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/OSAS-NIF 編

解説・手引・文法・操作書

3000-3-D79-10

前書き

■ 対象製品

・適用 OS：AIX V6.1，AIX V7.1，AIX V7.2

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

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

P-F1M64-32418 uCosminexus TP1/NET/OSAS-NIF 07-50

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

■ 輸出時の注意

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

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

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

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

UNIX は，The Open Group の米国ならびに他の国における登録商標です。

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

2017 年 7 月　3000-3-D79-10

■ 著作権

変更内容

変更内容（3000-3-D79-10）uCosminexus TP1/Message Control　07-51，uCosminexus TP1/NET/Library　07-51，uCosminexus TP1/NET/OSAS-NIF　07-50

追加・変更内容	変更個所
システムサービス共通情報定義の，次に示すオペランドの指定値の上限を拡張した。 • 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ある場合，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
次に示すバージョンの変更点を記載した。 • TP1/NET/OSAS-NIF 07-50	付録 A.1

uCosminexus TP1/Message Control　07-05，uCosminexus TP1/NET/Library　07-05，uCosminexus TP1/NET/OSAS-NIF　07-01

追加・変更内容	変更個所
コネクションの確立，解放，および状態取得について，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。	2.1.4，2.1.5， 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_tactcn， dc_mcf_tdctcn， dc_mcf_tlscn， 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('TACTCN ')， CBLDCMCF('TDCTCN ')， CBLDCMCF('TLSCN ')
コネクション確立処理中のオンライン終了に関する説明を追加した。	2.1.4(4)， 6.　システム定義 TP1/NET/OSAS-NIF 固有のシステム定義の種類， mcftpcmn（プロトコル共通定義）
論理端末の閉塞，閉塞解除，および状態取得について，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。	2.1.6， 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，

追加・変更内容	変更個所
論理端末の閉塞，閉塞解除，および状態取得について，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。	dc_mcf_tactle, dc_mcf_tdctle, dc_mcf_tlsle, 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 ')
着呼型コネクションの自動受付開始に関する説明を追加した。	2.1.7, 6.　システム定義 TP1/NET/OSAS-NIF 固有のシステム定義の種類, mcftpcmn（プロトコル共通定義), 9.1
過着呼による障害検出に関する説明を追加した。	2.1.8, 6.　システム定義 TP1/NET/OSAS-NIF 固有のシステム定義の種類, mcftpcmn（プロトコル共通定義), 9.1
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_73001 およびステータスコード 73001 の説明を変更した。	3.　C 言語のライブラリ関数 dc_mcf_sendrecv, 4.　COBOL-UAP 作成用プログラムインタフェース CBLDCMCF('SENDRECV'), SEND － メッセージの送信
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 ')

追加・変更内容	変更個所
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 F
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
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/OSAS-NIF 07-01	付録 A.2
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

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/OSAS-NIF 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 NIF/OSI プロトコルを使用するシステム管理者およびシステム設計者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，次のマニュアルの内容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/OSAS-NIF を使用したシステム間通信の概要について説明しています。

第 2 章　機能

TP1/NET/OSAS-NIF のコネクション，論理端末，およびメッセージ送受信に関する機能について説明しています。

第 3 章　C 言語のライブラリ関数

TP1/NET/OSAS-NIF 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/OSAS-NIF 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/OSAS-NIF に関連するユーザオウンコーディング（UOC）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

NIF/OSI 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/OSAS-NIF 固有のシステム定義および定義例について説明しています。

第７章　運用コマンド

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

第９章　障害対策

TP1/NET/OSAS-NIF の障害時の処理とユーザの対策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へ移行する場合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，および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

メッセージの送受信中に，障害が発生した場合の処理の流れについて説明しています。

付録 F　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 G　ユーザアプリケーションプログラムの作成例

TP1/NET/OSAS-NIF 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 H　UOC のコーディング例

TP1/NET/OSAS-NIF の入力メッセージ編集 UOC のコーディング例を示しています。

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

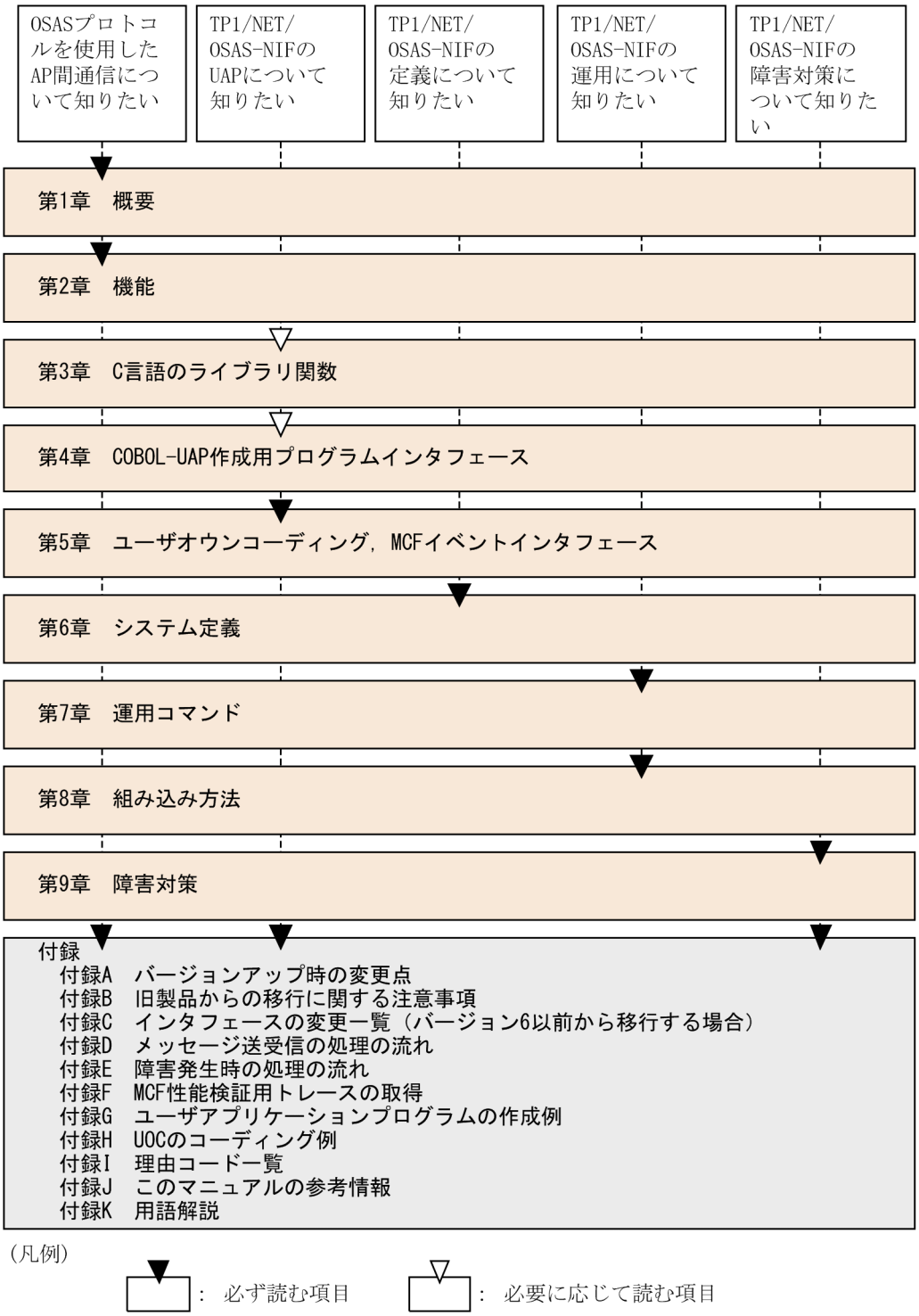
関連マニュアル，このマニュアルで使用している略語の意味などを説明しています。

付録 K　用語解説

TP1/NET/OSAS-NIF で使用する用語について説明しています。

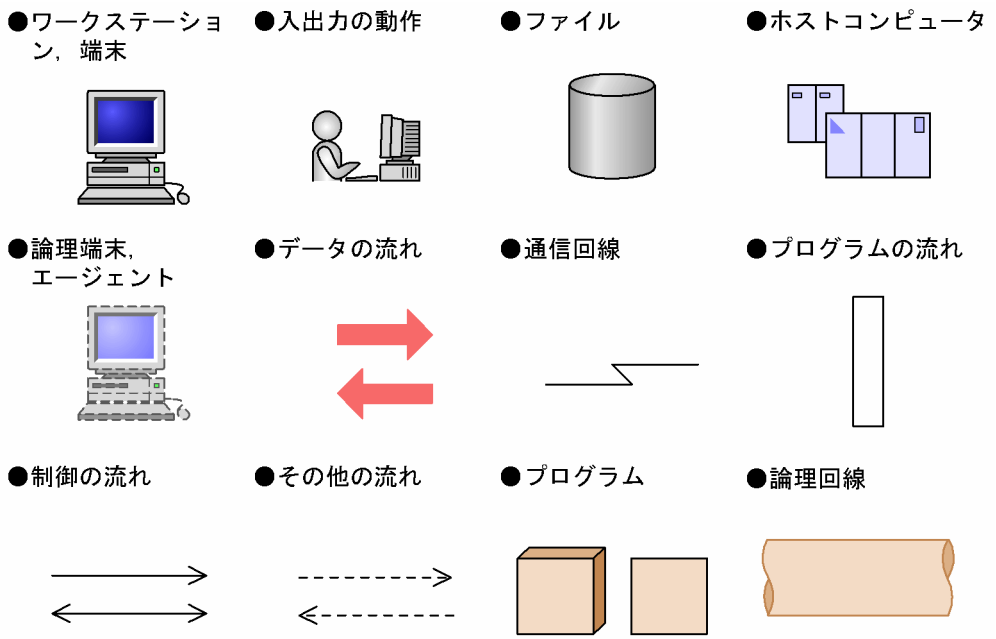
■ 読書手順

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



■ 図中で使用する記号

このマニュアルの図中で使用する記号を，次のように定義します。



■ 文法の記号

このマニュアルで使用する各種の記号を説明します。

(1) 文法記述記号

文法の記述形式について説明する記号です。

文法記述記号	意味
〔 〕	この記号で囲まれている項目は省略できることを示します。 (例) 〔-s MCF 通信プロセス識別子〕 -s オプションとそのオペランドを指定するか，何も指定しないことを示します。
| (ストローク)	この記号で仕切られた項目は選択できることを示します。 (例) -t reply | request -t オプションに reply または request を指定できることを示します。 ただし，C 言語のインタフェースの説明でこの記号を使用した場合は，C 言語の文法規則に従います。
{ }	この記号で囲まれている複数の項目のうちから一つを選択することを示します。 (例) {DCMCFESI | DCMCFEMI} DCMCFESI と DCMCFEMI のうち，どちらかを指定することを示します。
{{ }}	この記号で囲まれた複数の項目が一つの繰り返し項目の単位であり，繰り返し指定できることを示します。 (例) {{mcftalccn mcftalcle mcftalced}}

文法記述記号	意味
{{　}}	mcftalccn，mcftalcle および mcftalced を一組として，繰り返し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）コネクション ID1△コネクション ID2 　コネクション ID1 とコネクション ID2 の間に，空白を 1 個入力することを示します。

(2)　属性表示記号

ユーザ指定値の範囲などを説明する記号です。

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

(3)　構文要素記号

ユーザ指定値の内容を説明する記号です。

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

構文要素記号	意味
〈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）

■ 謝辞

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

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

COBOL は産業界の言語であり，特定の団体や組織の所有物ではない。

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

　次に示す著作権表示付資料の著作者及び著作権者

FLOW-MATIC（Sperry Rand Corporation の商標），

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

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

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

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

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

目次

1 概要

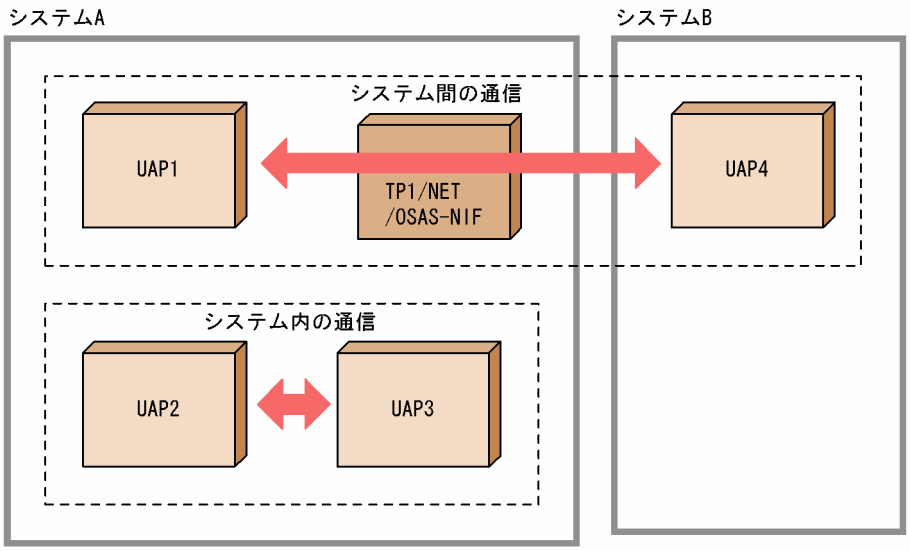
TP1/NET/OSAS-NIF は，OpenTP1 システムで NIF/OSI プロトコルを使用してシステム間通信をするためのプログラムです。この章では，TP1/NET/OSAS-NIF の概要について説明します。

1.1　システム間通信の概要

複数のシステムを接続して，それぞれのシステムのユーザアプリケーションプログラム間でメッセージの送受信をすることを**システム間通信**といいます。

メッセージの送受信には，システム間で通信する場合とシステム内で通信する場合があります。システム間の通信は，異なるシステムにあるアプリケーションプログラムとのメッセージの送受信をいいます。また，システム内の通信は，同じシステム内にあるアプリケーションプログラムとのメッセージの送受信をいいます。システム間とシステム内の通信の違いを次の図に示します。

図 1-1　システム間の通信とシステム内の通信



TP1/NET/OSAS-NIF は，OpenTP1 システムと異なるシステムとの通信をするプログラムです。TP1/NET/OSAS-NIF を使用すると，自システム内でのアプリケーション起動と同様の手順で，異なるシステムとメッセージの送受信ができます。

TP1/NET/OSAS-NIF には，メッセージの再送機能およびフロー制御機能があり，システム間通信の信頼性を高めています。

TP1/NET/OSAS-NIF は，OpenTP1 システムに組み込んで使用します。TP1/NET/OSAS-NIF は，拡張 HNA の上で動作し，NIF/OSI プロトコルを使用してシステム間通信をします。

NIF/OSI プロトコルは，OSI 上位層（5 層，6 層および 7 層の一部）の共通基盤である OSAS を使用して，システム間通信を提供するプロトコルです。

TP1/NET/OSAS-NIF のネットワーク構成例を図 1-2 に，TP1/NET/OSAS-NIF の OSI7 層構造の中での機能範囲を図 1-3 に示します。

