セグメント受信監視時間の最大値

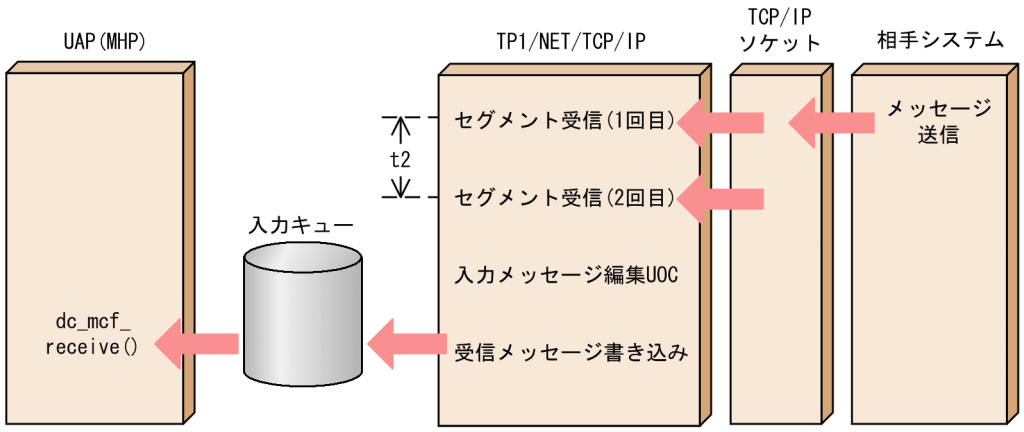
コネクション定義（mcftalccn -u）の masm オペランドの指定値	コネクション定義（mcftalccn -u）の ntimer オペランドの指定値	入力セグメント判定 UOC の dctcp_uoctimer_inf 構造体の timer_code の指定値	監視時間
yes	yes または省略	任意	コネクション定義（mcftalccn -u）の ntime オペランドの指定値
	no		時間監視をしません
no または省略	任意	DCTCP_TIME_SET	入力セグメント判定 UOC の dctcp_uoctimer_inf 構造体の timer_value の指定値
		DCTCP_TIME_NO_SET	時間監視をしません

タイムアウト発生時には，KFCA14823-W メッセージが出力され，コネクションを解放し，CERREVT を起動します。

一方送信メッセージの受信と各種タイマ監視値の関係

メッセージ送達確認機能を使用しない場合の一方送信メッセージの受信と各種タイマ監視値の関係を次の図に示します。

図 6-6　一方送信メッセージの受信と各種タイマ監視値の関係（メッセージ送達確認機能を使用しない場合）



図中の t2 について次に説明します。

t2：後続セグメント受信監視時間

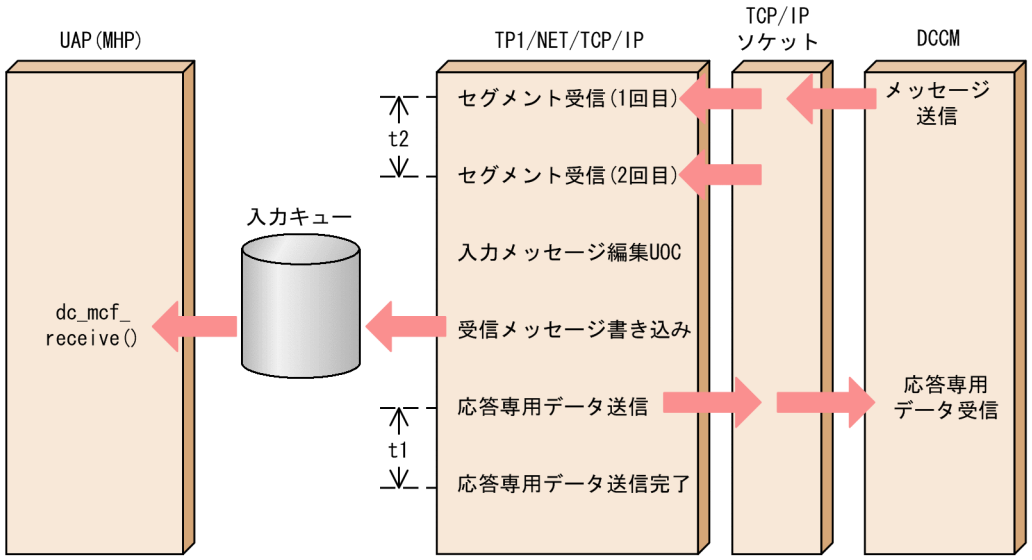
メッセージがネットワーク上で分割された場合，未完成のセグメントを受信してから次のセグメントを受信するまでの時間を監視します。

コネクション定義（mcftalccn -u）の masm オペランドの指定によって，監視時間の最大値が異なります。監視時間の詳細については，表 6-6 を参照してください。

タイムアウト発生時には，KFCA14823-W メッセージが出力され，コネクションを解放し，CERREVT を起動します。

DCCM とのメッセージ送達確認機能を使用する場合の一方送信メッセージの受信と各種タイマ監視値の関係を次の図に示します。

図 6-7　一方送信メッセージの受信と各種タイマ監視値の関係（DCCM とのメッセージ送達確認機能を使用する場合）



図中の t1，t2 について次に説明します。

t1：メッセージ送信完了監視時間

受信メッセージを入力キューに書き込んだ直後から応答専用データの送信完了（TCP/IP ソケットの送信バッファへのメッセージ書き込み完了）までの時間を監視します。

コネクション定義（mcftalccn -b）の sndcmptim オペランドの指定値が監視時間の最大値となります（省略または 0 を指定した場合は時間監視をしません）。

タイムアウト発生時には，KFCA14815-E メッセージが出力され，コネクションを解放し，CERREVT を起動します。

t2：後続セグメント受信監視時間

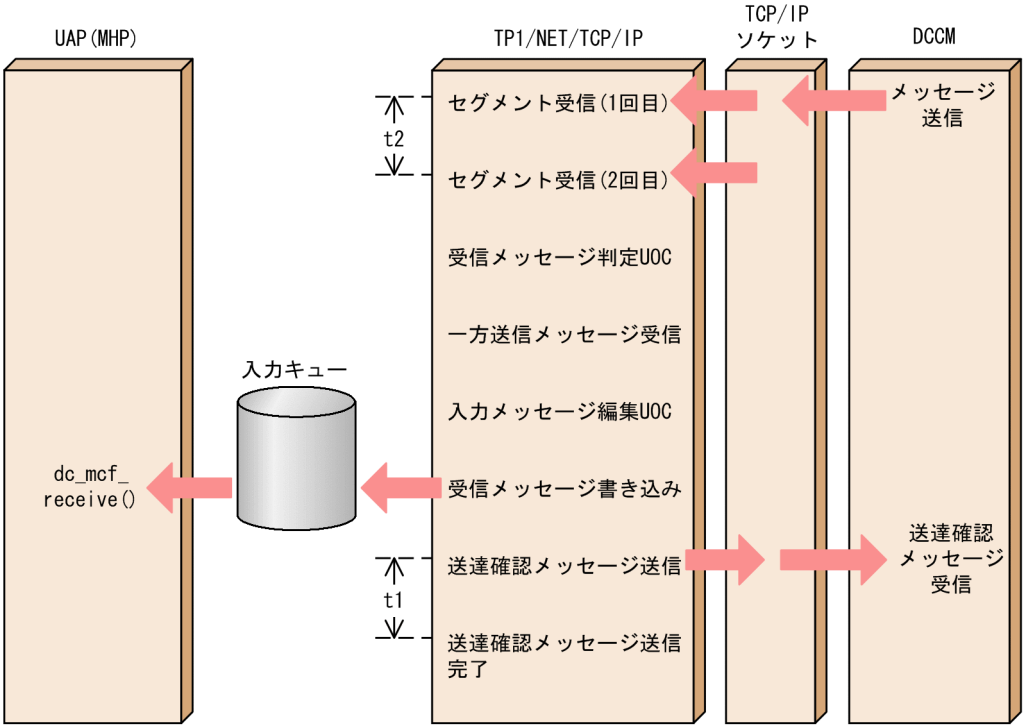
メッセージがネットワーク上で分割された場合，未完成のセグメントを受信してから次のセグメントを受信するまでの時間を監視します。

コネクション定義（mcftalccn -u）の ntime オペランドの指定値が監視時間の最大値となります（コネクション定義（mcftalccn -u）の ntimer オペランドに no を指定した場合は時間監視をしません）。

タイムアウト発生時には，KFCA14823-W メッセージが出力され，コネクションを解放し，CERREVT を起動します。

任意の相手システムとのメッセージ送達確認機能を使用する場合の一方送信メッセージの受信と各種タイマ監視値の関係を次の図に示します。

図 6-8　一方送信メッセージの受信と各種タイマ監視値の関係（任意の相手システムとのメッセージ送達確認機能を使用する場合）



図中の t2 については，図 6-6 の説明を参照してください。t1 について次に説明します。

t1：メッセージ送信完了監視時間

受信メッセージを入力キューに書き込んだ直後から送達確認メッセージの送信完了（TCP/IP ソケットの送信バッファへのメッセージ書き込み完了）までの時間を監視します。

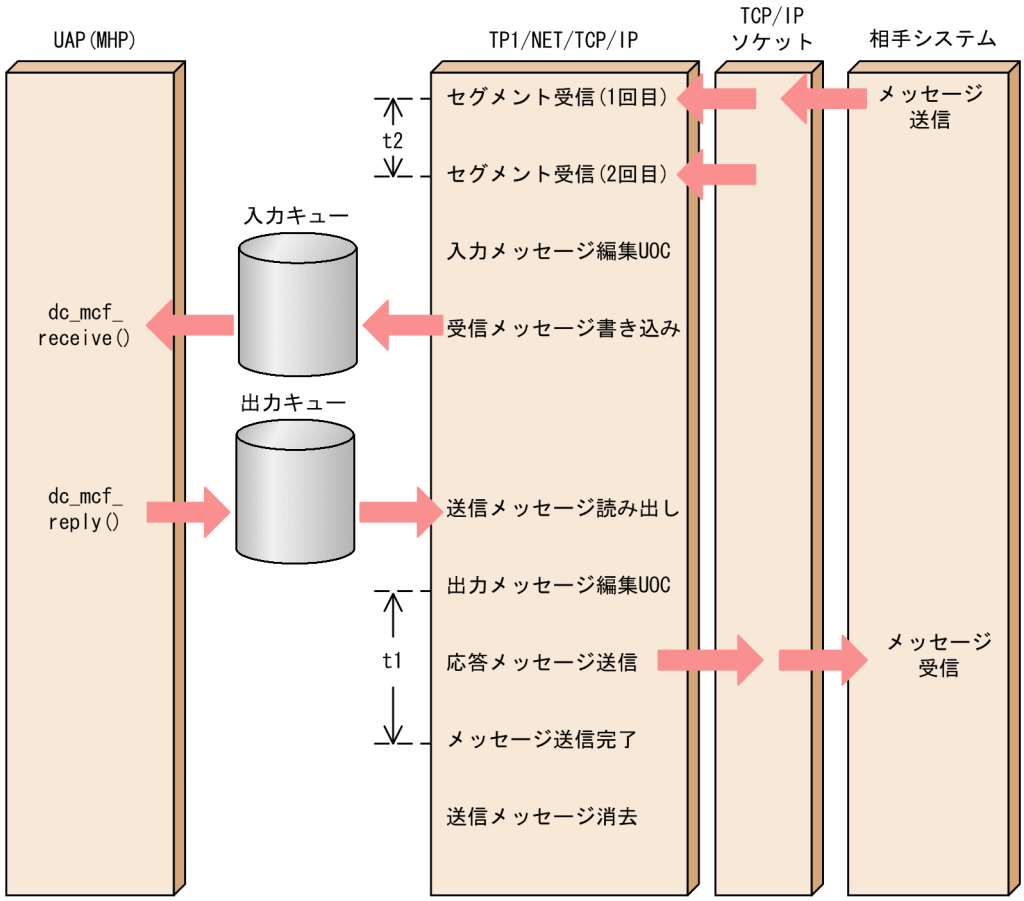
コネクション定義（mcftalccn -b）の sndcmptim オペランドの指定値が監視時間の最大値となります（省略または 0 を指定した場合は時間監視をしません）。

タイムアウト発生時には，KFCA14815-E メッセージが出力され，コネクションを解放し，CERREVT を起動します。

問い合わせメッセージの受信と各種タイマ監視値の関係

問い合わせメッセージの受信と各種タイマ監視値の関係を次の図に示します。

図 6-9　問い合わせメッセージの受信と各種タイマ監視値の関係



図中の t2 については，図 6-6 の説明を参照してください。t1 について次に説明します。

t1：メッセージ送信完了監視時間

出力メッセージ編集 UOC のリターン直後からメッセージの送信完了（TCP/IP ソケットの送信バッファへのメッセージ書き込み完了）までの時間を監視します。

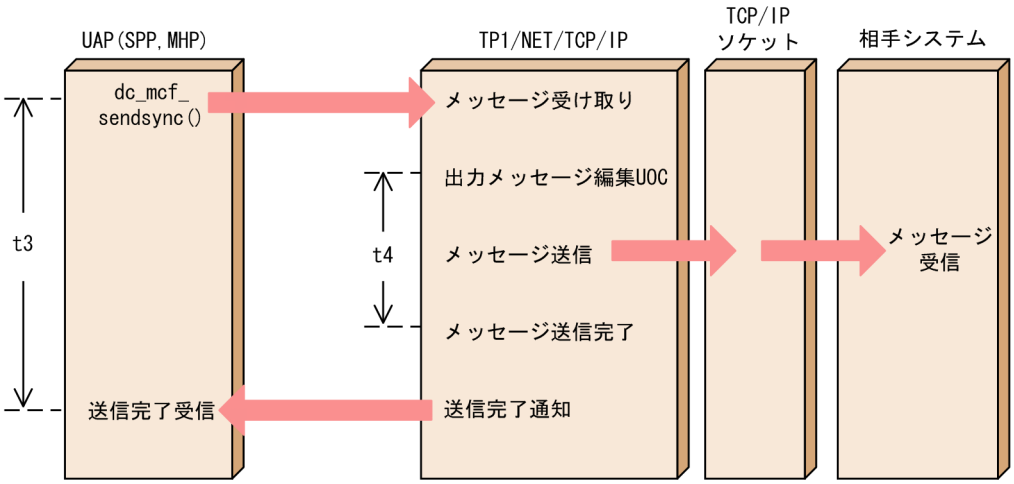
コネクション定義（mcftalccn -b）の sndcmptim オペランドの指定値が監視時間の最大値となります（省略または 0 を指定した場合は時間監視をしません）。

タイムアウト発生時には，KFCA14815-E メッセージが出力され，コネクションを解放し，CERREVT を起動します。

同期型メッセージの送信と各種タイマ監視値の関係

同期型メッセージの送信と各種タイマ監視値の関係を次の図に示します。

図 6-10　同期型メッセージの送信と各種タイマ監視値の関係



図中の t3，t4 について次に説明します。

t3：同期型送信監視時間(UAP)

dc_mcf_sendsync 関数を発行してから TP1/NET/TCP/IP から送信完了を受信するまでの時間を監視します。

監視時間の最大値を次の表に示します。

表 6-7　同期型送信監視時間（UAP）の最大値

dc_mcf_sendsync 関数の watchtime 引数の指定値	UAP 共通定義（mcfmuap -t）の sndtim オペランドの指定値	監視時間
プラス値	任意	dc_mcf_sendsync 関数の watchtime 引数の指定値に 60 を加算した値
0	0 以外	UAP 共通定義（mcfmuap -t）の sndtim オペランドの指定値に 60 を加算した値
	0 または省略	時間監視をしません
マイナス値	任意	時間監視をしません

タイムアウト発生時には，dc_mcf_sendsync 関数が DCMCFRTN_73001（-14001）でエラーリターンします。

t4：同期型送信監視時間（TP1/NET/TCP/IP）

出力メッセージ編集 UOC のリターン直後からメッセージの送信完了（TCP/IP ソケットの送信バッファへのメッセージ書き込み完了）までの時間を監視します。

監視時間の最大値を次の表に示します。

表 6-8　同期型送信監視時間（TP1/NET/TCP/IP）の最大値

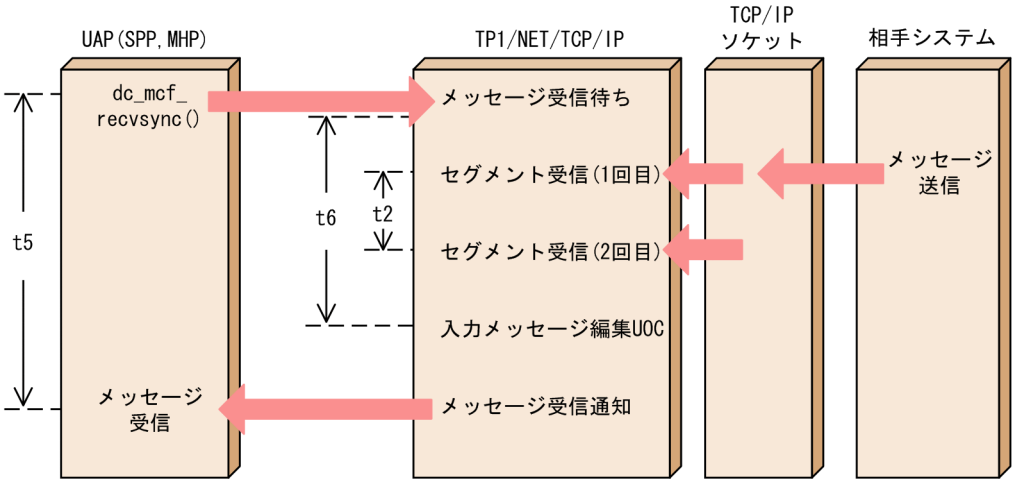
dc_mcf_sendsync 関数の watchtime 引数の指定値	UAP 共通定義（mcfmuap -t）の sndtim オペランドの指定値	監視時間
プラス値	任意	dc_mcf_sendsync 関数の watchtime 引数の指定値
0	0 以外	UAP 共通定義（mcfmuap -t）の sndtim オペランドの指定値
	0 または省略	時間監視をしません
マイナス値	任意	時間監視をしません

タイムアウト発生時には，KFCA14853-W メッセージが出力され，dc_mcf_sendsync 関数が DCMCFRTN_73005（-14005）でエラーリターンします。さらに，コネクション定義（mcftalccn -w）の srtimout オペランドを省略，または no を指定した場合は，コネクションを解放し，CERREVT を起動します。

同期型メッセージの受信と各種タイマ監視値の関係

同期型メッセージの受信と各種タイマ監視値の関係を次の図に示します。

図 6-11　同期型メッセージの受信と各種タイマ監視値の関係



図中の t2，t5，t6 について次に説明します。

t2：後続セグメント受信監視時間

メッセージがネットワーク上で分割された場合，未完成のセグメントを受信してから次のセグメントを受信するまでの時間を監視します。

コネクション定義（mcftalccn -u）の masm オペランドの指定によって，監視時間の最大値が異なります。監視時間の詳細については，表 6-6 を参照してください。

タイムアウト発生時には，KFCA14823-W メッセージが出力され，dc_mcf_recvsync 関数が DCMCFRTN_73005（-14005）でエラーリターンし，コネクションを解放し，CERREVT を起動します。

t5：同期型受信監視時間（UAP）

dc_mcf_recvsync 関数を発行してから TP1/NET/TCP/IP からメッセージを受信するまでの時間を監視します。

監視時間の最大値を次の表に示します。

表 6–9　同期型受信監視時間（UAP）の最大値

dc_mcf_recvsync 関数の watchtime 引数の指定値	UAP 共通定義（mcfmuap -t）の recvtim オペランドの指定値	監視時間
プラス値	任意	dc_mcf_recvsync 関数の watchtime 引数の指定値に 60 を加算した値
0	0 以外	UAP 共通定義（mcfmuap -t）の recvtim オペランドの指定値に 60 を加算した値
	0 または省略	時間監視をしません
マイナス値	任意	時間監視をしません

タイムアウト発生時には，dc_mcf_recvsync 関数が DCMCFRTN_73001（-14001）でエラーリターンします。

t6：同期型受信監視時間（TP1/NET/TCP/IP）

dc_mcf_recvsync 関数を受け付けてから入力メッセージ編集 UOC を呼び出す直前までの時間を監視します。

監視時間の最大値を次の表に示します。

表 6–10　同期型受信監視時間（TP1/NET/TCP/IP）の最大値

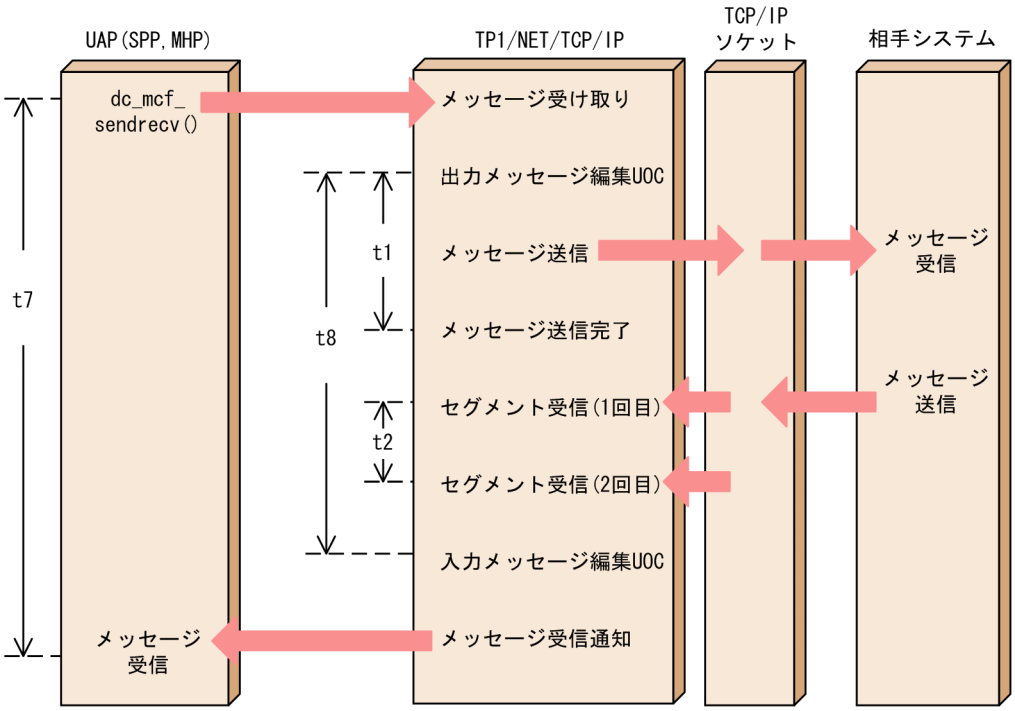
dc_mcf_recvsync 関数の watchtime 引数の指定値	UAP 共通定義（mcfmuap -t）の recvtim オペランドの指定値	監視時間
プラス値	任意	dc_mcf_recvsync 関数の watchtime 引数の指定値
0	0 以外	UAP 共通定義（mcfmuap -t）の recvtim オペランドの指定値
	0 または省略	時間監視をしません
マイナス値	任意	時間監視をしません

タイムアウト発生時には，KFCA14866-W メッセージが出力され，dc_mcf_recvsync 関数が DCMCFRTN_73005（-14005）でエラーリターンします。さらに，コネクション定義（mcftalccn -w）の srtimout オペランドを省略，または no を指定した場合は，コネクションを解放し，CERREVT を起動します。

同期型メッセージの送受信と各種タイマ監視値の関係

同期型メッセージの送受信と各種タイマ監視値の関係を次の図に示します。

図 6-12　同期型メッセージの送受信と各種タイマ監視値の関係



図中の t1，t2，t7，t8 について次に説明します。

t1：メッセージ送信完了監視時間

出力メッセージ編集 UOC のリターン直後からメッセージの送信完了（TCP/IP ソケットの送信バッファへのメッセージ書き込み完了）までの時間を監視します。

コネクション定義（mcftalccn -b）の sndcmptim オペランドの指定値が監視時間の最大値となります（省略または 0 を指定した場合は時間監視をしません）。

タイムアウト発生時には，KFCA14815-E メッセージが出力され，dc_mcf_sendrecv 関数が DCMCFRTN_73019（-14019）でエラーリターンし，論理端末を閉塞し，CERREVT を起動します。

t2：後続セグメント受信監視時間

メッセージがネットワーク上で分割された場合，未完成のセグメントを受信してから次のセグメントを受信するまでの時間を監視します。

コネクション定義（mcftalccn -u）の masm オペランドの指定によって，監視時間の最大値が異なります。監視時間の詳細については，表 6-6 を参照してください。

タイムアウト発生時には，KFCA14823-W メッセージが出力され，dc_mcf_sendrecv 関数が DCMCFRTN_73005（-14005）でエラーリターンし，コネクションを解放し，CERREVT を起動します。

t7：同期型送受信監視時間（UAP）

dc_mcf_sendrecv 関数を発行してから TP1/NET/TCP/IP からメッセージを受信するまでの時間を監視します。

監視時間の最大値を次の表に示します。

表 6–11　同期型送受信監視時間（UAP）の最大値

dc_mcf_sendrecv 関数の watchtime 引数の指定値	UAP 共通定義（mcfmuap -t）の sndrcvtim オペランドの指定値	監視時間
プラス値	任意	dc_mcf_sendrecv 関数の watchtime 引数の指定値に 60 を加算した値
0	0 以外	UAP 共通定義（mcfmuap -t）の sndrcvtim オペランドの指定値に 60 を加算した値
	0 または省略	時間監視をしません
マイナス値	任意	時間監視をしません

タイムアウト発生時には，dc_mcf_sendrecv 関数が DCMCFRTN_73001（-14001）でエラーリターンします。

t8：同期型送受信監視時間（TP1/NET/TCP/IP）

出力メッセージ編集 UOC のリターン直後から入力メッセージ編集 UOC を呼び出す直前までの時間を監視します。

監視時間の最大値を次の表に示します。

表 6–12　同期型送受信監視時間（TP1/NET/TCP/IP）の最大値

dc_mcf_sendrecv 関数の watchtime 引数の指定値	UAP 共通定義（mcfmuap -t）の sndrcvtim オペランドの指定値	監視時間
プラス値	任意	dc_mcf_sendrecv 関数の watchtime 引数の指定値
0	0 以外	UAP 共通定義（mcfmuap -t）の sndrcvtim オペランドの指定値
	0 または省略	時間監視をしません
マイナス値	任意	時間監視をしません

タイムアウト発生時には，KFCA14831-W メッセージが出力され，dc_mcf_sendrecv 関数が DCMCFRTN_73005（-14005）でエラーリターンします。さらに，コネクション定義（mcftalccn -w）の srtimout オペランドを省略，または no を指定した場合は，コネクションを解放し，CERREVT を起動します。

MCF トレースファイルの見積もり式

ここでは，トレース情報量の見積もり式，トレース情報が失われる経過時間の見積もり式，および具体的な見積もりの例について説明します。

トレース情報量の見積もり式

1 秒当たりに取得する MCF トレースファイルの，トレース情報量の見積もり式を次に示します。

```
1秒当たりのトレース情報量（単位：バイト）
                    ＝（A×B）＋（(C＋D)×E）＋（(F＋D)×G）
```

- A：コネクションの確立と解放時に取得するトレース情報量（単位：バイト）
  - クライアント型コネクションの場合：（32 ビットの場合）8000，（64 ビットの場合）9000
  - サーバ型コネクションの場合：（32 ビットの場合）6500，（64 ビットの場合）7000
- B：1 秒当たりのコネクション確立および解放の回数※
- C：メッセージ送信時に取得するトレース情報量（単位：バイト）
  - 一方送信メッセージまたは応答メッセージの送信の場合：（32 ビットの場合）2400，（64 ビットの場合）2700
  - 同期型メッセージの送信の場合：（32 ビットの場合）2000，（64 ビットの場合）2200
- D：次のどちらか小さい方の値（単位：バイト）
  - MCF トレースに取得する送受信メッセージのサイズ（mcfttrc -t msgsize）
  - 実際に送受信するメッセージの最大メッセージ長
- E：1 秒当たりのメッセージ送信回数※
- F：メッセージ受信時に取得するトレース情報量（単位：バイト）
  - 一方送信メッセージまたは問い合わせメッセージの受信の場合：（32 ビットの場合）2500，（64 ビットの場合）2800
  - 同期型メッセージの受信の場合：（32 ビットの場合）3500，（64 ビットの場合）3800
- G：1 秒当たりのメッセージ受信回数※

注※

MCF 通信プロセスが複数存在する場合は，MCF 通信プロセス単位で回数を算出してください。

トレース情報が失われる経過時間の見積もり式

MCF トレースファイルから，トレース情報が失われる経過時間の算出式を次に示します。

なお，算出式中の，「1 秒当たりのトレース情報量」とは，トレース情報量の見積もり式で算出した値です。

経過時間（秒） ＝ G×H×I／1秒当たりのトレース情報量（単位：バイト）

- G：トレースバッファの大きさ（mcfttrc -t size）
- H：トレースバッファの数（mcfttrc -t bufcnt）
- I：MCF トレースファイルの数（mcfttrc -t trccnt）

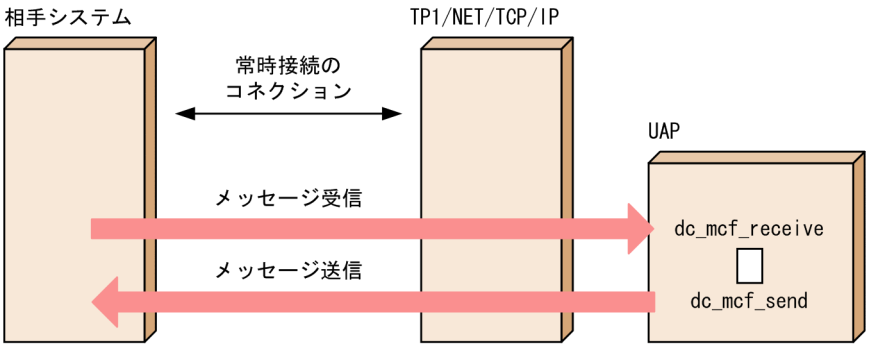
見積もり式の算出例

ここでは，トレース情報量の見積もり式，およびトレース情報が失われる経過時間の見積もり式の具体的な算出例を示します。

ここでは，次の場合を例に説明します。

1. 32 ビットでの常時接続のサーバ型コネクションで，一方送信メッセージの受信と送信をする場合
2. 32 ビットでのクライアント型コネクションで，メッセージ送受信ごとにコネクションの確立と解放をする場合

例 1　常時接続のサーバ型コネクションで，一方送信メッセージの受信と送信をする場合



この例では，次の値が想定されています。

項目	想定値
コネクションの確立と解放時に取得するトレース情報量	6000 バイト
1 分（60 秒）当たりのメッセージの受信から送信までの回数	120 回
送受信メッセージ長	1000 バイト
メッセージ送信時に取得するトレース情報量	2400 バイト
メッセージ受信時に取得するトレース情報量	2500 バイト
トレース環境定義（mcfttrc -t）のオペランドの指定値	• size ＝ 204800 • bufcnt ＝ 100 • trccnt ＝ 3 • msgsize ＝ 128

この例の場合の，計算例を次に示します。

トレース情報量の見積もり

```
(6000×0※) ＋ ((2400+128)×(120／60)) ＋ ((2500+128)×(120／60)) ＝ 10312
```

1 秒当たりのトレース情報量は，10312 バイトとなります。

注※

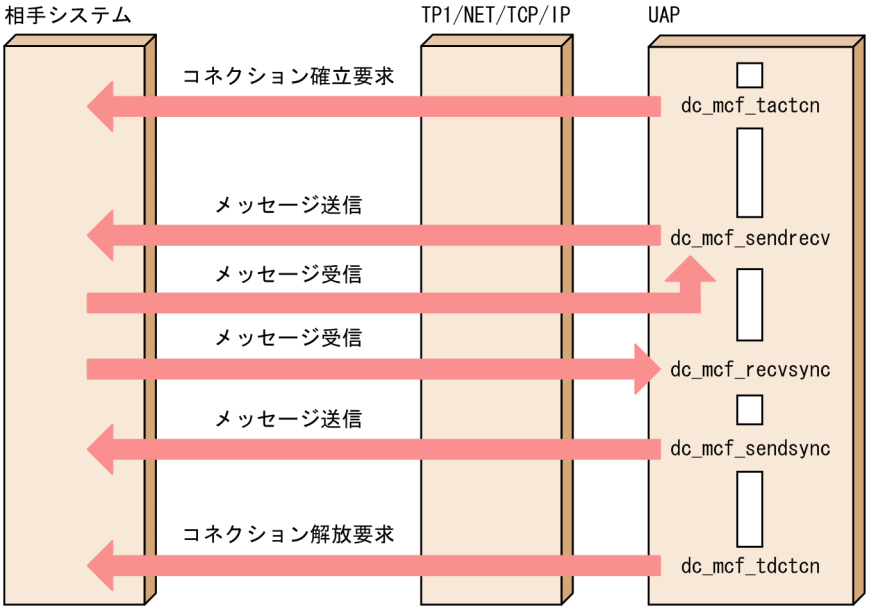
コネクションが常時接続であるため，1 秒当たりのコネクション確立および解放の回数は 0 として計算してください。

トレース情報が失われる経過時間の見積もり

```
204800×100×3／10312 ＝ 5958.1
```

トレース情報が失われる経過時間は，5958.1 秒（約 99 分）となります。

例 2　クライアント型コネクションで，メッセージ送受信ごとにコネクションの確立と解放をする場合



この例では，次の値が想定されています。

項目	想定値
コネクションの確立と解放時に取得するトレース情報量	8000 バイト
1 分（60 秒）当たりのメッセージの受信から送信までの回数	120 回
送受信メッセージ長	1000 バイト
メッセージ送信時に取得するトレース情報量	2000 バイト
メッセージ受信時に取得するトレース情報量	3500 バイト
トレース環境定義（mcfttrc -t）のオペランドの指定値	• size ＝ 204800 • bufcnt ＝ 100 • trccnt ＝ 3 • msgsize ＝ 128

この例の場合の，計算例を次に示します。

トレース情報量の見積もり

```
(8000×(120／60)) ＋ ((2000+128)×(120×2／60))
                              ＋ ((3500+128)×(120×2／60)) ＝ 39024
```

1 秒当たりのトレース情報量は，39024 バイトとなります。

トレース情報が失われる経過時間の見積もり

```
204800×100×3／39024 ＝ 1574.4
```

トレース情報が失われる経過時間は，1574.4 秒（約 26 分）となります。

DCCMII/TCP および DCCM3/TCP の通信定義との関係

TP1/NET/TCP/IP が DCCMII/TCP または DCCM3/TCP と接続し，メッセージ送達確認機能を使用する場合，TP1/NET/TCP/IP の定義と DCCMII/TCP または DCCM3/TCP の定義を対応させる必要があります。対応させる内容を次の表に示します。

表 6-13　DCCMII/TCP および DCCM3/TCP の定義との関係

相手システム（接続先）	相手システム（接続先）の定義	自システム（TP1/NET/TCP/IP）の定義
DCCMII/TCP	TERMINAL CONTENT=SLAVE	mcftalccn -u delichk = dccm2m
	TERMINAL CONTENT=MASTER	mcftalccn -u delichk = dccm2s
DCCM3/TCP	TERMINAL CONTENT=SLAVE	mcftalccn -u delichk = dccm3m
	TERMINAL CONTENT=MASTER	mcftalccn -u delichk = dccm3s

OpenTP1 システムの変更に影響する定義

OpenTP1 システムの変更に伴って見直しが必要となる定義および OpenTP1 ファイルについて説明します。

ホスト名または IP アドレスの変更

ホスト名または IP アドレスを変更する場合に，見直しが必要な定義および変更手順について説明します。

ホスト名または IP アドレスを変更する場合に見直しが必要な定義

ホスト名または IP アドレスを変更する場合，見直す必要のある定義の一覧と発生する条件を次の表に示します。

表 6-14　ホスト名または IP アドレスを変更する場合に見直しが必要な定義の一覧

定義ファイル名	定義	見直しが必要な条件
MCF 通信構成定義	mcftalccn -r ipaddr	無条件に見直しが必要
	mcftalccn -r hostname	無条件に見直しが必要
	mcftalccn -o oipaddr	相手アドレスチェックの抑止を行っていない場合，見直しが必要
	mcftalccn -o ohostname	相手アドレスチェックの抑止を行っていない場合，見直しが必要

ホスト名または IP アドレスの変更手順

ホスト名または IP アドレスは，次の手順で変更してください。

1. OpenTP1 を正常停止します。
2. MCF 通信構成定義について，変更前のホスト名または IP アドレスを検索します。
   OS が UNIX の場合は grep コマンド，Windows の場合は findstr コマンドを使用して検索します。
3. 検索の結果，変更前のホスト名または IP アドレスが見つかった場合には，変更します。
4. 変更した場合，MCF 通信構成定義の定義オブジェクトファイルを再作成します。

コネクション（論理端末）の追加

コネクション（論理端末）を追加する場合に見直す必要のある定義の一覧，および再見積もりが発生する条件を次の表に示します。

表 6-15　コネクション（論理端末）を追加する場合に見直しが必要な定義の一覧

定義ファイル名	定義	再見積もりが発生する条件
システム環境定義	static_shmpool_size※1	無条件に再見積もりが必要
	dynamic_shmpool_size※1	同時に送受信するメッセージ数が増加する場合

定義ファイル名	定義	再見積もりが発生する条件
MCF マネジャ定義	mcfmcomn -n	メッセージ出力通番を使用する場合
	mcfmcomn -p※2	無条件に再見積もりが必要
	mcfmexp -l※3	拡張予約定義を定義している場合
MCF 通信構成定義	mcftbuf -g count※4	無条件に再見積もりが必要
	mcftsts -l	状態を引き継ぐ論理端末が増える場合
システムサービス共通情報定義	max_open_fds※5	無条件に再見積もりが必要

注※1

詳細については，マニュアル「OpenTP1 システム定義」の「MCF サービス用の共用メモリの見積もり式」の説明を参照してください。

注※2

詳細については，マニュアル「OpenTP1 システム定義」の「mcfmcomn」と「MCF サービス用の共用メモリの見積もり式」の説明を参照してください。

注※3

詳細については，マニュアル「OpenTP1 システム定義」の「mcfmexp」の説明を参照してください。

注※4

詳細については，「6. システム定義」の「mcftalccn（コネクション定義の開始）」の注意事項とマニュアル「OpenTP1 システム定義」の「mcftbuf」の説明を参照してください。

注※5

詳細については，「6. システム定義」の「システムサービス共通情報定義」を参照してください。

見直す必要のある OpenTP1 ファイルの一覧，および再見積もりが発生する条件を次の表に示します。

表 6-16 コネクション（論理端末）を追加する場合に見直しが必要な OpenTP1 ファイルの一覧

OpenTP1 ファイル	再見積もりが発生する条件
ステータスファイル	次に示すどれかの条件の場合，再見積もりが必要 • メッセージ出力通番を使用する場合 • 拡張予約定義を定義している場合 • 状態を引き継ぐ論理端末が増える場合
メッセージキューファイル	入力キューまたは出力キューにディスクキューを割り当てていて，同時に送受信するメッセージ数が増加する場合

また，TP1/NET/TCP/IP のリリースノートを参照し，MCF 通信プロセスが使用するローカルメモリのメモリ所要量も見直してください。

定義例

ここでは，TP1/NET/TCP/IP のシステム構成例およびコネクション切り替え機能を使用する場合のシステム構成例，ならびにそれぞれの構成例に沿った定義例を示します。

TP1/NET/TCP/IP のシステム定義例

TP1/NET/TCP/IP のシステム構成例と，構成例に沿った定義例を次に示します。

構成例 1

UNIX 版で提供しているコーディング例におけるシステム構成例を次に示します。

これらの定義のコーディング例を次のファイルで提供しています。

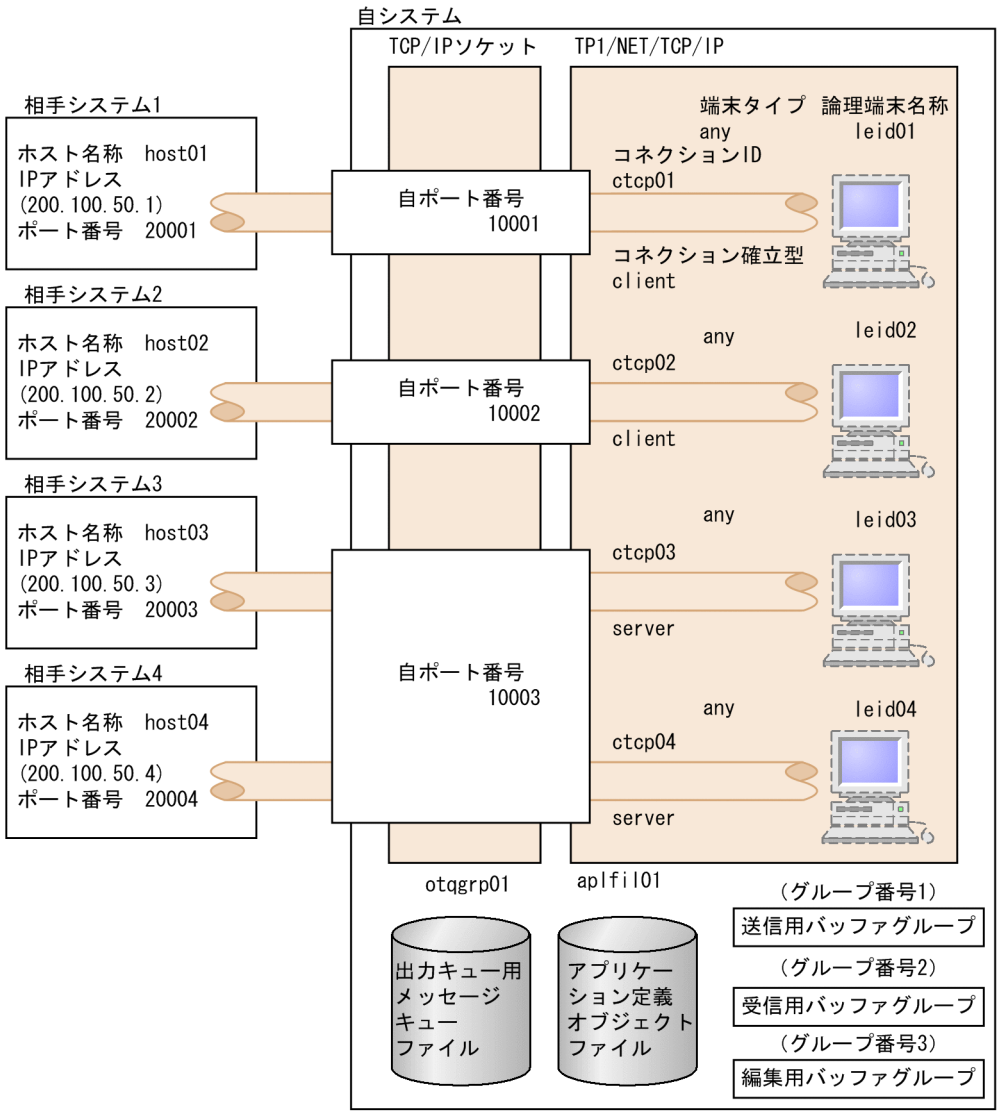
適用 OS が AIX，HP-UX または Solaris の場合

- /BeTRAN/examples/mcf/TCPIP/conf/com_c1
- /BeTRAN/examples/mcf/TCPIP/conf/com_d1

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/TCPIP/conf/com_c1
- /opt/OpenTP1/examples/mcf/TCPIP/conf/com_d1

図 6-13　TP1/NET/TCP/IP のシステム構成例 1




```
###################################################################
#    MCF通信構成定義（共通定義）                                  #
#    TCP/IPプロトコル対応                                         #
#    ファイル名：com_c1                                            #
###################################################################
#----MCF環境定義（mcftenv）----#
mcftenv    -s  01       ¥
           -a  aplfil01
#----MCF通信構成共通定義（mcftcomn）----#
mcftcomn
#----トレース環境定義（mcfttrc）----#
mcfttrc    -t  ”disk = yes”
#----バッファグループ定義（mcftbuf）----#
### バッファグループ定義（送信用）
mcftbuf    -g  ”groupno=1                                        ¥
               length =1024                                      ¥
               count  =128”
### バッファグループ定義（受信用）
mcftbuf    -g  ”groupno=2                                        ¥
               length =1024                                      ¥
```

```
                count  =128”
### バッファグループ定義（編集用）
mcftbuf    -g  ”groupno=3                                    ¥
                length =1024                                 ¥
                count  =128”
############################  終わり  ########################



#################################################################
#    MCF通信構成定義（固有定義）                                #
#    TCP/IPプロトコル対応                                       #
#    ファイル名：com_d1                                          #
#################################################################
#----コネクション定義（mcftalccn）----#
### コネクション定義開始（ctcp01）
mcftalccn  -c  ctcp01                                        ¥
           -p  tcp                                           ¥
           -g  ”sndbuf =1                                    ¥
                rcvbuf =2”                                   ¥
           -e  ”msgbuf=3                                     ¥
                count=1”                                     ¥
           -t  tcp                                           ¥
           -y  ”mode=client”                                 ¥
           -r  ”portno=10001”                                ¥
           -o  ”oipaddr=200.100.50.1                         ¥
                oportno=20001”
#-------------------------------------------------------------#
######## 論理端末定義（leid01）
mcftalcle  -l  leid01                                        ¥
           -t  any                                           ¥
           -k  ”quekind=disk                                 ¥
                quegrpid=otqgrp01”
#-------------------------------------------------------------#
### コネクション定義終了（ctcp01）
mcftalced
#-------------------------------------------------------------#
### コネクション定義開始（ctcp02）
mcftalccn  -c  ctcp02                                        ¥
           -p  tcp                                           ¥
           -g  ”sndbuf =1                                    ¥
                rcvbuf =2”                                   ¥
           -e  ”msgbuf=3                                     ¥
                count=1”                                     ¥
           -t  tcp                                           ¥
           -y  ”mode=client”                                 ¥
           -r  ”portno=10002”                                ¥
           -o  ”oipaddr=200.100.50.2                         ¥
                oportno=20002”
#-------------------------------------------------------------#
######## 論理端末定義（leid02）
mcftalcle  -l  leid02                                        ¥
           -t  any                                           ¥
           -k  ”quekind=disk                                 ¥
                quegrpid=otqgrp01”
#-------------------------------------------------------------#
### コネクション定義終了（ctcp02）
mcftalced
```

```
#------------------------------------------------------------#
### コネクション定義開始（ctcp03）
mcftalccn  -c  ctcp03                                        ¥
           -p  tcp                                           ¥
           -g  ”sndbuf =1                                    ¥
                rcvbuf =2”                                   ¥
           -e  ”msgbuf=3                                     ¥
                count=1”                                     ¥
           -t  tcp                                           ¥
           -y  ”mode=server”                                 ¥
           -r  ”portno=10003”                                ¥
           -o  ”oipaddr=200.100.50.3                         ¥
                oportno=20003”
#------------------------------------------------------------#
####### 論理端末定義（leid03）
mcftalcle  -l  leid03                                        ¥
           -t  any                                           ¥
           -k  ”quekind=disk                                 ¥
                quegrpid=otqgrp01”
#------------------------------------------------------------#
### コネクション定義終了（ctcp03）
mcftalced
#------------------------------------------------------------#
### コネクション定義開始（ctcp04）
mcftalccn  -c  ctcp04                                        ¥
           -p  tcp                                           ¥
           -g  ”sndbuf =1                                    ¥
                rcvbuf =2”                                   ¥
           -e  ”msgbuf=3                                     ¥
                count=1”                                     ¥
           -t  tcp                                           ¥
           -y  ”mode=server”                                 ¥
           -r  ”portno=10003”                                ¥
           -o  ”oipaddr=200.100.50.4                         ¥
                oportno=20004”
#------------------------------------------------------------#
####### 論理端末定義（leid04）
mcftalcle  -l  leid04                                        ¥
           -t  any                                           ¥
           -k  ”quekind=disk                                 ¥
                quegrpid=otqgrp01”
#------------------------------------------------------------#
### コネクション定義終了（ctcp04）
mcftalced
#########################  終わり  ############################
```

構成例 2

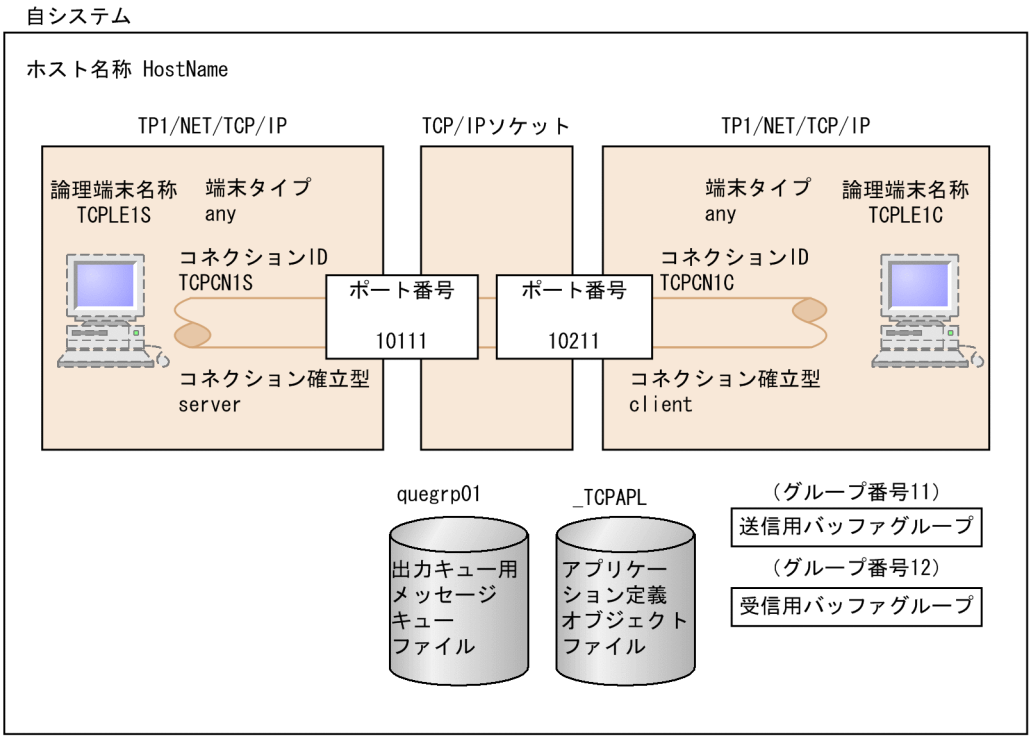
Windows 版で提供しているコーディング例でのシステム構成例を次に示します。

これらの定義のコーディング例を次のファイルで提供しています。

- %DCDIR%¥examples¥mcf¥tcpip¥conf¥source¥TCPCc
- %DCDIR%¥examples¥mcf¥tcpip¥conf¥source¥TCPCm
- %DCDIR%¥examples¥mcf¥tcpip¥conf¥source¥TCPSc

- %DCDIR%¥examples¥mcf¥tcpip¥conf¥source¥TCPSm

図 6-14　TP1/NET/TCP/IP のシステム構成例 2




```
###################################################################
#    MCF通信構成定義（共通定義）                                  #
#    TCP/IPプロトコル対応                                         #
#    ファイル名：TCPCc                                             #
###################################################################
#----MCF環境定義（mcftenv）----#
mcftenv     -s  01                          ¥
            -m  A                           ¥
            -a  _TCPAPL
#----MCF通信構成共通定義（mcftcomn）----#
mcftcomn
#----タイマ定義（mcfttim）----#
mcfttim     -p  "usertime=yes               ¥
                 msgsize=256"
#----トレース環境定義（mcfttrc）----#
mcfttrc
#----バッファグループ定義（mcftbuf）----#
### バッファグループ定義（送信用）
mcftbuf     -g  "groupno=11                 ¥
                 length=4096                ¥
                 count=2"
### バッファグループ定義（受信用）
mcftbuf     -g  "groupno=12                 ¥
                 length=4096                ¥
                 count=2"
############################  終わり  ##########################
```

```
###################################################################
#    MCF通信構成定義（固有定義）                                  #
#    TCP/IPプロトコル対応                                         #
#    ファイル名：TCPCm                                            #
###################################################################
#----コネクション定義（mcftalccn）----#
### コネクション定義開始（TCPCN1C）
mcftalccn       -c  TCPCN1C                 ¥
                -p  tcp                     ¥
                -g  ”sndbuf=11              ¥
                     rcvbuf=12”             ¥
                -t  tcp                     ¥
                -y  mode=client             ¥
                -r  portno=10211            ¥
                -o  ”ohostname=HostName     ¥
                     oportno=10111”         ¥
                -f  ”releaselog=2           ¥
                     cnerrlog=2”            ¥
                -u  ”masm=yes”
#---------------------------------------------------------#
####### 論理端末定義（TCPLE1C）
mcftalcle       -l  TCPLE1C                 ¥
                -t  any
#---------------------------------------------------------#
### コネクション定義終了（TCPCN1C）
mcftalced
############################  終わり  ########################



###################################################################
#    MCF通信構成定義（共通定義）                                  #
#    TCP/IPプロトコル対応                                         #
#    ファイル名：TCPSc                                             #
###################################################################
#----MCF環境定義（mcftenv）----#
mcftenv     -s  02                          ¥
            -m  A                           ¥
            -a  _TCPAPL
#----MCF通信構成共通定義（mcftcomn）----#
mcftcomn
#----トレース環境定義（mcfttrc）----#
mcfttrc
#----バッファグループ定義（mcftbuf）----#
### バッファグループ定義（送信用）
mcftbuf     -g  ”groupno=11                 ¥
                 length=4096                ¥
                 count=2”
### バッファグループ定義（受信用）
mcftbuf     -g  ”groupno=12                 ¥
                 length=4096                ¥
                 count=2”
############################  終わり  ########################



###################################################################
#    MCF通信構成定義（固有定義）                                  #
```

```
#    TCP/IPプロトコル対応                                    #
#    ファイル名：TCPSm                                       #
#############################################################
#----コネクション定義（mcftalccn）----#
### コネクション定義開始（TCPCN1S）
mcftalccn       -c  TCPCN1S                 ¥
                -p  tcp                     ¥
                -g  ”sndbuf=11              ¥
                     rcvbuf=12”             ¥
                -t  tcp                     ¥
                -y  mode=server             ¥
                -r  portno=10111            ¥
                -o  ”ohostname=HostName     ¥
                     oportno=10211”         ¥
                -f  ”releaselog=2           ¥
                     cnerrlog=2”            ¥
                -u  ”masm=yes”
#-----------------------------------------------------------#
####### 論理端末定義（TCPLE1S）
mcftalcle       -l  TCPLE1S                 ¥
                -t  any                     ¥
                -k  ”quekind=disk           ¥
                     quegrpid=quegrp01”     ¥
                -v  mhpap
#-----------------------------------------------------------#
### コネクション定義終了（TCPCN1S）
mcftalced
############################  終わり  ########################
```

コネクション切り替え機能を使用した TP1/NET/TCP/IP のシステム定義例

コネクション切り替え機能を使用する場合の TP1/NET/TCP/IP のシステム構成例と，構成例に沿った定義例を次に示します。

これらの定義のコーディング例を次のファイルで提供しています。Windows 版では提供していません。

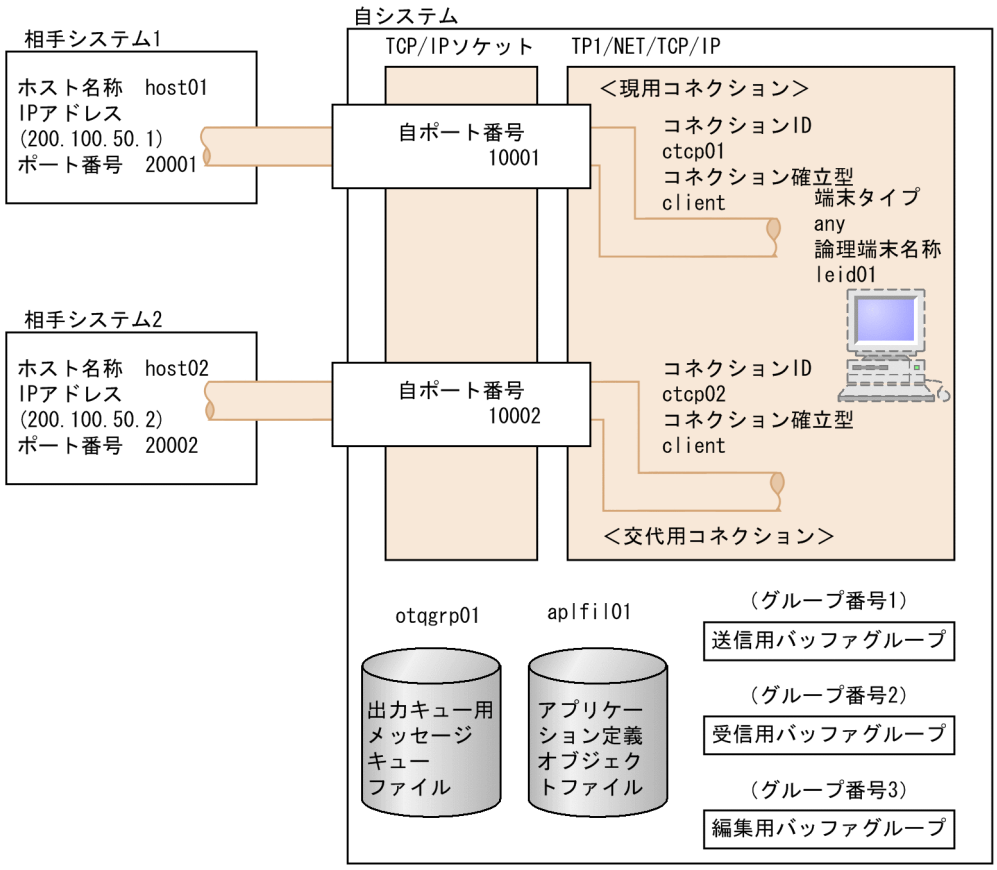
適用 OS が AIX，HP-UX または Solaris の場合

- /BeTRAN/examples/mcf/TCPIP/conf/com_c2
- /BeTRAN/examples/mcf/TCPIP/conf/com_d2

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/TCPIP/conf/com_c2
- /opt/OpenTP1/examples/mcf/TCPIP/conf/com_d2

