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/User Datagram Protocol 編

解説・手引・文法・操作書

3000-3-D75-10

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)

P-8164-3111 uCosminexus TP1/Message Control 07-50

P-8164-3211 uCosminexus TP1/NET/Library 07-50

P-F8164-3211U uCosminexus TP1/NET/User Datagram Protocol 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。

これらの製品は，ISO9001 および TickIT の認証を受けた品質マネジメントシステムで開発されました。

■ 輸出時の注意

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

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

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

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

■ 発行

2016 年 3 月　3000-3-D75-10

■ 著作権

変更内容

変更内容（3000-3-D75-10） uCosminexus TP1/Message Control 07-50，uCosminexus TP1/NET/Library 07-50，uCosminexus TP1/NET/User Datagram Protocol 07-50

追加・変更内容	変更個所
メッセージを受信したときに受信バッファが不足したり，入力メッセージ最大格納数を超過した場合，論理端末を閉塞するかどうかを指定できるようにした。 これに伴い，論理端末定義（mcftalcle -f）に rcvoverflow オペランドを追加した。	2.2.2, 6. システム定義 TP1/NET/UDP 固有のシステム定義の種類, mcftalcle（論理端末定義の開始), 9.1
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	6. システム定義 システムサービス共通情報定義
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。	6. システム定義 MCF 定義オブジェクトの生成
MCF 定義オブジェクトの解析コマンドを追加した。	6. システム定義 MCF 定義オブジェクトの解析
MCF トレースファイルの見積もり式の説明を追加した。	6. システム定義 MCF トレースファイルの見積もり式
次に示すバージョンの変更点を記載した。 • TP1/NET/User Datagram Protocol 07-50	付録 A.1

uCosminexus TP1/Message Control 07-05，uCosminexus TP1/NET/Library 07-05，uCosminexus TP1/NET/User Datagram Protocol 07-01

追加・変更内容	変更個所
論理端末の閉塞，閉塞解除，および状態表示について，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を使用する場合の説明を追加した。	2.1.1, 3. C 言語のライブラリ関数 C 言語のライブラリ関数の一覧, dc_mcf_tactle, dc_mcf_tdctle, dc_mcf_tlsle

追加・変更内容	変更個所
論理端末の閉塞，閉塞解除，および状態表示について，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を使用する場合の説明を追加した。	4.　COBOL-UAP 作成用プログラムインタフェース COBOL-UAP 作成用プログラムインタフェースの一覧， CBLDCMCF('TACTLE△△')， CBLDCMCF('TDCTLE△△')， CBLDCMCF('TLSLE△△△')
TP1/NET/User Datagram Protocol で使用するポート番号の説明を追加した。	2.1.1(3)
送信元特定マルチキャスト（SSM）を受信できるようにした。これに伴い，論理端末定義（mcftalcle -m）に shostname[1～8]オペランドを追加した。	2.1.4(7)(b)， 6.　システム定義 TP1/NET/UDP 固有のシステム定義の種類， mcftalcle（論理端末定義の開始）
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。	3.　C 言語のライブラリ関数 C 言語のライブラリ関数の一覧
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。	3.　C 言語のライブラリ関数 dc_mcf_resend， 4.　COBOL-UAP 作成用プログラムインタフェース CBLDCMCF('RESEND△△')
データ操作言語の通信文と C 言語のライブラリ関数の対応の説明を追加した。	4.　COBOL-UAP 作成用プログラムインタフェース COBOL-UAP 作成用プログラムインタフェースの一覧
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。	5.2.3(4)(b)，5.2.4
MCF 性能検証用トレースの説明を追加した。	6.　システム定義 システムサービス情報定義， システムサービス共通情報定義， 付録 G
OpenTP1 システムを変更する場合に，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。	6.　システム定義 OpenTP1 システムの変更に影響する定義
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。	7.　運用コマンド mcftlsle
スタート関数を呼び出したあとの注意事項を追加した。	8.2
次に示すバージョンの変更点を記載した。 • TP1/NET/User Datagram Protocol 07-01	付録 A.2

追加・変更内容	変更個所
バージョン 6 以前のインタフェースの変更一覧を追加した。	付録 C
ソケット関数の処理の流れを追加した。	付録 F

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルは，TP1/NET/User Datagram Protocol 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 User Datagram Protocol（UDP 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/User Datagram Protocol（TP1/NET/UDP）の概要について説明しています。

第 2 章　機能

TP1/NET/UDP での論理端末の運用，メッセージの種類と送受信の方法など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/UDP 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/UDP 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/UDP に関連するユーザオウンコーディング，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

UDP 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/UDP 固有のシステム定義，およびシステム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/UDP で使用す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/UDP を OpenTP1 システムに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/UDP の運用中に発生するおそれがある障害と，TP1/NET/UDP の対応処理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以前からバージョン７に移行する際の注意事項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

障害発生時の処理の流れについて説明しています。

付録 F　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 G　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 H　ユーザアプリケーションプログラムの作成例

TP1/NET/UDP 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 I　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 J　このマニュアルの参考情報

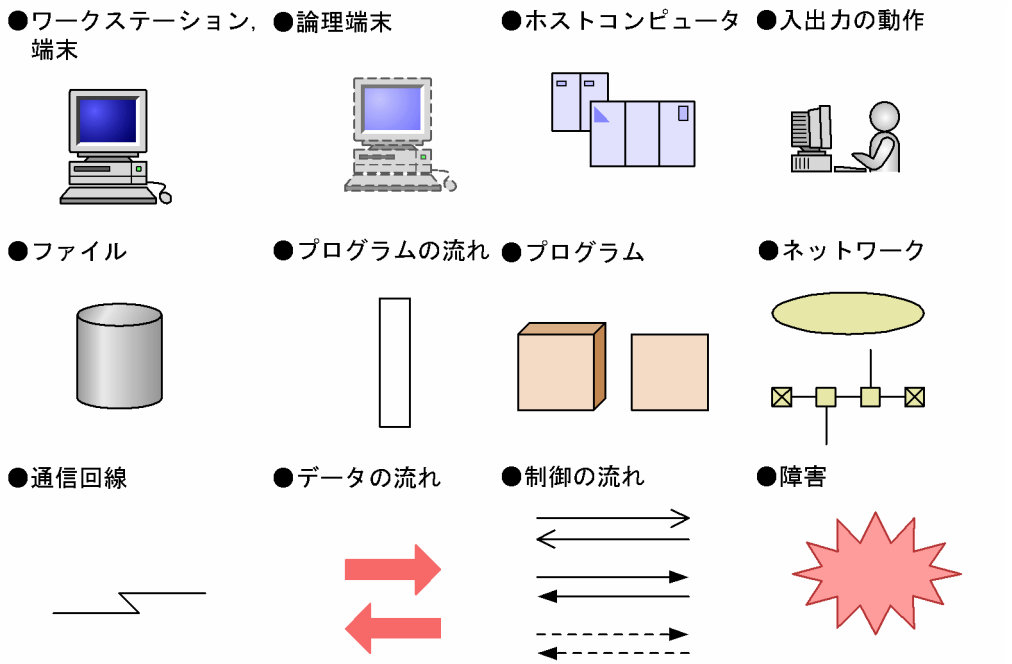
関連マニュアル，このマニュアルで使用している略語の意味などを説明しています。

付録 K　用語解説

TP1/NET/UDP で使用する用語について説明しています。

■ 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）〔-v アプリケーション名〕 　-v 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）-i auto | manual 　-i オプションに auto か manual 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）{DCMCFFRST | DCMCFSEG} 　DCMCFFRST と DCMCFSEG のうち，どちらかを指定できることを示します。
＿ （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）論理端末名称 1△論理端末名称 2 　論理端末名称 1 と論理端末名称 2 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（A～F，a～f）
＜識別子＞	先頭がアルファベットの英数字列
＜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，／，および.（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），および.（ピリオド）

■ 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

図目次

表目次

1 概要

TP1/NET/UDP 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 UDP 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/UDP の概要について説明します。

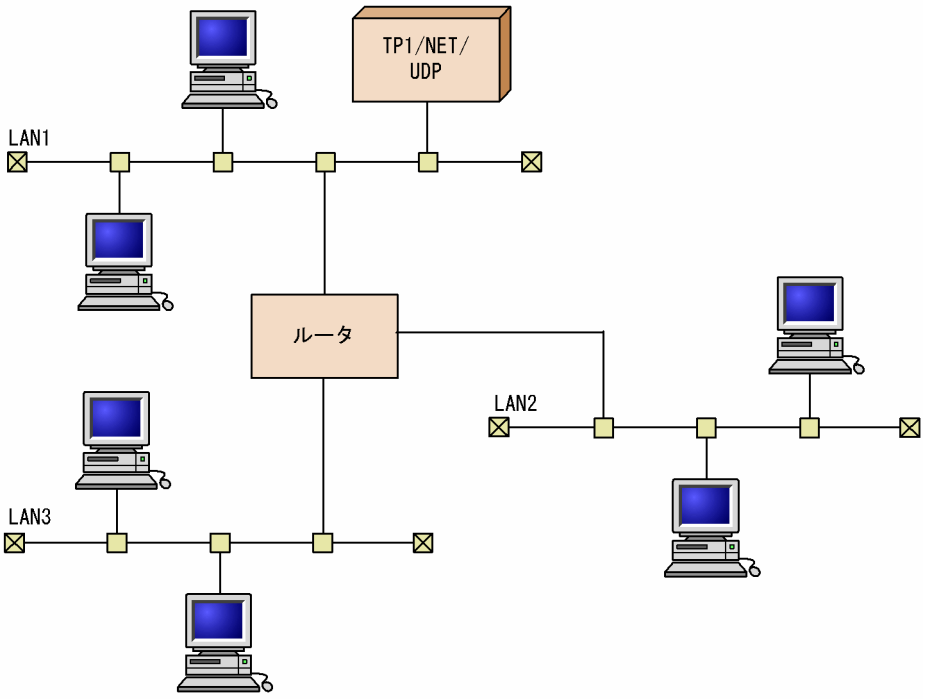
1.1　AP 間通信の概要

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/UDP は，OS が提供する UDP のソケットを利用して AP 間通信をするプログラムです。送受信できるメッセージの種類は，ブロードキャストメッセージ，ユニキャストメッセージ，およびマルチキャストメッセージの 3 種類です。

TP1/NET/UDP 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/UDP を使用したネットワーク構成の例



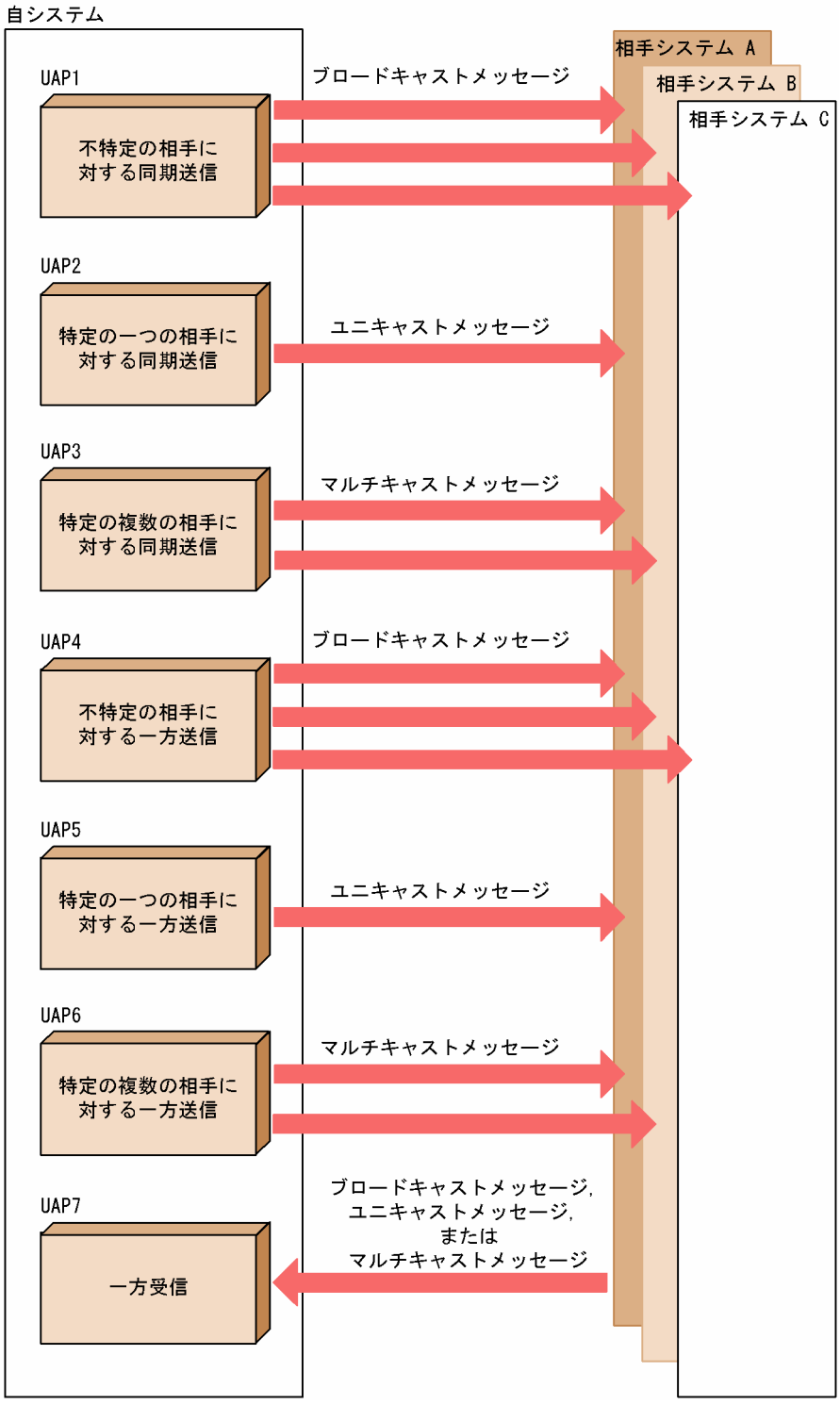
1.2 AP 間通信の形態

TP1/NET/UDP を使用した AP 間通信の形態には，以下の七つがあります。

- **不特定の相手に対する同期送信**
  ブロードキャストメッセージを同期送信することで実現します。
- **特定の一つの相手に対する同期送信**
  ユニキャストメッセージを同期送信することで実現します。
- **特定の複数の相手に対する同期送信**
  マルチキャストメッセージを同期送信することで実現します。
- **不特定の相手に対する一方送信**
  ブロードキャストメッセージを一方送信することで実現します。
- **特定の一つの相手に対する一方送信**
  ユニキャストメッセージを一方送信することで実現します。
- **特定の複数の相手に対する一方送信**
  マルチキャストメッセージを一方送信することで実現します。
- **一方受信**
  ブロードキャストメッセージ，ユニキャストメッセージ，およびマルチキャストメッセージの受信が該当します。なお，TP1/NET/UDP のユーザに対し，受信したメッセージの種類は通知されません。

TP1/NET/UDP を使用した AP 間通信の例を次の図に示します。

図 1-2　TP1/NET/UDP を使用した AP 間通信の例



注意事項

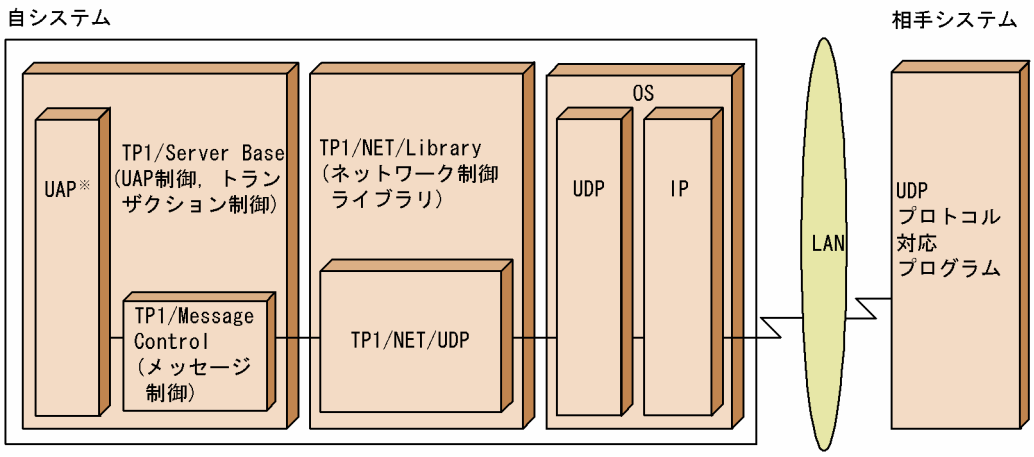
マルチキャスト通信をするためには，マルチキャストをサポートした LAN アダプタやルータなどのネットワーク機器が必要です。

1.3　ソフトウェアの構成の例

TP1/NET/UDP 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/UDP 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 1-3　TP1/NET/UDP を組み込んだソフトウェア構成の例



注※

TP1/NET/UDP 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。

2 機能

一般に，AP 間通信をするときには，自システムと相手システムとの間であらかじめ通信上の規約（プロトコル）を決める必要があります。TP1/NET/UDP は，論理端末を通してメッセージを送受信します。

この章では，TP1/NET/UDP での論理端末の運用，メッセージの種類と送受信の方法などについて説明します。

2.1　AP 間通信の仕組み

UDP プロトコルを使用した通信では，コネクションを確立することなく，メッセージの送受信ができます。そのため，TP1/NET/UDP を使用する場合も，コネクションを意識する必要がありません。
TP1/NET/UDP では，**論理端末**を通して自システムの UAP と相手システムの間でメッセージを送受信します。

この節では，TP1/NET/UDP を運用する上での論理端末の閉塞解除および閉塞の処理の流れと，論理端末とアプリケーションの型の関係について説明します。さらに，通信相手のアドレスの指定方法と AP 間通信の注意事項について説明します。

2.1.1　論理端末の閉塞解除と閉塞

論理端末の閉塞解除と閉塞の処理の流れについて説明します。

(1)　論理端末の閉塞解除

TP1/NET/UDP で AP 間通信をするためには，論理端末の閉塞を解除する必要があります。論理端末の閉塞を解除すると，自システムの UDP プロトコルに自ポート番号が登録されて，TP1/NET/UDP で AP 間通信ができる状態になります。

論理端末の閉塞を解除するには，次の 3 とおりの方法があります。

- OpenTP1 の開始時および再開始時に論理端末の閉塞を解除する方法
- 運用コマンド（mcftactle）の入力によって論理端末の閉塞を解除する方法
- API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）の発行によって論理端末の閉塞を解除する方法

(a)　OpenTP1 の開始時および再開始時の閉塞解除

OpenTP1 の開始時および再開始時に論理端末を自動的に閉塞解除できます。

論理端末を自動的に閉塞解除するには，論理端末定義（mcftalcle -i）に auto を指定します。

このオプションに manual を指定する場合，次のどちらかの方法で TP1/NET/UDP の起動後に論理端末の閉塞を解除する必要があります。

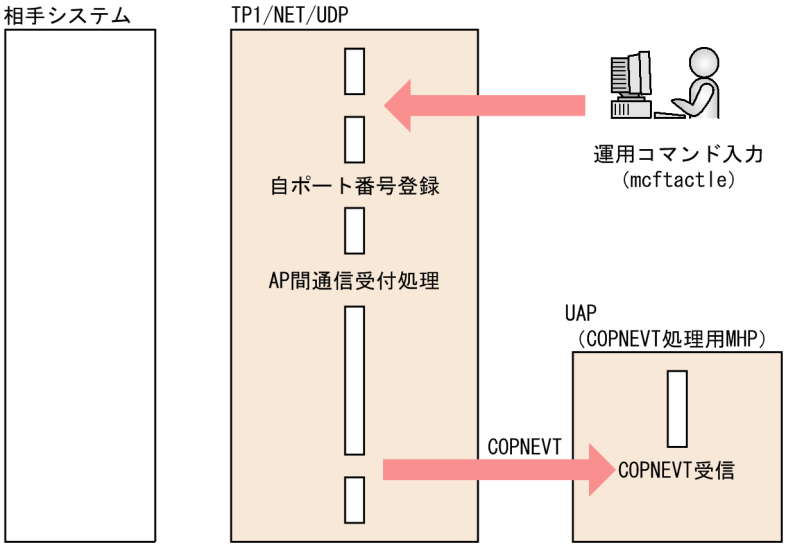
- TP1/NET/UDP の起動後に運用コマンド（mcftactle）を入力する
- API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する

デフォルトは manual です。

(b)　運用コマンドの入力による閉塞解除

運用コマンド（mcftactle）の入力によって論理端末の閉塞を解除する処理の流れを次の図に示します。

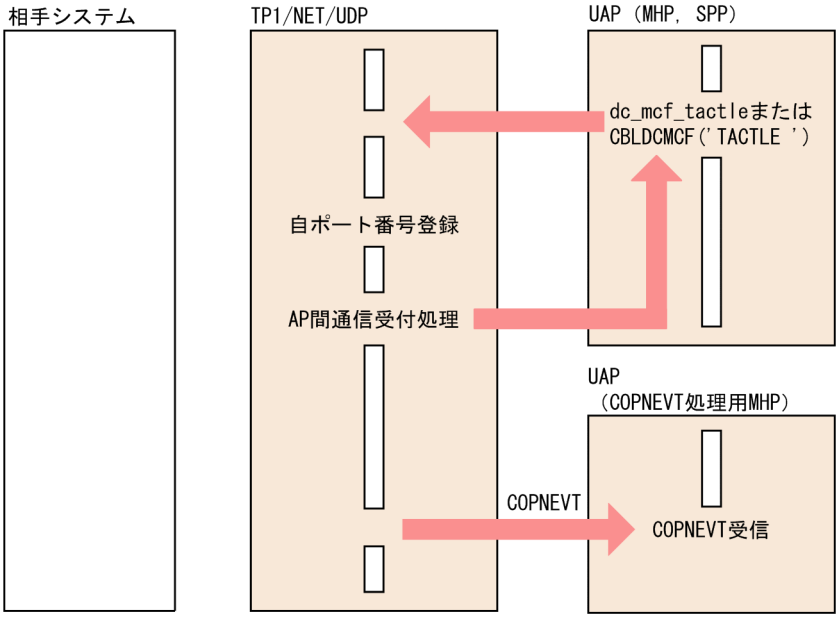
図 2-1　運用コマンド（mcftactle）の入力による論理端末の閉塞解除



(c)　API の発行による閉塞解除

API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）の発行によって論理端末の閉塞を解除する処理の流れを次の図に示します。

図 2-2　API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）の発行による論理端末の閉塞解除



論理端末の閉塞解除が完了すると，論理端末の閉塞解除を通知するメッセージ（KFCA18900-I）が出力されます。さらに，UAP に対して状態通知イベント（COPNEVT）が通知されます。したがって，COPNEVT の通知を契機に業務を開始するように UAP を設計してください。

論理端末の閉塞解除の処理中に障害が発生した場合は，論理端末の閉塞解除の失敗を通知するメッセージ（KFCA18903-E）が出力され，UAP に対して障害通知イベント（CERREVT）が通知されます。

注意事項

TP1/Message Control および TP1/NET/Library から，コネクションに関連するメッセージが出力される場合があります。これらのメッセージでコネクション名として出力される情報には，論理端末名称が出力されます。

(2)　論理端末の閉塞

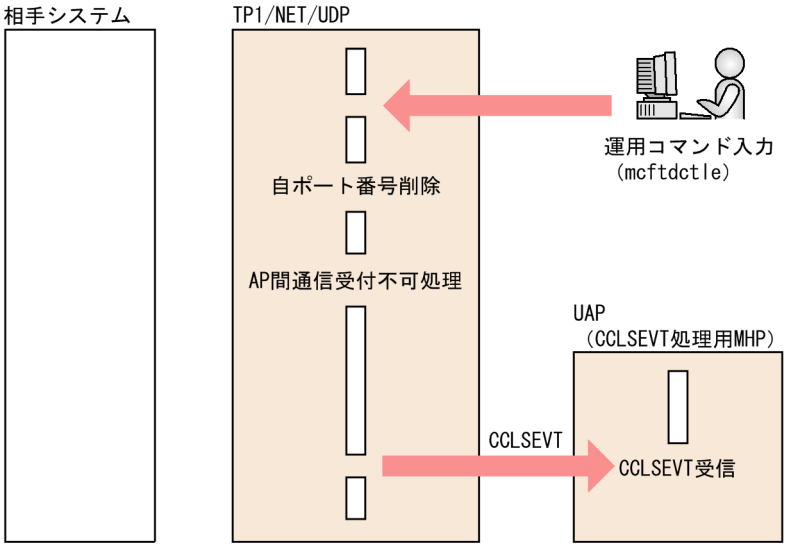
TP1NET/UDP は，次に示す場合に論理端末を閉塞します。

- オンラインが終了した場合
- 運用コマンド（mcftdctle）が入力された場合
- API（dc_mcf_tdctle 関数または CBLDCMCF('TDCTLE△△')）が発行された場合
- 論理端末に障害が発生した場合

論理端末を閉塞すると，自システムの UDP プロトコルから自ポート番号が削除されて，TP1/NET/UDP で AP 間通信ができない状態になります。

運用コマンド（mcftdctle）の入力によって論理端末を閉塞する処理の流れを次の図に示します。

図 2-3　運用コマンド（mcftdctle）の入力による論理端末の閉塞



論理端末の閉塞が完了すると，論理端末の閉塞を通知するメッセージ（KFCA18901-I）が出力されます。さらに，UAP に対して次に示すイベントが通知されます。

- 運用コマンド（mcftdctle）の入力，または API（dc_mcf_tdctle 関数もしくは CBLDCMCF('TDCTLE△△')）の発行によって論理端末を閉塞した場合は，状態通知イベント（CCLSEVT）が通知されます。
- 障害発生によって論理端末を閉塞した場合は，障害通知イベント（CERREVT）が通知されます。
- オンライン終了の場合は，イベントは通知されません。

(3) TP1/NET/UDP 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	mcftalcle -r portno	1～65535※	－	MCF 通信サービス

（凡例）

－：該当しません（省略できません）。

注※

予約ポート（1～1023）を使用する論理端末が一つでもある場合，OpenTP1 管理者のユーザ ID には，スーパユーザ権限を持つユーザを登録する必要があります。