図 1-2　TP1/NET/OSAS-NIF のネットワーク構成例

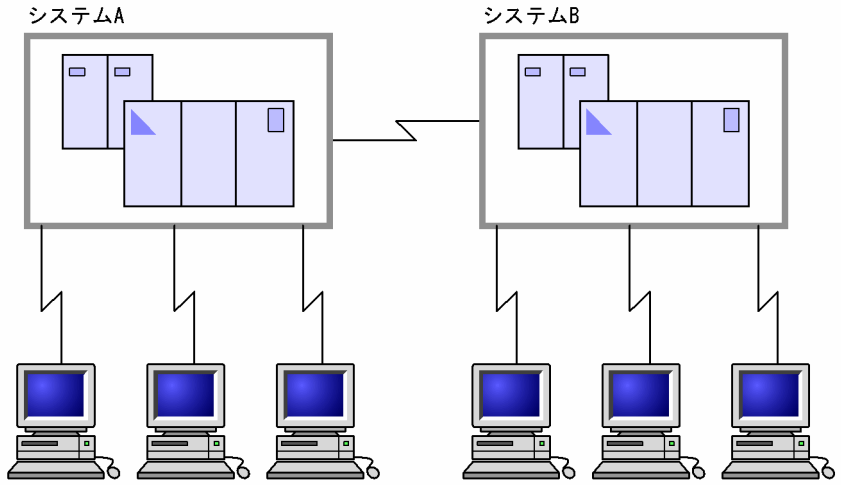
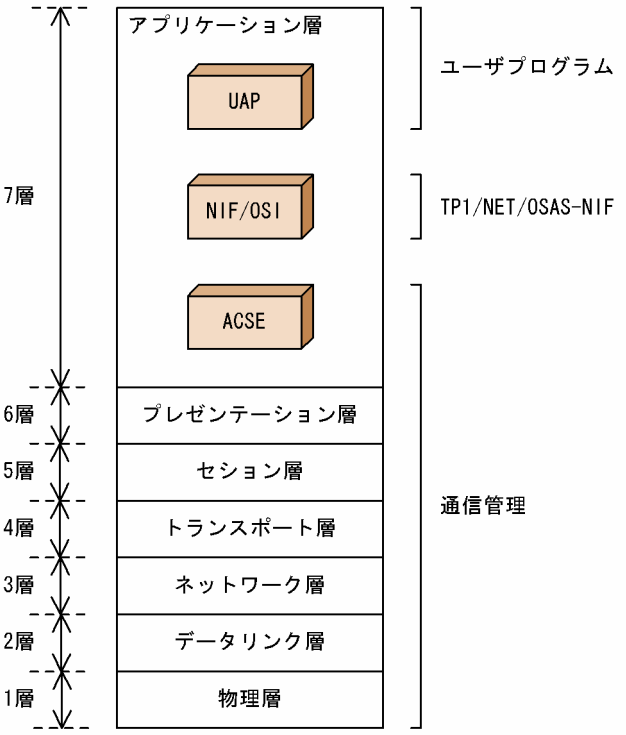


図 1-3　TP1/NET/OSAS-NIF の OSI7 層構造の中での機能範囲



1.2　システム間通信の形態

システム間通信を使用すると，自システムで発生したメッセージを相手システムで処理したり，相手システムで発生したメッセージを，自システムで処理したりできます。

TP1/NET/OSAS-NIF で行うシステム間通信の形態は，問い合わせ応答形態，同期型問い合わせ応答形態，分岐送信形態および一方受信形態です。それぞれの通信形態の説明をします。

1.2.1　問い合わせ応答形態

問い合わせ応答形態は，自システムで発生したメッセージを，相手システムへ送信し，その処理結果を受信する形態です。

自システムから相手システムへ送信するメッセージを，**問い合わせメッセージ**といいます。相手システムから受信する処理結果を**応答メッセージ**といいます。

問い合わせ応答形態のシステム間通信の例を次の図に示します。

図 1-4　問い合わせ応答形態のシステム間通信の例

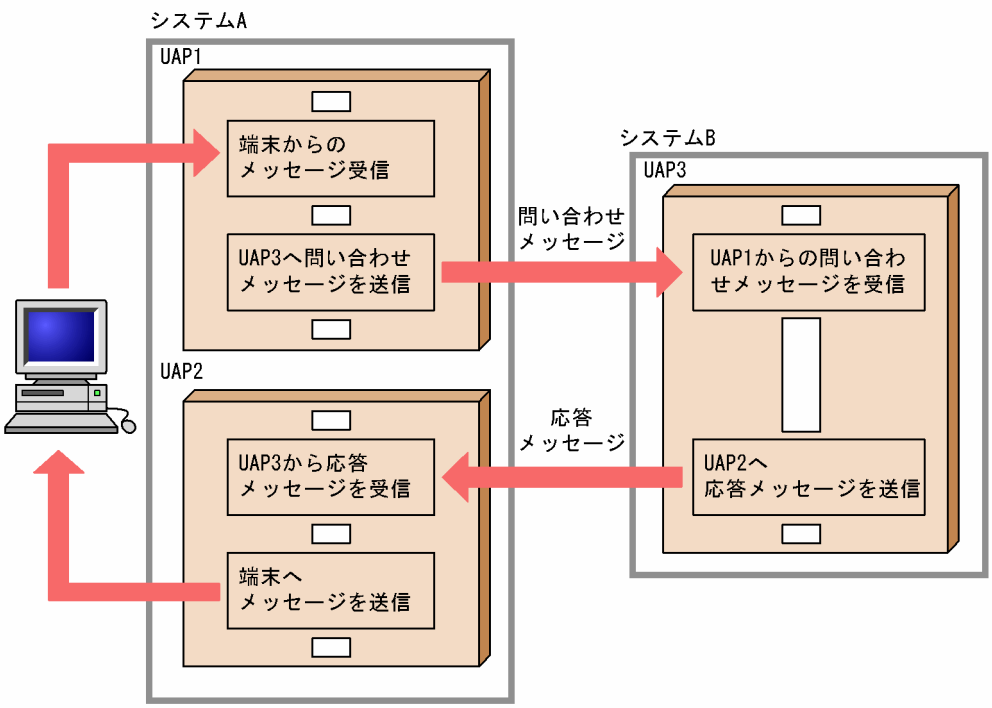


図 1-4 に示す通信形態のうち，TP1/NET/OSAS-NIF の通信形態システムを次の表に示します。

表 1-1　TP1/NET/OSAS-NIF の通信形態システム

通信形態	図 1-4 の通信形態システム
問い合わせ応答形態	システム A：問い合わせメッセージの送信と応答メッセージの受信

通信形態	図 1-4 の通信形態システム
問い合わせ応答形態	システム B：問い合わせメッセージの受信と応答メッセージの送信

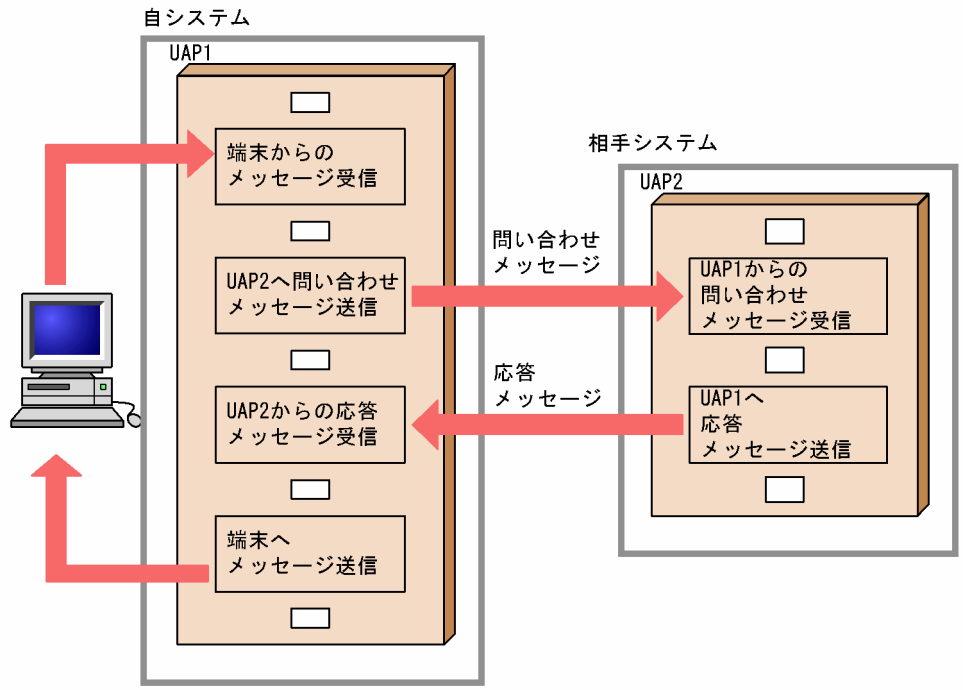
1.2.2　同期型問い合わせ応答形態

同期型問い合わせ応答形態は，自システムで発生したメッセージを相手システムへ送信し，その処理結果のメッセージを受信する形態です。UAP でのメッセージ送受信を，システム間で同期を取りたいときに使用します。

自システムから相手システムへ送信するメッセージを，**問い合わせメッセージ**といいます。相手システムから受信する処理結果のメッセージを**応答メッセージ**といいます。

同期型問い合わせ応答形態のシステム間通信の例を次の図に示します。

図 1-5　同期型問い合わせ応答形態のシステム間通信の例



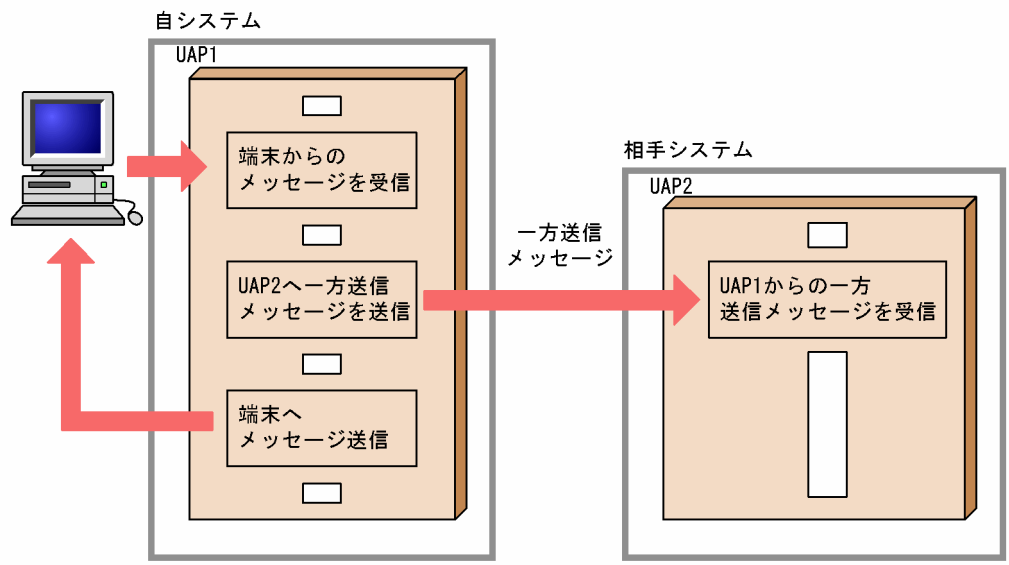
1.2.3　分岐送信形態

分岐送信形態は，自システムで発生したメッセージを，通信相手システムに送信する通信形態です。その場合，相手システムの処理結果の返送は期待しません。

自システムから相手システムに送信するメッセージを，**一方送信メッセージ**といいます。

分岐送信形態のシステム間通信の例を次の図に示します。

図 1-6　分岐送信形態のシステム間通信の例

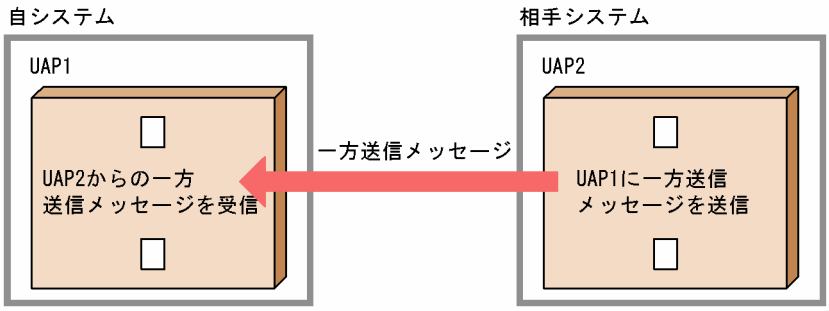


1.2.4　一方受信形態

一方受信形態は，相手システムから送信されたメッセージを受信する通信形態です。相手システムから受信するメッセージは，分岐送信形態の場合と同様に**一方送信メッセージ**といいます。

一方受信形態のシステム間通信の例を次の図に示します。

図 1-7　一方受信形態のシステム間通信の例

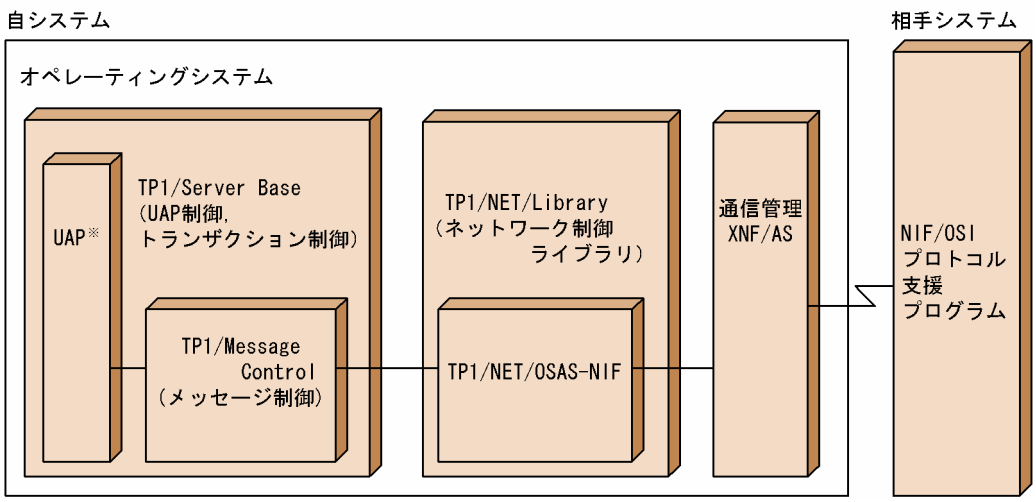


1.3　ソフトウェア構成の例

TP1/NET/OSAS-NIF 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/OSAS-NIF 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 1-8　TP1/NET/OSAS-NIF を組み込んだソフトウェア構成



注※

TP1/NET/OSAS-NIF 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。

また，TP1/NET/OSAS-NIF が AP 間通信で使用するプロトコル，および主な通信相手プログラムを次の表に示します。

表 1-2　TP1/NET/OSAS-NIF に適用する AP 間通信のプロトコル

プロトコル	主な通信相手
NIF/OSI プロトコル	XDM/DCCM3 システムの「VOS3 OSAS/UA2/DCCM3」※
	TMS-4V/SP システムの「VOS3 OSAS/NF/4VSP」