図 6-15　コネクション切り替え機能を使用する場合の TP1/NET/TCP/IP のシステム構成例




```
#################################################################
#    MCF通信構成定義（共通定義）                                #
#    TCP/IPプロトコル対応                                       #
#    ファイル名：com_c2                                          #
#################################################################
#----MCF環境定義（mcftenv）----#
mcftenv    -s  01       ¥
           -a  aplfil01
#----MCF通信構成共通定義（mcftcomn）----#
mcftcomn
#----トレース環境定義（mcfttrc）----#
mcfttrc    -t  ”disk = yes”
#----バッファグループ定義（mcftbuf）----#
### バッファグループ定義（送信用）
mcftbuf    -g  ”groupno=1                                    ¥
               length =1024                                  ¥
               count  =128”
### バッファグループ定義（受信用）
mcftbuf    -g  ”groupno=2                                    ¥
               length =1024                                  ¥
               count  =128”
### バッファグループ定義（編集用）
mcftbuf    -g  ”groupno=3                                    ¥
               length =1024                                  ¥
               count  =128”
########################  終わり  ############################
```

```
#################################################################
#    MCF通信構成定義（固有定義）                                #
#    TCP/IPプロトコル対応                                       #
#    ファイル名：com_d2                                          #
#################################################################
#----コネクション定義（mcftalccn）----#
### コネクション定義開始（ctcp01）
mcftalccn  -c  ctcp01                                          ¥
           -p  tcp                                             ¥
           -g  "sndbuf =1                                      ¥
                rcvbuf =2"                                     ¥
           -e  "msgbuf=3                                       ¥
                count=1"                                       ¥
           -t  tcp                                             ¥
           -y  "mode=client"                                   ¥
           -r  "portno=10001"                                  ¥
           -o  "oipaddr=200.100.50.1                           ¥
                oportno=20001"
#-------------------------------------------------------------#
#######論理端末定義（leid01）
mcftalcle  -l  leid01                                          ¥
           -t  any                                             ¥
           -k  "quekind=disk                                   ¥
                quegrpid=otqgrp01"
#-------------------------------------------------------------#
###コネクション定義終了（ctcp01）
mcftalced
#-------------------------------------------------------------#
### コネクション定義開始（ctcp02）
mcftalccn  -c  ctcp02                                          ¥
           -p  tcp                                             ¥
           -g  "sndbuf =1                                       ¥
                rcvbuf =2"                                      ¥
           -e  "msgbuf=3                                       ¥
                count=1"                                       ¥
           -t  tcp                                             ¥
           -y  "mode=client"                                   ¥
           -r  "portno=10002"                                  ¥
           -o  "oipaddr=200.100.50.2                           ¥
                oportno=20002"                                 ¥
           -A  "mastercn=ctcp01"
#-------------------------------------------------------------#
###コネクション定義終了（ctcp02）
mcftalced
########################  終わり  ############################
```

7 運用コマンド

この章では，TP1/NET/TCP/IP で使用できる運用コマンドについて説明します。

TP1/NET/TCP/IP の運用コマンド

ここでは，TP1/NET/TCP/IP に関係のあるオプションについてだけ説明しています。ほかのオプションおよびその他の運用コマンドについては，マニュアル「OpenTP1 運用と操作」を参照してください。

TP1/NET/TCP/IP で使用する運用コマンドの一覧を，次の表に示します。

表 7-1　TP1/NET/TCP/IP で使用する運用コマンドの一覧

機能		コマンド名称	定義中組み込み	オフライン中に実行	オンライン中に実行	UAP から実行
コネクション管理	コネクションの確立	mcftactcn	×	×	○	○
	コネクションの解放	mcftdctcn	×	×	○	○
	コネクションの切り替え※	mcftchcn	×	×	○	○
	コネクションの状態表示	mcftlscn	×	×	○	○
	ネットワークの状態表示	mcftlsln	×	×	○	○
	サーバ型コネクションの確立要求の受付終了	mcftofln	×	×	○	○
	サーバ型コネクションの確立要求の受付開始	mcftonln	×	×	○	○
論理端末管理	論理端末の閉塞解除	mcftactle	×	×	○	○
	論理端末の閉塞	mcftdctle	×	×	○	○
	論理端末の状態表示	mcftlsle	×	×	○	○
	継続問い合わせ応答処理の強制終了	mcftendct	×	×	○	○

（凡例）

○：組み込み，または実行ができます。

×：組み込み，または実行ができません。

注※

このコマンドは，TP1/NET/High Availability を組み込んでいる場合だけ使用できます。

mcftactcn（コネクションの確立）

形式

```
mcftactcn 〔-s MCF通信プロセス識別子〕 -c コネクションID
```

機能

コネクションを確立します。

オプション

●-s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象のコネクションを制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数を指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftactcn コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

●-c コネクション ID　～〈1～8 文字の識別子〉

確立するコネクションの名称を指定します。

コネクション ID は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数を指定するときは，引用符（"）で囲んで，コネクション ID とコネクション ID との間を空白で区切ります。同一コネクション ID は重複して指定できません。

また，コネクション ID は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外のコネクション ID とは，混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべてのコネクションを確立します。

先行文字列*：先行文字列で始まるすべてのコネクションを確立します。

〈複数指定の例〉 cnn1，cnn2，cnn3 を指定する場合

```
-c ”cnn1△cnn2△cnn3”
```

〈一括指定の例〉 cnn で始まるすべてのコネクションを指定する場合

```
-c ”cnn*”
```

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftactcn コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftactcn コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10371-I	mcftactcn コマンドを正常に受け付けました。	標準出力
KFCA10373-E	mcftactcn コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10391-E	mcftactcn コマンドはサポートされていません。	標準エラー出力
KFCA10500-I	ヘルプメッセージ	標準出力
KFCA14818-W	コネクション確立済みのため，運用コマンドは受け付けられません。	標準エラー出力
KFCA14819-W	コネクション確立処理中のため，運用コマンドは受け付けられません。	標準エラー出力
KFCA14825-E	指定されたコネクションに対する論理端末が接続されていないため，運用コマンドは受け付けられません。	標準エラー出力
KFCA14829-E	サーバ型のコネクションのため，運用コマンドは受け付けられません。	標準エラー出力
KFCA14845-W	コネクション解放処理中のため，運用コマンドは受け付けられません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftactcn コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftactle（論理端末の閉塞解除）

形式

```
mcftactle 〔-s MCF通信プロセス識別子〕 〔-c コネクションID〕
          -l 論理端末名称
```

機能

論理端末の閉塞を解除します。

オプション

●-s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象の論理端末を制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数を指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftactle コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

●-c コネクション ID　～〈1～8 文字の識別子〉

閉塞解除したい論理端末に対応するコネクションのコネクション ID を指定します。

コネクション ID は複数を指定できません。また，一括指定もできません。

●-l 論理端末名称　～〈1～8 文字の識別子〉

閉塞解除する論理端末の名称を指定します。

-c オプションを指定した場合は，指定したコネクション ID に対応する論理端末の名称を指定します。

論理端末名称は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数を指定するときは，引用符（"）で囲んで，論理端末名称と論理端末名称との間を空白で区切ります。同一論理端末名称は重複して指定できません。

また，論理端末名称は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外の論理端末名称とは，混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべての論理端末の閉塞を解除します。

先行文字列*：先行文字列で始まるすべての論理端末の閉塞を解除します。

〈複数指定の例〉 len1，len2，len3 を指定する場合

```
-l ”len1△len2△len3”
```

〈一括指定の例〉 len で始まるすべての論理端末を指定する場合

```
-l ”len*”
```

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftactle コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftactle コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10371-I	mcftactle コマンドを正常に受け付けました。	標準出力
KFCA10373-E	mcftactle コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10382-E	指定した論理端末は登録されていません。	標準エラー出力
KFCA10390-E	指定されたコネクション ID と論理端末名称の対応が正しくありません。	標準エラー出力
KFCA10391-E	mcftactle コマンドはサポートされていません。	標準エラー出力
KFCA10395-E	指定したコネクションには未接続の論理端末名称が指定されています。	標準エラー出力
KFCA10503-I	ヘルプメッセージ	標準出力
KFCA14820-W	コネクションが確立されていないため，運用コマンドは受け付けられません。	標準エラー出力

メッセージ ID	内容	出力先
KFCA14821-W	論理端末が閉塞解除済みのため，運用コマンドは受け付けられません。	標準エラー出力
KFCA14826-E	指定されたコネクションに対する論理端末が接続されていないため，論理端末の閉塞を解除できません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftactle コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftchcn（コネクションの切り替え）

形式

```
mcftchcn 〔-s MCF通信プロセス識別子〕 -f 切り替え元コネクションID
          -t 切り替え先コネクションID
```

機能

切り替え元コネクションと論理端末の接続を無効にして，切り替え先コネクションと論理端末を接続します。

mcftchcn コマンドを実行する前に，切り替え元のコネクション上の論理端末を閉塞してください。

このコマンドを入力するときは，切り替え元コネクションおよび切り替え先コネクションは確立されていない状態でなければなりません。このコマンドの処理では，コネクションの状態は変更されません。

オプション

●-s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象のコネクションを制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数を指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftchcn コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

●-f 切り替え元コネクション ID　～〈1～8 文字の識別子〉

切り替え元のコネクションのコネクション ID を指定します。

切り替え元のコネクション ID は複数を指定できません。

●-t 切り替え先コネクション ID　～〈1～8 文字の識別子〉

切り替え先のコネクションのコネクション ID を指定します。

切り替え先のコネクション ID は複数を指定できません。

出力メッセージ

メッセージ ID	内容	出力先
KFCA10110-E	閉塞されていない論理端末が存在するため，運用コマンドは受け付けられません。	標準エラー出力
KFCA10115-E	コネクションの切り替えができる構成になっていません。	標準エラー出力

メッセージ ID	内容	出力先
KFCA10350-I	mcftchcn コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftchcn コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10371-I	mcftchcn コマンドを正常に受け付けました。	標準出力
KFCA10373-E	mcftchcn コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10391-E	mcftchcn コマンドはサポートされていません。	標準エラー出力
KFCA10392-E	MCF 運用コマンドの実行に必要なプログラムプロダクトがインストールされていません。	標準エラー出力
KFCA10530-I	ヘルプメッセージ	標準出力
KFCA14827-E	指定した切り替え元コネクションが解放されていません。	標準エラー出力
KFCA14828-E	指定した切り替え元コネクションに対して，論理端末が接続されていません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftchcn コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftdctcn（コネクションの解放）

形式

```
mcftdctcn 〔-s MCF通信プロセス識別子〕 -c コネクションID 〔-f〕
```

機能

コネクションを解放します。該当コネクションが仕掛り中の場合でも，強制的に解放します。コネクション解放後，CCLSEVT または CERREVT を通知します。コネクション解放後に通知する MCF イベントを次の表に示します。

-f オプションの指定	コネクション定義（mcftalccn -f）の cnrelease オペランドの指定値	通知する MCF イベント
なし	任意	CCLSEVT
あり	fin	CCLSEVT
	rst	CERREVT

なお，仕掛り中の処理は，次のように扱います。

- 受信仕掛り中の場合，受信途中のメッセージを破棄します。
- 一方送信（dc_mcf_send）の仕掛り中の場合，送信処理を中断し，仕掛り中のメッセージを出力キューに戻します。出力キューに戻したメッセージは，次回のコネクション確立時に再送されます。
- 同期型メッセージの送受信関数（dc_mcf_sendrecv，dc_mcf_sendsync，または dc_mcf_recvsync）の仕掛り中の場合，送受信処理を中断し，エラーコード DCMCFRTN_73020 でリターンします。

オプション

●-s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象のコネクションを制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数を指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftdctcn コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

●-c コネクション ID　～〈1～8 文字の識別子〉

解放するコネクションのコネクション ID を指定します。

コネクション ID は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数を指定するときは，引用符（"）で囲んで，コネクション ID とコネクション ID との間を空白で区切ります。同一コネクション ID は重複して指定できません。

また，コネクション ID は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外のコネクション ID とは，混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべてのコネクションを解放します。

先行文字列*：先行文字列で始まるすべてのコネクションを解放します。

〈複数指定の例〉 cnn1，cnn2，cnn3 を指定する場合

```
-c "cnn1△cnn2△cnn3"
```

〈一括指定の例〉 cnn で始まるすべてのコネクションを指定する場合

```
-c "cnn*"
```

●-f

コネクション定義（mcftalccn -f）の cnrelease オペランドに rst を指定している場合，RST パケットを送信してコネクションを強制解放します。

コネクション定義（mcftalccn -f）の cnrelease オペランドを省略，または fin を指定している場合，指定しても無効となり，FIN パケットを送信してコネクションを正常に解放します。

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftdctcn コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力

メッセージ ID	内容	出力先
KFCA10359-W	mcftdctcn コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10371-I	mcftdctcn コマンドを正常に受け付けました。	標準出力
KFCA10373-E	mcftdctcn コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10391-E	mcftdctcn コマンドはサポートされていません。	標準エラー出力
KFCA10501-I	ヘルプメッセージ	標準出力
KFCA14820-W	コネクションが確立されていないため，運用コマンドは受け付けられません。	標準エラー出力
KFCA14845-W	コネクション解放処理中のため，運用コマンドは受け付けられません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftdctcn コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftdctle（論理端末の閉塞）

形式

```
mcftdctle 〔-s MCF通信プロセス識別子〕 〔-c コネクションID〕
          -l 論理端末名称
```

機能

論理端末を閉塞します。該当コネクションが同期型メッセージの送受信関数仕掛り中の場合，同期型メッセージの送受信関数（dc_mcf_sendsync または dc_mcf_sendrecv）をエラーコード DCMCFRTN_73020 でエラーリターンし，コネクションを解放します。

オプション

●-s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象の論理端末を制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数を指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftdctle コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

●-c コネクション ID　～〈1～8 文字の識別子〉

閉塞したい論理端末に対応するコネクションのコネクション ID を指定します。

コネクション ID は複数を指定できません。また，一括指定もできません。

●-l 論理端末名称　～〈1～8 文字の識別子〉

閉塞する論理端末の名称を指定します。

-c オプションを指定した場合は，指定したコネクション ID に対応する論理端末の名称を指定します。

論理端末名称は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数を指定するときは，引用符（"）で囲んで，論理端末名称と論理端末名称との間を空白で区切ります。同一論理端末名称は重複して指定できません。

また，論理端末名称は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外の論理端末名称とは，混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべての論理端末を閉塞します。

先行文字列*：先行文字列で始まるすべての論理端末を閉塞します。

〈複数指定の例〉 len1，len2，len3 を指定する場合

```
-l "len1△len2△len3"
```

〈一括指定の例〉 len で始まるすべての論理端末を指定する場合

```
-l "len*"
```

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftdctle コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftdctle コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10371-I	mcftdctle コマンドを正常に受け付けました。	標準出力
KFCA10373-E	mcftdctle コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10382-E	指定した論理端末は登録されていません。	標準エラー出力
KFCA10390-E	指定されたコネクション ID と論理端末名称の対応が正しくありません。	標準エラー出力
KFCA10391-E	mcftdctle コマンドはサポートされていません。	標準エラー出力
KFCA10395-E	指定したコネクションには未接続の論理端末名称が指定されています。	標準エラー出力

メッセージ ID	内容	出力先
KFCA10504-I	ヘルプメッセージ	標準出力
KFCA14820-W	コネクションが確立されていないため，運用コマンドは受け付けられません。	標準エラー出力
KFCA14822-W	論理端末が閉塞済みのため，運用コマンドは受け付けられません。	標準エラー出力
KFCA14881-W	論理端末は継続問い合わせ中のため，運用コマンドは受け付けられません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- 受信仕掛り中に運用コマンド（mcftdctle）を入力した場合は，論理端末が閉塞状態でも，メッセージを受信します。論理端末閉塞によるメッセージ受信処理への影響はありません。
- 送信仕掛り中に運用コマンド（mcftdctle）を入力した場合は，一方送信メッセージまたは応答メッセージの送信処理を中断しません。送信仕掛り中のメッセージの送信完了後に論理端末が閉塞されます。
- 継続問い合わせ応答中は論理端末を閉塞することはできません。
- 問い合わせ応答中に運用コマンド（mcftdctle）を入力した場合，応答メッセージが出力キューに滞留することがありますが，次に論理端末が閉塞解除しても送信しないで破棄します。
- mcftdctle コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftendct（継続問い合わせ応答処理の強制終了）

形式

```
mcftendct 〔-s MCF通信プロセス識別子〕 -l 論理端末名称〔-f〕
```

機能

継続問い合わせ応答処理を強制終了します。

オプション

●-s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象の論理端末を制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数を指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftendct コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

●-l 論理端末名称　～〈1～8 文字の識別子〉

継続問い合わせ応答処理中の論理端末の名称を指定します。

論理端末名称は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数を指定するときは，引用符（"）で囲んで，論理端末名称と論理端末名称との間を空白で区切ります。同一論理端末名称は重複して指定できません。

また，論理端末名称は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外の論理端末名称とは，混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべての論理端末を対象とします。

先行文字列*：先行文字列で始まるすべての論理端末を対象とします。

〈複数指定の例〉 len1，len2，len3 を指定する場合

```
-l ”len1△len2△len3”
```

〈一括指定の例〉 len で始まるすべての論理端末を指定する場合

```
-l ”len*”
```

●-f

継続問い合わせ応答処理を即時強制終了する場合に指定します。このオプションを指定すると，直ちに継続問い合わせ用一時記憶領域を解放し，該当する論理端末からの問い合わせを処理している MHP（エラーイベントを含む）は異常終了します。このときエラーイベントは起動しません。

-f オプションを指定した mcftendct コマンドの実行後に MHP が次に示す処理をしたとき，MHP が異常終了します。

- TP1/Message Control 以外のリソースマネジャにもアクセスする MHP のサービス終了時
- 継続問い合わせ用一時記憶領域アクセス時（dc_mcf_tempget 関数，dc_mcf_tempput 関数発行時）
- ロールバック要求時（dc_mcf_rollback 関数発行時（action：DCMCFRTRY，または DCMCFNRTN の場合））

このオプションの指定を省略すると，MHP が該当する論理端末からの問い合わせを処理中の場合，エラーメッセージが出力されます。

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftendct コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftendct コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10371-I	mcftendct コマンドを正常に受け付けました。	標準出力
KFCA10373-E	mcftendct コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10382-E	指定した論理端末は登録されていません。	標準エラー出力
KFCA10391-E	mcftendct コマンドはサポートされていません。	標準エラー出力
KFCA10521-I	ヘルプメッセージ	標準出力

メッセージ ID	内容	出力先
KFCA14878-E	継続問い合わせ応答終了処理で障害が発生しました。	標準エラー出力
KFCA14879-W	UAP が継続問い合わせ応答処理中のため，運用コマンドは受け付けられません。	標準エラー出力
KFCA14880-W	論理端末は継続問い合わせ中ではありません。	標準エラー出力
KFCA14882-W	論理端末は継続問い合わせ応答形態をサポートしていません。	標準エラー出力
KFCA14885-I	継続問い合わせ応答を強制終了しました。	メッセージログファイル
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftendct コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftlscn（コネクションの状態表示）

形式

```
mcftlscn 〔-s MCF通信プロセス識別子〕 -c コネクションID 〔-d〕
```

機能

コネクションの状態を標準出力で表示します。

オプション

●-s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象のコネクションを制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は，複数を指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftlscn コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

●-c コネクション ID　～〈1～8 文字の識別子〉

状態を表示するコネクションのコネクション ID を指定します。

コネクション ID は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数を指定するときは，引用符（"）で囲んで，コネクション ID とコネクション ID との間を空白で区切ります。同一コネクション ID は重複して指定できません。

また，コネクション ID は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外のコネクション ID とは，混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべてのコネクションの状態を表示します。

先行文字列*：先行文字列で始まるすべてのコネクションの状態を表示します。

〈複数指定の例〉　cnn1，cnn2，cnn3 を指定する場合

```
-c ”cnn1△cnn2△cnn3”
```

〈一括指定の例〉　cnn で始まるすべてのコネクションを指定する場合

```
-c ”cnn*”
```

●-d

コネクションの状態と該当するコネクションに対応する論理端末の情報を表示します。

このオプションの指定を省略すると，コネクションの状態だけを表示します。

出力形式

```
mmm cccccccc ppp sssss ddddd     ef            …1
    llllllll ttt aaa                           …2
    qqqqq (rrrrr)                              …3
```

- 1：-d オプションの指定に関係なく出力します。
- 2：-d オプションを指定した場合に出力します。
- 3：-d オプションを指定し，かつ，出力形式 2 で状態表示する（コネクション定義（mcftalccn -C）の lscnfmt オペランドに 2 を指定）場合に出力します。
- mmm：MCF 識別子
- cccccccc：コネクション ID
- ppp：プロトコル種別
- sssss：コネクションの状態
  ACT…確立状態
  ACT/B…確立処理中状態
  DCT…解放状態
  DCT/B…解放処理中状態
- ddddd：詳細ステータス（保守情報）
  INIT…初期状態
  CNC_W…コネクション確立待ち状態
  CNRTY…コネクション確立再試行待ち状態
  CNREQ…コネクション確立要求待ち状態
  SRREQ…データ送受信可能状態
  SND_W…データ送信処理中状態
  NEXTW…後続データ受信待ち状態
  SR_SW…同期型送受信のデータ送信処理中状態
  SR_W…同期型送受信のデータ受信待ち状態
  SR_NW…同期型送受信の後続データ受信待ち状態
  ACK_W…送達確認メッセージ受信待ち状態
  SD_SW…同期型送信のデータ送信処理中状態
  RV_W…同期型受信のデータ受信待ち状態
  ANS…問い合わせ応答中状態

CONT…継続問い合わせ応答中状態

MHPEW…継続問い合わせ応答中の MHP 処理完了待ち状態

*****…コネクション解放処理中状態

- e：問い合わせ応答状態

  問い合わせ応答および継続問い合わせ応答形態によるメッセージ送受信機能を使用する（コネクション定義（mcftalccn -l）の replymsg に yes を指定）場合に出力します。

  A…問い合わせ応答中

  C…継続問い合わせ応答中

- f：UAP 実行中状態

  問い合わせ応答および継続問い合わせ応答形態によるメッセージ送受信機能を使用する（コネクション定義（mcftalccn -l）の replymsg に yes を指定）場合に出力します。

  U…UAP 実行中状態

  U が表示される場合，-f オプションを指定しない mcftendct コマンドは受け付けられません。

- llllllll：論理端末名称
- ttt：論理端末タイプ
- aaa：論理端末とコネクションの接続状態（未接続の場合だけ）

  NOU…未接続

- qqqqq：現在保留している受信メッセージ数
- rrrrr：OpenTP1 システム開始以降に保留した，最大受信メッセージ数

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftlscn コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftlscn コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10360-I	状態表示を開始します。	標準出力

メッセージ ID	内容	出力先
KFCA10361-I	標準情報を表示します。	標準出力
KFCA10362-I	詳細情報を表示します。	標準出力
KFCA10369-I	状態表示を終了します。	標準出力
KFCA10373-E	mcftlscn コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10391-E	mcftlscn コマンドはサポートされていません。	標準エラー出力
KFCA10502-I	ヘルプメッセージ	標準出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力
KFCA16429-I	付加情報	標準出力

mcftlsle（論理端末の状態表示）

形式

```
mcftlsle 〔-s MCF通信プロセス識別子〕 〔-c コネクションID〕
          -l 論理端末名称 〔-q〕
```

機能

論理端末の状態を標準出力で表示します。

オプション

●-s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象の論理端末を制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。アプリケーション起動サービスのアプリケーション起動プロセス識別子は指定できません。

MCF 通信プロセス識別子は複数を指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftlsle コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

●-c コネクション ID　～〈1～8 文字の識別子〉

状態を表示したい論理端末に対応するコネクションのコネクション ID を指定します。

コネクション ID は複数を指定できません。また，一括指定もできません。

●-l 論理端末名称　～〈1～8 文字の識別子〉

状態を表示する論理端末の名称を指定します。

-c オプションを指定した場合，指定したコネクション ID に対応する論理端末名称を指定します。

論理端末名称は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数を指定するときは，引用符（"）で囲んで，論理端末名称と論理端末名称との間を空白で区切ります。同一論理端末名称は，重複して指定できません。

また，論理端末名称は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外の論理端末名称とは，混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべての論理端末の状態を表示します。

先行文字列*：先行文字列で始まるすべての論理端末の状態を表示します。

〈複数指定の例〉 len1，len2，len3 を指定する場合

```
-l ”len1△len2△len3”
```

〈一括指定の例〉 len で始まるすべての論理端末を指定する場合

```
-l ”len*”
```

●-q

指定した論理端末に対応する出力キューの保留状態を表示します。

このオプションの指定を省略すると，論理端末に対応する出力キューの保留状態は表示しません。

出力形式

```
mmm lllllll sss
    SYNC xxxxxxxxx yyyyyyyyyy zzzzzzzzzz
      IO      :          :      :
    PRIO      :          :      :
    NORM      :          :      :
    iii ooo
```

- mmm：MCF 識別子
- lllllll：論理端末名称
- sss：論理端末状態
  ACT…閉塞解除状態
  DCT…閉塞状態
- SYNC：同期型メッセージ
- IO：非同期型問い合わせ応答メッセージ
- PRIO：非同期型一方送信メッセージ（優先）
- NORM：非同期型一方送信メッセージ（一般）
- xxxxxxxxx：未送信メッセージ数
- yyyyyyyyyy：未送信メッセージの先頭の出力通番
- zzzzzzzzzz：未送信メッセージの最後の出力通番
- iii：出力キューの入力の保留状態（-q オプション指定時だけ表示）
  NOH…保留解除
  HLD…保留
- ooo：出力キューのスケジュールの保留状態（-q オプション指定時だけ 表示）
  NOH…保留解除

HLD…保留

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftlsle コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftlsle コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10360-I	状態表示を開始します。	標準出力
KFCA10364-I	表示情報	標準出力
KFCA10365-I	表示情報	標準出力
KFCA10369-I	状態表示を終了します。	標準出力
KFCA10373-E	mcftlsle コマンドが異常終了しました。	標準エラー出力
KFCA10378-I	表示情報	標準出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10382-E	指定した論理端末は登録されていません。	標準エラー出力
KFCA10390-E	指定したコネクション ID と論理端末名称の対応が正しくありません。	標準エラー出力
KFCA10391-E	mcftlsle コマンドはサポートされていません。	標準エラー出力
KFCA10395-E	指定したコネクションには未接続の論理端末名称が指定されています。	標準エラー出力
KFCA10505-I	ヘルプメッセージ	標準出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

mcftlsln（ネットワークの状態表示）

形式

```
mcftlsln 〔-s MCF通信プロセス識別子〕〔-t〕
```

機能

ネットワークの状態を標準出力で表示します。

オプション

●-s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象の MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数を指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftlsln コマンドを実行します。

●-t

相手システムとのメッセージ送受信に関するネットワークの状態を表示します。

このオプションの指定を省略すると，サーバ型コネクションの確立要求の受付状態を表示します。

出力形式と出力例

出力形式と出力例を，-t オプションを指定した場合と省略した場合とに分けて，それぞれ次に示します。

-t オプションを指定した場合

出力形式

```
mmm cccccccc ppp y sss AAAA.BBBB (CCCC.DDDD) EEEE.FFFF (GGGG.HHHH)
```

- mmm：MCF 識別子
- cccccccc：コネクション ID
- ppp：プロトコル種別
  tcp…TCP/IP プロトコル
- y ：コネクションの確立モード（mcftalccn -y mode）
  C…クライアント型のコネクション
  S…サーバ型のコネクション
- sss：メッセージ送受信に関するネットワークの状態

OPN…使用中
CLS…未使用

- AAAA：自システムの IP アドレス（dotted-decimal 形式表示）
  例：192.11.42.20
  ネットワークの状態が未使用の場合，*を出力します。
- BBBB：自システムのポート番号
  ネットワークの状態が未使用の場合，*を出力します。
- CCCC：コネクション定義の自システムのホスト名（mcftalccn -r hostname）または自システムのホストの IP アドレス（mcftalccn -r ipaddr）
  自システムのホスト名は最大 15 文字まで表示します。
  定義を省略した場合，*を出力します。
- DDDD：コネクション定義の自システムのポート番号（mcftalccn -r portno）
  定義を省略した場合，*を出力します。
- EEEE：相手システムの IP アドレス（dotted-decimal 形式表示）
  ネットワークの状態が未使用の場合，*を出力します。
- FFFF：相手システムのポート番号
  ネットワークの状態が未使用の場合，*を出力します。
- GGGG：コネクション定義の相手システムのホスト名（mcftalccn -o ohostname）または相手システムのホストの IP アドレス（mcftalccn -o oipaddr）
  相手システムのホスト名は最大 15 文字まで表示します。
  相手アドレス情報をチェックしない（mcftalccn -h addrchk に no を指定）場合，*を出力します。
- HHHH：コネクション定義の相手システムのポート番号（mcftalccn -o oportno）
  相手アドレス情報をチェックしない（mcftalccn -h addrchk に no を指定）場合，または，相手ポートが任意（mcftalccn -o oportno に free を指定）の場合，*を出力します。

出力例

```
A01 TCPC0001 tcp C OPN 172.18.50.64.32776  (*.*)          172.16.45.123.20000  (host02.20000)
A01 TCPC0002 tcp C CLS *.*                 (host01.5000)  *.*                  (host02.20001)
A02 TCPS0001 tcp S OPN 172.18.50.64.30000  (host01.30000) 172.16.45.123.2345   (host02.*)
A02 TCPS0002 tcp S CLS *.*                 (*.40000)      *.*                  (*.*)
```

-t オプションを省略した場合

出力形式

```
mmm cccccccc ppp sssss IIIII JJJJJ
```

- mmm：MCF 識別子
- cccccccc：コネクション ID

- ppp：プロトコル種別
  tcp…TCP/IP プロトコル
- sssss：サーバ型コネクションの確立要求の受付状態
  INIT…受付終了状態
  LISTEN…受付開始状態
  RETRY…受付開始処理の再試行中状態
- IIIII：自システムのポート番号（mcftalccn -r portno）
- JJJJJ：自システムの IP アドレス（mcftalccn -r ipaddr）（dotted-decimal 形式表示）
  定義を省略した場合，*を出力します。

出力例

```
A02 TCPS0001 tcp LISTEN 30000 172.18.50.64
A02 TCPS0002 tcp INIT   40000 *
```

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftlsln コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftlsln コマンド入力元への応答に失敗しました。	メッセージログファイル，または標準エラー出力
KFCA10360-I	状態表示を開始します。	標準出力
KFCA10369-I	状態表示を終了します。	標準出力
KFCA10372-E	mcftlsln コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10391-E	mcftlsln コマンドはサポートされていません。	標準エラー出力
KFCA10561-I	ヘルプメッセージ	標準出力

メッセージ ID	内容	出力先
KFCA14862-W	サーバ型のコネクションは存在しません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力
KFCA16432-I	ネットワーク情報の表示	標準出力

注意事項

- mcftlsln コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftofln（サーバ型コネクションの確立要求の受付終了）

形式

```
mcftofln -s MCF通信プロセス識別子
```

機能

サーバ型コネクションの確立要求の受付を終了します。

オプション

●-s MCF 通信プロセス識別子　　～〈数字（0～9），a～f〉((01～ef))

処理対象の MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数を指定できません。

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftofln コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftofln コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10370-I	mcftofln コマンドを正常に受け付けました。	標準出力
KFCA10372-E	mcftofln コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10391-E	mcftofln コマンドはサポートされていません。	標準エラー出力
KFCA10560-I	ヘルプメッセージ	標準出力
KFCA14861-W	着呼受付が終了しています。	標準エラー出力

メッセージ ID	内容	出力先
KFCA14862-W	サーバ型のコネクションは存在しません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftofln コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftonln（サーバ型コネクションの確立要求の受付開始）

形式

```
mcftonln -s MCF通信プロセス識別子
```

機能

サーバ型コネクションの確立要求の受付を開始します。

オプション

●-s MCF 通信プロセス識別子　　～〈数字（0～9），a～f〉((01～ef))

処理対象の MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数を指定できません。

出力メッセージ

メッセージ ID	内容	出力先
KFCA10350-I	mcftonln コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftonln コマンド入力元への応答に失敗しました。	メッセージログファイル
KFCA10370-I	mcftonln コマンドを正常に受け付けました。	標準出力
KFCA10372-E	mcftonln コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10391-E	mcftonln コマンドはサポートされていません。	標準エラー出力
KFCA10559-I	ヘルプメッセージ	標準出力
KFCA14860-W	着呼受付は開始しています。	標準エラー出力

メッセージ ID	内容	出力先
KFCA14862-W	サーバ型のコネクションは存在しません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftonln コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

8 組み込み方法

この章では，TP1/NET/TCP/IP を OpenTP1 システムに組み込む方法について説明します。

8.1　TP1/NET/TCP/IP の組み込みの流れ

TP1/NET/TCP/IP を OpenTP1 システムに組み込むときの作業の流れを示します。

8.1.1　MCF メイン関数の作成

TP1/NET/TCP/IP を起動するためには，MCF メイン関数をコーディングし，コンパイル，およびリンケージしておく必要があります。詳細は，「8.2　MCF メイン関数の作成」を参照してください。

8.1.2　MCF サービス名の登録

TP1/NET/TCP/IP を実行するために，MCF サービス名をシステムサービス構成定義で定義しておく必要があります。

MCF サービス名は MCF マネジャ定義オブジェクトファイル名と一致させてください。

詳細は，マニュアル「OpenTP1 システム定義」を参照してください。

8.1.3　システムサービス情報定義ファイルの作成

システムサービス情報定義ファイルをテキストエディタで作成します。作成するファイルのパス名は，「$DCDIR/lib/sysconf/システムサービス情報定義ファイル名」です。ファイルの定義形式については，「6.　システム定義」の「システムサービス情報定義」を参照してください。

8.1.4　定義オブジェクトファイルの生成

OpenTP1 のネットワークコミュニケーション定義の各ソースファイルから定義オブジェクトファイルを生成します。詳細は，「8.3　定義オブジェクトファイルの生成」を参照してください。

8.2 MCF メイン関数の作成

TP1/NET/TCP/IP は，OpenTP1 プロセスサービスによって起動されます。

TP1/NET/TCP/IP を起動するためには，MCF メイン関数を作成し，コンパイル，およびリンケージを行って TP1/NET/TCP/IP の実行形式プログラムを作成する必要があります。リンケージには，mcfpltcp コマンドを使用します。

MCF メイン関数では，スタート関数（dc_mcf_svstart）を呼び出します。UOC を使用する場合は，MCF メイン関数で UOC の関数アドレスを指定してください。

UOC は，MCF メイン関数と同じ言語（ANSI C，C++または K&R 版 C）で作成してください。

MCF メイン関数のコーディング概要を図 8-1，図 8-2 に示します。また，ディレクトリへの組み込み方法を図 8-3 に示します。

なお，これらのコーディング例は，次のファイルで提供しています。

適用 OS が Windows の場合

- %DCDIR%¥examples¥mcf¥tcpip¥cmlib¥c¥com.c

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/TCPIP/cmlib/ansi/com.c
- /opt/OpenTP1/examples/mcf/TCPIP/cmlib/c/com.c

その他の OS の場合

- /BeTRAN/examples/mcf/TCPIP/cmlib/ansi/com.c
- /BeTRAN/examples/mcf/TCPIP/cmlib/c/com.c

図 8-1　MCF メイン関数のコーディング概要（ANSI C，C++の場合）

```
#include <dcmtcp.h>                          /*TP1/NET/TCP/IP用ヘッダファイル      */   1.

extern DCLONG segchk01(dctcp_uoc_sgck *);※
                                             /*入力セグメント判定UOC関数extern宣言 */
extern DCLONG msgrcv01(dcmcf_uoc_min_n *);
                                             /*入力メッセージ編集UOC関数extern宣言 */   2.
extern DCLONG msgsend01(dcmcf_uoc_mout_n *);
                                             /*出力メッセージ編集UOC関数extern宣言 */

extern dcmcf_uoc_t  dcmcf_uoctbl;            /*UOCテーブルextern宣言               */   3.

int main()
{
    dcmcf_uoctbl.segchk = (dcmcf_uocfunc)segchk01;※
                                             /*入力セグメント判定UOCアドレス設定   */
    dcmcf_uoctbl.msgrcv = (dcmcf_uocfunc)msgrcv01;
                                             /*入力メッセージ編集UOCアドレス設定   */   4.
    dcmcf_uoctbl.msgsend = (dcmcf_uocfunc)msgsend01;
                                             /*出力メッセージ編集UOCアドレス設定   */

    dc_mcf_svstart();                         /*スタート関数呼び出し               */   5.

    return 0;
}
```

注※

TP1/NET/TCP/IP が標準提供する入力セグメント判定 UOC を使用する場合は，「segchk01」の代わりに「dc_mcf_stduoc_tcp_segchk」をコーディングしてください。

図 8-2　MCF メイン関数のコーディング概要（K&R 版 C の場合）

```
#include <dcmtcp.h>                        /*TP1/NET/TCP/IP用ヘッダファイル      */   1.

extern  DCLONG    segchk01();※              /*入力セグメント判定UOC関数extern宣言*/
extern  DCLONG    msgrcv01();               /*入力メッセージ編集UOC関数extern宣言*/   2.
extern  DCLONG    msgsend01();              /*出力メッセージ編集UOC関数extern宣言 */

extern  dcmcf_uoc_t     dcmcf_uoctbl;  /*UOCテーブルextern宣言               */   3.

main()
{
   dcmcf_uoctbl.segchk = segchk01;※      /*入力セグメント判定UOCアドレス設定  */
   dcmcf_uoctbl.msgrcv = msgrcv01;       /*入力メッセージ編集UOCアドレス設定  */   4.
   dcmcf_uoctbl.msgsend = msgsend01;     /*出力メッセージ編集UOCアドレス設定  */

   dc_mcf_svstart();                     /*スタート関数呼び出し               */   5.
 }
```

注※

TP1/NET/TCP/IP が標準提供する入力セグメント判定 UOC を使用する場合は，「segchk01」の代わりに「dc_mcf_stduoc_tcp_segchk」をコーディングしてください。

1. TP1/NET/TCP/IP で提供するヘッダファイルを取り込みます。
2. 使用する UOC 関数を extern 宣言します。UOC のリターン値は DCLONG 型にしてください。
   UOC をまったく使用しない場合，このコーディングは必要ありません。

3. UOC テーブルを extern 宣言します。UOC を使用する場合，必ずこのとおりにコーディングしてください。

   UOC をまったく使用しない場合，このコーディングは必要ありません。

4. 各 UOC 関数のアドレスを，次に示すシステム提供変数に設定します。使用する UOC だけコーディングしてください。

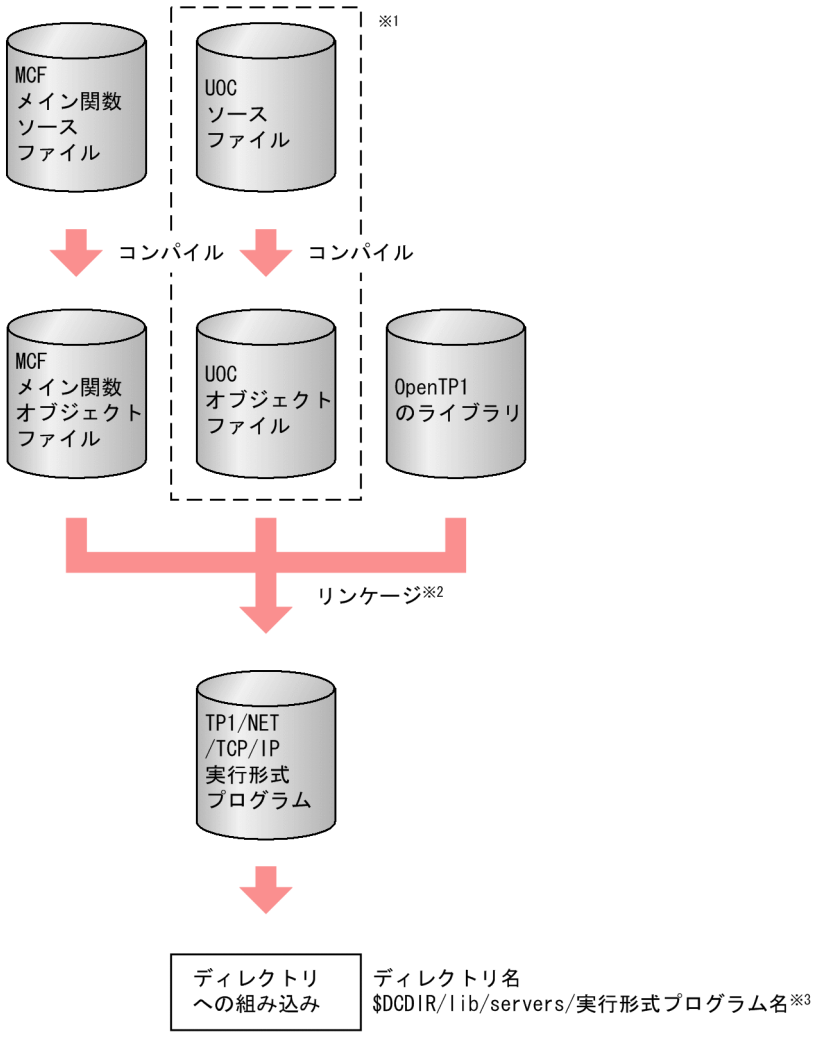
```
dcmcf_uoctbl.segchk  /*入力セグメント判定UOCアドレス*/
dcmcf_uoctbl.msgrcv  /*入力メッセージ編集UOCアドレス*/
dcmcf_uoctbl.msgsend /*出力メッセージ編集UOCアドレス*/
```

   UOC をまったく使用しない場合，このコーディングは必要ありません。

5. スタート関数を呼び出します。MCF メイン関数には必ずコーディングしてください。

   スタート関数を呼び出したあとは，MCF メイン関数に制御が戻りません。そのため，スタート関数のあとにコーディングした処理は実行されませんので，ご注意ください。

図 8-3　MCF メイン関数のディレクトリへの組み込み方法の概要



注※1

UOC を使用しない場合は，必要ありません。

注※2

mcfpltcp コマンドでリンケージします。

mcfpltcp コマンドの詳細については，TP1/NET/TCP/IP の「リリースノート」を参照してください。

注※3

TP1/NET/TCP/IP の実行形式プログラム名は，先頭が mcfu で始まる 8 文字以内の名称にしてください。

8.3 定義オブジェクトファイルの生成

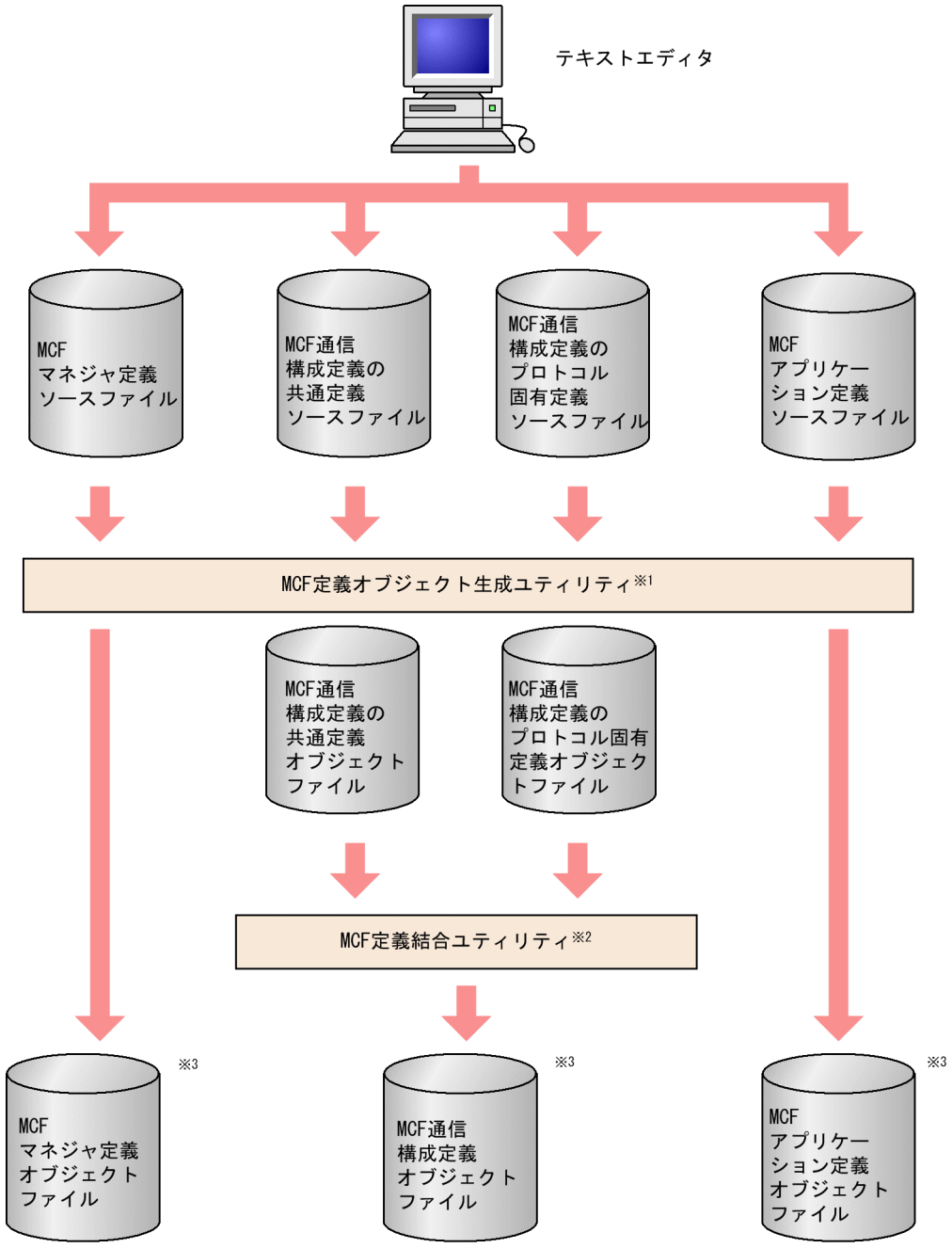
定義オブジェクトファイルを次の手順で生成します。

ただし，開始から再開始の間に定義オブジェクトファイルを変更しないでください。変更した場合，再開始時に正常に動作しない場合がありますのでご注意ください。

1. OS のテキストエディタを使用して，MCF の定義ファイルから，次に示す定義ソースファイルを作成します。
   - MCF マネジャ定義ソースファイル
   - MCF 通信構成定義の共通定義ソースファイル
   - MCF 通信構成定義の TP1/NET/TCP/IP のプロトコル固有定義ソースファイル
   - MCF アプリケーション定義ソースファイル
2. MCF 定義オブジェクト生成ユティリティを使用して，定義ソースファイルから，次に示すオブジェクトファイルを作成します。
   - MCF マネジャ定義オブジェクトファイル
   - MCF 通信構成定義の共通定義オブジェクトファイル
   - MCF 通信構成定義の TP1/NET/TCP/IP のプロトコル固有定義オブジェクトファイル
   - MCF アプリケーション定義オブジェクトファイル
3. MCF 定義結合ユティリティを使用して，MCF 通信構成定義の共通定義とプロトコル固有定義のオブジェクトファイルを結合し，次に示すオブジェクトファイルを作成します。
   - MCF 通信構成定義オブジェクトファイル

定義オブジェクトファイルの作成方法の概要を次の図に示します。

図 8-4　定義オブジェクトファイルの作成方法の概要



注※1

次に示すコマンドで生成します。

```
mcfXXXX△-i△〔パス名〕入力ファイル名
        △-o△〔パス名〕出力オブジェクトファイル名
```

mcfXXXX は，ソースファイルごとに異なります。

- mcfmngr：MCF マネジャ定義ソースファイル
- mcfcomn：MCF 通信構成定義の共通定義ソースファイル
- mcftcp ：MCF 通信構成定義のプロトコル（TP1/NET/TCP/IP）固有定義ソースファイル

- mcfapli：MCF アプリケーション定義ソースファイル

MCF 定義オブジェクト生成ユティリティの mcftcp コマンドについては，「6. システム定義」の「MCF 定義オブジェクトの生成」を，その他のコマンドについてはマニュアル「OpenTP1 システム定義」を参照してください。

注※2

次に示すコマンドで，MCF 通信構成定義の二つのオブジェクトファイルを結合します。

```
mcflink△-i△共通定義オブジェクトファイル名
        △TP1/NET/TCP/IP定義オブジェクトファイル名
        △-o△出力オブジェクトファイル名
```

注※3

定義オブジェクトファイルはシステム環境定義の DCCONFPATH で指定したディレクトリに格納してください。システム環境定義については，マニュアル「OpenTP1 システム定義」を参照してください。

9 障害対策

この章では，TP1/NET/TCP/IP 運用中に発生する障害と，TP1/NET/TCP/IP の対応処理，およびメッセージの処理について説明します。

9.1　障害の種類と対応処理

運用中に障害が発生すると，TP1/NET/TCP/IP はシステムを回復します。このとき，システム定義の指定によって，MCF イベント処理用 MHP も起動できます。

TP1/NET/TCP/IP 運用中の障害と対応処理について障害の種類ごとに説明します。

9.1.1　コネクション障害

表 9-1　コネクション障害と対応処理

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
コネクション確立時障害（クライアント型）	1. コネクション確立時障害を通知するメッセージログ（KFCA14803-E）を出力します。 2. CERREVT（コネクション確立不可能）を起動します。	「10.2.2　KFCA14803-E メッセージが出力された場合」を参照してください。
コネクション確立時障害（サーバ型）	1. サーバ側コネクション確立時障害を通知するメッセージログ（KFCA14834-E）を出力します。 2. 確立リトライ処理をします。	「10.2.6　KFCA14834-E または KFCA14835-E メッセージが出力された場合」を参照してください。
コネクション確立時リトライオーバ（サーバ型）	1. コネクション確立リトライオーバを通知するメッセージログ（KFCA14835-E）を出力します。	
コネクション切断	1. コネクション障害を通知するメッセージログ（KFCA14802-E）を出力します。 2. アドレス情報を通知するメッセージログ（KFCA14876-I）を出力します※1。 3. 問い合わせ応答中または継続問い合わせ応答中の場合，問い合わせ応答または継続問い合わせ応答を終了します。 4. CERREVT（コネクション切断）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※2）を出力します。	「10.2.1　KFCA14802-E メッセージが出力された場合」を参照してください。
メッセージ送信時障害	（一方送信メッセージの場合） 1. コネクション障害を通知するメッセージログ（KFCA14802-E）を出力します。 2. アドレス情報を通知するメッセージログ（KFCA14876-I）を出力します※1。 3. CERREVT（コネクション切断）を起動します。 4. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※2）を出力します。 5. 送信メッセージを出力キューに戻します。 6. メッセージの送信中断を通知するメッセージログ（KFCA10608-W）を出力します。	
	（応答メッセージの場合）	

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
メッセージ送信時障害	1. コネクション障害を通知するメッセージログ（KFCA14802-E）を出力します。 2. アドレス情報を通知するメッセージログ（KFCA14876-I）を出力します※1。 3. 問い合わせ応答中または継続問い合わせ応答中の場合，問い合わせ応答または継続問い合わせ応答を終了します。 4. CERREVT（コネクション切断）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※2）を出力します。 6. 送信メッセージを破棄します。 7. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。	「10.2.1　KFCA14802-E メッセージが出力された場合」を参照してください。
	（同期型メッセージの場合） 1. コネクション障害を通知するメッセージログ（KFCA14802-E）を出力します。 2. アドレス情報を通知するメッセージログ（KFCA14876-I）を出力します※1。 3. CERREVT（コネクション切断）を起動します。 4. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※2）を出力します。 5. UAP にエラーリターンします。	
メッセージ送信完了タイムアウト	（一方送信メッセージの場合） 1. メッセージ送信失敗を通知するメッセージログ（KFCA14815-E）を出力します。 2. 送信メッセージを出力キューに戻します。 3. メッセージの送信中断を通知するメッセージログ（KFCA10608-W）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（メッセージ送信完了タイムアウト）を起動します。	「10.2.3　KFCA14815-E メッセージが出力された場合」を参照してください。
	（応答メッセージの場合） 1. メッセージ送信失敗を通知するメッセージログ（KFCA14815-E）を出力します。 2. 送信メッセージを破棄します。 3. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 4. コネクションを解放します。 5. 問い合わせ応答中または継続問い合わせ応答中の場合，問い合わせ応答または継続問い合わせ応答を終了します。 6. CERREVT（メッセージ送信完了タイムアウト）を起動します。	

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
メッセージ送信完了タイムアウト	7. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※2）を出力します。 （同期型メッセージの場合） 1. メッセージ送信失敗を通知するメッセージログ（KFCA14815-E）を出力します。 2. 送信メッセージを破棄します。 3. UAP にエラーリターンします。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（メッセージ送信完了タイムアウト）を起動します。	「10.2.3　KFCA14815-E メッセージが出力された場合」を参照してください。

注※1

コネクション定義（mcftalccn -f）の cnerrlog オペランドの指定によって，KFCA14876-I を出力します。

注※2

コネクション定義（mcftalccn -f）の releaselog オペランドの指定によって，KFCA14801-I または KFCA14875-I のどちらかを出力します。

9.1.2　受信スケジュール関係障害（入力キュー，入力メッセージ編集 UOC）

表 9–2　受信スケジュール関係の障害と対応処理

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
アプリケーション名未定義	1. 入力キュー障害を通知するメッセージログ（KFCA10604-E）を出力します。 2. ERREVT1 を起動します。 3. 継続問い合わせ応答中の場合，継続問い合わせ応答を終了します。	システム定義を見直してください。
MCF イベント名未定義 ・ERREVT1 ・ERREVT2	1. 受信メッセージおよび MCF イベントを破棄します。 2. 継続問い合わせ応答中の場合，継続問い合わせ応答を終了します。	ありません。
MCF イベント名未定義 ・COPNEVT ・CCLSEVT ・CERREVT ・RHLDEVT ・MDELEVT	受信メッセージおよび MCF イベントを破棄します。	ありません。
MHP サービス，サービスグループ閉塞	1. 入力キュー障害を通知するメッセージログ（KFCA10604-E）を出力します。 2. ERREVT2 を起動します。	MHP サービス，サービスグループ閉塞の要因を取り除いてください。

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
入力キュー書き込み障害	1. 入力キュー障害を通知するメッセージログ（KFCA10604-E）を出力します。 2. ERREVT2 を起動します。	入力キュー書き込み障害の要因を取り除いてください。
入力メッセージ編集 UOC パラメタ不正	（一方送信メッセージの場合） 1. UOC パラメタ不正を通知するメッセージログ（KFCA10620-E）を出力します。 2. アプリケーション名取得失敗を通知するメッセージログ（KFCA10610-E）を出力します。 3. 受信メッセージを破棄します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（UOC 障害）を起動します。	入力メッセージ編集 UOC を見直してください。
	（問い合わせメッセージの場合） 1. UOC パラメタ不正を通知するメッセージログ（KFCA10620-E）を出力します。 2. アプリケーション名取得失敗を通知するメッセージログ（KFCA10610-E）を出力します。 3. 受信メッセージを破棄します。 4. 継続問い合わせ応答中の場合，継続問い合わせ応答を終了します。 5. 論理端末を閉塞します。 6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（UOC 障害）を起動します。	
	（同期型メッセージの場合） 1. UOC パラメタ不正を通知するメッセージログ（KFCA10620-E）を出力します。 2. 受信メッセージを破棄します。 3. UAP にエラーリターンします。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（UOC 障害）を起動します。	
入力メッセージ編集 UOC エラーリターン	（一方送信メッセージの場合） 1. UOC エラーリターンを通知するメッセージログ（KFCA10611-E）を出力します。 2. アプリケーション名取得失敗を通知するメッセージログ（KFCA10610-E）を出力します。 3. 受信メッセージを破棄します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。	入力メッセージ編集 UOC を見直してください。

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
入力メッセージ編集 UOC エラーリターン	6. CERREVT（UOC 障害）を起動します。 （問い合わせメッセージの場合） 1. UOC エラーリターンを通知するメッセージログ（KFCA10611-E）を出力します。 2. アプリケーション名取得失敗を通知するメッセージログ（KFCA10610-E）を出力します。 3. 受信メッセージを破棄します。 4. 継続問い合わせ応答中の場合，継続問い合わせ応答を終了します。 5. 論理端末を閉塞します。 6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（UOC 障害）を起動します。 （同期型メッセージの場合） 1. UOC エラーリターンを通知するメッセージログ（KFCA10611-E）を出力します。 2. 受信メッセージを破棄します。 3. UAP にエラーリターンします。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（UOC 障害）を起動します。	入力メッセージ編集 UOC を見直してください。
アプリケーション名形式不正	1. アプリケーション名取得失敗を通知するメッセージログ（KFCA10610-E）を出力します。 2. ERREVT1 を起動します。 3. 継続問い合わせ応答中の場合，継続問い合わせ応答を終了します。	アプリケーション名取得失敗の要因を取り除いてください。
次起動アプリケーション型不正（継続問い合わせ応答中に入力メッセージ編集 UOC で設定したアプリケーション名が cont 型でない）	1. 受信メッセージ破棄を通知するメッセージログ（KFCA14806-W）を出力します。 2. 継続問い合わせ応答を終了します。 3. コネクションを解放します。 4. CERREVT（次起動アプリケーション型不正）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	メッセージ入力編集 UOC およびシステム定義を見直してください。
受信バッファオーバフロー	1. メッセージ受信失敗を通知するメッセージログ（KFCA14816-E）を出力します。 2. 受信メッセージ破棄を通知するメッセージログ（KFCA14806-W）を出力します。 3. 受信バッファオーバフローを検出したときの追加情報を通知するメッセージログ（KFCA14865-I）を出力します。 4. コネクションを解放します。 5. 受信メッセージを破棄します。	「10.2.4　KFCA14816-E メッセージが出力された場合」を参照してください。

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
受信バッファオーバフロー	6. CERREVT（受信バッファオーバフロー）を起動します。 7. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	「10.2.4　KFCA14816-E メッセージが出力された場合」を参照してください。
受信バッファ数不足	1. 受信バッファの空き待ちを通知するメッセージログ（KFCA14877-I）を出力し，ほかで使用中の受信バッファの空きを待ちます。 （一定時間内に空きを検出した場合） 2.処理を続行します。 （一定時間内に空きを検出しなかった場合） 2.バッファ取得失敗を通知するメッセージログ（KFCA14847-E）を出力します。 3.コネクションを解放します。 4.受信メッセージを破棄します。 5.CERREVT（受信バッファ取得失敗）を起動します。 6.コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	「10.2.7　KFCA14877-I メッセージが出力された場合」を参照してください。
入力セグメント判定 UOC パラメタ不正	1. UOC パラメタ不正を通知するメッセージログ（KFCA14832-E）を出力します。 2. コネクションを解放します。 3. 受信メッセージを破棄します。 4. CERREVT（UOC 障害）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	入力セグメント判定 UOC，または相手システムを見直してください。
入力セグメント判定 UOC エラーリターン	1. UOC エラーを通知するメッセージログ（KFCA14808-E）を出力します。 2. コネクションを解放します。 3. 受信メッセージを破棄します。 4. CERREVT（UOC 障害）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	入力セグメント判定 UOC を見直してください。
後続セグメント受信監視タイムアウト	1. 後続メッセージ監視を通知するメッセージログ（KFCA14823-W）を出力します。 2. コネクションを解放します。 3. 受信メッセージを破棄します。 4. CERREVT（後続メッセージ監視タイムアウト）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	システム構成および運用を見直してください。
問い合わせ応答中の論理端末に対するメッセージの受信	1. 問い合わせ応答中の論理端末がメッセージを受信したことを通知するメッセージログ（KFCA14884-W）を出力します。 2. MDELEVT を起動します。	相手システム，または運用を見直してください。

注※

コネクション定義（mcftalccn -f）の releaselog オペランドの指定によって，KFCA14801-I または KFCA14875-I のどちらかを出力します。