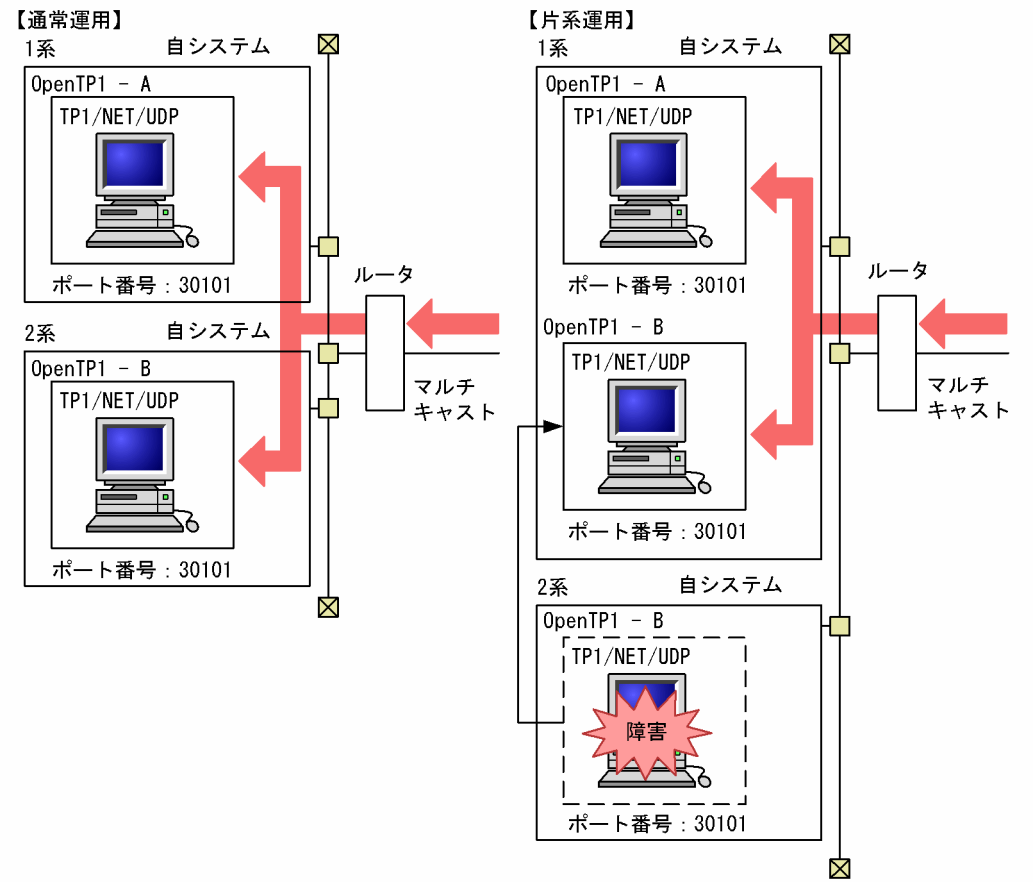
(4) 複数の MCF 通信プロセスによる同一ポート番号の共用

TP1/NET/UDP では，同一マシン上の複数の MCF 通信プロセス間で，同一の自ポート番号を共用できます。共用しているポート番号あてのブロードキャストメッセージおよびマルチキャストメッセージは，複数の MCF 通信プロセスで受信できます。

同一の自ポート番号を複数の MCF 通信プロセスで共用した構成で，論理端末の閉塞を解除するには，ポート番号を共用するすべての MCF 通信プロセスの論理端末定義（mcftalcle -r）の reuse オペランドに yes を指定してください。

これによって，次の図に示すような相互ホットスタンバイ構成での運用ができます。

図 2-4　相互ホットスタンバイ構成での運用例



注意事項

- 一つの MCF 通信プロセスの複数の論理端末で，同一の自ポート番号は共用できません。MCF 通信構成定義の定義オブジェクト生成時にエラーとなります。複数の論理端末で同一のポートを共用する場合は，MCF 通信プロセスを分割してください。
- 複数の MCF 通信プロセスで同一のポート番号を共用する場合，MCF 通信構成定義の指定値は統一してください。指定値が混在する場合，二つ目以降の論理端末の閉塞解除に失敗します。

2.1.2　論理端末とアプリケーションの型の関係

TP1/NET/UDP 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/UDP 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/UDP のアプリケーションの型は非応答型（noans）です。

論理端末の端末タイプ，メッセージ，アプリケーションの型，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 2-2　論理端末の端末タイプ，メッセージ，アプリケーションの型，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	メッセージ	アプリケーションの型	UAP インタフェース	通信形態
any（任意型論理端末）	一方送信メッセージ	非応答型（noans）	send	一方送信
			receive	一方受信
			sendsync	同期送信

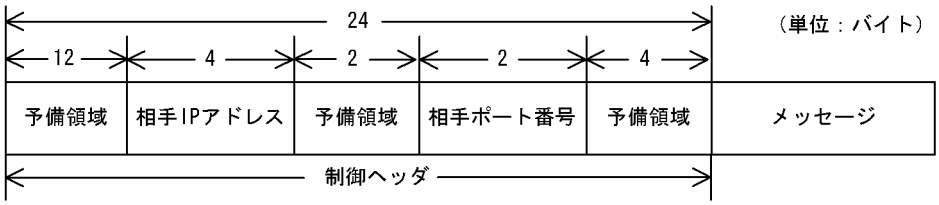
2.1.3　通信相手のアドレスの指定

任意の通信相手とメッセージを送受信するために，TP1/NET/UDP では相手アドレス指定用の**制御ヘッダ**をメッセージに付加して使用します。

メッセージの送信時には，制御ヘッダに指定する相手のアドレスを変更することで，送信のたびに相手を変更できます。メッセージの受信時には，TP1/NET/UDP がメッセージの先頭に制御ヘッダを付加します。

通信相手のアドレスを指定するための制御ヘッダの形式を次の図に示します。

図 2-5　制御ヘッダの形式

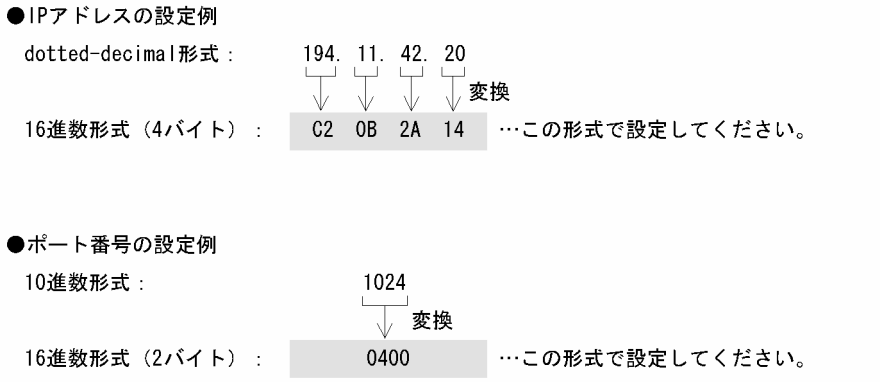


メッセージの送信時には，制御ヘッダの所定の領域に通信相手の **IP アドレス**と**ポート番号**を設定する必要があります。

IP アドレスは 16 進数形式で設定します。dotted-decimal 形式では指定できません。ポート番号は 10 進数を 16 進数形式に変換したあと，2 バイト表記で設定します。予備領域はすべてゼロクリアしてください。

制御ヘッダに設定する IP アドレスとポート番号の例を次の図に示します。

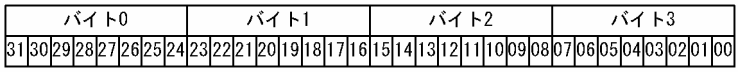
図 2-6　通信相手の IP アドレスとポート番号の設定例



メッセージの受信時には，TP1/NET/UDP が通信相手の IP アドレスとポート番号を制御ヘッダの所定の領域に設定します。この内容を利用して，送信元のアドレスを識別できます。ホスト名が必要な場合は，設定された IP アドレスを基に gethostbyaddr()関数などを使用して，UAP または UOC で取得してください。

なお，通信相手の IP アドレスと IP ポート番号のバイト順序は，ビッグエンディアンでなければなりません。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。ビッグエンディアンでのバイト順序を次の図に示します。

図 2-7　ビッグエンディアンでのバイト順序



2.1.4　AP 間通信の注意事項

AP 間通信を行う上での注意事項を示します。

(1)　メッセージの保証

TP1/NET/UDP では，メッセージの欠落，重複，および到着順序のチェックやフロー制御は実施しません。また，下位プロトコルに UDP プロトコルを使用するため，メッセージは保証されません。したがって，UAP 間でメッセージをやり取りする上で，次に示す項目を事前に取り決めておく必要があります。

- メッセージの送達確認方法
- メッセージの重複および欠落のチェック方法と対処方法
- メッセージのフロー制御方法

(2)　送受信時のメッセージ長

UDP プロトコルを使用する通信では，送受信するメッセージ長とシステムバッファ長の整合性に注意する必要があります。**システムバッファ**とは，UDP プロトコルが持つ内部バッファのことです。システムバッ

ファ長は，論理端末定義（mcftalcle -s）で指定します。指定方法については，6 章の「mcftalcle（論理端末定義の開始）」を参照してください。

- システムバッファ長の最大値および実際に確保されるシステムバッファ長はご使用の OS の実装に依存します。
- TP1/NET/UDP で送信するメッセージ長が，システムの送信バッファ長よりも大きい場合，メッセージの送信は失敗し，論理端末が閉塞します。
- TP1/NET/UDP の受信バッファ長が，相手システムから受信したメッセージ長より小さい場合，かつ，相手システムから受信したメッセージ長がシステムの受信バッファ長より小さい場合，受信は正常に完了します。ただし，受信できるメッセージのサイズは TP1/NET/UDP のバッファ長までとなります。残りは UDP プロトコルによって破棄されます。
- TP1/NET/UDP がメッセージを受信する前に，システムの受信バッファが満杯になった場合，そのあとに受信したメッセージは OS が破棄します。システムの受信バッファには，運用形態に合わせて余裕を持った値を指定してください。

(3) メッセージの種類の判別

TP1/NET/UDP で送受信できるメッセージの種類には，ブロードキャストメッセージ，ユニキャストメッセージ，およびマルチキャストメッセージの 3 種類がありますが，メッセージ送受信時にメッセージの種類の判定はされません。

メッセージの送信時は，指定する相手アドレスの種別によって，次の表のように区別してください。

表 2-3 相手アドレスとメッセージの種類の対応

相手アドレス	メッセージの種類
ブロードキャストアドレス	ブロードキャストメッセージ
特定の相手ノードアドレス	ユニキャストメッセージ
マルチキャストアドレス	マルチキャストメッセージ

メッセージの受信時は，受信したメッセージの内容または送信元の相手アドレスから，必要に応じてメッセージの種類を判定してください。

(4) メッセージ送受信時の相手アドレスのチェック

TP1/NET/UDP は任意の相手と通信するため，メッセージ送受信時に相手アドレスのチェックを実施しません。

メッセージの送信時には，TP1/NET/UDP はユーザが設定した相手アドレスをそのまま UDP プロトコルに渡します。ユーザが不正な相手アドレスを設定した場合は，UDP プロトコルの相手アドレスのチェックによって送信に失敗する場合があります。

メッセージの受信時には，TP1/NET/UDP はメッセージに付加した制御ヘッダの所定の領域に相手アドレスを設定します。この相手アドレスから，送信元を識別してください。

(5) ブロードキャストメッセージの送信

(a) 最大メッセージ長

ブロードキャストメッセージを送信する場合の最大メッセージ長は，ご使用の OS によって異なります。詳しくは，OS のマニュアル，技術資料または実装を確認してください。

(6) マルチキャストメッセージの送信

(a) マルチキャストパケットの生存期間（TTL）

マルチキャストメッセージを送信する場合の TTL は，論理端末定義（mcftalcle -m）の multicastttl オペランドで指定します。ご使用のネットワーク構成に応じて適切な値を指定してください。このオペランドを省略した場合，TTL は OS が設定します。TTL のデフォルト値については，ご使用の OS のマニュアルを参照してください。

(7) マルチキャストメッセージの受信

(a) マルチキャストメッセージの受信に使用する IP アドレスまたはホスト名

自システムをホットスタンバイ構成とする場合，または，自システムがマルチホームドホスト形態（一つのマシン内に複数の IP アドレスが割り当てられている環境）の場合，マルチキャストメッセージの受信に使用する IP アドレスまたはホスト名を論理端末定義（mcftalcle -m）の ripaddr オペランドまたは rhostname オペランドに指定します。どちらも省略した場合，使用する IP アドレスは OS が設定します。

(b) 送信元特定マルチキャスト（SSM）

特定の相手システムから送信されたマルチキャストメッセージ（送信元特定マルチキャスト（SSM））だけを受信する場合，論理端末定義（mcftalcle -m）の shostname[1～8]オペランドに，マルチキャストメッセージ受信を許可する相手システムのホスト名または IP アドレスを指定します。指定した相手システムのホスト名や IP アドレスに誤りがある場合，論理端末の閉塞解除に失敗します。

異なるネットワーク上の相手システムから SSM を受信する場合，ネットワーク機器（ルータ等）のルーティングプロトコルに PIM-SSM を使用してください。また，SSM 用の IP アドレスは，232.0.0.0～232.255.255.255 と規定されていますが，実際に使用できるアドレスはご使用のネットワーク機器によって異なる場合があります。詳しくは，ネットワーク機器のマニュアル，技術資料または実装を確認してください。

2.2　AP 間通信メッセージの送受信

TP1/NET/UDP では，一方送信メッセージ，一方受信メッセージ，および同期型の送信メッセージを使用して，メッセージを送受信します。

この節では，メッセージの送受信の処理について説明します。さらに，TP1/NET/UDP がアプリケーション名を決定する処理手順について説明します。

2.2.1　一方送信メッセージの送信

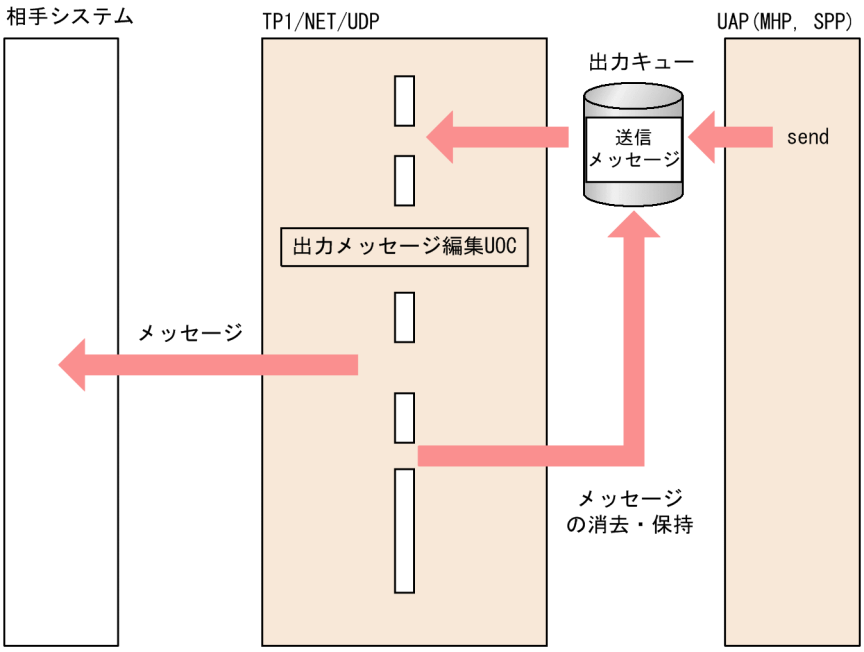
一方送信メッセージの送信は，自システムの UAP から**出力キュー**を介して相手システムへメッセージを送信する形態です。

UAP が一方送信メッセージの送信要求（send）を行うと，メッセージが出力キューに登録されます。TP1/NET/UDP は，出力キューからメッセージを取り出し，相手システムに送信します。送信処理が完了すると，出力キューがメモリキューの場合，TP1/NET/UDP 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。出力キューがディスクキューの場合，出力キューに送信済みメッセージのデータを保持します。保持した送信済みのメッセージは，メッセージの再送に使用できます。メッセージの再送については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

なお，一方送信メッセージの送信（send）が正常リターンしたことは，メッセージを出力キューに登録できたことを意味します。同期型メッセージの送信（sendsync）の場合と異なり，相手システムへの一方送信メッセージの送信要求が完了したというわけではありません。

一方送信メッセージの送信処理の流れを次の図に示します。

図 2-8　一方送信メッセージの送信処理の流れ



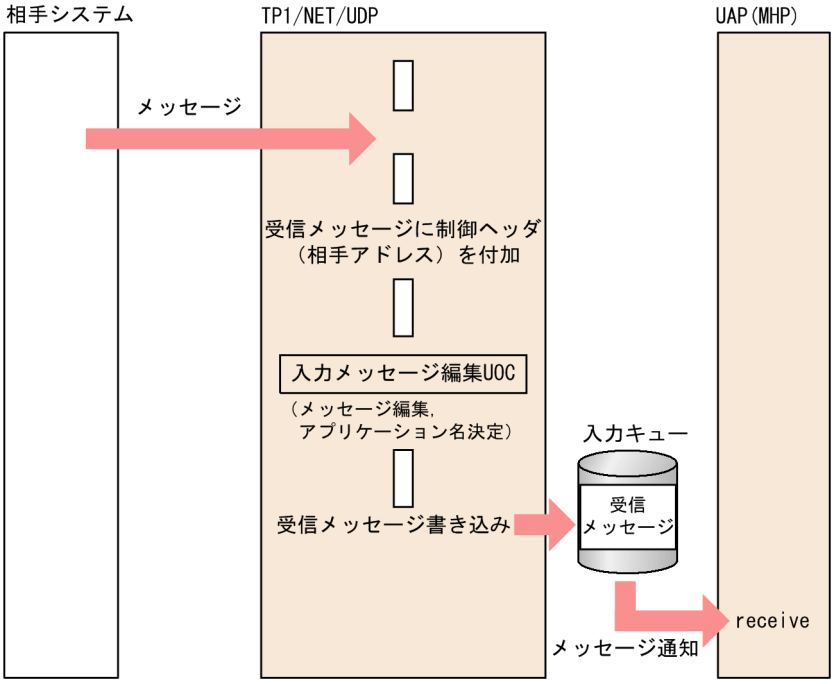
2.2.2　一方送信メッセージの受信

一方送信メッセージの受信は，**入力キュー**を介して相手システムからのメッセージを受信する形態です。

TP1/NET/UDP は，メッセージの受信を通知されると，UDP プロトコルからメッセージを取り出して受信バッファに格納します。その際，受信したメッセージ内に，UDP プロトコルから通知された相手アドレスを制御ヘッダとして設定した上で，入力キューにメッセージを書き込みます。入力キューを監視している MCF は，入力キューへのメッセージ書き込みを契機に，アプリケーション名に対応する UAP に受信メッセージを通知（receive）します。

一方送信メッセージの受信処理の流れを次の図に示します。

図 2-9　一方送信メッセージの受信処理の流れ



受信バッファ数および入力キューの容量は，システムのトラフィック量に応じて見積もってください。

メッセージを受信したときに受信バッファが不足したり，入力メッセージ最大格納数を超過した場合，論理端末を閉塞するかどうかを，論理端末定義（mcftalcle -f）の rcvoverflow オペランドで指定できます。

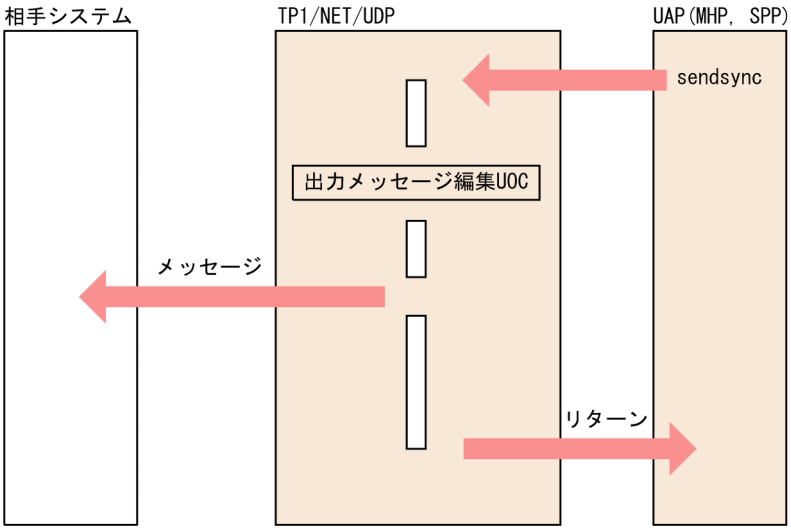
2.2.3　同期型メッセージの送信

同期型メッセージの送信は，出力キューを介さないで相手システムにメッセージを送信する形態です。

同期型メッセージの送信要求（sendsync）は UDP プロトコルに対する送信処理の完了を契機にリターンします。TP1/NET/UDP では，UDP プロトコルに対する送信要求が即時にリターンするため，同期型メッセージの送信要求を受け付けてからリターンするまでの時間監視は実施しません。

同期型メッセージの送信処理の流れを次の図に示します。

図 2-10　同期型メッセージの送信処理の流れ

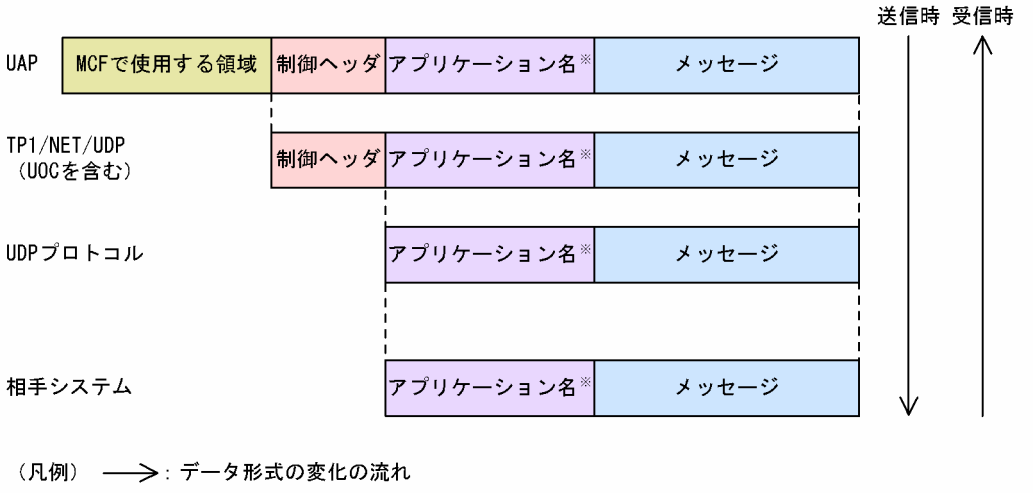


2.2.4　送受信するメッセージのデータ形式

TP1/NET/UDP を使用した AP 間通信で送受信するメッセージは，自システムの UAP と相手システムの間の受け渡しの過程でデータ形式が変化します。

送受信するメッセージのデータ形式の変化を次の図に示します。

図 2-11　メッセージのデータ形式



注※

アプリケーション名は，必要な場合だけ設定します。

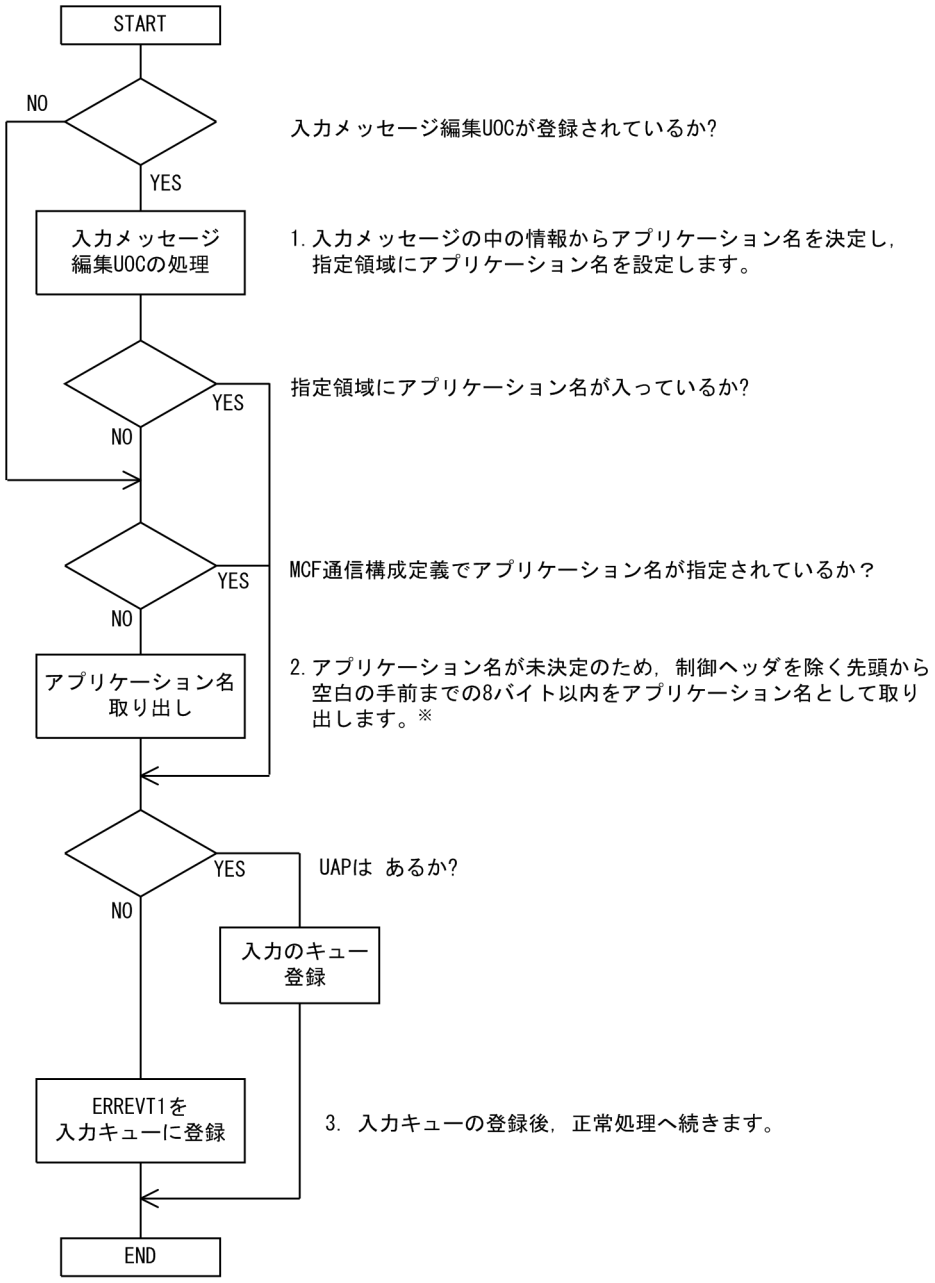
2.2.5　アプリケーション名の決定

TP1/NET/UDP 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 での決定
2. 論理端末定義（mcftalcle -v）での決定
3. 入力メッセージの先頭から空白の手前までの 8 バイト以内をアプリケーション名として決定