注※

「VOS3 OSAS/UA2/DCCM3」との通信では，問い合わせ応答形態および同期型問い合わせ応答形態は使用できません。

2 機能

TP1/NET/OSAS-NIF はシステム間通信機能を提供し，メッセージの送受信をします。この章では，コネクションの確立や論理端末の端末タイプなどのシステム間通信の機能，システム間通信のメッセージの送受信および TP1/NET/OSAS-NIF 固有の機能について説明します。

2.1　システム間通信の機能

TP1/NET/OSAS-NIF は，NIF/OSI プロトコルを支援している相手システムとシステム間通信をします。システム間通信をするには，論理端末，アプリケーションプログラムなどを関連づけてシステムを作成する必要があります。この節では，論理端末やアプリケーションプログラムを使用したシステム間通信の機能について説明します。

2.1.1　コネクションと論理端末の関係

TP1/NET/OSAS-NIF を使用してメッセージの送受信をする場合，ユーザはコネクションと論理端末を対応づけて定義をする必要があります。

コネクションは，システム間でメッセージの送受信をするための論理的な通信路です。コネクションは TP1/NET/OSAS-NIF の通信管理側の通信接点であり，TP1/NET/OSAS-NIF と通信管理はコネクション単位にメッセージの送受信をします。コネクションは NIF/OSI プロトコルのアソシエーションに対応します。

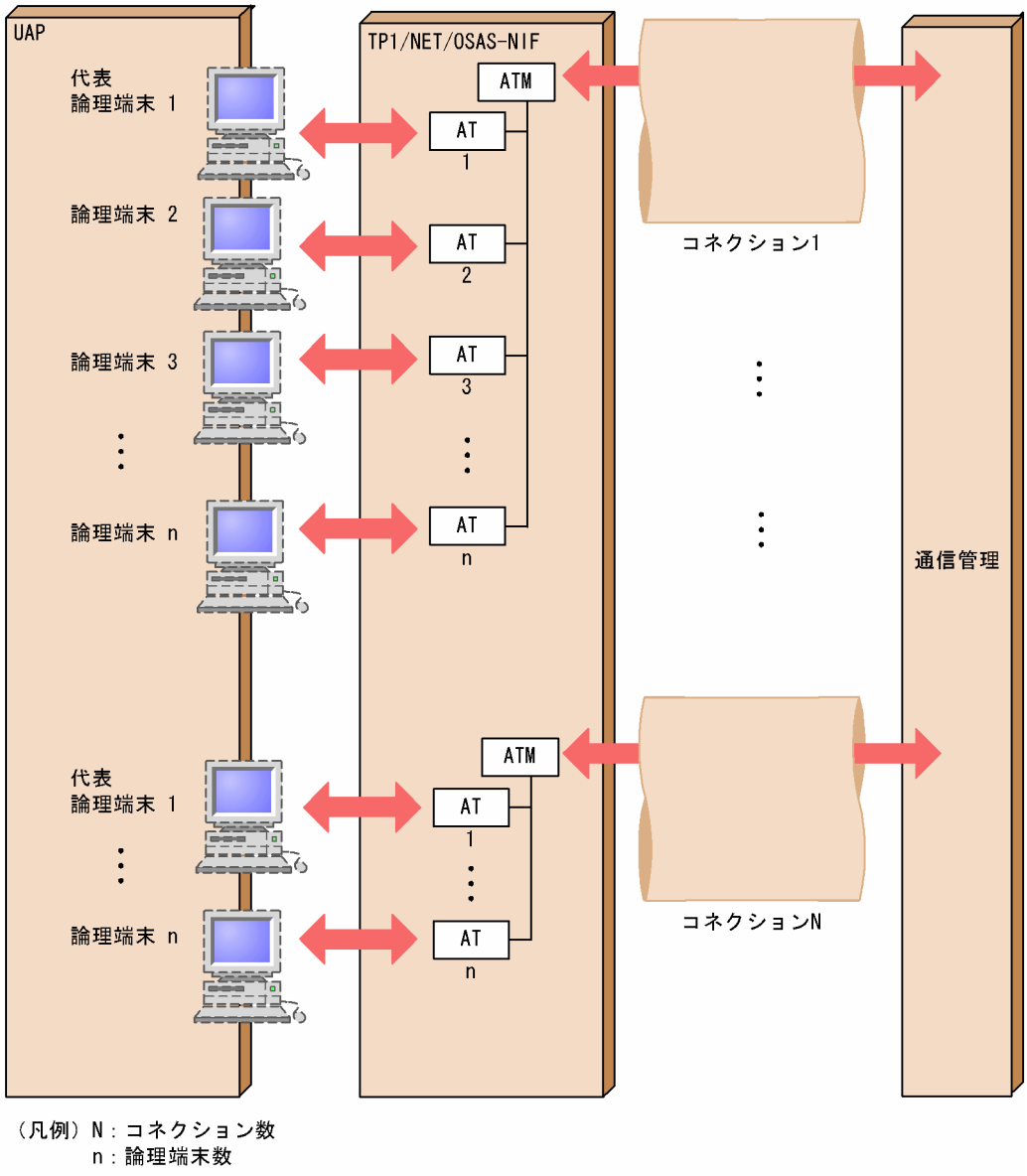
コネクションはコネクション定義（mcftalccn）で指定します。TP1/NET/OSAS-NIF では，システムに 1～512 のコネクションを定義できます。

論理端末（LE）は，通信相手システムとのメッセージの送受信のため，コネクションに対して通信を要求します。論理端末は TP1/NET/OSAS-NIF の UAP 側の通信接点であり，TP1/NET/OSAS-NIF と UAP は論理端末単位にメッセージの送受信をします。論理端末は NIF/OSI プロトコルのエージェント（AT）に対応します。エージェントには，コネクションの管理，およびエージェントの初期化，管理をするエージェントマネジャ（ATM）があります。

論理端末は論理端末定義（mcftalcle）で指定します。TP1/NET/OSAS-NIF では，論理端末は一つのコネクションに対し，複数指定できます。論理端末を複数指定した場合，その中の一つを代表論理端末にします。代表論理端末は，コネクションに関するイベントを受信します。

コネクションと論理端末の関係を次の図に示します。

図 2-1　コネクションと論理端末の関係



2.1.2　論理端末とアプリケーション

TP1/NET/OSAS-NIF では，システム間通信をするために，論理端末の端末タイプおよびアプリケーションの型を定義します。

(1)　論理端末の端末タイプ

論理端末は，相手システムとのメッセージの送受信をするための通信接点です。TP1/NET/OSAS-NIF では利用形態によって，次に示す端末タイプがあります。**論理端末の端末タイプ**は，論理端末定義（mcftalcle -t）で指定します。

- request 型：問い合わせ型論理端末

- request 型（同期型）：問い合わせ型論理端末（同期型）※
- reply 型：応答型論理端末
- send 型：送信型論理端末
- receive 型：受信型論理端末

注※

論理端末定義（mcftalcle -d）の sync オペランドで yes を指定した場合

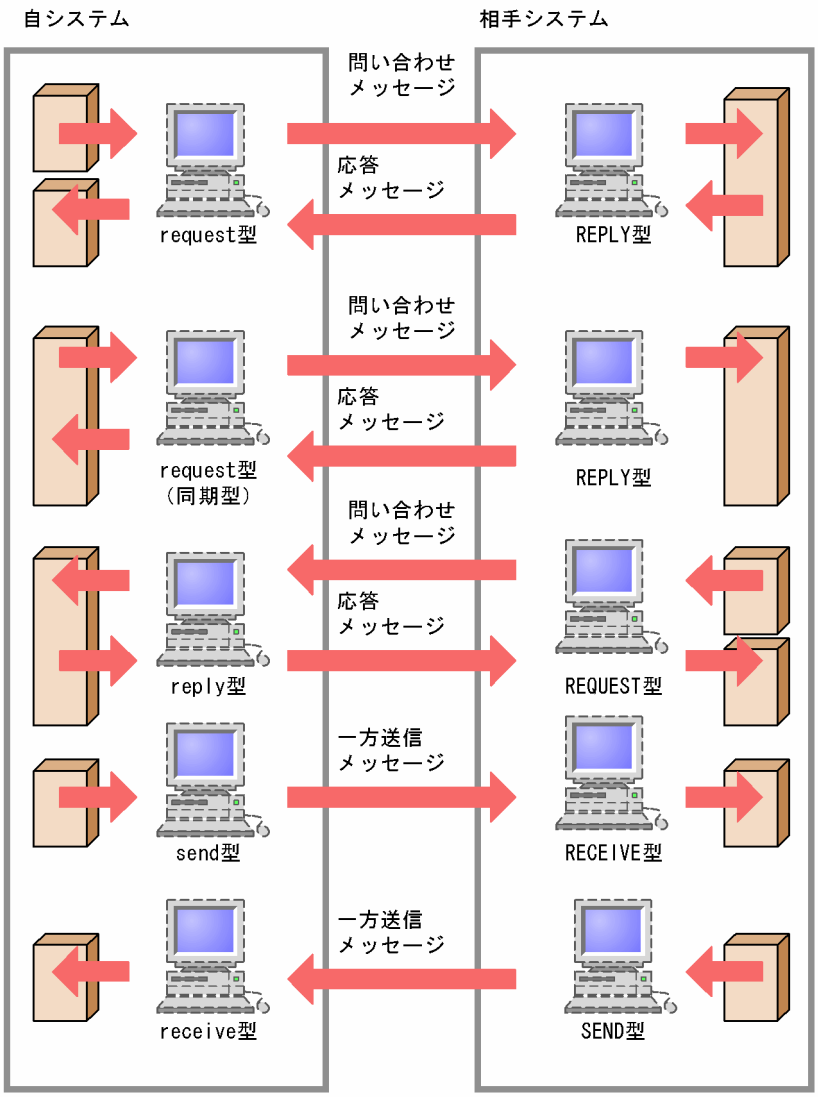
自システムから相手システムへ問い合わせメッセージを送信する場合，request 型の論理端末から相手システムの REPLY 型のエージェントへ送信します。また，相手システムから応答メッセージを受信する場合は，相手システムの REPLY 型のエージェントから送信したメッセージを，自システムの request 型の論理端末で受信します。

相手システムからの問い合わせメッセージを受信する場合，相手システムの REQUEST 型のエージェントからメッセージを送信し，自システムの reply 型の論理端末がメッセージを受信します。また，相手システムへ応答メッセージを送信する場合は，自システムの reply 型の論理端末から相手システムの REQUEST 型のエージェントへ送信します。

自システムから相手システムに一方送信メッセージを送信する場合，send 型の論理端末から相手システムの RECEIVE 型のエージェントに送信します。また，相手システムからメッセージを送信する場合は，相手システムの SEND 型のエージェントからメッセージを送信し，自システムの receive 型の論理端末がメッセージを受信します。

システム間通信と論理端末の端末タイプの関係の例を次の図に示します。

図 2-2　システム間通信と論理端末の端末タイプの関係



(2)　論理端末とアプリケーションの型の関係

システム間通信をするには，論理端末の端末タイプ，メッセージの種類，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を関連づけます。関連づける項目の関係を次の表に示します。

表 2-1　論理端末の端末タイプ，メッセージの種類，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	メッセージの種類	アプリケーションの型	UAP インタフェース	通信形態
request 型（問い合わせ型）	問い合わせメッセージ	任意	EXECAP	問い合わせ応答形態
			SEND	
			RESEND	
	応答メッセージ		RECEIVE	

論理端末の端末タイプ	メッセージの種類	アプリケーションの型	UAP インタフェース	通信形態
request 型（同期型）（問い合わせ型（同期型））	問い合わせメッセージ	任意	SENDRECV※1	同期型問い合わせ応答形態
	応答メッセージ		RECVSYNC※2	
reply 型（応答型）	問い合わせメッセージ	応答型	RECEIVE	問い合わせ応答形態
	応答メッセージ		REPLY	
send 型（送信型）	一方送信メッセージ	任意	EXECAP	分岐送信形態
			SEND	
			RESEND	
receive 型（受信型）		非応答型	RECEIVE	一方受信形態

注※1

セグメント送信時と先頭セグメント受信時だけ使用できます。

注※2

後続セグメント受信時だけ使用できます。

(3) システム間通信の形態

TP1/NET/OSAS-NIF で実現できるシステム間通信の形態を示します。

(a) 問い合わせ応答形態を使用した通信形態

UAP から EXECAP 要求を発行して，問い合わせメッセージを送信する場合，EXECAP 要求でアプリケーション名を指定します。このとき，アプリケーション名に対応するアプリケーション属性定義（mcfaalcap -n）の lname オペランドで request 型論理端末を指定してください。または，アプリケーション属性定義（mcfaalcap -n）の cname オペランドで request 型論理端末の定義があるコネクションのコネクション ID を指定してください。コネクション ID を指定した場合，TP1/NET/OSAS-NIF は指定されたコネクションから request 型論理端末を使用します。

該当する論理端末が使用中で応答メッセージ未受信の状態であるとき，EXECAP 要求を発行できません。論理端末が応答メッセージを受信して，論理端末の使用状態を解除したあと，EXECAP 要求を発行できます。

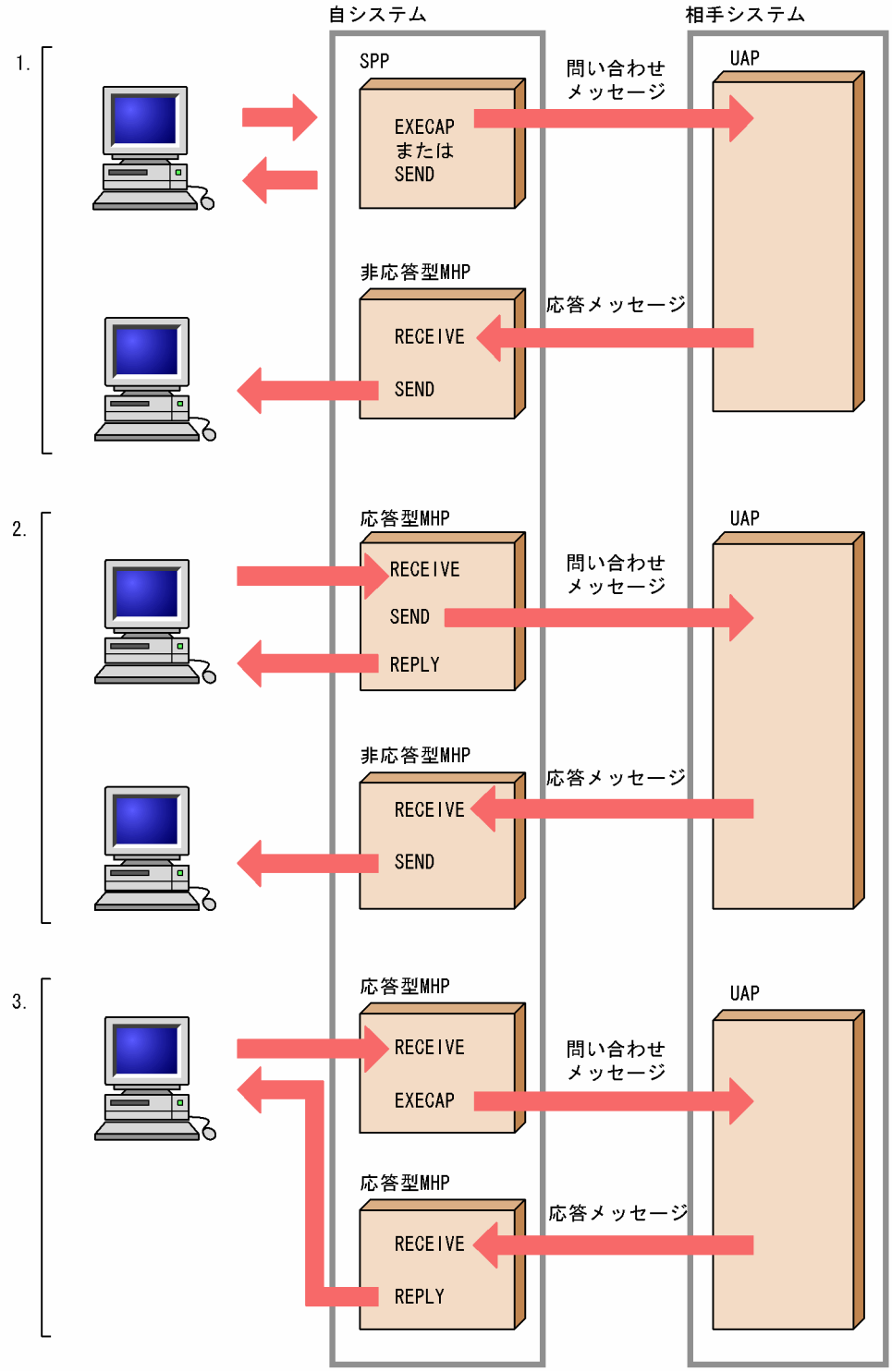
UAP から SEND 要求を発行して，問い合わせメッセージを送信する場合，SEND 要求で request 型論理端末を指定します。