9.1.3　送信スケジュール関係障害（出力キュー，出力メッセージ編集 UOC）

表 9-3　送信スケジュール関係の障害と対応処理

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
出力キュー読み出し障害	（一方送信メッセージの場合） 1. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 2. 送信メッセージを破棄します。 3. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（出力キュー読み出し障害）を起動します。	「10.1.1　取得情報」で示す保守情報を退避してください。
	（応答メッセージの場合） 1. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 2. 送信メッセージを破棄します。 3. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 4. 問い合わせ応答中または継続問い合わせ応答中の場合，問い合わせ応答または継続問い合わせ応答を終了します。 5. 論理端末を閉塞します。 6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（出力キュー読み出し障害）を起動します。	
	（同期型メッセージの場合） 1. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 2. 送信メッセージを破棄します。 3. UAP にエラーリターンします。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（出力キュー読み出し障害）を起動します。	
出力メッセージ編集 UOC エラーリターン	（一方送信メッセージの場合） 1. UOC エラーリターンを通知するメッセージログ（KFCA10611-E）を出力します。	運用，または出力メッセージ編集 UOC を見直してください。

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
出力メッセージ編集 UOC エラーリターン	2. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 3. 送信メッセージを破棄します。 4. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 5. 論理端末を閉塞します。 6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（UOC 障害）を起動します。	運用，または出力メッセージ編集 UOC を見直してください。
	（応答メッセージの場合） 1. UOC エラーリターンを通知するメッセージログ（KFCA10611-E）を出力します。 2. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 3. 送信メッセージを破棄します。 4. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 5. 問い合わせ応答中または継続問い合わせ応答中の場合，問い合わせ応答または継続問い合わせ応答を終了します。 6. 論理端末を閉塞します。 7. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 8. CERREVT（UOC 障害）を起動します。	
	（同期型メッセージの場合） 1. UOC エラーリターンを通知するメッセージログ（KFCA10611-E）を出力します。 2. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 3. 送信メッセージを破棄します。 4. UAP にエラーリターンします。 5. 論理端末を閉塞します。 6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（UOC 障害）を起動します。	
出力メッセージ編集 UOC パラメタ不正	（一方送信メッセージの場合） 1. UOC パラメタ不正を通知するメッセージログ（KFCA10620-E）を出力します。 2. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 3. 送信メッセージを破棄します。 4. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 5. 論理端末を閉塞します。	出力メッセージ編集 UOC を見直してください。

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
出力メッセージ編集 UOC パラメタ不正	6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（UOC 障害）を起動します。	出力メッセージ編集 UOC を見直してください。
	（応答メッセージの場合） 1. UOC パラメタ不正を通知するメッセージログ（KFCA10620-E）を出力します。 2. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 3. 送信メッセージを破棄します。 4. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 5. 問い合わせ応答中または継続問い合わせ応答中の場合，問い合わせ応答または継続問い合わせ応答を終了します。 6. 論理端末を閉塞します。 7. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 8. CERREVT（UOC 障害）を起動します。	
	（同期型メッセージの場合） 1. UOC パラメタ不正を通知するメッセージログ（KFCA10620-E）を出力します。 2. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 3. 送信メッセージを破棄します。 4. UAP にエラーリターンします。 5. 論理端末を閉塞します。 6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（UOC 障害）を起動します。	
メッセージ消し込み障害	1. メッセージ送信完了障害を通知するメッセージログ（KFCA10617-E）を出力します。 2. 処理を続行します。	「10.1.1　取得情報」で示す保守情報（$DCDIR/spool 下）を退避してください。
送信バッファオーバフロー	（一方送信メッセージの場合） 1. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 2. 送信メッセージを破棄します。 3. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（送信バッファオーバフロー）を起動します。	システム定義，または UAP を見直してください。
	（応答メッセージの場合）	

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
送信バッファオーバフロー	1. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 2. 送信メッセージを破棄します。 3. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 4. 問い合わせ応答中または継続問い合わせ応答中の場合，問い合わせ応答または継続問い合わせ応答を終了します。 5. 論理端末を閉塞します。 6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（送信バッファオーバフロー）を起動します。	システム定義，または UAP を見直してください。
	（同期型メッセージの場合） 1. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 2. 送信メッセージを破棄します。 3. UAP にエラーリターンします。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（送信バッファオーバフロー）を起動します。	
送信バッファ数不足	（一方送信メッセージの場合） 1. バッファ取得失敗を通知するメッセージログ（KFCA10618-E）を出力します。 2. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 3. 送信メッセージを破棄します。 4. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 5. 論理端末を閉塞します。 6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（送信バッファ取得失敗）を起動します。	システム定義を見直してください。
	（応答メッセージの場合） 1. バッファ取得失敗を通知するメッセージログ（KFCA10618-E）を出力します。 2. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 3. 送信メッセージを破棄します。 4. 送信メッセージ破棄を通知するメッセージログ（KFCA10607-W）を出力します。 5. 問い合わせ応答中または継続問い合わせ応答中の場合，問い合わせ応答または継続問い合わせ応答を終了します。 6. 論理端末を閉塞します。	

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
送信バッファ数不足	7. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 8. CERREVT（送信バッファ取得失敗）を起動します。	システム定義を見直してください。
	（同期型メッセージの場合） 1. バッファ取得失敗を通知するメッセージログ（KFCA10618-E）を出力します。 2. 出力キュー障害を通知するメッセージログ（KFCA10605-E）を出力します。 3. 送信メッセージを破棄します。 4. UAP にエラーリターンします。 5. 論理端末を閉塞します。 6. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 7. CERREVT（送信バッファ取得失敗）を起動します。	

9.1.4 メッセージ送達確認関係障害

DCCM とのメッセージ送達確認を行う場合，および任意の相手システムとのメッセージ送達確認を行う場合に発生する可能性がある障害に対する TP1/NET/TCP/IP の処理を，以降の表に示します。

表 9-4 メッセージ送達確認関係の障害と対応処理（mcftalccn -u delichk=dccm2s, dccm2m，dccm3s，dccm3m の場合）

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
応答専用データ用送信バッファ数不足	1. バッファ取得失敗を通知するメッセージログ（KFCA10618-E）を出力します。 2. 受信メッセージを破棄します。 3. コネクションを解放します。 4. CERREVT（応答専用データ用送信バッファ取得失敗）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	定義を見直してください。
不正データ受信（11 バイト未満のメッセージ受信，セグメント情報不正）	1. 不正データ受信を通知するメッセージログ（KFCA14848-E）を出力します。 2. 受信メッセージを破棄します。 3. コネクションを解放します。 4. CERREVT（不正データ受信）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	相手システムを見直してください。
メッセージ ID 不一致	1. 受信メッセージ破棄を通知するメッセージログ（KFCA14806-W）を出力します。	相手システムを見直してください。

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
メッセージ ID 不一致	2. 処理を続行します。	相手システムを見直してください。
応答を待ち合わせていないときに，応答専用データを受信	1. 受信メッセージ破棄を通知するメッセージログ（KFCA14806-W）を出力します。 2. 処理を続行します。	相手システムを見直してください。
送受信メッセージ衝突	（dccm2s，dccm3s の場合） 1. メッセージの送信中断を通知するメッセージログ（KFCA10608-W）を出力します。 2. 処理を続行します。	相手システムまたは運用を見直してください。
	（dccm2m の場合） 1. 受信メッセージ破棄を通知するメッセージログ（KFCA14806-W）を出力します。 2. 送信メッセージを出力キューに戻します。 3. メッセージの送信中断を通知するメッセージログ（KFCA10608-W）を出力します。 4. 受信メッセージを破棄します。 5. コネクションを解放します。 6. CERREVT（送受信メッセージ衝突）を起動します。 7. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	
	（dccm3m の場合） 1. 受信メッセージ破棄を通知するメッセージログ（KFCA14806-W）を出力します。 2. 処理を続行します。	
応答専用データを待ち合わせたまま，送信完了監視タイムアウト	1. メッセージ送信失敗を通知するメッセージログ（KFCA14815-E）を出力します。 2. 送信メッセージを出力キューに戻します。 3. メッセージの送信中断を通知するメッセージログ（KFCA10608-W）を出力します。 4. コネクションを解放します。 5. CERREVT（応答専用データ受信監視タイムアウト）を起動します。 6. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	定義または相手システムを見直してください。
応答専用データ送信処理中の送信完了監視タイムアウト	1. メッセージ送信失敗を通知するメッセージログ（KFCA14815-E）を出力します。 2. コネクションを解放します。 3. CERREVT（応答専用データ送信完了監視タイムアウト）を起動します。 4. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	定義または相手システムを見直してください。

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
応答専用データ送信処理中の一方送信メッセージ受信	1. メッセージ受信失敗を通知するメッセージログ（KFCA14816-E）を出力します。 2. 受信メッセージを破棄します。 3. コネクションを解放します。 4. CERREVT（応答専用データ送信処理中一方送信メッセージの受信）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	相手システムを見直してください。
受信メッセージ通知失敗	表 9-6 を参照してください。	定義または運用を見直してください。

注※

コネクション定義（mcftalccn -f）の releaselog オペランドの指定によって，KFCA14801-I または KFCA14875-I のどちらかを出力します。

表 9-5　メッセージ送達確認関係の障害と対応処理（mcftalccn -u delichk=use の場合）

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
受信メッセージ判定 UOC 未登録	1. UOC 未登録を通知するメッセージログ（KFCA14844-E）を出力します。 2. OpenTP1 が異常終了します。	受信メッセージ判定 UOC を作成し，登録してください。
送達確認メッセージ用送信バッファ数不足	1. バッファ取得失敗を通知するメッセージログ（KFCA10618-E）を出力します。 2. 受信メッセージを破棄します。 3. コネクションを解放します。 4. CERREVT（送達確認メッセージ用送信バッファ取得失敗）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	定義を見直してください。
送達確認メッセージを待ち合わせていないときに，受信メッセージ判定 UOC にてメッセージ種別に送達確認メッセージを指定	1. 受信メッセージ破棄を通知するメッセージログ（KFCA14806-W）を出力します。 2. 処理を続行します。	相手システムを見直してください。
送達確認メッセージを待ち合わせたまま，送信完了監視タイムアウト	1. メッセージ送信失敗を通知するメッセージログ（KFCA14815-E）を出力します。 2. 送信メッセージを出力キューに戻します。 3. メッセージの送信中断を通知するメッセージログ（KFCA10608-W）を出力します。 4. コネクションを解放します。 5. CERREVT（送達確認メッセージ受信監視タイムアウト）を起動します。 6. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	定義または相手システムを見直してください。

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
送達確認メッセージ送信処理中の送達確認メッセージ送信完了監視タイムアウト	1. メッセージ送信失敗を通知するメッセージログ（KFCA14815-E）を出力します。 2. コネクションを解放します。 3. CERREVT（送達確認メッセージ送信完了監視タイムアウト）を起動します。 4. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	定義または相手システムを見直してください。
送達確認メッセージ送信処理中の一方送信メッセージ受信	1. メッセージ受信失敗を通知するメッセージログ（KFCA14816-E）を出力します。 2. 受信メッセージを破棄します。 3. コネクションを解放します。 4. CERREVT（送達確認メッセージ送信処理中一方送信メッセージの受信）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	相手システムを見直してください。
受信メッセージ判定 UOC でメッセージ種別に破棄メッセージを指定	1. 受信メッセージ判定 UOC による受信メッセージ破棄を通知するメッセージログ（KFCA14849-I）を出力します。 2. MDELEVT を起動します。 3. 処理を続行します。	なし。
受信メッセージ判定 UOC でメッセージ種別にコネクション解放を指定	1. 受信メッセージ判定 UOC によるコネクション解放を通知するメッセージログ（KFCA14850-I）を出力します。 2. 受信メッセージを破棄します。 3. コネクションを解放します。 4. CERREVT（受信メッセージ判定 UOC でのコネクション解放指示）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	相手システムを見直してください。
受信メッセージ判定 UOC で送信処理の継続可否に送信停止を指定	1. 受信メッセージ判定 UOC による論理端末閉塞を通知するメッセージログ（KFCA14851-I）を出力します。 2. 送信メッセージを出力キューに戻します。 3. メッセージの送信中断を通知するメッセージログ（KFCA10608-W）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA14809-I）を出力します。 6. CERREVT（受信メッセージ判定 UOC での送信停止指示）を起動します。	なし。
受信メッセージ判定 UOC エラーリターン	1. UOC エラーリターンを通知するメッセージログ（KFCA14808-E）を出力します。 2. 受信メッセージを破棄します。 3. コネクションを解放します。 4. CERREVT（UOC 障害）を起動します。	受信メッセージ判定 UOC を見直してください。

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
受信メッセージ判定 UOC エラーリターン	5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	受信メッセージ判定 UOC を見直してください。
受信メッセージ判定 UOC パラメタ不正	1. UOC パラメタ不正を通知するメッセージログ（KFCA14852-E）を出力します。 2. 受信メッセージを破棄します。 3. コネクションを解放します。 4. CERREVT（受信メッセージ判定 UOC でのパラメタ不正）を起動します。 5. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	受信メッセージ判定 UOC を見直してください。
受信メッセージ通知失敗	表 9-6 を参照してください。	定義または運用を見直してください。

注※

コネクション定義（mcftalccn -f）の releaselog オペランドの指定によって，KFCA14801-I または KFCA14875-I のどちらかを出力します。

表 9-6　受信メッセージ通知失敗に対する TP1/NET/TCP/IP の処理

障害の内容	ERREVT1	ERREVT2	TP1/NET/TCP/IP の処理
アプリケーション名未定義	○	−	1. 送達確認メッセージを送信します。 2. 処理を続行します。
	×	−	1. 受信メッセージを破棄します。 2. コネクションを解放します。 3. CERREVT（受信メッセージ通知失敗）を起動します。 4. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。
	未定義	−	
スケジュール失敗	−	○	1. 送達確認メッセージを送信します。 2. 処理を続行します。
	−	×	1. 受信メッセージを破棄します。 2. コネクションを解放します。 3. CERREVT（受信メッセージ通知失敗）を起動します。 4. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。
	−	未定義	
入力メッセージ編集 UOC エラーリターン	−	−	1. 受信メッセージを破棄します。 2. コネクションを解放します。 3. CERREVT（UOC 障害）を起動します。

障害の内容	ERREVT1	ERREVT2	TP1/NET/TCP/IP の処理
入力メッセージ編集 UOC エラーリターン	－	－	4. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。
入力メッセージ編集 UOC パラメタ不正	－	－	1. 受信メッセージを破棄します。 2. コネクションを解放します。 3. CERREVT（UOC 障害）を起動します。 4. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。

（凡例）

○：通知成功

×：通知失敗

－：通知なし

注※

コネクション定義（mcftalccn -f）の releaselog オペランドの指定によって，KFCA14801-I または KFCA14875-I のどちらかを出力します。

9.1.5 UAP の障害

表 9–7 UAP の障害と対応処理

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
メッセージ受信前の UAP 異常終了	ERREVT2 を起動します。	UAP を見直してください。
メッセージ受信後の UAP 異常終了	ERREVT3 を起動します。	

9.1.6 TP1/NET/TCP/IP の障害

表 9–8 TP1/NET/TCP/IP の障害と対応処理

障害の内容	TP1/NET/TCP/IP の処理	ユーザの処理
内部論理矛盾，または他プログラムプロダクトとのインタフェースエラー	1. 論理エラーを通知するメッセージログ（KFCA14811-E）を出力します。 2. コネクションを解放します。 3. CERREVT（内部論理矛盾）を起動します。 4. コネクション解放を通知するメッセージログ（KFCA14801-I または KFCA14875-I※）を出力します。	「10.1.1 取得情報」で示す保守情報（$DCDIR/spool 下）を退避してください。

注※

コネクション定義（mcftalccn -f）の releaselog オペランドの指定によって，KFCA14801-I または KFCA14875-I のどちらかを出力します。

9.2　コネクション障害時の処理

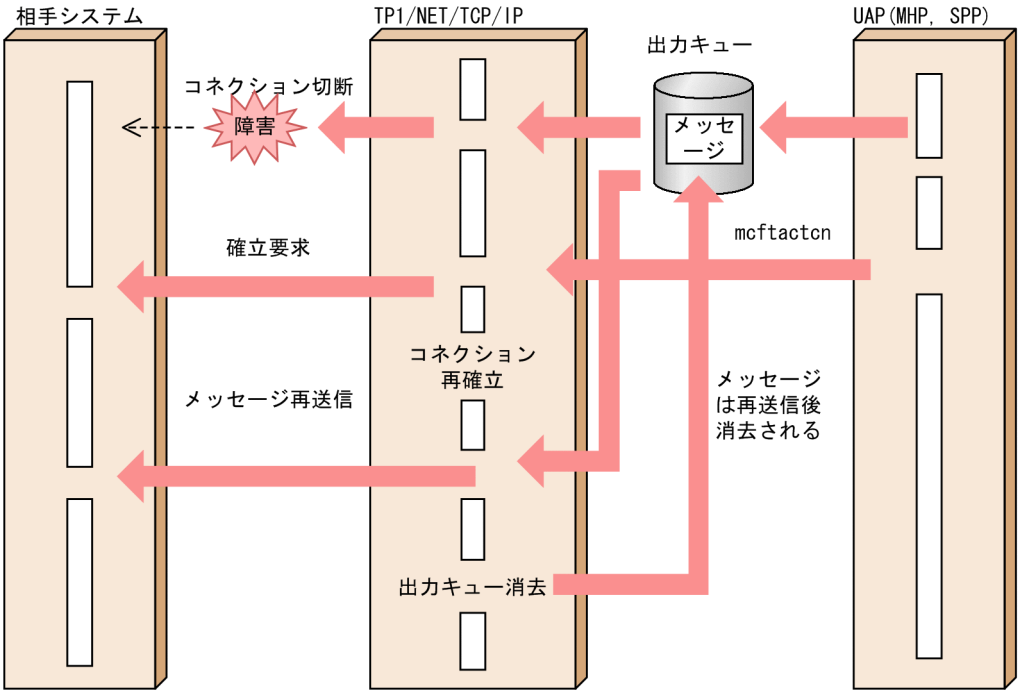
コネクションが切断された場合，クライアント型コネクションであれば障害の原因を取り除いたあと，運用コマンド（mcftactcn）の入力，または API（dc_mcf_tactcn 関数もしくは CBLDCMCF('TACTCN △△')）の発行によってコネクションを再確立します。サーバ型コネクションであれば，自動的に相手システムからのコネクション確立を待ちます。

9.2.1　メッセージ送信時のコネクション障害

メッセージはロールバックされ，コネクション再確立時に再送信されます。

一方メッセージ送信時のコネクション障害の例を次の図に示します。

図 9-1　メッセージ送信時のコネクション障害時の処理

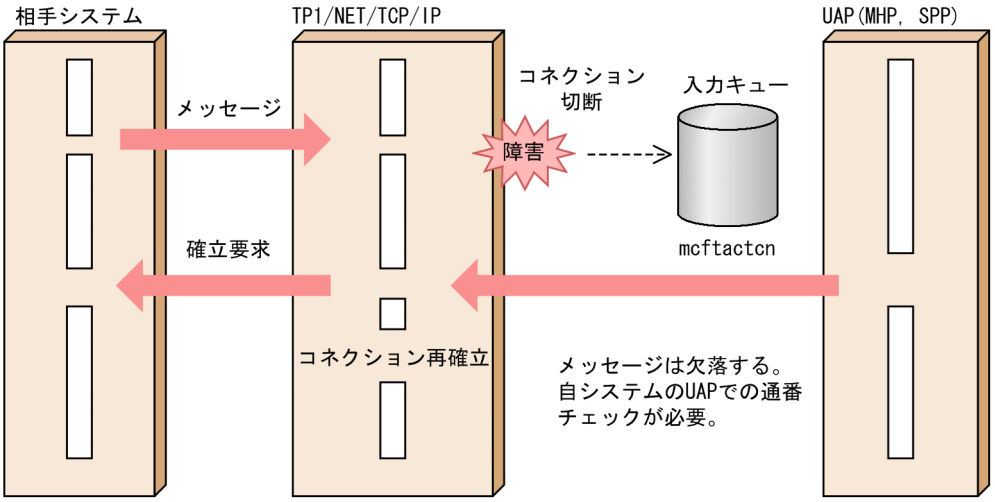


9.2.2　メッセージ受信時のコネクション障害

メッセージは欠落します。

メッセージ受信時のコネクション障害の例を次の図に示します。

図 9-2　メッセージ受信時のコネクション障害時の処理



9.3　ユーザアプリケーションプログラム異常終了時の処理

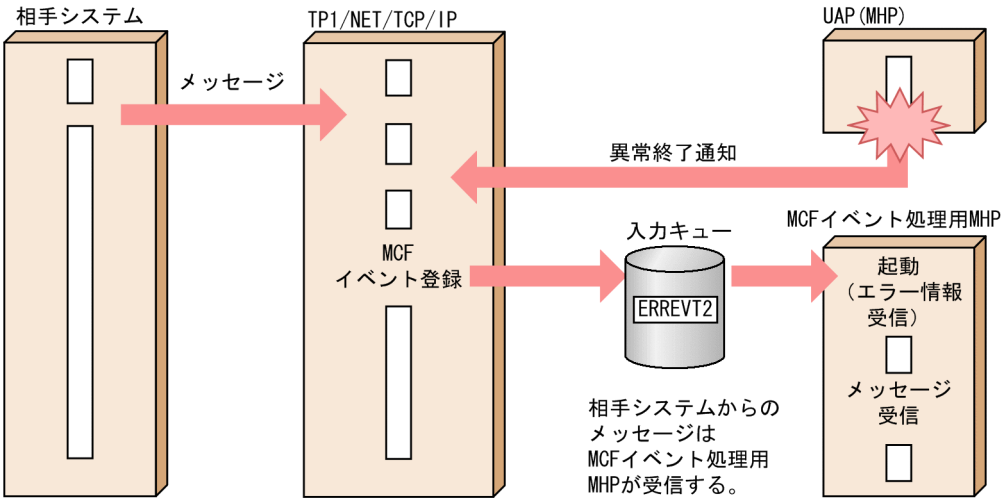
UAP が異常終了した場合，TP1/NET/TCP/IP は入力キューに MCF イベントを登録し，MCF イベント処理用 MHP を起動します。起動された MCF イベント処理用 MHP は，エラー情報と入力されたメッセージを受信します。

9.3.1　メッセージ受信前の UAP 異常終了

TP1/NET/TCP/IP は入力キューに MCF イベント ERREVT2 を登録し，MCF イベント処理用 MHP を起動します。起動された MCF イベント処理用 MHP は，エラー情報と入力されたメッセージを受信します。

メッセージ受信前の UAP 異常終了の例を次の図に示します。

図 9-3　メッセージ受信前の UAP 異常終了時の処理

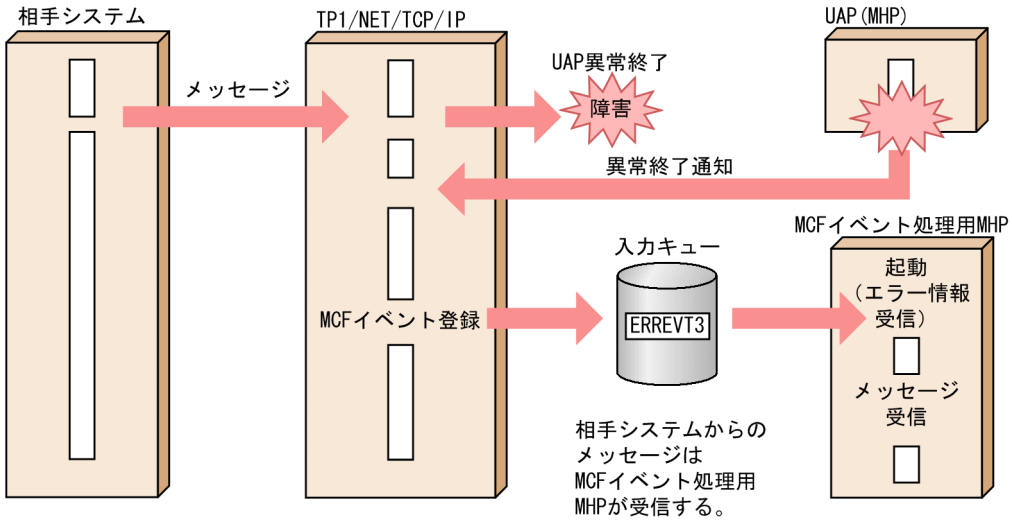


9.3.2　メッセージ受信後の UAP 異常終了

TP1/NET/TCP/IP は入力キューに MCF イベント ERREVT3 を登録し，MCF イベント処理用 MHP を起動します。起動された MCF イベント処理用 MHP は，エラー情報と入力されたメッセージを受信します。

メッセージ受信後の UAP 異常終了の例を次の図に示します。

図 9-4　メッセージ受信後の UAP 異常終了時の処理

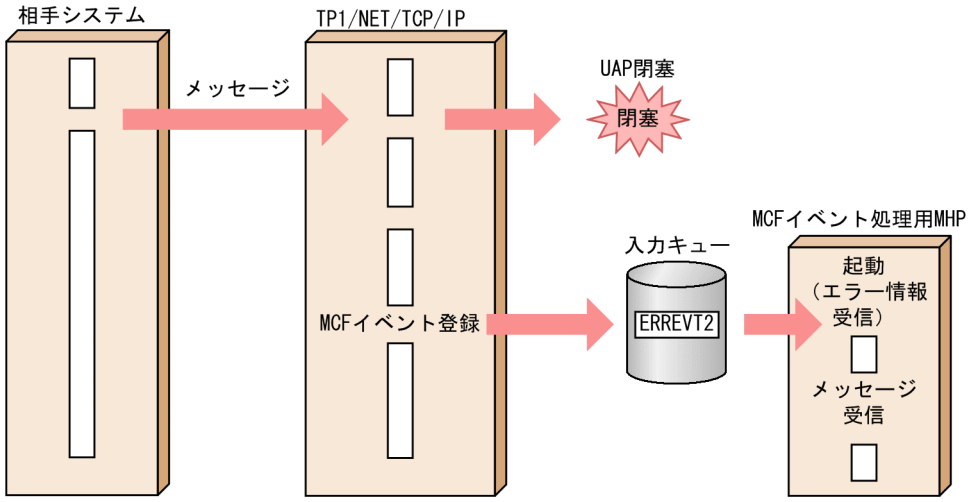


9.4　ユーザアプリケーションプログラム閉塞時の処理

UAP が閉塞していて使用できない場合，TP1/NET/TCP/IP は入力キューに MCF イベント ERREVT2 を登録し，MCF イベント処理用 MHP を起動します。起動された MCF イベント処理用 MHP は，エラー情報と入力されたメッセージを受信します。

UAP 閉塞による障害の例を次の図に示します。

図 9-5　UAP 閉塞による障害時の処理

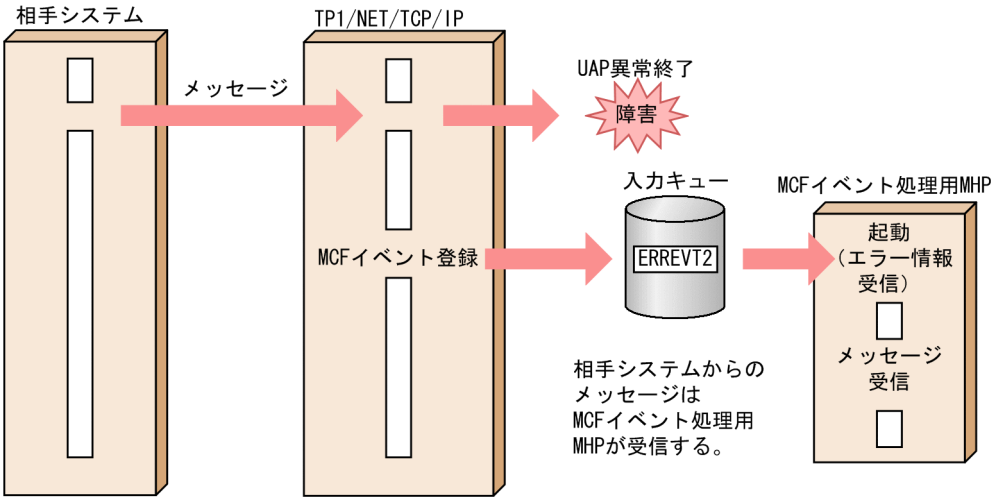


9.5　入力キュー障害時の処理

メッセージを入力キューに書き込むときに障害が発生した場合，TP1/NET/TCP/IP は入力キューに MCF イベント ERREVT2 を登録し，MCF イベント処理用 MHP を起動します。起動された MCF イベント処理用 MHP は，エラー情報と入力されたメッセージを受信します。

入力キュー障害の例を次の図に示します。

図 9-6　入力キュー障害時の処理

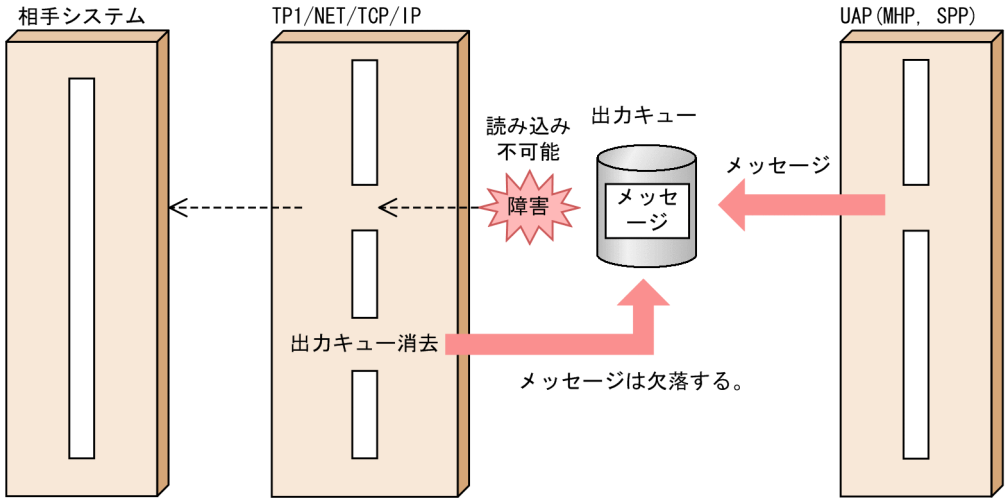


9.6　出力キュー障害時の処理

メッセージを出力キューから読み込むときに障害が発生した場合，メッセージは欠落します。

出力キュー障害の例を次の図に示します。

図 9-7　出力キュー障害時の処理



10 トラブル発生時の調査手順

障害が発生したときに取得する情報および調査手順について説明します。

「10.1　取得情報と確認事項」では障害発生時に取得する情報や確認する事項のうち，あらゆる障害に共通する内容について説明します。

「10.2　調査手順」では具体的な障害事例を基に，障害発生時の調査手順について説明します。

10.1　取得情報と確認事項

ここでは障害発生時に取得する情報や確認する事項のうち，あらゆる障害に共通する内容について説明します。

10.1.1　取得情報

障害要因を調査する場合に必要な情報を示します。

ここでは調査に必要な基本的な情報を記載しています。「10.2　調査手順」で説明する個々の障害事例に記載していない障害については，ここで説明する情報を基に障害の要因を調査してください。

障害要因の調査に必要な情報を次の表に示します。なお，通信先相手システムも OpenTP1 システムである場合，現象発生ノードだけでなく，通信先相手システムのノードについても次の表の資料を取得してください。

表 10–1　障害要因の調査に必要な情報

取得する情報	備考
現象発生ノードの betran.log（OpenTP1 の標準出力および標準エラー出力先ファイル）	UNIX 版の場合，$DCDIR/bin/prcout に指定したファイルに出力されます。指定していない場合は，次のファイルに出力されます。 • $DCDIR/spool/prclog1 • $DCDIR/spool/prclog2 Windows 版の場合，イベントログ（アプリケーションログおよびシステムログ）を取得してください。また，標準出力リダイレクト機能を使用しているときは，システム環境定義の redirect_file_name オペランドに指定したファイルも取得してください。指定していない場合は，次のファイルに出力されます。 • %DCDIR%¥spool¥prclog1 • %DCDIR%¥spool¥prclog2
現象発生ノードの syslog ファイル	Windows 版の場合，取得する必要はありません。
現象発生ノードの$DCDIR/spool 配下のファイル	spool 配下のすべてのファイルを取得するのが困難な場合は，spool 配下をすべて退避したあと，次に示すファイルを取得してください。 • $DCDIR/spool/cmdlog/cmdlog* • $DCDIR/spool/mcftXXXnn（MCF トレース※） XXX：MCF 識別子 nn：通し番号 なお，MCF トレースファイルの見積もり式については，「6.　システム定義」の「MCF トレースファイルの見積もり式」を参照してください。
現象発生ノードの$DCDIR/tmp 配下のファイル	システム共通定義の prc_current_work_path オペランドを指定している場合は，その配下もすべて取得してください。

取得する情報	備考
現象発生ノードの$DCDIR/conf 配下および$DCCONFPATH 配下のファイル	$DCUAPCONFPATH を設定している場合は，$DCUAPCONFPATH 配下のファイルも取得してください。定義ファイルを取得するときに，ユーザサービス定義も取得してください。
次に示す定義 • MCF マネジャ定義 • MCF 通信構成定義（共通定義およびプロトコル固有定義） • MCF アプリケーション定義	定義の格納先，およびファイル名は任意に指定されたものになります。また，オンラインで使用するオブジェクトファイルではなく，オブジェクト生成前の定義ファイルを取得してください。
システムサービス共通情報定義（$DCDIR/lib/sysconf/mcf）	－
システムサービス情報定義（$DCDIR/lib/sysconf/配下の先頭 4 文字が「mcfu」で始まるファイル名）	－
現象発生ノードの$DCDIR のディレクトリ情報	UNIX 版の場合，次に示すコマンドを実行し，情報を取得してください。 `ls -laR $DCDIR`

（凡例）

－：該当しません。

注※

MCF トレースの取得については，次の注意が必要です。

- 現象発生後，速やかに MCF トレースを退避してください。現象発生から長時間経過した場合，MCF トレースファイルがラップアラウンドし，該当時間帯のトレース情報が失われるおそれがあります。
- オンライン中に MCF トレースを取得する場合，mcftswptr コマンドを実行し，MCF トレースをスワップしてください。なお，コマンドを実行する前にディスクに残っている MCF トレースはあらかじめ退避してください。
- トレース環境定義でディスク出力機能を使用していない（mcfttrc -t disk=no）場合，MCF トレースを取得できないため，障害要因の調査ができないおそれがあります。ディスク出力機能を使用しないで MCF トレースを取得したい場合は，mcftstrtr コマンドを実行してください。

注意事項

障害要因の調査に必要な情報には，パイプファイルなどの特殊ファイルが含まれています。このような特殊ファイルをファイル転送コマンドで直接取得した場合，正常に処理されないことがあります。tar コマンドで一つのファイルに結合してから資料を採取してください。また，ファイル転送をするときは，転送モードに注意が必要です。

10.1.2　障害が発生したときの確認事項

障害が発生したときの確認事項を説明します。

- OpenTP1 製品のバージョン

  正確なバージョンを確認してください。

  修正版の場合

  バージョン表記に追加されたコードも確認してください。

  例：07-00-01，06-03-/A など

  UNIX 版の場合

  日立 PP インストーラのコマンド（hitachi_setup）で，製品のバージョンを確認できます。日立 PP インストーラの使用方法は，ご使用の製品のリリースノートを参照してください。

  Windows 版の場合

  ご使用の製品の readme ファイルで，製品のバージョンを確認できます。%DCDIR%¥readme フォルダに格納されている，readme ファイルを参照してください。

- 現象発生日時
- ノード間の時刻差

  複数ノードが関連する場合だけ確認してください。

- 現象発生時のマシンの負荷状態

  CPU が高負荷であるなど，わかる範囲で確認してください。

- 現象発生時の相手システムおよびネットワーク機器の状態

  機器の起動状態やエラーの有無など，わかる範囲で確認してください。

- 現象が発生した環境

  本番環境かどうかを確認してください。本番環境の場合，業務への影響度を確認してください。

- 現象発生直前の OpenTP1 環境および運用に対する変更点

  ネットワーク構成，OS のパラメタ，OpenTP1 システムの環境および運用などを確認してください。

10.2　調査手順

具体的な障害事例を基に，障害発生時の調査手順について説明します。

10.2.1　KFCA14802-E メッセージが出力された場合

KFCA14802-E メッセージが出力された場合の調査手順，および対処について説明します。

(1)　現象

次のメッセージが出力されて，コネクションが解放されました。

```
KFCA14802-E mmm コネクション障害が発生しました。
コネクション名＝ aa....aa 関数＝ bb....bb 詳細エラーコード＝ cc....cc
```

（凡例）

mmm：MCF 通信プロセス識別子
aa....aa：コネクション名
bb....bb：TCP/IP ソケットの関数名
cc....cc：詳細エラーコード

(2)　現象発生時の確認事項

- 現象発生日時
- 現象発生時の相手システムおよびネットワーク機器の状態
- 現象発生直前の接続先相手システムに対する変更点

(3)　取得情報

- 障害要因の調査に必要な情報
  「10.1.1　取得情報」を参照してください。
- 自システムと相手システム間のパケットトレース

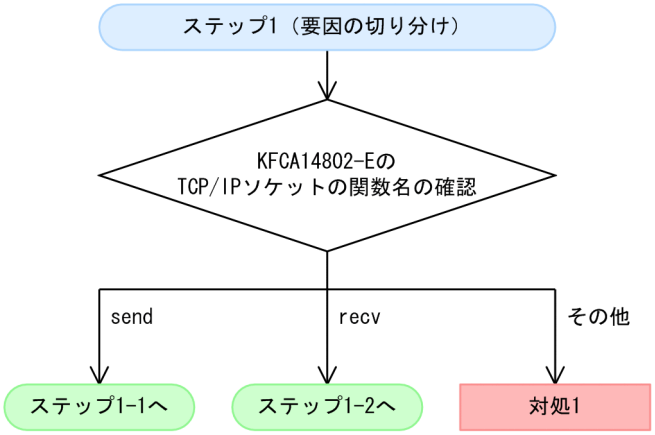
(4)　原因の調査と対処

この障害の主な原因を次に示します。

- 相手システムのネットワーク機器，または自システムと相手システム間のネットワーク機器からのコネクション強制解放
- 自システムのネットワーク障害の検出

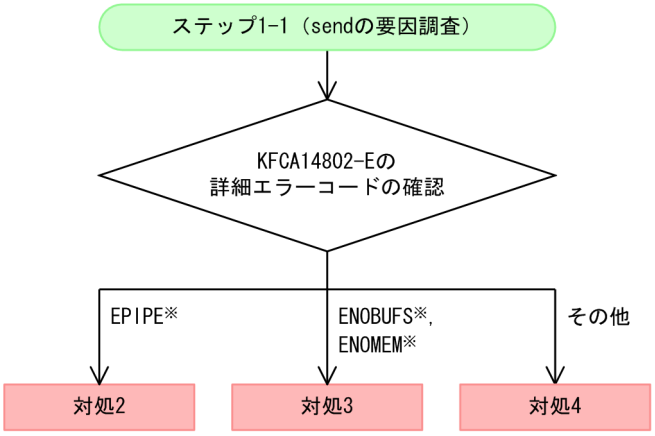
次に示すフローに従って原因の調査と対処をしてください。

図 10–1　KFCA14802-E が出力された場合の調査手順（ステップ 1）



対処	内容
対処 1	KFCA14802-E の TCP/IP ソケットの関数名と詳細エラーコードから，ご使用の OS のマニュアルで詳細エラーコードの意味を確認し，想定される要因に該当するかどうかを調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

図 10–2　KFCA14802-E が出力された場合の調査手順（ステップ 1-1）

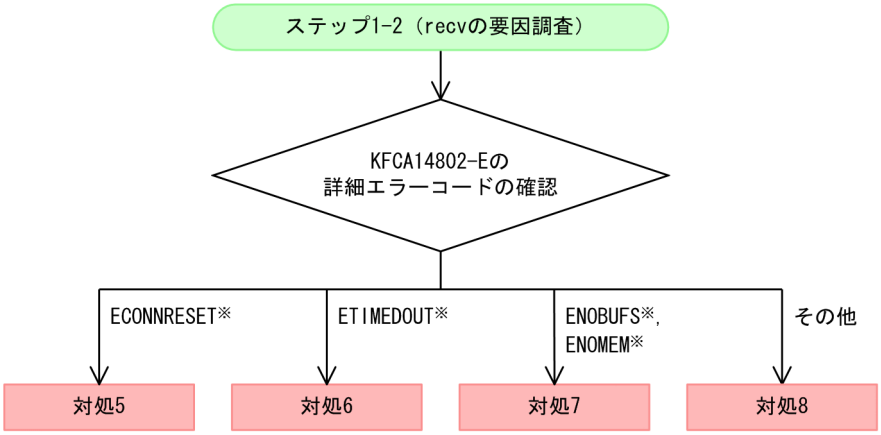


注※

OS が返した errno です。実際にメッセージに出力される値は，使用する OS によって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

対処	内容
対処 2	メッセージ送信中にコネクションを強制解放されたことが原因です。 相手システム，または自システムと相手システム間のネットワーク機器を調査してください。
対処 3	自システムの資源不足が原因です。 自システムの負荷状況を調査してください。
対処 4	ご使用の OS のマニュアルで send 関数に対応する詳細エラーコードの意味を確認し，想定される要因に該当するかどうかを調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

図 10-3　KFCA14802-E が出力された場合の調査手順（ステップ 1-2）



注※

OS が返した errno です。実際にメッセージに出力される値は，使用する OS によって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

対処	内容
対処 5	コネクションが強制解放されたことが原因です。※1 相手システム，または自システムと相手システム間のネットワーク機器を調査してください。
対処 6	自システムがネットワーク障害を検出したことが原因です。※2 相手システム，または自システムと相手システム間のネットワーク機器を調査してください。
対処 7	自システムの資源不足が原因です。 自システムの負荷状況を調査してください。
対処 8	ご使用の OS のマニュアルで，recv 関数に対応する詳細エラーコードの意味を確認し，想定される要因に該当するかどうか調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

注※1

自システムからのメッセージ送信と相手システムからのコネクション解放のすれ違いによって発生することもあります。

注※2

TCP/IP 上のパケット再送リトライアウトやキープアライブなどが該当します。

10.2.2　KFCA14803-E メッセージが出力された場合

KFCA14803-E メッセージが出力された場合の調査手順，および対処について説明します。

(1)　現象

次のメッセージが出力されて，相手システムとコネクションが確立できませんでした。

KFCA14803-E mmm コネクション確立時に障害が発生しました。

```
コネクション名＝ aa....aa 関数＝ bb....bb 詳細エラーコード＝ cc....cc
```

（凡例）

mmm：MCF 通信プロセス識別子
aa....aa：コネクション名
bb....bb：TCP/IP ソケットの関数名
cc....cc：詳細エラーコード

(2)　現象発生時の確認事項

- 現象発生日時
- 相手システムとの接続実績
- 現象発生時の相手システムおよびネットワーク機器の状態
- 現象発生直前の OpenTP1 環境に対する変更点
- 現象発生直前の接続先相手システムに対する変更点

(3)　取得情報

- 障害要因の調査に必要な情報
  「10.1.1　取得情報」を参照してください。
- hosts ファイル（DNS を使用している場合は不要）
- 自システムの状態
  自システムで netstat -a コマンドを実行し，取得した実行結果を残してください。
- 相手システムの状態
  相手システムで netstat -a コマンドを実行し，取得した実行結果を残してください。
- 自システムと相手システム間のパケットトレース

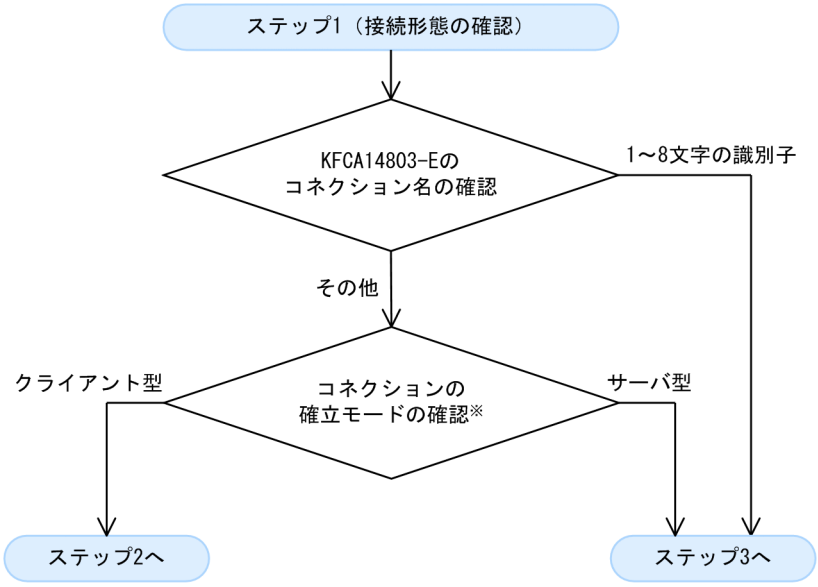
(4)　原因の調査と対処

この障害の主な原因を次に示します。

- コネクション定義に指定した相手システムの IP アドレスまたはポート番号が不正
- 相手システム，または自システムと相手システム間のネットワーク機器の障害

次に示すフローに従って原因の調査と対処をしてください。

図 10-4　KFCA14803-E が出力された場合の調査手順（ステップ 1）

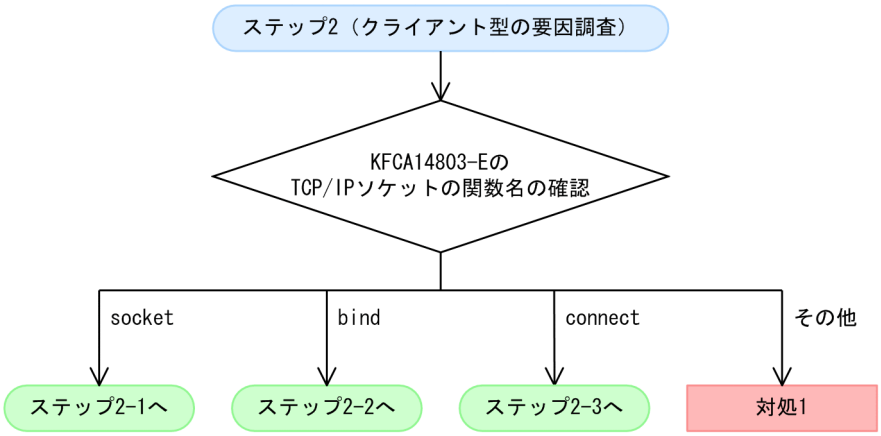


注※

コネクション定義の確立モード（mcftalccn -y mode）の指定値を確認してください。指定値が「client」の場合はクライアント型，指定値が「server」の場合はサーバ型です。

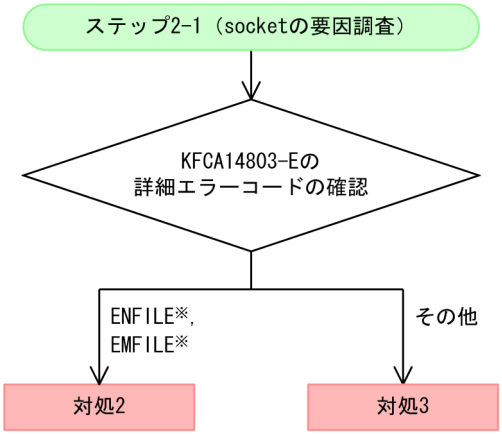
コネクションの確立モードは，運用コマンド（mcftlsln -t）を実行して確認することもできます。出力内容が「C」の場合はクライアント型，出力内容が「S」の場合はサーバ型です。

図 10-5　KFCA14803-E が出力された場合の調査手順（ステップ 2）



対処	内容
対処 1	KFCA14803-E の TCP/IP ソケットの関数名と詳細エラーコードから，ご使用の OS のマニュアルで詳細エラーコードの意味を確認し，想定される要因に該当するかどうかを調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

図 10-6　KFCA14803-E が出力された場合の調査手順（ステップ 2-1）



注※

OS が返した errno です。実際にメッセージに出力される値は，使用する OS によって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

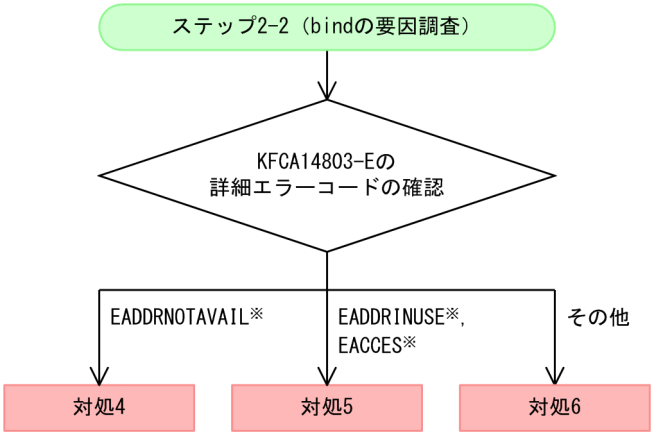
また，詳細エラーコードが ENFILE の場合はシステム全体で使用できるファイル記述子の上限を，EMFILE の場合は 1 プロセスで使用できるファイル記述子の上限を見直してください。

対処	内容
対処 2	MCF 通信プロセスで使用できるファイル記述子の指定値が不足していることが原因です。システムサービス共通情報定義の max_open_fds オペランドの指定値，および OS のカーネルパラメタ※の指定値を見直してください。
対処 3	ご使用の OS のマニュアルで socket 関数に対応する詳細エラーコードの意味を確認し，想定される要因に該当するかどうかを調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

注※

カーネルパラメタの指定値を変更する場合は，余裕を持った値を指定してください。

図 10-7　KFCA14803-E が出力された場合の調査手順（ステップ 2-2）



注※

OS が返した errno です。実際にメッセージに出力される値は，使用する OS によって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

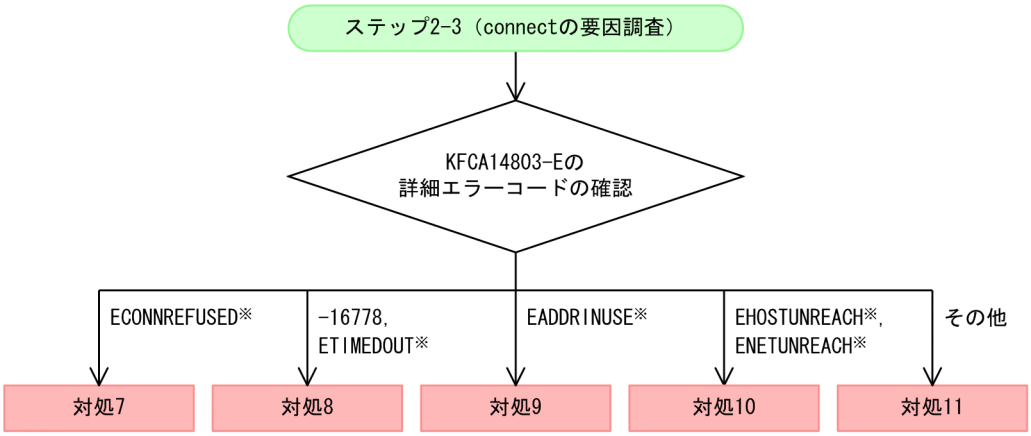
対処	内容
対処 4	割り当てられない IP アドレスを指定したことが原因です。 コネクション定義の自システムの IP アドレス（mcftalccn -r ipaddr），またはホスト名（mcftalccn -r hostname）の指定値を見直してください。 直前に KFCA14868-I が出力されている場合は，dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')で設定した自システムの IP アドレスの設定値を見直してください。
対処 5	他のプログラムが使用しているポート番号を指定したことが原因です。 コネクション定義の自システムのポート番号（mcftalccn -r portno）の指定値を見直してください。 直前に KFCA14868-I が出力されている場合は，dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')で設定した自システムのポート番号の設定値を見直してください。※
対処 6	ご使用の OS のマニュアルで，bind 関数に対応する詳細エラーコードの意味を確認し，想定される要因に該当するかどうか調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

注※

他のアプリケーションとの重複を避けるため，OS が任意に割り当てるポート番号（動的ポートまたは短命ポートと呼ばれるポート番号）を使用しないでください。

OS が任意に割り当てるポート番号の範囲は，OS の種別やバージョンによって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

図 10–8　KFCA14803-E が出力された場合の調査手順（ステップ 2-3）



注※

OS が返した errno です。実際にメッセージに出力される値は，使用する OS によって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

対処	内容
対処 7	次に示す要因のどれかで，確立要求を相手システムが拒否したことが原因です。 • 相手システムが使用していないポート番号にコネクション確立要求を行った。

対処	内容
対処 7	• 相手システムの処理能力を超えるコネクション確立要求が集中した。 次に示す調査または見直しをしてください。 • コネクション定義の相手システムの IP アドレス（mcftalccn -o oipaddr）またはホスト名（mcftalccn -o ohostname）およびポート番号（mcftalccn -o oportno）の指定値を見直してください。 • 直前に KFCA14868-I が出力されている場合，dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')で設定した相手システムの IP アドレスおよびポート番号の設定値を見直してください。 • 相手システム内の接続先アプリケーションの起動状態，および障害が発生した時間帯の負荷状態を調査してください。
対処 8	次に示す要因のどれかで，コネクション確立要求がタイムアウトしたことが原因です。 • 相手システムの電源が未投入などにより，指定した IP アドレスがネットワーク上に存在しない。 • 相手システムの処理能力を超えるコネクション確立要求が集中した。 • OS やネットワーク機器のパケットフィルタリング機能でコネクション確立要求が破棄された。 次に示す調査または見直しをしてください。 • コネクション定義の相手システムの IP アドレス（mcftalccn -o oipaddr）またはホスト名（mcftalccn -o ohostname）の指定値を見直してください。 直前に KFCA14868-I が出力されている場合，dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')で設定した相手システムの IP アドレスの設定値を見直してください。 • 相手システムの起動状態，および該当時間帯の負荷状態を調査してください。 • 自システムと相手システム間のネットワークの状態を調査してください。 • 相手システムおよびネットワーク機器のパケットフィルタリング機能の設定を見直してください。 • 詳細エラーコードが-16778 の場合，コネクション定義のコネクション確立時監視時間（mcftalccn -b concmptim）の指定値が適切であるか見直してください。※1
対処 9	TIME_WAIT 状態※2 のコネクションに確立要求を行ったことが原因です。 コネクション定義のコネクション確立障害時の確立再試行回数（mcftalccn -b bretrycnt）および確立再試行間隔（mcftalccn -b bretryint）の指定値を見直してください。
対処 10	到達できないネットワークの IP アドレスを指定したことが原因です。 次に示す調査または見直しをしてください。 • コネクション定義の相手システムの IP アドレス（mcftalccn -o oipaddr）またはホスト名（mcftalccn -o ohostname）の指定値を見直してください。 • 直前に KFCA14868-I が出力されている場合，dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')で設定した相手システムの IP アドレスの設定値を見直してください。 • 自システムと相手システム間のネットワーク機器を調査してください。
対処 11	ご使用の OS のマニュアルで，connect 関数に対応する詳細エラーコードの意味を確認し，想定される要因に該当するかどうか調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

注※1

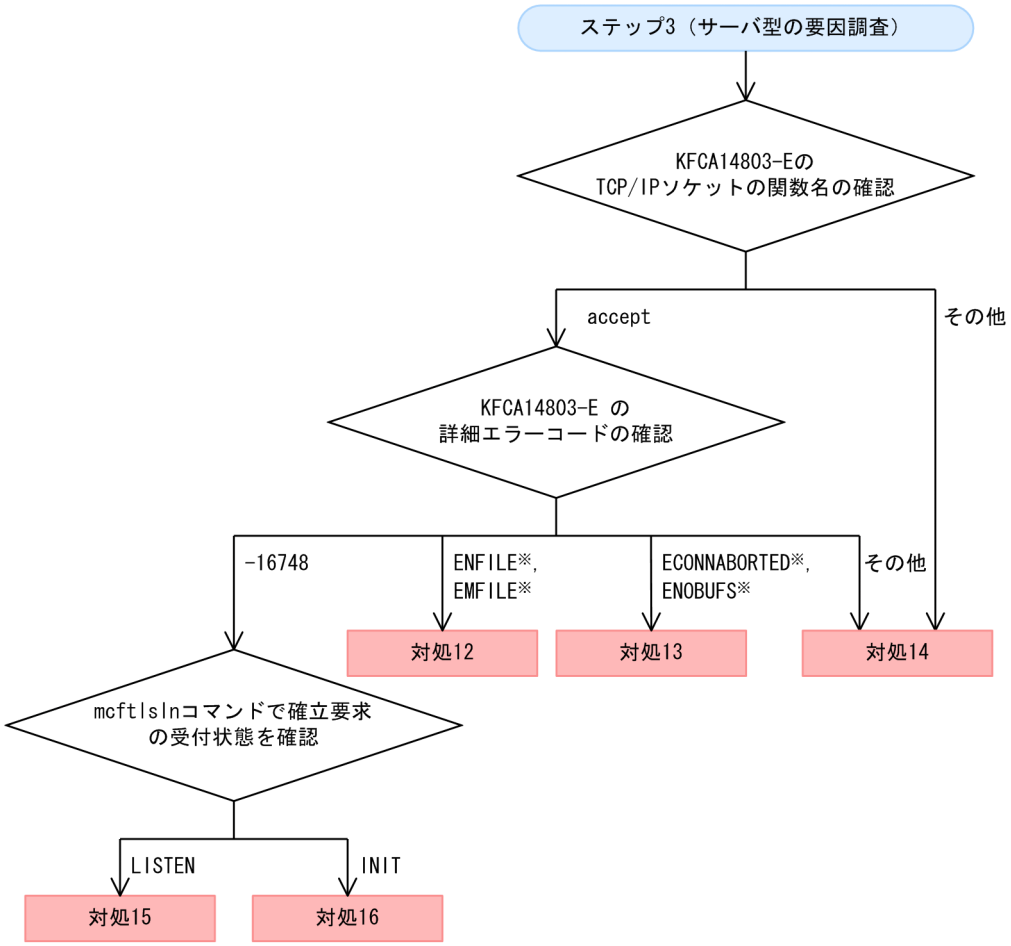
OpenTP1 システムの時間の精度は 1 秒です。

注※2

自システムからコネクションを解放すると，OS のコネクションの状態が TIME_WAIT 状態となります。TIME_WAIT 状態の所要時間は OS によって異なります。

自システムからコネクションを解放した直後に再度コネクションを確立する必要がある場合は，コネクション定義の自システムのポート番号（mcftalccn -r portno）は指定しないで，OS が自動的に割り当てたポート番号を使用してください。

図 10-9　KFCA14803-E が出力された場合の調査手順（ステップ 3）



注※

OS が返した errno です。実際にメッセージに出力される値は，使用する OS によって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

また，詳細エラーコードが ENFILE の場合はシステム全体で使用できるファイル記述子の上限を，EMFILE の場合は 1 プロセスで使用できるファイル記述子の上限を見直してください。

対処	内容
対処 12	MCF 通信プロセスで使用できるファイル記述子の指定値が不足していることが原因です。 システムサービス共通情報定義の max_open_fds オペランドの指定値および OS のカーネルパラメタ※1 の指定値を見直してください。
対処 13	TP1/NET/TCP/IP がコネクション確立要求を受け付けた直後，相手システムからコネクションを強制解放されたことが原因です。 相手システムのネットワーク機器，および自システムと相手システム間のネットワーク機器を調査してください。 また，ポートスキャンが行われていたかどうかを調査してください。