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 2-12　アプリケーション名の決定の処理手順



注※
　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。

3 C 言語のライブラリ関数

この章では，TP1/NET/UDP 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/UDP で使用する C 言語のライブラリ関数の一覧を，次の表に示します。

表 3-1　C 言語のライブラリ関数の一覧

関数名	機能
dc_mcf_receive	一方送信メッセージの受信
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_sendsync	同期型メッセージの送信
dc_mcf_tactle	論理端末の閉塞解除
dc_mcf_tdctle	論理端末の閉塞
dc_mcf_tlsle	論理端末の状態取得

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_receive － 一方送信メッセージの受信（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。

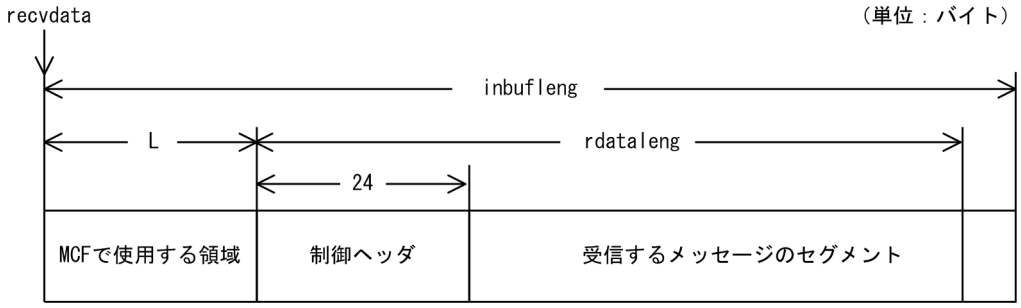
受信できるメッセージの一つのセグメントの最大長は，制御ヘッダの 24 バイトを含めて 65551 バイトまでです。

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/UDP 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。



制御ヘッダ（dcmudp_dc_header）は<dcmudpu.h>に，次のように定義されています。

```
typedef struct {
    unsigned  char   r_exipaddr[12];       …予備
    DCULONG          r_ipaddr;             …相手IPアドレス
    unsigned  char   rsv1[2];              …予備
    unsigned  short  r_portno;             …相手ポート番号
    unsigned  char   rsv2[4];              …予備
} dcmudp_dc_header;
```

相手 IP アドレスと相手ポート番号の設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

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指定します。

```
{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 から値が返ります。

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。

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を受信する場合は，ここで返された論理端末名称を termnam に設定してください。

●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。 このリターン値が返されたあとに，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_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するとともに，メッセージキューファイルの容量および保持メッセージ数（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）に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，一般か優先かどうか，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になります。

●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。論理端末名称の長さは最大８バイトです。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。
DCMCFRID_PRIO を設定した場合は，otermnam に出力先の論理端末名称を設定できません。

●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になります。

●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。論理端末名称の長さは最大８バイトです。論理端末名称の後ろ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。

リターン値	リターン値（数値）	意味
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。
		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。
		opcd 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。
		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。

リターン値	リターン値（数値）	意味
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 オプションおよび-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。指定内容によっては，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。同じ出力通番を持つメッセージが存在する場合，どのメッセージを再送するかは保証できません。

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

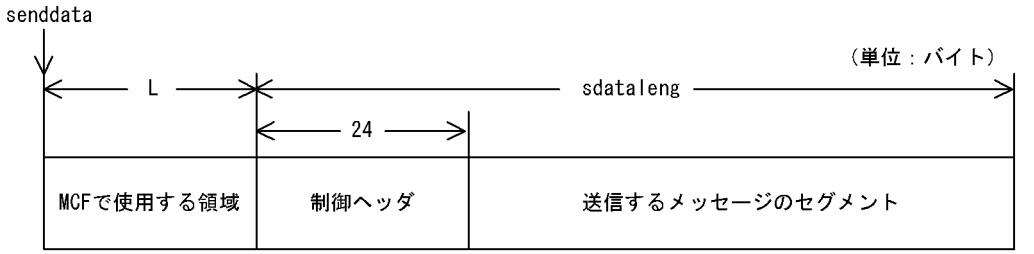
```
#include <dcmcf.h>
int  dc_mcf_send(action, commform, termnam, resv01, senddata,
                 sdataleng, resv02, opcd)
DCLONG          action;
DCLONG          commform;
char            *termnam;
char            *resv01;
char            *senddata;
DCLONG          sdataleng;
char            *resv02;
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，制御ヘッダの 24 バイトを含めて 32000 バイトまでです。

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。



制御ヘッダ（dcmudp_dc_header）は<dcmudpu.h>に，次のように定義されています。

```
typedef struct {
    unsigned  char   r_exipaddr[12];       …予備
    DCULONG          r_ipaddr;             …相手IPアドレス
    unsigned  char   rsv1[2];              …予備
```

```
    unsigned  short  r_portno;              …相手ポート番号
    unsigned  char   rsv2[4];               …予備
} dcmudp_dc_header;
```

相手 IP アドレスと相手ポート番号の設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

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，一般か優先かどうか，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 〔| {DCMCFNORM|DCMCFPRIO} 〕 〔| {DCMCFSEQ|DCMCFNSEQ} 〕 〔| {DCMCFBUF1|DCMCFBUF2} 〕
```

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

　出力通番を付ける場合に設定します。

DCMCFNSEQ

　出力通番を付け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

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送信領域を設定します。制御ヘッダの 24 バイトを含めて 32000 バイトまで送信できます。

送信領域の制御ヘッダ内の予備領域は，必ずすべて 0 に設定してください。

●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	termnam に設定した論理端末名称が間違っています。

リターン値	リターン値（数値）	意味
DCMCFRTN_72001	-13001	termnam に設定した論理端末名称は，定義されていません。
		dc_mcf_send 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。
		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。
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，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。

注意事項

MCF が UAP の送信要求を TP1/NET/UDP にスケジュールする優先順位は，高い方から同期型メッセージの送信，一方送信メッセージの送信（優先），一方送信メッセージの送信（一般）です。同一論理端末上で同期型メッセージの送信と一方送信メッセージの送信を併用する場合，送信メッセージの追い越しが発生する場合があります。

このため，同一論理端末上での，dc_mcf_send 関数と dc_mcf_sendsync 関数の併用は避けてください。同様に，COBOL 言語およびデータ操作言語での，同期型メッセージの送信と一方送信メッセージの送信の併用は避けてください。

併用する場合は，dc_mcf_sendsync 関数の送信メッセージが dc_mcf_send 関数の送信メッセージを追い越した場合でも，問題が生じないように運用してください。

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

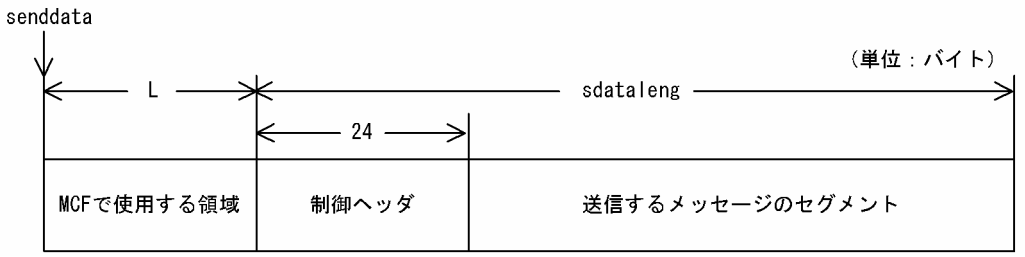
```
#include <dcmcf.h>
int dc_mcf_sendsync (action, commform, termnam, resv01,
                     senddata, sdataleng, resv02, opcd,
                     watchtime)
DCLONG          action;
DCLONG          commform;
char            *termnam;
char            *resv01;
char            *senddata;
DCLONG          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，制御ヘッダの 24 バイトを含めて 32000 バイトまでです。

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。



制御ヘッダ（dcmudp_dc_header）は<dcmudpu.h>に，次のように定義されています。

```
typedef struct {
    unsigned  char    r_exipaddr[12];        …予備
```

```
    DCULONG          r_ipaddr;              …相手IPアドレス
    unsigned  char   rsv1[2];               …予備
    unsigned  short  r_portno;              …相手ポート番号
    unsigned  char   rsv2[4];               …予備
} dcmudp_dc_header;
```

相手 IP アドレスと相手ポート番号の設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

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，および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

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送信領域を設定します。制御ヘッダの 24 バイトを含めて 32000 バイトまで送信できます。

送信領域の制御ヘッダ内の予備領域は必ずすべて 0 に設定してください。

●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

TP1/NET/UDP では時間監視を行わないため，負の値を設定してください。

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 に設定した論理端末名称は，定義されていません。
		dc_mcf_sendsync 関数を呼び出せない論理端末を設定しています。
DCMCFRTN_72016	-13016	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_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，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_73001	-14001	出力先の論理端末で障害が発生しました。
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_73011	-14011	sdataleng に 1 以上 24 以下の値を設定しています。
DCMCFRTN_73019	-14019	システムに対する送信時に障害が発生しました。
DCMCFRTN_73020	-14020	出力先の論理端末は閉塞しています。
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。

注意事項

同一論理端末上で，同期型メッセージの送信と一方送信メッセージの送信を併用するのは避けてください。詳細は，この章の「dc_mcf_send − 一方送信メッセージの送信（C 言語）」を参照してください。

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/UDP 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。

- 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

  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。
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。

リターン値	リターン値（数値）	意味
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_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/UDP 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論理端末の情報を，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。

- 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

  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。
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。

リターン値	リターン値（数値）	意味
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。
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_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論理端末の情報を，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。

- 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

  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。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。

- 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。
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。

リターン値	リターン値（数値）	意味
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。

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/UDP 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/UDP 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	'RECEIVE△'	一方送信メッセージの受信
	'RESEND△△'	メッセージの再送
	'SEND△△△△'	一方送信メッセージの送信
	'SENDSYNC'	同期型メッセージの送信
	'TACTLE△△'	論理端末の閉塞解除
	'TDCTLE△△'	論理端末の閉塞
	'TLSLE△△△'	論理端末の状態取得

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 CALL インタフェース
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	

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/UDP 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.
STATUS KEY	B	B	B	B
SYMBOLIC TERMINAL	B	U_1※1	U_1※2	U_1
MESSAGE DATE	B	B	−	−
MESSAGE TIME	B	B	−	−
SYNCHRONOUS MODE	U_2	U_2	U_2	U_1
SWITCHING MODE	−	−	U_2	−
DETAIL MODE	−	−	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.：一方送信メッセージの単一セグメントの非同期送信（SEND）

4.：単一セグメントの同期送信（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。

注※1

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。

注※2

FOR 句に I-O を設定し，UAP 共通定義（mcfmuap -c）の noansreply オペランドに yes を指定し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

注※

TP1/NET/UDP ではサポートしていない関数です。

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※1※2
－	○	○	－	－	dc_mcf_sendsync
－	○	○	－	○	dc_mcf_sendrecv※1

（凡例）

○：指定あり

－：指定なし

注※1

TP1/NET/UDP ではサポートしていない関数です。

注※2

UAP 共通定義（mcfmuap -c）の noansreply オペランドに yes を指定した場合は dc_mcf_send 関数に対応します。

CBLDCMCF('RECEIVE ') － 一方送信メッセージの受信（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W1   PIC X(12)  VALUE LOW-VALUE.
    02  データ名W2   PIC X(4).
    02  データ名W3   PIC X(2)  VALUE LOW-VALUE.
    02  データ名W4   PIC X(2).
    02  データ名W5   PIC X(4)  VALUE LOW-VALUE.
    02  データ名W6   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。

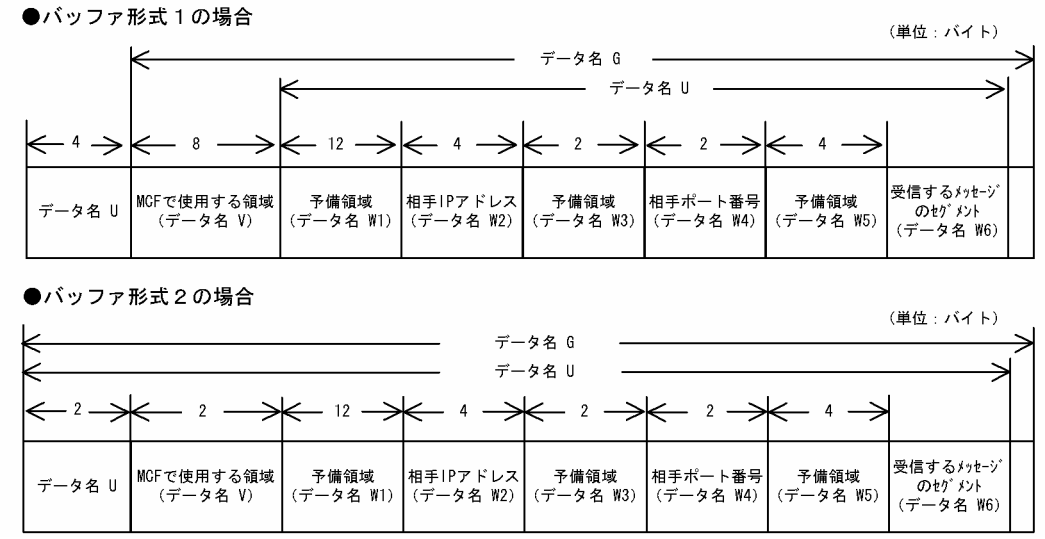
受信できるメッセージの一つのセグメントの最大長は，制御ヘッダの 24 バイトを含めて 65551 バイトまでです。

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/UDP 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。



相手 IP アドレスと相手ポート番号の設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

UAP で値を設定するデータ領域

●データ名 A

メッセージの受信を示す要求コードを「VALUE 'RECEIVE△'」と設定します。

●データ名 C

メッセージ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

中間セグメントまたは最終セグメントを受信する場合，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 バイトに満たない場合，論理端末名称の後ろを空白で埋めてください。

先頭セグメントまたは単一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。

●データ名 W1，データ名 W3，データ名 W5

予備領域です。

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，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 W2

送信元の IP アドレスが返されます。

●データ名 W4

送信元のポート番号が返されます。

●データ名 W6

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。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。
	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。

ステータスコード	意味
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('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，メッセージキューファイルの容量および保持メッセージ数（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）に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 M6，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バイトに満たない場合，論理端末名称の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バイトに満たない場合，論理端末名称の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になります。

●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になります。

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。

ステータスコード	意味
70904	• 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）。
	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，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('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。
	データ名 P に設定した論理端末名称は，定義されていません。
	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。

ステータスコード	意味
72016	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 オプションおよび-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，CBLDCMCF（'RESEND△△'）で使用する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，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。同じ出力通番を持つメッセージが存在する場合，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

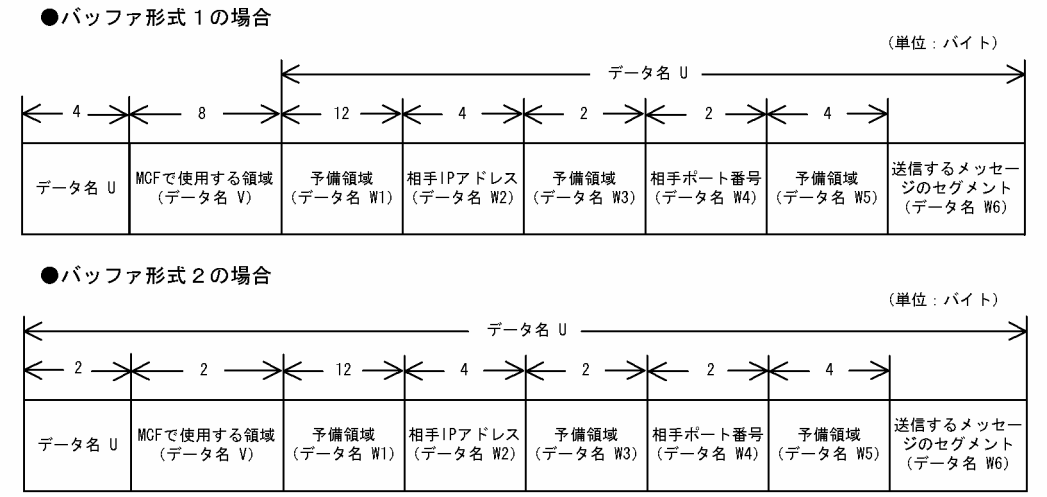
```
01  一意名1.
    02  データ名A    PIC X(8)  VALUE ’SEND    ’.
    02  データ名B    PIC X(5).
    02  FILLER       PIC X(3).
    02  データ名C    PIC X(4)  VALUE SPACE.
    02  データ名D    PIC X(4)  VALUE SPACE.
    02  データ名E    PIC 9(8).
    02  データ名F    PIC 9(8).
    02  データ名G    PIC 9(9)  COMP VALUE ZERO.
    02  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W1   PIC X(12)  VALUE LOW-VALUE.
    02  データ名W2   PIC X(4).
    02  データ名W3   PIC X(2)  VALUE LOW-VALUE.
    02  データ名W4   PIC X(2).
    02  データ名W5   PIC X(4)  VALUE LOW-VALUE.
    02  データ名W6   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，制御ヘッダの 24 バイトを含めて 32000 バイトまでです。

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。



相手 IP アドレスと相手ポート番号の設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

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

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 バイトに満たない場合，論理端末名称の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＋ 24（制御ヘッダ）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＋ 28（制御ヘッダ，データ名 U，データ名 V）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 W1，データ名 W3，データ名 W5

0 を設定します。

●データ名 W2

送信相手の IP アドレスを設定します。

設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

●データ名 W4

送信相手のポート番号を設定します。

設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

●データ名 W6

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 31976 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。
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。
	データ名 P に設定した論理端末名称は，定義されていません。
	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 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。
	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。

ステータスコード	意味
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。

注意事項

同一論理端末上で，同期型メッセージの送信と一方送信メッセージの送信を併用するのは避けてください。詳細は，3 章の「dc_mcf_send － 一方送信メッセージの送信（C 言語）」を参照してください。

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)  VALUE ’EMI ’.
    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 S9(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W1   PIC X(12)  VALUE LOW-VALUE.
    02  データ名W2   PIC X(4).
    02  データ名W3   PIC X(2)  VALUE LOW-VALUE.
    02  データ名W4   PIC X(2).
    02  データ名W5   PIC X(4)  VALUE LOW-VALUE.
    02  データ名W6   PIC X(n).
```

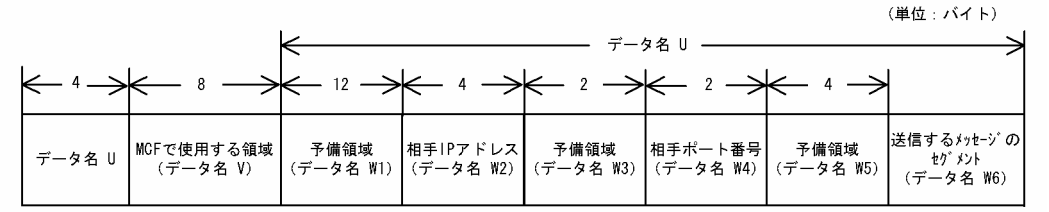
機能

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。

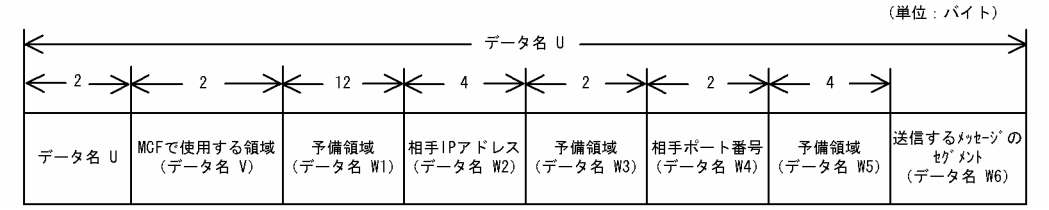
送信できるメッセージの一つのセグメントの最大長は，制御ヘッダの 24 バイトを含めて 32000 バイトまでです。

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



相手 IP アドレスと相手ポート番号の設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

UAP で値を設定するデータ領域

●データ名 A

同期型メッセージの送信を示す要求コードを「VALUE 'SENDSYNC'」と設定します。

●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')を呼び出しから終了するまでの監視時間を設定します。TP1/NET/UDP では時間監視を行わないため，負の値を設定してください。

●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

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 バイトに満たない場合，論理端末名称の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＋ 24（制御ヘッダ）を設定します。

【バッファ形式 2 の場合】 PIC9(4)

送信するセグメントの長さ＋ 28（制御ヘッダ，データ名 U，データ名 V）を設定します。

●データ名 V

【バッファ形式 1 の場合】 PICX(8)

【バッファ形式 2 の場合】 PICX(2)

MCF で使用する領域です。

●データ名 W1，W3，W5

0 を設定します。

●データ名 W2

送信相手の IP アドレスを設定します。

設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

●データ名 W4

送信相手のポート番号を設定します。

設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

●データ名 W6

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 31976 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。

ステータスコード	意味
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 に設定した論理端末名称は，定義されていません。
	CBLDCMCF('SENDSYNC')を呼び出せない論理端末を設定しています。
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。
73001	出力先の論理端末で障害が発生しました。
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。
73011	バッファ形式 1 の場合はデータ名 U に 1 以上 24 以下の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 5 以上 28 以下の値を設定しています。
73019	システムに対する送信時に障害が発生しました。
73020	出力先の論理端末は閉塞しています。
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。

注意事項

同一論理端末上で，同期型メッセージの送信と一方送信メッセージの送信を併用するのは避けてください。詳細は，3 章の「dc_mcf_send － 一方送信メッセージの送信（C 言語）」を参照してください。

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/UDP 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△△△')を用いて論理端末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。

●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。
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。

ステータスコード	意味
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('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/UDP 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△△△')を用いて論理端末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。

●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。
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。

ステータスコード	意味
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('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)  LOW-VALUE.
      03 データ名P   PIC X(4).
      03 データ名Q   PIC X(40) 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。

●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，およびデータ名 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。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。

●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。

VALUE 'ACT△'

閉塞解除状態

VALUE '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 以外の値が設定されています。

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  {ASYNC|データ名6} 〕.
```

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C1  PIC X(12)  VALUE LOW-VALUE.
    02  データ名C2  PIC X(4).
    02  データ名C3  PIC X(2)  VALUE LOW-VALUE.
    02  データ名C4  PIC X(2).
    02  データ名C5  PIC X(4)  VALUE LOW-VALUE.
    02  データ名C6  PIC X(n).
```

PROCEDURE DIVISION（通信文）の指定

```
RECEIVE  通信記述名
     〔FIRST〕  SEGMENT
          INTO 一意名1.
```

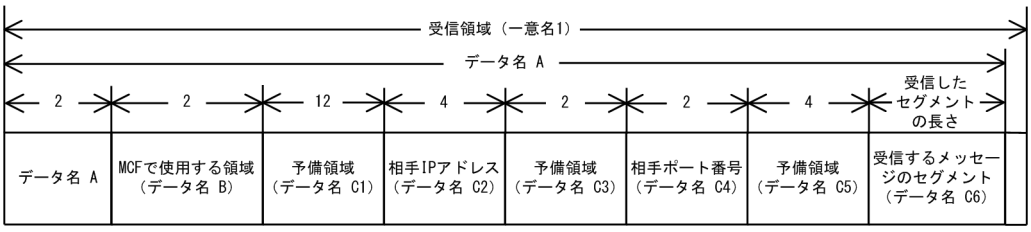
機能

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，制御ヘッダの 24 バイトを含めて 32763 バイトです。

- 一方送信メッセージの受信　CBLDCMCF('RECEIVE△')

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。

（単位：バイト）



相手 IP アドレスと相手ポート番号の設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

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 または I-O

　非同期型のメッセージの受信

●SYMBOLIC TERMINAL 句

中間セグメントまたは最終セグメントを受信する場合，データ名 2 に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 バイトに満たない場合，論理端末名称の後ろを空白で埋めてください。

先頭セグメントまたは単一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

非同期型でのメッセージ受信を示す，次のどちらかの値を設定してください。

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

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。

●データ名 B

MCF で使用する領域です。

●データ名 C1，データ名 C3，データ名 C5

予備領域です。

●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りません。データ名 1 にステータスコードが返されます。

●SYMBOLIC TERMINAL 句

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 C2

送信元の IP アドレスが返されます。

●データ名 C4

送信元のポート番号が返されます。

●データ名 C6

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 32739 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。

ステータスコード	意味
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 句に設定した論理端末名称は，定義されていません。
	RECEIVE 文を実行できない論理端末を設定しています。
	SYNCHRONOUS MODE 句に SYNC が設定されているか，データ名 6 に'1'が設定されています。
72013	一意名 1 のサイズを超えるセグメントを受信しました。一意名 1 のサイズを超える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	通信文に WAITING 句が設定されています。
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	一意名 1 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。
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} 〕
  〔DETAIL MODE IS データ名10〕.
```

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C1  PIC X(12)  VALUE LOW-VALUE.
    02  データ名C2  PIC X(4).
    02  データ名C3  PIC X(2)  VALUE LOW-VALUE.
    02  データ名C4  PIC X(2).
    02  データ名C5  PIC X(4)  VALUE LOW-VALUE.
    02  データ名C6  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} 〕.
```