該当する論理端末が使用中で応答メッセージ未受信の状態でも，論理端末定義（mcftalcle -m）で指定した出力メッセージ格納数を上限として，SEND 要求を発行できます。

なお，同じコネクションで EXECAP 要求と SEND 要求を混在して使用しないでください。EXECAP 要求で使用中の request 型論理端末に対して，SEND 要求が発行されることがあります。

問い合わせ応答形態を使用したシステム間通信の形態例を次の図に示します。

図 2-3　問い合わせ応答形態を使用した通信形態の例



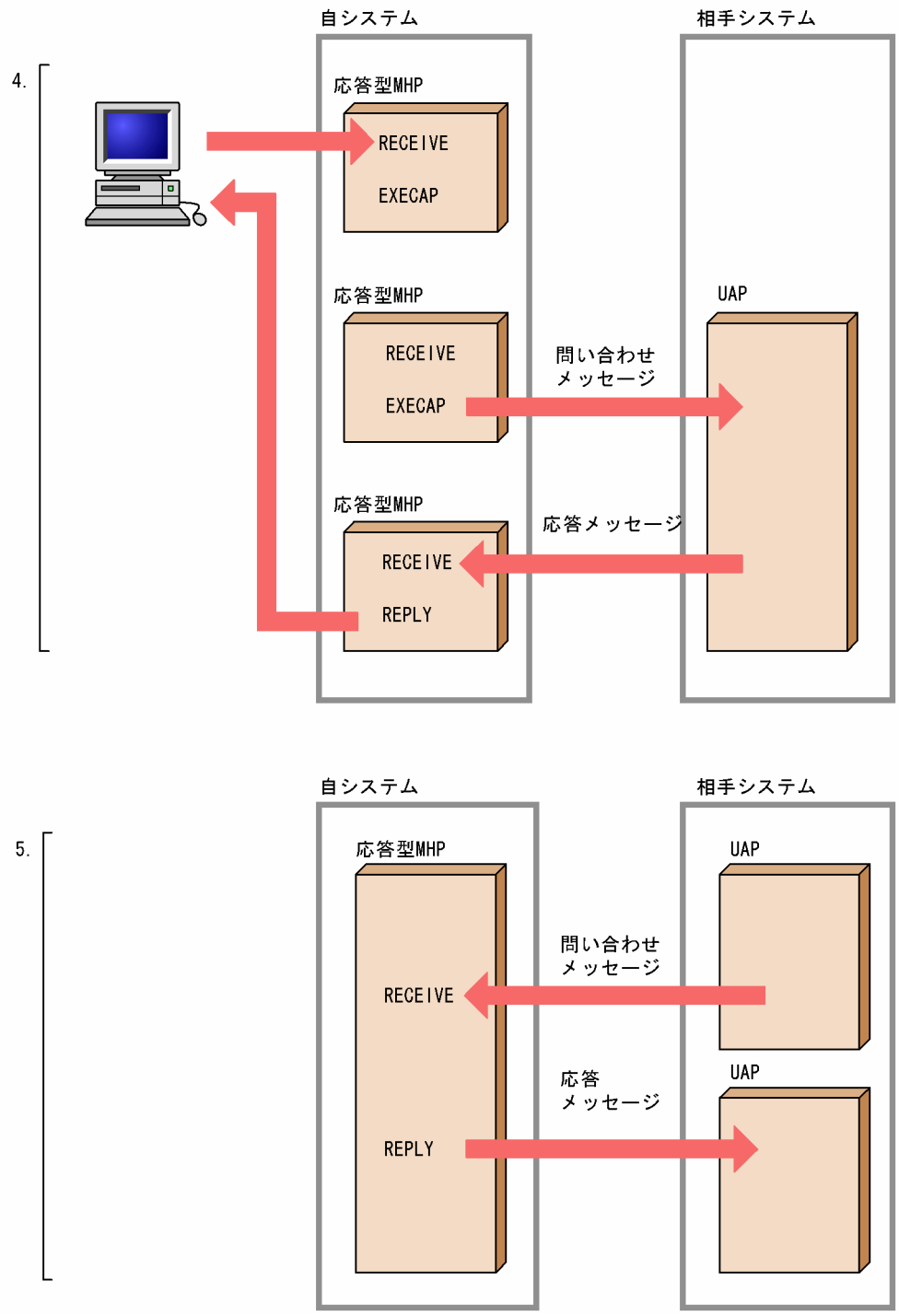


図 2-3 に示した五つの通信形態例について説明します。

1. 端末から SPP が起動され，EXECAP 要求または SEND 要求を発行して，問い合わせメッセージを相手システムへ送信します。その後，相手システムからの応答メッセージを受信して非応答型 MHP を起動し，RECEIVE 要求を発行して相手システムからのメッセージを受信する通信形態です。
2. 端末から応答型 MHP が起動され，SEND 要求を発行して，問い合わせメッセージを相手システムへ送信します。その後，相手システムからの応答メッセージを受信して非応答型 MHP を起動し，RECEIVE 要求を発行して相手システムからのメッセージを受信する通信形態です。

また，端末から起動された応答型 MHP は，SEND 要求発行後，REPLY 要求を発行することによって，応答メッセージを端末へ送信できます。

3. 端末から応答型 MHP が起動され，EXECAP 要求を発行して，問い合わせメッセージを相手システムへ送信します。その後，相手システムからの応答メッセージを受信して応答型 MHP を起動し，RECEIVE 要求を発行して相手システムからのメッセージを受信する通信形態です。

   request 型論理端末に対する EXECAP 要求は，発行元の MHP の応答型の属性を引き継ぐため，端末から起動された応答型 MHP は，EXECAP 要求発行後，REPLY 要求を発行できません。入力元の端末に対する応答メッセージの送信は，相手システムからの応答メッセージ受信で起動された応答型 MHP から REPLY 要求を発行することによって行います。

4. 端末から応答型 MHP が起動され，EXECAP 要求を発行して，自システム内の応答型 MHP を起動します。自システム内で起動された MHP は EXECAP 要求を発行して，問い合わせメッセージを相手システムへ送信します。その後，相手システムからの応答メッセージを受信して応答型 MHP を起動し，RECEIVE 要求を発行して相手システムからのメッセージを受信する通信形態です。

   request 型論理端末に対する EXECAP 要求は，発行元の MHP の応答型の属性を引き継ぐため，端末から起動された応答型 MHP は，EXECAP 要求発行後，REPLY 要求を発行できません。入力元の端末に対する応答メッセージの送信は，相手システムからの応答メッセージ受信で起動された応答型 MHP から REPLY 要求を発行することによって行います。

5. 相手システムから問い合わせメッセージを受信することによって，応答型 MHP が起動され，RECEIVE 要求を発行して相手システムからのメッセージを受信します。その後，REPLY 要求を発行して，相手システムへ応答メッセージを送信する通信形態です。

なお，端末からの UAP の起動，自システム内での MHP の起動および UAP から端末へのメッセージ送信は，MCF またはほかの OpenTP1 プロトコルで支援している機能です。

(b) 同期型問い合わせ応答形態を使用した通信形態

UAP から SENDRECV 要求を発行して，問い合わせメッセージを送信する場合，SENDRECV 要求で request 型論理端末（同期型）を指定します。

該当する request 型論理端末が使用中で，応答メッセージ未受信の状態であるとき，SENDRECV 要求は発行できません。応答メッセージを受信して，論理端末の使用状態が解除されたあと，SENDRECV 要求を発行できます。

同期型問い合わせ応答形態を使用した通信形態例を次の図に示します。

図 2-4　同期型問い合わせ応答形態を使用した通信形態の例

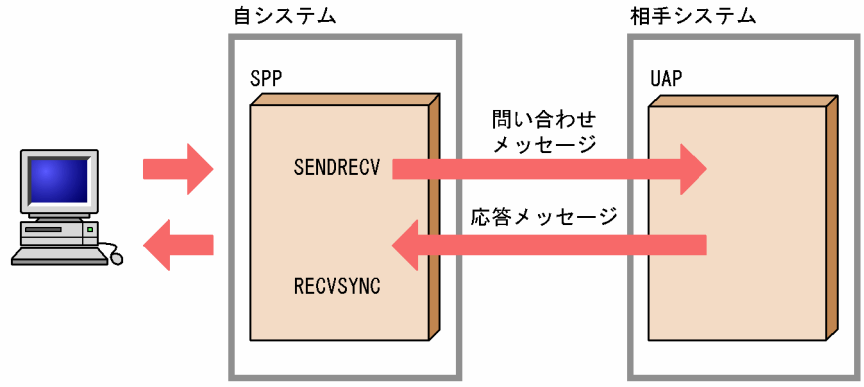


図 2-4 は，端末から SPP が起動され，SENDRECV 要求を発行して，問い合わせメッセージを相手システムへ送信する例です。相手システムからの応答メッセージ受信を契機に SENDRECV 要求がリターンすることで，SPP は先頭セグメントを受信します。その後，SPP は RECVSYNC 要求を発行して，後続セグメントを受信する通信形態です。SPP は，TP1/NET/OSAS-NIF を介して問い合わせ応答をしたあと，端末のクライアント側の UAP へ制御を移せます。

なお，端末からの UAP の起動および UAP から端末へのメッセージ送信は，MCF またはほかの OpenTP1 プロトコルで支援している機能です。

(c)　分岐送信形態および一方受信形態を使用した通信形態

UAP から EXECAP 要求を発行して，一方送信メッセージを送信する場合，EXECAP 要求でアプリケーション名を指定します。このとき，アプリケーション名に対応するアプリケーション属性定義（mcfaalcap -n）の lname オペランドで send 型論理端末を指定します。

UAP から SEND 要求を発行して，一方送信メッセージを送信する場合，SEND 要求で send 型論理端末を指定します。

該当する論理端末が送信中の状態でも論理端末定義（mcftalcle -m）で指定した出力メッセージ格納数を上限として EXECAP 要求または SEND 要求を発行できます。

分岐送信形態および一方受信形態を使用した通信形態例を次の図に示します。

図 2-5　分岐送信形態および一方受信形態を使用した通信形態の例

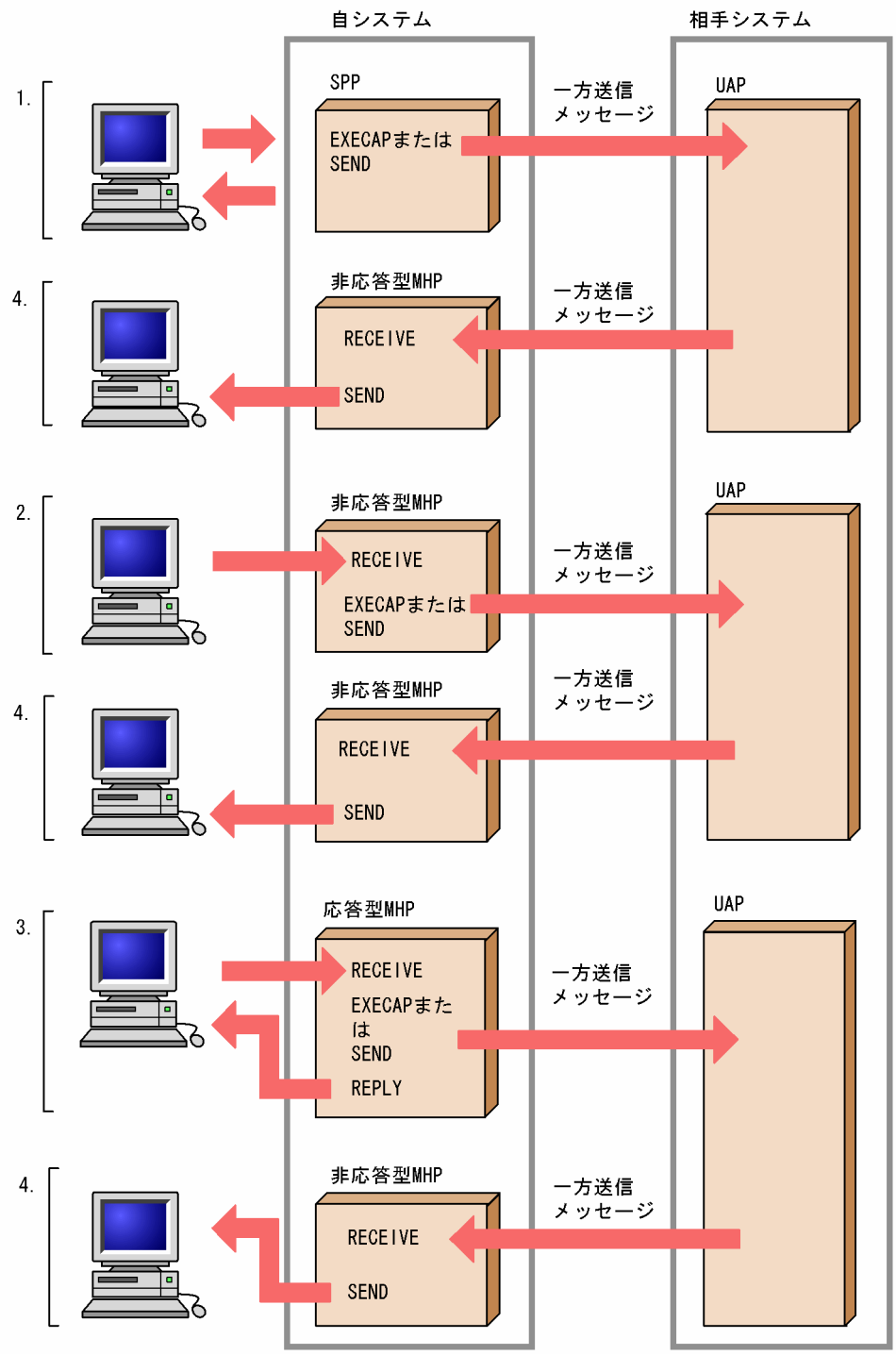


図 2-5 に示した四つの通信形態例について説明します。

1. 端末から SPP が起動され，EXECAP 要求または SEND 要求を発行して，一方送信メッセージを送信する通信形態です。
2. 端末から非応答型 MHP が起動され，EXECAP 要求または SEND 要求を発行して，一方送信メッセージを送信する通信形態です。