対処	内容
対処 14	KFCA14803-E の TCP/IP ソケットの関数名，および詳細エラーコードから，ご使用の OS のマニュアルで詳細エラーコードの意味を確認し，想定される要因に該当するかどうか調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。
対処 15	相手システムの IP アドレスおよびポート番号がコネクション定義に定義されていないことが原因です。 相手システムの IP アドレスとポート番号については，KFCA14803-E の直後に出力された KFCA14854-I で確認してください。 次に示す調査または見直しをしてください。 • コネクション定義の相手システムの IP アドレス（mcftalccn -o oipaddr），ホスト名（mcftalccn -o ohostname）※2 およびポート番号（mcftalccn -o oportno）※3 の指定値を見直してください。 • コネクション定義の自システムのポート番号（mcftalccn -r portno）の指定値と同じポート番号を使用するアプリケーション（自システム内の他の TP1/NET/TCP/IP の通信プロセスを含みます）が自システム内に存在していないか調査してください。※4
対処 16	コネクション確立要求の受け付けを開始していないことが原因です。 mcftonln コマンドを実行して確立要求の受け付けを開始してください。

注※1

カーネルパラメタの指定値を変更する場合は，余裕を持った値を指定してください。

注※2

相手システムをホスト名で指定している場合，自システムの hosts ファイルまたは DNS の指定内容を確認してください。ただし，OpenTP1 のオンライン中に相手システムの IP アドレスが変更された場合，OpenTP1 を再立ち上げするまで反映されないので注意が必要です。

注※3

相手システムのポート番号が不定の場合，相手システムのポート番号（mcftalccn -o oportno）には free を指定してください。

注※4

自システムのポート番号が他のアプリケーション（自システム内の他の TP1/NET/TCP/IP の通信プロセスを含みます）と重複していると，他のアプリケーションに対するコネクション確立要求を受け付けるおそれがあります。

自システム内に TP1/NET/TCP/IP の MCF 通信プロセスが複数存在する場合は，KFCA14834-E が出力されていないか確認してください。

10.2.3 KFCA14815-E メッセージが出力された場合

KFCA14815-E メッセージが出力された場合の調査手順，および対処について説明します。

(1) 現象

次のメッセージが出力されて，メッセージの送信に失敗しました。

```
KFCA14815-E mmm メッセージの送信に失敗しました。
コネクション名＝ aa....aa 論理端末名称＝ bb....bb 障害コード＝ cc....cc
```

（凡例）

mmm：MCF 通信プロセス識別子

aa....aa：コネクション名

bb....bb：論理端末名称
cc....cc：障害コード（詳細エラーコード）

(2)　現象発生時の確認事項

- 現象発生日時
- 現象発生時の相手システムの状態
- 現象発生直前の接続先相手システムに対する変更点

(3)　取得情報

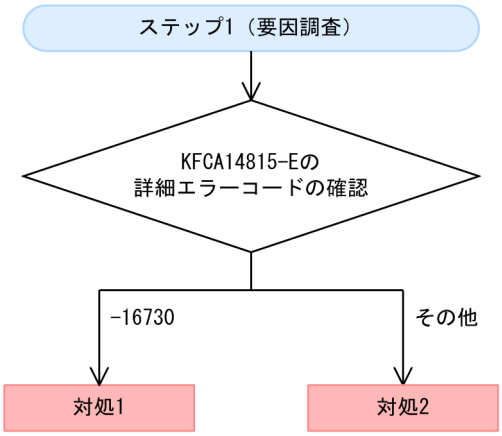
- 障害要因の調査に必要な情報
  「10.1.1　取得情報」を参照してください。
- 自システムと相手システム間のパケットトレース

(4)　原因の調査と対処

KFCA14815-E は，メッセージ送信完了監視時間（mcftalccn -b sndcmptim）がタイムアウトした場合に出力されます。この障害は主に相手システムのスローダウンなどによって，自システムが送信したメッセージを相手システムが受信できていないことが原因で発生します。

次に示すフローに従って原因の調査と対処をしてください。

図 10–10　KFCA14815-E が出力された場合の調査手順（ステップ 1）



対処	内容
対処 1	メッセージ送信完了監視時間がタイムアウトしたことが原因です。 次に示す調査または見直しをしてください。 • コネクション定義のメッセージ送信完了監視時間（mcftalccn -b sndcmptim）の指定値が適切であるか見直してください。※ • 相手システムの動作を調査してください。 • 相手システムのアプリケーションが TP1/NET/TCP/IP の MCF 通信プロセスの場合，バッファグループ定義の受信バッファ面数（mcftbuf -g count）の指定値が適切かどうか見直してください。

対処	内容
対処 2	情報を取得してサポートセンタへ問い合わせてください。

注※

OpenTP1 システムの時間の精度は 1 秒です。

10.2.4　KFCA14816-E メッセージが出力された場合

KFCA14816-E メッセージが出力された場合の調査手順，および対処について説明します。

(1)　現象

次のメッセージが出力されて，コネクションが解放されました。

```
KFCA14816-E mmm メッセージの受信に失敗しました。
コネクション名＝ aa....aa 論理端末名称＝ bb....bb 詳細エラーコード＝ cc....cc
```

（凡例）

mmm：MCF 通信プロセス識別子
aa....aa：コネクション名
bb....bb：論理端末名称
cc....cc：詳細エラーコード

(2)　現象発生時の確認事項

- 現象発生日時
- 相手システムとの接続実績
- 現象発生時の相手システムおよびネットワーク機器の状態
- 現象発生直前の接続先相手システムに対する変更点

(3)　取得情報

- 障害要因の調査に必要な情報
  「10.1.1　取得情報」を参照してください。
- 自システムと相手システム間のパケットトレース

(4)　原因の調査と対処

この障害の主な原因を次に示します。

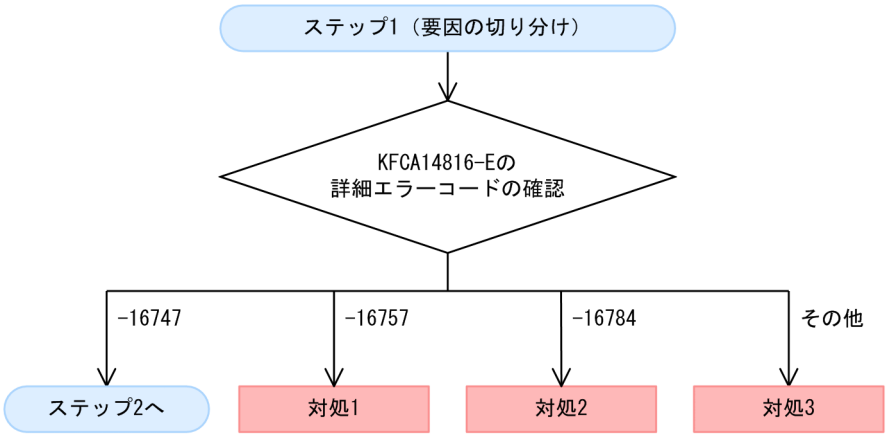
- 受信バッファのバッファ長不足（mcftbuf -g length）
- 相手システムから受信したメッセージの形式が不正

- MCF 通信プロセスに組み込んだ入力セグメント判定 UOC の設定誤り

相手システムから受信したメッセージの内容は，パケットトレースで確認してください。パケットトレースを取得していない場合，MCF トレースを取得してサポートセンタへ問い合わせてください。

次に示すフローに従って原因の調査と対処をしてください。

図 10–11　KFCA14816-E が出力された場合の調査手順（ステップ 1）



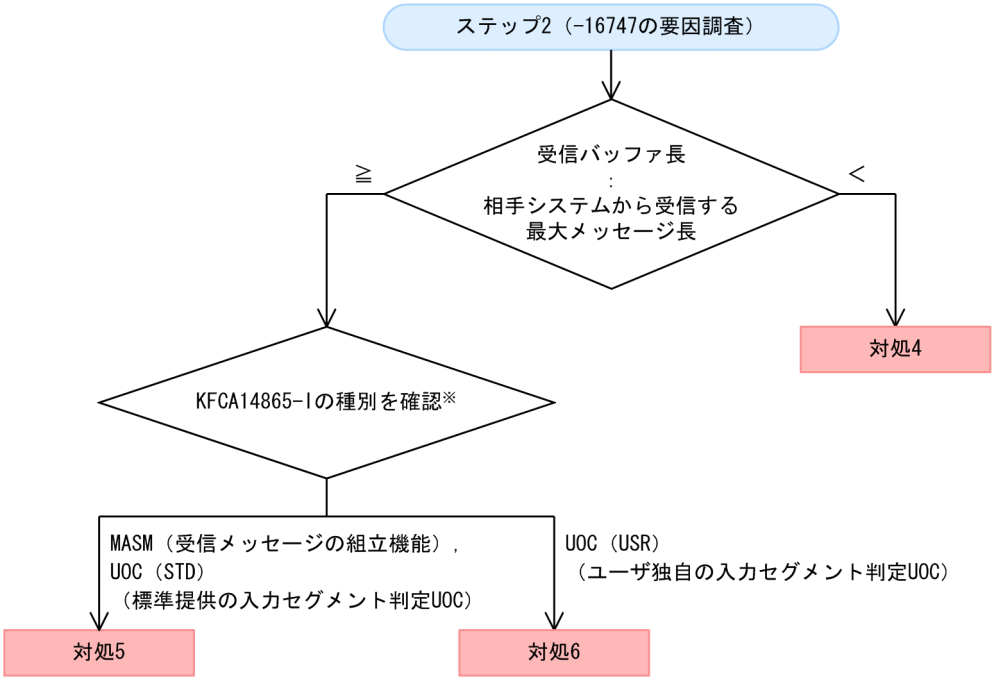
対処	内容
対処 1	受信メッセージの組み立て機能を使用（mcftalccn -u masm=yes）している場合に，メッセージ長（メッセージの先頭 4 バイト）が 4 バイト以下※に設定されたメッセージを相手システムから受信したことが原因です。 相手システムの動作を調査してください。
対処 2	メッセージ送達確認機能を使用（(mcftalccn -u delichk）に nouse 以外の値を指定）している場合に，送達確認メッセージ（応答専用データ）の送信処理中に相手システムから一方送信メッセージを受信したことが原因です。 相手システムの動作を調査してください。
対処 3	情報を取得してサポートセンタへ問い合わせてください。

注※

メッセージ長に負の値が設定されている場合も該当します。

メッセージ長はビックエンディアンで設定する必要があります。相手システムがリトルエンディアンの OS（Windows や Linux など）の場合，メッセージ長をビックエンディアンに変換して設定しているか確認してください。

図 10–12　KFCA14816-E が出力された場合の調査手順（ステップ 2）



注※

KFCA14865-I は，TP1/NET/TCP/IP 07-02 以降の場合に出力されます。

TP1/NET/TCP/IP 07-01 以前の場合，受信メッセージの組み立て機能（mcftalccn -u masm）の使用の有無，および MCF 通信サービスのメイン関数を参照し，メッセージの組み立て方式を確認してください。

対処	内容
対処 4	受信バッファのバッファ長不足が原因です。 バッファグループ定義のバッファ長（mcftbuf -g length）を見直してください。
対処 5	メッセージ長（メッセージの先頭 4 バイト）に受信バッファ長を超える値※が設定されたメッセージを相手システムから受信したことが原因です。 相手システムの動作を調査してください。
対処 6	TP1/NET/TCP/IP の受信済みメッセージと，入力セグメント判定 UOC で設定した残っている該当メッセージのサイズ（dctcp_sguoc_prot の rest_data_size）の和が，受信バッファ長を超えていることが原因です。 入力セグメント判定 UOC および相手システムの動作を調査してください。

注※

メッセージ長はビックエンディアンで設定する必要があります。相手システムがリトルエンディアンの OS（Windows や Linux など）の場合，メッセージ長をビックエンディアンに変換して設定しているか確認してください。

10.2.5　KFCA14830-E または KFCA14841-E メッセージが出力された場合

KFCA14830-E または KFCA14841-E メッセージが出力された場合の調査手順，および対処について説明します。

(1)　現象

次のどちらかのメッセージが出力されて，相手システムとコネクションが確立できませんでした。

```
KFCA14830-E mmm ポートフリー型コネクションからのコネクション確立要求が定義数を超えたため拒否します。
クライアントアドレス（aa....aa，bb....bb） 自ポート番号＝ cc....cc
```

（凡例）

mmm：MCF 通信プロセス識別子

aa....aa：相手システムのホスト名

bb....bb：相手システムの IP アドレス

cc....cc：自システムのポート番号

注

コネクションが確立済みの場合に，同じポート番号に対して同じ相手システムから繰り返し確立要求を受け付けたとき，KFCA14830-E は一度だけ出力します。

いったんコネクションを解放すると，再び KFCA14830-E は出力されるようになります。

また，異なる自ポート番号に対して繰り返し確立要求を受け付けている場合，それぞれの自ポート番号ごとに KFCA14830-E を一度だけ出力します。

```
KFCA14841-E mmm 未確立のコネクションが存在しないため，確立要求を拒否しました。
自ポート番号=aa....aa 相手アドレス=bb....bb 相手ポート番号=cc....cc
```

（凡例）

mmm：MCF 通信プロセス識別子

aa....aa：自システムのポート番号

bb....bb：相手システムの IP アドレス

cc....cc：相手システムのポート番号

(2)　現象発生時の確認事項

- 現象発生日時
- 現象発生時の相手システムおよびネットワーク機器の状態
- 現象発生直前の OpenTP1 環境に対する変更点
- 現象発生直前の接続先相手システムに対する変更点

(3)　取得情報

- 障害要因の調査に必要な情報

  「10.1.1　取得情報」を参照してください。

- ネットワークの状態
  mcftlsln -t コマンドを実行し，取得した実行結果を残してください。
- コネクションの状態
  mcftlscn コマンドを実行し，取得した実行結果を残してください。
- 仕掛り中のトランザクションの状態
  間隔をあけて複数回 trnls -ta コマンドを実行し，取得した実行結果を残してください。

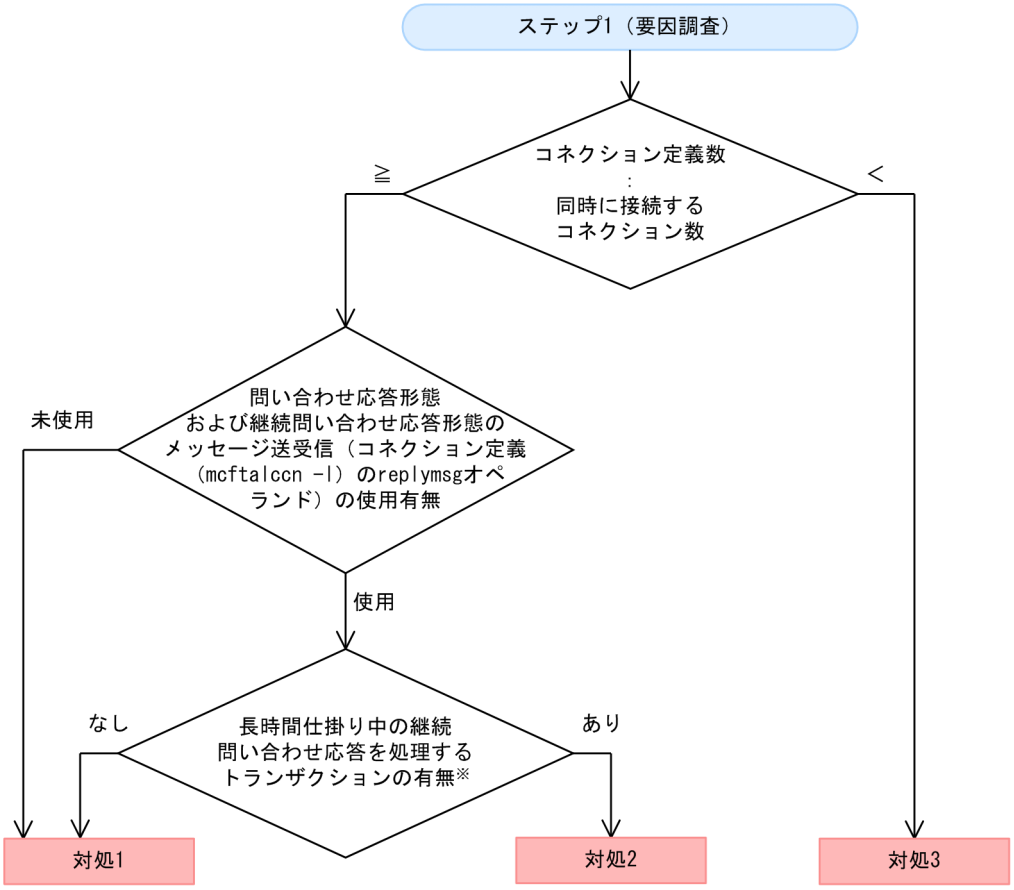
(4) 原因の調査と対処

この障害の主な原因を次に示します。

- 同時に接続するコネクション数が MCF 通信構成定義に定義したコネクション数を超えている
- 自システムがコネクション障害を検出していない状態で，コネクション障害を検出した相手システムからコネクションの確立要求を再度受け付けた
- 継続問い合わせ応答を処理するトランザクションが長時間終了しない

次に示すフローに従って原因の調査と対処をしてください。

図 10–13　KFCA14830-E または KFCA14841-E が出力された場合の調査手順



注※　継続問い合わせ応答のUAP実行中かどうかは，mcftlscnコマンドの出力結果（問い合わせ応答状態がC，UAP実行中状態がUの場合）で確認できます。仕掛り中のトランザクションの有無はtrnls -taコマンドで確認できます。

対処	内容
対処 1	自システムがコネクション障害を検出していない状態で，コネクション障害を検出した相手システムからコネクションの確立要求を再度受け付けたことが原因です。 OpenTP1 システムと相手システムの状態不一致を解消するため，次の中から自システムに必要な対策をしてください。 • UAP でコネクションの生存確認（定期的なメッセージ送受信など）をする • 無通信状態監視を使用する（コネクション定義（mcftalccn -k）の notrftime オペランドに 0 以外の値を指定） • キープアライブを使用する（コネクション定義（mcftalccn -k）の keepalive オペランドに yes を指定） • コネクションリプレースを使用する（コネクション定義（mcftalccn -h）の chgconn オペランドに replace を指定）
対処 2	継続問い合わせ応答の UAP 実行中のため，コネクションを割り当てられないことが原因です。 UAP の動作を調査してください。 仕掛り中のトランザクションが終了するまで待ち合わせないで，相手システムとコネクションを確立したい場合，mcftendct -f コマンドを実行してください。
対処 3	MCF 通信構成定義のコネクション定義数が不足していることが原因です。

対処	内容
対処 3	MCF 通信構成定義のコネクション定義数および同時に接続するコネクション数を見直してください。

10.2.6　KFCA14834-E または KFCA14835-E メッセージが出力された場合

KFCA14834-E または KFCA14835-E メッセージが出力された場合の調査手順，および対処について説明します。

(1)　現象

次のどちらかのメッセージが出力されて，サーバ型コネクションの確立準備に失敗しました。

```
KFCA14834-E mmm サーバ側コネクション確立準備に失敗しました。
コネクション名＝ aa....aa 自ポート番号＝ bb....bb
関数＝ cc....cc 詳細エラーコード＝ dd....dd
```

（凡例）

mmm：MCF 通信プロセス識別子
aa....aa：コネクション名
bb....bb：自システムのポート番号
cc....cc：TCP/IP ソケットの関数名
dd....dd：詳細エラーコード

```
KFCA14835-E mmm サーバ側コネクション確立準備のリトライが失敗しました。
コネクション名＝ aa....aa 自ポート番号＝ bb....bb
関数＝ cc....cc 詳細エラーコード＝ dd....dd
```

（凡例）

mmm：MCF 通信プロセス識別子
aa....aa：コネクション名
bb....bb：自システムのポート番号
cc....cc：TCP/IP ソケットの関数名
dd....dd：詳細エラーコード

(2)　現象発生時の確認事項

- 現象発生日時
- 現象発生時の自システムの状態
- 現象発生直前の OpenTP1 環境に対する変更点

(3)　取得情報

- 障害要因の調査に必要な情報

「10.1.1　取得情報」を参照してください。

- 自システムの状態

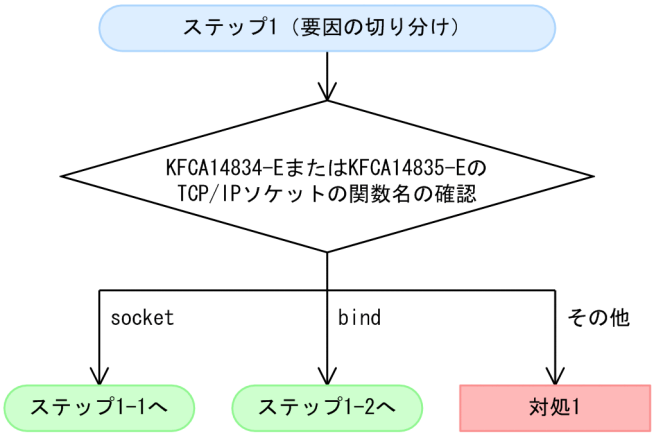
  自システムで netstat -a コマンドを実行し，取得した実行結果を残してください。

(4)　原因の調査と対処

KFCA14834-E または KFCA14835-E は，コネクションの確立準備に失敗した場合に出力されます。この障害は，主に自システム内の他のアプリケーション（自システム内のほかの TP1/NET/TCP/IP の通信プロセスを含みます）と自システムのポート番号が重複していることが原因で発生します。

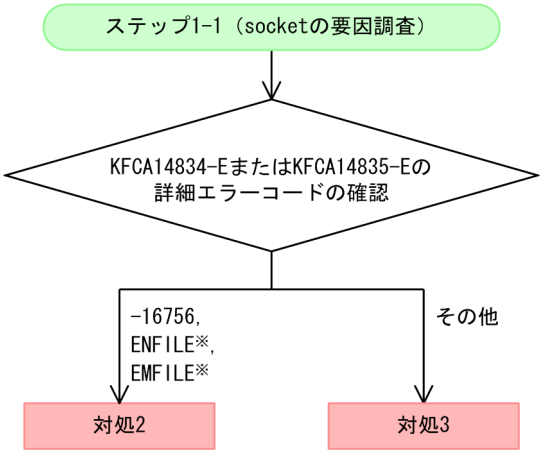
次に示すフローに従って原因の調査と対処をしてください。

図 10–14　KFCA14834-E または KFCA14835-E が出力された場合の調査手順（ステップ 1）



対処	内容
対処 1	KFCA14834-E または KFCA14835-E の TCP/IP ソケットの関数名と詳細エラーコードから，ご使用の OS のマニュアルで詳細エラーコードの意味を確認し，想定される要因に該当するかどうかを調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

図 10–15　KFCA14834-E または KFCA14835-E が出力された場合の調査手順（ステップ 1-1）



注※

OS が返した errno です。実際にメッセージに出力される値は，使用する OS によって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

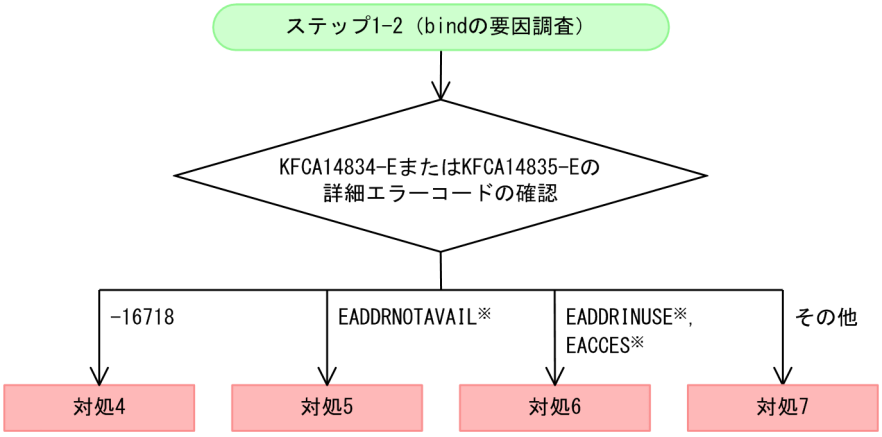
また，詳細エラーコードが ENFILE の場合はシステム全体で使用できるファイル記述子の上限を，EMFILE の場合は 1 プロセスで使用できるファイル記述子の上限を見直してください。

対処	内容
対処 2	MCF 通信プロセスで使用できるファイル記述子の指定値が不足していることが原因です。システムサービス共通情報定義の max_open_fds オペランドの指定値，および OS のカーネルパラメタ※の指定値を見直してください。
対処 3	ご使用の OS のマニュアルで socket 関数に対応する詳細エラーコードの意味を確認し，想定される要因に該当するかどうかを調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

注※

カーネルパラメタの指定値を変更する場合は，余裕を持った値を指定してください。

図 10–16　KFCA14834-E または KFCA14835-E が出力された場合の調査手順（ステップ 1-2）



注※

OS が返した errno です。実際にメッセージに出力される値は，使用する OS によって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

対処	内容
対処 4	コネクション定義の次に示すオプションおよびオペランドの指定値が，すべて一致するコネクションが複数存在することが原因です。 • 自システムの IP アドレス（mcftalccn -r ipaddr）またはホスト名（mcftalccn -r hostname） • 自システムのポート番号（mcftalccn -r portno） • 相手システムの IP アドレス（mcftalccn -o oipaddr）またはホスト名（mcftalccn -o ohostname） • 相手システムのポート番号（mcftalccn -o oportno）に指定した free 以外の値 上記のコネクション定義の指定値を見直してください。
対処 5	割り当てられない IP アドレスを指定したことが原因です。

対処	内容
対処 5	コネクション定義の自システムの IP アドレス（mcftalccn -r ipaddr），またはホスト名（mcftalccn -r hostname）の指定値を見直してください。 直前に KFCA14868-I が出力されている場合は，dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')で設定した自システムの IP アドレスの設定値を見直してください。
対処 6	他のプログラムが使用しているポート番号を指定したことが原因です。 コネクション定義の自システムのポート番号（mcftalccn -r portno）の指定値を見直してください。 直前に KFCA14868-I が出力されている場合は，dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')で設定した自システムのポート番号の設定値を見直してください。※
対処 7	ご使用の OS のマニュアルで，bind 関数に対応する詳細エラーコードの意味を確認し，想定される要因に該当するかどうか調査してください。 解決しない場合は，情報を取得してサポートセンタへ問い合わせてください。

注※

他のアプリケーションとの重複を避けるため，OS が任意に割り当てるポート番号（動的ポートまたは短命ポートと呼ばれるポート番号）を使用しないでください。

OS が任意に割り当てるポート番号の範囲は，OS の種別やバージョンによって異なります。詳細については，ご使用の OS のマニュアルまたは技術情報を参照してください。

10.2.7　KFCA14877-I メッセージが出力された場合

KFCA14877-I メッセージが出力された場合の調査手順，および対処について説明します。

(1)　現象

次のメッセージが出力されて，通信性能が劣化しました。

```
KFCA14877-I mmm 受信バッファの空きがないため，メッセージの受信を待ち合わせます。
コネクション名＝ aa....aa バッファグループ番号＝ bb....bb
```

（凡例）

mmm：MCF 通信プロセス識別子

aa....aa：コネクション名

bb....bb：受信バッファのバッファグループ番号

または，KFCA14877-I のメッセージが出力されたあと，さらに次のメッセージが出力されて，コネクションが解放されました。

```
KFCA14847-E mmm バッファ不足のためバッファ取得に失敗しました。
コネクション名＝ aa....aa バッファ種別＝ bb....bb 障害コード＝ cc....cc
```

（凡例）

mmm：MCF 通信プロセス識別子

aa....aa：コネクション名

bb....bb：バッファ種別

cc....cc：障害コード

(2)　現象発生時の確認事項

- 現象発生日時
- 現象発生直前の OpenTP1 環境に対する変更点
- 現象発生直前の接続先相手システムに対する変更点

(3)　取得情報

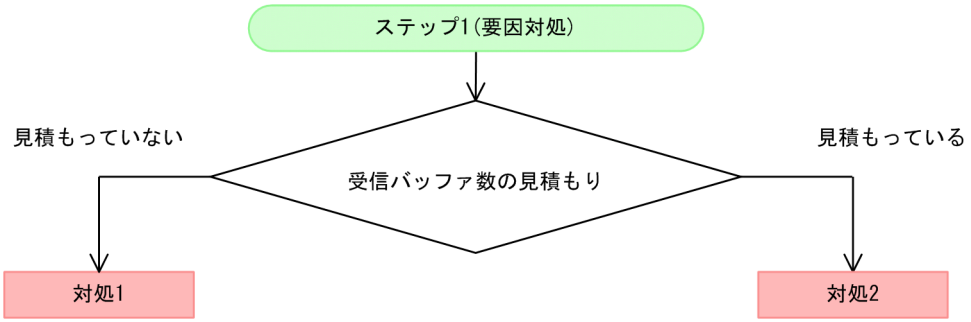
- 障害要因の調査に必要な情報
  「10.1.1　取得情報」を参照してください。

(4)　原因の調査と対処

KFCA14877-I および KFCA14847-E は，通信で使用する受信バッファが確保できなかった場合に出力されます。受信バッファの上限値は，コネクション定義（mcftalccn -g）の rcvbuf オペランドで指定されたバッファグループ番号に対応するバッファグループ定義（mcftbuf -g）の count オペランドの指定値と，extend オペランドの指定値の和です。この障害は，主に，受信バッファの上限値の見積もり不正による受信バッファ不足が原因で発生します。

次に示すフローに従って対処をしてください。

図 10–17　KFCA14877-I が出力された場合の調査手順



対処	内容
対処 1	「6.　システム定義」の「mcftalccn（コネクション定義の開始）」の見積もり計算式を参照して見積もってください。
対処 2	一度に大量のメッセージを受信していることが原因です。 次の点を見直してください。 • 単位時間あたりの受信メッセージ件数が妥当か，相手システムを見直してください。 • 受信バッファ数の安全値を見直してください。安全値を設定していない場合は，1.2 倍を目安に再見積もりをしてください。 • 受信バッファのバッファ長（バッファグループ定義（mcftbuf -g）の length オペランド）の指定値が適切であるかを見直してください。受信バッファのバッファ長

対処	内容
対処 2	が受信メッセージ長よりも非常に大きい場合，一時的に大量の受信バッファを必要とすることがあります。

付録

付録 A　バージョンアップ時の変更点

各バージョンでの変更点を次に示す分類ごとに示します。

- 関数，定義およびコマンドの追加・変更・削除
- 動作の変更
- 関数，定義およびコマンドのデフォルト値の変更

付録 A.1　07-50 での変更点

TP1/NET/TCP/IP 07-50 での関数，定義およびコマンドの追加・変更・削除はありません。

TP1/NET/TCP/IP 07-50 での動作の変更点を次に示します。

表 A–1　TP1/NET/TCP/IP 07-50 での動作の変更点

分類	内容
定義	コネクション定義の開始，論理端末定義，コネクション定義の終了の指定数を 1024 から 2048 に変更

TP1/NET/TCP/IP 07-50 での関数，定義およびコマンドのデフォルト値の変更を次の表に示します。

表 A–2　TP1/NET/TCP/IP 07-50 でのデフォルト値の変更

分類	内容
定義	コネクション定義の開始（mcftalccn） • -f オプションの releaselog オペランドのデフォルト値（1）を 2 に変更 • -f オプションの cnerrlog オペランドのデフォルト値（1）を 2 に変更

付録 A.2　07-04 での変更点

TP1/NET/TCP/IP 07-04 での関数，定義およびコマンドの追加・変更・削除を次の表に示します。

表 A–3　TP1/NET/TCP/IP 07-04 での関数，定義およびコマンドの追加・変更・削除

種別	分類	内容
追加	関数	dc_mcf_contend
		dc_mcf_reply
		dc_mcf_tdctcn • action の設定値に DCMCFFRC を追加
		dc_mcf_tempget

種別	分類	内容
追加	関数	dc_mcf_tempput
		CBLDCMCF('CONTEND△')
		CBLDCMCF('REPLY△△△')
		CBLDCMCF('TDCTCN△△') • データ名 D1 の設定値に 0，1 を追加
		CBLDCMCF('TEMPGET△')
		CBLDCMCF('TEMPPUT△')
		DISABLE － 継続問い合わせ応答の終了
		RECEIVE － メッセージの受信 • SYNCHRONOUS MODE 句の設定値に ASYNC を追加 • SYNCHRONOUS MODE 句のデータ名 6 の設定値に 0，空白を追加
		RECEIVE － 一時記憶データの受け取り
		SEND － メッセージの送信 • NEXT TRANSACTION 句 • WITH 句
		SEND － 一時記憶データの更新
	定義	コネクション定義の開始（mcftalccn） • -f オプションの cnrelease オペランド • -f オプションの releaselog オペランド • -f オプションの cnerrlog オペランド • -l オプション
		アプリケーション属性定義（mcfaalcap） • -n オプション
	コマンド	mcftcp コマンド • -r オプション
		mcftendct コマンド
変更		なし
削除		なし

TP1/NET/TCP/IP 07-04 での動作の変更点を次に示します。

表 A–4　TP1/NET/TCP/IP 07-04 での動作の変更点

分類	内容
その他	受信バッファ数不足を検出した場合，メッセージ KFCA14877-I を出力するように変更

TP1/NET/TCP/IP 07-04 での関数，定義およびコマンドのデフォルト値の変更はありません。

付録 A.3　07-03 での変更点

TP1/NET/TCP/IP 07-03 での関数，定義およびコマンドの追加・変更・削除を次の表に示します。

表 A–5　TP1/NET/TCP/IP 07-03 での関数，定義およびコマンドの追加・変更・削除

種別	分類	内容
追加	関数	送受信メッセージ廃棄通知イベント（MDELEVT）
	定義	論理端末定義（mcftalcle） • -d オプション
	コマンド	なし
変更		なし
削除		なし

TP1/NET/TCP/IP 07-03 での動作の変更点はありません。

TP1/NET/TCP/IP 07-03 での関数，定義およびコマンドのデフォルト値の変更はありません。

付録 A.4　07-02 での変更点

TP1/NET/TCP/IP 07-02 での関数，定義およびコマンドの追加・変更・削除を次の表に示します。

表 A–6　TP1/NET/TCP/IP 07-02 での関数，定義およびコマンドの追加・変更・削除

種別	分類	内容
追加	関数	dc_mcf_recvsync
		dc_mcf_tactcn
		dc_mcf_tactle
		dc_mcf_tdctcn
		dc_mcf_tdctle
		dc_mcf_tlscn
		dc_mcf_tlsle
		dc_mcf_tlsln
		dc_mcf_tofln
		dc_mcf_tonln
		CBLDCMCF('RECVSYNC')
		CBLDCMCF('TACTCN△△')

種別	分類	内容
追加	関数	CBLDCMCF('TACTLE△△')
		CBLDCMCF('TDCTCN△△')
		CBLDCMCF('TDCTLE△△')
		CBLDCMCF('TLSCN△△△')
		CBLDCMCF('TLSLE△△△')
		CBLDCMCF('TLSLN△△△')
		CBLDCMCF('TOFLN△△△')
		CBLDCMCF('TONLN△△△')
		入力セグメント判定 UOC • dctcp_uoc_sgck 構造体の rcv_buf_size
		受信メッセージ保留判定 UOC
		受信メッセージ保留通知イベント（RHLDEVT）
	定義	UAP 共通定義（mcfmuap） • -t オプションの recvtim オペランド
		コネクション定義の開始（mcftalccn） • -u オプションの msghold オペランド • -u オプションの holdlimit オペランド • -w オプション • -C オプション
	コマンド	mcftlsln コマンド • -t オプション
変更		なし
削除		なし

TP1/NET/TCP/IP 07-02 での動作の変更点を次に示します。

表 A–7　TP1/NET/TCP/IP 07-02 での動作の変更点

分類	内容
コマンド	mcftlscn コマンド • mcftalccn 定義コマンドの-C オプションの lscnfmt オペランドに 2 を指定した場合の，次のメッセージ数を追加 ・ -d オプションの出力内容に現在保留している受信メッセージ数 ・ OpenTP1 システム開始以降に保留している受信メッセージの最大数 • 詳細ステータスの出力内容に，同期型受信のデータ受信待ち状態を追加
その他	受信バッファオーバフローを検出した場合，メッセージ KFCA14865-I を出力するように変更

TP1/NET/TCP/IP 07-02 での関数，定義およびコマンドのデフォルト値の変更はありません。

付録 A.5　07-01 での変更点

TP1/NET/TCP/IP 07-01 での関数，定義およびコマンドの追加・変更・削除を次の表に示します。

表 A–8　TP1/NET/TCP/IP 07-01 での関数，定義およびコマンドの追加・変更・削除

種別	分類	内容
追加	関数	なし
	定義	コネクション定義の開始（mcftalccn） • -h オプションの listen オペランド • -N オプションの modelname オペランド
		論理端末定義（mcftalcle） • -N オプションの modelname オペランド
	コマンド	mcftlsln コマンド
		mcftofln コマンド
		mcftonln コマンド
変更		なし
削除		なし

TP1/NET/TCP/IP 07-01 での動作の変更点を次に示します。

表 A–9　TP1/NET/TCP/IP 07-01 での動作の変更点

分類	内容
その他	サーバ型コネクションのアドレス割り当てに失敗した場合，MCF 通信プロセスの起動を中断しないように変更。

TP1/NET/TCP/IP 07-01 での関数，定義およびコマンドのデフォルト値の変更はありません。

付録 A.6　07-00 での変更点

TP1/NET/TCP/IP 07-00 での関数，定義およびコマンドの追加・変更・削除を次の表に示します。

表 A–10　TP1/NET/TCP/IP 07-00 での関数，定義およびコマンドの追加・変更・削除

種別	分類	内容
追加	関数	送信完了イベント（SCMPEVT）

種別	分類	内容
追加	定義	なし
	コマンド	なし
変更	関数	C 言語の関数の 64 ビットアーキテクチャでのインタフェースを，32 ビットアーキテクチャでのインタフェースと統一。 詳細については，「付録 C　インタフェースの変更一覧（バージョン 6 以前から移行する場合）」を参照してください。
削除		なし

TP1/NET/TCP/IP 07-00 での動作の変更点を次に示します。

表 A–11　TP1/NET/TCP/IP 07-00 での動作の変更点

分類	内容
その他	コネクション確立要求を格納するキューの長さを，固定値から定義したコネクション数に応じて，TP1/NET/TCP/IP が自動的に調整するように変更。

TP1/NET/TCP/IP 07-00 での関数，定義およびコマンドのデフォルト値の変更はありません。

付録 B　旧製品からの移行に関する注意事項

ここでは，バージョン 6 からバージョン 7 へ移行する場合，およびバージョン 5 以前からバージョン 7 へ移行する場合の注意事項について説明します。

付録 B.1　バージョン 6 からの移行

バージョン 6 からバージョン 7 へ移行する場合，次の点に注意してください。

(1)　ソースの互換性

バージョン 6 からバージョン 7 へ移行する場合の各種ソースファイルの互換性について説明します。

バージョン 6 からバージョン 7 へ移行する場合，バージョン 6 で使用していたソースファイルをそのまま使用できないことがあります。ソースファイルの互換性は，次の表に示すとおりです。

表 B-1　バージョン 6 で使用していたソースファイルの互換性

ソースファイルの種類	ソースファイルを作成した言語		互換性
UAP	C 言語	32 ビット	32 ビットアーキテクチャのバージョン 7 を使用する場合は，ソースファイルを変更しないで使用できます。 それ以外の場合は，ソースファイルを変更する必要があります。※
		64 ビット	64 ビットアーキテクチャのバージョン 7 を使用する場合は，ソースファイルを変更しないで使用できます。 それ以外の場合は，ソースファイルを変更する必要があります。※
	COBOL 言語		ソースファイルを変更しないで使用できます。
UOC	C 言語	32 ビット	32 ビットアーキテクチャのバージョン 7 を使用する場合は，ソースファイルを変更しないで使用できます。 それ以外の場合は，ソースファイルを変更する必要があります。※
		64 ビット	64 ビットアーキテクチャのバージョン 7 を使用する場合は，ソースファイルを変更しないで使用できます。 それ以外の場合は，ソースファイルを変更する必要があります。※
MCF 通信構成定義（プロトコル固有の定義）	－		ソースファイルを変更しないで使用できます。

（凡例）

－：該当する内容がないことを表します。

注※

バージョン 7 では，メッセージ送受信インタフェース，UOC，および MCF イベントインタフェースのそれぞれの引数ならびにパラメタの型が変更されています。そのため，バージョン 6 の UAP および UOC のソースファイルを見直す必要があります。

なお，この変更による UAP や UOC の処理への影響はありません。

詳細については，「付録 C　インタフェースの変更一覧（バージョン 6 以前から移行する場合）」を参照してください。

(2)　コネクション確立要求を格納するキューの長さ

バージョン 6 ではコネクション数に関係なく固定としていましたが，バージョン 7 では定義したコネクション数に応じて TP1/NET/TCP/IP が自動的に調整するように変更しました。

付録 B.2　バージョン 5 以前からの移行

バージョン 5 以前からバージョン 7 へ移行する場合，次の点に注意してください。

(1)　ソースの互換性

バージョン 5 以前からバージョン 7 へ移行する場合の各種ソースファイルの互換性について説明します。

バージョン 5 以前からバージョン 7 へ移行する場合，バージョン 5 以前で使用していたソースファイルをそのまま使用できないことがあります。ソースファイルの互換性は，次の表に示すとおりです。

表 B-2　バージョン 5 以前で使用していたソースファイルの互換性

ソースファイルの種類	ソースファイルを作成した言語	互換性
UAP	C 言語	32 ビットアーキテクチャのバージョン 7 を使用する場合は，ソースファイルを変更しないで使用できます。 それ以外の場合は，ソースファイルを変更する必要があります。※
	COBOL 言語	ソースファイルを変更しないで使用できます。
UOC	C 言語	32 ビットアーキテクチャのバージョン 7 を使用する場合は，ソースファイルを変更しないで使用できます。 それ以外の場合は，ソースファイルを変更する必要があります。※
MCF 通信構成定義（プロトコル固有の定義）	－	ソースファイルを変更しないで使用できます。

（凡例）

－：該当する内容がないことを表します。

注※

バージョン 7 では，メッセージ送受信インタフェース，UOC，および MCF イベントインタフェースのそれぞれの引数ならびにパラメタの型が変更されています。そのため，UAP および UOC のソースファイルを見直す必要があります。

なお，この変更による UAP や UOC の処理への影響はありません。

詳細については，「付録 C　インタフェースの変更一覧（バージョン 6 以前から移行する場合）」を参照してください。

(2) サーバ型コネクションの自 IP アドレス

バージョン 5 以前では，サーバ型コネクションに指定した自システムの IP アドレスおよびホスト名を無効としていましたが，バージョン 7 では，コネクション定義に指定した自システムの IP アドレスまたはホスト名に対応する IP アドレスを開始時にバインドします。

指定した IP アドレスまたはホスト名に対応する IP アドレスへのバインドに失敗した場合，-b オプションの bretry，bretrycnt，bretryint オペランドの指定値に従ってバインドを再試行します。

ホスト名に対応する IP アドレスの取得に失敗した場合，開始処理を中断しますのでご注意ください。

(3) メッセージ送信の再試行

バージョン 5 以前に存在したコネクション定義のメッセージ送信の再試行（mcftalccn -b dretry，dretryint，dretrycnt）は廃止し，定義しても無効となります。

バージョン 5 以前で行っていたメッセージ送信の再試行をバージョン 7 でも引き続き行い，再試行間隔（mcftalccn -b dretryint）および再試行回数（mcftalccn -b dretrycnt）にデフォルト値以外の値を設定していた場合，メッセージ送信完了監視時間（mcftalccn -b sndcmptim）に次の計算式で求めた値を設定してください。

```
メッセージ送信完了監視時間（sndcmptim）（秒）
  ＝再試行間隔（ミリ秒）× 再試行回数／1000（端数切り上げ）
    (dretryint)           (dretrycnt)
```

（例）

再試行間隔に 250（ミリ秒），再試行回数に 15（回）を指定していた場合の，メッセージ送信完了監視時間を求めるときの計算式を次に示します。

```
250 × 15 / 1000 ＝ 3.75
                ≒ 4
```

この場合は，メッセージ送信完了時間に 4（秒）を指定します。

(4) MCF トレースで取得する送受信メッセージのサイズ

バージョン 5 以前では，TP1/NET/TCP/IP が送受信したメッセージの全内容を MCF トレースに取得していましたが，バージョン 7 では，送受信したメッセージの先頭から最大 128 バイトを MCF トレースに取得します。

128 バイトを超える送受信メッセージを MCF トレースに取得する場合は，トレース環境定義の MCF トレースに取得する送受信メッセージのサイズ（mcfttrc -t msgsize）に，送受信するメッセージ長以上の値を指定してください。

(5) 未確立コネクションの論理端末に対する運用コマンドの動作

バージョン 7 では，未確立コネクションの論理端末に対する運用コマンド（mcftactle と mcftdctle）の動作が，バージョン 5 以前と異なります。未確立コネクションの論理端末に対する運用コマンドの動作を次に示します。

表 B-3 未確立コネクションの論理端末に対する運用コマンドの動作

運用コマンド	バージョン	論理端末の状態	
		閉塞	閉塞解除
mcftactle	バージョン 5 以前	論理端末の閉塞解除をします。	KFCA14821-W を出力し，運用コマンドを受け付けません。
	バージョン 7	KFCA14820-W を出力し，運用コマンドを受け付けません。	該当する状態※にはなりません。
mcftdctle	バージョン 5 以前	KFCA14822-W を出力し，運用コマンドを受け付けません。	論理端末を閉塞します。
	バージョン 7	KFCA14820-W を出力し，運用コマンドを受け付けません。	該当する状態※にはなりません。

注※

コネクションが未確立で，かつ論理端末が閉塞解除している状態。

(6) クライアント型コネクションの確立処理

バージョン 7 では，クライアント型コネクションの確立処理が，バージョン 5 以前と異なります。

バージョン 7 では，クライアント型コネクションの確立処理を変更し，相手システムからの応答を待ち合わせる最大時間として，コネクション確立時の監視時間をコネクション定義（mcftalccn -b）の concmptim オペランドで指定できます。

クライアント型コネクションの確立処理を変更したことによって，次の処理に必要な時間がバージョン 5 以前と異なります。

- TP1/NET/TCP/IP がコネクション確立を検出するまでの時間
- 相手システムまたはネットワークの障害によって確立処理がリトライオーバするまでの時間

OS が相手システムとコネクションを確立したあと，TP1/NET/TCP/IP がコネクションの確立を検出するために，バージョン 5 以前では最大で，コネクション定義（mcftalccn -b）の bretryint オペランドで指定したコネクション確立障害時の確立再試行間隔だけ時間を必要としました。一方，バージョン 7 ではコネクションの確立を即時検出できます。

バージョン 7 の確立処理のリトライオーバまでの所要時間の計算式を次に示します。

確立処理のリトライオーバまでの最大所要時間（秒）
＝（コネクション確立障害時の確立再試行間隔 ＋ 確立障害検出時間※）

```
    (mcftalccn -b bretryint)
×コネクション確立障害時の確立再試行回数
    (mcftalccn -b bretrycnt)
```

注※

コネクション確立時の監視時間と OS が確立障害を検出する時間のどちらか小さい方の値が設定されます。

バージョン 5 以前での，クライアント型コネクションの確立処理の流れは，「付録 B.3　バージョン 5 以前でのソケット関数の処理の流れ」を参照してください。また，バージョン 7 での，クライアント型コネクションの確立手順の詳細は，「付録 G　ソケット関数の処理の流れ」を参照してください。

(7)　コネクション当たりの受信バッファ数

バージョン 5 以前では，1 コネクション当たりに必要な受信バッファは 3 以上でしたが，バージョン 7 では次の計算式で求めた値と 2 のどちらか大きい方の値以上が必要となります。

詳細については，「6.　システム定義」の「mcftalccn（コネクション定義の開始）」の注意事項を参照してください。

```
1コネクション当たりの受信バッファ数(小数点以下を切り上げ)
  ＝受信バッファのバッファサイズ（バイト）／受信するメッセージの最小長
                                        (mcftbuf -b length)
```

(8)　システムサービス共通情報定義の max_open_fds オペランド

バージョン 5 以前ではプロトコル制御で使用するファイル記述子数は，コネクションの総数とサーバ型コネクションで使用するポート数でしたが，バージョン 7 ではコネクションの総数，サーバ型コネクションの自システムのホストの IP アドレス（ホスト名），およびポート番号を組み合わせた数です。

max_open_fds オペランドについては，「6.　システム定義」の「システムサービス共通情報定義」を参照してください。

(9)　Windows 版でのシステムサービス共通情報定義の max_socket_descriptors オペランドと max_open_fds オペランド

Windows 版のバージョン 5 ではシステムサービス共通情報定義の max_socket_descriptors にプロトコル制御で使用するファイル記述子数（コネクション数×2）を追加する必要がありましたが，バージョン 7 では UNIX 版と同様に max_open_fds オペランドにプロトコル制御で使用するファイル記述子を追加してください。

(10)　コネクション確立要求を格納するキューの長さ

バージョン 5 以前ではコネクション数にかかわらず固定としていましたが，バージョン 7 では定義したコネクション数に応じて TP1/NET/TCP/IP が自動的に調整するように変更しました。

(11)　OpenTP1 終了中の相手システムからのコネクション確立

バージョン 5 以前では OpenTP1 終了中においても相手システムからのコネクション確立要求を受け付けていましたが，バージョン 7 では相手システムからのコネクション確立要求を受け付けないように変更しました。

(12)　受信バッファ数不足発生時のメッセージログ

バージョン 5 以前では KFCA10618-E を出力していましたが，バージョン 7 では KFCA14847-E を出力するように変更しました。

(13)　追加機能

バージョン 5 以前と比較して，バージョン 7 では「付録 A　バージョンアップ時の変更点」に記載した追加機能のほかに次の機能を追加しています。

- 相手アドレスチェックの抑止

  サーバ型コネクションにおいて，相手システムのアドレス情報のチェックを抑止し，任意の相手システムからのコネクション確立要求を受け付ける機能です。詳細は「2.1.7　相手アドレスチェックの抑止（サーバ型コネクション）」を参照してください。

- コネクションリプレース

  サーバ型コネクションにおいて，割り当て可能な未確立コネクションがない時に相手システムからコネクション確立要求を受け付けた場合，確立状態にある既存のコネクションを解放し，新しい確立要求を受け付ける機能です。詳細は「2.1.10　コネクションリプレース（サーバ型コネクション）」を参照してください。

- 無通信状態監視

  一定時間以上相手システムと無通信状態が続いていないかを監視する機能です。詳細は「2.1.15　無通信状態監視」を参照してください。

- 同期型メッセージの送信

  sendsync 関数または CBLDCMCF('SENDSYNC')にて同期型メッセージを送信できる機能です。詳細は「2.3.2(1)　同期型メッセージの送信」を参照してください。

- メッセージ送達の確認

  DCCMII/TCP および DCCM3/TCP のメッセージ送達確認プロトコルに従ったメッセージ送達確認を行うか，または，受信メッセージ判定 UOC を使用した任意の相手システムとのメッセージ送達確認を行う機能です。詳細は「2.3.10(2)　TP1/NET/TCP/IP が提供する送達確認機能を使用する場合」を参照してください。

- 受信バッファの空き待ち

  受信バッファの空きが不足した場合，受信バッファの空きを待ち合わせる機能です。詳細については，「2.3.7(3)　受信バッファの空き待ち」を参照してください。

付録 B.3　バージョン 5 以前でのソケット関数の処理の流れ

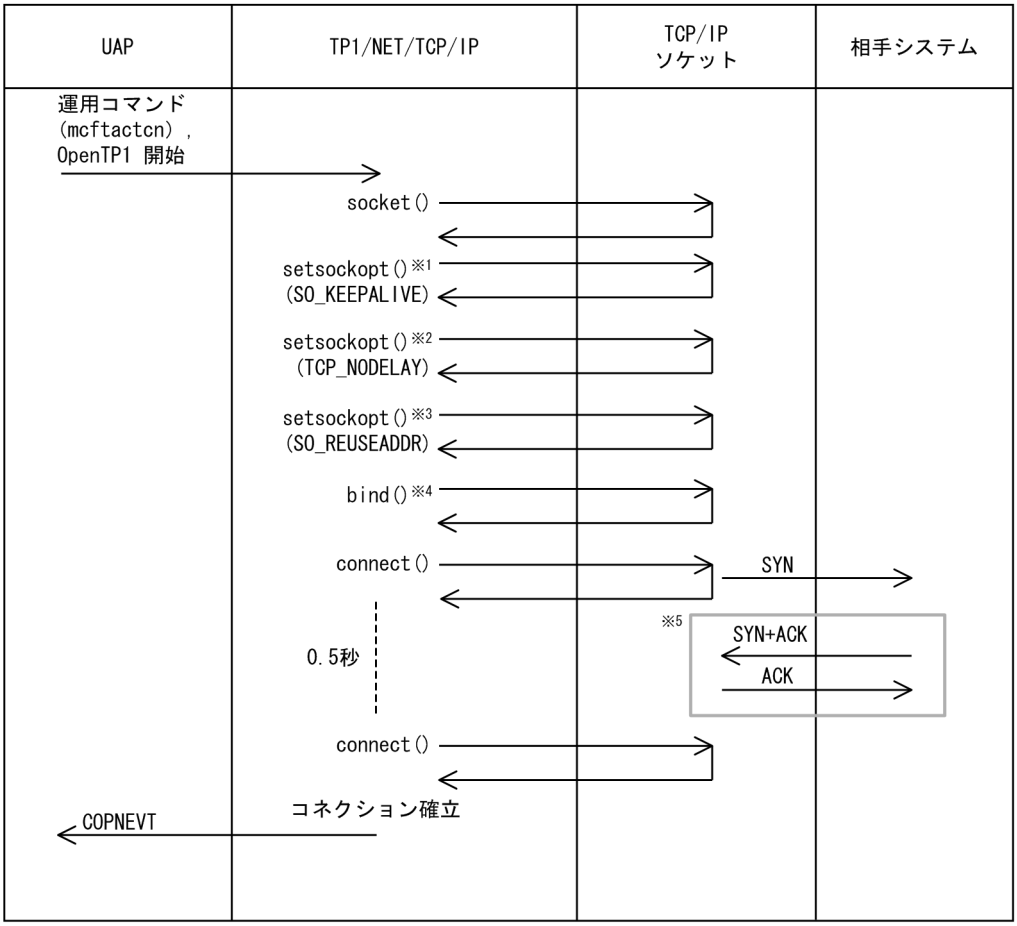
ここでは，バージョン 5 以前の TP1/NET/TCP/IP が，次の場合に発行するソケット関数の処理の流れを次の図に示します。

- 図 B-1　クライアント型コネクションの確立（バージョン 5 以前）
- 図 B-2　クライアント型コネクションの確立障害（相手システムのサービス未起動の場合）（バージョン 5 以前）
- 図 B-3　クライアント型コネクションの確立障害（相手システム未起動またはネットワーク障害）（バージョン 5 以前）

TCP/IP ソケットと相手システム間のパケット送受信のタイミングは TP1/NET/TCP/IP が動作する OS の実装に依存します。実装によっては，実際のパケット送受信のタイミングが，図に示したパケット送受信のタイミングと異なる場合があります。

なお，バージョン 7 でのソケット関数の処理の流れについては，「付録 G　ソケット関数の処理の流れ」を参照してください。

図 B-1　クライアント型コネクションの確立（バージョン 5 以前）



（凡例）

ACK：確認応答フラグ

SYN：コネクション確立要求フラグ

注※1

キープアライブを使用する場合（コネクション定義（mcftalccn -k）の keepalive オペランドに yes を指定），発行します。

注※2

TCP_NODELAY を使用する場合（コネクション定義（mcftalccn -k）の nodelay オペランドに yes を指定），発行します。

注※3

自システムのポート番号を指定した場合，発行します。

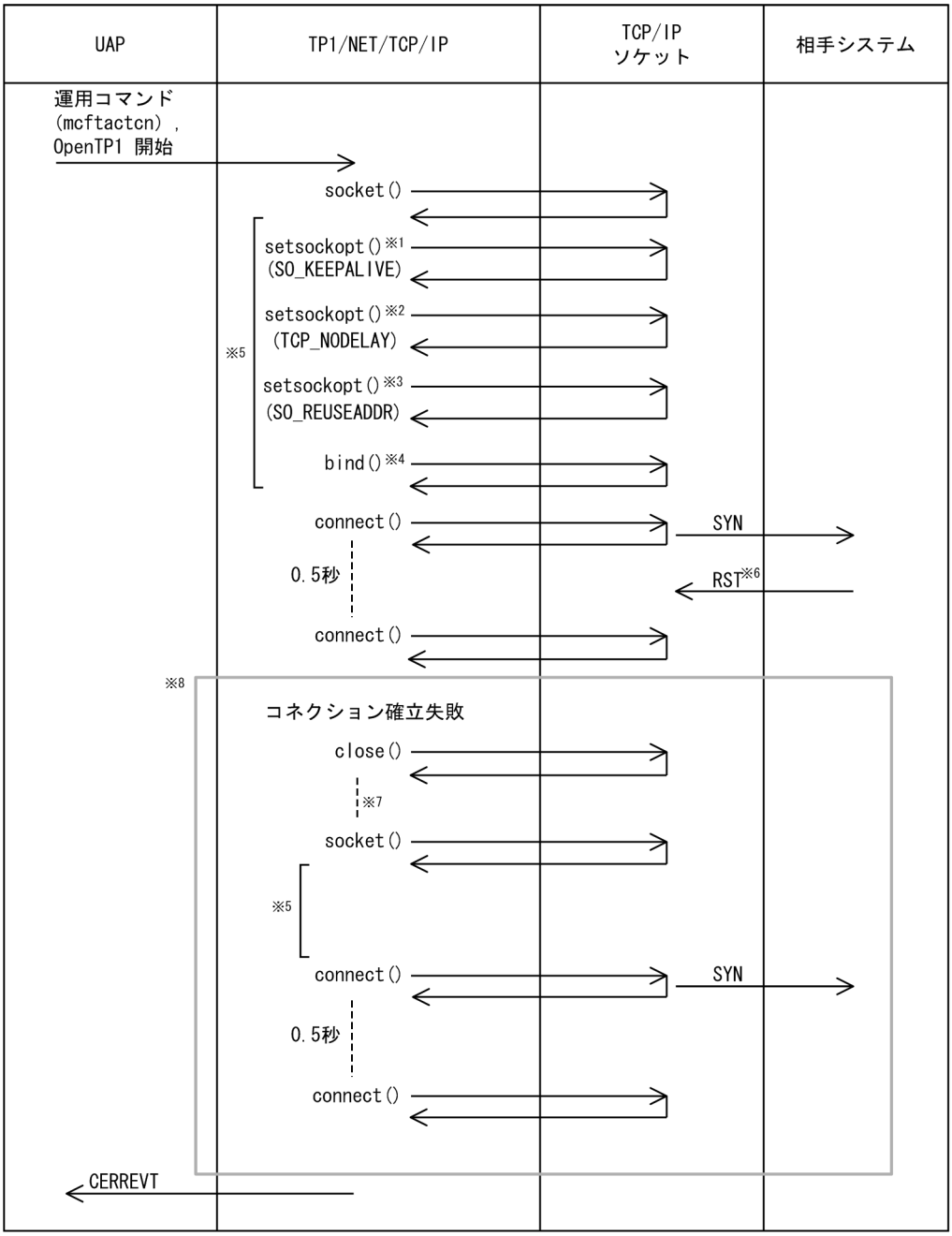
注※4

自システムのホスト名，IP アドレスまたはポート番号を指定した場合，発行します。

注※5

パケットの送受信は，TP1/NET/TCP/IP と非同期に行われます。バージョン 5 以前の TP1/NET/TCP/IP は，相手システムからの SYN および ACK を待ち合わせません。