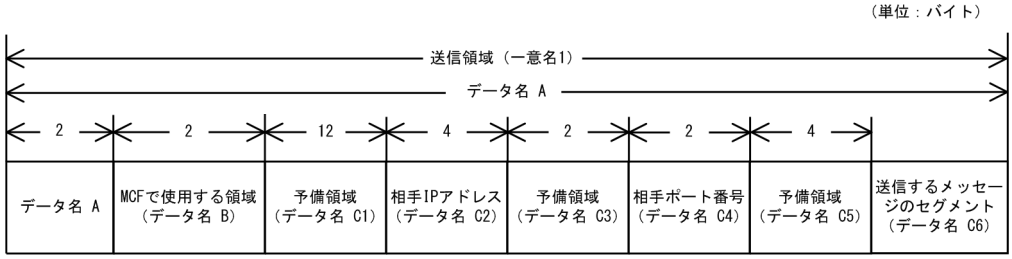
機能

次に示す CALL インタフェースの機能を実現します。

- 一方送信メッセージの送信　CBLDCMCF('SEND△△△△')
- 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，制御ヘッダの 24 バイトを含めて 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。



相手 IP アドレスと相手ポート番号の設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

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

同期型メッセージの送信または一方送信メッセージの送信

一方送信メッセージの送信を行う場合，UAP 共通定義（mcfmuap -c）の noansreply オペランドに yes を指定してください。no を指定した場合，SEND 文はステータスコード（72000）を返します。

●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 バイトに満たない場合，論理端末名称の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

同期型メッセージの送信

同期型メッセージの送信のとき設定します。

ASYNC

非同期型メッセージの送信

一方送信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'△'：非同期型メッセージの送信

'1'：同期型メッセージの送信

省略した場合は，ASYNC（非同期型メッセージの送信）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 FOR 句に I-O を設定した場合は，NORMAL（一般の一方送信メッセージ）が設定されます。

●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 FOR 句に I-O を設定した場合は，出力通番を付けません。

●データ名 A

送信するセグメントの長さ＋ 28（制御ヘッダ，データ名 A，データ名 B）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 C1

予備領域です。

●データ名 C2

送信相手の IP アドレスを設定します。

設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

●データ名 C3

予備領域です。

●データ名 C4

送信相手のポート番号を設定します。

設定方法については，「2.1.3　通信相手のアドレスの指定」を参照してください。

●データ名 C5

予備領域です。

●データ名 C6

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 31976 バイトまで送信できます。

●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。

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。
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。
	データ名 A 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	＜ 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。

ステータスコード	意味
72000	• FOR 句に I-O を設定した一方送信メッセージを送信する SEND 文を実行していますが，UAP 共通定義（mcfmuap -c）の noansreply オペランドに no を指定しています。
	＜ 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，FOR 句に I-O を設定した一方送信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 句に設定した論理端末名称は，定義されていません。
	SEND 文を実行できない論理端末を設定しています。
72017	DETAIL MODE 句に設定した値が間違っています。
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。
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。
73001	出力先の論理端末で障害が発生しました。
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。
73011	データ名 A に 5 以上 28 以下の値を設定しています。
73019	システムに対する送信時に障害が発生しました。
73020	出力先の論理端末は閉塞しています。
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。

注意事項

- データ操作言語の SEND 文を使用してメッセージを送信する場合は，時間監視を行いません。したがって，MCF マネジャ定義の UAP 共通定義（mcfmuap コマンド）で同期送信監視時間を指定する際には，0 を指定してください。
- 同一論理端末上で，同期型メッセージの送信と一方送信メッセージの送信を併用するのは避けてください。詳細は，3 章の「dc_mcf_send － 一方送信メッセージの送信（C 言語）」を参照してください。

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/UDP に関連するユーザオウンコーディング，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（UOC）**といいます。独自の処理でメッセージを編集したり，アプリケーション名を決定したりする場合は，ユーザが UOC を作成してください。

TP1/NET/UDP で使用できる UOC を次に示します。

- 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

UOC は C 言語で作成します。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。

5.1.1　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。また，受信した論理メッセージを基に，ユーザ任意のアプリケーション名を決定できます。

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。ただし，MCF イベント発生時と，UAP からのアプリケーションプログラム起動時は UOC は起動しません。

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，必要に応じて論理端末定義（mcftalcle -e）で，メッセージ編集用バッファグループ番号を定義します。

(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 MCF 通信構成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。

(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。

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，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は「2.2.5　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，エラー処理 UAP を起動したい場合は，ユーザ任意のエラー処理 UAP 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，または DCMCF_UOC_MSG_OK_RCV でリターンします。この場合，MCF は正常なメッセージとして処理するため，受信メッセージの破棄などの障害処理はしません。

(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。入力メッセージの編集 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/UDP 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。

5.1.2　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>
DCLONG   uoc_func(dcmcf_uoc_min_n *parm)
```

K&R 版 C の場合

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
DCLONG   uoc_func(parm)

dcmcf_uoc_min_n  *parm ;
```

(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;           …MCF使用領域
    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使用領域
    DCMLONG buff_addr;              …MCF使用領域
```

```
    char    reserve2[4];            …予備
} dcmcf_uocbufinf_n;
```

(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_UDP

  UDP 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

  一方受信メッセージ

- 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。

  メッセージ編集用バッファが未定義の場合，つまり，論理端末定義（mcftalcle）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

  論理端末定義（mcftalcle）の-v オプションで指定したアプリケーション名が設定されます。

  論理端末定義（mcftalcle）の-v オプションを省略した場合，受信したメッセージに指定されたアプリケーション名が設定されます。このアプリケーション名は，メッセージの送信時に，制御ヘッダの直後に 8 バイト以内で設定された名称です。9 バイト目までにデリミタ'0x20'（スペース）が設定されている必要があります。

(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。

(5) ユーザが値を設定する項目

(a) dcmcf_uoc_min_n

- aplname
  UOC で決定したアプリケーション名を設定します。
- 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 を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 の範囲で設定してください。

(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を設定します。
  使用バッファ情報数には，1 を設定してください。

(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

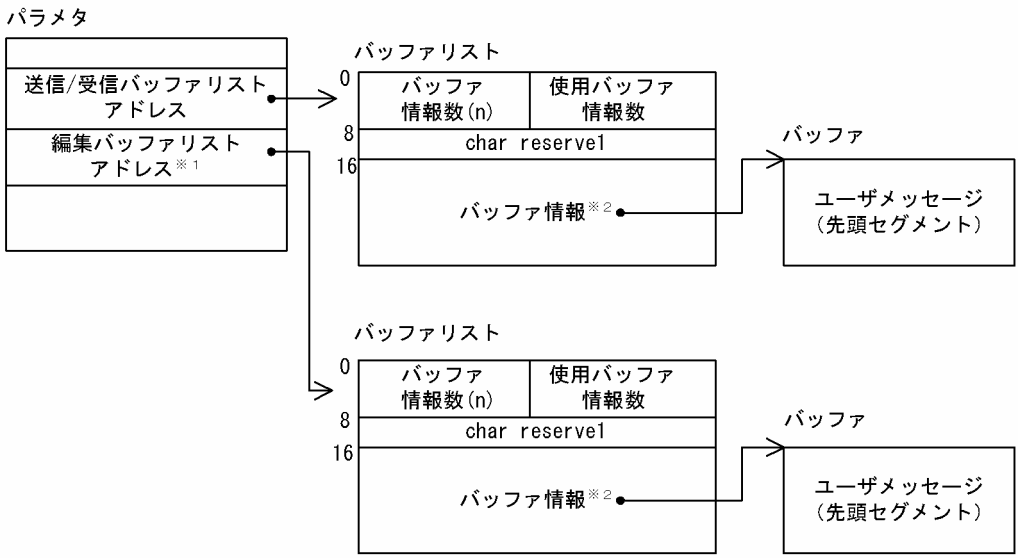
uoc_func()は次のコードでリターンしてください。

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 5-1 UOC インタフェース用のパラメタとバッファの関係

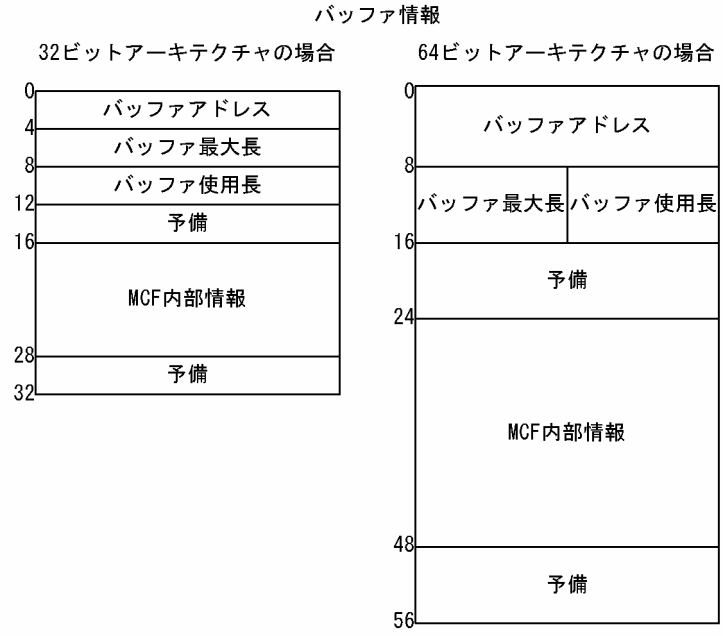


注※1

論理端末定義（mcftalcle -e）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。



5.1.3　出力メッセージの編集

出力メッセージの編集 UOC は，送信する論理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，必要に応じて論理端末定義（mcftalcle -e）で，メッセージ編集用バッファグループ番号を定義します。

(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 MCF 通信構成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，送信するメッセージのセグメントは，送信バッファ，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。

(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

入力メッセージ編集 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　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

「5.1.2　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_UDP

  　UDP 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。
- 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。
  メッセージ編集用バッファが未定義の場合，つまり，論理端末定義（mcftalcle）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。

  'n'
  一般の一方送信メッセージ

  'p'
  優先の一方送信メッセージ

  's'
  同期型送信メッセージ
- 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にします。
- 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）

「5.1.2 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 を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 の範囲で設定してください。

(b) dcmcf_uocbuff_list_n（バッファリスト），dcmcf_uocbufinf_n（バッファ情報）

「5.1.2 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。
「5.1.2(7) パラメタとバッファの関係」を参照してください。

5.1.5 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。送信メッセージの通番編集 UOC は，UAP が send 関数または 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	正常リターン
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。

(d) メイン関数への登録例

UAP のメイン関数への登録例を次に示します。

- 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();
}
```

- 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();
}
```

5.1.6　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

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。

5.1.7 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設計してください。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／システムコール（例：shmctl，shmget，shmop）
- メモリ操作（例：memcpy）
- 文字列操作（例：strcpy）

- 時間取得関数

(3) UOC の異常処理

TP1/NET/UDP の UOC で異常を検出した場合，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 が実行されてもよいように作成してください。

5.2 MCF イベントインタフェース

OpenTP1 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の編集 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/UDP 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/UDP 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名がなかったことを報告します。
メッセージ廃棄通知イベント	ERREVT2	次の理由で，受信メッセージを廃棄しました。 • 入力キューに障害が発生しました。 • 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ました。 • MHP に受信したセグメントを渡す前に MHP の異常終了が発生しました。 • 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。 • アプリケーションの即時起動時に障害が発生しました。 • MHP のアプリケーション，サービスグループがセキュア状態です。	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 に受信したセグメントを渡したあとに，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。	未処理送信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
未処理送信メッセージ廃棄通知イベント	ERREVTA	• 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。
障害通知イベント	CERREVT	論理端末障害が発生しました。	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	論理端末が閉塞解除しました。	論理端末が閉塞解除したことを報告します。
	CCLSEVT	論理端末が正常に閉塞しました。	論理端末が閉塞したことを報告します。

注※

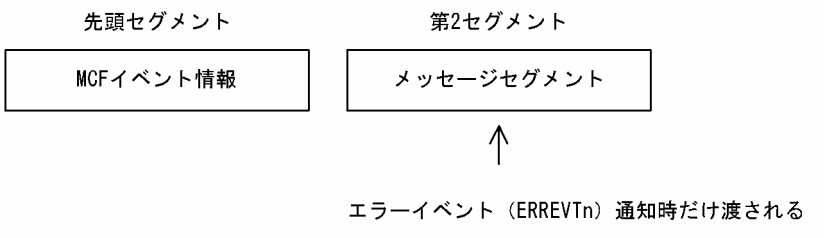
MCF アプリケーション定義（mcfaalcap -g）の recvmsg オペランドに r を指定した場合，または dc_mcf_rollback の action に DCMCFRTRY または DCMCFRRTN を指定した場合は除きます。

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（ERREVTn）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 5-2　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（ERREVTn）で使用する構造体は，<dcmcf.h>で定義してあります。

COPNEVT，CCLSEVT および CERREVT で使用する構造体は，<dcmudpu.h>で定義されています。また，dc_mcf_evtheader は，<dcmcf.h>で定義されているので，<dcmudpu.h>の前に<dcmcf.h>を取り込んでおいてください。

(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] ;             … 論理端末名称
  char cn_name[9] ;              … MCF使用領域
  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，または ERREVT3 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。

  ERREVT2 および ERREVT3 で，次に示す場合には，'*'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 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が設定されます。

  COPNEVT，CCLSEVT では，閉塞解除または閉塞した論理端末名称が設定されます。

- 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。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。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Ⅰ　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。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。

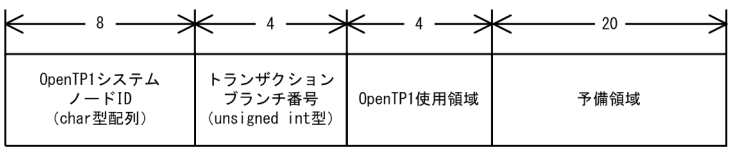
- serv_grp_name

  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。

（単位：バイト）



(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) 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)
    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   reserve1[48];        ... 予備
} dcmudp_cerrevt ;
```

(b) MCF イベントとして設定される項目

- err_fact

  CERREVT の障害要因コードが，次に示す値で設定されます。

  $(00000031)_{16}$

  論理端末障害発生

- 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 I　理由コード一覧」を参照してください。

(7)　COPNEVT，CCLSEVT

(a)　形式

```
typedef struct {
  struct dc_mcf_evtheader  header ;
                                ... MCFイベント共通ヘッダ
  char  reserve1[60];           ... 予備
} dcmudp_statevt ;
```

(b)　MCF イベントとして設定される項目

ありません。

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	−	−
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	次に示すどちらかが設定されます。 • 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	−	−

項目	位置(バイト)		長さ(バイト)	属性	内容
	形式 1	形式 2			
予備	64	60	8	−	−
コネクション名	72	68	8	−	MCF が使用します。
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	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	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	−	−
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	−	−

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	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	−	MCF が使用します。
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	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	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	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※

理由コードの内容については，「付録Ｉ　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。

項目	位置(バイト)		長さ(バイト)	属性	内容
	形式 1	形式 2			
入力元論理端末名称	12	8	8	英数字	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。
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	−	MCF が使用します。
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	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	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。

項目	位置(バイト)		長さ(バイト)	属性	内容
	形式 1	形式 2			
トランザクションブランチ ID (BID)	192	188	36	英数字	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	英数字	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	−	MCF が使用します。
予備	80	76	16	−	−

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	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	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（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	障害の発生した論理端末名称が設定されます
予備	16	16	－	－
メッセージが入力された日付	32	8	外部 10 進	CERREVT を通知した日付です。 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	40	8	外部 10 進	CERREVT を通知した時刻です。 HHMMSS00 の形式です。

項目	位置 (バイト)	長さ (バイト)	属性	内容
メッセージが入力された時刻	40	8	外部 10 進	HH：時 MM：分 SS：秒 00 は固定です。
障害要因コード	48	4	2 進	障害要因コードが設定されます。 $(00000031)_{16}$：論理端末障害
理由コード 1※	52	4	2 進	理由コード 1 が設定されます。
理由コード 2※	56	4	2 進	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 については，「付録 I　理由コード一覧」を参照してください。

表 5-8　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	入力元論理端末名称が設定されます。
予備	16	16	－	－
メッセージが入力された日付	32	8	外部 10 進	COPNEVT，CCLSEVT を通知した日付です。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	40	8	外部 10 進	COPNEVT，CCLSEVT を通知した時刻です。HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
予備	48	60	－	－

（凡例）

－：該当しません。または，使用されません。

6 システム定義

この章では，UDP 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/UDP 固有のシステム定義，およびシステム定義例について説明します。

TP1/NET/UDP の定義の概要

TP1/NET/UDP 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/UDP 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/UDP 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。表 6-1 の定義の中で，TP1/NET/UDP に固有の定義があるものは，「MCF 通信構成定義」だけです。

この章では，TP1/NET/UDP 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，「mcftalcle（論理端末定義の開始）」の注意事項に記載してあります。

TP1/NET/UDP 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/UDP 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/UDP 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/UDP 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/UDP に固有の定義の一覧を次の表に示します。

表 6-2　TP1/NET/UDP 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 通信構成定義	共通定義	プロトコル共通のコマンドだけで指定できます。共通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。				
	プロトコル固有定義	mcftalcle （論理端末定義の開始） 指定数:1～512	-l	－	論理端末名称	1～8 文字の識別子
			-p	－	プロトコルの種別	udp
			-t	－	論理端末の端末タイプ	any
			-i	－	論理端末の閉塞解除方法	auto | 《manual》
			-v	－	アプリケーション名	1～8 文字の識別子
			-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))
			-s	syssndsize	システムのメッセージ送信バッファ長	符号なし整数((1～2147483647))（単位：バイト）
				sysrcvsize	システムのメッセージ受信バッファ長	符号なし整数((1～2147483647))（単位：バイト）
			-r	ipaddr	自システムの 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	自システムのポート番号	符号なし整数((1～65535))
				reuse	同一ポートを共用するかどうかを指定	yes | 《no》
			-m	multicast	マルチキャスト通信機能を使用するかどうかを指定	yes | 《no》

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	mcftalcle （論理端末定義の開始） 指定数:1～512	-m	hostgroupname	参加するホストグループのホスト名	1～255 文字のホスト名
				hostgroupaddr	参加するホストグループの IP アドレス	符号なし整数((0～255))(nnn.nnn.nnn.nnn)
				ripaddr	自システムの IP アドレス	符号なし整数((0～255))(nnn.nnn.nnn.nnn)
				rhostname	自システムのホスト名	1～255 文字のホスト名
				multicastttl	マルチキャストパケットの生存期間（TTL）	符号なし整数((0～255))
				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》
				shostname1 shostname2 shostname3 shostname4 shostname5 shostname6 shostname7 shostname8	マルチキャスト送信元システムのホスト名または IP アドレス	1～255 文字のホスト名または符号なし整数((0～255))(nnn.nnn.nnn.nnn)
			-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
			-f	rcvoverflow	入力メッセージ最大格納数超過および受信バッファ数不足時に論理端末閉塞するかどうかを指定	《error》｜continue
		mcftalced （論理端末定義の終了） 指定数:mcftalcle と同数	－	－	論理端末定義の終了	－

（凡例）

－：該当しません。

注※

TP1/NET/UDP 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。

mcftalcle コマンドの指定条件

mcftalcle コマンドのオプションおよびオペランドの指定条件を次の表に示します。

表 6-3　mcftalcle コマンドのオプションおよびオペランドの指定条件

オプションおよびオペランド		送信するメッセージの種別			受信するメッセージの種別		
		ユニキャスト	ブロードキャスト	マルチキャスト	ユニキャスト	ブロードキャスト	マルチキャスト
-l	論理端末名称	◎	◎	◎	◎	◎	◎
-p	udp	◎	◎	◎	◎	◎	◎
-t	any	◎	◎	◎	◎	◎	◎
-i	auto | manual	○	○	○	○	○	○
-v	アプリケーション名	−	−	−	○	○	○
-g	sndbuf	◎	◎	◎	◎※1	◎※1	◎※1
	rcvbuf	◎※1	◎※1	◎※1	◎	◎	◎
-e	msgbuf	○	○	○	○	○	○
	count	○	○	○	○	○	○
-s	syssndsize	○	○	○	−	−	−
	sysrcvsize	−	−	−	○	○	○
-r	ipaddr	○※2	○※2	○※2	×	×	×
	hostname						
	portno	◎	◎	◎	◎	◎	◎
	reuse	−	−	−	−	○	○
-m	multicast	−	−	◎	−	−	◎
	hostgroupname	−	−	−	−	−	◎※2
	hostgroupaddr	−	−	−	−	−	
	ripaddr	−	−	−	−	−	○※2
	rhostname	−	−	−	−	−	
	multicastttl	−	−	○	−	−	−

オプションおよびオペランド		送信するメッセージの種別			受信するメッセージの種別		
		ユニキャスト	ブロードキャスト	マルチキャスト	ユニキャスト	ブロードキャスト	マルチキャスト
-m	mmsgcnt	○	○	○	−	−	−
	dmsgcnt	○	○	○	−	−	−
	shostname1 shostname2 shostname3 shostname4 shostname5 shostname6 shostname7 shostname8	−	−	−	−	−	○
-k	quekind	○	○	○	−	−	−
	quegrpid	○	○	○	−	−	−
-o	aj	○	○	○	−	−	−
-f	rcvoverflow	−	−	−	○	○	○

（凡例）

◎：必ず指定する項目

○：指定できる項目

×：指定してはいけない項目

−：指定しても無効になる項目

注※1

必ず指定します。ただし，TP1/NET/UDP は指定内容を使用しません。

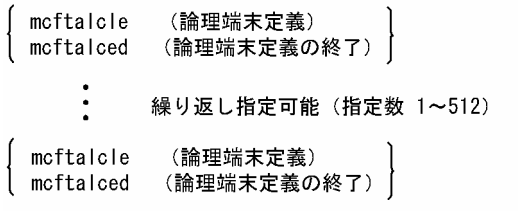
注※2

どちらか一方だけを指定してください。

定義の指定順序

TP1/NET/UDP 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/UDP のプロトコル固有定義コマンドの指定順序



mcftalcle（論理端末定義の開始）

形式

```
mcftalcle  -l  論理端末名称
           -p  udp
           -t  any
          〔-i  auto|manual〕
          〔-v  アプリケーション名〕
           -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"〕
          〔-s  "〔syssndsize=システムのメッセージ送信バッファ長〕
                〔sysrcvsize=システムのメッセージ受信バッファ長〕"〕
           -r  "〔ipaddr=自システムの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
                〔reuse=yes|no〕"
          〔-m  "〔multicast=yes|no〕
                〔hostgroupname=参加するホストグループのホスト名〕
                〔hostgroupaddr=参加するホストグループのIPアドレス〕
                〔ripaddr=自システムのIPアドレス〕
                〔rhostname=自システムのホスト名〕
                〔multicastttl=送信するマルチキャストパケットの生存期間（TTL）〕
                〔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〕
                〔shostname1=マルチキャスト送信元システムのホスト名またはIPアドレス〕
                〔shostname2=マルチキャスト送信元システムのホスト名またはIPアドレス〕
                〔shostname3=マルチキャスト送信元システムのホスト名またはIPアドレス〕
                〔shostname4=マルチキャスト送信元システムのホスト名またはIPアドレス〕
                〔shostname5=マルチキャスト送信元システムのホスト名またはIPアドレス〕
                〔shostname6=マルチキャスト送信元システムのホスト名またはIPアドレス〕
                〔shostname7=マルチキャスト送信元システムのホスト名またはIPアドレス〕
                〔shostname8=マルチキャスト送信元システムのホスト名またはIPアドレス〕"〕
          〔-k  "〔quekind=memory|disk〕
                〔quegrpid=キューグループID〕"〕
          〔-o  "〔aj=yes|no〕"〕
          〔-f  "〔rcvoverflow=error|continue〕"〕
```

機能

論理端末に関する環境を定義します。

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。

●-p udp

プロトコルの種別を指定します。

udp

UDP プロトコル（TP1/NET/UDP）

●-t any

この論理端末の端末タイプを指定します。

any

任意型論理端末

●-i auto | manual　～《manual》

OpenTP1 システム開始時および再開始時に論理端末を自動的に閉塞解除するかどうかを指定します。

auto

OpenTP1 システム開始時および再開始時に論理端末を自動的に閉塞解除します。

manual

MCF 起動後，論理端末を閉塞解除します。論理端末の閉塞解除は，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△△')）の発行で行います。

●-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。MCF アプリケーション属性定義（mcfaalcap -n オプションの name オペランド）で定義した名称を指定してください。MCF 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については，「2.2.5　アプリケーション名の決定」を参照してください。

●-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))

メッセージ送信用の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 groupno オペランドに指定した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))

メッセージ受信用の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 groupno オペランドに指定した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，および出力メッセージ編集 UOC で，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，メッセージ編集用バッファは確保されません。

mcftbuf コマンドの-g オプションの groupno オペランドに指定したバッファグループ番号を指定してください。

count=メッセージ編集用バッファ数　～<符号なし整数>((1～131070))

入力，および出力メッセージ編集 UOC で，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。

メッセージ編集用バッファグループ番号（msgbuf オペランドで指定）に対応する mcftbuf コマンドで指定するバッファ数（-g オプションの count オペランドおよび extend オペランド）の中から，メッセージ編集用に使用するバッファ数を指定してください。2 以上の値を指定しても，入力，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。

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 コマンドで指定するバッファ数（-g オプションの count オペランドおよび extend オペランド）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，このオペランドの指定は無効です。

●-s

（オペランド）

syssndsize=システムのメッセージ送信バッファ長　～<符号なし整数>((1～2147483647))（単位：バイト）

OS が提供するソケットオプション「SO_SNDBUF」（システムのメッセージ送信バッファ長）を指定します。

sysrcvsize=システムのメッセージ受信バッファ長　～<符号なし整数>((1～2147483647))（単位：バイト）

OS が提供するソケットオプション「SO_RCVBUF」（システムのメッセージ受信バッファ長）を指定します。

syssndsize オペランドおよび sysrcvsize オペランドで指定できるバッファ長の範囲は OS に依存します。指定を省略すると，OS のデフォルト値が仮定されます。詳しくは，ご使用の OS のマニュアルで，UDP プロトコルについて説明している個所を参照してください。

なお，syssndsize オペランドおよび sysrcvsize オペランドの指定値を決定する際には，「2.1.4　AP 間通信の注意事項」のメッセージ長に関する項目を参照してください。

●-r

（オペランド）

ipaddr=自システムの IP アドレス　～（nnn.nnn.nnn.nnn）<符号なし整数>((0～255))

自システムをホットスタンバイ構成とする場合，または，自システムがマルチホームドホスト形態（一つのマシン内に複数の IP アドレスが割り当てられている環境）の場合に，メッセージの送信に使用する IP アドレスを指定します。

ipaddr オペランドまたは hostname オペランドのどちらか一方を指定してください。両方を指定した場合は，定義オブジェクトの生成時にエラーとなります。どちらも省略した場合，使用する IP アドレスは OS が設定します。