3. 端末から応答型 MHP が起動され，EXECAP 要求または SEND 要求を発行して，一方送信メッセージを送信する通信形態です。また，EXECAP 要求または SEND 要求発行後，REPLY 要求を発行することによって，応答メッセージを端末へ送信できます。
4. 相手システムから一方送信メッセージを受信して，非応答型の MHP を起動する通信形態です。起動された MHP は，RECEIVE 要求を発行して相手システムからのメッセージを受信します。また，端末へは SEND 要求を発行してメッセージを送信できます。

なお，端末からの UAP の起動および UAP から端末へのメッセージ送信は，MCF またはほかの OpenTP1 プロトコルで支援している機能です。

2.1.3 セグメントの分割と組み立て

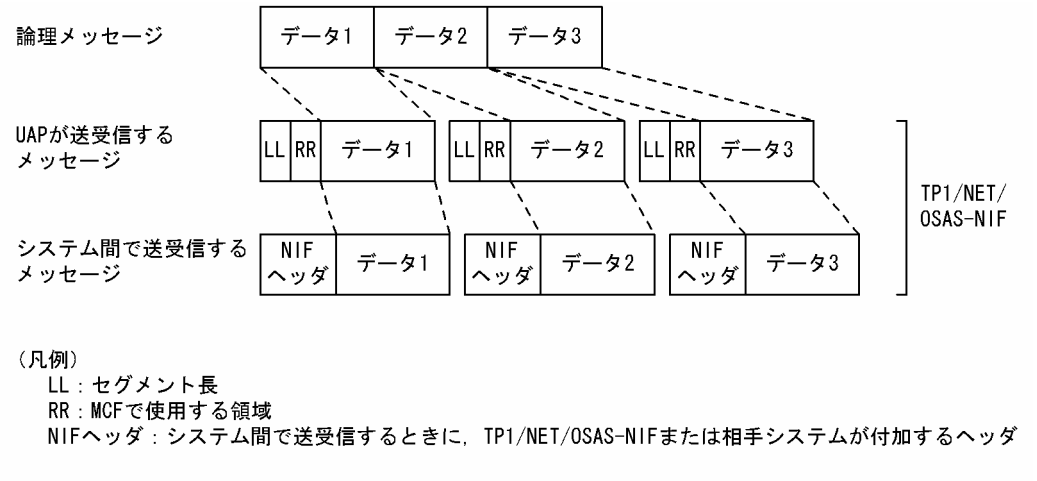
メッセージは，UAP でセグメント単位に分割してシステム間で送受信をします。

メッセージの送信時，TP1/NET/OSAS-NIF は，UAP から受け取ったメッセージをセグメント単位で相手システムへ送信します。このとき，TP1/NET/OSAS-NIF はセグメントの先頭に NIF ヘッダを付けます。送信できる 1 セグメントの長さは，1～32000 バイトです。ただし，相手システムによって，受信できるセグメント長が異なりますので確認してから送信してください。

また，メッセージの受信時は，相手システムから受信したメッセージをセグメント単位で UAP へ渡します。受信できる 1 セグメントの長さは 1～65462 バイトです。

メッセージを送受信するときのメッセージの形式の変化について，次の図に示します。

図 2-6 メッセージの形式の変化



UAP がメッセージ送受信の関数で処理するセグメントの先頭には，MCF で使用するヘッダ領域があります。このヘッダ領域の長さによって，バッファ形式 1 とバッファ形式 2 があります。通常，バッファ形式 1 を使用します。

2.1.4　コネクションの確立

TP1/NET/OSAS-NIF は，相手システムとの間で，論理的な通信路（コネクション）を確立してメッセージの送受信をします。

コネクションの確立には，**発呼型**と**着呼型**があります。

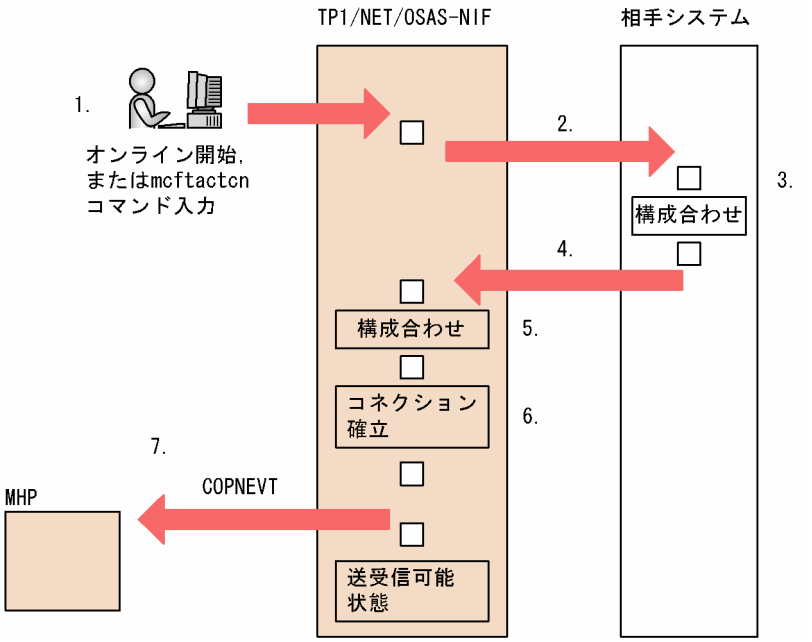
(1)　発呼型のコネクション確立

発呼型は，TP1/NET/OSAS-NIF からコネクションを確立する方法です。次に示す三つがあります。

- オンライン開始時に自動的に確立します（コネクション定義（mcftalccn -i）で auto を指定）。
- 運用コマンド（mcftactcn）で確立します。
- API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）で確立します。

オンライン開始または運用コマンド入力による発呼型のコネクションの確立について，次の図に示します。

図 2-7　発呼型のコネクションの確立（オンライン開始または運用コマンド入力）

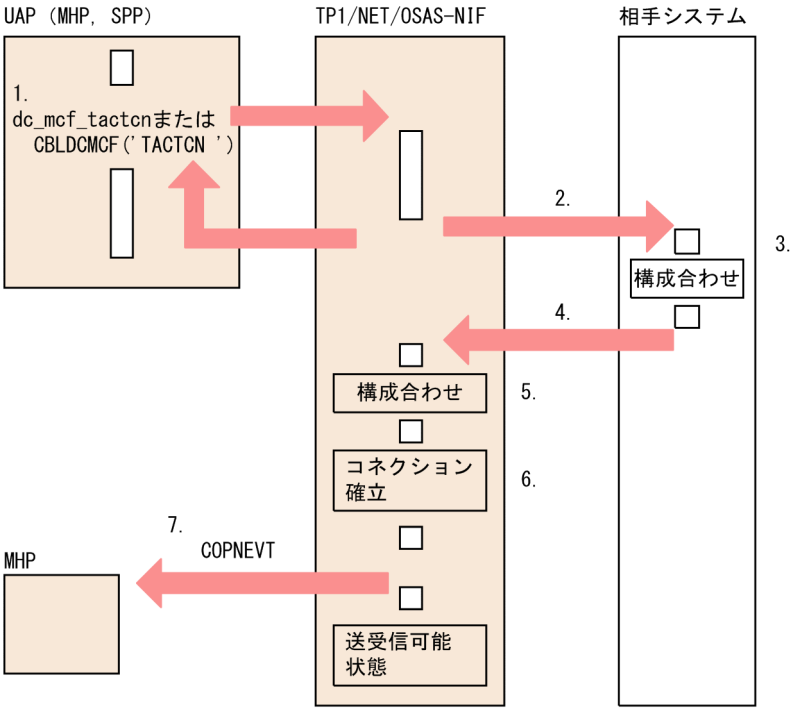


1. オンラインを開始します。または，運用コマンド（mcftactcn）を入力します。
2. TP1/NET/OSAS-NIF は，MCF 通信構成定義情報に基づいて，初期設定要求を相手システムに送信します。
3. 相手システムは TP1/NET/OSAS-NIF から受信した初期設定要求の構成情報の内容と突き合わせます。
4. 相手システムから初期設定回答として構成情報を受信します。
5. 初期設定回答に従い，構成合わせをします。なお，構成合わせについては，「(3)　システム間の構成合わせ」を参照してください。
6. コネクションが確立されます。

7. TP1/NET/OSAS-NIF は，コネクションが確立されると状態通知イベント（COPNEVT）を通知します。アプリケーション定義に COPNEVT が定義されていない場合は，MHP は起動しません。

API 発行による発呼型のコネクションの確立について，次の図に示します。

図 2-8　発呼型のコネクションの確立（API 発行）



1. API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）を発行します。
2. TP1/NET/OSAS-NIF は，MCF 通信構成定義情報に基づいて，初期設定要求を相手システムに送信します。
3. 相手システムは TP1/NET/OSAS-NIF から受信した初期設定要求の構成情報の内容と突き合わせます。
4. 相手システムから初期設定回答として構成情報を受信します。
5. 初期設定回答に従い，構成合わせをします。なお，構成合わせについては，「(3)　システム間の構成合わせ」を参照してください。
6. コネクションが確立されます。
7. TP1/NET/OSAS-NIF は，コネクションが確立されると状態通知イベント（COPNEVT）を通知します。アプリケーション定義に COPNEVT が定義されていない場合は，MHP は起動しません。

発呼型の場合の**コネクションの確立方式**には，**重畳型**と**非重畳型**があり，相手システムと確立方式を合わせておく必要があります。コネクションの確立方式は，コネクション定義（mcftalccn -x）で指定します。

(2)　着呼型のコネクション確立

着呼型は，相手システムからの要求でコネクションを確立する方法です。

TP1/NET/OSAS-NIF は，あらかじめ，次のどれかを行う必要があります。

- コネクション定義（mcftalccn -i）に auto を指定する。
- 運用コマンド（mcftactcn）を入力する。
- API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）を発行する。

その後，相手システムからのコネクション確立要求の初期設定要求を受け，初期設定回答を送信すると，コネクションが確立します。コネクションの確立方式は，相手システムの確立方式に合わせます。

(3)　システム間の構成合わせ

TP1/NET/OSAS-NIF は，コネクション確立時に相手システムとの構成情報を突き合わせます。次に示す三つの情報を突き合わせ，相手システムと構成の同期を取ります。

- NIF 通番の最大値
- タイマ監視値
- 論理端末構成

(a)　NIF 通番の最大値の突き合わせ

NIF 通番とは，送受信メッセージごとに TP1/NET/OSAS-NIF が設定する通し番号のことです。

TP1/NET/OSAS-NIF は，NIF 通番の最大値の突き合わせをします。突き合わせの結果，不一致となる場合は，**小さい方の値**を最大値とします。

NIF 通番については，「2.5.2　NIF 通番」を参照してください。

(b)　タイマ監視値の突き合わせ

TP1/NET/OSAS-NIF は，次に示す監視時間の値を突き合わせます。

- 正常処理監視時間
- ユーザ処理監視時間
- 送達確認監視時間
- 連続送信監視時間

突き合わせの結果，値が不一致となる場合は，**大きい値**に合わせます。ただし，どちらかが 0 を指定した場合は，タイマ監視はしません。また，終了処理監視時間は突き合わせをしません。

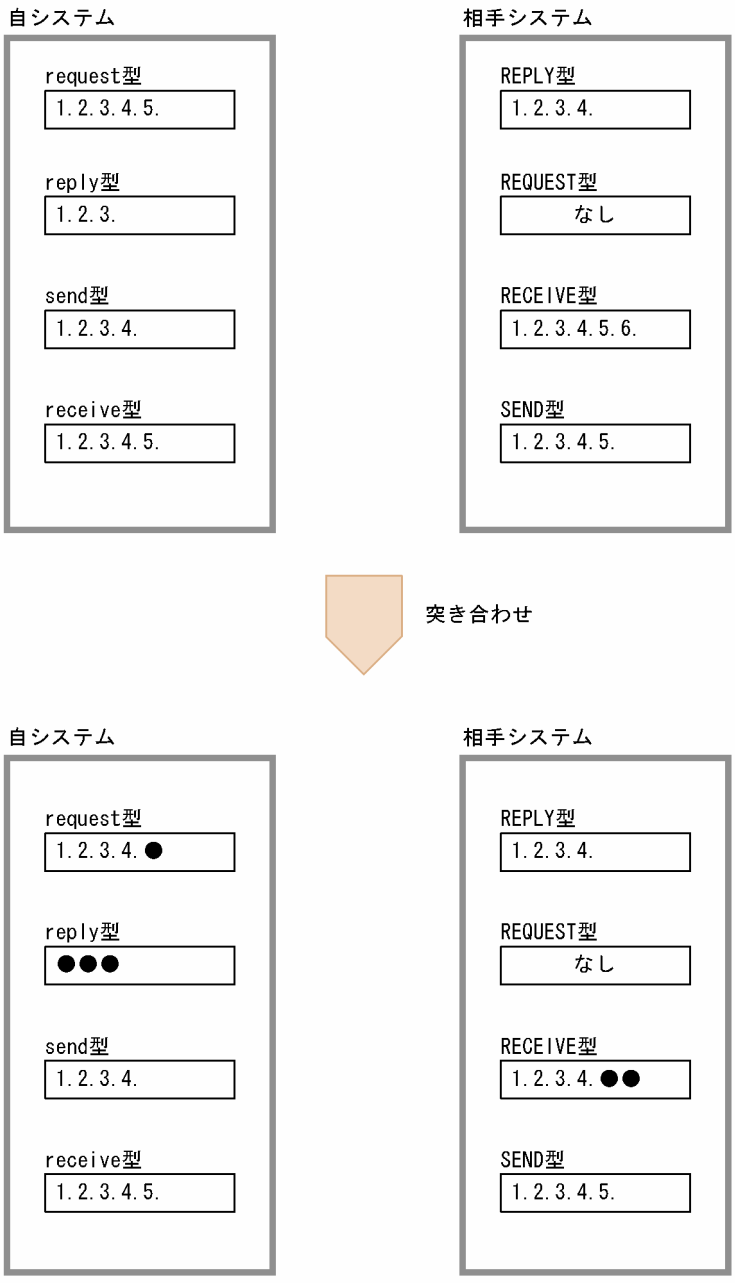
タイマ監視の詳細については，「2.6　タイマ監視」を参照してください。

(c)　論理端末構成の突き合わせ

TP1/NET/OSAS-NIF は，コネクション確立時に，対応する自システムの論理端末と相手システムのエージェントの個数を突き合わせます。突き合わせの結果，不一致となる場合は**少ない方の個数**に合わせてシステム間通信の条件を設定します。論理端末数が 0 となるとき，TP1/NET/OSAS-NIF はコネクションを確立しません。