図 B-2　クライアント型コネクションの確立障害（相手システムのサービス未起動）（バージョン 5 以前）



（凡例）

SYN：コネクション確立要求フラグ

RST：コネクション強制切断フラグ

注※1

キープアライブを使用する場合（コネクション定義（mcftalccn -k）の keepalive オペランドに yes を指定），発行します。

注※2

TCP_NODELAY を使用する場合（コネクション定義（mcftalccn -k）の nodelay オペランドに yes を指定），発行します。

注※3

自システムのポート番号を指定した場合，発行します。

注※4

自システムのホスト名，IP アドレスまたはポート番号を指定した場合，発行します。

注※5

同じ処理を行います。

注※6

パケットの送受信は，TP1/NET/TCP/IP と非同期に行われます。バージョン 5 の TP1/NET/TCP/IP は，相手システムからの RST を待ち合わせません。

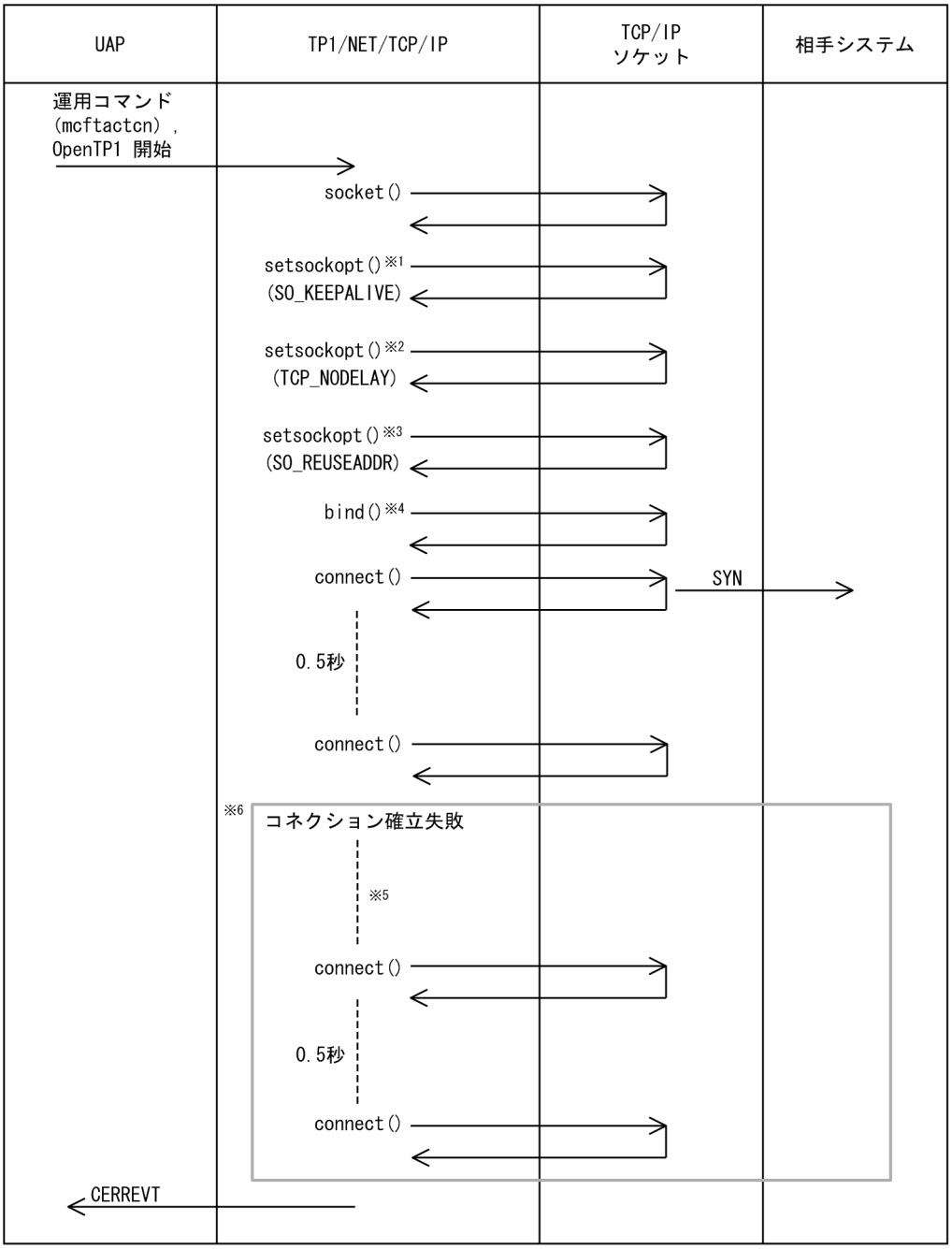
注※7

コネクション確立障害時の確立再試行間隔（コネクション定義（mcftalccn -b）の bretryint オペランドの指定値）だけ待ち合わせます。

注※8

最大でコネクション確立障害時の確立再試行回数（コネクション定義（mcftalccn -b）の bretrycnt オペランドの指定値）だけ繰り返します。

図 B-3　クライアント型コネクションの確立障害（相手システム未起動またはネットワーク障害）（バージョン 5 以前）



（凡例）

SYN：コネクション確立要求フラグ

注※1

キープアライブを使用する場合（コネクション定義（mcftalccn -k）の keepalive オペランドに yes を指定），発行します。

注※2

TCP_NODELAY を使用する場合（コネクション定義（mcftalccn -k）の nodelay オペランドに yes を指定），発行します。

注※3

自システムのポート番号を指定した場合，発行します。

注※4

自システムのホスト名，IP アドレスまたはポート番号を指定した場合，発行します。

注※5

コネクション確立障害時の確立再試行間隔（コネクション定義（mcftalccn -b）の bretryint オペランドの指定値）だけ待ち合わせます。

注※6

最大でコネクション確立障害時の確立再試行回数（コネクション定義（mcftalccn -b）の bretrycnt オペランドの指定値）だけ繰り返します。

付録 C　インタフェースの変更一覧（バージョン 6 以前から移行する場合）

TP1/NET/TCP/IP のバージョン 6 以前から移行するときに，32 ビットアーキテクチャ以外の場合は，C 言語のソースファイルを見直す必要があります。ここでは，バージョン 6 以前のインタフェースの変更一覧を示します。

ここで説明するインタフェースを次に示します。

表 C-1　インタフェースの変更一覧

変更されたインタフェース		バージョン 7 のマニュアルの該当個所
メッセージ送受信インタフェース	dc_mcf_receive	3.　dc_mcf_receive － メッセージの受信（C 言語）
	dc_mcf_resend	3.　dc_mcf_resend － メッセージの再送（C 言語）
	dc_mcf_send	3.　dc_mcf_send － 一方送信メッセージの送信（C 言語）
	dc_mcf_sendrecv	3.　dc_mcf_sendrecv － 同期型メッセージの送受信（C 言語）
	dc_mcf_sendsync	3.　dc_mcf_sendsync － 同期型メッセージの送信（C 言語）
ユーザオウンコーディング	入力セグメント判定 UOC	5.1.2　入力セグメント判定 UOC インタフェース
	入力メッセージ編集 UOC	5.1.4　入力メッセージ編集 UOC インタフェース
	出力メッセージ編集 UOC	5.1.6　出力メッセージ編集 UOC インタフェース
	送信メッセージの通番編集 UOC	5.1.8　送信メッセージの通番編集 UOC インタフェース
	コネクション確立 UOC	5.1.10　コネクション確立 UOC インタフェース
	受信メッセージ判定 UOC	5.1.12　受信メッセージ判定 UOC インタフェース
MCF イベントインタフェース		5.2.3　MCF イベント情報の形式（C 言語）
MCF メイン関数のコーディング概要		8.2　MCF メイン関数の作成
ユーザアプリケーションプログラムの作成例		付録 I.1　コーディング例 (1)　C 言語（K&R 版）

以降，バージョン 6 以前のインタフェースと，バージョン 7 以前のインタフェースの変更一覧を示します。変更箇所には，下線を付与しています。

付録 C.1　メッセージ送受信インタフェース

ここでは，メッセージ送受信インタフェースの変更一覧を示します。

(1)　dc_mcf_receive － メッセージの受信

(a)　ANSI C，C++の形式

バージョン 6 以前

＜ 32 ビットアーキテクチャの場合＞

```
#include <dcmcf.h>
  int  dc_mcf_receive(long action,
                      long commform,
                      char *termnam,
                      char *resv01,
                      char *recvdata,
                      long *rdataleng,
                      long inbufleng,
                      long *time)
```

＜ 64 ビットアーキテクチャの場合＞

```
#include <dcmcf.h>
  int  dc_mcf_receive(int action,
                      int commform,
                      char *termnam,
                      char *resv01,
                      char *recvdata,
                      int *rdataleng,
                      int inbufleng,
                      int *time)
```

バージョン 7

```
#include <dcmcf.h>
  int  dc_mcf_receive(DCLONG action,
                      DCLONG commform,
                      char *termnam,
                      char *resv01,
                      char *recvdata,
                      DCLONG *rdataleng,
                      DCLONG inbufleng,
                      DCLONG *time)
```

(b)　K&R 版 C の形式

バージョン 6 以前

＜ 32 ビットアーキテクチャの場合＞

```
#include <dcmcf.h>
  int  dc_mcf_receive(action,
                      commform,
                      termnam,
                      resv01,
                      recvdata,
                      rdataleng,
                      inbufleng,
                      time)
  long      action;
  long      commform;
  char      *termnam;
  char      *resv01;
  char      *recvdata;
  long      *rdataleng;
  long      inbufleng;
  long      *time;
```

＜ 64 ビットアーキテクチャの場合＞

```
  #include <dcmcf.h>
  int  dc_mcf_receive(action,
```

バージョン 7

```
#include <dcmcf.h>
  int  dc_mcf_receive(action,
                      commform,
                      termnam,
                      resv01,
                      recvdata,
                      rdataleng,
                      inbufleng,
                      time)
  DCLONG    action;
  DCLONG    commform;
  char      *termnam;
  char      *resv01;
  char      *recvdata;
  DCLONG    *rdataleng;
  DCLONG    inbufleng;
  DCLONG    *time;
```

バージョン 6 以前	バージョン 7
commform, termnam, resv01, recvdata, rdataleng, inbufleng, time) int action; int commform; char *termnam; char *resv01; char *recvdata; int *rdataleng; int inbufleng; int *time;	#include <dcmcf.h> int dc_mcf_receive(action, commform, termnam, resv01, recvdata, rdataleng, inbufleng, time) DCLONG action; DCLONG commform; char *termnam; char *resv01; char *recvdata; DCLONG *rdataleng; DCLONG inbufleng; DCLONG *time;

(2) dc_mcf_resend － メッセージの再送

(a) ANSI C，C++の形式

バージョン 6 以前	バージョン 7
＜32 ビットアーキテクチャの場合＞ #include <dcmcf.h> int dc_mcf_resend(long action, long commform, char *rtermnam, char *resv01, long oseqid, long orgseq, char *otermnam, char *resv02, char *resv03, char *resv04, long opcd)	#include <dcmcf.h> int dc_mcf_resend(DCLONG action, DCLONG commform, char *rtermnam, char *resv01, DCLONG oseqid, DCLONG orgseq, char *otermnam, char *resv02, char *resv03, char *resv04, DCLONG opcd)
＜64 ビットアーキテクチャの場合＞ #include <dcmcf.h> int dc_mcf_resend(int action, int commform, char *rtermnam, char *resv01, int oseqid, int orgseq, char *otermnam, char *resv02, char *resv03, char *resv04, int opcd)	

(b) K&R 版 C の形式

バージョン 6 以前	バージョン 7

＜32 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`  int  dc_mcf_resend(action,`
`                    commform,`
`                    rtermnam,`
`                    resv01,`
`                    oseqid,`
`                    orgseq,`
`                    otermnam,`
`                    resv02,`
`                    resv03,`
`                    resv04,`
`                    opcd)`
`  long      action;`
`  long      commform;`
`  char      *rtermnam;`
`  char      *resv01;`
`  long      oseqid;`
`  long      orgseq;`
`  char      *otermnam;`
`  char      *resv02;`
`  char      *resv03;`
`  char      *resv04;`
`  long      opcd;`
	`#include <dcmcf.h>`
	`  int  dc_mcf_resend(action,`
	`                    commform,`
	`                    rtermnam,`
	`                    resv01,`
	`                    oseqid,`
	`                    orgseq,`
	`                    otermnam,`
	`                    resv02,`
	`                    resv03,`
	`                    resv04,`
	`                    opcd)`
	`  DCLONG    action;`
	`  DCLONG    commform;`
	`  char      *rtermnam;`
	`  char      *resv01;`
	`  DCLONG    oseqid;`
	`  DCLONG    orgseq;`
	`  char      *otermnam;`
	`  char      *resv02;`
	`  char      *resv03;`
	`  char      *resv04;`
	`  DCLONG    opcd;`

＜64 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`  int  dc_mcf_resend(action,`
`                    commform,`
`                    rtermnam,`
`                    resv01,`
`                    oseqid,`
`                    orgseq,`
`                    otermnam,`
`                    resv02,`
`                    resv03,`
`                    resv04,`
`                    opcd)`
`  int       action;`
`  int       commform;`
`  char      *rtermnam;`
`  char      *resv01;`
`  int       oseqid;`
`  int       orgseq;`
`  char      *otermnam;`
`  char      *resv02;`
`  char      *resv03;`
`  char      *resv04;`
`  int       opcd;`	

(3) dc_mcf_send － 一方送信メッセージの送信

(a) ANSI C，C++の形式

バージョン 6 以前	バージョン 7
＜32 ビットアーキテクチャの場合＞	`#include <dcmcf.h>` `  int  dc_mcf_send(DCLONG action,`

バージョン 6 以前	バージョン 7

#include <dcmcf.h>
  int dc_mcf_send(long action,
  long commform,
  char *termnam,
  char *resv01,
  char *senddata,
  long sdataleng,
  char *resv02,
  long opcd)
	DCLONG commform,
	  char *termnam,
	  char *resv01,
	  char *senddata,
	  DCLONG sdataleng,
	  char *resv02,
	  DCLONG opcd)

＜ 64 ビットアーキテクチャの場合＞
#include <dcmcf.h>
  int dc_mcf_send(int action,
  int commform,
  char *termnam,
  char *resv01,
  char *senddata,
  int sdataleng,
  char *resv02,
  int opcd)	

(b)　K&R 版 C の形式

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
#include <dcmcf.h>
  int dc_mcf_send(action,
  commform,
  termnam,
  resv01,
  senddata,
  sdataleng,
  resv02,
  opcd)
  long action;
  long commform;
  char *termnam;
  char *resv01;
  char *senddata;
  long sdataleng;
  char *resv02;
  long opcd;
	#include <dcmcf.h>
	  int dc_mcf_send(action,
	  commform,
	  termnam,
	  resv01,
	  senddata,
	  sdataleng,
	  resv02,
	  opcd)
	  DCLONG action;
	  DCLONG commform;
	  char *termnam;
	  char *resv01;
	  char *senddata;
	  DCLONG sdataleng;
	  char *resv02;
	  DCLONG opcd;

＜ 64 ビットアーキテクチャの場合＞
#include <dcmcf.h>
  int dc_mcf_send(action,
  commform,
  termnam,
  resv01,
  senddata,
  sdataleng,
  resv02,
  opcd)
  int action;
  int commform;
  char *termnam;
  char *resv01;
  char *senddata;
  int sdataleng;	

バージョン 6 以前	バージョン 7
char *resv02; int opcd;	#include <dcmcf.h> int dc_mcf_send(action, commform, termnam, resv01, senddata, sdataleng, resv02, opcd) DCLONG action; DCLONG commform; char *termnam; char *resv01; char *senddata; DCLONG sdataleng; char *resv02; DCLONG opcd;

(4) dc_mcf_sendrecv － 同期型メッセージの送受信

(a) ANSI C，C++の形式

バージョン 6 以前	バージョン 7
＜ 32 ビットアーキテクチャの場合＞ #include <dcmcf.h> int dc_mcf_sendrecv(long action, long commform, char *termnam, char *resv01, char *senddata, long sdataleng, char *recvdata, long *rdataleng, long inbufleng, long *time, long watchtime)	#include <dcmcf.h> int dc_mcf_sendrecv(DCLONG action, DCLONG commform, char *termnam, char *resv01, char *senddata, DCLONG sdataleng, char *recvdata, DCLONG *rdataleng, DCLONG inbufleng, DCLONG *time, DCLONG watchtime)
＜ 64 ビットアーキテクチャの場合＞ #include <dcmcf.h> int dc_mcf_sendrecv(int action, int commform, char *termnam, char *resv01, char *senddata, int sdataleng, char *recvdata, int *rdataleng, int inbufleng, int *time, int watchtime)	

(b) K&R 版 C の形式

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`  int  dc_mcf_sendrecv(action,`
`                      commform,`
`                      termnam,`
`                      resv01,`
`                      senddata,`
`                      sdataleng,`
`                      recvdata,`
`                      rdataleng,`
`                      inbufleng,`
`                      time,`
`                      watchtime)`
`  long      action;`
`  long      commform;`
`  char      *termnam;`
`  char      *resv01;`
`  char      *senddata;`
`  long      sdataleng;`
`  char      *recvdata;`
`  long      *rdataleng;`
`  long      inbufleng;`
`  long      *time;`
`  long      watchtime;`
	`#include <dcmcf.h>`
	`  int  dc_mcf_sendrecv(action,`
	`                      commform,`
	`                      termnam,`
	`                      resv01,`
	`                      senddata,`
	`                      sdataleng,`
	`                      recvdata,`
	`                      rdataleng,`
	`                      inbufleng,`
	`                      time,`
	`                      watchtime)`
	`  DCLONG    action;`
	`  DCLONG    commform;`
	`  char      *termnam;`
	`  char      *resv01;`
	`  char      *senddata;`
	`  DCLONG    sdataleng;`
	`  char      *recvdata;`
	`  DCLONG    *rdataleng;`
	`  DCLONG    inbufleng;`
	`  DCLONG    *time;`
	`  DCLONG    watchtime;`

＜ 64 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`  int  dc_mcf_sendrecv(action,`
`                      commform,`
`                      termnam,`
`                      resv01,`
`                      senddata,`
`                      sdataleng,`
`                      recvdata,`
`                      rdataleng,`
`                      inbufleng,`
`                      time,`
`                      watchtime)`
`  int       action;`
`  int       commform;`
`  char      *termnam;`
`  char      *resv01;`
`  char      *senddata;`
`  int       sdataleng;`
`  char      *recvdata;`
`  int       *rdataleng;`
`  int       inbufleng;`
`  int       *time;`
`  int       watchtime;`	

(5) dc_mcf_sendsync － 同期型メッセージの送信

(a) ANSI C，C++の形式

バージョン 6 以前	バージョン 7
＜ 32 ビットアーキテクチャの場合＞	`#include <dcmcf.h>` `int  dc_mcf_sendsync(DCLONG action,`

バージョン 6 以前	バージョン 7

#include <dcmcf.h>
  int dc_mcf_sendsync(long action,
  long commform,
  char *termnam,
  char *resv01,
  char *senddata,
  long sdataleng,
  char *resv02,
  long opcd,
  long watchtime)
	DCLONG commform,
	  char *termnam,
	  char *resv01,
	  char *senddata,
	  DCLONG sdataleng,
	  char *resv02,
	  DCLONG opcd,
	  DCLONG watchtime)

＜ 64 ビットアーキテクチャの場合＞
#include <dcmcf.h>
  int dc_mcf_sendsync(int action,
  int commform,
  char *termnam,
  char *resv01,
  char *senddata,
  int sdataleng,
  char *resv02,
  int opcd,
  int watchtime)	

(b)　K&R 版 C の形式

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
#include <dcmcf.h>
  int dc_mcf_sendsync(action,
  commform,
  termnam,
  resv01,
  senddata,
  sdataleng,
  resv02,
  opcd,
  watchtime)
  long action;
  long commform;
  char *termnam;
  char *resv01;
  char *senddata;
  long sdataleng;
  char *resv02;
  long opcd;
  long watchtime;
	#include <dcmcf.h>
	  int dc_mcf_sendsync(action,
	  commform,
	  termnam,
	  resv01,
	  senddata,
	  sdataleng,
	  resv02,
	  opcd,
	  watchtime)
	  DCLONG action;
	  DCLONG commform;
	  char *termnam;
	  char *resv01;
	  char *senddata;
	  DCLONG sdataleng;
	  char *resv02;
	  DCLONG opcd;
	  DCLONG watchtime;

＜ 64 ビットアーキテクチャの場合＞
#include <dcmcf.h>
  int dc_mcf_sendsync(action,
  commform,
  termnam,
  resv01,
  senddata,
  sdataleng,
  resv02,
  opcd,
  watchtime)
  int action;
  int commform;	

バージョン 6 以前	バージョン 7
char *termnam; char *resv01; char *senddata; int sdataleng; char *resv02; int opcd; int watchtime;	#include <dcmcf.h> int dc_mcf_sendsync(action, commform, termnam, resv01, senddata, sdataleng, resv02, opcd, watchtime) DCLONG action; DCLONG commform; char *termnam; char *resv01; char *senddata; DCLONG sdataleng; char *resv02; DCLONG opcd; DCLONG watchtime;

付録 C.2　ユーザオウンコーディング

ここでは，ユーザオウンコーディングの変更一覧を示します。

(1)　入力セグメント判定 UOC

(a)　形式

ANSI C，C++の形式

バージョン 6 以前	バージョン 7
＜ 32 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> long uoc_func(dctcp_uoc_sgck *parm)	#include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> DCLONG uoc_func(dctcp_uoc_sgck *parm)
＜ 64 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> int uoc_func(dctcp_uoc_sgck *parm)	

K&R 版 C の形式

バージョン 6 以前	バージョン 7
＜ 32 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h>	#include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> DCLONG uoc_func(parm)

バージョン 6 以前	バージョン 7
long uoc_func(parm) dctcp_uoc_sgck *parm ;	dctcp_uoc_sgck *parm ;
＜ 64 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> int uoc_func(parm) dctcp_uoc_sgck *parm ;	

(b) パラメタの内容

dctcp_uoc_sgck の内容

バージョン 6 以前	バージョン 7
＜ 32 ビットアーキテクチャの場合＞ typedef struct { long pro_kind; char le_name[9]; char reserve1[7]; long rcv_prim; char *rcv_data_adr; long rcv_data_size; char *uoc_inf_adr; long uoc_inf_size; dctcp_sguoc_prot *pro_indv_ifa; dctcp_uoctimer_inf *ptimerinf_adr; long rtn_detail; long reserve2[2]; } dctcp_uoc_sgck;	typedef struct { DCLONG pro_kind; char le_name[9]; char reserve1[7]; DCLONG rcv_prim; char *rcv_data_adr; DCLONG rcv_data_size; char *uoc_inf_adr; DCLONG uoc_inf_size; dctcp_sguoc_prot *pro_indv_ifa; dctcp_uoctimer_inf *ptimerinf_adr; DCLONG rtn_detail; DCLONG rcv_buf_size; DCLONG reserve2[1]; } dctcp_uoc_sgck;
＜ 64 ビットアーキテクチャの場合＞ typedef struct { int pro_kind; char le_name[9]; char reserve1[7]; int rcv_prim; char *rcv_data_adr; int rcv_data_size; char *uoc_inf_adr; int uoc_inf_size; dctcp_sguoc_prot *pro_indv_ifa; dctcp_uoctimer_inf *ptimerinf_adr; int rtn_detail; int reserve2[2]; } dctcp_uoc_sgck;	

dctcp_uoctimer_inf（タイマ情報），dctcp_sguoc_prot（領域アドレス）の内容

バージョン 6 以前	バージョン 7
＜ 32 ビットアーキテクチャの場合＞	typedef struct { DCLONG timer_code;

バージョン 6 以前	バージョン 7

typedef struct {
    long timer_code;
    long timer_value;
    long timer_result;
    long reserve1[2];
} dctcp_uoctimer_inf;
 	
typedef struct {
    long rest_data_size;
    long next_data_size;
    char *next_data_adr;
    long now_data_size;
    long reserve[2];
} dctcp_sguoc_prot;
	DCLONG timer_value;
	    DCLONG timer_result;
	    DCLONG reserve1[2];
	} dctcp_uoctimer_inf;
	
	typedef struct {
	    DCLONG rest_data_size;
	    DCLONG next_data_size;
	    char *next_data_adr;
	    DCLONG now_data_size;
	    DCLONG reserve[2];
	} dctcp_sguoc_prot;

＜64 ビットアーキテクチャの場合＞
typedef struct {
    int timer_code;
    int timer_value;
    int timer_result;
    int reserve1[2];
} dctcp_uoctimer_inf;
 	
typedef struct {
    int rest_data_size;
    int next_data_size;
    char *next_data_adr;
    int now_data_size;
    int reserve[2];
} dctcp_sguoc_prot;	

(2)　入力メッセージ編集 UOC

(a)　形式

ANSI C，C++の形式

バージョン 6 以前	バージョン 7
＜32 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> long uoc_func(dcmcf_uoc_min_n *parm)	#include <dcmcf.h> #include <dcmcfuoc.h> DCLONG uoc_func(dcmcf_uoc_min_n *parm)
＜64 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> int uoc_func(dcmcf_uoc_min_n *parm)	

K&R 版 C の形式

バージョン 6 以前	バージョン 7
＜32 ビットアーキテクチャの場合＞	#include <dcmcf.h> #include <dcmcfuoc.h> DCLONG uoc_func(parm)

バージョン 6 以前	バージョン 7
#include <dcmcf.h> #include <dcmcfuoc.h> long uoc_func(parm) dcmcf_uoc_min_n *parm ;	dcmcf_uoc_min_n *parm ;
＜ 64 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> int uoc_func(parm) dcmcf_uoc_min_n *parm ;	

(b) パラメタの内容

dcmcf_uoc_min_n の内容

バージョン 6 以前	バージョン 7
＜ 32 ビットアーキテクチャの場合＞ typedef struct { long pro_kind; char le_name[9]; char reserve1[7]; long rcv_prim; dcmcf_uocbuff_list_n *buflist_adr; dcmcf_uocbuff_list_n *ebuflist_adr; char aplname[9]; char reserve2[7]; char *pro_indv_ifa; long rtn_detail; char reserve3[8]; } dcmcf_uoc_min_n;	typedef struct { DCLONG pro_kind; char le_name[9]; char reserve1[7]; DCLONG rcv_prim; dcmcf_uocbuff_list_n *buflist_adr; dcmcf_uocbuff_list_n *ebuflist_adr; char aplname[9]; char reserve2[7]; char *pro_indv_ifa; DCLONG rtn_detail; char reserve3[8]; } dcmcf_uoc_min_n;
＜ 64 ビットアーキテクチャの場合＞ typedef struct { int pro_kind; char le_name[9]; char reserve1[7]; int rcv_prim; dcmcf_uocbuff_list_n *buflist_adr; dcmcf_uocbuff_list_n *ebuflist_adr; char aplname[9]; char reserve2[7]; char *pro_indv_ifa; int rtn_detail; char reserve3[8]; } dcmcf_uoc_min_n;	

dcmcf_uocbuff_list_n（バッファリスト）の内容

バージョン 6 以前	バージョン 7
＜ 32 ビットアーキテクチャの場合＞ typedef struct { long buf_num; long used_buf_num;	typedef struct { DCLONG buf_num; DCLONG used_buf_num; char reserve1[8]; dcmcf_uocbufinf_n

バージョン 6 以前	バージョン 7
char reserve1[8]; dcmcf_uocbufinf_n buf_array[DCMCF_UOC_BUFF_MAX]; } dcmcf_uocbuff_list_n;	buf_array[DCMCF_UOC_BUFF_MAX]; } dcmcf_uocbuff_list_n;
＜64 ビットアーキテクチャの場合＞ typedef struct { int buf_num; int used_buf_num; char reserve1[8]; dcmcf_uocbufinf_n buf_array[DCMCF_UOC_BUFF_MAX]; } dcmcf_uocbuff_list_n;	

dcmcf_uocbufinf_n（バッファ情報）の内容

バージョン 6 以前	バージョン 7
＜32 ビットアーキテクチャの場合＞ typedef struct { char *buf_adr; unsigned long buf_size; unsigned long seg_size; char reserve1[4]; dcmcfuoc_w_type buff_id; long buff_addr; char reserve2[4]; } dcmcf_uocbufinf_n;	typedef struct { char *buf_adr; DCULONG buf_size; DCULONG seg_size; char reserve1[4]; dcmcfuoc_w_type buff_id; DCMLONG buff_addr; char reserve2[4]; } dcmcf_uocbufinf_n;
＜64 ビットアーキテクチャの場合＞ typedef struct { char *buf_adr; unsigned int buf_size; unsigned int seg_size; char reserve1[4]; dcmcfuoc_w_type buff_id; char *buff_addr; char reserve2[4]; } dcmcf_uocbufinf_n;	

(3) 出力メッセージ編集 UOC

(a) 形式

ANSI C，C++の形式

バージョン 6 以前	バージョン 7
＜32 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> long uoc_func(dcmcf_uoc_mout_n *parm)	#include <dcmcf.h> #include <dcmcfuoc.h> DCLONG uoc_func(dcmcf_uoc_mout_n *parm)
＜64 ビットアーキテクチャの場合＞	

バージョン 6 以前	バージョン 7

`#include <dcmcf.h>`
`#include <dcmcfuoc.h>`
`int  uoc_func(dcmcf_uoc_mout_n *parm)`
	`#include <dcmcf.h>`
	`#include <dcmcfuoc.h>`
	`DCLONG  uoc_func(dcmcf_uoc_mout_n *parm)`

K&R 版 C の形式

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`#include <dcmcfuoc.h>`
`long  uoc_func(parm)`
 	
`dcmcf_uoc_mout_n *parm ;`
	`#include <dcmcf.h>`
	`#include <dcmcfuoc.h>`
	`DCLONG  uoc_func(parm)`
	
	`dcmcf_uoc_mout_n *parm ;`

＜ 64 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`#include <dcmcfuoc.h>`
`int  uoc_func(parm)`
 	
`dcmcf_uoc_mout_n *parm ;`	

(b)　パラメタの内容

dcmcf_uoc_mout_n の内容

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
`typedef struct {`
`    long pro_kind;`
`    char le_name[9];`
`    char reserve1[7];`
`    dcmcf_uocbuff_list_n *buflist_adr;`
`    dcmcf_uocbuff_list_n *ebuflist_adr;`
`    long output_no;`
`    char msg_type;`
`    char outputno_flag;`
`    char resend_flag;`
`    char reserve2[1];`
`    char *pro_indv_ifa;`
`    long rtn_detail;`
`    char reserve3[20];`
`} dcmcf_uoc_mout_n;`
	`typedef struct {`
	`    DCLONG pro_kind;`
	`    char le_name[9];`
	`    char reserve1[7];`
	`    dcmcf_uocbuff_list_n *buflist_adr;`
	`    dcmcf_uocbuff_list_n *ebuflist_adr;`
	`    DCLONG output_no;`
	`    char msg_type;`
	`    char outputno_flag;`
	`    char resend_flag;`
	`    char reserve2[1];`
	`    char *pro_indv_ifa;`
	`    DCLONG rtn_detail;`
	`    char reserve3[20];`
	`} dcmcf_uoc_mout_n;`

＜ 64 ビットアーキテクチャの場合＞
`typedef struct {`
`    int pro_kind;`
`    char le_name[9];`
`    char reserve1[7];`
`    dcmcf_uocbuff_list_n *buflist_adr;`
`    dcmcf_uocbuff_list_n *ebuflist_adr;`
`    int output_no;`
`    char msg_type;`
`    char outputno_flag;`
`    char resend_flag;`
`    char reserve2[1];`	

バージョン 6 以前	バージョン 7

`    char *pro_indv_ifa;`
`    int rtn_detail;`
`    char reserve3[20];`
`} dcmcf_uoc_mout_n;`
	`typedef struct {`
	`    DCLONG pro_kind;`
	`    char le_name[9];`
	`    char reserve1[7];`
	`    dcmcf_uocbuff_list_n *buflist_adr;`
	`    dcmcf_uocbuff_list_n *ebuflist_adr;`
	`    DCLONG output_no;`
	`    char msg_type;`
	`    char outputno_flag;`
	`    char resend_flag;`
	`    char reserve2[1];`
	`    char *pro_indv_ifa;`
	`    DCLONG rtn_detail;`
	`    char reserve3[20];`
	`} dcmcf_uoc_mout_n;`

dcmcf_uocbuff_list_n（バッファリスト），dcmcf_uocbufinf_n（バッファ情報）の内容

入力メッセージ編集 UOC のパラメタの内容と同じです。「付録 C.2(2)(b)　パラメタの内容」を参照してください。

(4)　送信メッセージの通番編集 UOC

(a)　形式

ANSI C，C++の形式

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`long send_uoc(long flags,`
`                char *termname,`
`                long sendno,`
`                long sendid,`
`                long dataleng,`
`                char *senddata)`
＜ 64 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`int send_uoc(int flags,`
`               char *termname,`
`               int sendno,`
`               int sendid,`
`               int dataleng,`
`               char *senddata)`
	`#include <dcmcf.h>`
	`DCLONG send_uoc(DCLONG flags,`
	`                  char *termname,`
	`                  DCLONG sendno,`
	`                  DCLONG sendid,`
	`                  DCLONG dataleng,`
	`                  char *senddata)`

K&R 版 C の形式

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`  long send_uoc(flags,`
`               termname,`
	`#include <dcmcf.h>`
	`  DCLONG send_uoc(flags,`
	`                  termname,`
	`                  sendno,`

バージョン 6 以前	バージョン 7
sendno, sendid, dataleng, senddata) long flags; char *termname; long sendno; long sendid; long dataleng; char *senddata;	sendid, dataleng, senddata) DCLONG flags; char *termname; DCLONG sendno; DCLONG sendid; DCLONG dataleng; char *senddata;
＜64 ビットアーキテクチャの場合＞ #include <dcmcf.h> int send_uoc(flags, termname, sendno, sendid, dataleng, senddata) int flags; char *termname; int sendno; int sendid; int dataleng; char *senddata;	

(5) コネクション確立 UOC

(a) 形式

ANSI C，C++の形式

バージョン 6 以前	バージョン 7
＜32 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> long con_uoc(dcmtcp_uoc_con_n *parm)	#include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> DCLONG con_uoc(dcmtcp_uoc_con_n *parm)
＜64 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> int con_uoc(dcmtcp_uoc_con_n *parm)	

K&R 版 C の形式

バージョン 6 以前	バージョン 7
＜32 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> long con_uoc(parm)	#include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> DCLONG con_uoc(parm) dcmtcp_uoc_con_n *parm;

バージョン 6 以前	バージョン 7
dcmtcp_uoc_con_n *parm; ＜ 64 ビットアーキテクチャの場合＞ #include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> int con_uoc(parm) dcmtcp_uoc_con_n *parm;	#include <dcmcf.h> #include <dcmcfuoc.h> #include <dcmtcpu.h> DCLONG con_uoc(parm) dcmtcp_uoc_con_n *parm;

(b) パラメタの内容

dcmtcp_uoc_con_n の内容

バージョン 6 以前	バージョン 7
＜ 32 ビットアーキテクチャの場合＞ typedef struct { char cn_name[9]; char reserve[7]; dcmtcp_cnuoc_oaddr *oaddr; unsigned short oportno; unsigned short iportno; long connect_permit; long reject_reason; long rtn_detail; } dcmtcp_uoc_con_n; ＜ 64 ビットアーキテクチャの場合＞ typedef struct { char cn_name[9]; char reserve[7]; dcmtcp_cnuoc_oaddr *oaddr; unsigned short oportno; unsigned short iportno; int connect_permit; int reject_reason; int rtn_detail; } dcmtcp_uoc_con_n;	typedef struct { char cn_name[9]; char reserve[7]; dcmtcp_cnuoc_oaddr *oaddr; unsigned short oportno; unsigned short iportno; DCLONG connect_permit; DCLONG reject_reason; DCLONG rtn_detail; } dcmtcp_uoc_con_n;

dcmtcp_cnuoc_oaddr（相手システムのアドレス情報）の内容

バージョン 6 以前とバージョン 7 で，差異はありません。

(6) 受信メッセージ判定 UOC

(a) 形式

ANSI C，C++の形式

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`#include <dcmcfuoc.h>`
`#include <dcmtcpu.h>`
`long msgrep_uoc(dctcp_uoc_msgrep *parm)`
	`#include <dcmcf.h>`
	`#include <dcmcfuoc.h>`
	`#include <dcmtcpu.h>`
	`DCLONG msgrep_uoc(dctcp_uoc_msgrep *parm)`

＜ 64 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`#include <dcmcfuoc.h>`
`#include <dcmtcpu.h>`
`int msgrep_uoc(dctcp_uoc_msgrep *parm)`	

K&R 版 C の形式

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`#include <dcmcfuoc.h>`
`#include <dcmtcpu.h>`
`long msgrep_uoc(parm)`
 	
`dctcp_uoc_msgrep *parm;`
	`#include <dcmcf.h>`
	`#include <dcmcfuoc.h>`
	`#include <dcmtcpu.h>`
	`DCLONG msgrep_uoc(parm)`
	
	`dctcp_uoc_msgrep *parm;`

＜ 64 ビットアーキテクチャの場合＞
`#include <dcmcf.h>`
`#include <dcmcfuoc.h>`
`#include <dcmtcpu.h>`
`int msgrep_uoc(parm)`
 	
`dctcp_uoc_msgrep *parm;`	

(b) パラメタの内容

dctcp_uoc_msgrep の内容

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
`typedef struct {`
`long pro_kind;`
`char cn_name[9];`
`char reserve1[7];`
`char le_name[9];`
`char reserve2[7];`
`long snd_status;`
`char *rcv_data_adr;`
	`typedef struct {`
	`DCLONG pro_kind;`
	`char cn_name[9];`
	`char reserve1[7];`
	`char le_name[9];`
	`char reserve2[7];`
	`DCLONG snd_status;`
	`char *rcv_data_adr;`
	`DCLONG rcv_data_size;`
	`DCLONG msg_type;`

バージョン 6 以前	バージョン 7
long rcv_data_size; long msg_type; long reason; long msg_reply; char *snd_data_adr; long snd_buff_size; long snd_data_size; long snd_cont; long rtn_detail; char reserve3[16]; }dctcp_uoc_msgrep;	DCLONG reason; DCLONG msg_reply; char *snd_data_adr; DCLONG snd_buff_size; DCLONG snd_data_size; DCLONG snd_cont; DCLONG rtn_detail; char reserve3[16]; }dctcp_uoc_msgrep;
＜64 ビットアーキテクチャの場合＞ typedef struct { int pro_kind; char cn_name[9]; char reserve1[7]; char le_name[9]; char reserve2[7]; int snd_status; char *rcv_data_adr; int rcv_data_size; int msg_type; int reason; int msg_reply; char *snd_data_adr; int snd_buff_size; int snd_data_size; int snd_cont; int rtn_detail; char reserve3[16]; }dctcp_uoc_msgrep;	

付録 C.3　MCF イベントインタフェース

ここでは，MCF イベントインタフェースの変更一覧を示します。

(1)　MCF イベントの共通ヘッダの形式

バージョン 6 以前	バージョン 7

＜32 ビットアーキテクチャの場合＞
struct dc_mcf_evtheader {
char mcfevt_name[9] ;
char le_name[16] ;
char cn_name[9] ;
unsigned char format_kind;
char reserve01;
long time ;
};
	struct dc_mcf_evtheader {
	char mcfevt_name[9] ;
	char le_name[16] ;
	char cn_name[9] ;
	unsigned char format_kind;
	char reserve01;
	DCLONG time ;
	};

＜64 ビットアーキテクチャの場合＞ struct dc_mcf_evtheader { char mcfevt_name[9] ; char le_name[16] ; char cn_name[9] ;	

バージョン６以前	バージョン７

unsigned char format_kind;
char reserve01;
int time ;
};
	struct dc_mcf_evtheader {
	char mcfevt_name[9] ;
	char le_name[16] ;
	char cn_name[9] ;
	unsigned char format_kind;
	char reserve01;
	DCLONG time ;
	};

(2) ERREVT1 の形式

バージョン６以前とバージョン７で，差異はありません。

(3) ERREVT2 の形式

バージョン６以前とバージョン７で，差異はありません。

(4) ERREVT3 の形式

バージョン６以前とバージョン７で，差異はありません。

(5) ERREVTA の形式

バージョン６以前	バージョン７
＜32 ビットアーキテクチャの場合＞ struct dc_mcf_evta_type { struct dc_mcf_evtheader evtheader ; char reserve01[12] ; char reserve02[10] ; char reserve03[2] ; char ap_name[10] ; char reserve04[2] ; char reserve05[32] ; char reserve06[32] ; long user_leng ; char user_data[16] ; char reserve07[16] ; } ;	struct dc_mcf_evta_type { struct dc_mcf_evtheader evtheader ; char reserve01[12] ; char reserve02[10] ; char reserve03[2] ; char ap_name[10] ; char reserve04[2] ; char reserve05[32] ; char reserve06[32] ; DCLONG user_leng ; char user_data[16] ; char reserve07[16] ; } ;
＜64 ビットアーキテクチャの場合＞ struct dc_mcf_evta_type { struct dc_mcf_evtheader evtheader ; char reserve01[12] ; char reserve02[10] ; char reserve03[2] ; char ap_name[10] ; char reserve04[2] ; char reserve05[32] ; char reserve06[32] ; int user_leng ; char user_data[16] ; char reserve07[16] ; } ;	

(6) CERREVT の形式

バージョン 6 以前	バージョン 7

＜ 32 ビットアーキテクチャの場合＞
`typedef struct {`
`  struct dc_mcf_evtheader  header ;`
`  long err_fact ;`
`  long err_reason1 ;`
`  long err_reason2 ;`
 	
`  char reserve1[48] ;`
`} dcmtcp_cerrevt ;`
	`typedef struct {`
	`  struct dc_mcf_evtheader  header ;`
	`  DCLONG err_fact ;`
	`  DCLONG err_reason1 ;`
	`  DCLONG err_reason2 ;`
	`  char group_name[16];`
	`  char reserve1[32] ;`
	`} dcmtcp_cerrevt ;`

＜ 64 ビットアーキテクチャの場合＞
`typedef struct {`
`  struct dc_mcf_evtheader  header ;`
`  int err_fact ;`
`  int err_reason1 ;`
`  int err_reason2 ;`
 	
`  char reserve1[48] ;`
`} dcmtcp_cerrevt ;`	

(7) COPNEVT，CCLSEVT の形式

バージョン 6 以前	バージョン 7

`typedef struct {`
`  struct dc_mcf_evtheader  header ;`
 	
`  char reserve1[58] ;`
`} dcmtcp_statevt ;`
	`typedef struct {`
	`  struct dc_mcf_evtheader  header ;`
	`  char group_name[16];`
	`  char reserve1[40] ;`
	`} dcmtcp_statevt ;`

付録 C.4 MCF メイン関数のコーディング概要

MCF メイン関数のコーディング概要の変更一覧を示します。変更箇所は，図中の網掛け部分です。

(1) ANSI C，C++の場合

(a) バージョン６以前

```
#include <dcmtcp.h>                     /*TP1/NET/TCP/IP用ヘッダファイル      */

extern long segchk01(dcmcf_uoc_sgck *);※
                                        /*入力セグメント判定UOC関数extern宣言 */
extern long  msgrcv01(dcmcf_uoc_min_n *);
                                        /*入力メッセージ編集UOC関数extern宣言 */
extern long msgsend01(dcmcf_uoc_mout_n *);
                                        /*出力メッセージ編集UOC関数extern宣言 */

extern dcmcf_uoc_t  dcmcf_uoctbl;       /*UOCテーブルextern宣言               */

int main()
{
    dcmcf_uoctbl.segchk = (dcmcf_uocfunc)segchk01;※
                                        /*入力セグメント判定UOCアドレス設定   */
    dcmcf_uoctbl.msgrcv = (dcmcf_uocfunc)msgrcv01;
                                        /*入力メッセージ編集UOCアドレス設定   */
    dcmcf_uoctbl.msgsend = (dcmcf_uocfunc)msgsend01;
                                        /*出力メッセージ編集UOCアドレス設定   */

    dc_mcf_svstart();                    /*スタート関数呼び出し                */

    return 0;
}
```

注※

TP1/NET/TCP/IP が標準提供する入力セグメント判定 UOC を使用する場合は，「segchk01」の代わりに「dc_mcf_stduoc_tcp_segchk」をコーディングしてください。

(b) バージョン７

```
#include <dcmtcp.h>                     /*TP1/NET/TCP/IP用ヘッダファイル      */

extern DCLONG segchk01(dcmcf_uoc_sgck *);※
                                        /*入力セグメント判定UOC関数extern宣言 */
extern DCLONG msgrcv01(dcmcf_uoc_min_n *);
                                        /*入力メッセージ編集UOC関数extern宣言 */
extern DCLONG msgsend01(dcmcf_uoc_mout_n *);
                                        /*出力メッセージ編集UOC関数extern宣言 */

extern dcmcf_uoc_t  dcmcf_uoctbl;       /*UOCテーブルextern宣言               */

int main()
{
    dcmcf_uoctbl.segchk = (dcmcf_uocfunc)segchk01;※
                                        /*入力セグメント判定UOCアドレス設定   */
    dcmcf_uoctbl.msgrcv = (dcmcf_uocfunc)msgrcv01;
                                        /*入力メッセージ編集UOCアドレス設定   */
    dcmcf_uoctbl.msgsend = (dcmcf_uocfunc)msgsend01;
                                        /*出力メッセージ編集UOCアドレス設定   */

    dc_mcf_svstart();                    /*スタート関数呼び出し                */

    return 0;
}
```

注※

TP1/NET/TCP/IP が標準提供する入力セグメント判定 UOC を使用する場合は，「segchk01」の代わりに「dc_mcf_stduoc_tcp_segchk」をコーディングしてください。

(2) K&R 版 C の場合

(a) バージョン 6 以前

```
#include <dcmtcp.h>                     /*TP1/NET/TCP/IP用ヘッダファイル      */

extern  long    segchk01();※            /*入力セグメント判定UOC関数extern宣言*/
extern  long    msgrcv01();             /*入力メッセージ編集UOC関数extern宣言*/
extern  long    msgsend01();           /*出力メッセージ編集UOC関数extern宣言 */

extern  dcmcf_uoc_t     dcmcf_uoctbl;  /*UOCテーブルextern宣言               */

main()
{
   dcmcf_uoctbl.segchk = segchk01;※     /*入力セグメント判定UOCアドレス設定  */
   dcmcf_uoctbl.msgrcv = msgrcv01;      /*入力メッセージ編集UOCアドレス設定  */
   dcmcf_uoctbl.msgsend = msgsend01;    /*出力メッセージ編集UOCアドレス設定  */

   dc_mcf_svstart();                    /*スタート関数呼び出し               */
 }
```

注※

TP1/NET/TCP/IP が標準提供する入力セグメント判定 UOC を使用する場合は，「segchk01」の代わりに「dc_mcf_stduoc_tcp_segchk」をコーディングしてください。

(b) バージョン 7

```
#include <dcmtcp.h>                     /*TP1/NET/TCP/IP用ヘッダファイル      */

extern  DCLONG    segchk01();※          /*入力セグメント判定UOC関数extern宣言*/
extern  DCLONG    msgrcv01();           /*入力メッセージ編集UOC関数extern宣言*/
extern  DCLONG    msgsend01();         /*出力メッセージ編集UOC関数extern宣言 */

extern  dcmcf_uoc_t     dcmcf_uoctbl;  /*UOCテーブルextern宣言               */

main()
{
   dcmcf_uoctbl.segchk = segchk01;※     /*入力セグメント判定UOCアドレス設定  */
   dcmcf_uoctbl.msgrcv = msgrcv01;      /*入力メッセージ編集UOCアドレス設定  */
   dcmcf_uoctbl.msgsend = msgsend01;    /*出力メッセージ編集UOCアドレス設定  */

   dc_mcf_svstart();                    /*スタート関数呼び出し               */
 }
```

注※

TP1/NET/TCP/IP が標準提供する入力セグメント判定 UOC を使用する場合は，「segchk01」の代わりに「dc_mcf_stduoc_tcp_segchk」をコーディングしてください。

付録 C.5　ユーザアプリケーションプログラムの作成例

ユーザアプリケーションプログラムの作成例の変更一覧を示します。

(1)　バージョン 6 以前

```
/*********************************************************/
/*    C言語を使用したUAP作成例                           */
/*********************************************************/
#include <dcmcf.h>

void ex_uap1()
{
    char termnam[9];
    long rdataleng;
    long time;
    struct{
        char mcfctl[8];
        long msglen;
        char recvdata[2036];
    }recvmsg;
    struct{
        char mcfctl[8];
        long msglen;
        char senddata1[500];
    }sendmsg;
    char senddata2[512];
    static char resv01[9] = "¥0" ;  /*** 予備領域の初期化 ***/
    static char resv02[9] = "¥0" ;  /*** 予備領域の初期化 ***/
    static char resv03[9] = "¥0" ;  /*** 予備領域の初期化 ***/
    char *workadd = (char *)&recvmsg;

    dc_mcf_receive(DCMCFFRST,DCNOFLAGS,termnam,resv01,
                   workadd,&rdataleng,2048,&time) ;
                                /* 一方送信メッセージの受信 */
    /*************************************/
    /*********   データの処理    **********/
    /*************************************/

    dc_mcf_send(DCMCFEMI,DCMCFOUT,"PRINTER1",resv01,
                senddata2,504,resv02,DCNOFLAGS);
                                /* 一方送信メッセージの送信 */
    sendmsg.msglen = 504;       /* セグメント長の設定 */
    dc_mcf_send(DCMCFEMI,DCMCFOUT,"TERMNAM1",resv01,
                (char *)&sendmsg,504,resv02,DCNOFLAGS);
/* メッセージの送信 */
}
/********   C言語によるUAP 終わり    ********/
```

(2)　バージョン 7

```
/*********************************************************/
/*    C言語を使用したUAP作成例                           */
/*********************************************************/
```

```
#include <dcmcf.h>

void ex_uap1()
{
    char termnam[9];
    DCLONG rdataleng;
    DCLONG time;
    struct{
        char mcfctl[8];
        DCLONG msglen;
        char recvdata[2036];
    }recvmsg;
    struct{
        char mcfctl[8];
        DCLONG msglen;
        char senddata1[500];
    }sendmsg;
    char senddata2[512];
    static char resv01[9] = "¥0" ;  /*** 予備領域の初期化 ***/
    static char resv02[9] = "¥0" ;  /*** 予備領域の初期化 ***/
    static char resv03[9] = "¥0" ;  /*** 予備領域の初期化 ***/
    char *workadd = (char *)&recvmsg;

    dc_mcf_receive(DCMCFFRST,DCNOFLAGS,termnam,resv01,
                   workadd,&rdataleng,2048,&time) ;
                                  /* 一方送信メッセージの受信 */
    /*************************************/
    /*********   データの処理      ***********/
    /*************************************/

    dc_mcf_send(DCMCFEMI,DCMCFOUT,"PRINTER1",resv01,
                senddata2,504,resv02,DCNOFLAGS);
                                  /* 一方送信メッセージの送信 */
    sendmsg.msglen = 504;        /* セグメント長の設定 */
    dc_mcf_send(DCMCFEMI,DCMCFOUT,"TERMNAM1",resv01,
                (char *)&sendmsg,504,resv02,DCNOFLAGS);
/* メッセージの送信 */
}
/********   C言語によるUAP 終わり   ********/
```

付録 D　メッセージ送受信の処理の流れ

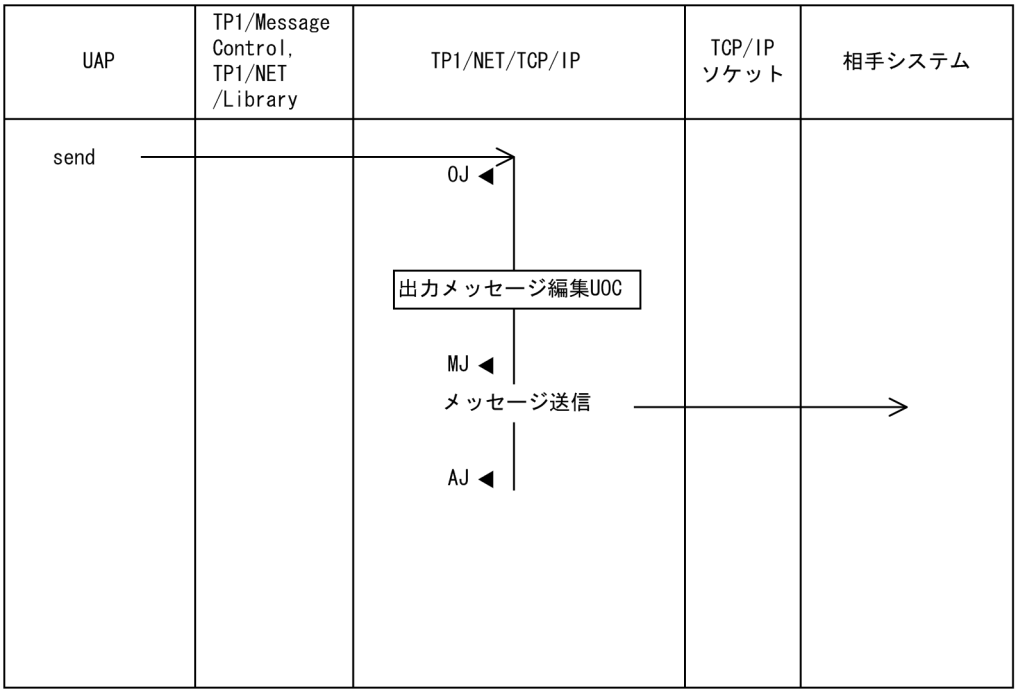
ここでは，次のときのメッセージを送受信するときのデータの流れ，ジャーナルの取得タイミングなどについて説明します。

- 一方送信メッセージの送信時（メッセージ送達確認機能を使用しない場合）（図 D-1）
- 一方送信メッセージの送信時（メッセージ送達確認機能を使用する場合）（図 D-2）
- 一方送信メッセージの受信時（図 D-3）
- 同期型メッセージ送信時（図 D-4）
- 同期型メッセージ受信時（図 D-5）
- 同期型メッセージ送受信時（図 D-6）
- 後続メッセージ監視時（図 D-7）

それぞれについて，以降の図に示します。

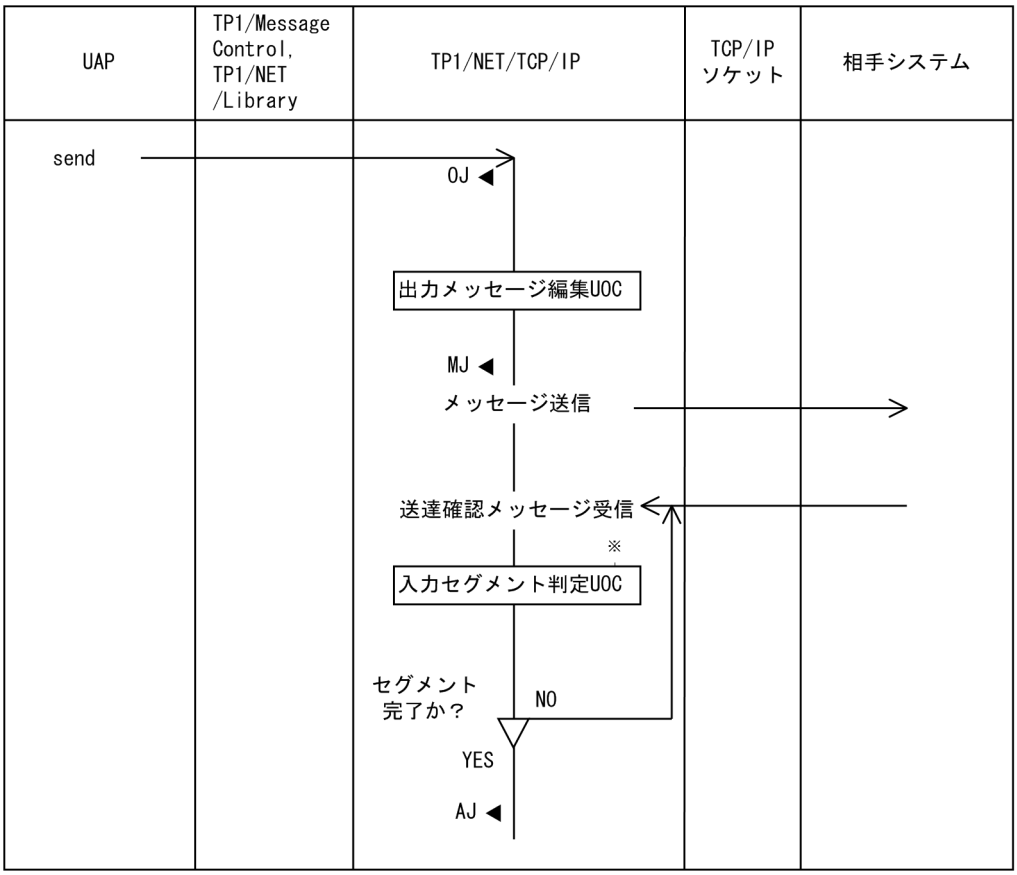
なお，ジャーナルのフォーマットについては，マニュアル「OpenTP1 運用と操作」を参照してください。