メッセージ受信に使用する論理端末の場合，ipaddr オペランドを指定してはいけません。

hostname=自システムのホスト名　～< 1～255 文字のホスト名>

自システムをホットスタンバイ構成とする場合，または，自システムがマルチホームドホスト形態の場合に，メッセージの送信に使用するホスト名を指定します。

hostname オペランドまたは ipaddr オペランドのどちらか一方を指定してください。両方を指定した場合は，定義オブジェクトの生成時にエラーとなります。どちらも省略した場合，使用する IP アドレスは OS が設定します。

メッセージ受信に使用する論理端末の場合，hostname オペランドを指定してはいけません。

portno=自システムのポート番号　～<符号なし整数>((1～65535))

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。

portno オペランドで指定するポート番号は，重複を避けるため次のポート番号とは異なる値を指定してください。

- ほかの mcftalcle コマンドの-r オプションで指定するポート番号
- 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）

  OS が任意に割り当てる番号は，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。

reuse=yes | no　～《no》

OS が提供するソケットオプション「SO_REUSEPORT」または「SO_REUSEADDR」を使用して，複数の MCF 通信プロセスで同一のポート番号を共用するかどうかを指定します。

yes

複数の MCF 通信プロセスでポート番号を共用します。

no

複数の MCF 通信プロセスでポート番号を共用しません。

●-m

(オペランド)

multicast=yes | no　～《no》

マルチキャストを使用したメッセージの送受信の有無を指定します。

yes

マルチキャストを使用したメッセージの送受信を行います。

no

マルチキャストを使用したメッセージの送受信を行いません。

hostgroupname=参加するホストグループのホスト名　～< 1～255 文字のホスト名>

マルチキャストメッセージを受信する場合，参加するホストグループのホスト名を指定します。

hostgroupname オペランドまたは hostgroupaddr オペランドのどちらか一方を必ず指定してください。両方を指定した場合は，定義オブジェクトの生成時にエラーとなります。

また，multicast オペランドに no を指定した場合に hostgroupname オペランドを指定すると，定義オブジェクトの生成時にエラーとなります。

hostgroupaddr=参加するホストグループの IP アドレス　～（nnn.nnn.nnn.nnn）<符号なし整数>((0～255))

マルチキャストメッセージを受信する場合，参加するホストグループの IP アドレスを指定します。

hostgroupaddr オペランドまたは hostgroupname オペランドのどちらか一方を必ず指定してください。両方を指定した場合は，定義オブジェクトの生成時にエラーとなります。

また，multicast オペランドに no を指定した場合に hostgroupname オペランドを指定すると，定義オブジェクトの生成時にエラーとなります。

ripaddr=自システムの IP アドレス　～（nnn.nnn.nnn.nnn）<符号なし整数>((0～255))

自システムをホットスタンバイ構成とする場合，または，自システムがマルチホームドホスト形態の場合に，マルチキャストメッセージの受信に使用する IP アドレスを指定します。

ripaddr オペランドまたは rhostname オペランドのどちらか一方を指定してください。両方を指定した場合は，定義オブジェクトの生成時にエラーとなります。どちらも省略した場合，使用する IP アドレスは OS が設定します。

また，multicast オペランドに no を指定した場合に ripaddr オペランドを指定すると，定義オブジェクトの生成時にエラーとなります。

rhostname=自システムのホスト名　～< 1～255 文字のホスト名>

自システムをホットスタンバイ構成とする場合，または，自システムがマルチホームドホスト形態の場合に，マルチキャストメッセージの受信に使用するホスト名を指定します。

rhostname オペランドまたは ripaddr オペランドのどちらか一方を指定してください。両方を指定した場合は，定義オブジェクトの生成時にエラーとなります。どちらも省略した場合，使用する IP アドレスは OS が設定します。

また，multicast オペランドに no を指定した場合に rhostname オペランドを指定すると，定義オブジェクトの生成時にエラーとなります。

multicastttl=送信するマルチキャストパケットの生存期間（TTL）　～<符号なし整数>((0～255))

OS が提供するソケットオプション「IP_MULTICAST_TTL」を使用して，マルチキャストメッセージ送信時のマルチキャストパケットの生存期間（TTL）を指定します。このオペランドを省略した場合，TTL は OS が設定します。TTL のデフォルト値については，ご使用の OS のマニュアルを参照してください。

multicast オペランドに no を指定した場合に multicastttl オペランドを指定すると，定義オブジェクトの生成時にエラーとなります。

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 SEND 文）はエラーリターン（リターン値 DCMCFRTN_71003 またはステータスコード 71003）となります。

0 を指定したか，または省略した場合，メモリキューで待ち合わせ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 SEND 文）はエラーリターン（リターン値 DCMCFRTN_71003 またはステータスコード 71003）となります。

0 を指定したか，または省略した場合，ディスクキューで待ち合わせ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。

shostname1～shostname8=マルチキャスト送信元システムのホスト名または IP アドレス　～< 1～255 文字のホスト名>または（nnn.nnn.nnn.nnn）<符号なし整数>((0～255))

最大 8 個のマルチキャストメッセージ受信を許可する送信元システムのホスト名または IP アドレスを指定します。

マルチキャストメッセージ受信を許可する送信元システムのホスト名または IP アドレスは shostname1 オペランドから順番に指定する必要があります。

multicast オペランドに no を指定した場合にこのオペランドを指定すると，定義オブジェクトの生成時にエラーとなります。

hostgroupname オペランドまたは hostgroupaddr オペランドのどちらも省略した場合は，このオペランドは無効となります。

●-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>

ディスクキューで待ち合わせ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，sendsync 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。

●-f

（オペランド）

rcvoverflow=error | continue　～《error》

相手システムからのメッセージ受信時に，入力メッセージ最大格納数（アプリケーション属性定義（mcfaalcap -n）の msgcnt オペランド指定値）を超過した場合，および，受信バッファ数（バッファグループ定義（mcftbuf -g）の count オペランド指定値）が不足した場合の処理を指定します。

error

入力メッセージ最大格納数を超過した場合，および，受信バッファ数が不足した場合，障害として扱い，論理端末を閉塞します。

continue

入力メッセージ最大格納数を超過した場合，および，受信バッファ数が不足した場合，正常として扱い，処理を続行します。

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 論理端末単位に次の表に示す資源が必要です。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
sndbuf	最大送信長＋ TP1/NET/UDP 制御ヘッダ長（24 バイト）以上	1 以上

バッファ種別	length オペランド	count オペランド
rcvbuf	最大受信メッセージ長以上	相手システムから連続して送られてくるメッセージ数※
msgbuf	sndbuf と rcvbuf のどちらか大きい方の値	sndbuf と同じ

注※

受信バッファの面積は，システムのトラフィック量に応じて，受信バッファ不足が発生しないよう見積もってください。

mcftalced（論理端末定義の終了）

形式

```
mcftalced
```

機能

論理端末定義の終了を示します。

オプション

ありません。

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/UDP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/UDP 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/UDP 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/UDP で定義したシステム構成の内容によっては，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/UDP の場合，MCF 通信構成定義に定義した論理端末の総数になります。実際に通信を行う論理端末の総数ではありません。

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，注意してください。

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
```

ただし，TP1/NET/UDP の MCF 通信プロセスで使用できるファイル記述子の最大数は，適用 OS によって次のように異なります。

OS	1 プロセスで使用できるファイル記述子の最大数
AIX	2048
Linux	1024

TP1/NET/UDP の MCF 通信プロセスで，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，TP1/NET/UDP の MCF 通信プロセスで使用できるファイル記述子数は，1 プロセスで使用できるファイル記述子の最大数に強制的に補正されます。

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/UDP 固有の出力情報や取得タイミングについては，「付録 G　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。システム定義の変更などによって，オペランドの指定値の合計が増加する場合は，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

```
mcfudp -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/UDP 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るためご注意ください。

TP1/NET/UDP 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

```
mcfudpr -i 〔パス名〕解析対象オブジェクトファイル名
```

機能

TP1/NET/UDP のプロトコル固有定義オブジェクトファイル（または，これと共通定義を結合した MCF 通信構成定義オブジェクトファイル）を解析し，定義ソースの形式で標準出力します。

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。

出力例

```
#############################################################
         MCF communication configuration definition
                        UDP definition
#############################################################
   OBJECT FILE NAME : xxxxxxxx
   VV-RR            : vv-rr
   DATE             : yyyy-mm-dd hh:mm:ss
#############################################################

 mcftalcle
   -l            = leid01
   -p            = udp
   -t            = any
   -i            = manual
   -g sndbuf     = 1
   -g rcvbuf     = 2
   -s syssndsize = 32000
   -s sysrcvsize = 61680
   -r portno     = 20001

 .
 .
 .

 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の表に示します。

表 6-4　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。
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) mcftalcle -l leid01	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) mcftalcle -l = leid01
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) mcftalcle -l leid01 ¥ -p udp	継続記号は出力しない。 (例) mcftalcle -l = leid01 -p = udp
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) mcftalcle -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) mcftalcle -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。

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)＋((B＋D)×E)
```

- A：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の送信の場合：1600
  - 同期型メッセージの送信の場合：1700
- B：次のどちらか小さい方の値（単位：バイト）
  - 512
  - 実際に送受信するメッセージの最大メッセージ長（制御ヘッダの 24 バイトを除く）
- C：1 秒当たりのメッセージ送信回数※
- D：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）
  - 1900
- E：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。

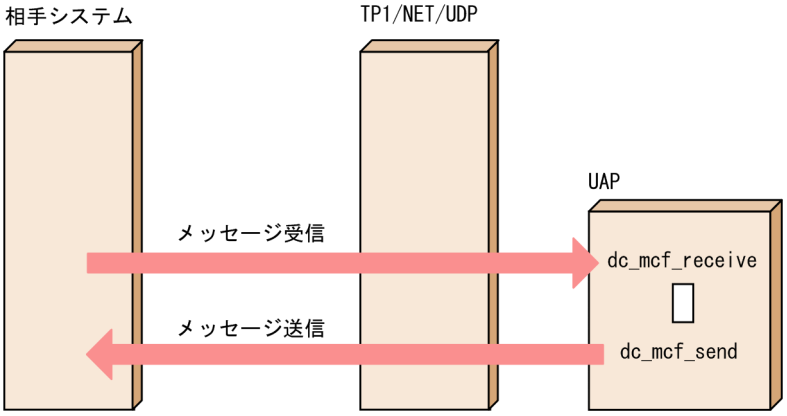
```
経過時間（秒） ＝ F×G×H／1秒当たりのトレース情報量（単位：バイト）
```

- F：トレースバッファの大きさ（mcfttrc -t size）
- G：トレースバッファの数（mcfttrc -t bufcnt）
- H：MCF トレースファイルの数（mcfttrc -t trccnt）

見積もり式の算出例

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，一方送信メッセージの受信と送信をする場合を例に説明します。



この例では，次の値が想定されています。

項目	想定値
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	1600 バイト
メッセージ受信時に取得するトレース情報量	1900 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

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

```
((1600+512)×(120／60)) ＋ ((1900+512)×(120／60)) ＝ 9048
```

1 秒当たりのトレース情報量は，9048 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

```
204800×100×3／9048 ＝ 6790.5
```

トレース情報が失われる経過時間は，6790.5 秒（約 113 分）となります。

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 6-5　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	mcftalcle -r ipaddr	無条件に見直しが必要
	mcftalcle -r hostname	無条件に見直しが必要
	mcftalcle -m hostgroupname	マルチキャスト通信機能を使用する場合，見直しが必要
	mcftalcle -m hostgroupaddr	
	mcftalcle -m ripaddr	
	mcftalcle -m rhostname	

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 grep 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。

論理端末の追加

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 6-6　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

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	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
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，「mcftalcle（論理端末定義の開始）」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，「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 6-7　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

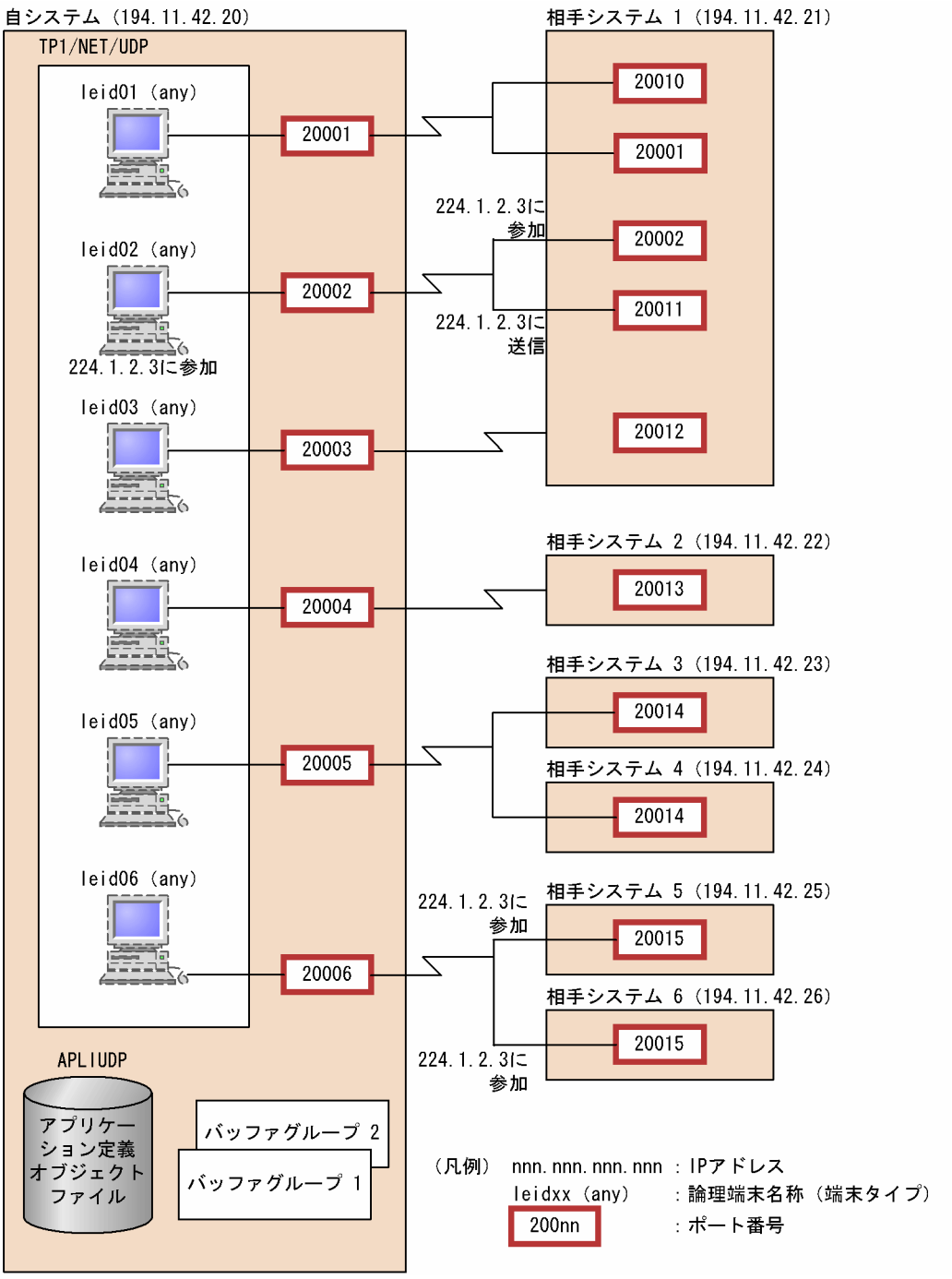
OpenTP1 ファイル	再見積もりが発生する条件
ステータスファイル	次に示すどれかの条件の場合，再見積もりが必要 • メッセージ出力通番を使用する場合 • 拡張予約定義を定義している場合 • 状態を引き継ぐ論理端末が増える場合
メッセージキューファイル	入力キューまたは出力キューにディスクキューを割り当てていて，同時に送受信するメッセージ数が増加する場合

また，TP1/NET/UDP 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

TP1/NET/UDP を使用したシステム構成例を次の図に示し，その構成に沿った定義例を示します。

図 6-2　TP1/NET/UDP のシステム構成例



TP1/NET/UDP では，このシステム構成例に対応する定義例を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/UDPIP/conf/com_c1
- /opt/OpenTP1/examples/mcf/UDPIP/conf/com_d1

その他の OS の場合

- /BeTRAN/examples/mcf/UDPIP/conf/com_c1
- /BeTRAN/examples/mcf/UDPIP/conf/com_d1

TP1/NET/UDP のシステム定義例

```
#####################################################################
#                  MCF 通信構成定義（共通定義）                     #
#  TP1/NET/User Datagram Protocol 対応（提供ファイル名：com_c1）#
#####################################################################
#------------------------- mcftenv ---------------------------#
mcftenv    -s   01                    ¥
           -a   APLIUDP
#------------------------- mcftcomn --------------------------#
mcftcomn
#------------------------- mcfttrc ---------------------------#
mcfttrc    -t  ”disk = yes”           ¥
#------------------------- mcftbuf ---------------------------#
mcftbuf    -g  ”groupno = 1           ¥
                length  = 32768       ¥
                count   = 6”
mcftbuf    -g  ”groupno = 2           ¥
                length  = 32768       ¥
                count   = 18”
##############################  End  ##############################

#####################################################################
#                  MCF 通信構成定義（固有定義）                     #
#  TP1/NET/User Datagram Protocol 対応（提供ファイル名：com_d1）#
#####################################################################
######### LE definition(leid01)
mcftalcle  -l   leid01                  ¥
           -p   udp                     ¥
           -t   any                     ¥
           -g  ”sndbuf     = 1          ¥
                rcvbuf     = 2”         ¥
           -s  ”syssndsize = 32000      ¥
                sysrcvsize = 61680”     ¥
           -r  ”portno = 20001”
mcftalced
######### LE definition(leid02)
mcftalcle  -l   leid02                  ¥
           -p   udp                     ¥
           -t   any                     ¥
           -g  ”sndbuf    = 1           ¥
                rcvbuf    = 2”          ¥
           -s  ”syssndsize = 32000      ¥
                sysrcvsize = 61680”     ¥
           -r  ”portno = 20002”         ¥
           -m  ”multicast = yes         ¥
                hostgroupaddr = 224.1.2.3”
mcftalced
######### LE definition(leid03)
mcftalcle  -l   leid03                  ¥
           -p   udp                     ¥
```

```
           -t   any                    ¥
           -g  "sndbuf    = 1          ¥
                rcvbuf    = 2"         ¥
           -s  "syssndsize = 32000     ¥
                sysrcvsize = 61680"    ¥
           -r  "portno = 20003"
mcftalced
######### LE definition(leid04)
mcftalcle  -l   leid04                 ¥
           -p   udp                    ¥
           -t   any                    ¥
           -g  "sndbuf    = 1          ¥
                rcvbuf    = 2"         ¥
           -s  "syssndsize = 32000     ¥
                sysrcvsize = 61680"    ¥
           -r  "portno = 20004"
mcftalced
######### LE definition(leid05)
mcftalcle  -l   leid05                 ¥
           -p   udp                    ¥
           -t   any                    ¥
           -g  "sndbuf     = 1         ¥
                rcvbuf     = 2"        ¥
           -s  "syssndsize = 32000     ¥
                sysrcvsize = 61680"    ¥
           -r  "portno = 20005"
mcftalced
######### LE definition(leid06)
mcftalcle  -l   leid06                 ¥
           -p   udp                    ¥
           -t   any                    ¥
           -g  "sndbuf    = 1          ¥
                rcvbuf    = 2"         ¥
           -s  "syssndsize = 32000     ¥
                sysrcvsize = 61680"    ¥
           -r  "portno = 20006"        ¥
           -m  "multicast = yes"
mcftalced

############################  End  ############################
```

7 運用コマンド

この章では，TP1/NET/UDP で使用する運用コマンドについて説明します。

TP1/NET/UDP の運用コマンド

ここでは，TP1/NET/UDP 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/UDP 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/UDP 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
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	×	×	○	○

（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。

mcftactle（論理端末の閉塞解除）

形式

```
mcftactle 〔-s MCF通信プロセス識別子〕 -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。

●-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。

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を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を失敗しました。	メッセージログファイル，または標準エラー出力
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
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	mcftactle コマンドはサポートされていません。	標準エラー出力
KFCA10503-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
KFCA18923-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

mcftdctle（論理端末の閉塞）

形式

```
mcftdctle 〔-s MCF通信プロセス識別子〕 -l 論理端末名称
```

機能

論理端末を閉塞します。

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。

●-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。

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を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を失敗しました。	メッセージログファイル，または標準エラー出力
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
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	mcftdctle コマンドはサポートされていません。	標準エラー出力
KFCA10504-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
KFCA18924-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

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，受信仕掛り中のメッセージは破棄されます。以降の受信メッセージは入力キューに格納されません。
- 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。

mcftlsle（論理端末の状態表示）

形式

```
mcftlsle 〔-s MCF通信プロセス識別子〕 -l 論理端末名称〔-q〕〔-t〕
```

機能

論理端末の状態を標準出力に出力します。

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。

●-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。

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を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。

●-t

指定した論理端末がセキュア状態かどうかを表示します。

出力形式

```
mmm llllllll sss [ggg] [tttt]
    SYNC xxxxxxxxxx yyyyyyyyyy zzzzzzzzzz
      IO     :          :          :
    PRIO     :          :          :
    NORM     :          :          :
    [iii ooo]
```

- mmm：MCF 識別子
- llllllll：論理端末名称
- sss：論理端末の状態
  ACT…閉塞解除状態
  DCT…閉塞状態
- ggg：論理端末のセキュア状態（-t オプションの指定時だけ表示）
  NOS…非セキュア状態
  SEC…セキュア状態
- tttt：論理端末のテストモード状態（TP1/Message Control/Tester 使用時だけ表示）
  TEST…テストモード
  空白…非テストモード
- 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）
- xxxxxxxxxx：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（int の上限値まで表示可能）
- zzzzzzzzzz：未送信メッセージの最後の出力通番（int の上限値まで表示可能）
- iii：出力キューの入力の保留状態（-q オプションの指定時だけ表示）

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 オプションの指定時だけ表示）
  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を失敗しました。	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	上記の出力形式を参照してください。	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
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	mcftlsle コマンドはサポートされていません。	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

8 組み込み方法

この章では，TP1/NET/UDP 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/UDP の組み込みの流れ

TP1/NET/UDP 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/UDP 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/UDP 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 章の「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/UDP は，OpenTP1 プロセスサービスによって起動されます。

TP1/NET/UDP を起動するためには，ユーザが MCF メイン関数をコーディングし，コンパイル，およびリンケージを行って TP1/NET/UDP 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，mcfpludp 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を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/UDPIP/cmlib/ansi/com.c
- /opt/OpenTP1/examples/mcf/UDPIP/cmlib/c/com.c

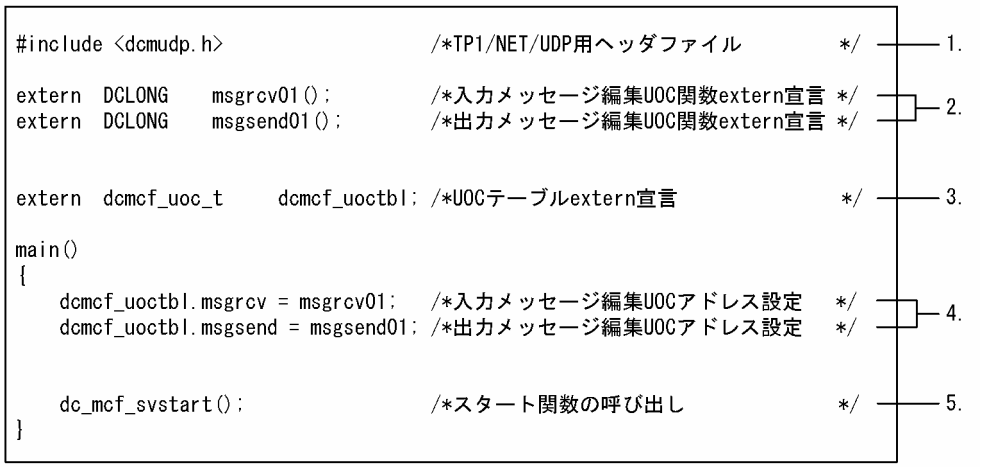
その他の OS の場合

- /BeTRAN/examples/mcf/UDPIP/cmlib/ansi/com.c
- /BeTRAN/examples/mcf/UDPIP/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 <dcmudp.h>                          /*TP1/NET/UDP用ヘッダファイル          */ ―― 1.

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 *);                                          ―― 2.
                                             /*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.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;                                  ―― 4.
                                             /*出力メッセージ編集UOCアドレス設定    */


    dc_mcf_svstart();                        /*スタート関数の呼び出し               */ ―― 5.
    return 0;
}
```

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）



1. TP1/NET/UDP 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。

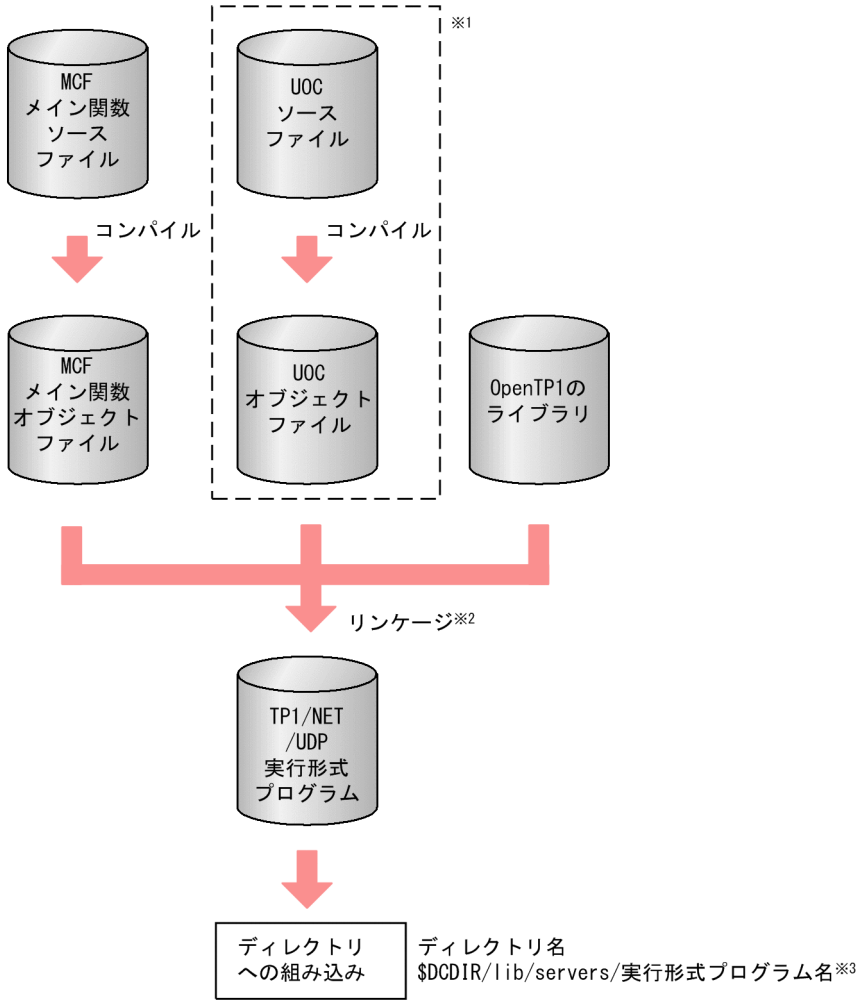
   UOC をまったく使用しない場合，このコーディングは必要ありません。
4. 各 UOC 関数のアドレスを，次に示すシステム提供変数に設定します。使用する UOC だけコーディングしてください。