論理端末構成の突き合わせの例を次の図に示します。

図 2-9　論理端末構成の突き合わせの例



（凡例）1.，2.，3.，4.，5.，6.：論理端末。
●：突き合わせの結果，使用できない論理端末。

(4)　コネクション確立処理中のオンライン終了

コネクションの状態が確立処理中※でのオンライン終了時動作をプロトコル共通定義（mcftpcmn -e）の termactb オペランドで指定できます。

注※

コネクションが次に示す状態のときを，コネクション確立処理中といいます。

- 発呼型のコネクションで，コネクション確立を再試行しているとき。
- 着呼型のコネクションで，相手システムからのコネクション確立要求を待っているとき。

termactb オペランドの指定内容と TP1/NET/OSAS-NIF の動作を次の表に示します。

表 2-2　termactb オペランドの指定内容と TP1/NET/OSAS-NIF の動作

termactb オペランドの指定内容	TP1/NET/OSAS-NIF の動作
wait	コネクション確立処理中にオンラインの終了を受け付けた場合，コネクション確立処理が完了するまで終了処理を待ち合わせます。 コネクション確立処理は，相手システムとコネクションを確立するか，コネクション確立再試行回数を超過した場合に完了します。 長時間コネクション確立処理が完了しない場合，dcstop コマンドがタイムアウトし，OpenTP1 が異常終了します。
cont	コネクション確立処理中にオンラインの終了を受け付けた場合，コネクション確立処理を停止し，終了処理を続行します。

termactb オペランドの指定が wait の場合は，あらかじめ，すべてのコネクションを解放したあとに，オンラインを終了してください。オンライン終了中は，運用コマンドや API によるコネクション解放を受け付けられません。

2.1.5　コネクションの解放

TP1/NET/OSAS-NIF は，コネクションを解放してからシステム間通信を終了します。

コネクションの解放には，正常解放と強制解放があります。

(1)　正常解放

TP1/NET/OSAS-NIF は，次に示す場合，コネクションを正常解放し，状態通知イベント（CCLSEVT）を通知します。

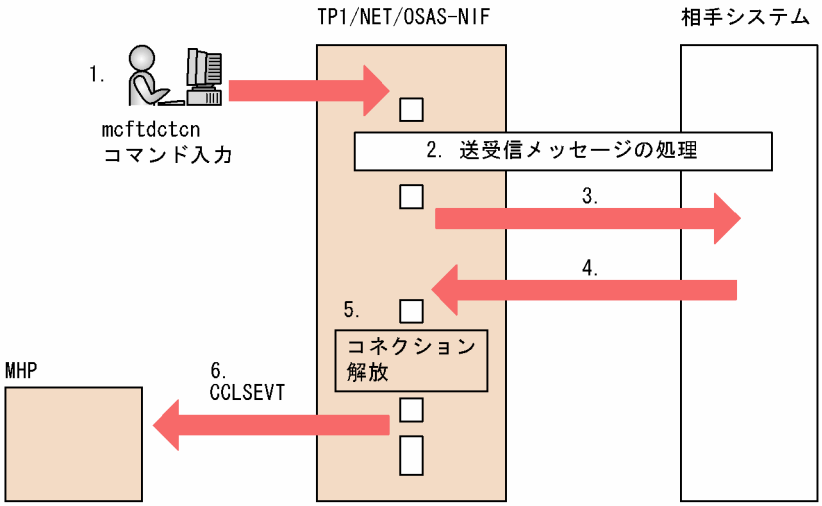
- 自システムから運用コマンド（mcftdctcn）を入力した場合
- 自システムから API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）を発行した場合
- 相手システムからの解放要求を受信した場合
- オンライン正常終了する場合※

注※

この場合は状態通知イベントを通知しません。

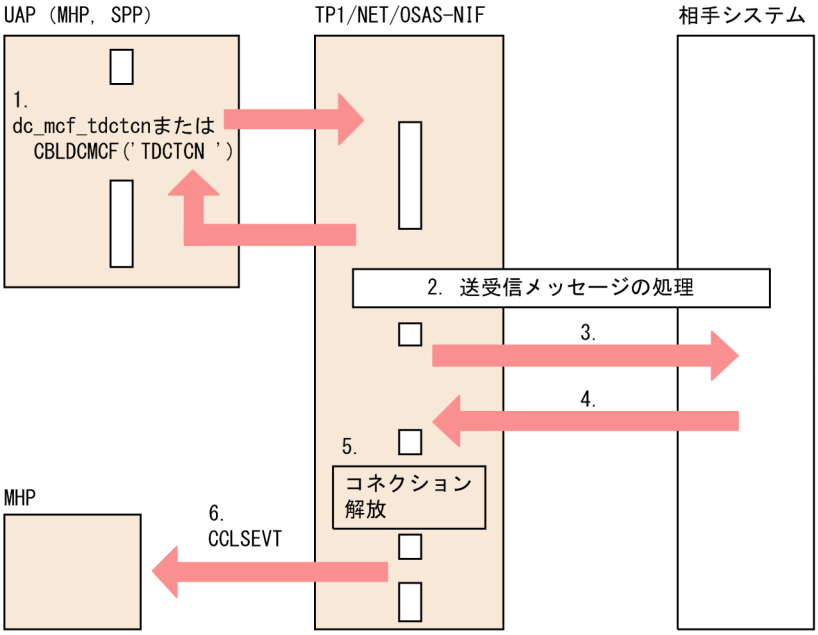
運用コマンドによるコネクションの正常解放を図 2-10 に，API によるコネクションの正常解放を図 2-11 に，相手システムからの解放要求によるコネクションの正常解放を図 2-12 に，オンライン正常終了によるコネクションの正常解放を図 2-13 に示します。

図 2-10　運用コマンドによるコネクションの正常解放



1. mcftdctcn コマンド（-f オプションなし）を入力します。
2. TP1/NET/OSAS-NIF は，仕掛り中の送受信メッセージを処理します。メッセージの処理については，「(3)　コネクション解放時のメッセージの処理」を参照してください。
3. TP1/NET/OSAS-NIF は，NIF アソシエーション解放要求を送信します。
4. 相手システムから NIF アソシエーション解放の回答を受信します。
5. コネクションを解放します。
6. TP1/NET/OSAS-NIF は，コネクションが解放されると状態通知イベント（CCLSEVT）を通知します。アプリケーション定義に CCLSEVT が定義されていない場合は，MHP は起動しません。

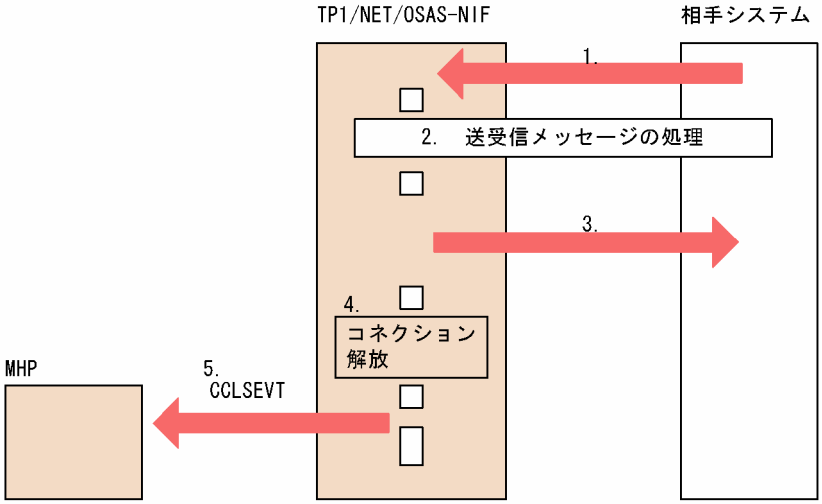
図 2-11　API によるコネクションの正常解放



1. API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）を発行します。

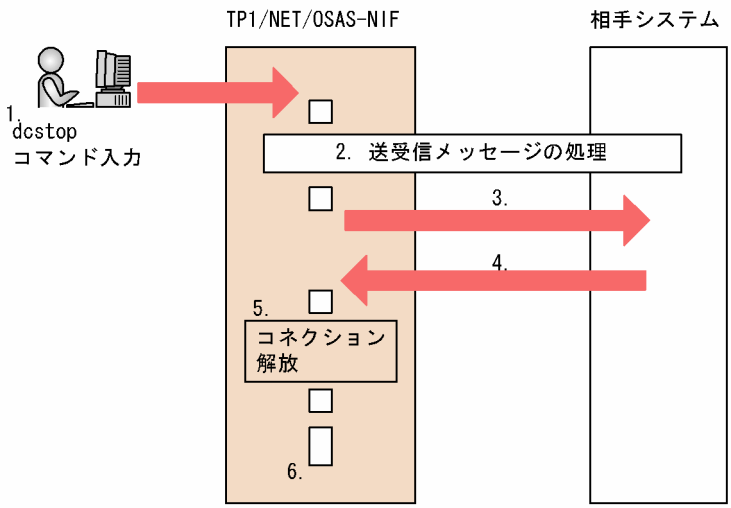
2. TP1/NET/OSAS-NIF は，仕掛り中の送受信メッセージを処理します。メッセージの処理については，「(3)　コネクション解放時のメッセージの処理」を参照してください。
3. TP1/NET/OSAS-NIF は，NIF アソシエーション解放要求を送信します。
4. 相手システムから NIF アソシエーション解放の回答を受信します。
5. コネクションを解放します。
6. TP1/NET/OSAS-NIF は，コネクションが解放されると状態通知イベント（CCLSEVT）を通知します。アプリケーション定義に CCLSEVT が定義されていない場合は，MHP は起動しません。

図 2-12　相手システムからの解放要求によるコネクションの正常解放



1. 相手システムから，NIF アソシエーション解放要求を受信します。
2. TP1/NET/OSAS-NIF は，仕掛り中の送受信メッセージを処理します。メッセージの処理については，「(3)　コネクション解放時のメッセージの処理」を参照してください。
3. TP1/NET/OSAS-NIF は，NIF アソシエーション解放回答を相手システムへ送信します。
4. コネクションを解放します。
5. TP1/NET/OSAS-NIF は，コネクションが解放されると状態通知イベント（CCLSEVT）を通知します。アプリケーション定義に CCLSEVT が定義されていない場合は，MHP は起動しません。

図 2-13　オンライン正常終了によるコネクションの正常解放



1. dcstop コマンドを入力します。
2. TP1/NET/OSAS-NIF は，仕掛り中の送受信メッセージを処理します。メッセージの処理については，「(3)　コネクション解放時のメッセージの処理」を参照してください。
3. TP1/NET/OSAS-NIF は，NIF アソシエーション解放要求を相手システムへ送信します。
4. 相手システムから，NIF アソシエーション解放の回答を受信します。
5. コネクションを解放します。
6. TP1/NET/OSAS-NIF は，終了します。

(2)　強制解放

TP1/NET/OSAS-NIF は，次に示す場合，コネクションを強制的に解放し，障害通知イベント（CERREVT）を通知します。

- 運用コマンド（mcftdctcn -f）を入力した場合
- 強制解放オプション※1 を指定した API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）を発行した場合
- 送受信バッファおよび編集バッファの資源が不足した場合
- 通信管理から回線障害を報告された場合
- 相手システムからの強制解放を受信した場合
- MCF 通信プロセスが終了した場合，または強制停止した場合※2

注※1

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

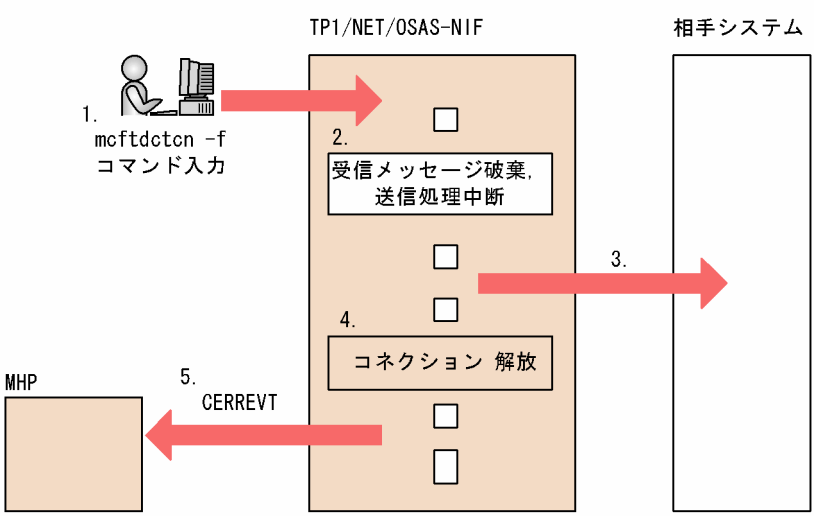
CBLDCMCF('TDCTCN△△')の場合，データ名 D1 に'1'を指定します。

注※2

この場合は状態通知イベントを通知しません。

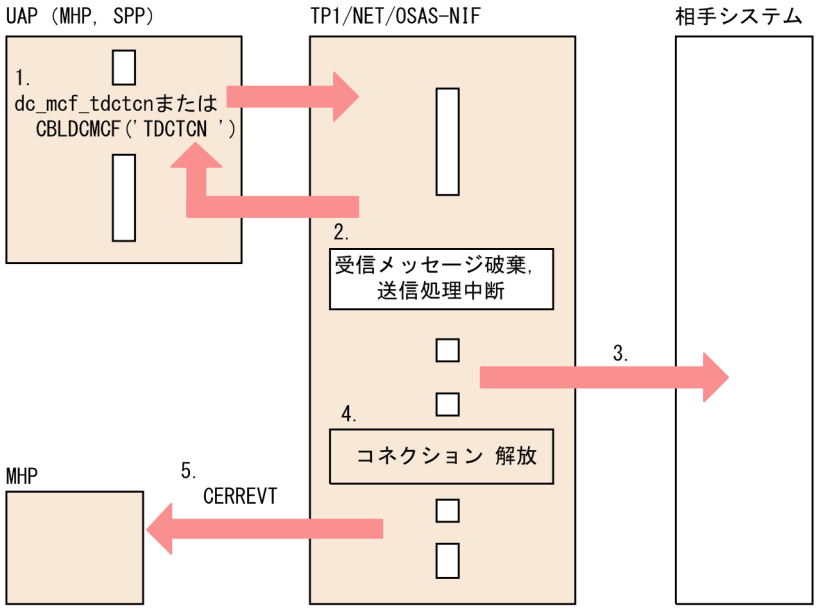
運用コマンドによるコネクションの強制解放を図 2-14 に，API によるコネクションの強制解放を図 2-15 に，相手システムからの強制解放によるコネクションの強制解放を図 2-16 に示します。