図 D-1　一方送信メッセージの送信時の処理の流れ（メッセージ送達確認機能を使用しない場合）



（凡例）　OJ ◀：メッセージ出力ジャーナル取得
AJ ◀：メッセージ送信完了ジャーナル取得
MJ ◀：メッセージジャーナル取得

図 D-2　一方送信メッセージの送信時の処理の流れ（メッセージ送達確認機能を使用する場合）



（凡例）　OJ ◀：メッセージ出力ジャーナル取得

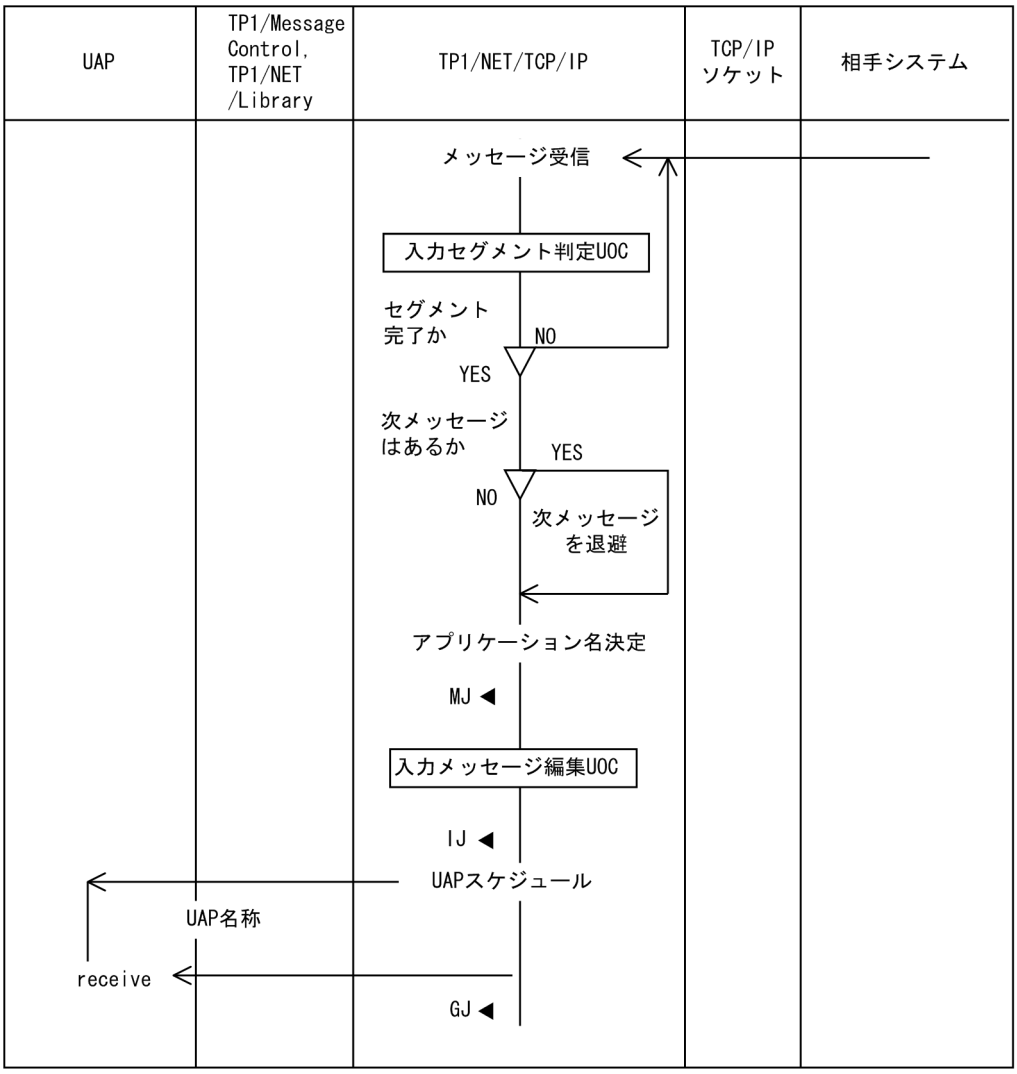
AJ ◀：メッセージ送信完了ジャーナル取得

MJ ◀：メッセージジャーナル取得

注※

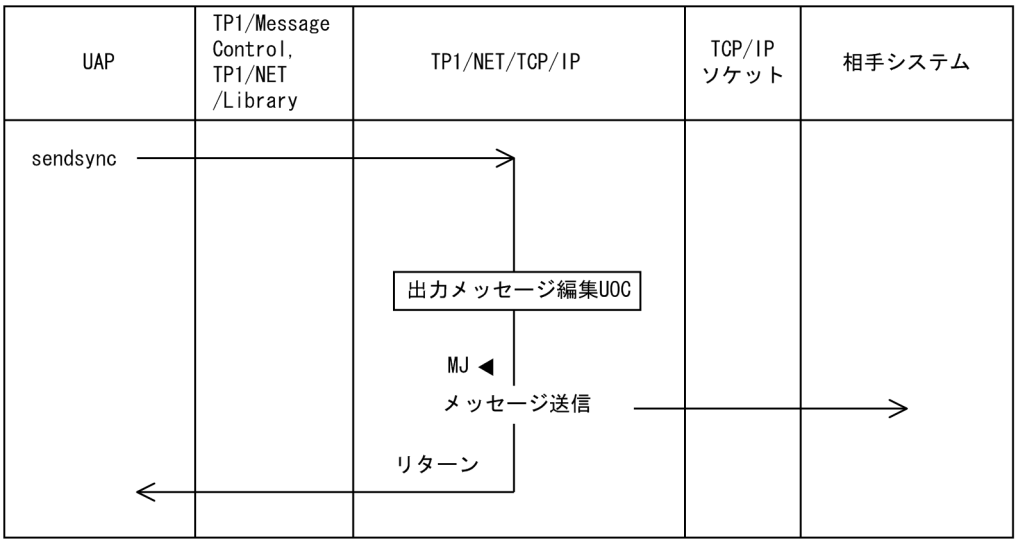
DCCMとのメッセージ送達確認機能を使用する場合，入力セグメント判定UOCを使用しないで，受信メッセージの組み立て機能を使用して入力セグメントを組み立てます。

図 D-3　一方送信メッセージの受信時の処理の流れ



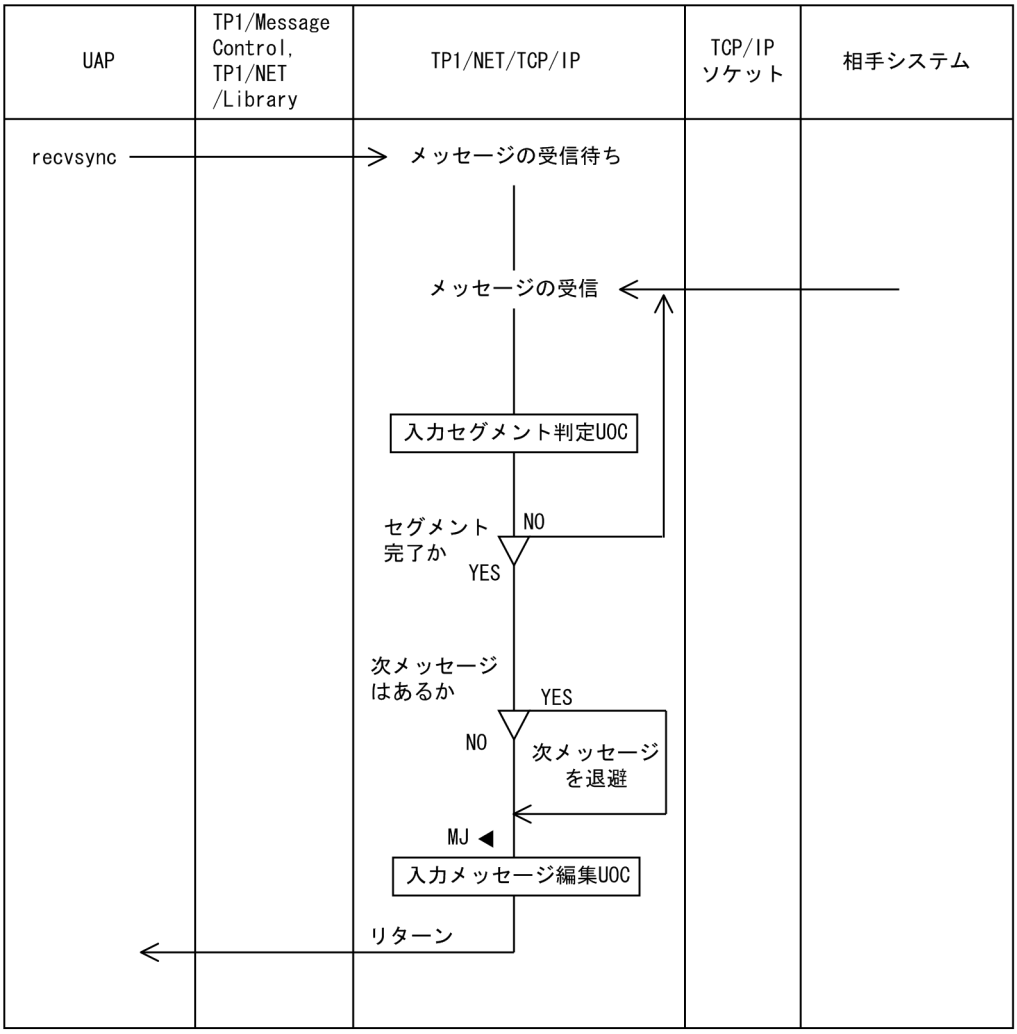
（凡例）MJ◀: メッセージジャーナル取得
IJ◀: メッセージ入力ジャーナル取得
GJ◀: メッセージ受信ジャーナル取得

図 D-4　同期型メッセージ送信時の処理の流れ



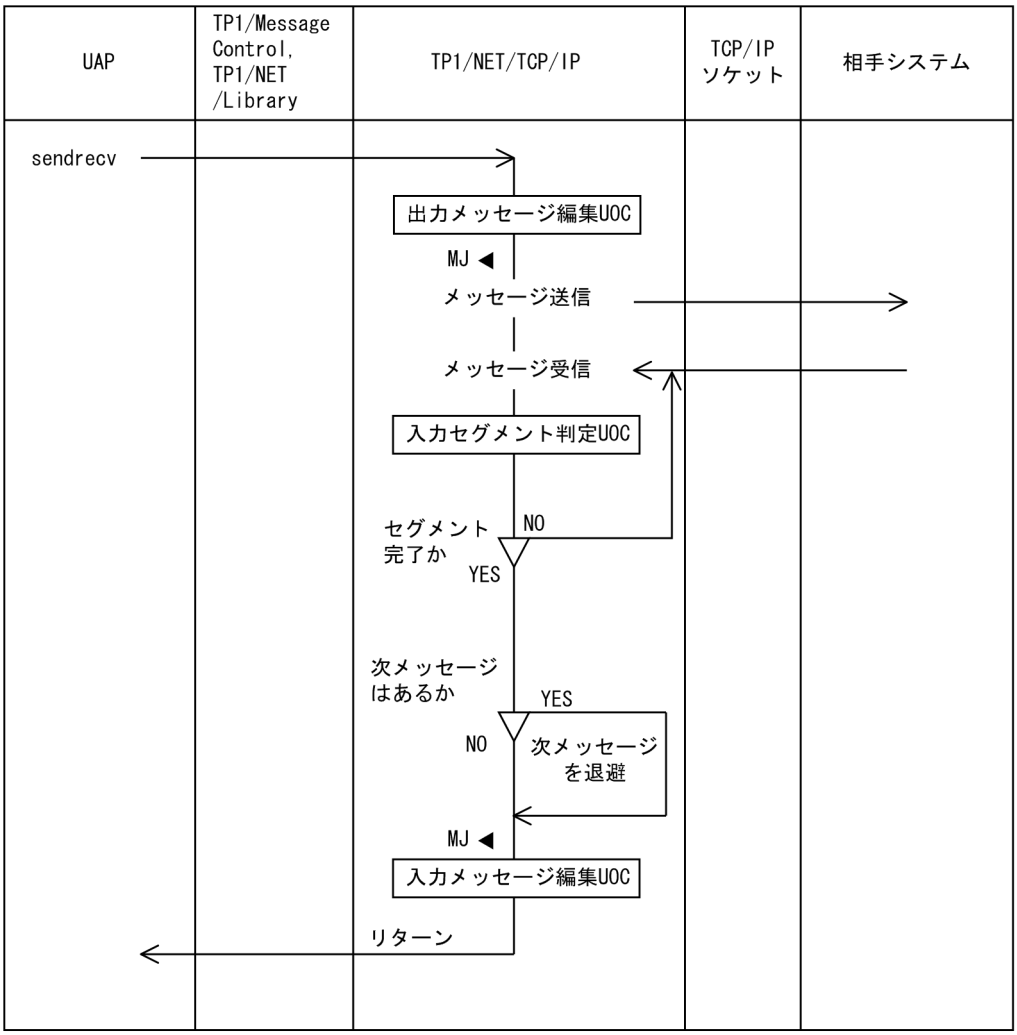
（凡例）　MJ ◀: メッセージジャーナル取得

図 D-5　同期型メッセージ受信時の処理の流れ



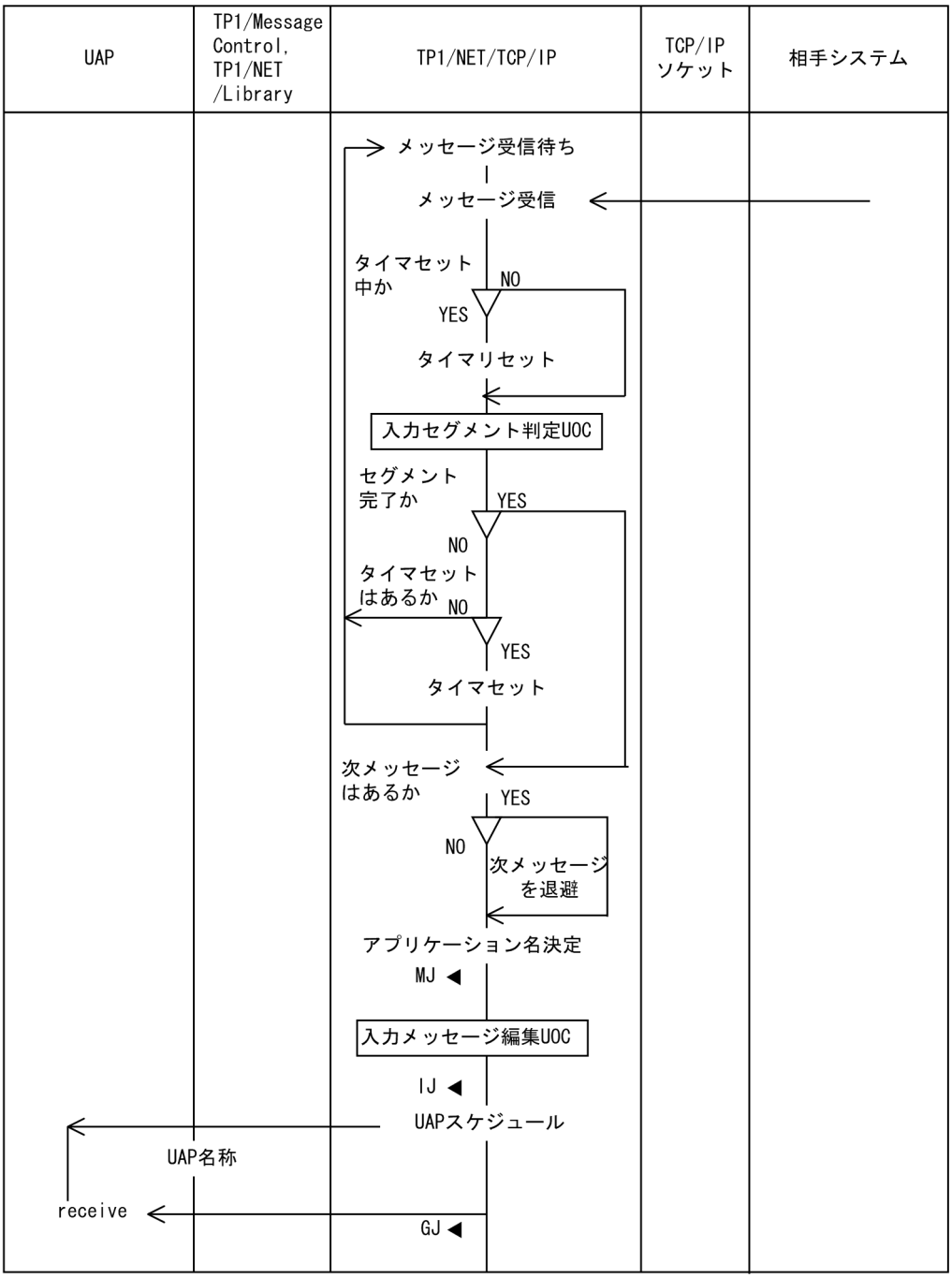
（凡例）　MJ ◀: メッセージジャーナル取得

図 D-6　同期型メッセージ送受信時の処理の流れ



（凡例）　MJ ◀: メッセージジャーナル取得

図 D-7　後続メッセージ監視時の処理の流れ



（凡例）MJ◀：メッセージジャーナル取得

IJ◀：メッセージ入力ジャーナル取得

GJ◀：メッセージ受信ジャーナル取得

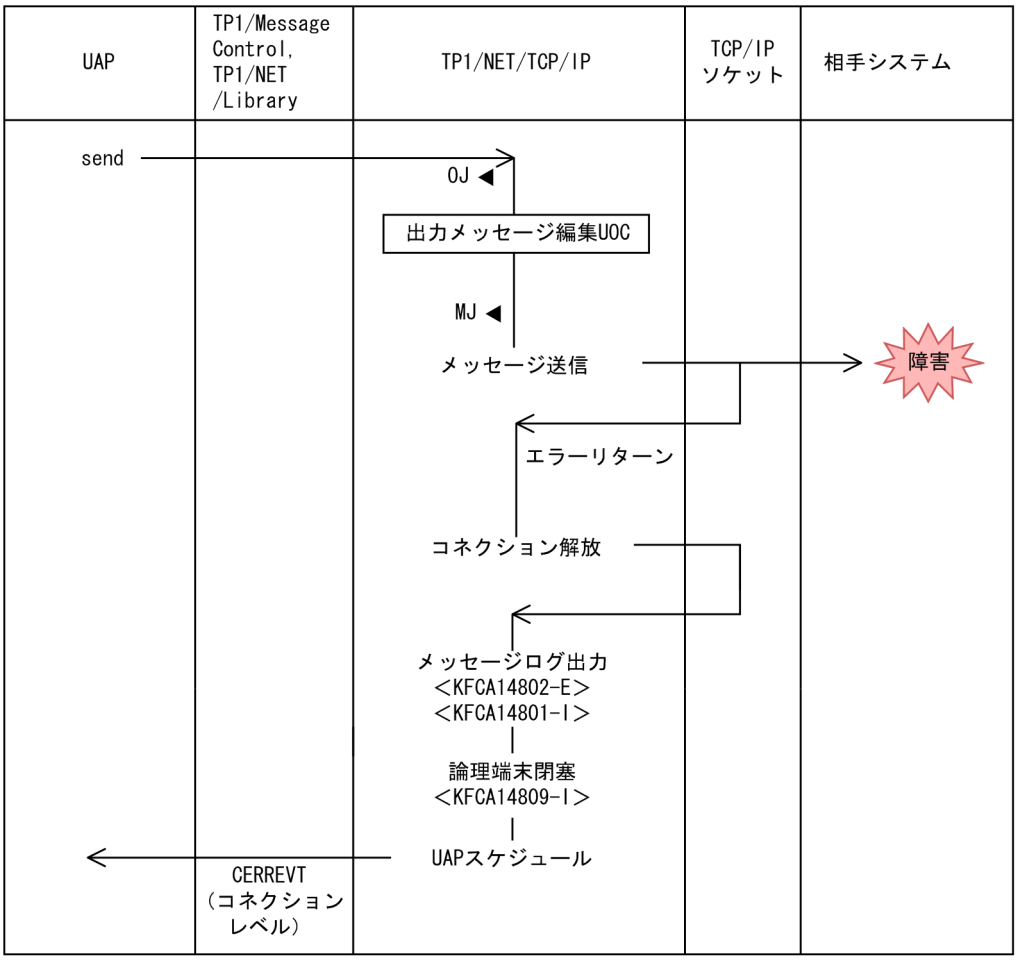
付録 E　障害発生時の処理の流れ

ここでは，次のときの障害発生時の処理の流れについて説明します。

- メッセージ送信でのコネクション障害時（図 E-1）
- メッセージ送信中のコネクション解放時（図 E-2）
- コネクション切断時（図 E-3）
- 入力セグメント判定 UOC エラー時（図 E-4）
- 入力メッセージ編集 UOC エラー時（図 E-5）
- 出力メッセージ編集 UOC エラー時（図 E-6）
- 後続メッセージ監視タイムアウト時（図 E-7）
- コネクション確立監視時間タイムアウト時（図 E-8）
- 送信完了監視時間タイムアウト時（図 E-9）
- 無通信状態監視時間タイムアウト時（図 E-10）

それぞれについて，以降の図に示します。

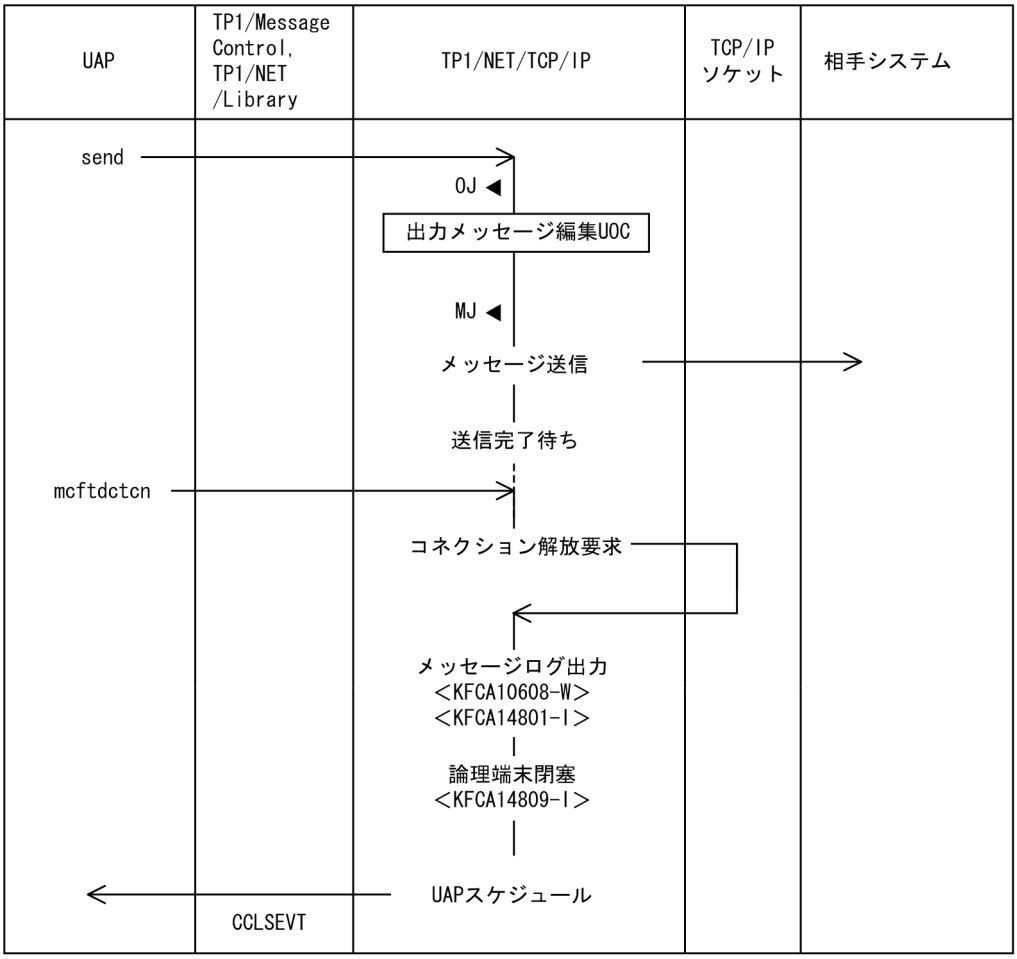
図 E-1　メッセージ送信でのコネクション障害時の処理の流れ



（凡例）　OJ ◀：メッセージ出力ジャーナル取得

MJ ◀：メッセージジャーナル取得

図 E-2　メッセージ送信中のコネクション解放時の処理の流れ



（凡例）　OJ◀：メッセージ出力ジャーナル取得

MJ◀：メッセージジャーナル取得

図 E-3　コネクション切断時の処理の流れ

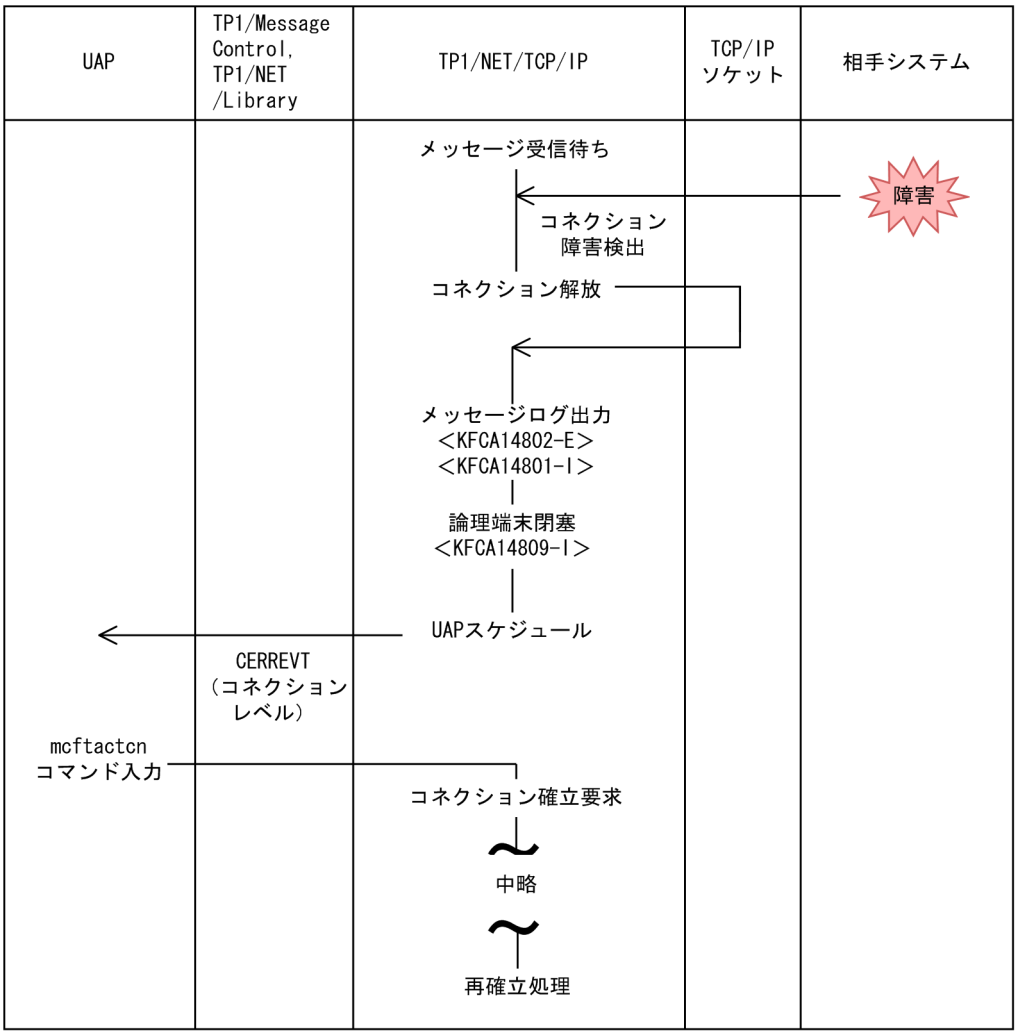


図 E-4　入力セグメント判定 UOC エラー時の処理の流れ

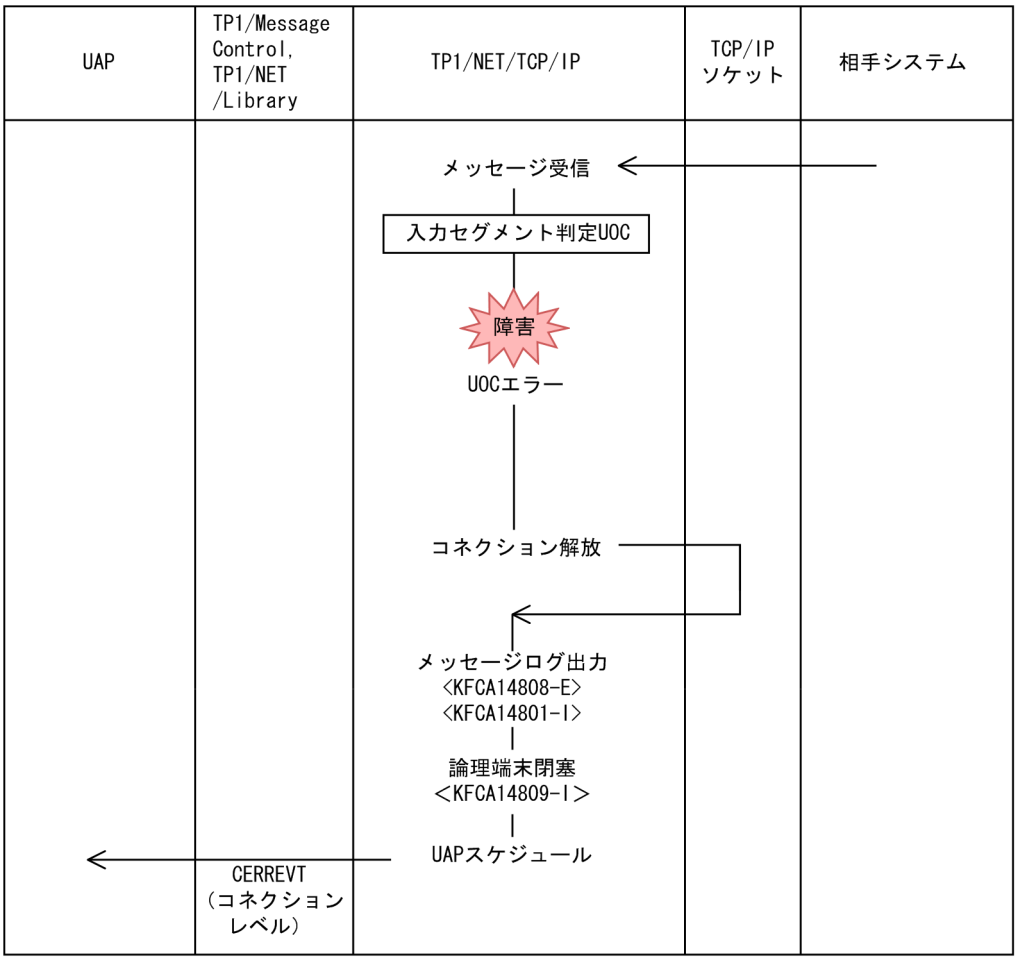
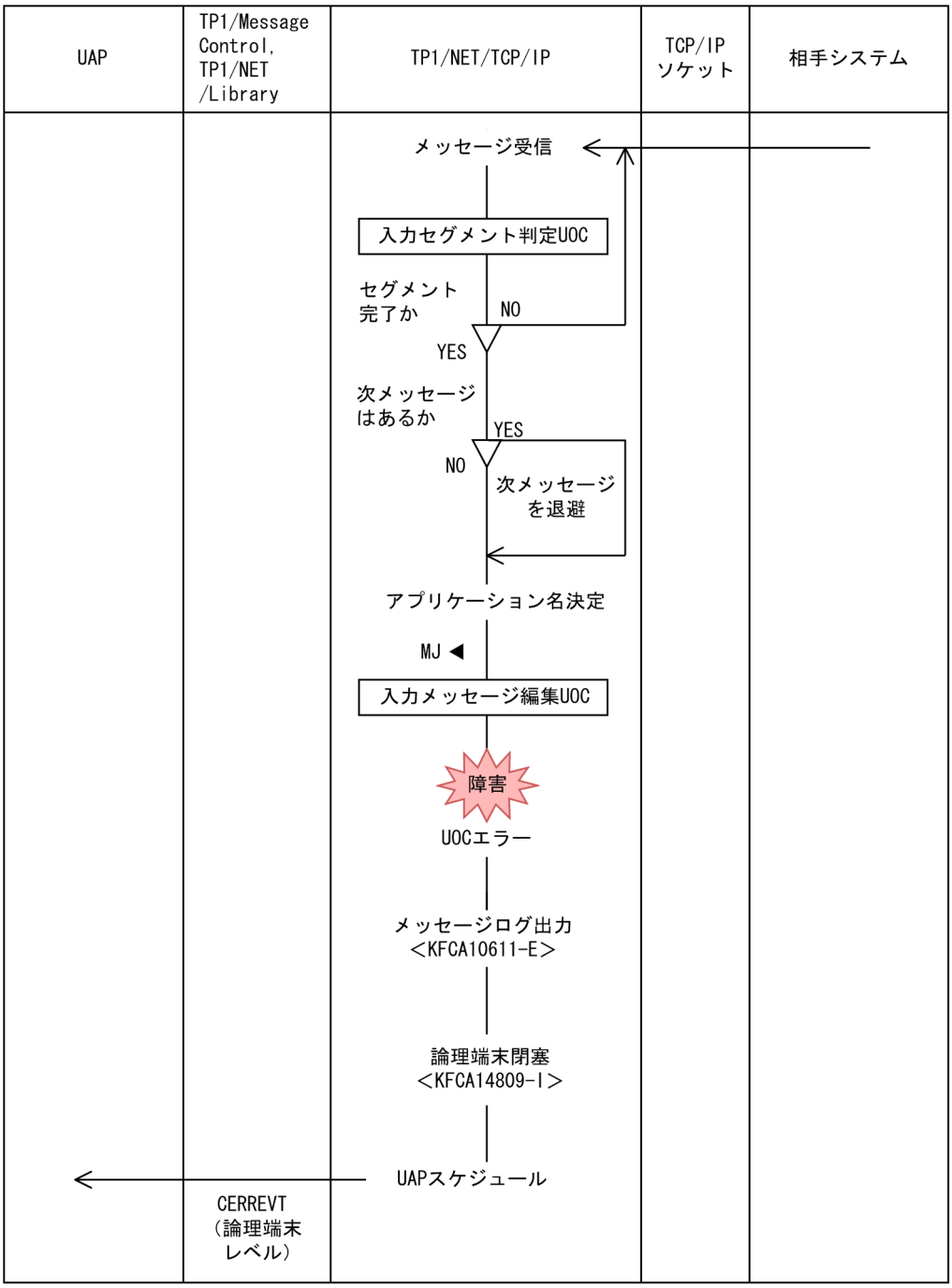
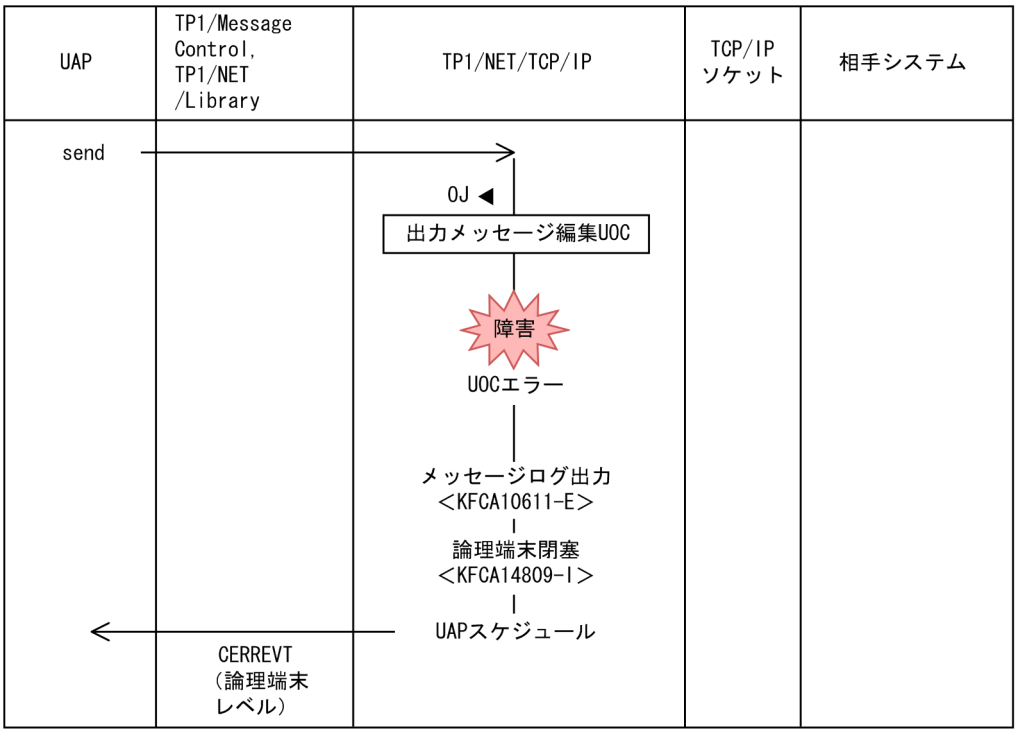


図 E-5　入力メッセージ編集 UOC エラー時の処理の流れ



（凡例）MJ ◀: メッセージジャーナル取得

図 E-6　出力メッセージ編集 UOC エラー時の処理の流れ



（凡例）　OJ◀：メッセージ出力ジャーナル取得

図 E-7　後続メッセージ監視タイムアウト時の処理の流れ

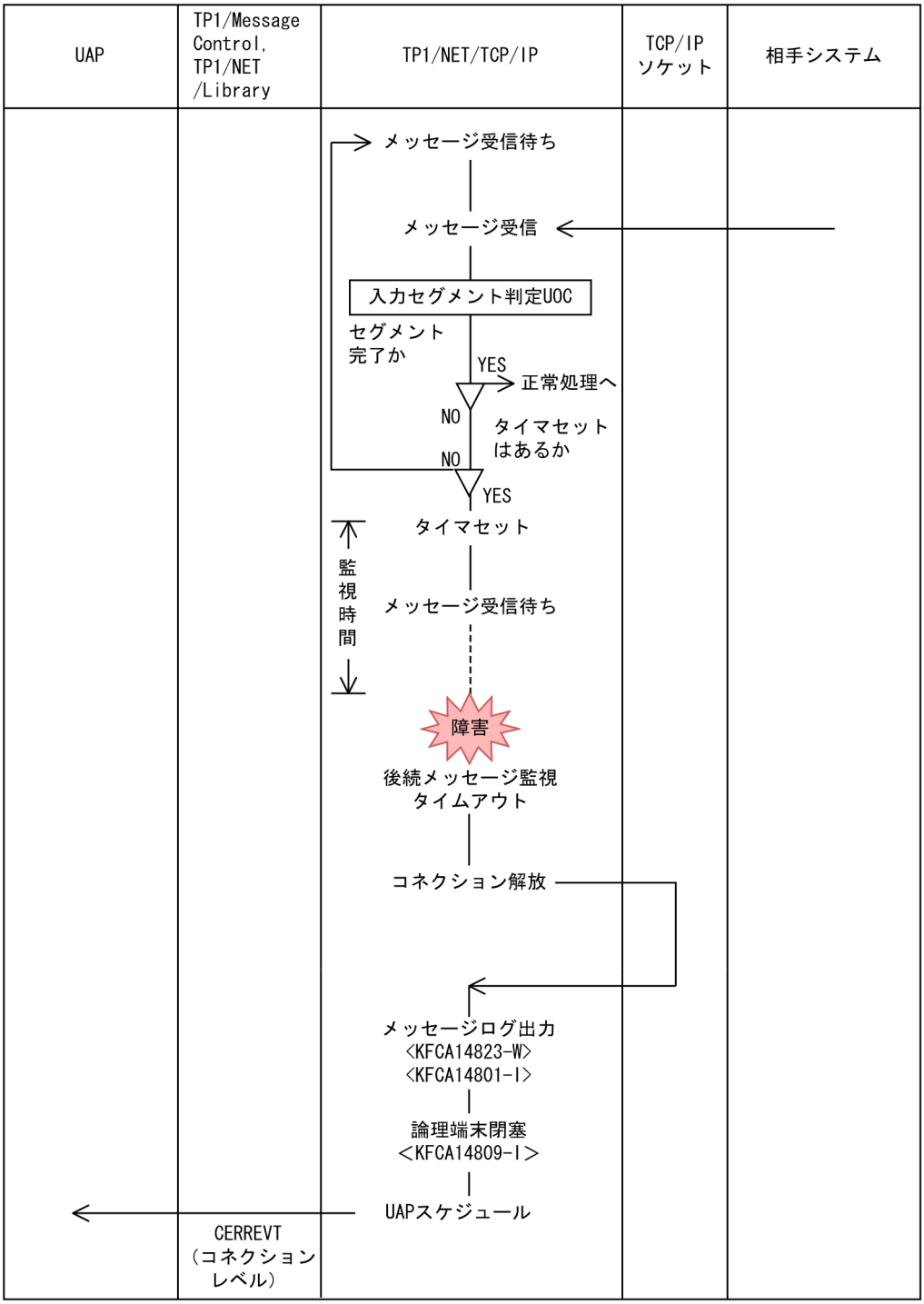
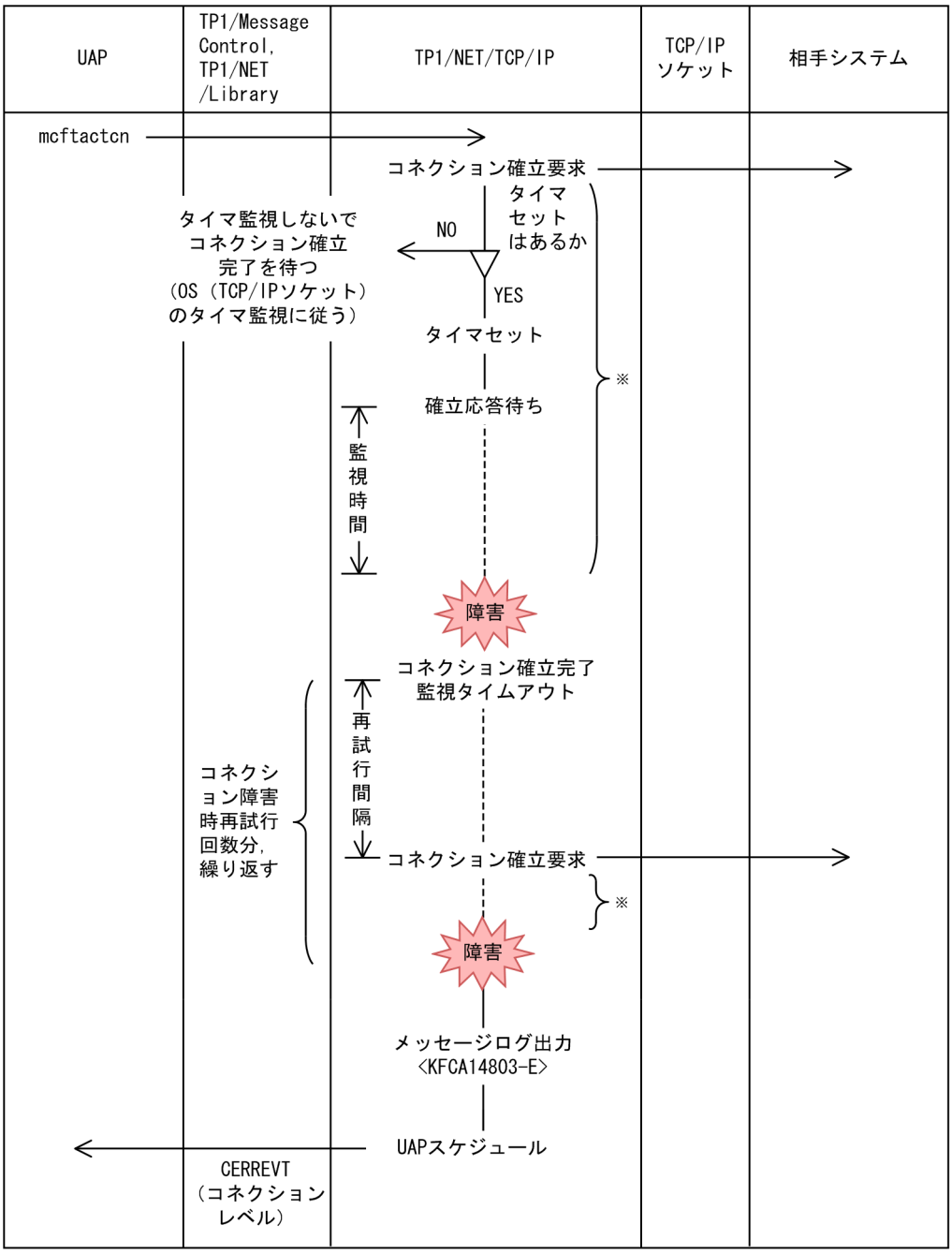


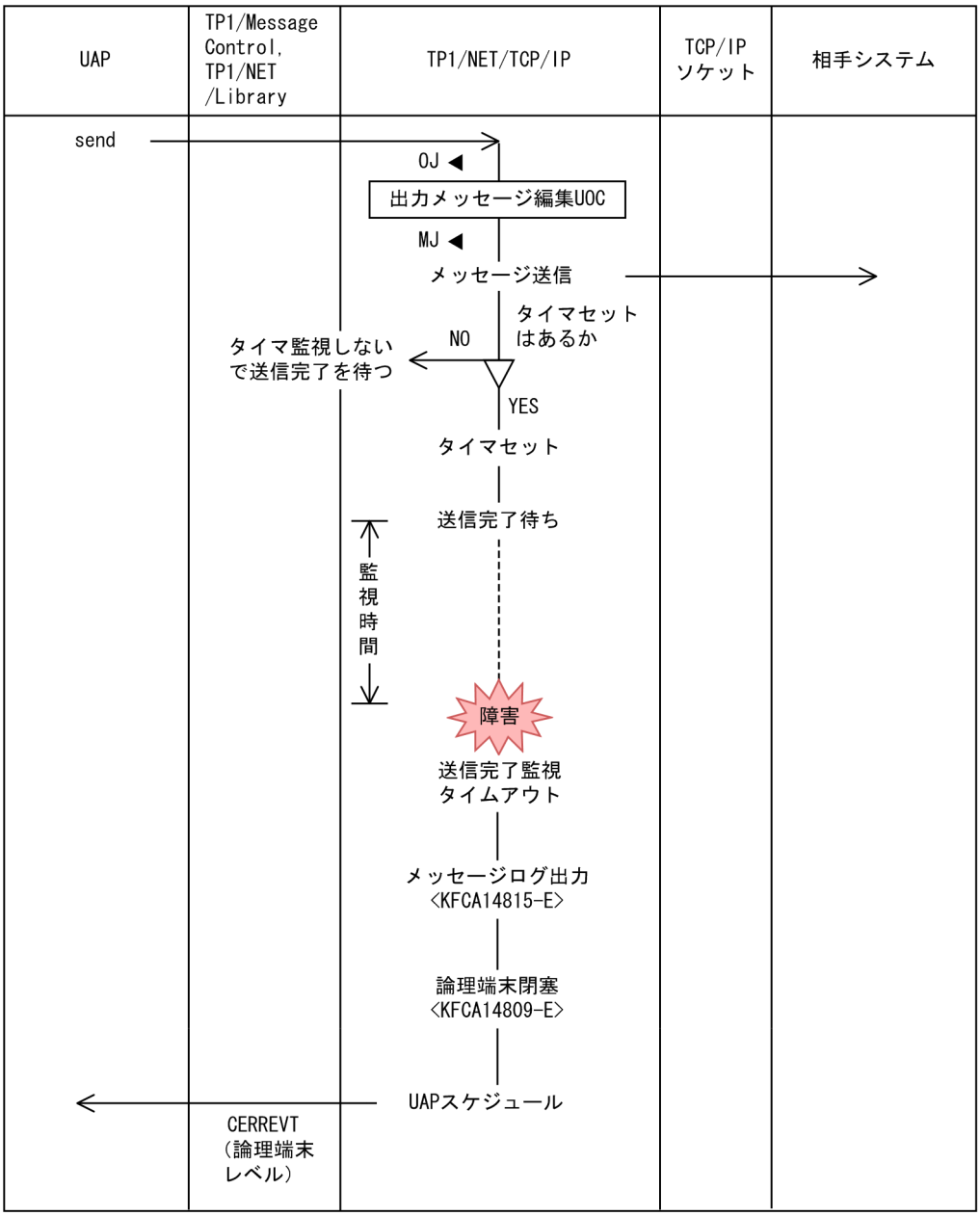
図 E-8　コネクション確立監視時間タイムアウト時の処理の流れ



注※

同じ処理を行います。

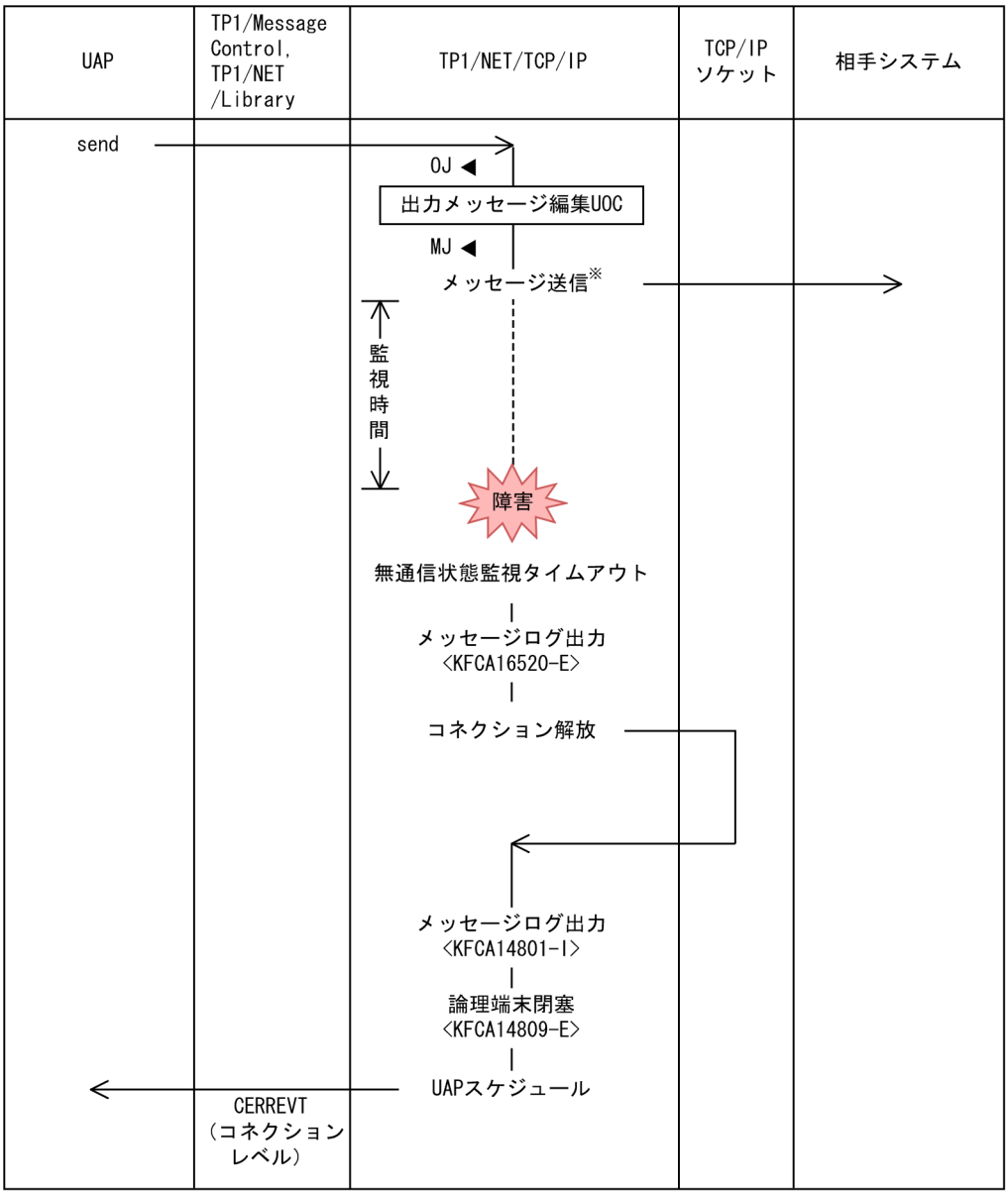
図 E-9　送信完了監視時間タイムアウト時の処理の流れ



（凡例）　OJ ◀: メッセージ出力ジャーナル取得

MJ ◀: メッセージジャーナル取得

図 E-10　無通信状態監視時間タイムアウト時の処理の流れ



（凡例）　OJ ◀：メッセージ出力ジャーナル取得

MJ ◀：メッセージジャーナル取得

注※

論理端末の閉塞解除，およびメッセージ受信でも同様に状態監視を開始します。

付録 F　送受信メッセージの衝突時の処理の流れ（メッセージ送達確認機能使用時）

ここでは，DCCM とのメッセージの衝突，および任意の相手システムとのメッセージ衝突の，処理の流れについて説明します。

付録 F.1　DCCM とのメッセージ衝突

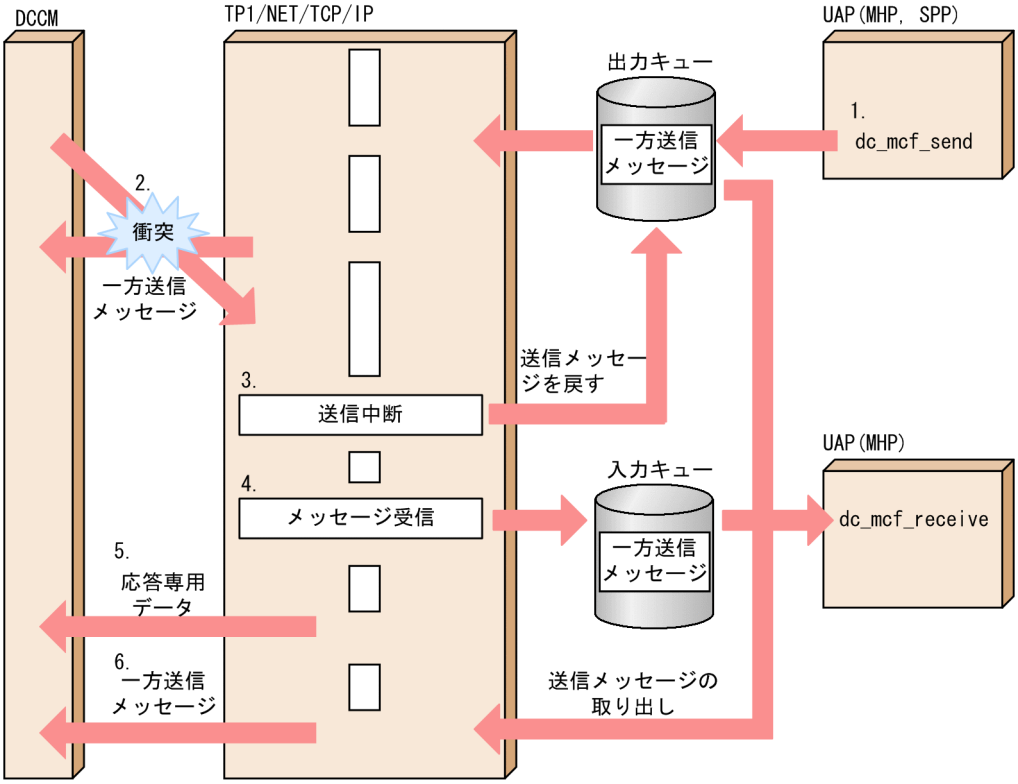
TP1/NET/TCP/IP のメッセージ送信と，DCCM のメッセージ送信が衝突するときの処理の流れを，次に示すあらかじめ決められた定義の対応ごとに説明します。

1. TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm2s（DCCMII/TCP の定義が TERMINAL CONTENT=MASTER）の場合
2. TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm2m（DCCMII/TCP の定義が TERMINAL CONTENT=SLAVE）の場合
3. TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm3s（DCCM3/TCP の定義が TERMINAL CONTENT=MASTER）の場合
4. TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm3m（DCCM3/TCP の定義が TERMINAL CONTENT=SLAVE）の場合

(1)　TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm2s，または mcftalccn -u delichk = dccm3s の場合

TP1/NET/TCP/IP の送信処理を中断し，メッセージ受信処理を継続します。メッセージ受信処理完了後，中断したメッセージを再送します。この場合の処理の流れを次の図に示します。

図 F-1　メッセージ衝突時の処理の流れ（TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm2s，または mcftalccn -u delichk = dccm3s の場合）

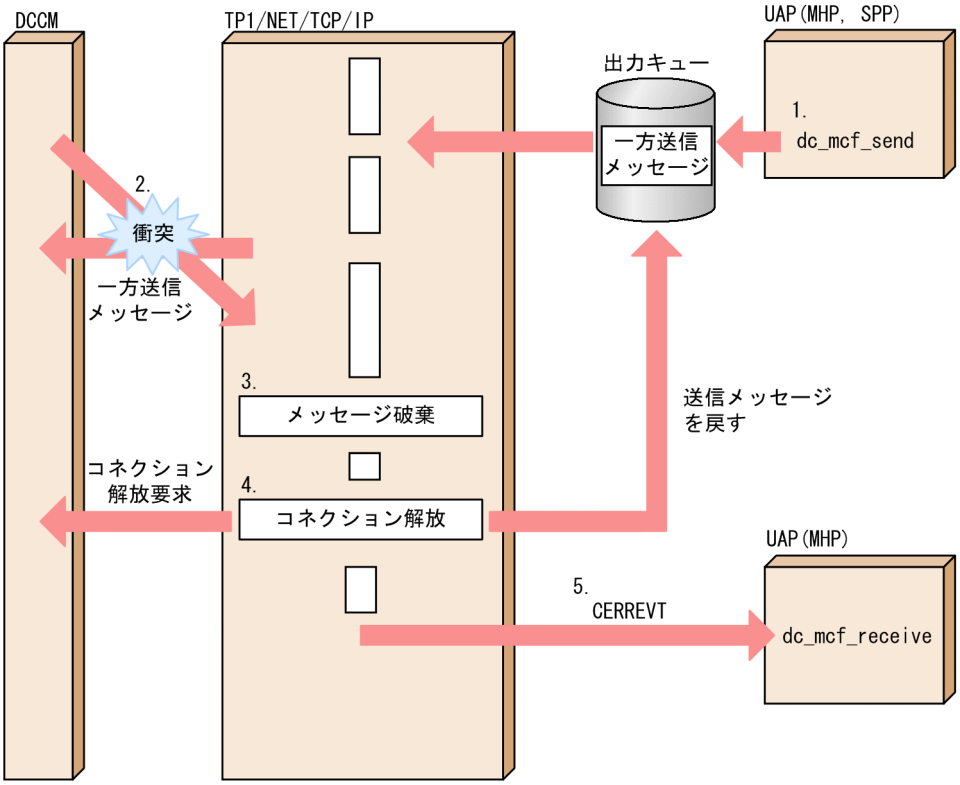


1. UAP は DCCM への一方送信メッセージを送信します。
2. DCCM へ一方送信メッセージを送信する時，DCCM から UAP への一方送信メッセージと衝突しました。
3. 送信メッセージを出力キューに戻し，UAP から DCCM への一方送信メッセージ送信処理を中断します。
4. DCCM から UAP への一方送信メッセージを受信し，UAP へ通知します。
5. DCCM へ応答専用データを送信します。
6. 出力キューから送信メッセージを取り出し，DCCM へ一方送信します。

(2)　TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm2m の場合

TP1/NET/TCP/IP は受信メッセージを破棄し，コネクションを解放し，コネクションレベルの CERREVT を起動します。このとき，送信メッセージは出力キューに残ります。この場合の処理の流れを次の図に示します。

図 F-2　メッセージ衝突時の処理の流れ（TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm2m の場合）