   ```
   dcmcf_uoctbl.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注意してください。

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

mcfpludp コマンドでリンケージします。

mcfpludp コマンドの詳細については，TP1/NET/UDP の「リリースノート」を参照してください。

注※3

TP1/NET/UDP 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

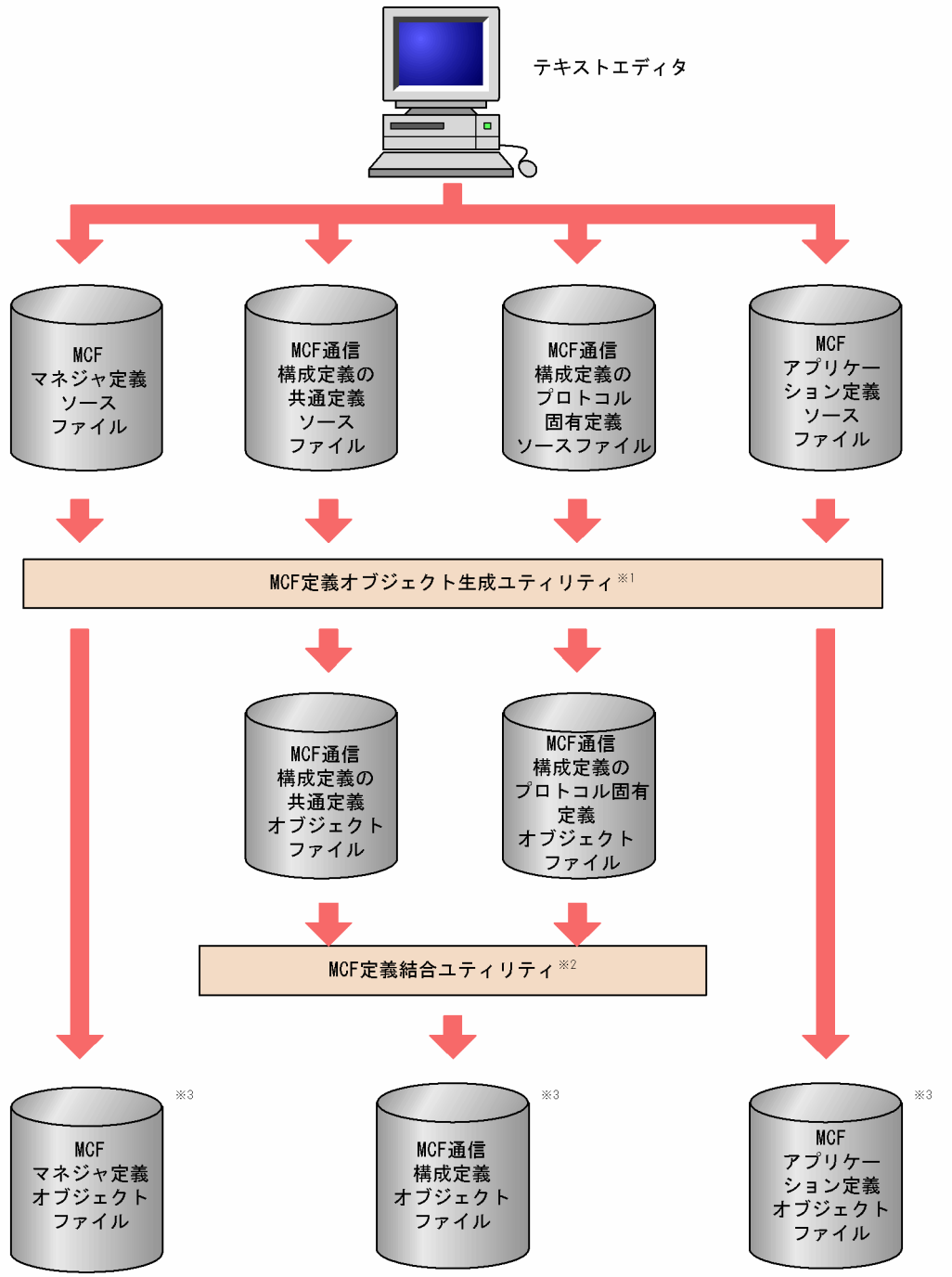
定義オブジェクトファイルを次の手順で生成します。

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おそれがあるためご注意ください。

1. 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/UDP 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/UDP 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 マネジャ定義のソースファイル
- mcfcomn：MCF 通信構成定義のソースファイル

- mcfudp：MCF 通信構成定義のプロトコル（TP1/NET/UDP）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 mcfudp コマンドについては 6 章の「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/UDP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/UDP の運用中に発生するおそれがある障害と，TP1/NET/UDP の対応処理について説明します。

9.1　障害の種類と対応処理

TP1/NET/UDP の障害発生時の処理について，次に示す障害の種類ごとに説明します。

運用中に障害が発生すると，TP1/NET/UDP 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/UDP 運用中の障害と対応処理を次に示します。

理由コードの内容については，「付録Ｉ　理由コード一覧」を参照してください。

9.1.1　論理端末の閉塞解除失敗

表 9–1　論理端末の閉塞解除失敗と対応処理

障害の内容	TP1/NET/UDP の処理	ユーザの処理
論理端末の閉塞解除失敗	1. 論理端末の閉塞解除失敗を通知するメッセージログ（KFCA18903-E）を出力します。 2. CERREVT（論理端末閉塞解除）を起動します。	障害の要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する

9.1.2　バッファ障害

表 9–2　バッファ障害と対応処理

障害の内容	TP1/NET/UDP の処理	ユーザの処理
受信バッファ不足	論理端末定義（mcftalcle -f）の rcvoverflow オペランドが error の場合 1. バッファ不足を通知するメッセージログ（KFCA10618-E），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 受信メッセージを破棄します。 3. 論理端末を閉塞します。 4. CERREVT（受信バッファ取得失敗）を起動します。	定義誤りの場合，誤りを訂正し，定義オブジェクトを作り直して OpenTP1 を再開始します。
	論理端末定義（mcftalcle -f）の rcvoverflow オペランドが continue の場合 1. 受信メッセージを破棄します。 2. 処理を続行します。	ありません。

障害の内容	TP1/NET/UDP の処理	ユーザの処理
送信バッファ不足	1. バッファ不足を通知するメッセージログ（KFCA10618-E），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（送信バッファ取得失敗）を起動します。 4. 同期型メッセージの送信をしている場合，UAP にエラーリターンします。	定義誤りの場合，誤りを訂正し，定義オブジェクトを作り直して OpenTP1 を再開始します。
送信バッファオーバフロー	1. メッセージ出力障害を通知するメッセージログ（KFCA10605-E 障害コード：-10341），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（送信バッファオーバフロー）を起動します。 4. 同期型メッセージの送信をしている場合，UAP にエラーリターンします。	定義誤りの場合，誤りを訂正し，定義オブジェクトを作り直して OpenTP1 を再開始します。

9.1.3 受信スケジュール関係障害（入力キュー，メッセージ受信）

表 9-3 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/UDP の処理	ユーザの処理
・入力キュー障害（入力メッセージ最大格納数超過を除く） ・スケジュール障害 ・UAP 閉塞 ・サービス，サービスグループ閉塞	1. メッセージ入力障害を通知するメッセージログ（KFCA10604-E），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. ERREVT2（メッセージ破棄通知）を起動します。 4. CERREVT（メッセージ入力障害）を起動します。	障害の要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する
入力キュー障害（入力メッセージ最大格納数超過）	論理端末定義（mcftalcle -f）の rcvoverflow オペランドが error の場合 メッセージ入力障害を通知するメッセージログ（KFCA10604-E），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 論理端末を閉塞します。 ERREVT2（メッセージ破棄通知）を起動します。 CERREVT（メッセージ入力障害）を起動します。	障害の要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する
	論理端末定義（mcftalcle -f）の rcvoverflow オペランドが continue の場合 1. 受信メッセージを破棄します。 2. 処理を続行します。	ありません。

障害の内容	TP1/NET/UDP の処理	ユーザの処理
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），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. ERREVT1（不正アプリケーション名検出）を起動します。 4. CERREVT（アプリケーション名取得障害）を起動します。	障害の要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する
UDP プロトコル受信障害	1. 論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（メッセージ受信障害）を起動します。	障害の要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する

9.1.4 送信スケジュール関係障害（メッセージ送信）

表 9-4 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/UDP の処理	ユーザの処理
メッセージ読み込み障害	1. メッセージ出力障害を通知するメッセージログ（KFCA10605-E 障害コード：-10341 以外），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（送信メッセージ取得失敗）を起動します。 4. 同期型メッセージの送信をしている場合，UAP にエラーリターンします。	定義誤りの場合，誤りを訂正し，定義オブジェクトを作り直して OpenTP1 を再開始します。
送信メッセージ長不正 （メッセージ長≦制御ヘッダ長）	1. 論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（送信メッセージサイズ不正）を起動します。 4. 同期型メッセージの送信をしている場合，UAP にエラーリターンします。	UAP を見直し，障害の要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する
UAP 応答障害	1. 論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（UAP 同期応答障害）を起動します。	UAP を見直し，障害の要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する

障害の内容	TP1/NET/UDP の処理	ユーザの処理
UAP 応答障害	1. 論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（UAP 同期応答障害）を起動します。	・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する
UDP プロトコル送信障害	1. 論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（メッセージ送信障害）を起動します。 4. 同期型メッセージの送信をしている場合，UAP にエラーリターンします。	障害の要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する

9.1.5 UOC 障害

表 9-5 UOC の障害と対応処理

障害の内容	TP1/NET/UDP の処理	ユーザの処理
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（KFCA10611-E），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（メッセージ編集 UOC エラーリターン）を起動します。 4. 同期型メッセージの送信をした場合に，出力メッセージ編集 UOC で障害が発生したときは，UAP にエラーリターンします。	UOC を見直してください。
バッファ数不正	1. UOC 不正を通知するメッセージログ（KFCA10620-E 障害コード：-11548），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（使用バッファ数不正）を起動します。 4. 同期型メッセージの送信をした場合に，出力メッセージ編集 UOC で障害が発生したときは，UAP にエラーリターンします。	UOC を見直してください。
有効セグメント長不正	1. UOC 不正を通知するメッセージログ（KFCA10620-E 障害コード：-11580），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（有効セグメント不正）を起動します。 4. 同期型メッセージの送信をした場合に，出力メッセージ編集 UOC で障害が発生したときは，UAP にエラーリターンします。	UOC を見直してください。

障害の内容	TP1/NET/UDP の処理	ユーザの処理
編集バッファリストアドレス不正	1. UOC 不正を通知するメッセージログ（KFCA10620-E 障害コード：-11581），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 2. 論理端末を閉塞します。 3. CERREVT（編集バッファアドレス不正）を起動します。 4. UOC 不正を通知するメッセージログ（KFCA10620-E 障害コード：-11581），および論理端末障害を通知するメッセージログ（KFCA18902-E）を出力します。 5. 同期型メッセージの送信をした場合に，出力メッセージ編集 UOC で障害が発生したときは，UAP にエラーリターンします。	UOC を見直してください。

9.1.6　プロシジャ障害

表 9–6　プロシジャの障害と対応処理

障害の内容	TP1/NET/UDP の処理	ユーザの処理
内部論理矛盾	1. 内部論理矛盾を通知するメッセージログ（KFCA18999-E）を出力します。 2. メモリダンプを出力します。 3. プロセスを異常終了します。	保守情報（$DCDIR/spool ディレクトリ以下）を退避してください。

付録

付録 A　バージョンアップ時の変更点

各バージョンでの変更点を次に示す分類ごとに示します。

- 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/UDP 07-50 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–1　TP1/NET/UDP 07-50 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	RECEIVE － メッセージの受信 • FOR 句の設定値に I-O を追加 • SYNCHRONOUS MODE 句
		SEND － メッセージの送信 • WITH 句
	定義	mcftalcle（論理端末定義の開始） • -f オプションの rcvoverflow オペランド
	コマンド	mcfudp コマンド • -r オプション
		mcfudpr コマンド
変更		なし
削除		なし

TP1/NET/UDP 07-50 での動作の変更点はありません。

TP1/NET/UDP 07-50 でのデフォルト値の変更はありません。

付録 A.2　07-01 での変更点

TP1/NET/UDP 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–2　TP1/NET/UDP 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	mcftalcle（論理端末定義の開始） • -m オプションの shostname1 オペランド • -m オプションの shostname2 オペランド • -m オプションの shostname3 オペランド • -m オプションの shostname4 オペランド • -m オプションの shostname5 オペランド • -m オプションの shostname6 オペランド • -m オプションの shostname7 オペランド • -m オプションの shostname8 オペランド
	コマンド	なし
変更		なし
削除		なし

TP1/NET/UDP 07-01 での動作の変更点はありません。

TP1/NET/UDP 07-01 でのデフォルト値の変更はありません。

付録 A.3　07-00 での変更点

TP1/NET/UDP 07-00 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/UDP 07-00 での動作の変更点を次の表に示します。

表 A–3　TP1/NET/UDP 07-00 での動作の変更点

分類	内容
メッセージ	日本語 UTF-8 メッセージ（LANG ja_JP.UTF-8）に対応※
	英文メッセージを追加

注※

ご使用の OS が Linux の場合だけサポートしています。

TP1/NET/UDP 07-00 でのデフォルト値の変更はありません。

付録 A.4　06-02 での変更点

TP1/NET/UDP 06-02 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–4　TP1/NET/UDP 06-02 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	なし
	定義	mcftalcle（論理端末定義の開始） • -r オプションの reuse オペランド
	コマンド	なし
変更		なし
削除		なし

TP1/NET/UDP 06-02 での動作の変更点はありません。

TP1/NET/UDP 06-02 で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

付録 B.1　ソースの互換性

バージョン 6 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 6 以前からバージョン 7 へ移行する場合，バージョン 6 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-1　バージョン 6 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。 64 ビットアーキテクチャを使用する場合は，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。 64 ビットアーキテクチャを使用する場合は，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。

付録 B.2　メッセージログの英語出力

バージョン 7 では，TP1/NET/UDP のメッセージログ（KFCA189XX-X）を日本語または英語で出力できます。

言語種別は，システム共通定義の putenv 形式の環境変数 LANG の指定値を変更することで設定できます。言語種別の設定方法についての詳細は，マニュアル「OpenTP1 システム定義」のシステム共通定義について説明している個所を参照してください。

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 － 一方送信メッセージの受信（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_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	入力メッセージ編集 UOC	5.1.2　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	5.1.4　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	5.1.6　送信メッセージの通番編集 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

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 のインタフェースの変更一覧を示します。変更個所には，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

(1)　dc_mcf_receive － 一方送信メッセージの受信

(a)　ANSI C，C++の形式

バージョン 6 以前

```
#include <dcmcf.h>
int  dc_mcf_receive(long action,
                    long commform,
                    char *termnam,
                    char *resv01,
                    char *recvdata,
                    long *rdataleng,
                    long inbufleng,
                    long *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	バージョン 7

`#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;`
	`#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

`#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)`

(b) K&R 版 C の形式

バージョン 6 以前	バージョン 7

`#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;`
	`#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;`

バージョン 6 以前

```
long      oseqid;
long      orgseq;
char      *otermnam;
char      *resv02;
char      *resv03;
char      *resv04;
long      opcd;
```

バージョン 7

```
DCLONG    oseqid;
DCLONG    orgseq;
char      *otermnam;
char      *resv02;
char      *resv03;
char      *resv04;
DCLONG    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

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

バージョン 7

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

(b) K&R 版 C の形式

バージョン 6 以前

```
#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;
```

バージョン 7

```
#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_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

```
#include <dcmcf.h>
int  dc_mcf_sendsync(long action,
                     long commform,
```

バージョン 7

```
#include <dcmcf.h>
int  dc_mcf_sendsync(DCLONG action,
                     DCLONG commform,
```

バージョン 6 以前	バージョン 7
char *termnam, char *resv01, char *senddata, long sdataleng, char *resv02, long opcd, long watchtime)	char *termnam, char *resv01, char *senddata, DCLONG sdataleng, char *resv02, DCLONG opcd, DCLONG watchtime)

(b) K&R 版 C の形式

バージョン 6 以前	バージョン 7
#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;

付録 C.2 ユーザオウンコーディング

(1) 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
#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)

K&R 版 C の形式

バージョン 6 以前	バージョン 7
#include <dcmcf.h> #include <dcmcfuoc.h> long uoc_func(parm)	#include <dcmcf.h> #include <dcmcfuoc.h> DCLONG uoc_func(parm)

バージョン 6 以前	バージョン 7
dcmcf_uoc_min_n　*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
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;

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
typedef struct { 　　long buf_num; 　　long used_buf_num; 　　char reserve1[8]; 　　dcmcf_uocbufinf_n 　　　　buf_array[DCMCF_UOC_BUFF_MAX]; } dcmcf_uocbuff_list_n;	typedef struct { 　　DCLONG buf_num; 　　DCLONG 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
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;

(2)　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

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
long  uoc_func(dcmcf_uoc_mout_n *parm)
```

バージョン 7

```
#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

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
long  uoc_func(parm)

dcmcf_uoc_mout_n *parm ;
```

バージョン 7

```
#include <dcmcf.h>
#include <dcmcfuoc.h>
DCLONG  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

```
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;
```

バージョン 7

```
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(1)(b)　パラメタの内容」を参照してください。

(3) 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

```
#include <dcmcf.h>
long send_uoc(long flags,
              char *termname,
              long sendno,
              long sendid,
              long dataleng,
              char *senddata)
```

バージョン 7

```
#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

```
#include <dcmcf.h>
long send_uoc(flags,
              termname,
              sendno,
              sendid,
              dataleng,
              senddata)
long    flags;
char    *termname;
long    sendno;
long    sendid;
long    dataleng;
char    *senddata;
```

バージョン 7

```
#include <dcmcf.h>
DCLONG send_uoc(flags,
                termname,
                sendno,
                sendid,
                dataleng,
                senddata)
DCLONG    flags;
char      *termname;
DCLONG    sendno;
DCLONG    sendid;
DCLONG    dataleng;
char      *senddata;
```

付録 C.3 MCF イベントインタフェース

(1) MCF イベントの共通ヘッダの形式

バージョン 6 以前

```
struct dc_mcf_evtheader {
  char mcfevt_name[9] ;
  char le_name[16] ;
  char cn_name[9] ;
  unsigned char format_kind;
  char reserve01;
  long time ;
};
```

バージョン 7

```
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

```
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] ;
} ;
```

バージョン 7

```
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] ;
} ;
```

(6)　CERREVT の形式

バージョン 6 以前

```
typedef struct {
    struct dc_mcf_evtheader  header ;
    long err_fact ;
    long err_reason1 ;
    long err_reason2 ;
    char reserve1[48] ;
} dcmudp_cerrevt ;
```

バージョン 7

```
typedef struct {
    struct dc_mcf_evtheader  header ;
    DCLONG err_fact ;
    DCLONG err_reason1 ;
    DCLONG err_reason2 ;
    char reserve1[48] ;
} dcmudp_cerrevt ;
```

(7)　COPNEVT，CCLSEVT 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。

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。変更個所は，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 <dcmudp.h>                          /*TP1/NET/UDP用ヘッダファイル          */

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.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();                        /*スタート関数の呼び出し               */
    return 0;
}
```

(b) バージョン７

```
#include <dcmudp.h>                          /*TP1/NET/UDP用ヘッダファイル          */

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.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();                        /*スタート関数の呼び出し               */
    return 0;
}
```

(2)　K&R 版 C の場合

(a)　バージョン 6 以前

```
#include <dcmudp.h>                          /*TP1/NET/UDP用ヘッダファイル          */

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                */

int main()
{
    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();                        /*スタート関数の呼び出し               */
}
```

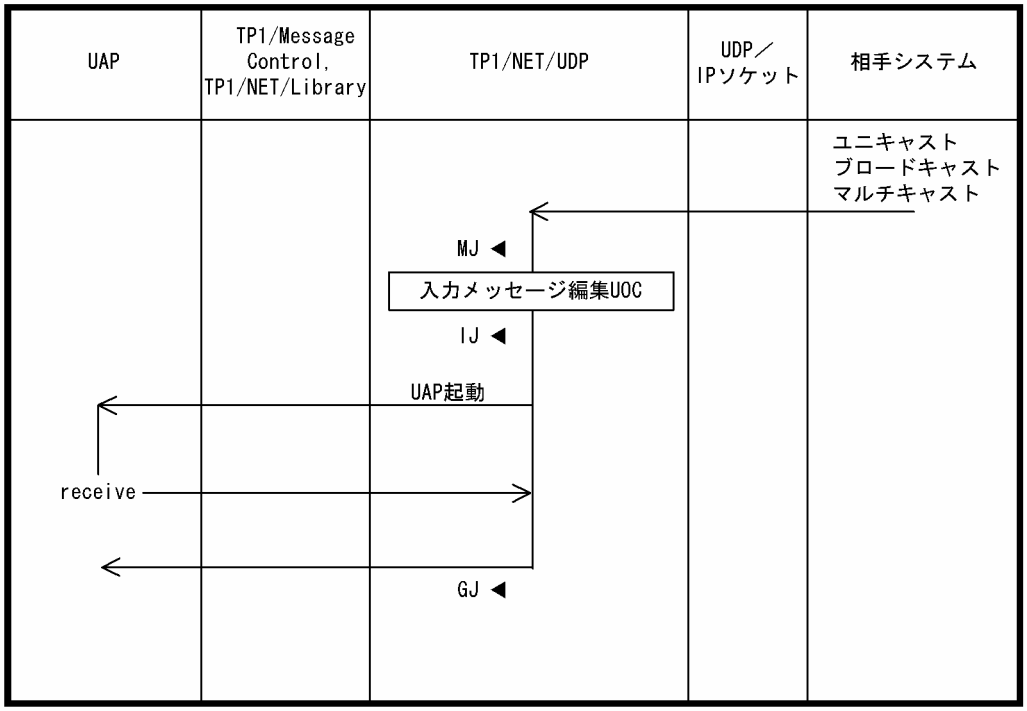
(b)　バージョン 7

```
#include <dcmudp.h>                          /*TP1/NET/UDP用ヘッダファイル          */

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.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();                        /*スタート関数の呼び出し               */
}
```

付録 D　メッセージ送受信の処理の流れ

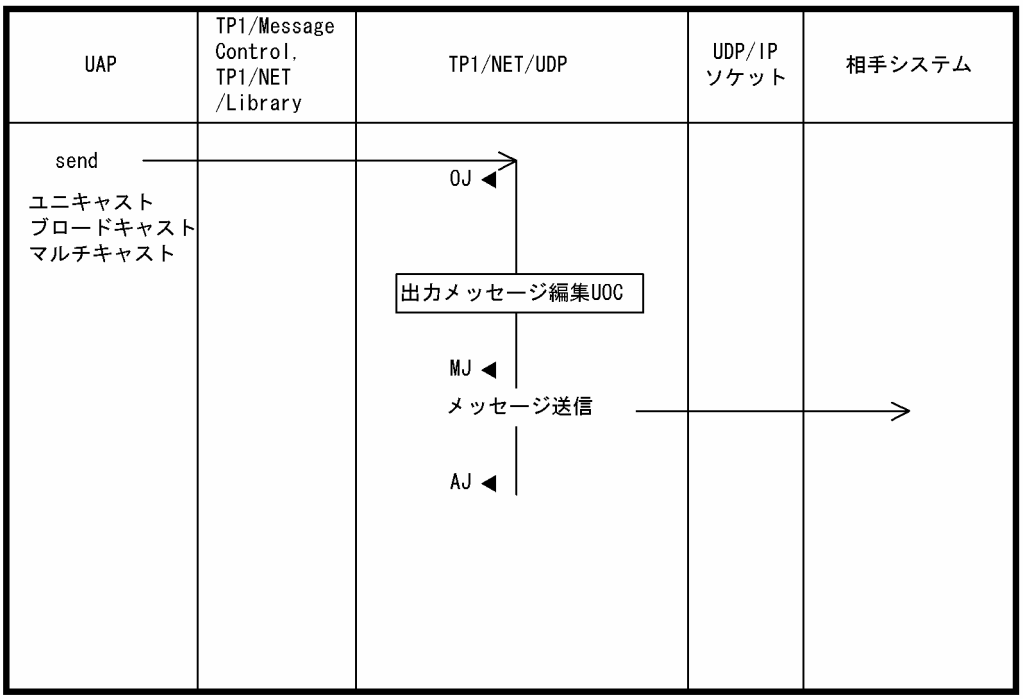
メッセージを送受信するときのデータの流れ，ジャーナル取得のタイミングを図 D-1～図 D-3 に示します。

図 D-1　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-2　一方送信メッセージの送信時の処理の流れ