図 2-14　運用コマンドによるコネクションの強制解放



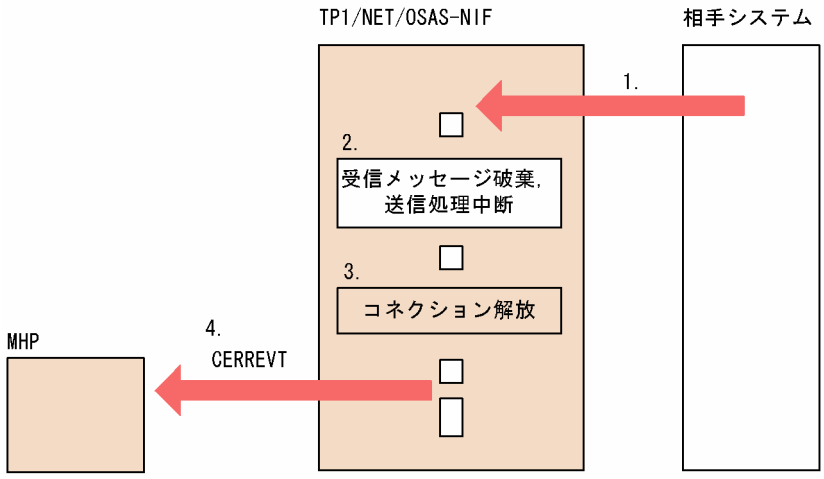
1. mcftdctcn -f コマンドを入力します。
2. 仕掛り中の受信メッセージを破棄し，送信処理を中断します。メッセージの処理については，「(3)　コネクション解放時のメッセージの処理」を参照してください。
3. TP1/NET/OSAS-NIF は，NIF アソシエーション緊急解放要求を送信します。
4. コネクションを解放します。
5. TP1/NET/OSAS-NIF は，コネクションが解放されると状態通知イベント（CERREVT）を通知します。アプリケーション定義に CERREVT が定義されていない場合は，MHP は起動しません。

図 2-15　API によるコネクションの強制解放



1. 強制解放オプションを指定した API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）を発行します。
2. 仕掛り中の受信メッセージを破棄し，送信処理を中断します。メッセージの処理については，「(3)　コネクション解放時のメッセージの処理」を参照してください。
3. TP1/NET/OSAS-NIF は，NIF アソシエーション緊急解放要求を送信します。
4. コネクションを解放します。
5. TP1/NET/OSAS-NIF は，コネクションが解放されると状態通知イベント（CERREVT）を通知します。アプリケーション定義に CERREVT が定義されていない場合は，MHP は起動しません。

図 2-16　相手システムからの強制解放によるコネクションの強制解放



1. 相手システムから，NIF アソシエーション緊急解放要求を受信します。
2. 仕掛り中の受信メッセージを破棄し，送信処理を中断します。メッセージの処理については，「(3)　コネクション解放時のメッセージの処理」を参照してください。

3. コネクションを解放します。
4. TP1/NET/OSAS-NIF は，コネクションが解放されると状態通知イベント（CERREVT）を通知します。アプリケーション定義に CERREVT が定義されていない場合は，MHP は起動しません。

(3) コネクション解放時のメッセージの処理

コネクションの正常解放時に送受信中のメッセージがある場合，TP1/NET/OSAS-NIF は，メッセージの送受信を完了してからコネクションを解放します。

コネクションの強制解放時に送受信中のメッセージがある場合，メッセージの扱いは，論理端末の端末タイプによって異なります。強制解放時の送受信中メッセージの扱いを次の表に示します。

表 2-3 コネクションの強制解放時の送受信中メッセージの扱い

端末タイプ	送信メッセージ			受信メッセージ	
	送信中断	メッセージの破棄	再確立時の再送※	メッセージの破棄	再確立時の再受信
request 型	○	×	○	○	△
request 型（同期型）	○	○	×	○	×
reply 型	○	×	△	○	△
send 型	○	×	○	−	−
receive 型	−	−	−	○	△

（凡例）

○：処理をします。

×：処理をしません。

△：相手システムに依存します。

−：該当しません。

注※

再送の詳細については，「2.5 再送機能」を参照してください。

(4) コネクションを解放するときの注意事項

コネクションの状態が解放処理中の場合は，オンラインを終了することができません。次のどれかでコネクションを強制解放してから，オンラインを終了してください。

- 運用コマンド（mcftdctcn -f）を入力する。
- 強制解放オプションを指定した API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）を発行する。

コネクションの状態を確認するときは，次のどれかを行ってください。

- 運用コマンド（mcftlscn）を入力する。
- API（dc_mcf_tlscn 関数または CBLDCMCF('TLSCN△△△')）を発行する。

2.1.6　論理端末の閉塞と閉塞解除

TP1/NET/OSAS-NIF は，相手システムとのコネクションを確立するとき，論理端末の閉塞を解除してメッセージを送受信します。

(1)　論理端末の閉塞

TP1/NET/OSAS-NIF は，次に示すような障害要因を検知すると，該当する論理端末を閉塞します。論理端末を閉塞すると，障害通知イベント（CERREVT）を通知し，対応する MHP を起動します。

(a)　運用コマンド（mcftdctle）による閉塞

TP1/NET/OSAS-NIF は，運用コマンド（mcftdctle）を入力したとき，次に示す処理をします。

メッセージ送信中の場合

request 型論理端末および send 型論理端末は，メッセージ送信を中断し，相手システムへ処理中断連絡（送信中断）を送信したあと，論理端末を閉塞します。

request 型論理端末（同期型）は，送信メッセージを破棄し，相手システムへ送信中断連絡（送信中断）を送信したあと，論理端末を閉塞します。

reply 型論理端末は，メッセージの送信が完了した時点で，論理端末を閉塞します。相手システムには，次にメッセージを受信したとき，処理中断連絡（受信拒否）を送信します。

メッセージ受信中の場合

request 型論理端末および request 型論理端末（同期型）はメッセージの受信が完了した時点で，論理端末を閉塞します。

reply 型論理端末および receive 型論理端末は受信メッセージを破棄し，相手システムへ処理中断連絡（受信拒否）を送信したあと，論理端末を閉塞します。

ただし，送受信処理中のメッセージがない場合は，処理中断連絡を送信しません。また，閉塞している論理端末に対して，新たにメッセージを受信したときは処理中断連絡（受信拒否）を送信します。

(b)　API（dc_mcf_tdctle 関数または CBLDCMCF('TDCTLE△△')）による閉塞

TP1/NET/OSAS-NIF は，API（dc_mcf_tdctle 関数または CBLDCMCF('TDCTLE△△')）を発行したとき，次に示す処理をします。

メッセージ送信中の場合

request 型論理端末および send 型論理端末は，メッセージ送信を中断し，相手システムへ処理中断連絡（送信中断）を送信したあと，論理端末を閉塞します。

request 型論理端末（同期型）は，送信メッセージを破棄し，相手システムへ送信中断連絡（送信中断）を送信したあと，論理端末を閉塞します。

reply 型論理端末は，メッセージの送信が完了した時点で，論理端末を閉塞します。相手システムには，次にメッセージを受信したとき，処理中断連絡（受信拒否）を送信します。

メッセージ受信中の場合

request 型論理端末および request 型論理端末（同期型）はメッセージの受信が完了した時点で，論理端末を閉塞します。

reply 型論理端末および receive 型論理端末は受信メッセージを破棄し，相手システムへ処理中断連絡（受信拒否）を送信したあと，論理端末を閉塞します。

ただし，送受信処理中のメッセージがない場合は，処理中断連絡を送信しません。また，閉塞している論理端末に対して，新たにメッセージを受信したときは処理中断連絡（受信拒否）を送信します。

(c) 相手システムからの強制閉塞

TP1/NET/OSAS-NIF は，相手システムから処理中断連絡（送信中断または受信拒否）を受信すると，次に示す処理をします。

メッセージ送信中の場合

request 型論理端末および send 型論理端末は，メッセージ送信を中断し，論理端末を閉塞します。

request 型論理端末（同期型）および reply 型論理端末は，送信中のメッセージを破棄し，論理端末を閉塞します。

メッセージ受信中の場合

request 型論理端末，request 型論理端末（同期型），reply 型論理端末および receive 型論理端末は受信メッセージを破棄し，論理端末を閉塞します。

(d) コネクション解放による閉塞

TP1/NET/OSAS-NIF は，コネクションを解放すると，論理端末を閉塞します。コネクション解放時は論理端末の障害通知イベント（CERREVT）は起動しません。

このときの送受信メッセージの扱いについては，「2.1.5(3) コネクション解放時のメッセージの処理」を参照してください。

(e) NIF-REJECT 送信または受信による強制閉塞

TP1/NET/OSAS-NIF は，相手システムへ NIF-REJECT を送信するか，または相手システムから NIF-REJECT を受信すると，次に示す処理をします。

メッセージ送信中の場合

request 型論理端末，reply 型論理端末および send 型論理端末は，メッセージ送信を中断し，論理端末を閉塞します。

request 型論理端末（同期型）は，送信メッセージを破棄し，論理端末を閉塞します。

メッセージ受信中の場合

request 型論理端末，request 型論理端末（同期型），reply 型論理端末および receive 型論理端末は，受信メッセージを破棄し，論理端末を閉塞します。

(f)　メッセージ送受信処理障害による閉塞

TP1/NET/OSAS-NIF は，メッセージ入力障害などによる障害が発生した場合，論理端末を閉塞します。詳細については，「9.1　障害の種類と対応処理」を参照してください。

(2)　論理端末の閉塞解除

TP1/NET/OSAS-NIF は，コネクション確立中に，論理端末を閉塞した場合，次のどれかで閉塞していた論理端末の閉塞を解除します。

- 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。
- 相手システムの障害復旧を認識する。
- コネクションの再確立をする。

なお，コネクション確立時に，論理端末の突き合わせで使用できなくなった論理端末の閉塞は解除できません。

(a)　運用コマンド（mcftactle）による閉塞解除

TP1/NET/OSAS-NIF は，運用コマンド（mcftactle）を入力したとき，次に示す処理をします。

相手システムから処理中断連絡（受信拒否）を受信して閉塞した場合

request 型論理端末，request 型論理端末（同期型），reply 型論理端末および send 型論理端末は，運用コマンド（mcftactle）を受け付けません。

相手システムから処理中断連絡（送信中断）を受信して閉塞した場合

request 型論理端末および request 型論理端末（同期型）は論理端末を閉塞解除します。

reply 型論理端末および receive 型論理端末は運用コマンド（mcftactle）を受け付けません。

相手システムへ処理中断連絡（受信拒否）を送信して閉塞した場合

reply 型論理端末および receive 型論理端末は，相手システムへ受信拒否解除を送信して，論理端末を閉塞解除します。

相手システムへ処理中断連絡（送信中断）を送信して閉塞した場合

request 型論理端末および send 型論理端末は，論理端末を閉塞解除します。中断していたメッセージ送信を再処理します。

request 型論理端末（同期型）および reply 型論理端末は，論理端末を閉塞解除します。

NIF-REJECT 送信または受信によって閉塞した場合

request 型論理端末，request 型論理端末（同期型）および send 型論理端末は，論理端末を閉塞解除します。

reply 型論理端末および receive 型論理端末は，運用コマンド（mcftactle）を受け付けません。

上記以外の場合

論理端末を閉塞解除します。

(b) API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）による閉塞解除

TP1/NET/OSAS-NIF は，API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行したとき，次に示す処理をします。

相手システムから処理中断連絡（受信拒否）を受信して閉塞した場合

request 型論理端末，request 型論理端末（同期型），reply 型論理端末および send 型論理端末は，API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を受け付けません。

相手システムから処理中断連絡（送信中断）を受信して閉塞した場合

request 型論理端末および request 型論理端末（同期型）は論理端末を閉塞解除します。

reply 型論理端末および receive 型論理端末は API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を受け付けません。

相手システムへ処理中断連絡（受信拒否）を送信して閉塞した場合

reply 型論理端末および receive 型論理端末は，相手システムへ受信拒否解除を送信して，論理端末を閉塞解除します。

相手システムへ処理中断連絡（送信中断）を送信して閉塞した場合

request 型論理端末および send 型論理端末は，論理端末を閉塞解除します。中断していたメッセージ送信を再処理します。

request 型論理端末（同期型）および reply 型論理端末は，論理端末を閉塞解除します。

NIF-REJECT 送信または受信によって閉塞した場合

request 型論理端末，request 型論理端末（同期型）および send 型論理端末は，論理端末を閉塞解除します。

reply 型論理端末および receive 型論理端末は，API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を受け付けません。

上記以外の場合

論理端末を閉塞解除します。

(c) 相手システムの障害復旧による閉塞解除

TP1/NET/OSAS-NIF は，相手システムから処理中断連絡（受信拒否/送信中断）を受信して閉塞した場合，または NIF-REJECT を送受信して閉塞した場合，相手システムの障害復旧によって，自動的に論理端末の閉塞を解除します。

相手システムから処理中断連絡（受信拒否）を受信して閉塞した場合

request 型論理端末および send 型論理端末は，相手システムから受信拒否解除を受信することによって，相手システムの障害復旧を認識し，論理端末を閉塞解除します。中断していたメッセージ送信を再処理します。

request 型論理端末（同期型）および reply 型論理端末は，論理端末を閉塞解除します。

相手システムから処理中断連絡（送信中断）を受信して閉塞した場合

reply 型論理端末および receive 型論理端末は，相手システムからメッセージを受信することによって，相手システムの障害復旧を認識し，論理端末を閉塞解除します。

NIF-REJECT 送信または受信によって閉塞した場合

reply 型論理端末および receive 型論理端末は，相手システムからメッセージを受信することによって，相手システムの障害復旧を認識し，論理端末を閉塞解除します。

2.1.7　着呼型コネクションの自動受付開始

着呼型コネクションの自動受付開始は，着呼型のコネクションを解放したあとに相手システムからのコネクション確立要求を自動的に受け付けできるようにする機能です。この機能を使用すると，コネクションを解放したあとの運用コマンド（mcftactcn）の入力，および API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）の発行が不要となります。着呼型コネクションの自動受付開始を使用するかどうかは，プロトコル共通定義（mcftpcmn -r）の autoonln オペランドで指定します。

なお，この機能を使用する場合，確立処理中の運用コマンド（mcftdctcn）および API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）の動作に差異があります。確立処理中の運用コマンドおよび API の動作差異を次の表に示します。

表 2-4　確立処理中の運用コマンドおよび API の動作差異

autoonln オペランドの指定内容	運用コマンド（mcftdctcn）の動作		API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）の動作	
	正常解放	強制解放	正常解放	強制解放
no	運用コマンドは無効です。	運用コマンドを正常に受け付けます。 相手システムからのコネクション確立要求は受け付けられなくなります。	API はエラーリターンします。 （リターン値 DCMCFRTN_71010 またはステータスコード 71010）	API は正常リターンします。 相手システムからのコネクション確立要求は受け付けられなくなります。
yes	運用コマンドは無効です。 （KFCA13530-E の理由コード：CN_PASI）		API はエラーリターンします。 （リターン値 DCMCFRTN_71010 またはステータスコード 71010）	