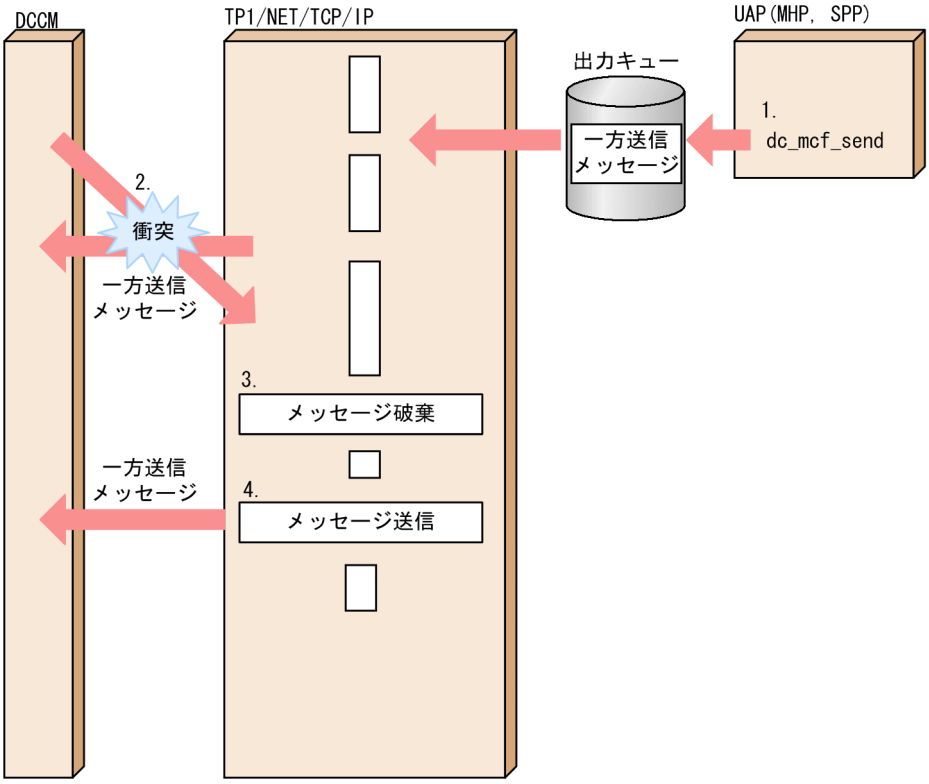


1. UAP は DCCM への一方送信メッセージを送信します。
2. DCCM へ一方送信メッセージを送信する時，DCCM から UAP への一方送信メッセージと衝突しました。
3. DCCM から受信したメッセージを破棄します。
4. 送信メッセージを出力キューに戻し，コネクションを解放します。
5. UAP に CERREVT を通知します。

(3)　TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm3m の場合

TP1/NET/TCP/IP は受信したメッセージを破棄し，メッセージ送信処理を継続します。この場合の処理の流れを次の図に示します。

図 F-3　メッセージ衝突時の処理の流れ（TP1/NET/TCP/IP の定義が mcftalccn -u delichk = dccm3m の場合）



1. UAP は DCCM への一方送信メッセージを送信します。
2. DCCM へ一方送信メッセージを送信する時，DCCM から UAP への一方送信メッセージと衝突しました。
3. DCCM から受信したメッセージを破棄します。
4. メッセージ送信処理を継続します。

付録 F.2　任意の相手システムとのメッセージ衝突

TP1/NET/TCP/IP のメッセージ送信と，任意の相手システムからのメッセージ送信が衝突した場合，TP1/NET/TCP/IP は受信メッセージ判定 UOC の指定内容に従って動作します。UOC で，受信したメッセージの種別，および送信処理中のメッセージの処理内容を指定し，送受信メッセージが衝突した場合の TP1/NET/TCP/IP の動作を決定してください。メッセージの処理内容については，「5.1.12(5)　ユーザが値を設定する項目」を参照してください。

送受信メッセージ衝突時の処理の流れを，次に示す受信メッセージ判定 UOC の設定の例ごとに説明します。

1. メッセージ種別が一方送信メッセージ，送達確認メッセージ送信要，送信処理継続の設定の場合
2. メッセージ種別が一方送信メッセージ，送達確認メッセージ送信要，送信処理再送の設定の場合
3. メッセージ種別が一方送信メッセージ，送達確認メッセージ送信要，送信処理停止の設定の場合

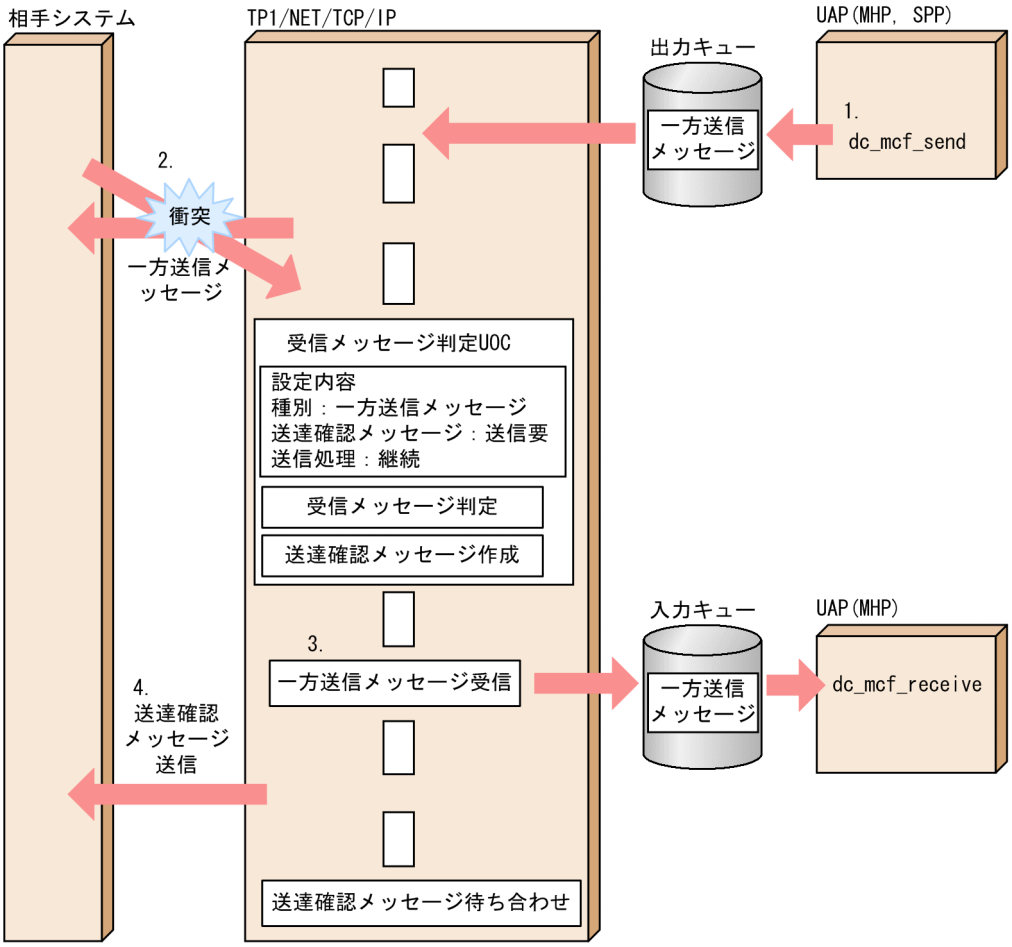
4. メッセージ種別が破棄メッセージ，送信処理継続の設定の場合

5. メッセージ種別がコネクション解放の設定の場合

(1) メッセージ種別が一方送信メッセージ，送達確認メッセージ送信要，送信処理継続の設定の場合

送信処理を継続します。UAP に受信メッセージを通知したあと，相手システムに送達確認メッセージを送信します。この場合の処理の流れを次の図に示します。

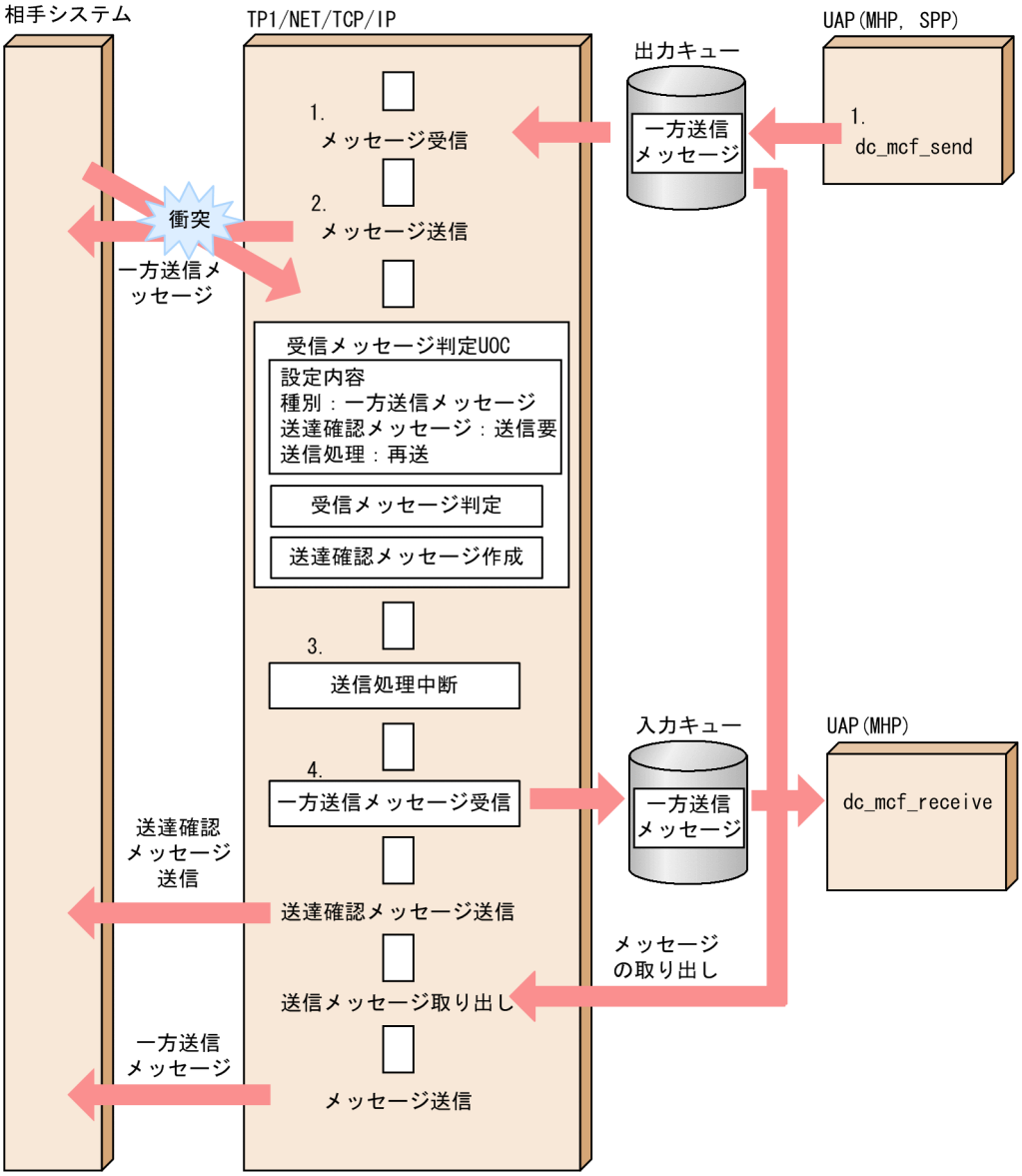
図 F-4 メッセージ衝突時の処理の流れ（メッセージ種別が一方送信メッセージ，送達確認メッセージ送信要，送信処理継続の設定の場合）



(2) メッセージ種別が一方送信メッセージ，送達確認メッセージ送信要，送信処理再送の設定の場合

UAP からの送信処理を一時的に中断します。相手システムからの一方送信メッセージ受信処理完了後，相手システムに UAP からの一方送信メッセージを再送します。この場合の処理の流れを次の図に示します。

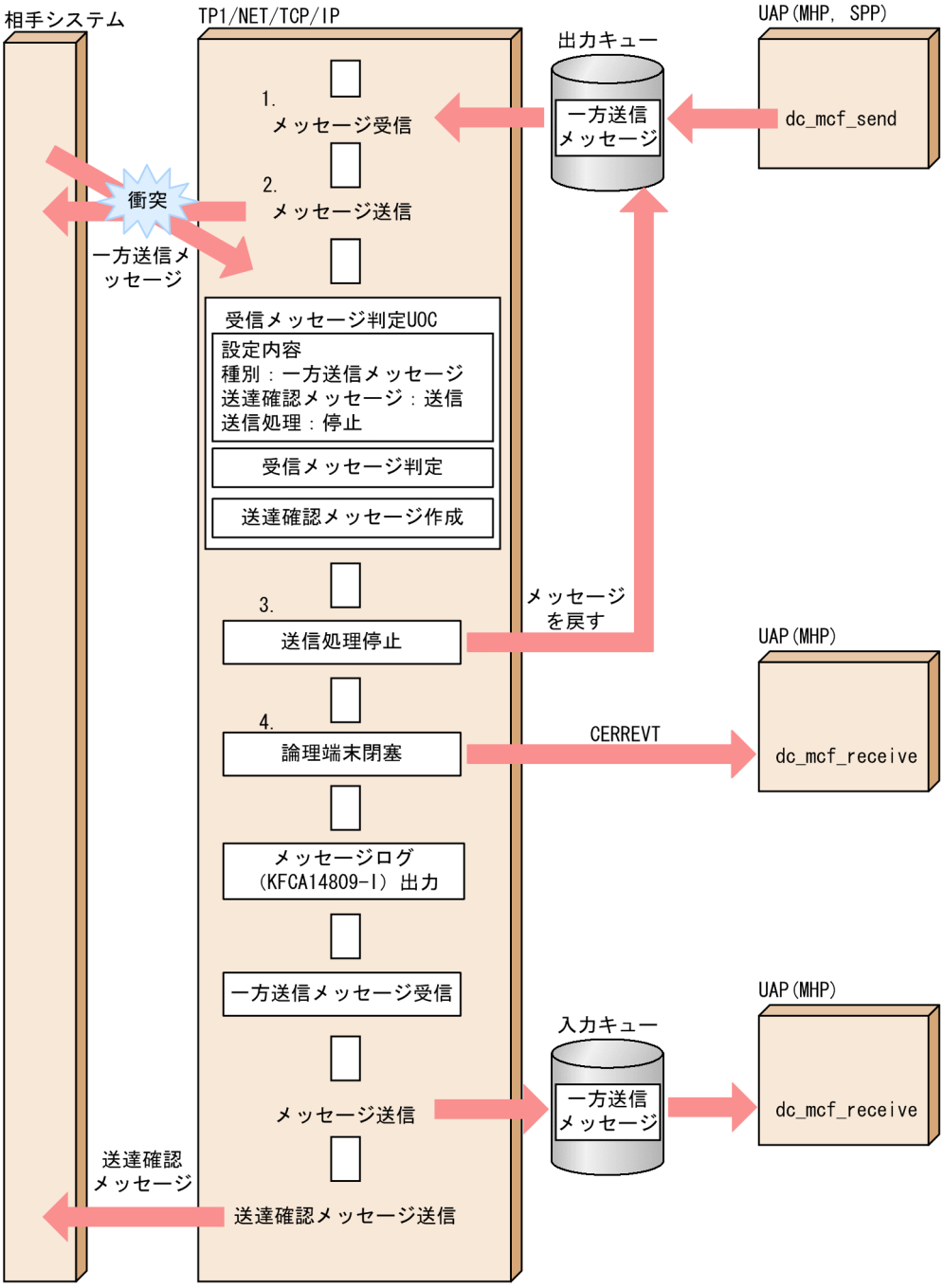
図 F-5　メッセージ衝突時の処理の流れ（メッセージ種別が一方送信メッセージ，送達確認メッセージ送信要，送信処理再送の設定の場合）



(3)　メッセージ種別が一方送信メッセージ，送達確認メッセージ送信要，送信処理停止の設定の場合

論理端末を閉塞して送信処理を停止し，受信処理を続行します。この場合の処理の流れを次の図に示します。

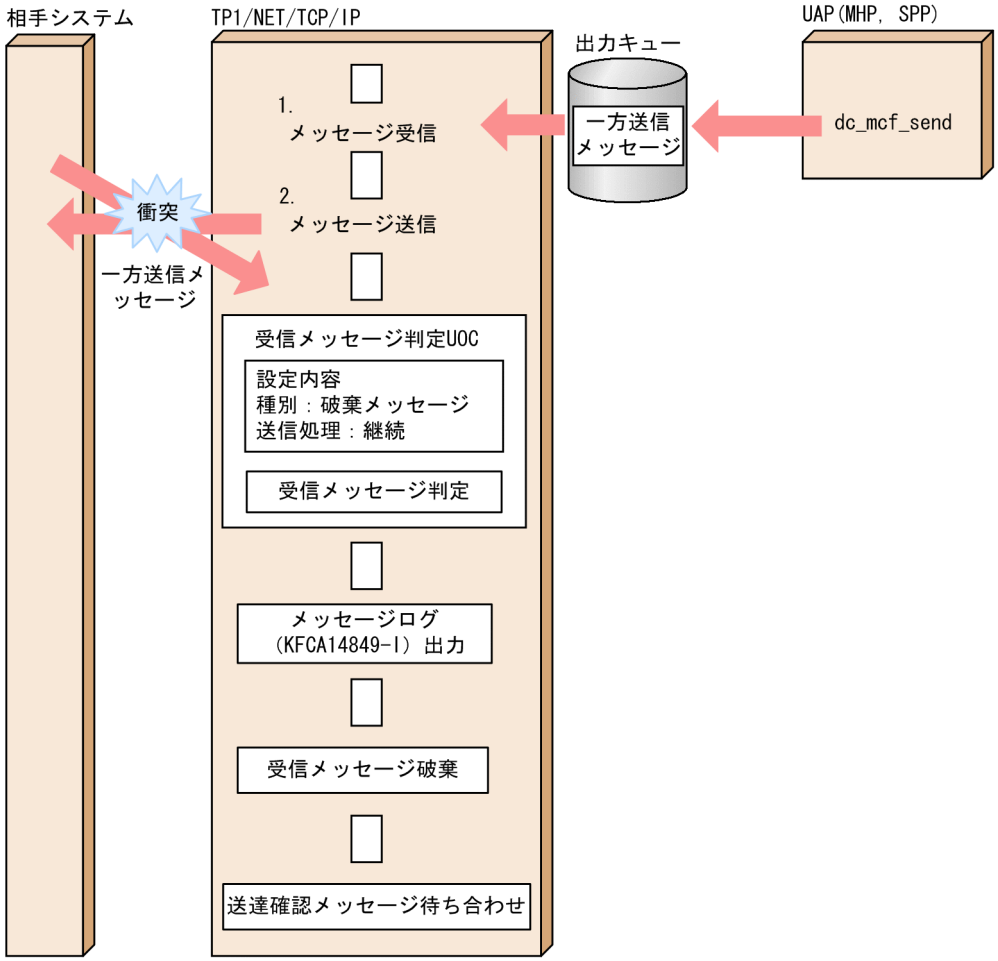
図 F-6　メッセージ衝突時の処理の流れ（メッセージ種別が一方送信メッセージ，送達確認メッセージ送信要，送信処理停止の設定の場合）



(4)　メッセージ種別が破棄メッセージ，送信処理継続の設定の場合

受信メッセージを破棄し，送信処理を継続します。この場合の処理の流れを次の図に示します。

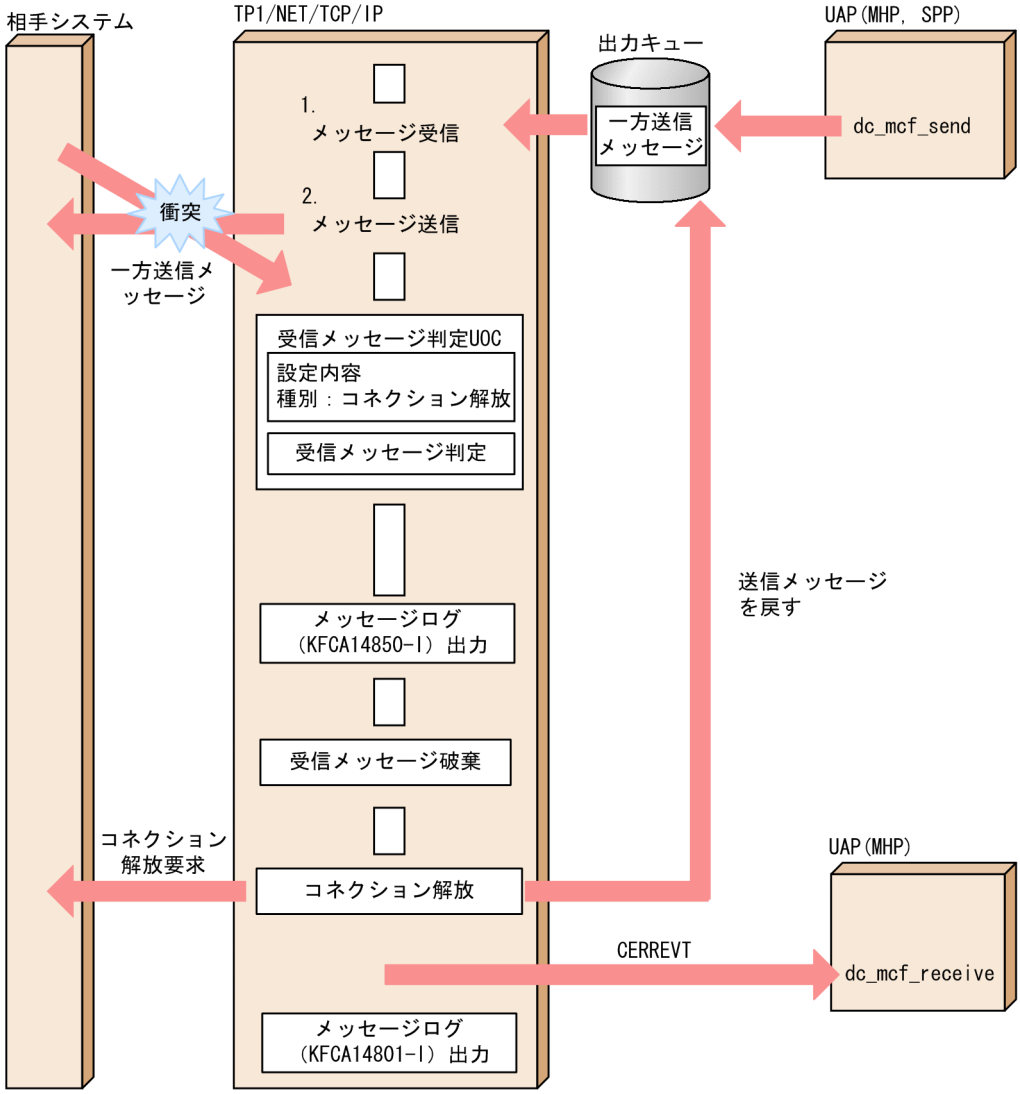
図 F-7　メッセージ衝突時の処理の流れ（メッセージ種別が破棄メッセージ，送信処理継続の設定の場合）



(5)　メッセージ種別がコネクション解放の設定の場合

受信メッセージを破棄し，コネクション解放します。この場合の処理の流れを次の図に示します。

図 F-8　メッセージ衝突時の処理の流れ（メッセージ種別がコネクション解放の設定の場合）



付録 G　ソケット関数の処理の流れ

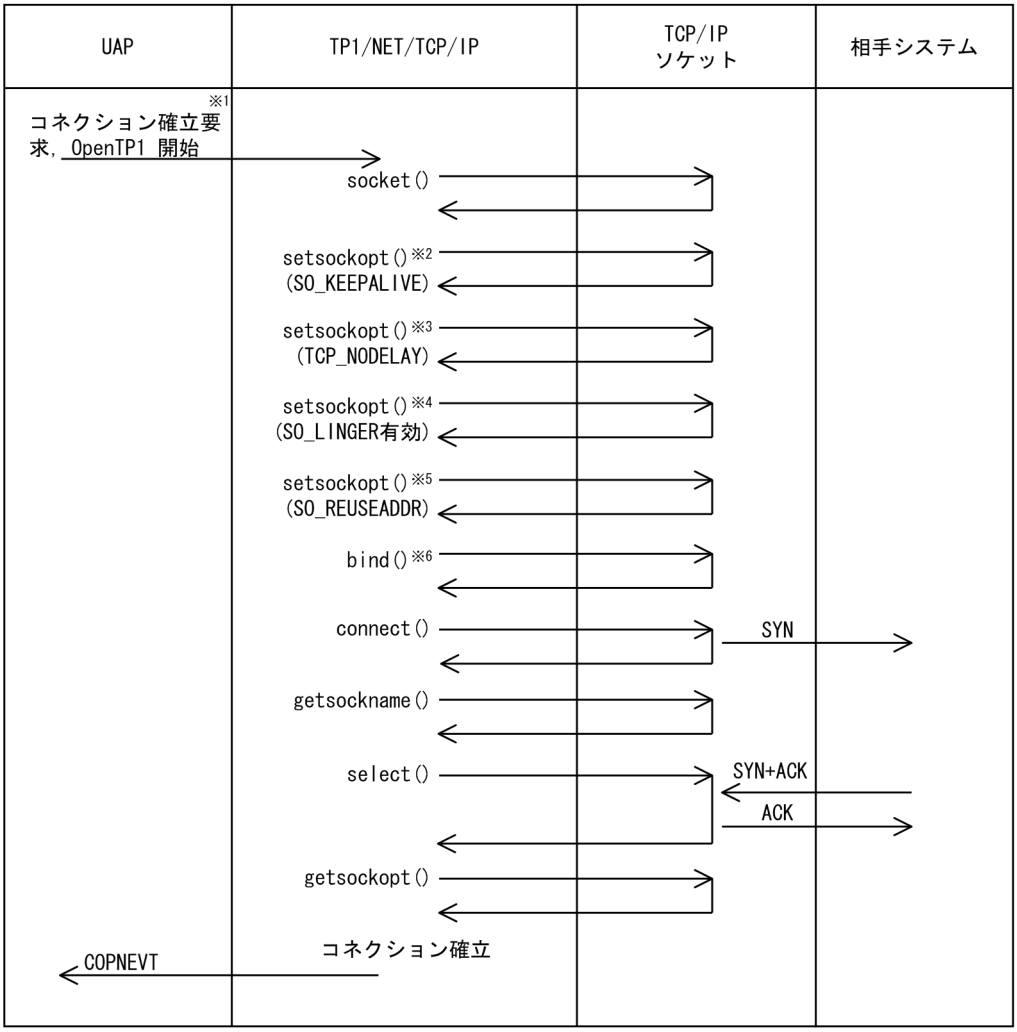
ここでは，次の場合に TP1/NET/TCP/IP が発行するソケット関数の処理の流れを示します。

- クライアント型コネクションの確立
- クライアント型コネクションの確立障害（相手システムのサービス未起動の場合）
- クライアント型コネクションの確立障害（相手システム未起動またはネットワーク障害）
- サーバ型コネクションの確立
- サーバ型コネクションの確立拒否
- メッセージ送信
- メッセージ送信の再試行
- メッセージ受信
- 自システムからのコネクションの解放（FIN）
- 自システムからのコネクションの解放（RST）
- 相手システムからのコネクションの解放（FIN）
- 相手システムからのコネクションの解放（RST）

TCP/IP ソケットと相手システム間のパケット送受信のタイミングは TP1/NET/TCP/IP が動作する OS の実装に依存します。実装によっては，実際のパケット送受信のタイミングが，図に示したパケット送受信のタイミングと異なる場合があります。

なお，バージョン 5 以前でのソケット関数の処理の流れについては，「付録 B.3　バージョン 5 以前でのソケット関数の処理の流れ」を参照してください。

図 G-1　クライアント型コネクションの確立



(凡例)

ACK：確認応答フラグ

SYN：コネクション確立要求フラグ

注※1

コネクション確立要求は，運用コマンド（mcftactcn）の入力，または API（dc_mcf_tactcn 関数もしくは CBLDCMCF('TACTCN△△')）の発行で行います。

注※2

キープアライブを使用する場合（コネクション定義（mcftalccn -k）の keepalive オペランドに yes を指定），発行します。

注※3

TCP_NODELAY を使用する場合（コネクション定義（mcftalccn -k）の nodelay オペランドに yes を指定），発行します。

注※4

障害などによるコネクション解放時に RST パケットを送信する場合（コネクション定義（mcftalccn -f）の cnrelease オペランドに rst を指定），発行します。

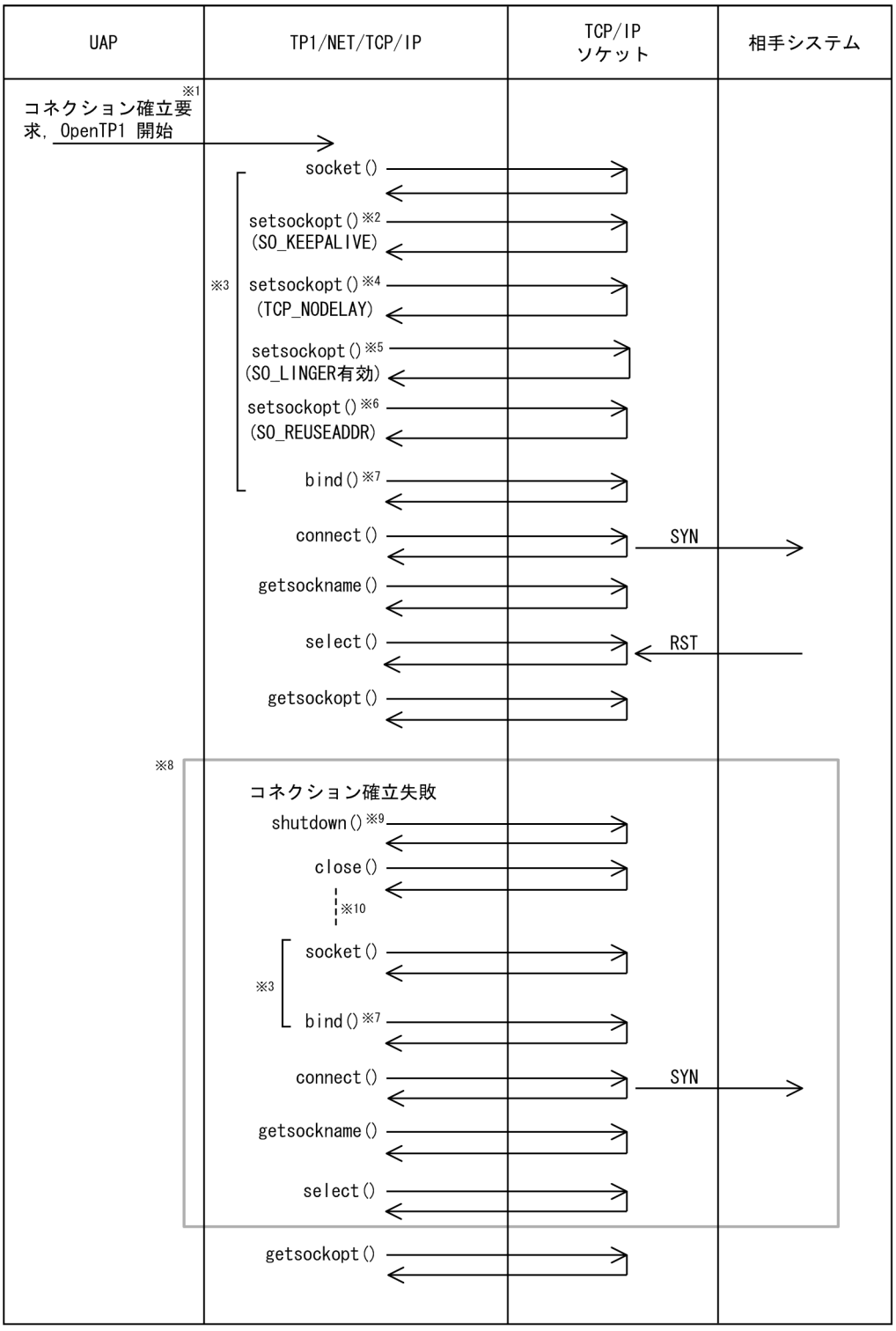
注※5

自システムのポート番号を指定した場合，発行します。

注※6

自システムのホスト名，IP アドレスまたはポート番号を指定した場合，発行します。

図 G-2　クライアント型コネクションの確立障害（相手システムのサービス未起動の場合）



（凡例）

SYN：コネクション確立要求フラグ

RST：コネクション強制切断フラグ

注※1

コネクション確立要求は，運用コマンド（mcftactcn）の入力，またはAPI（dc_mcf_tactcn関数もしくはCBLDCMCF('TACTCN△△')）の発行で行います。

注※2

キープアライブを使用する場合（コネクション定義（mcftalccn -k）のkeepaliveオペランドにyesを指定），発行します。

注※3

同じ処理を行います。

注※4

TCP_NODELAYを使用する場合（コネクション定義（mcftalccn -k）のnodelayオペランドにyesを指定），発行します。

注※5

障害などによるコネクション解放時にRSTパケットを送信する場合（コネクション定義（mcftalccn -f）のcnreleaseオペランドにrstを指定），発行します。

注※6

自システムのポート番号を指定した場合，発行します。

注※7

自システムのホスト名，IPアドレスまたはポート番号を指定した場合，発行します。

注※8

最大でコネクション確立障害時の確立再試行回数（コネクション定義（mcftalccn -b）のbretrycntオペランドの指定値）だけ繰り返します。

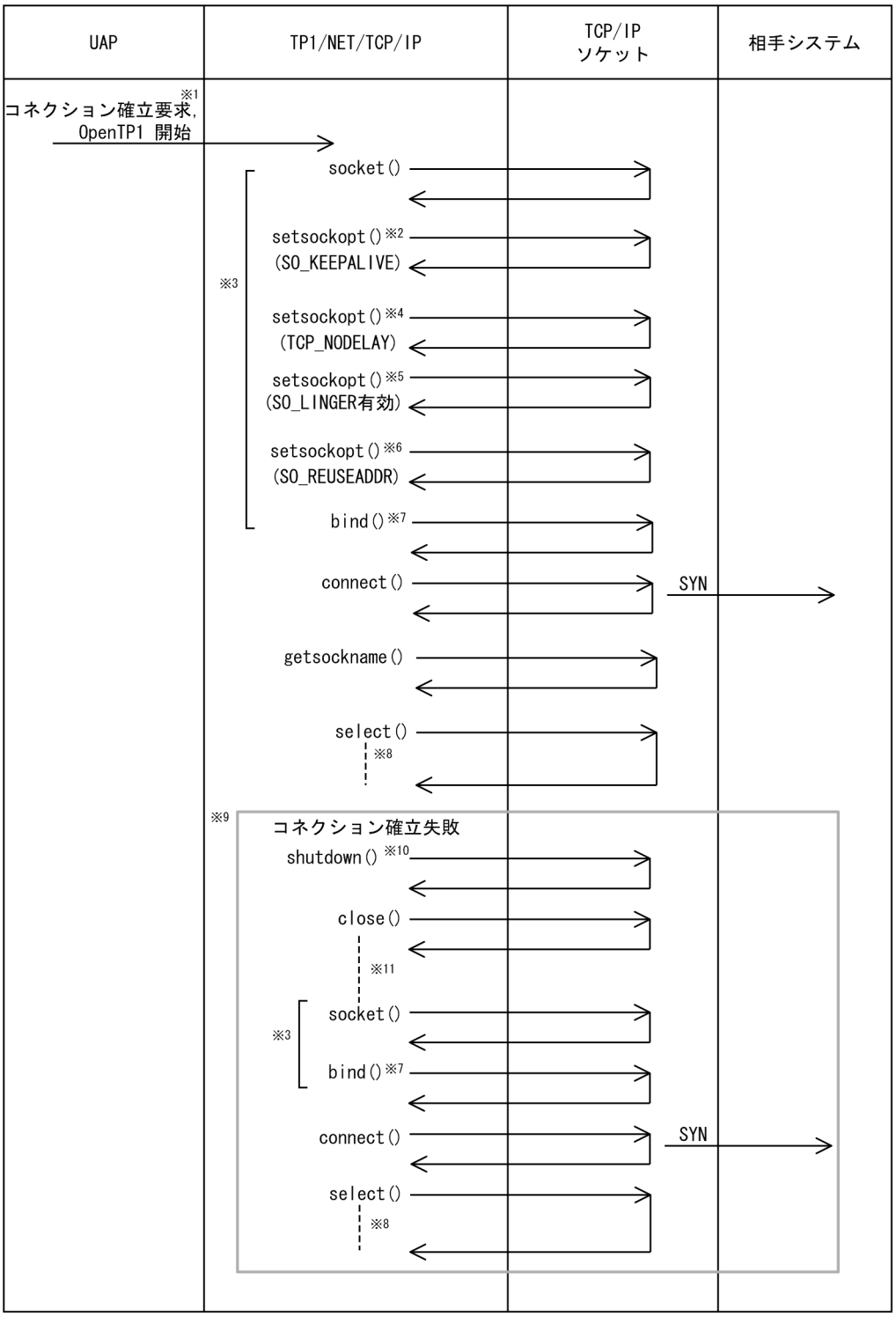
注※9

障害などによるコネクション解放時にRSTパケットを送信する場合（コネクション定義（mcftalccn -f）のcnreleaseオペランドにrstを指定），発行しません。

注※10

コネクション確立障害時の確立再試行間隔（コネクション定義（mcftalccn -b）のbretryintオペランドの指定値）だけ待ち合わせます。

図 G-3　クライアント型コネクションの確立障害（相手システム未起動またはネットワーク障害）



（凡例）

SYN：コネクション確立要求フラグ

注※1

コネクション確立要求は，運用コマンド（mcftactcn）の入力，または API（dc_mcf_tactcn 関数もしくは CBLDCMCF('TACTCN△△')）の発行で行います。

注※2

キープアライブを使用する場合（コネクション定義（mcftalccn -k）の keepalive オペランドに yes を指定），発行します。

注※3

同じ処理を行います。

注※4

TCP_NODELAY を使用する場合（コネクション定義（mcftalccn -k）の nodelay オペランドに yes を指定），発行します。

注※5

障害などによるコネクション解放時に RST パケットを送信する場合（コネクション定義（mcftalccn -f）の cnrelease オペランドに rst を指定），発行します。

注※6

自システムのポート番号を指定した場合，発行します。

注※7

自システムのホスト名，IP アドレスまたはポート番号を指定した場合，発行します。

注※8

コネクション確立監視時間（コネクション定義（mcftalccn -b）の concmptim オペランドの指定値），または TCP/IP ソケットが確立障害を検出するまで待ち合わせます。

注※9

最大でコネクション確立障害時の確立再試行回数（コネクション定義（mcftalccn -b）の bretrycnt オペランドの指定値）だけ繰り返します。

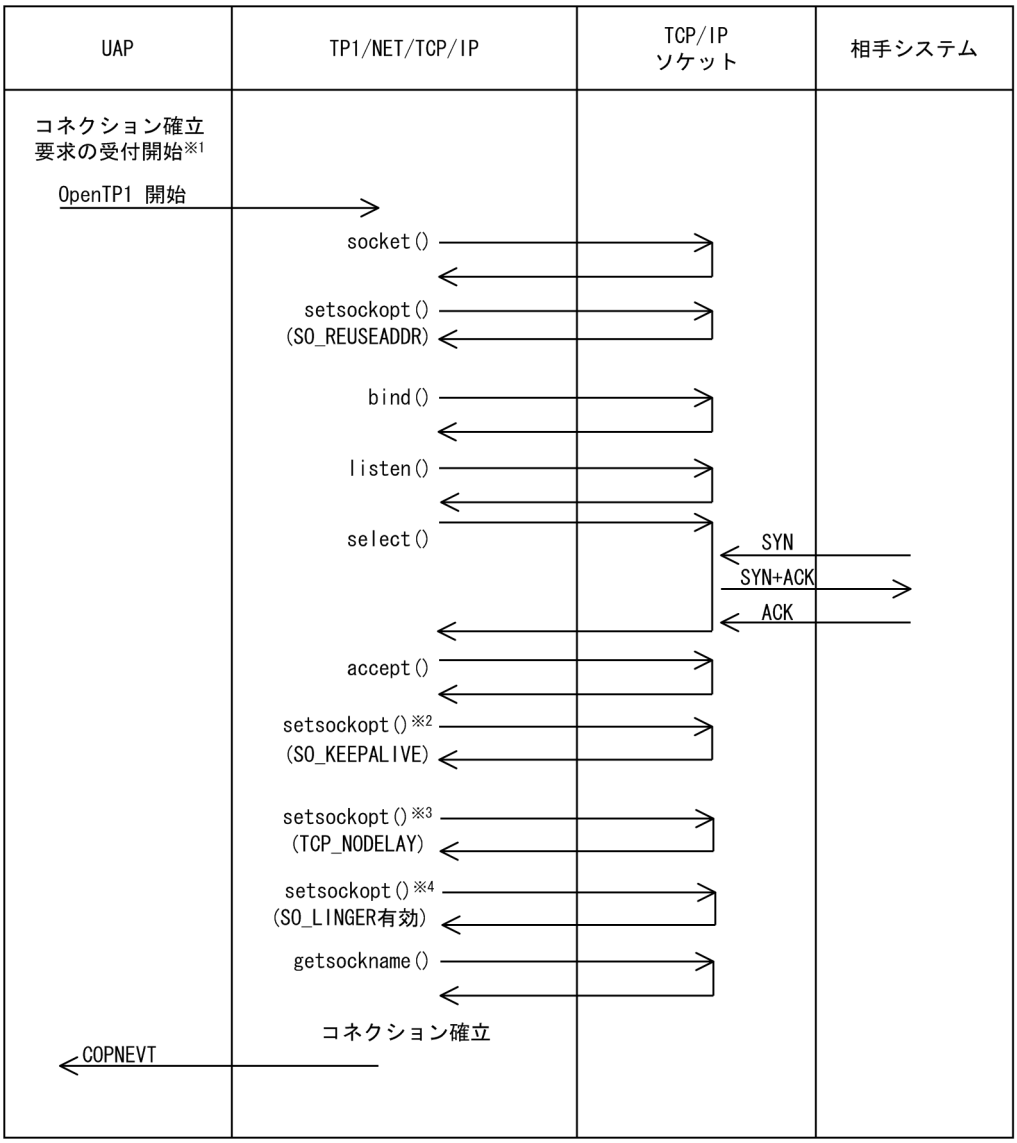
注※10

障害などによるコネクション解放時に RST パケットを送信する場合（コネクション定義（mcftalccn -f）の cnrelease オペランドに rst を指定），発行しません。

注※11

コネクション確立障害時の確立再試行間隔（コネクション定義（mcftalccn -b）の bretryint オペランドの指定値）だけ待ち合わせます。

図 G-4　サーバ型コネクションの確立



(凡例)

ACK：確認応答フラグ

SYN：コネクション確立要求フラグ

注※1

コネクション確立要求の受付開始は，運用コマンド（mcftonln）の入力，または API（dc_mcf_tonln 関数もしくは CBLDCMCF('TONLN△△△')）の発行で行います。

注※2

キープアライブを使用する場合（コネクション定義（mcftalccn -k）の keepalive オペランドに yes を指定），発行します。

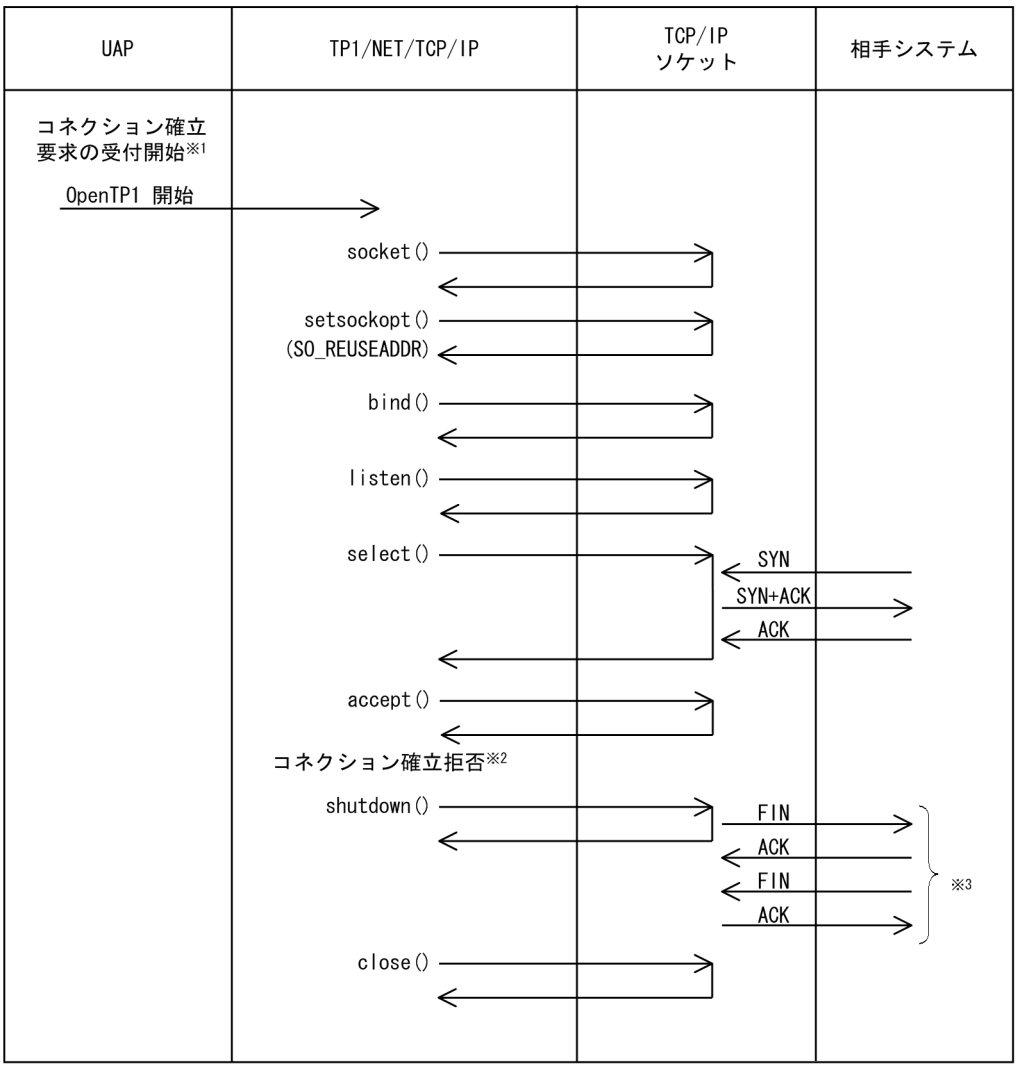
注※3

TCP_NODELAY を使用する場合（コネクション定義（mcftalccn -k）の nodelay オペランドに yes を指定），発行します。

注※4

障害などによるコネクション解放時に RST パケットを送信する場合（コネクション定義（mcftalccn -f）の cnrelease オペランドに rst を指定），発行します。

図 G-5　サーバ型コネクションの確立拒否



(凡例)

ACK：確認応答フラグ

FIN：通信終了フラグ

SYN：コネクション確立要求フラグ

注※1

コネクション確立要求の受付開始は，運用コマンド（mcftonln）の入力，または API（dc_mcf_tonln 関数もしくは CBLDCMCF('TONLN△△△')）の発行で行います。

注※2

次の条件のどれかに該当する場合，コネクションの確立を拒否します。

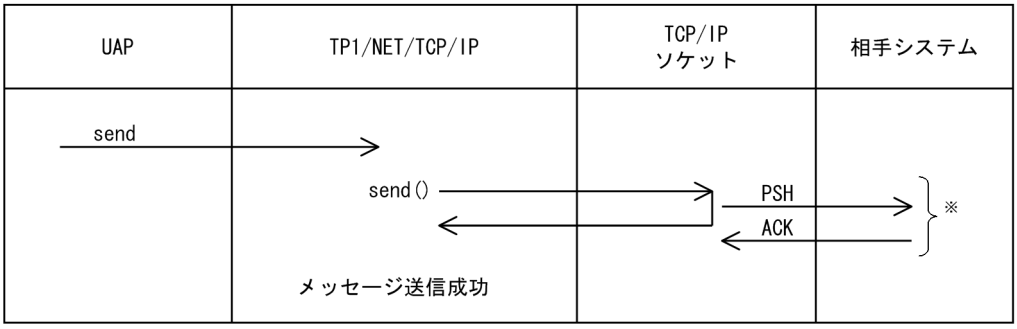
- 未定義の相手システムからの確立要求を受信した場合

- 空きコネクション（ポートフリーまたは相手アドレスチェックの抑止を指定した未確立コネクション）がない状態で確立要求を受信した場合
- コネクション確立 UOC で確立要求を拒否した場合

注※3

パケットの送受信は，TP1/NET/TCP/IP と非同期に行われます。TP1/NET/TCP/IP は，相手システムからの ACK および FIN を待ち合わせません。

図 G-6　メッセージ送信



（凡例）

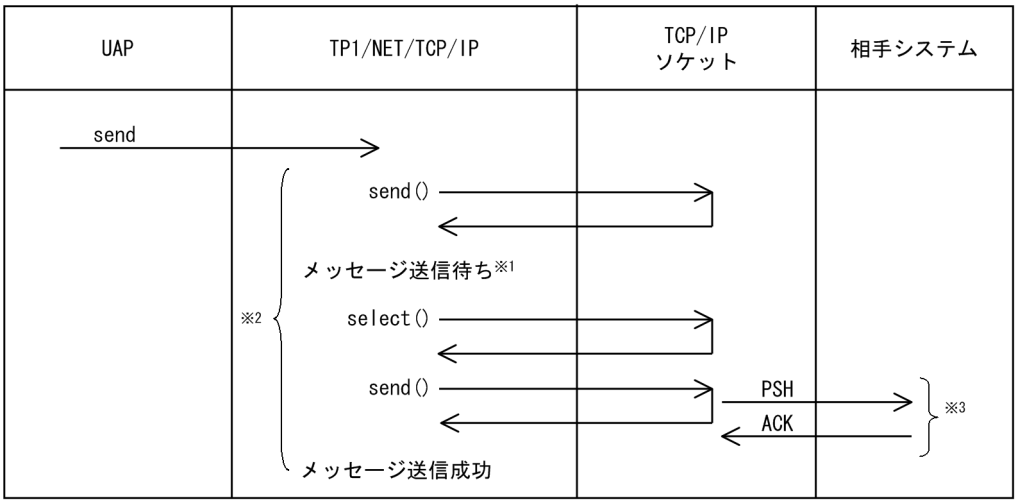
ACK：確認応答フラグ

PSH：プッシュフラグ

注※

パケットの送受信は，TP1/NET/TCP/IP と非同期に行われます。TP1/NET/TCP/IP は，相手システムからの ACK を待ち合わせません。

図 G-7　メッセージ送信の再試行



（凡例）

ACK：確認応答フラグ

PSH：プッシュフラグ

注※1

TCP/IP ソケットの送信バッファが満杯の場合，送信バッファに空き領域ができるまでメッセージ送信を待ち合わせます。

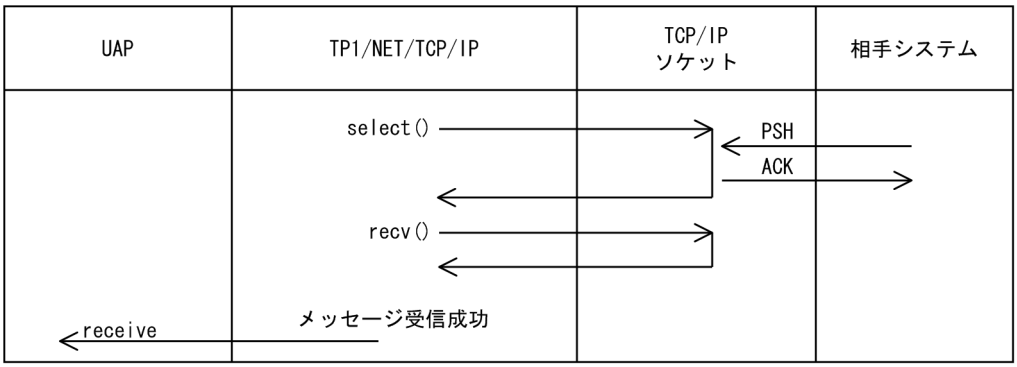
注※2

コネクション定義に指定したメッセージ送信完了監視時間（mcftalccn -b sndcmptim）だけ送信処理を監視します。送達確認機能を使用する場合は，メッセージ送信成功後に相手システムからの送達確認メッセージの受信までを監視します。

注※3

パケットの送受信は，TP1/NET/TCP/IP と非同期に行われます。TP1/NET/TCP/IP は，相手システムからの ACK を待ち合わせません。

図 G-8　メッセージ受信

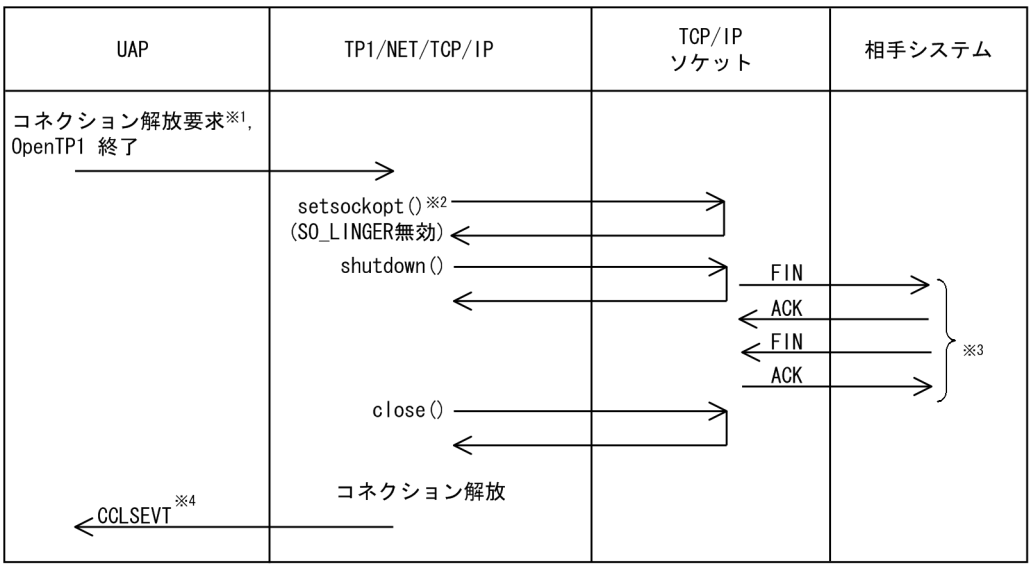


（凡例）

ACK：確認応答フラグ

PSH：プッシュフラグ

図 G-9　自システムからのコネクションの解放（FIN）



（凡例）

ACK：確認応答フラグ

FIN：通信終了フラグ

注※1

コネクション解放要求は，運用コマンド（mcftdctcn）の入力，またはAPI（dc_mcf_tdctcn関数もしくはCBLDCMCF('TDCTCN△△')）の発行で行います。

注※2

障害などによるコネクション解放時にRSTパケットを送信する場合（コネクション定義（mcftalccn -f）のcnreleaseオペランドにrstを指定），発行します。

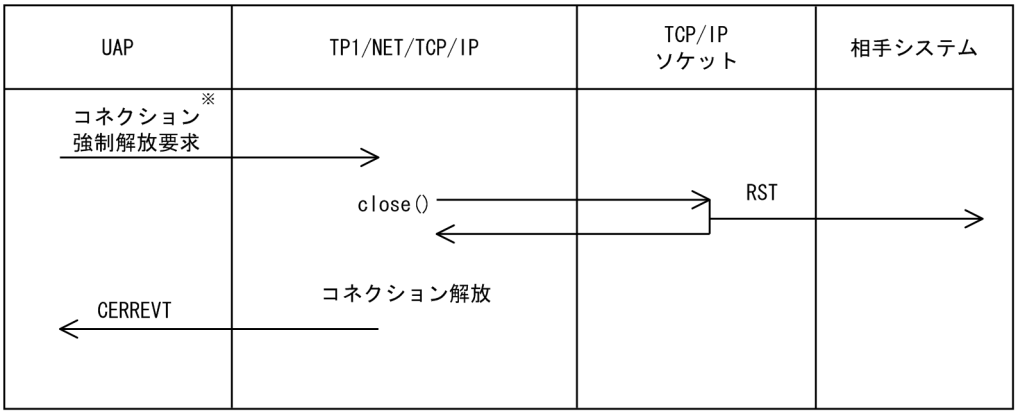
注※3

パケットの送受信は，TP1/NET/TCP/IPと非同期に行われます。TP1/NET/TCP/IPは，相手システムからのACKおよびFINを待ち合わせません。

注※4

OpenTP1終了時のコネクション解放では，CCLSEVTは発行されません。

図 G-10　自システムからのコネクションの解放（RST）



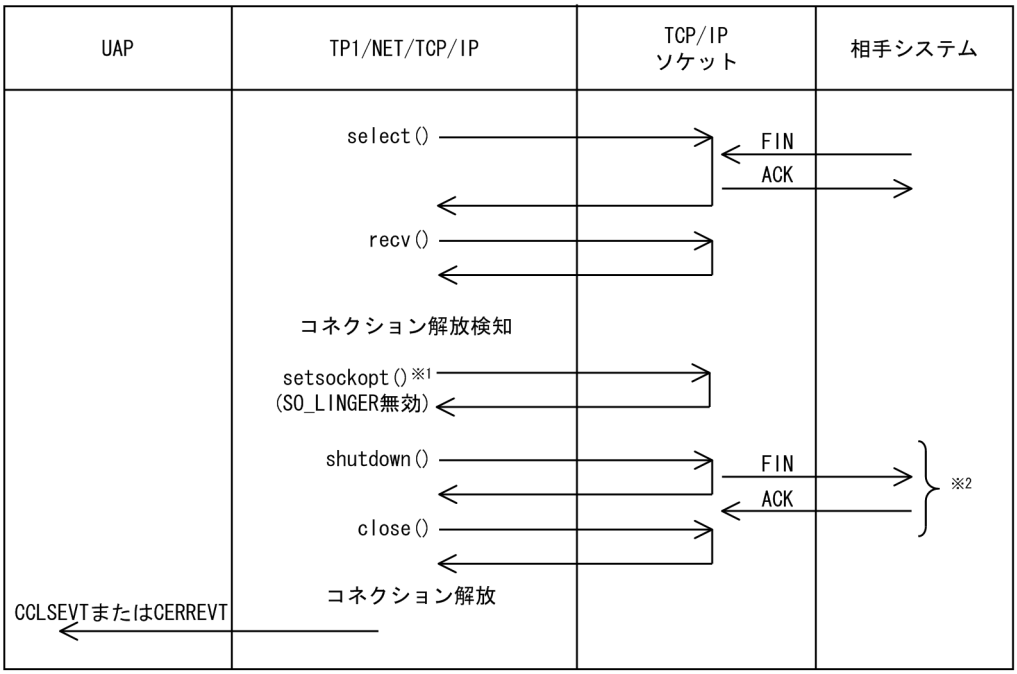
（凡例）

RST：コネクション強制切断フラグ

注※

コネクション強制解放要求は，コネクション定義（mcftalccn -f）のcnreleaseオペランドにrstを指定したときに，運用コマンド（mcftdctcn -f）の入力，またはAPI（action引数にDCMCFFRCを指定したdc_mcf_tdctcn関数もしくはデータ名D1に'1'を指定したCBLDCMCF('TDCTCN△△')）の発行で行います。