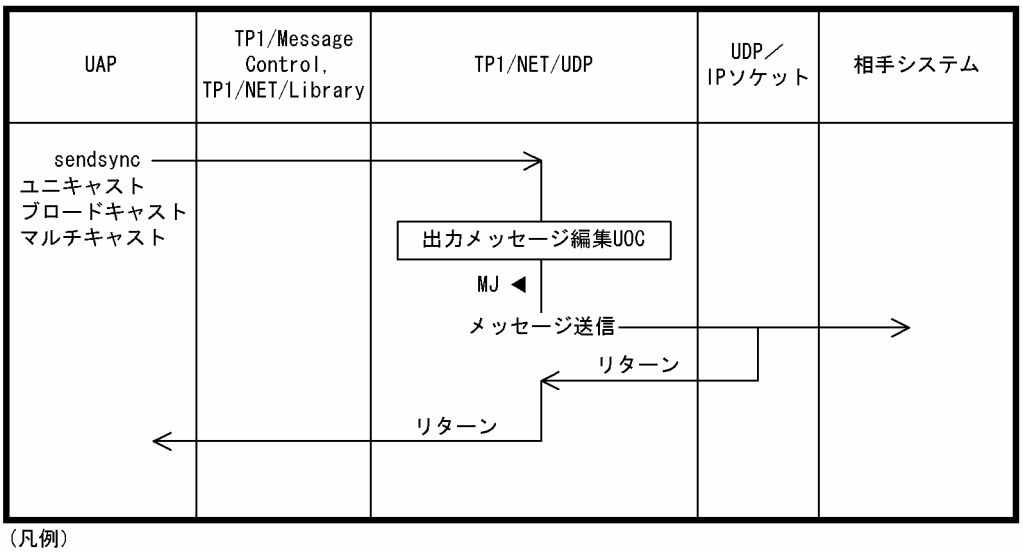
（凡例）

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

AJ◀：メッセージ送信完了ジャーナル取得

図 D-3　同期型メッセージの送信時の処理の流れ



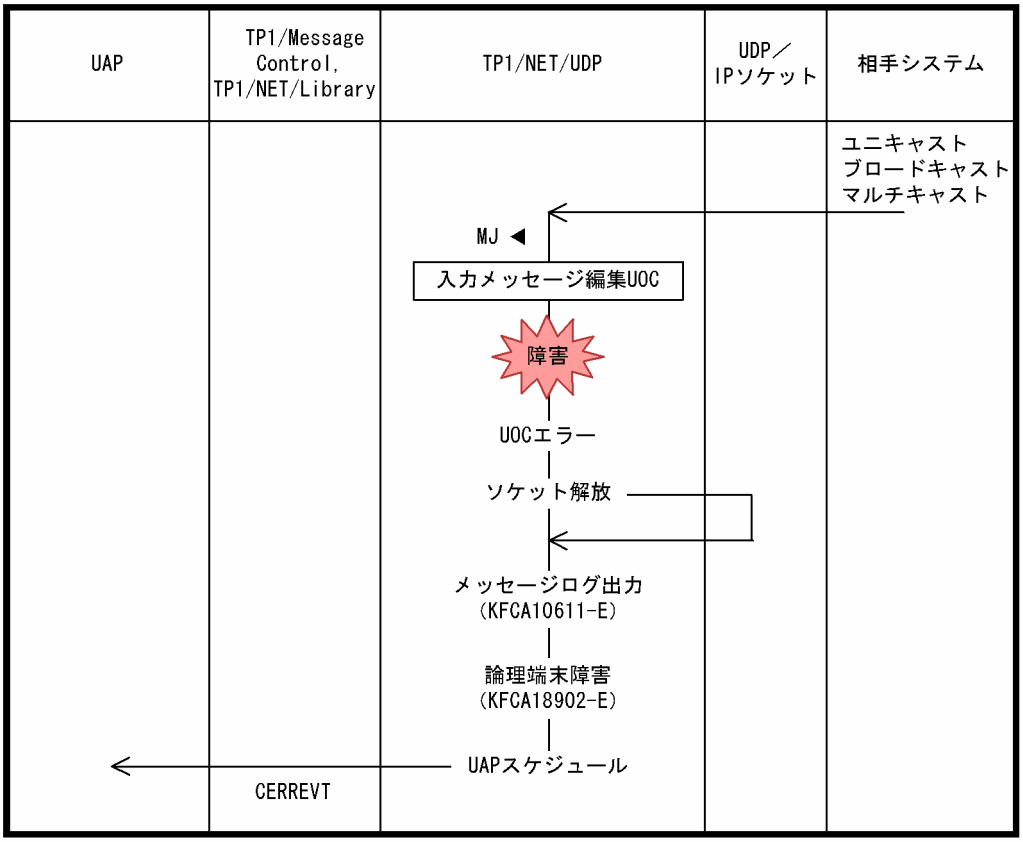
（凡例）

MJ◀：メッセージジャーナル取得

付録 E　障害発生時の処理の流れ

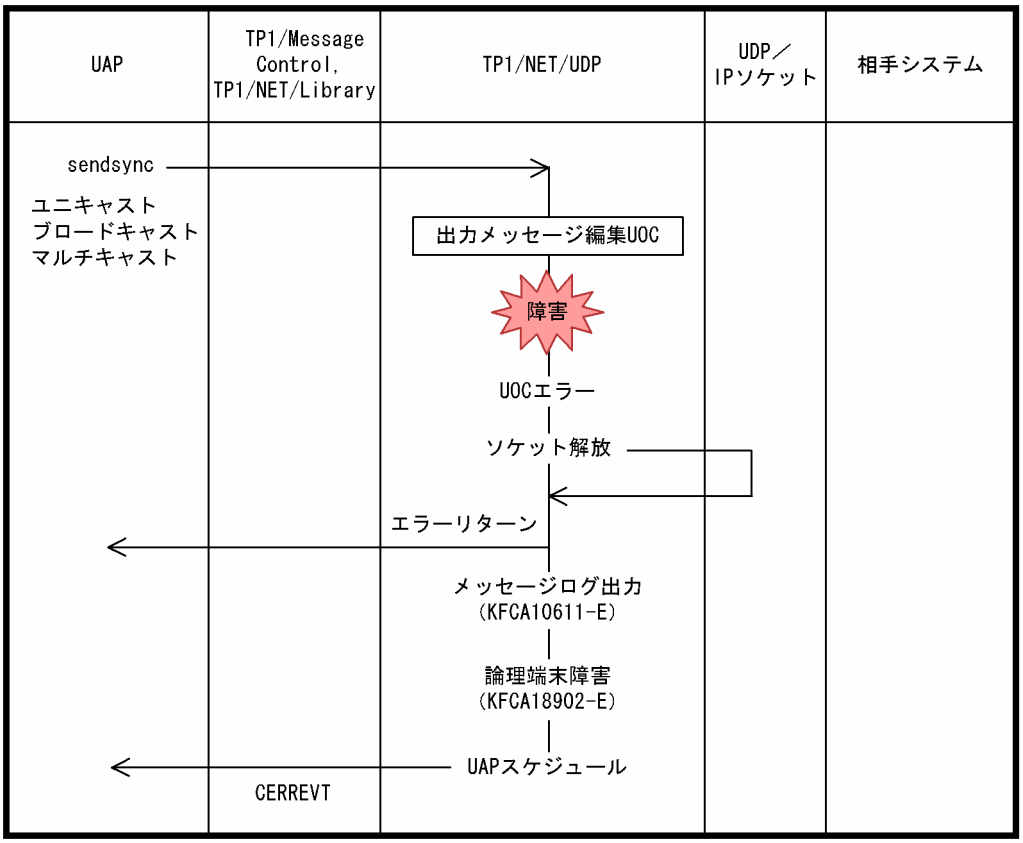
障害発生時の処理の流れを図 E-1 および図 E-2 に示します。

図 E-1　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-2　出力メッセージ編集 UOC エラーの処理の流れ



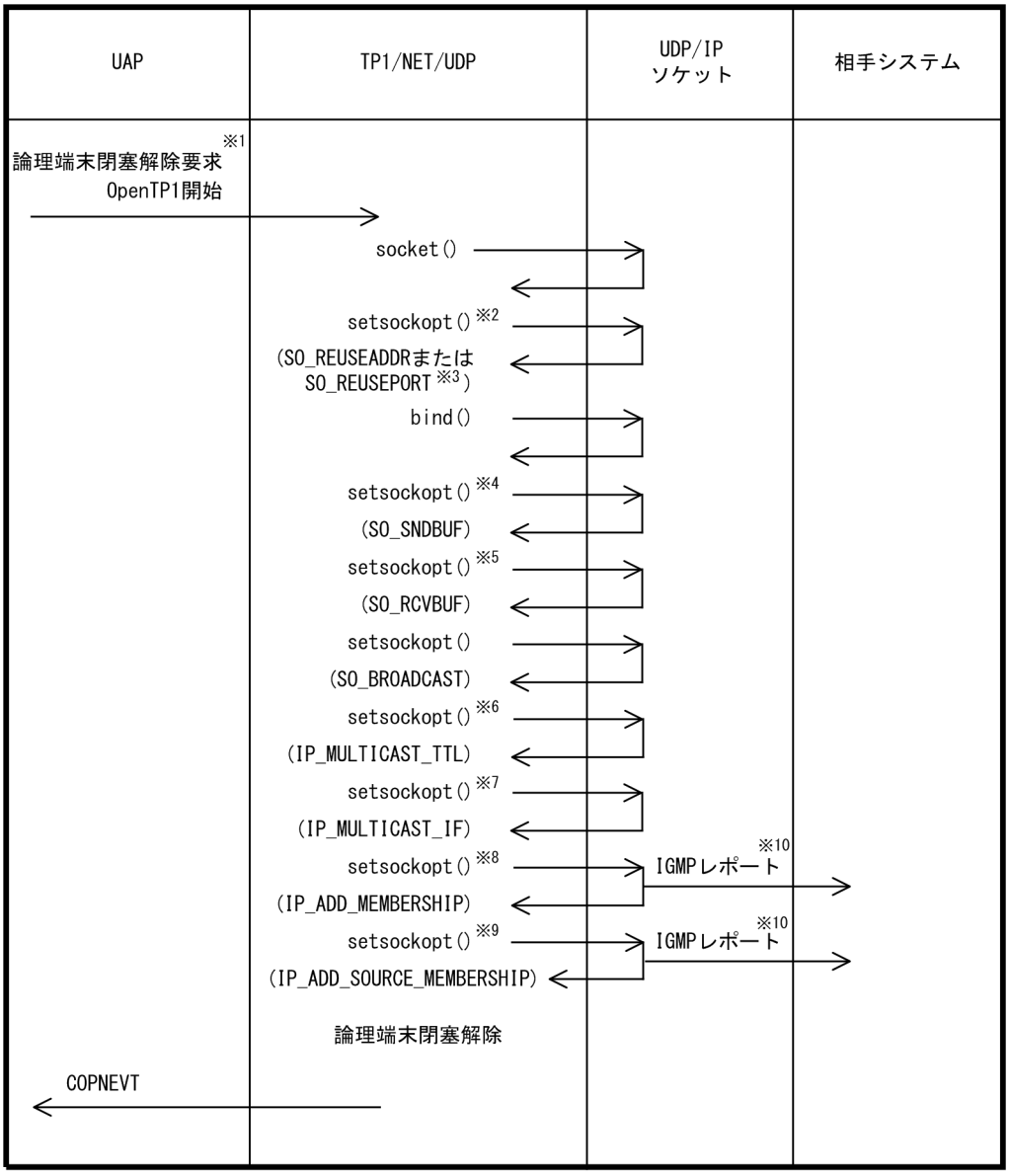
付録 F　ソケット関数の処理の流れ

ここでは，次の場合に TP1/NET/UDP が発行するソケット関数の処理の流れを示します。

- 論理端末の閉塞解除
- メッセージ送信
- メッセージ受信
- 論理端末の閉塞

UDP/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/UDP 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。

図 F-1　論理端末の閉塞解除



注※1

論理端末閉塞解除要求は，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△△'））の発行で行います。

注※2

ポート番号を共用する場合（論理端末定義（mcftalcle -r）の reuse オペランドに yes を指定），発行します。

注※3

適用 OS が Linux の場合，「SO_REUSEADDR」を発行します。適用 OS が AIX の場合，「SO_REUSEPORT」を発行します。

注※4

システムのメッセージ送信バッファ長（論理端末定義（mcftalcle -s）の syssndsize オペランド）を指定した場合，発行します。

注※5

システムのメッセージ受信バッファ長（論理端末定義（mcftalcle -s）の sysrcvsize オペランド）を指定した場合，発行します。

注※6

マルチキャスト通信機能を使用し（論理端末定義（mcftalcle -m）の multicast オペランドに yes を指定），送信するマルチキャストパケットの生存期間（論理端末定義（mcftalcle -m）の multicastttl オペランド）を指定した場合，発行します。

注※7

マルチキャスト通信機能を使用し，自システムのホスト名または IP アドレスを指定した場合，発行します。

注※8

マルチキャスト通信機能を使用し，マルチキャスト送信元システムのホスト名または IP アドレス（論理端末定義（mcftalcle -m）の shostname[1～8]オペランド）を指定していない場合，発行します。

注※9

マルチキャスト通信機能を使用し，マルチキャスト送信元システムのホスト名または IP アドレス（論理端末定義（mcftalcle -m）の shostname[1～8]オペランド）を指定した場合，発行します。

注※10

自システムと相手システム間のネットワーク機器に送信します。

図 F-2　メッセージ送信

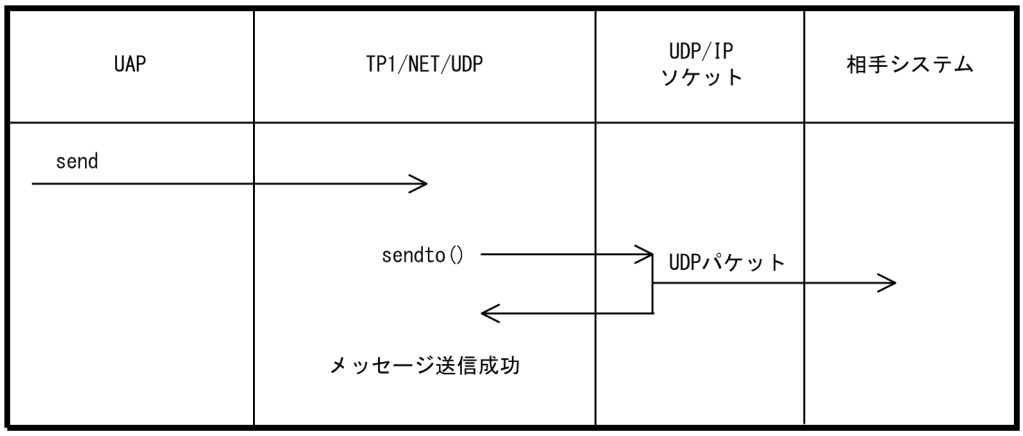


図 F-3　メッセージ受信

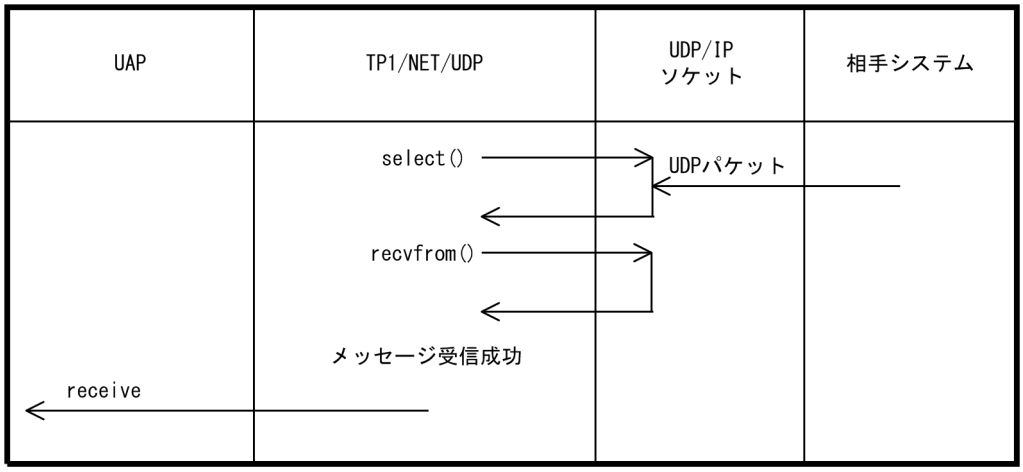
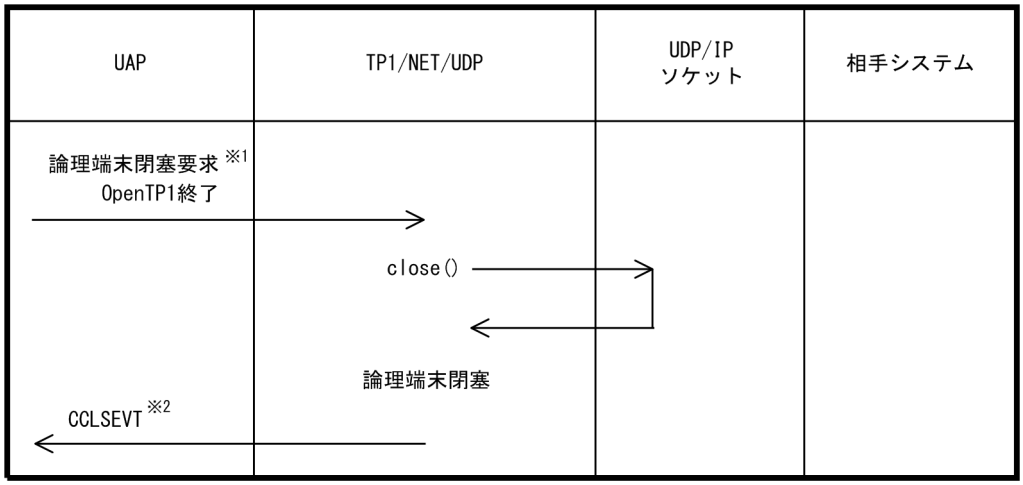


図 F-4　論理端末の閉塞



注※1

論理端末閉塞要求は，運用コマンド（mcftdctle）の入力，または API（dc_mcf_tdctle 関数もしくは CBLDCMCF（'TDCTLE△△'））の発行で行います。

注※2

OpenTP1 終了時の論理端末閉塞では，CCLSEVT は発行されません。

付録 G　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 G.1　MCF 固有情報の出力情報

ここでは，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)　UOC 呼び出し時

TP1/NET/UDP 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 G-1　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 G-2　UOC 名称の出力情報

UOC の種類	UOC 名称出力情報
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"

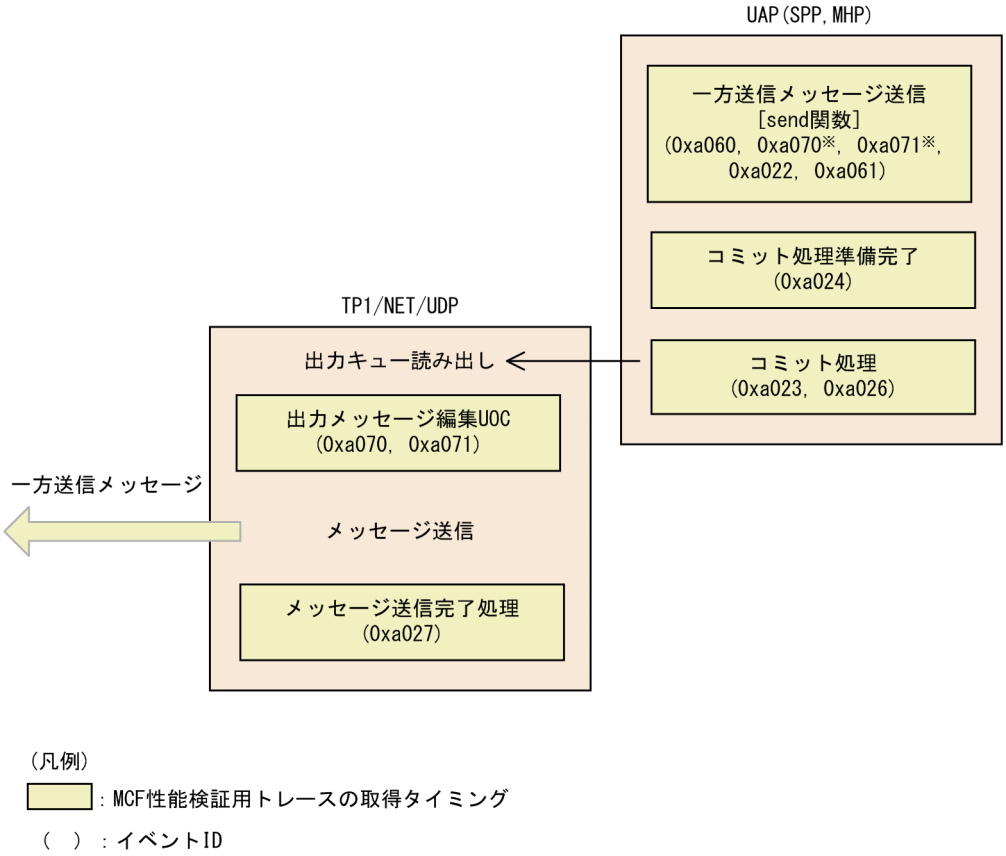
付録 G.2　MCF 性能検証用トレースの取得タイミング

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 G-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



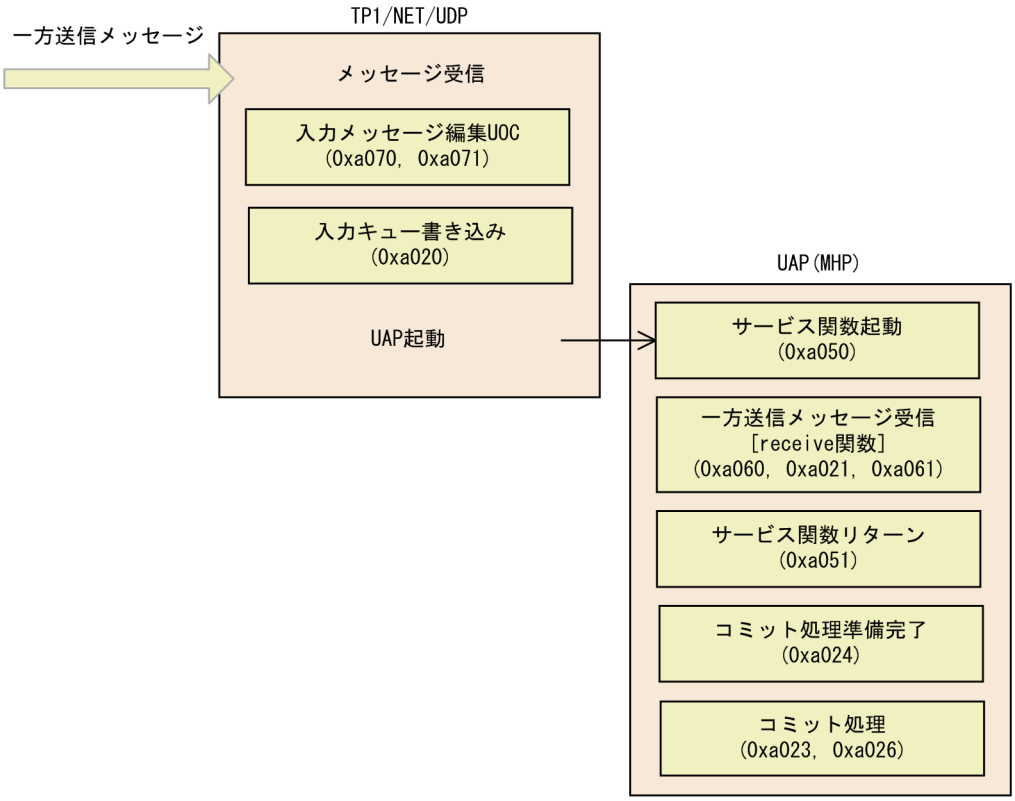
注※

送信メッセージの通番編集 UOC 使用時に該当します。

(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 G-2　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



（凡例）

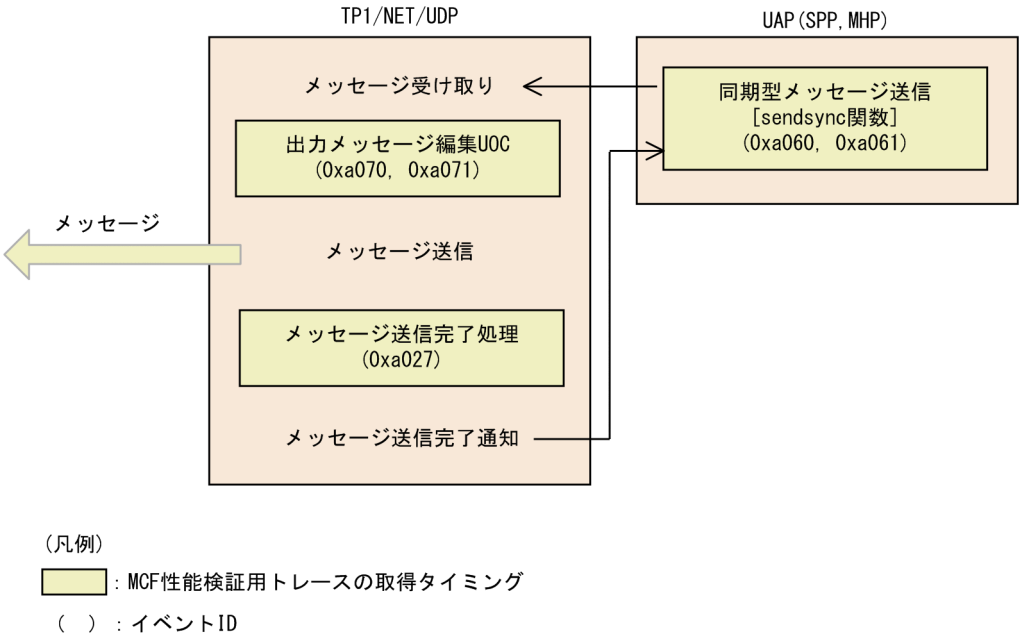
：MCF性能検証用トレースの取得タイミング

（　）：イベントID

(3)　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 G-3　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