図 G-11　相手システムからのコネクションの解放（FIN）



（凡例）

ACK：確認応答フラグ

FIN：通信終了フラグ

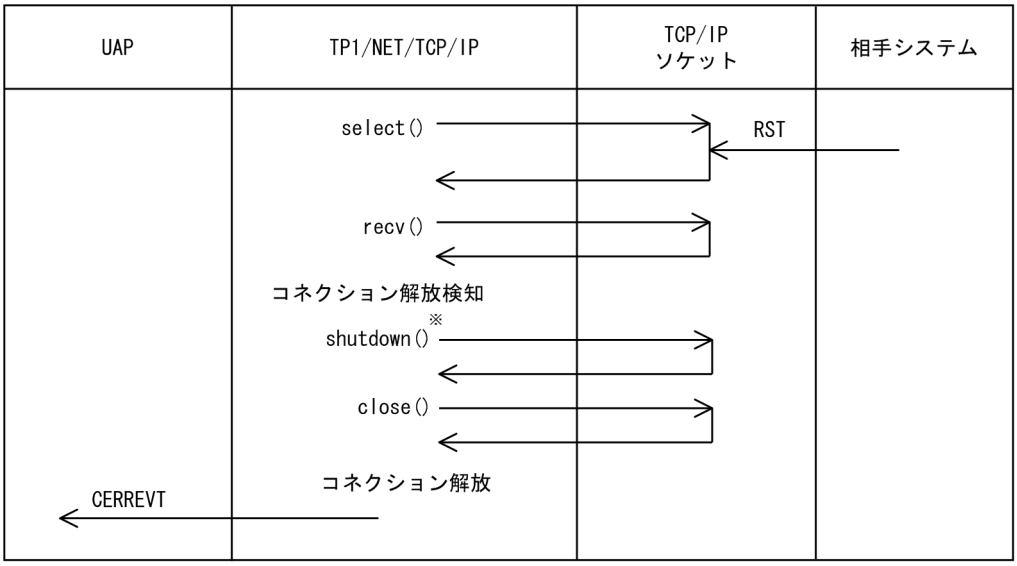
注※1

障害などによるコネクション解放時に RST パケットを送信する場合（コネクション定義（mcftalccn -f）の cnrelease オペランドに rst を指定），発行します。

注※2

パケットの送受信は，TP1/NET/TCP/IP と非同期に行われます。TP1/NET/TCP/IP は，相手システムからの ACK を待ち合わせません。

図 G-12　相手システムからのコネクションの解放（RST）



（凡例）

RST：コネクション強制切断フラグ

注※

障害などによるコネクション解放時に RST パケットを送信する場合（コネクション定義（mcftalccn -f）の cnrelease オペランドに rst を指定），発行しません。

付録 H　MCF 性能検証用トレースの取得

TP1/Message Control を使用したメッセージ送受信での主なイベントで，MCF 識別子などのトレース情報を取得しています。これを MCF 性能検証用トレースと呼びます。

ここでは，MCF 性能検証用トレースの MCF 固有情報の出力情報，取得タイミング，および取得量について説明します。

付録 H.1　MCF 固有情報の出力情報

ここでは，メッセージ送受信時，および UOC 呼び出し時の MCF 性能検証用トレースのダンプ出力情報について説明します。

(1)　メッセージ送受信時

論理端末単位に相手システムと送受信するメッセージの情報を取得します。MCF 固有情報のダンプ出力情報を，次の表に示します。

表 H-1　メッセージ送受信時の MCF 固有情報のダンプ出力情報

イベント ID	オフセット				
	0x0000～0x0003	0x0004～0x0007	0x0008～0x000f	0x0010～0x0017	0x0018～0x001f
0xa000	MCF 識別子	スレッド ID	－	入力元論理端末名	－
0xa001			－	出力先論理端末名	－

（凡例）

－：情報を取得しません。

(2)　UOC 呼び出し時

TP1/NET/TCP/IP で使用する UOC の情報を取得します。MCF 固有情報のダンプ出力情報，および UOC 名称の出力情報を，以降の表に示します。

表 H-2　UOC 呼び出し時の MCF 固有情報のダンプ出力情報

イベント ID	オフセット				
	0x0000～0x0003	0x0004～0x0007	0x0008～0x000f	0x0010～0x0017	0x0018～0x001f
0xa070	MCF 識別子	スレッド ID	－	－	UOC 名称
0xa071			－	－	

（凡例）

－：情報を取得しません。

表 H-3　UOC 名称の出力情報

UOC の種類	UOC 名称出力情報
入力セグメント判定 UOC	"SEGCHK"
入力メッセージ編集 UOC	"MSGRCV"
出力メッセージ編集 UOC	"MSGSEND"
送信メッセージ通番編集 UOC	"SEND_UOC"
コネクション確立 UOC	"ASSCTN"
受信メッセージ判定 UOC	"MSGREP"
受信メッセージ保留判定 UOC	"MSGHLD"

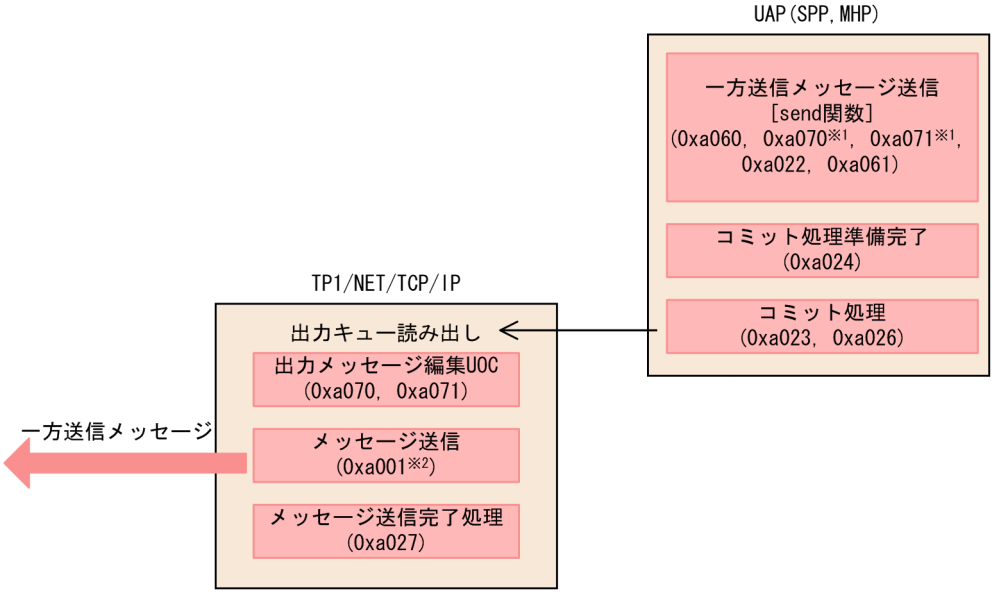
付録 H.2　MCF 性能検証用トレースの取得タイミング

MCF 性能検証用トレースの取得タイミングを，使用する送受信形態別に説明します。

(1)　一方送信メッセージ送信時

一方送信メッセージ送信時の MCF 性能検証用トレースの取得タイミングについて，次の図に示します。

図 H-1　一方送信メッセージ送信時の MCF 性能検証用トレースの取得タイミング（メッセージ送達確認機能を使用しない場合）

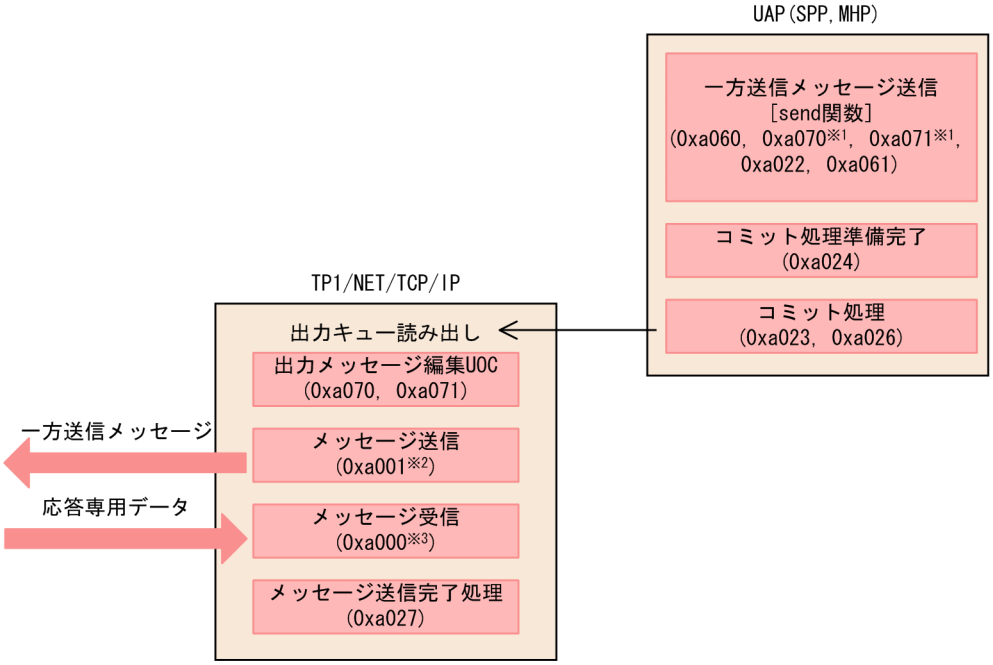


（凡例）
：MCF性能検証用トレースの取得タイミング
（　）：イベントID

注※1
送信メッセージの通番編集UOC使用時に該当します。

注※2
メッセージ送信時にTCP/IPソケットの送信バッファが満杯になった場合，トレースを複数個取得することがあります。

図 H-2　一方送信メッセージ送信時の MCF 性能検証用トレースの取得タイミング（DCCM とのメッセージ送達確認機能を使用する場合）



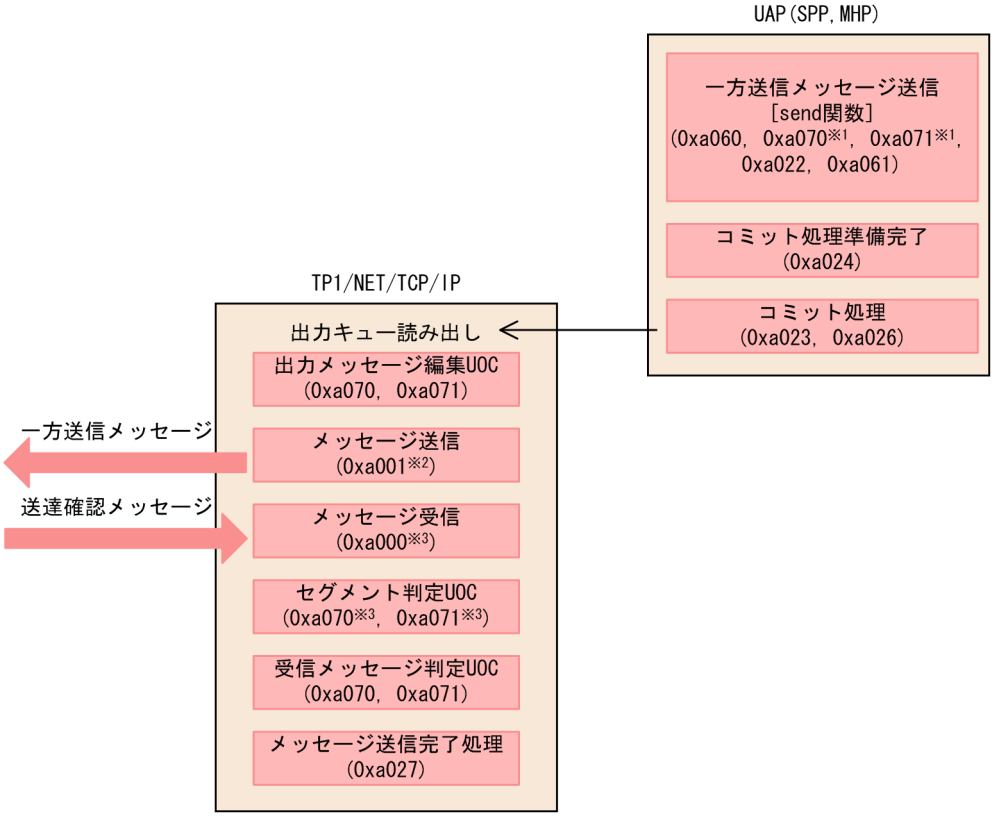
（凡例）
：MCF性能検証用トレースの取得タイミング
（　）：イベントID

注※1
送信メッセージの通番編集UOC使用時に該当します。

注※2
メッセージ送信時にTCP/IPソケットの送信バッファが満杯になった場合，トレースを複数個取得することがあります。

注※3
応答専用データがネットワーク上で分割された場合，トレースを複数個取得することがあります。

図 H-3　一方送信メッセージ送信時の MCF 性能検証用トレースの取得タイミング（任意の相手システムとのメッセージ送達確認機能を使用する場合）



（凡例）
：MCF性能検証用トレースの取得タイミング
（　）：イベントID

注※1
送信メッセージの通番編集UOC使用時に該当します。

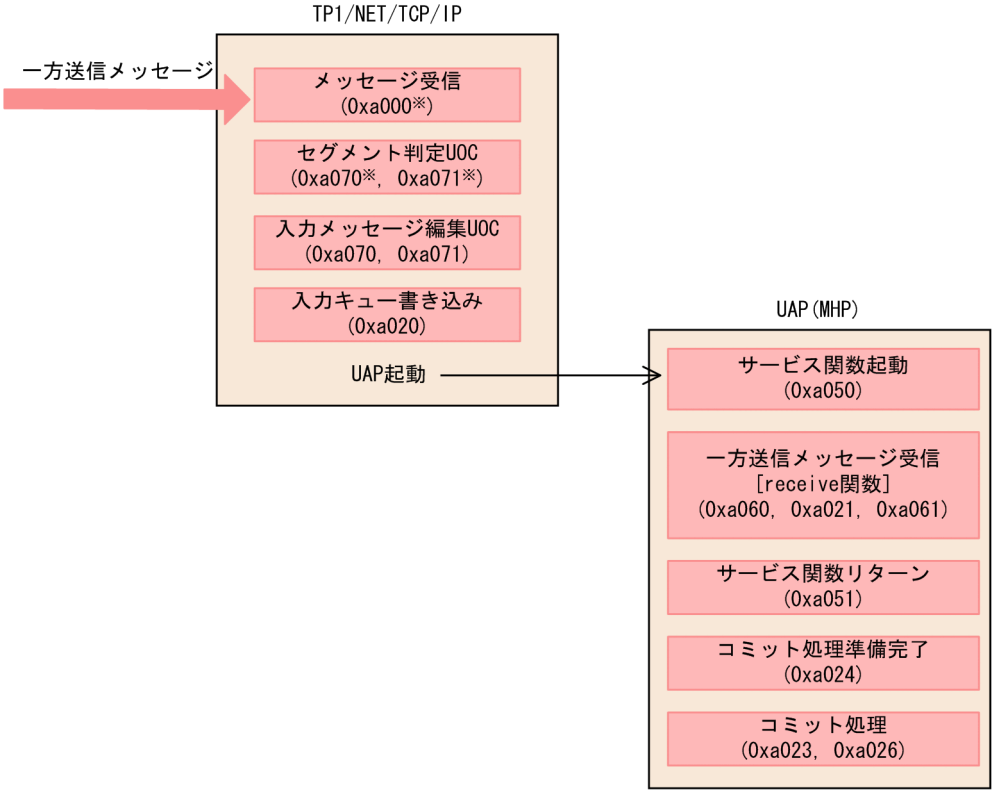
注※2
メッセージ送信時にTCP/IPソケットの送信バッファが満杯になった場合，トレースを複数個取得することがあります。

注※3
送達確認メッセージがネットワーク上で分割された場合，トレースを複数個取得することがあります。

(2)　一方送信メッセージ受信時

一方送信メッセージ受信時の MCF 性能検証用トレースの取得タイミングについて，次の図に示します。

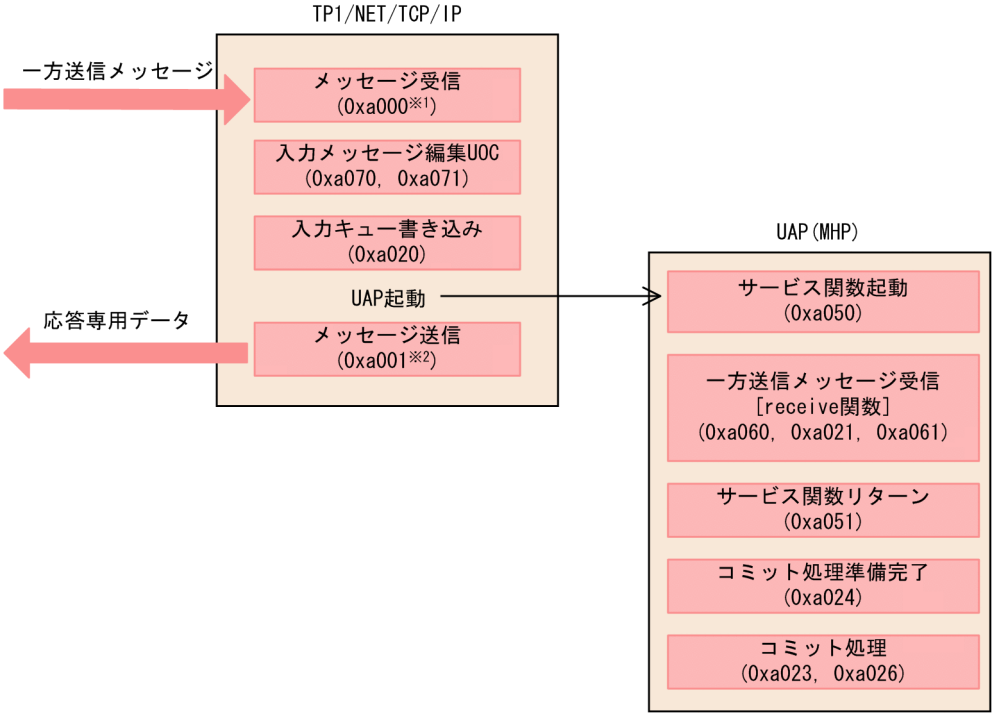
図 H-4　一方送信メッセージ受信時の MCF 性能検証用トレースの取得タイミング（メッセージ送達確認機能を使用しない場合）



（凡例）
：MCF性能検証用トレースの取得タイミング
（　）：イベントID

注※
一方送信メッセージがネットワーク上で分割された場合，トレースを複数個取得することがあります。

図 H-5　一方送信メッセージ受信時の MCF 性能検証用トレースの取得タイミング（DCCM とのメッセージ送達確認機能を使用する場合）

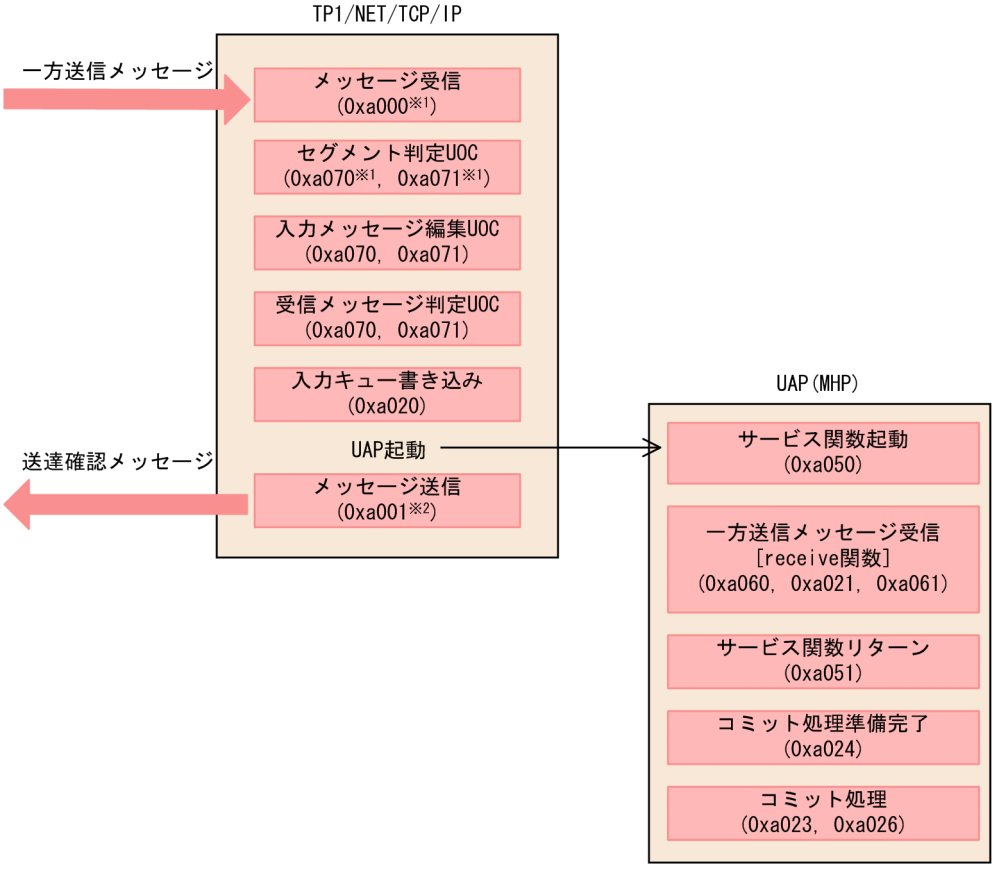


（凡例）
：MCF性能検証用トレースの取得タイミング
（　）：イベントID

注※1
一方送信メッセージがネットワーク上で分割された場合，トレースを複数個取得することがあります。

注※2
メッセージ送信時にTCP/IPソケットの送信バッファが満杯になった場合，トレースを複数個取得することがあります。

図 H-6　一方送信メッセージ受信時の MCF 性能検証用トレースの取得タイミング（任意の相手システムとのメッセージ送達確認機能を使用する場合）



（凡例）
：MCF性能検証用トレースの取得タイミング
（ ）：イベントID

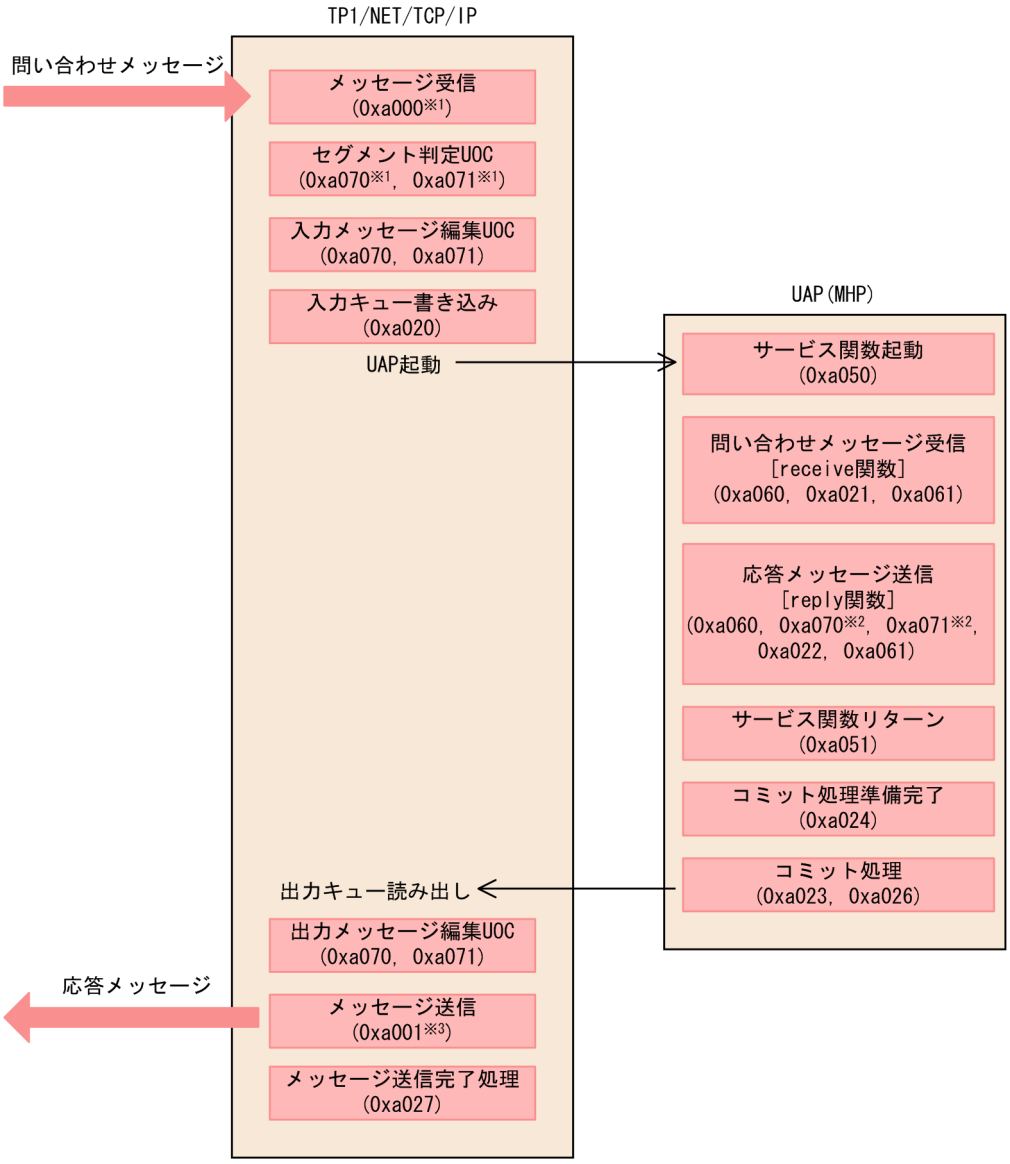
注※1
一方送信メッセージがネットワーク上で分割された場合，トレースを複数個取得することがあります。

注※2
メッセージ送信時にTCP/IPソケットの送信バッファが満杯になった場合，トレースを複数個取得することがあります。

(3)　問い合わせメッセージ受信時

問い合わせメッセージ受信時の MCF 性能検証用トレースの取得タイミングについて，次の図に示します。

図 H-7　問い合わせメッセージ受信時の MCF 性能検証用トレースの取得タイミング



（凡例）
：MCF性能検証用トレースの取得タイミング
（　）：イベントID

注※1
問い合わせメッセージがネットワーク上で分割された場合，トレースを複数個取得することがあります。

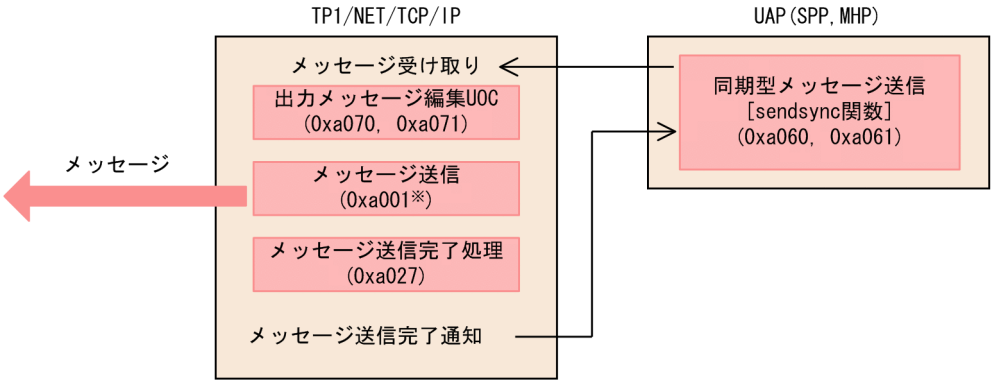
注※2
送信メッセージの通番編集UOC使用時に該当します。

注※3
メッセージ送信時にTCP/IPソケットの送信バッファが満杯になった場合，トレースを複数個取得することがあります。

(4)　同期型メッセージ送信時

同期型メッセージ送信時の MCF 性能検証用トレースの取得タイミングについて，次の図に示します。

図 H-8　同期型メッセージ送信時の MCF 性能検証用トレースの取得タイミング



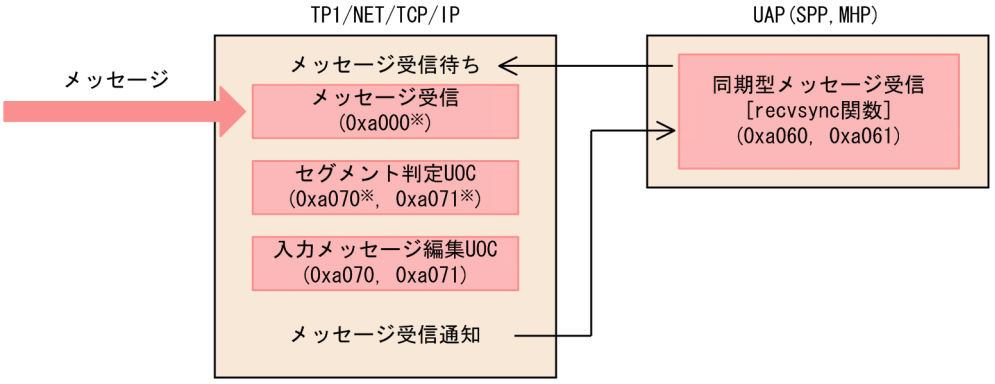
（凡例）
：MCF性能検証用トレースの取得タイミング
（　）：イベントID

注※
メッセージ送信時にTCP/IPソケットの送信バッファが満杯になった場合，トレースを複数個取得することがあります。

(5)　同期型メッセージ受信時

同期型メッセージ受信時の MCF 性能検証用トレースの取得タイミングについて，次の図に示します。

図 H-9　同期型メッセージ受信時の MCF 性能検証用トレースの取得タイミング



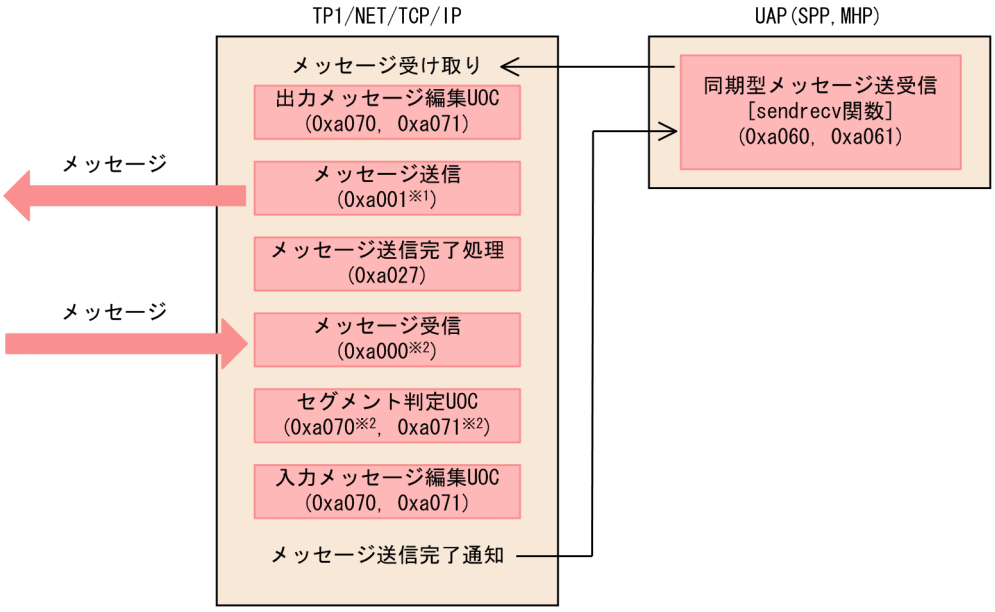
（凡例）
：MCF性能検証用トレースの取得タイミング
（　）：イベントID

注※
メッセージがネットワーク上で分割された場合，トレースを複数個取得することがあります。

(6)　同期型メッセージ送受信時

同期型メッセージ送受信時の MCF 性能検証用トレースの取得タイミングについて，次の図に示します。

図 H-10　同期型メッセージ送受信時の MCF 性能検証用トレースの取得タイミング



（凡例）
：MCF性能検証用トレースの取得タイミング
（　）：イベントID

注※1
メッセージ送信時にTCP/IPソケットの送信バッファが満杯になった場合，トレースを複数個取得することがあります。

注※2
メッセージがネットワーク上で分割された場合，トレースを複数個取得することがあります。

付録 H.3　MCF 性能検証用トレースの取得量

1 回のメッセージ送受信で取得する MCF 性能検証用トレースのトレース取得量を，次の表に示します。

表 H-4　MCF 性能検証用トレースの取得量

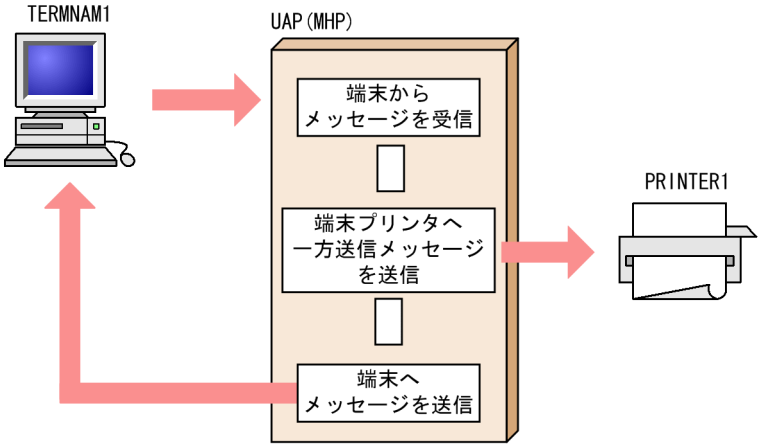
メッセージの送受信形態		トレース取得量 (単位：キロバイト)
一方送信メッセージの送信	メッセージ送達確認機能未使用	1.8
	DCCM とのメッセージ送達確認機能を使用	1.9
	任意の相手システムとのメッセージ送達確認機能を使用	2.4
一方送信メッセージの受信	メッセージ送達確認機能未使用	2.3
	DCCM とのメッセージ送達確認機能を使用	2.5
	任意の相手システムとのメッセージ送達確認機能を使用	2.8
問い合わせメッセージの受信		3.5

メッセージの送受信形態	トレース取得量 (単位：キロバイト)
同期型メッセージの送信	0.8
同期型メッセージの受信	0.9
同期型メッセージの送受信	1.4

付録 I　ユーザアプリケーションプログラムの作成例

メッセージ送受信処理の流れを，次の図に示します。

図 I-1　処理の流れ



ここでは，図に示した処理の流れのコーディング例，および TP1/NET/TCP/IP が提供しているサンプルの格納場所について説明します。

付録 I.1　コーディング例

図 I-1 の処理の流れについて，使用する言語ごとに UAP のコーディング例を示します。

(1)　C 言語（K&R 版）

C 言語（K&R 版）を使用した UAP のコーディング例を次に示します。

```
/*********************************************************/
/*     C言語を使用したUAP作成例                          */
/*********************************************************/
#include <dcmcf.h>

void ex_uap1()
{
    char termnam[9];
    DCLONG rdataleng;
    DCLONG time;
    struct{
        char mcfctl[8];
        DCLONG msglen;
        char recvdata[2036];
    }recvmsg;
    struct{
        char mcfctl[8];
        DCLONG msglen;
        char senddata1[500];
    }sendmsg;
```

```
    char senddata2[512];
    static char resv01[9] = "¥0" ;  /*** 予備領域の初期化 ***/
    static char resv02[9] = "¥0" ;  /*** 予備領域の初期化 ***/
    static char resv03[9] = "¥0" ;  /*** 予備領域の初期化 ***/
    char *workadd = (char *)&recvmsg;

    dc_mcf_receive(DCMCFFRST,DCNOFLAGS,termnam,resv01,
                   workadd,&rdataleng,2048,&time) ;
                                /* 一方送信メッセージの受信 */
    /*****************************************/
    /*********    データの処理      ***********/
    /*****************************************/

    dc_mcf_send(DCMCFEMI,DCMCFOUT,"PRINTER1",resv01,
                senddata2,504,resv02,DCNOFLAGS);
                                /* 一方送信メッセージの送信 */
    sendmsg.msglen = 504;       /* セグメント長の設定 */
    dc_mcf_send(DCMCFEMI,DCMCFOUT,"TERMNAM1",resv01,
                (char *)&sendmsg,504,resv02,DCNOFLAGS);
/* メッセージの送信 */
}
/********   C言語によるUAP 終わり    ********/
```

(2) COBOL 言語

COBOL 言語を使用した UAP のコーディング例を次に示します。

```
/*********************************************************/
/******     COBOL言語を使用したUAP作成例            ********/
/*********************************************************/

IDENTIFICATION DIVISION.
PROGRAM-ID. EXUAP1.
          :
ENVIRONMENT DIVISION.
          :
DATA DIVISION.
          :
WORKING-STORAGE SECTION.
          :
01  RCV.
  02  MSG-REC         PIC X(8) VALUE 'RECEIVE '.
  02  STATUS-CODE1    PIC X(5).
  02  FILLER          PIC X(3).
  02  SEG-CODE        PIC X(4) VALUE 'FRST'.
  02  RTN-CODE        PIC X(4) VALUE SPACE.
  02  DAY-ID          PIC 9(8).
  02  TIME-ID         PIC 9(8).
  02  SEG-LENG        PIC 9(9) COMP VALUE 2048.
  02  MCFUSE01        PIC X(4) VALUE SPACE.
  02  MCFUSE02        PIC X(4) VALUE SPACE.
  02  MCFUSE03        PIC X(4) VALUE SPACE.
  02  MCFUSE04        PIC X(4) VALUE SPACE.
  02  MCFUSE05        PIC X(8) VALUE SPACE.
  02  MCFUSE06        PIC X(4) VALUE SPACE.
  02  MCFUSE07        PIC X(8) VALUE SPACE.
```

```
  02  MCFUSE08        PIC X(4) VALUE SPACE.
  02  MCFUSE09        PIC 9(9) COMP VALUE ZERO.
  02  MCFUSE10        PIC 9(9) COMP VALUE ZERO.
  02  MCFUSE11        PIC X(1) VALUE SPACE.
  02  MCFUSE12        PIC X(1) VALUE SPACE.
  02  MCFUSE13        PIC X(14) VALUE LOW-VALUE.
01  CD1.
  02  SEG-CODE1       PIC X(4) VALUE SPACE.
  02  TERM-CODE       PIC X(8).
  02  MCFUSE14        PIC X(8).
  02  MCFUSE15        PIC X(8) VALUE SPACE.
  02  MCFUSE16        PIC X(28) VALUE LOW-VALUE.
01  DATA1.
  02  MSGSEG-LENG1    PIC 9(9) COMP.
  02  MCFUSE17        PIC X(4).
  02  MCFUSE18        PIC X(2).
  02  MCFUSE19        PIC X(1).
  02  MCFUSE20        PIC X(1).
  02  REC-MSGSEG1.
    03  MSG-LENG      PIC 9(9) COMP.
    03  RECV-DATA     PIC X(2036).
01  SND1.
  02  MSG-SEND        PIC X(8) VALUE 'SEND    '.
  02  STATUS-CODE2    PIC X(5).
  02  FILLER          PIC X(3).
  02  MCFUSE21        PIC X(4) VALUE SPACE.
  02  MCFUSE22        PIC X(4) VALUE SPACE.
  02  MCFUSE23        PIC 9(8).
  02  MCFUSE24        PIC 9(8).
  02  MCFUSE25        PIC 9(9) COMP VALUE ZERO.
  02  SEND-SEG        PIC X(4) VALUE 'EMI '.
  02  MCFUSE26        PIC X(4) VALUE SPACE.
  02  SEND-NORM       PIC X(4) VALUE 'NORM'.
  02  SEND-NO         PIC X(4) VALUE 'NSEQ'.
  02  MCFUSE27        PIC X(8) VALUE SPACE.
  02  MCFUSE28        PIC X(4) VALUE SPACE.
  02  MCFUSE29        PIC X(8) VALUE SPACE.
  02  MCFUSE30        PIC X(4) VALUE SPACE.
  02  MCFUSE31        PIC 9(9) COMP VALUE ZERO.
  02  MCFUSE32        PIC 9(9) COMP.
  02  MCFUSE33        PIC X(1) VALUE SPACE.
  02  MCFUSE34        PIC X(1) VALUE SPACE.
  02  MCFUSE35        PIC X(14) VALUE LOW-VALUE.
01  CD2.
  02  SENDSEG-CODE2   PIC X(4) VALUE 'OUT '.
  02  TERM-CODE2      PIC X(8) VALUE 'PRINTER1'.
  02  MCFUSE36        PIC X(8) VALUE SPACE.
  02  MCFUSE37        PIC X(8) VALUE SPACE.
  02  MCFUSE38        PIC X(28) VALUE LOW-VALUE.
01  DATA2.
  02  MSGSEG-LENG2    PIC 9(9) COMP VALUE 512.
  02  MCFUSE39        PIC X(8).
  02  REC-MSGSEG2     PIC X(512).
01  SND2.
  02  MSG-SEND        PIC X(8) VALUE 'SEND    '.
  02  STATUS-CODE3    PIC X(5).
  02  FILLER          PIC X(3).
  02  MCFUSE40        PIC X(4) VALUE SPACE.
```

```
  02  MCFUSE41        PIC X(4) VALUE SPACE.
  02  MCFUSE42        PIC 9(8).
  02  MCFUSE43        PIC 9(8).
  02  MCFUSE44        PIC 9(9) COMP VALUE ZERO.
  02  SEND-SEG1       PIC X(4) VALUE 'EMI '.
  02  MCFUSE45        PIC X(4) VALUE SPACE.
  02  SEND-NORM1      PIC X(4) VALUE 'NORM'.
  02  SEND-NO1        PIC X(4) VALUE 'NSEQ'.
  02  MCFUSE46        PIC X(8) VALUE SPACE.
  02  MCFUSE47        PIC X(4) VALUE SPACE.
  02  MCFUSE48        PIC X(8) VALUE SPACE.
  02  MCFUSE49        PIC X(4) VALUE SPACE.
  02  MCFUSE50        PIC 9(9) COMP VALUE ZERO.
  02  MCFUSE51        PIC 9(9) COMP.
  02  MCFUSE52        PIC X(1) VALUE SPACE.
  02  MCFUSE53        PIC X(1) VALUE SPACE.
  02  MCFUSE54        PIC X(14) VALUE LOW-VALUE.
01  CD3.
  02  SENDSEG-CODE3   PIC X(4) VALUE 'OUT '.
  02  TERM-CODE3      PIC X(8) VALUE 'TERMNAM1'.
  02  MCFUSE55        PIC X(8) VALUE SPACE.
  02  MCFUSE56        PIC X(8) VALUE SPACE.
  02  MCFUSE57        PIC X(28) VALUE LOW-VALUE.
01  DATA3.
  02  MSGSEG-LENG3    PIC 9(9) COMP VALUE 512.
  02  MCFUSE58        PIC X(8).
  02  REC-MSGSEG3.
    03  MSG-LENG      PIC 9(9) COMP.
    03  SEND-DATA     PIC X(508).
          :
          :
PROCEDURE DIVISION.
CALL 'CBLDCMCF' USING RCV CD1 DATA1.
/* 一方送信メッセージの受信 */

/**********************************/
/*********    処理1     ************/
/**********************************/

CALL 'CBLDCMCF' USING SND1 CD2 DATA2.
/* 一方送信メッセージの送信 */

/**********************************/
/*********    処理2     ************/
/**********************************/

MOVE 512 TO MSG-LENG OF REC-MSGSEG3.
/* セグメント長の設定 */
CALL 'CBLDCMCF' USING SND2 CD3 DATA3.
/* メッセージの送信 */

/**********************************/
/*********    処理3     ************/
/**********************************/

EXIT PROGRAM.
```

```
/******    COBOL言語によるUAP 終わり    ******/
```

(3) データ操作言語

データ操作言語を使用した UAP のコーディング例を次に示します。

```
*
******************************************************
*    MHPサービスプログラム                            *
******************************************************
*
 IDENTIFICATION DIVISION.

 PROGRAM-ID. SVRA.

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
******************************************************
*    ワーク変数                                      *
******************************************************
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
******************************************************
*    メッセージ受信領域                              *
******************************************************
*
  01  RECV-AREA.
    02  RE-DATALENG       PIC 9(4) COMP VALUE 1028.
    02  RE-RSV1           PIC X(2).
    02  RECV-MSG.
      03  MSG-LENG        PIC 9(9).
      03  REC-DATA        PIC X(1020).
*
******************************************************
*    一方送信メッセージ送信領域                      *
******************************************************
*
  01  SEND-IO-AREA1.
    02  IO-DATALENG       PIC 9(4) COMP VALUE 512.
    02  IO-RSV1           PIC X(2).
    02  IO-DATA           PIC X(508).
*
******************************************************
*    メッセージ送信領域                              *
******************************************************
*
  01  SEND-IO-AREA2.
    02  IO-DATALENG       PIC 9(4) COMP VALUE 516.
    02  IO-RSV1           PIC X(2).
    02  SEND-MSG.
      03  MSG-LENG        PIC 9(9) COMP.
      03  SEND-DATA       PIC X(508).
```

```
*
*****************************************************
*    通信記述項                                     *
*****************************************************
*
 COMMUNICATION SECTION.
*
*****************************************************
*    メッセージの受信の通信記述項                   *
*****************************************************
*
 CD RECV-INF
   FOR INPUT
   STATUS KEY IS        RE-STATUS
   SYMBOLIC TERMINAL IS RE-TERMNAM
   MESSAGE DATE IS      RE-DATE
   MESSAGE TIME IS      RE-TIME.
*
*****************************************************
*    一方送信メッセージの送信の通信記述項           *
*****************************************************
*
 CD SEND-IO-1
   FOR OUTPUT
   STATUS KEY IS        SE-STATUS-IO-1
   SYMBOLIC TERMINAL IS SE-TERMNAM-IO-1.
*
*****************************************************
*    メッセージの送信の通信記述項                   *
*****************************************************
*
 CD SEND-IO-2
   FOR OUTPUT
   STATUS KEY IS        SE-STATUS-IO-2
   SYMBOLIC TERMINAL IS SE-TERMNAM-IO-2.
*

 PROCEDURE DIVISION.

*
*****************************************************
*    メッセージの受信の通信記述項                   *
*****************************************************
*
   RECEIVE RECV-INF
           FIRST SEGMENT
           INTO RECV-AREA.
*
*****************************************************
*    一方送信メッセージの送信の通信記述項           *
*****************************************************
*
  MOVE 'PRINTER1' TO SE-TERMNAM-IO-1.
  SEND SEND-IO-1
       FROM SEND-IO-AREA1.
*
*****************************************************
*    メッセージの送信の通信記述項                   *
```

```
**************************************************
*
  MOVE 512 TO MSG-LENG OF SEND-MSG.
  MOVE 'TERMNAM1' TO SE-TERMNAM-IO-2.
  SEND SEND-IO-2
       FROM SEND-IO-AREA2.
  EXIT PROGRAM.
```

付録 I.2　提供するサンプルコーディング

ここでは，TP1/NET/TCP/IP が提供するサンプルコーディングの格納場所について言語ごとに示します。なお，図 I-1 に示した処理の流れのサンプルコーディングは UNIX 版だけが提供しています。Windows 版で提供しているサンプルコーディングの格納場所については TP1/NET/TCP/IP のリリースノートを参照してください。

(1)　C 言語

C 言語のサンプルコーディングは，次のファイルで提供しています。

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/TCPIP/aplib/c/ap.c

適用 OS が AIX，HP-UX または Solaris の場合

- /BeTRAN/examples/mcf/TCPIP/aplib/c/ap.c

(2)　COBOL 言語

COBOL 言語のサンプルコーディングは，次のファイルで提供しています。

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/TCPIP/aplib/cobol/ap.cbl

適用 OS が AIX，HP-UX または Solaris の場合

- /BeTRAN/examples/mcf/TCPIP/aplib/cobol/ap.cbl

(3)　データ操作言語

データ操作言語のサンプルコーディングは，次のファイルで提供しています。

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/TCPIP/aplib/dml/ap.cbl

適用 OS が AIX，HP-UX または Solaris の場合

- /BeTRAN/examples/mcf/TCPIP/aplib/dml/ap.cbl

付録 J　理由コード一覧

ここでは，ERREVT2 発生時，CERREVT 発生時，および MDELEVT 発生時の理由コードを示します。

ERREVT2 発生時の理由コードを次の表に示します。

表 J-1　ERREVT2 の理由コード

C 言語の理由コード（16 進数字）	COBOL 言語の理由コード（外部 10 進数字）	ERREVT2 の通知理由
DCMCF_NO_SERV（0010）	0010	アプリケーション名に相当する MHP のサービスがありません。
DCMCF_SCD_ERR（0020）	0020	ユーザサーバ未起動などによって MHP の起動に失敗しました。
DCMCF_QUE_BUF_ERR（0030）	0030	動的共用メモリが不足しました。
DCMCF_QUE_FIL_OVER（0031）	0031	キューファイルが満杯になりました。
DCMCF_QUE_LIMIT_OVER（0032）	0032	入力メッセージ最大格納数を超過しました。
DCMCF_QUE_IO_ERR（0033）	0033	入力キューに障害が発生しました。
DCMCF_AP_CLOSE（0040）	0040	MHP のアプリケーションが閉塞しています。
DCMCF_AP_SECURE（0041）	0041	MHP のアプリケーションがセキュア状態です。
DCMCF_SERV_CLOSE（0042）	0042	• MHP のサービスまたはサービスグループが閉塞しています。 • スケジュール閉塞されているサービスグループの入力キューに未処理受信メッセージが残った状態で，OpenTP1 を正常終了または計画停止 A で終了しました。
DCMCF_SERV_SECURE（0043）	0043	MHP のサービスグループがセキュア状態です。
DCMCF_ABNORMAL_END（0050）	0050	• MHP で呼び出す receive 関数にセグメントを渡す前に，MHP が異常終了しました。 • DBMS の障害などによって，トランザクションの開始に失敗しました。

CERREVT 発生時の理由コードを，次の表に示します。

表 J-2　CERREVT 発生時の理由コード

理由コード 1（16 進数字）	理由コード 2（16 進数字）	発生条件	発生個所※1
DCMTCP_RSN1_MCF（00000001）（MCF 障害）	DCMTCP_RSN2_CNNC（00000000）	コネクション確立不可能	コネクション

理由コード 1 (16 進数字)	理由コード 2（16 進数字）	発生条件	発生個所※1
DCMTCP_RSN1_MCF （00000001）（MCF 障害）	DCMTCP_RSN2_CNER （00000001）	コネクション切断	コネクション
	DCMTCP_RSN2_RBFOF （00000002）	受信バッファオーバフロー	コネクション
	DCMTCP_RSN2_SBFOF （00000003）	送信バッファオーバフロー	論理端末
	DCMTCP_RSN2_ERROR （00000004）	TP1/NET/TCP/IP 内部論理矛盾	コネクション
	DCMTCP_RSN2_OTGET （00000005）	出力キュー読み込み障害	論理端末
	DCMTCP_RSN2_NOBUF （00000007）	受信バッファ取得失敗	コネクション
	DCMTCP_RSN2_SNDER （00000008）	メッセージ送信不可能	コネクション 論理端末
	DCMTCP_RSN2_NXTTO （00000009）	後続メッセージ監視タイムアウト	コネクション
	DCMTCP_RSN2_SNBUF （0000000A）	送信バッファ取得失敗	論理端末
	DCMTCP_RSN2_SRTO （0000000B）	同期型メッセージの送受信タイムアウト	コネクション
	DCMTCP_RSN2_DCTLE （0000000C）	論理端末仕掛り中の mcftdctle コマンド要求	コネクション
	DCMTCP_RSN2_RMSGLN （0000000D）	メッセージサイズエリア内容不正	コネクション
	DCMTCP_RSN2_NOTRFTO （0000000F）	無通信状態監視によるタイムアウト発生	コネクション
	DCMTCP_RSN2_ANBUF （00000010）	送達確認メッセージ（応答専用データ）用送信バッファ取得失敗	コネクション
	DCMTCP_RSN2_ERRMSG （00000011）	不正データ受信	コネクション
	DCMTCP_RSN2_ACKRTO （00000012）	送達確認メッセージ（応答専用データ）受信監視タイムアウト	コネクション
	DCMTCP_RSN2_ACKSTO （00000013）	送達確認メッセージ（応答専用データ）送信完了監視タイムアウト	コネクション
	DCMTCP_RSN2_RCVCOL （00000014）	送達確認メッセージ（応答専用データ）送信処理中一方送信メッセージの受信	コネクション

理由コード 1 (16 進数字)	理由コード 2（16 進数字）	発生条件	発生個所※1
DCMTCP_RSN1_MCF （00000001）（MCF 障害）	DCMTCP_RSN2_RVFAIL （00000015）	受信メッセージ通知失敗	コネクション
	DCMTCP_RSN2_SRCOL （00000016）	送受信メッセージ衝突	コネクション
	DCMTCP_RSN2_DCTCNF （00000017）	運用コマンドまたは API による強制解放	コネクション
	DCMTCP_RSN2_REPLACE （00000018）	コネクションリプレースによる切断	コネクション
	DCMTCP_RSN2_HLDLIMT （00000019）	受信メッセージの最大保留数の超過	コネクション
	DCMTCP_RSN2_NOCONT （0000001A）	次起動アプリケーション型不正	コネクション
DCMTCP_RSN1_UOC （00000003）（UOC 障害）	UOC からの詳細リターンコード	入力セグメント判定 UOC エラーリターン	コネクション
	UOC からの詳細リターンコード	入力，および出力メッセージ編集 UOC エラーリターン	論理端末
	DCMTCP_RSN2_BCNT （00000001）	使用バッファ数不正	論理端末
	DCMTCP_RSN2_SEG （00000002）	有効セグメント不正	論理端末
	DCMTCP_RSN2_BADR （00000003）	編集バッファアドレス不正	論理端末
	DCMTCP_RSN2_TIVL （00000005）	タイマ値不正	コネクション
	DCMTCP_RSN2_TICD （00000006）	タイマセット指示種別不正	コネクション
	DCMTCP_RSN2_SEGEND （00000007）	セグメント完/未完種別不正	コネクション
	DCMTCP_RSN2_NXTSIZ （00000008）	次メッセージサイズ不正	コネクション
	DCMTCP_RSN2_NXTADR （00000009）	次メッセージ先頭アドレス不正	コネクション
	DCMTCP_RSN2_NOWSIZ （0000000A）	該当メッセージサイズ不正	コネクション
	DCMTCP_RSN2_NOWSSI （0000000B）	残留該当メッセージサイズ不正	コネクション

理由コード 1 (16 進数字)	理由コード 2（16 進数字）	発生条件	発生個所※1
DCMTCP_RSN1_UOC （00000003）（UOC 障害）	DCMTCP_RSN2_SPCCLS （00000012）	受信メッセージ判定 UOC でのコネクション解放指示	コネクション
	DCMTCP_RSN2_SPCSTP （00000013）	受信メッセージ判定 UOC での送信停止指示	論理端末
	DCMTCP_RSN2_MRPARM （00000014）	受信メッセージ判定 UOC でのパラメタ不正	コネクション
	DCMTCP_RSN2_MHPARM （00000015）	受信メッセージ保留判定 UOC でのパラメタ不正	コネクション
DCMTCP_RSN1_CLS※2 （00000004）	－	相手局からのコネクション解放	コネクション

（凡例）

－：該当する内容がないことを表します。

注※1

発生個所に対応した障害要因コードが，CERREVT の err_fact に設定されます。

注※2

コネクション定義（mcftalccn -f）で cerr を指定した場合だけ有効になります。

MDELEVT 発生時の理由コードを，次の表に示します。

表 J-3　MDELEVT 発生時の理由コード

理由コード（16 進数字）	MDELEVT の通知理由
DCMTCP_DRSN_UIND（00000001）	任意の相手システムとのメッセージ送達確認を使用している（コネクション定義（mcftalcle -u）の delichk オペランドに use を指定）場合に，受信メッセージ判定 UOC で受信したメッセージの種別に破棄メッセージを指定しました。
DCMTCP_DRSN_CNDCT（00000003）	コネクション再確立時の未送信メッセージの送信抑止機能を使用している（論理端末定義（mcftalcle -d）の replacemsg オペランドに discard を指定）場合に，MHP でメッセージ受信後にコネクションが再確立されました。
DCMTCP_DRSN_RCVDUP(00000004)	問い合わせ応答中および継続問い合わせ応答中の論理端末がメッセージを受信したため，破棄しました。
DCMTCP_DRSN_SRANS(00000005)	メッセージ送信処理中に問い合わせメッセージを受信したため，破棄しました。
DCMTCP_DRSN_SRCONT(00000006)	メッセージ送信処理中に継続問い合わせメッセージを受信したため，破棄しました。

付録 K　用語解説

(英字)

AJ（メッセージ送信完了ジャーナル）

MCF が取得する統計用の履歴情報の一つです。メッセージ出力通番，出力先論理端末名などで構成されます。

AJ の取得タイミングは，メッセージ送達確認機能を使用しているかどうかで異なります。メッセージ送達確認機能を使用していない場合は，メッセージを TCP/IP ソケットの送信バッファにすべて格納した直後です。メッセージ送達確認機能を使用している場合は，メッセージを TCP/IP ソケットの送信バッファにすべて格納したあと，相手システムから送達確認メッセージまたは応答専用データを受信した直後です。

GJ（メッセージ受信ジャーナル）

MCF が取得する統計用の履歴情報の一つです。入力メッセージサイズ，入力論理端末名などで構成されます。

GJ の取得タイミングは，非同期受信関数を発行して，MHP が受信メッセージを入力キューから取り出した直後です。

IJ（メッセージ入力ジャーナル）

MCF が取得する統計用の履歴情報の一つです。メッセージ入力通番，入力論理端末名，入力メッセージ種別，および入力メッセージなどで構成されます。

IJ の取得タイミングは，相手システムから受信したメッセージを入力キューに格納する直前です。

MJ（メッセージジャーナル）

MCF が取得する統計用の履歴情報の一つです。入力論理端末名または出力論理端末名，メッセージジャーナル種別，入力または出力メッセージ（入力メッセージ編集前のデータ，または出力メッセージ編集後のデータ）などで構成されます。

MJ の取得タイミングは，入力メッセージの場合，入力メッセージ編集 UOC を呼び出す直前です。また，出力メッセージの場合，出力メッセージ編集 UOC を呼び出した直後です。

OJ（メッセージ出力ジャーナル）

MCF が取得する統計用の履歴情報の一つです。メッセージ出力通番，出力論理端末名，出力メッセージ種別，出力メッセージなどで構成されます。

OJ の取得タイミングは，非同期送信関数を発行して，UAP が送信メッセージを出力キューに格納した直後です。

TCP/IP プロトコル

米国防省を中心に開発されたプロトコル群の総称です。主として，LAN 上のワークステーションおよびパーソナルコンピュータ間での通信プロトコルに使用され，事実上の標準プロトコルとなっています。

（カ行）

キープアライブ

相手システムに一定間隔でパケットを送信することで，コネクション切断を検出するための機能です。

クライアント

コネクション確立時，相手システムに対して，確立要求をする機能およびシステムです。

コネクション

TP1/NET/TCP/IP が AP 間通信をするときに，自システムと相手システムの間に確立する論理的通信路です。

（サ行）

サーバ

コネクション確立時，相手システムからの確立要求を受ける機能およびシステムです。

出力キュー

処理の結果として出力メッセージを管理する待ち行列です。

（ナ行）

入力キュー

処理要求として入力メッセージを管理する待ち行列です。

（ハ行）

ポート番号

ネットワーク上で提供する，各サービスに付ける固有の番号です。

（ラ行）

論理端末

TP1/NET/TCP/IP と UAP との通信接点です。TP1/NET/TCP/IP と UAP は，論理端末単位にメッセージの送受信をします。

索引

E

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

あ

い

う

お

か

き

く

け

こ

さ

し

そ

た

つ

て

と

も

ゆ

よ

り

ろ