付録 G.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。

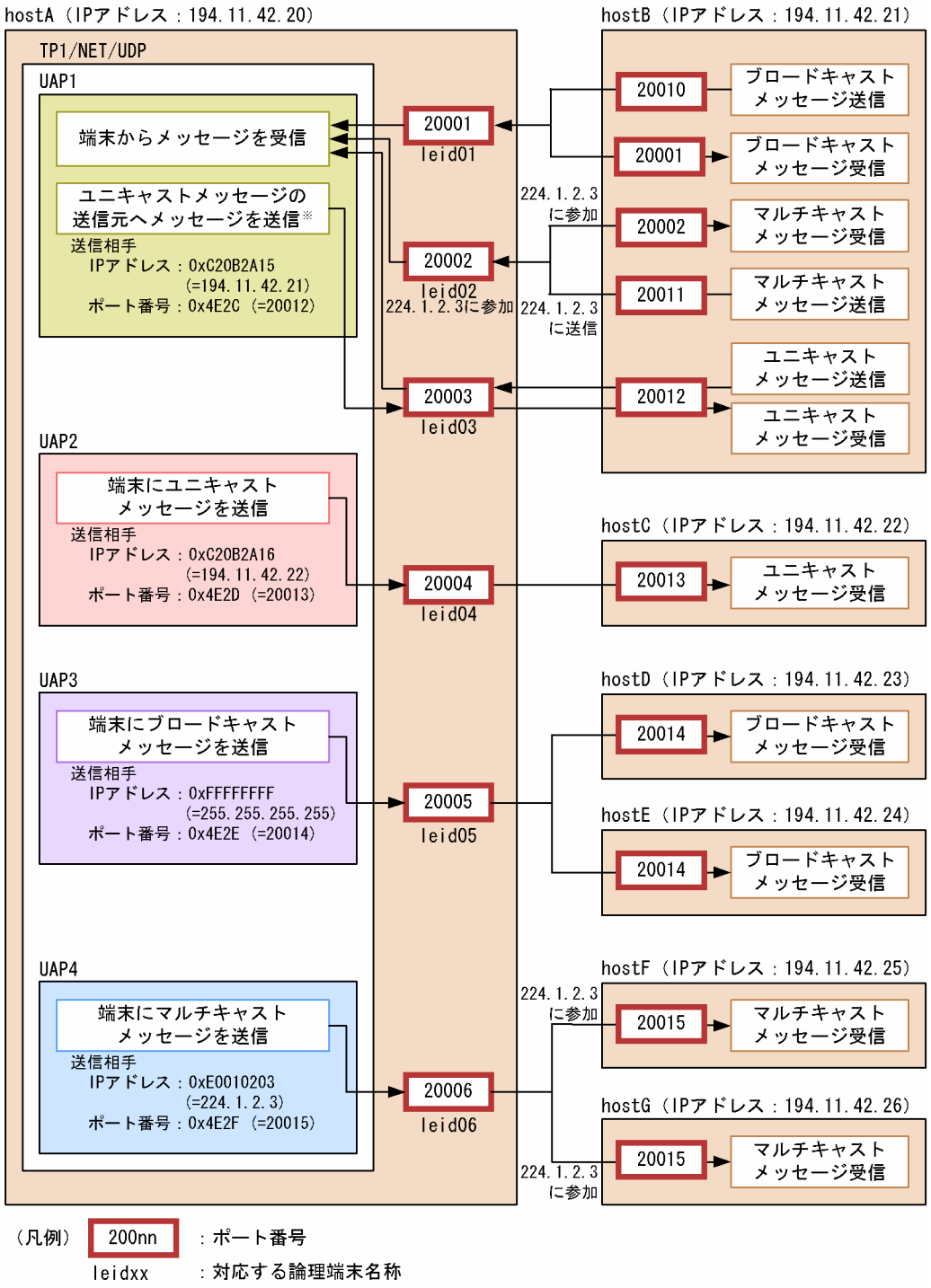
表 G-3　MCF 性能検証用トレースの取得量

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	1.7
一方送信メッセージの受信	2.0
同期型メッセージの送信	0.7

付録 H　ユーザアプリケーションプログラムの作成例

メッセージ送受信の処理の流れを次の図に示し，その流れに沿ったコーディング例を C 言語（K&R 版）で示します。

図 H-1　処理の流れ



注※

端末から受信したメッセージがブロードキャストメッセージまたはマルチキャストメッセージの場合は，メッセージ受信後の処理は不要です。

付録 H.1　コーディング例

使用するメッセージの種類ごとに UAP のコーディング例を示します。

(1)　メッセージ受信処理とユニキャストメッセージ送信（UAP1）

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <dcmcf.h>
#include <dcmudpu.h>

#define SEND_ADDR   0xC20B2A15   /* 0xC20B2A15 == 194.11.42.21 */
#define BRD_PORT    0x4E2A       /* 0x4E2A     == 20010        */
#define MLT_PORT    0x4E2B       /* 0x4E2B     == 20011        */
#define UNI_PORT    0x4E2C       /* 0x4E2C     == 20012        */
#define SRV_MSGSIZE    480
typedef struct srvmsg_s{
    char    mcfarea[8];             /*   8bytes == DCMCFBUF1   */
    dcmudp_dc_header dchead;        /*  24bytes                */
    char    srv_msg[SRV_MSGSIZE];   /* 480bytes == SRV_MSGSIZE */
} srvmsg_def;                       /* 512bytes                */

void    ex_uap1() {
    DCLONG          action;
    static DCLONG   commform   = DCNOFLAGS;
    char            termnam[16];
    static   char   resv01[16] = "¥0";
    static   char   resv02[16] = "¥0";
    srvmsg_def      srvdata_area;
    char            *recvdata;
    char            *senddata;
    DCLONG          sdataleng;
    DCLONG          rdataleng;
    DCLONG          inbufleng;
    static DCLONG   opcd       = DCNOFLAGS;
    static DCLONG   watchtime  = -1;
    DCLONG          time;
    int             rtn;

    recvdata = (char *)&srvdata_area;
    senddata = (char *)&srvdata_area;

    (void)memset((void *)&srvdata_area,
                 0x00,
                 sizeof(srvdata_area));

    action   = DCMCFFRST;
```

```
    inbufleng = sizeof(srvdata_area);

    rtn = dc_mcf_receive(action,
                         commform,
                         termnam,
                         resv01,
                         recvdata,
                         &rdataleng,
                         inbufleng,
                         &time);

    if( DCMCFRTN_00000 == rtn ) {
        if( SEND_ADDR == ntohl(srvdata_area.dchead.r_ipaddr) ) {
            switch( ntohs(srvdata_area.dchead.r_portno) ) {
            case BRD_PORT:
                /*********************************************/
                /*  User Service Part.                       */
                /*********************************************/
                break;

            case MLT_PORT:
                /*********************************************/
                /*  User Service Part.                       */
                /*********************************************/
                break;

            case UNI_PORT:
                /*********************************************/
                /*  User Service Part.                       */
                /*********************************************/
                action   = DCMCFEMI;
                sdataleng = rdataleng;
                rtn = dc_mcf_sendsync(action,
                                      commform,
                                      termnam,
                                      resv01,
                                      senddata,
                                      sdataleng,
                                      resv02,
                                      opcd,
                                      watchtime);

                if( DCMCFRTN_00000 == rtn) {
                    /*****************************************/
                    /*  User Service Part.                   */
                    /*****************************************/
                } else {
                    /*****************************************/
                    /*  User Service Part.                   */
                    /*****************************************/
                }
                break;

            default:
                /*********************************************/
                /*  User Service Part.                       */
                /*********************************************/
```

```
                break;
            }
        } else {
                /*********************************************/
                /*  User Service Part.                       */
                /*********************************************/
        }
    } else {
        /*********************************************/
        /*  User Service Part.                       */
        /*********************************************/
    }
    return;
}
```

(2) ユニキャストメッセージ送信（UAP2）

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <dcmcf.h>
#include <dcmudpu.h>

#define SEND_ADDR   0xC20B2A16   /* 0xC20B2A16 == 194.11.42.22 */
#define SEND_PORT   0x4E2D       /* 0x4E2D     == 20013        */
#define SRV_MSGSIZE    480
typedef struct srvmsg_s{
    char    mcfarea[8];             /*   8bytes == DCMCFBUF1   */
    dcmudp_dc_header dchead;        /*  24bytes                */
    char    srv_msg[SRV_MSGSIZE];   /* 480bytes == SRV_MSGSIZE */
} srvmsg_def;                       /* 512bytes                */

void    ex_uap2() {
    static DCLONG   action     = DCMCFEMI;
    static DCLONG   commform   = DCNOFLAGS;
    static   char   *termnam   = "leid04";
    static   char   resv01[16] = "¥0";
    static   char   resv02[16] = "¥0";
    static DCLONG   opcd       = DCNOFLAGS;
    static DCLONG   watchtime  = -1;
    char            *senddata;
    srvmsg_def      senddata_area;
    DCLONG          sdataleng;
    int             rtn;

    senddata = (char *)&senddata_area;
    (void)memset((void *)&senddata_area,
                 0x00,
                 (size_t)sizeof(senddata_area));

    /*********************************************/
    /*  User Service Part.                       */
    /*********************************************/

    sdataleng = sizeof(senddata_area)
```

```
                - sizeof(senddata_area.mcfarea);

    senddata_area.dchead.r_ipaddr = htonl(SEND_ADDR);

    senddata_area.dchead.r_portno = htons(SEND_PORT);

    rtn = dc_mcf_sendsync(action,
                          commform,
                          termnam,
                          resv01,
                          senddata,
                          sdataleng,
                          resv02,
                          opcd,
                          watchtime);

    if( DCMCFRTN_00000 == rtn) {
        /************************************************/
        /*  User Service Part.                          */
        /************************************************/
    } else {
        /************************************************/
        /*  User Service Part.                          */
        /************************************************/
    }
    return;
}
```

(3) ブロードキャストメッセージ送信（UAP3）

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <dcmcf.h>
#include <dcmudpu.h>

#define SEND_PORT      0x4E2E       /* 0x4E2E     == 20014     */
#define SRV_MSGSIZE    480
typedef struct srvmsg_s{
    char    mcfarea[8];             /*   8bytes == DCMCFBUF1   */
    dcmudp_dc_header dchead;        /*  24bytes                */
    char    srv_msg[SRV_MSGSIZE];   /* 480bytes == SRV_MSGSIZE */
} srvmsg_def;                       /* 512bytes                */

void    ex_uap3(){
    static DCLONG   action     = DCMCFEMI;
    static DCLONG   commform   = DCNOFLAGS;
    static   char   *termnam   = "leid05";
    static   char   resv01[16] = "¥0";
    static   char   resv02[16] = "¥0";
    static DCLONG   opcd       = DCNOFLAGS;
    static DCLONG   watchtime  = -1;
    char            *senddata;
    srvmsg_def      senddata_area;
    DCLONG          sdataleng;
```

```
    int             rtn;

    senddata = (char *)&senddata_area;
    (void)memset((void *)&senddata_area,
                 0x00,
                 sizeof(senddata_area));

    /*************************************************/
    /*  User Service Part.                           */
    /*************************************************/

    sdataleng = sizeof(senddata_area)
                - sizeof(senddata_area.mcfarea);

    senddata_area.dchead.r_ipaddr = htonl(INADDR_BROADCAST);

    senddata_area.dchead.r_portno = htons(SEND_PORT);

    rtn = dc_mcf_sendsync(action,
                          commform,
                          termnam,
                          resv01,
                          senddata,
                          sdataleng,
                          resv02,
                          opcd,
                          watchtime);
    if( DCMCFRTN_00000 == rtn) {
        /*************************************************/
        /*  User Service Part.                           */
        /*************************************************/
    } else {
        /*************************************************/
        /*  User Service Part.                           */
        /*************************************************/
    }
    return;
}
```

(4) マルチキャストメッセージ送信（UAP4）

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <dcmcf.h>
#include <dcmudpu.h>

#define SEND_ADDR     0xE0010203   /* 0xE0010203 == 224.1.2.3  */
#define SEND_PORT     0x4E2F       /* 0x4E2F     == 20015      */
#define SRV_MSGSIZE   480
typedef struct srvmsg_s{
    char    mcfarea[8];             /*   8bytes == DCMCFBUF1   */
    dcmudp_dc_header dchead;        /*  24bytes                */
    char    srv_msg[SRV_MSGSIZE];   /* 480bytes == SRV_MSGSIZE */
} srvmsg_def;                       /* 512bytes                */
```

```
void    ex_uap4() {
    static DCLONG   action     = DCMCFEMI;
    static DCLONG   commform   = DCNOFLAGS;
    static   char   *termnam   = "leid06";
    static   char   resv01[16] = "¥0";
    static   char   resv02[16] = "¥0";
    static DCLONG   opcd       = DCNOFLAGS;
    static DCLONG   watchtime  = -1;
    char            *senddata;
    srvmsg_def      senddata_area;
    DCLONG          sdataleng;
    int             rtn;

    senddata = (char *)&senddata_area;
    (void)memset((void *)&senddata_area,
                 0x00,
                 (size_t)sizeof(senddata_area));

    /*************************************************/
    /*  User Service Part.                           */
    /*************************************************/

    sdataleng = sizeof(senddata_area)
                - sizeof(senddata_area.mcfarea);

    senddata_area.dchead.r_ipaddr = htonl(SEND_ADDR);

    senddata_area.dchead.r_portno = htons(SEND_PORT);

    rtn = dc_mcf_sendsync(action,
                          commform,
                          termnam,
                          resv01,
                          senddata,
                          sdataleng,
                          resv02,
                          opcd,
                          watchtime);

    if( DCMCFRTN_00000 == rtn) {
        /*************************************************/
        /*  User Service Part.                           */
        /*************************************************/
    } else {
        /*************************************************/
        /*  User Service Part.                           */
        /*************************************************/
    }
    return;
}
```

付録 H.2　提供するサンプルコーディング

ここでは，TP1/NET/UDP が提供するサンプルコーディングの格納場所について使用するメッセージの種類ごとに示します。

(1)　メッセージ受信処理とユニキャストメッセージ送信（UAP1）

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/UDPIP/aplib/c/ap1.c

適用 OS が AIX の場合

- /BeTRAN/examples/mcf/UDPIP/aplib/c/ap1.c

(2)　ユニキャストメッセージ送信（UAP2）

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/UDPIP/aplib/c/ap2.c

適用 OS が AIX の場合

- /BeTRAN/examples/mcf/UDPIP/aplib/c/ap2.c

(3)　ブロードキャストメッセージ送信（UAP3）

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/UDPIP/aplib/c/ap3.c

適用 OS が AIX の場合

- /BeTRAN/examples/mcf/UDPIP/aplib/c/ap3.c

(4)　マルチキャストメッセージ送信（UAP4）

適用 OS が Linux の場合

- /opt/OpenTP1/examples/mcf/UDPIP/aplib/c/ap4.c

適用 OS が AIX の場合

- /BeTRAN/examples/mcf/UDPIP/aplib/c/ap4.c

付録 I　理由コード一覧

ERREVT2 発生時の理由コードを表 I-1 に，CERREVT 発生時の理由コードを表 I-2 に示します。

表 I-1　ERREVT2 の理由コード一覧

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	RPC 障害，サーバ未起動などによって MHP または SPP の起動に失敗しました。
DCMCF_QUE_BUF_ERR (0030)	0030	メモリ不足のため，入力キューへの書き込みに失敗しました。
DCMCF_QUE_FIL_OVER (0031)	0031	キューファイルが満杯のため，入力キューへの書き込みに失敗しました。
DCMCF_QUE_LIMIT_OVER (0032)	0032	入力メッセージ最大格納数の定義指定値を超えたため，入力キューに書き込みませんでした。
DCMCF_QUE_IO_ERR (0033)	0033	入力キューへの書き込み時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中です。
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中です。
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 のセグメント受信関数にセグメントを渡す前に，MHP の異常が発生しました。

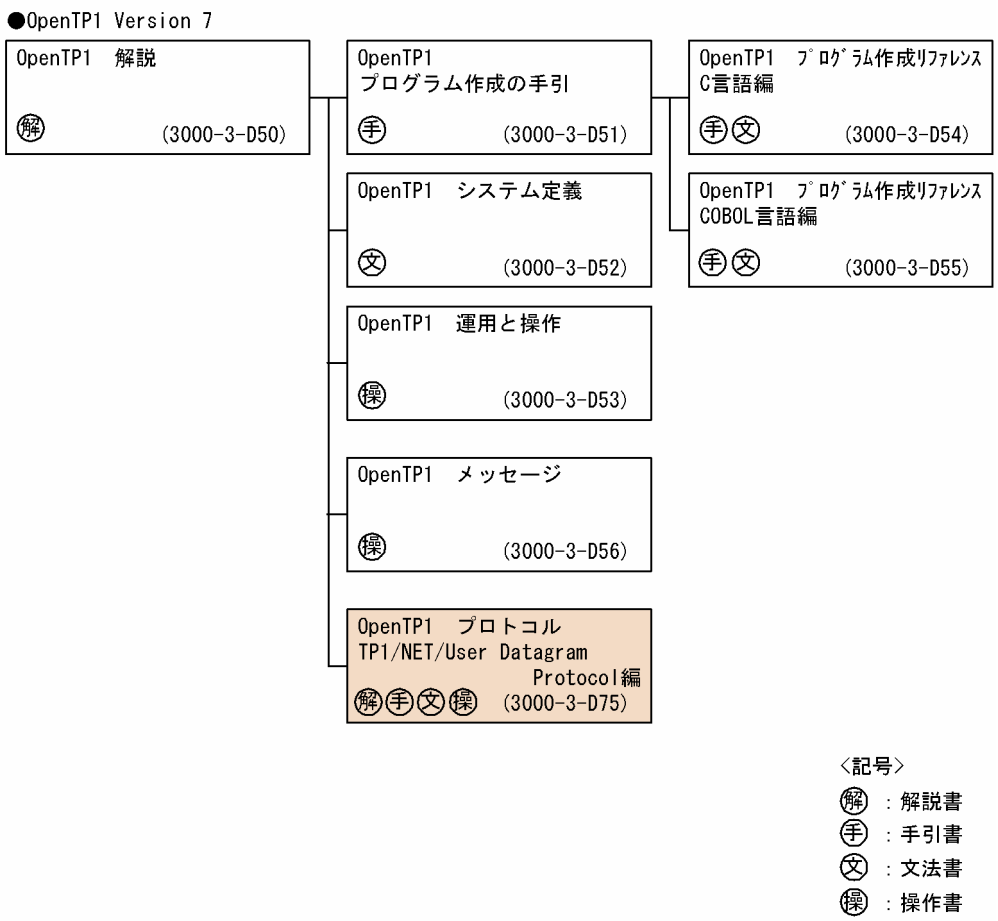
表 I-2　CERREVT の理由コード一覧

理由コード 1 (16 進数字)	理由コード 2 (16 進数字)	内容
DCMUDP_RSN1_MCF (00000001)（MCF 障害）	DCMUDP_RSN2_ACTLE (00000000)	論理端末の閉塞解除
	DCMUDP_RSN2_SNBUF	送信バッファ取得失敗

理由コード 1 (16 進数字)	理由コード 2 (16 進数字)	内容
DCMUDP_RSN1_MCF (00000001)（MCF 障害）	(00000001)	送信バッファ取得失敗
	DCMUDP_RSN2_REBUF (00000002)	受信バッファ取得失敗
	DCMUDP_RSN2_SBFOF (00000003)	送信バッファオーバフロー
	DCMUDP_RSN2_MSGET (00000004)	送信メッセージ取得失敗
	DCMUDP_RSN2_SMSGLN (00000005)	送信メッセージサイズ不正
	DCMUDP_RSN2_RSPUAP (00000006)	UAP 同期応答障害
	DCMUDP_RSN2_SNDER (00000007)	メッセージ送信障害
	DCMUDP_RSN2_ITPUT (00000008)	メッセージ入力障害
	DCMUDP_RSN2_APNAM (00000009)	アプリケーション名取得障害
	DCMUDP_RSN2_RCVER (0000000A)	メッセージ受信障害
DCMUDP_RSN1_UOC (00000003)（UOC 障害）	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
	DCMUDP_RSN2_BCNT (00000001)	使用バッファ数不正
	DCMUDP_RSN2_SEG (00000002)	有効セグメント不正
	DCMUDP_RSN2_BADR (00000003)	編集バッファアドレス不正

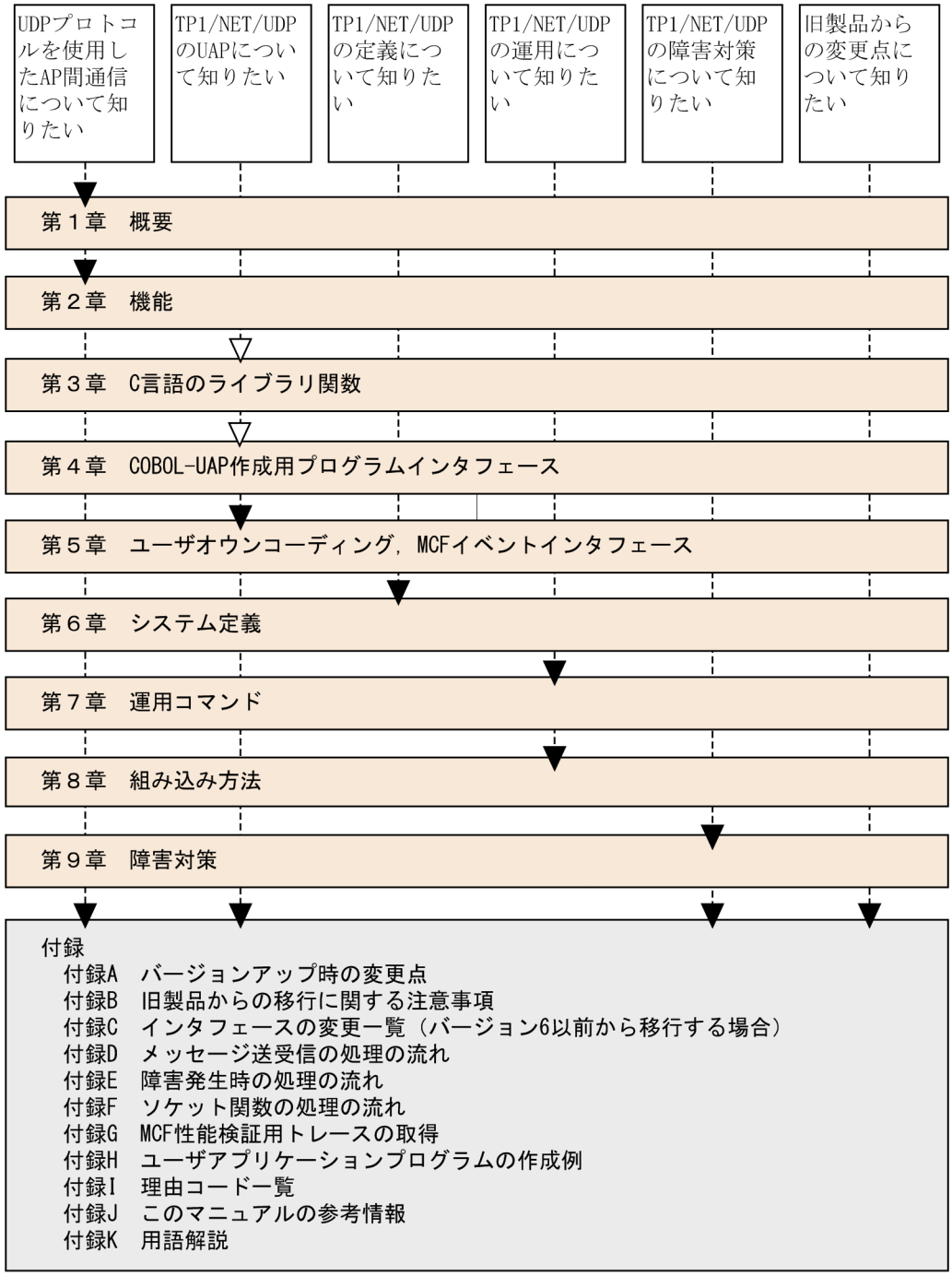
付録 J　このマニュアルの参考情報

付録 J.1　関連マニュアル



付録 J.2　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。



付録 J.3　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 5L V5.1	AIX	
AIX 5L V5.2		
AIX 5L V5.3		
AIX V6.1		
AIX V7.1		
Itanium(R) Processor Family	IPF	
uCosminexus TP1/Message Control	TP1/Message Control	
uCosminexus TP1/Message Control Tester	TP1/Message Control/Tester	
uCosminexus TP1/NET/Library	TP1/NET/Library	
uCosminexus TP1/NET/User Datagram Protocol	TP1/NET/UDP	
UNIX(R)	UNIX	
Linux(R)		Linux
Red Hat Enterprise Linux 5 (AMD/Intel 64)	Linux（x86，AMD/Intel 64）（32 ビット用）， Linux（AMD/Intel 64）（64 ビット用）	
Red Hat Enterprise Linux 5 (x86)		
Red Hat Enterprise Linux 5 Advanced Platform (AMD/Intel 64)		
Red Hat Enterprise Linux 5 Advanced Platform (x86)		
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 5 (Intel Itanium)	Linux（IPF）	
Red Hat Enterprise Linux 5 Advanced Platform (Intel Itanium)		

AIX および Linux を総称して UNIX と表記しています。

(2) 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 • Linux（x86，AMD/Intel 64）（32 ビット用）

マニュアルの表記	OS
64 ビットアーキテクチャの場合	• Linux（IPF）

(3) 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）

付録 J.4 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
AP	Application Program
FDDI	Fiber Distributed Data Interface
IP	Internet Protocol
LAN	Local Area Network
MCF	Message Control Facility
MHP	Message Handling Program
MTU	Maximum Transmission Unit
OS	Operating System
SPP	Service Providing Program
TTL	Time to Live
UAP	User Application Program
UOC	User Own Cording

付録 J.5　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。

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，メッセージを UDP/IP ソケットの送信バッファにすべて格納した直後です。

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。

UDP プロトコル

RFC 768 で規約されたプロトコル規約です。

（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）

ブロードキャストメッセージ

同一の LAN に接続されているすべての端末に送信されるメッセージです。

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。

ホストグループ

一つのマルチキャストアドレスを共有するネットワーク上の端末の集まりです。

（マ行）

マルチキャスト

このマニュアルでは，RFC 1112 で規約された IP マルチキャストを指します。

マルチキャストメッセージ

特定の一つのホストグループを指定して送信するメッセージです。

ホストグループに参加している複数の端末が受信します。

（ユ行）

ユニキャストメッセージ

特定の端末を指定して送信するメッセージです。

（ラ行）

論理端末

TP1/NET/UDP と UAP との通信接点です。TP1/NET/UDP では，論理端末を通してメッセージの送受信をします。

索引

N

O

P

R

S

T

U

あ