注意事項

着呼型コネクションの自動受付開始を使用する場合，次の点に注意してください。

- プロトコル共通定義（mcftpcmn -e）の termactb オペランドに cont を指定する必要があります。wait を指定，または省略した場合，定義オブジェクト生成時にエラーとなります。
- 運用コマンド（mcftactcn）の入力および API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）の発行は，次の条件を満たしている場合に行えます。
  ・コネクション定義（mcftalccn）の-i オプションに manual を指定，または省略している。

・オンライン開始後，運用コマンド（mcftactcn）の入力や，API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）の発行を行っていない。

システム移行などによって既存の UAP を再利用する場合，MCF イベント（CCLSEVT や CERREVT など）を処理する UAP で，自動的に運用コマンド（mcftactcn）を入力したり，API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）を発行したりする処理がないか確認してください。問題がある場合，処理を削除するなどの対策を行ってください。

- 着呼型コネクションの自動受付開始を使用するコネクションと使用しないコネクションが混在する場合，使用するコネクションと使用しないコネクションで MCF 通信プロセスを分割してください。

  このとき，MCF イベントを処理する UAP を各々の MCF 通信プロセスで共用する場合，コネクション解放後に UAP からコネクション確立要求をするかどうかは，MCF が通知したコネクション ID を基に判断してください。

2.1.8　過着呼による障害検出

着呼型のコネクションでは，相手システム，または，自システムと相手システム間のネットワークの障害を長時間検出できないことがあります。このとき，TP1/NET/OSAS-NIF は，障害を検出できていないコネクションを確立状態と判断します。

確立状態の着呼型のコネクションに対して相手システムから新たにコネクション確立要求を受け付けることを**過着呼**と呼びます。過着呼による障害検出は，過着呼をきっかけとして着呼型のコネクションの障害を検出する機能です。

過着呼による障害検出を使用しない場合と使用する場合に分け，過着呼を検出したときの処理について説明します。

(1)　過着呼による障害検出を使用しない場合の処理

過着呼が発生した場合，TP1/NET/OSAS-NIF は新たに受け付けたコネクション確立要求を拒否します。

確立状態にある既存のコネクションを解放するか自システムで障害を検出するまで，コネクションは再確立できません。

自システムで無通信状態を監視し，必要に応じてコネクションを強制解放するなどの処置をしてください。

(2)　過着呼による障害検出を使用する場合の処理

過着呼が発生した場合，TP1/NET/OSAS-NIF は確立状態にある既存のコネクションを強制解放します。

その後の TP1/NET/OSAS-NIF の動作は，着呼型コネクションの自動受付開始を使用するかどうかで異なります。

過着呼が発生した場合の TP1/NET/OSAS-NIF の動作を次の表に示します。

表 2-5　過着呼が発生した場合の TP1/NET/OSAS-NIF の動作

着呼型コネクションの自動受付開始	TP1/NET/OSAS-NIF の動作
使用しない	1. 強制解放を通知するメッセージログ（KFCA13588-I）を出力します。 2. 確立状態にある既存のコネクションを強制解放します。 3. 確立要求拒否を通知するメッセージログ（KFCA13590-W）を出力します。 4. 相手システムへ確立拒否を送信します。 5. アソシエーション層障害を通知するメッセージログ（KFCA13586-E）を出力します。 6. NIF アソシエーション解放を通知するメッセージログ（KFCA13552-I）を出力します。 7. コネクション障害を通知するメッセージログ（KFCA13502-E）を出力します。 8. CERREVT を起動します。
使用する	1. 強制解放を通知するメッセージログ（KFCA13588-I）を出力します。 2. 確立状態にある既存のコネクションを強制解放します。 3. リプレース※を通知するメッセージログ（KFCA13590-W）を出力します。 4. NIF アソシエーション解放を通知するメッセージログ（KFCA13552-I）を出力します。 5. コネクション障害を通知するメッセージログ（KFCA13502-E）を出力します。 6. CERREVT を起動します。 7. 相手システムへ初期設定回答を送信します。 8. NIF アソシエーション確立を通知するメッセージログ（KFCA13550-I）を出力します。 9. COPNEVT を起動します。

注※

確立状態にあるコネクションを強制解放し，新たに受け付けたコネクション確立要求に切り替えることを**リプレース**と呼びます。

リプレース中に相手システムから再度コネクション確立要求を受け付けた場合，リプレース中のコネクションを強制解放し，再度受け付けたコネクション確立要求を受け付けます。

また，リプレース中に新しいコネクション確立要求の障害を検出した場合，リプレース中のコネクションを強制解放し，相手システムからのコネクション確立要求を待ちます。

(3)　注意事項

過着呼による障害検出を使用する場合，次の点に注意してください。

- 着呼型コネクションの自動受付開始を使用しない場合，過着呼の検出と同時にコネクションを確立することはできません。相手システム側はコネクションの確立拒否を受信したとしても一定回数コネクション確立要求を再試行する運用を行ってください。
- 過着呼による障害検出を使用し，着呼型コネクションの自動受付開始を使用しない構成は，既存の UAP を再利用する環境での使用を想定しています。新規にシステムを構築する場合は，過着呼による障害検出と着呼型コネクションの自動受付開始を併用することをお勧めします。

2.2　システム間通信メッセージの送受信

TP1/NET/OSAS-NIF は，システム間通信メッセージの送受信をします。TP1/NET/OSAS-NIF で使用するメッセージには次の種類があります。

- 問い合わせメッセージ
- 応答メッセージ
- 一方送信メッセージ

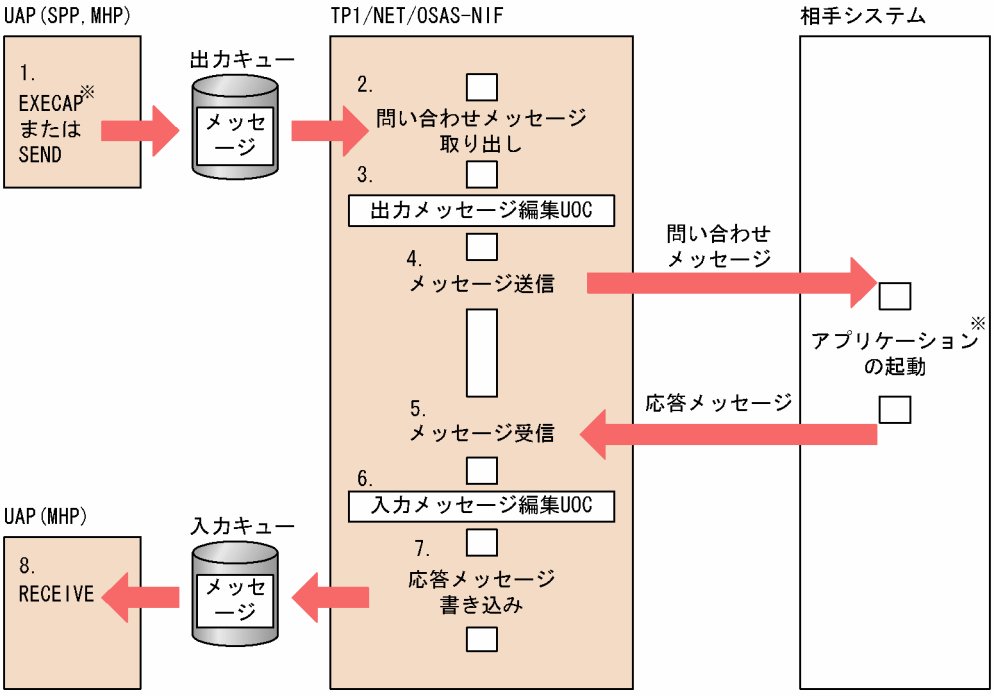
この節では，TP1/NET/OSAS-NIF のメッセージの送信と受信の流れについて説明します。

2.2.1　問い合わせメッセージの送信

TP1/NET/OSAS-NIF が，相手システムへ問い合わせメッセージを送信し，相手システムからの応答メッセージを受信した場合の処理の流れを説明します。

問い合わせメッセージの送信と応答メッセージの受信の流れを次の図に示します。

図 2-17　問い合わせメッセージの送信と応答メッセージの受信の流れ



注※

EXECAP要求の場合，相手システムのアプリケーションを起動できます。

1. SPP または MHP から，request 型論理端末あてに問い合わせメッセージを送信します。

   EXECAP 要求を呼び出した場合，request 型論理端末は使用状態となり，同じ論理端末あてに EXECAP 要求を発行できません。

2. TP1/NET/OSAS-NIF は出力キューに書き込まれたメッセージを取り出します。

3. 出力メッセージ編集 UOC でメッセージを編集します。

4. 相手システムへ問い合わせメッセージを送信します。

5. 相手システムから応答メッセージを受信します。

6. 入力メッセージ編集 UOC でメッセージを編集します。

7. TP1/NET/OSAS-NIF は入力キューへ応答メッセージを書き込みます。

   1.で EXECAP 要求を呼び出した場合，1.で使用状態とした request 型論理端末の状態を解除します。同じ論理端末あてに EXECAP 要求を発行できます。

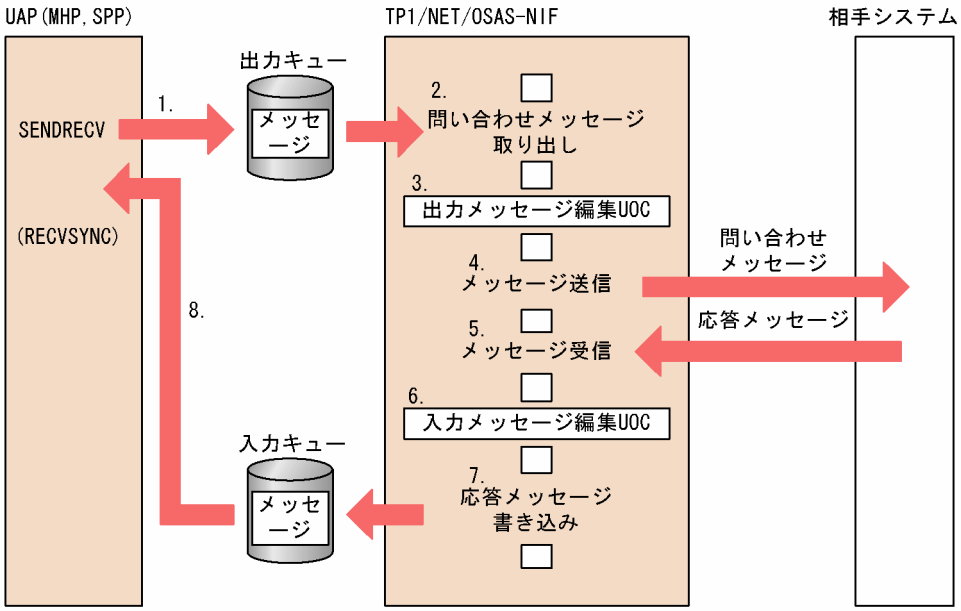
8. MHP を起動させます。MHP は RECEIVE 要求を発行してメッセージを受け取ります。

2.2.2 同期型の問い合わせメッセージの送信

TP1/NET/OSAS-NIF が，相手システムへ同期型の問い合わせメッセージを送信し，相手システムからの応答メッセージを受信した場合の処理の流れを説明します。

同期型の問い合わせメッセージの送信と応答メッセージの受信の流れを次の図に示します。

図 2-18 同期型の問い合わせメッセージの送信と応答メッセージの受信の流れ



1. MHP または SPP から，request 型論理端末（同期型）あてに問い合わせメッセージを送信します。

   request 型論理端末（同期型）は使用状態となり，同じ論理端末あてに SENDRECV 要求を発行できません。

2. TP1/NET/OSAS-NIF は出力キューに書き込まれたメッセージを取り出します。

3. 出力メッセージ編集 UOC でメッセージを編集します。

4. 相手システムへ問い合わせメッセージを送信します。

5. 相手システムから応答メッセージを受信します。

6. 入力メッセージ編集 UOC でメッセージを編集します。

7. TP1/NET/OSAS-NIF は入力キューへ応答メッセージを書き込みます。

   1.で使用状態とした request 型論理端末（同期型）の状態を解除し，同じ論理端末あてに SENDRECV 要求を発行可能とします。

8. 問い合わせメッセージを送信した UAP に制御が渡され，応答メッセージを受け取ります。

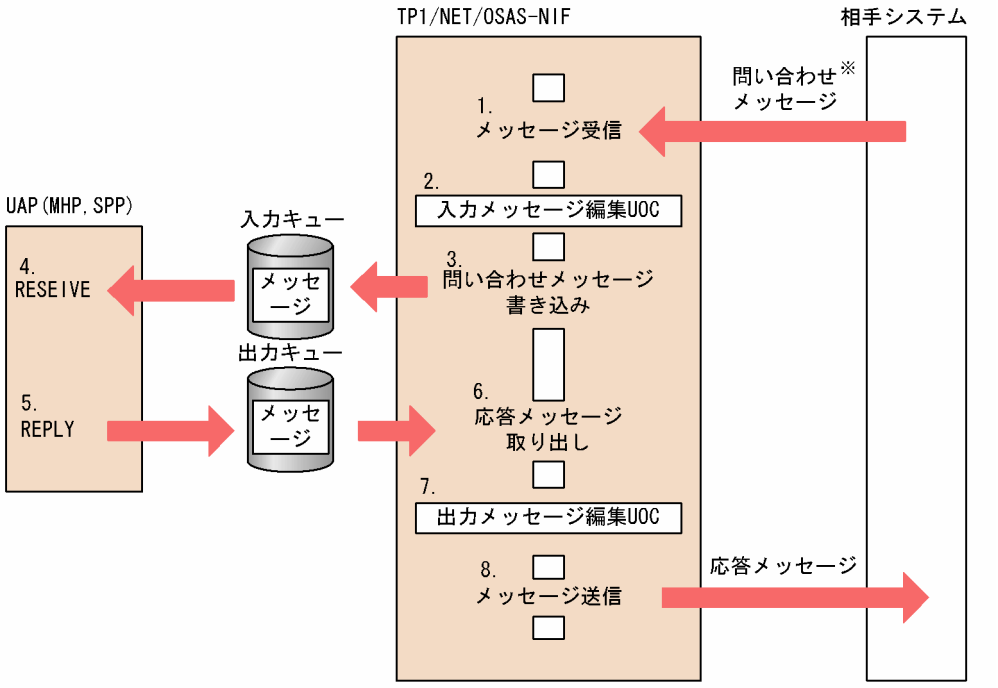
   応答メッセージが複数のセグメントの場合，RECVSYNC 要求を発行して，後続セグメントを受信します。

2.2.3 応答メッセージの送信

TP1/NET/OSAS-NIF が，相手システムからの問い合わせメッセージを受信し，相手システムへ応答メッセージを送信する場合の処理の流れを説明します。

問い合わせメッセージの受信と応答メッセージの送信の流れを次の図に示します。

図 2-19 問い合わせメッセージの受信と応答メッセージの送信の流れ



注※

相手システムからのEXECAP要求を受信できます。その場合は，入力メッセージ編集UOCがなくても自システムのアプリケーションを自動起動できます。

1. 相手システムからの問い合わせメッセージを reply 型論理端末で受信します。

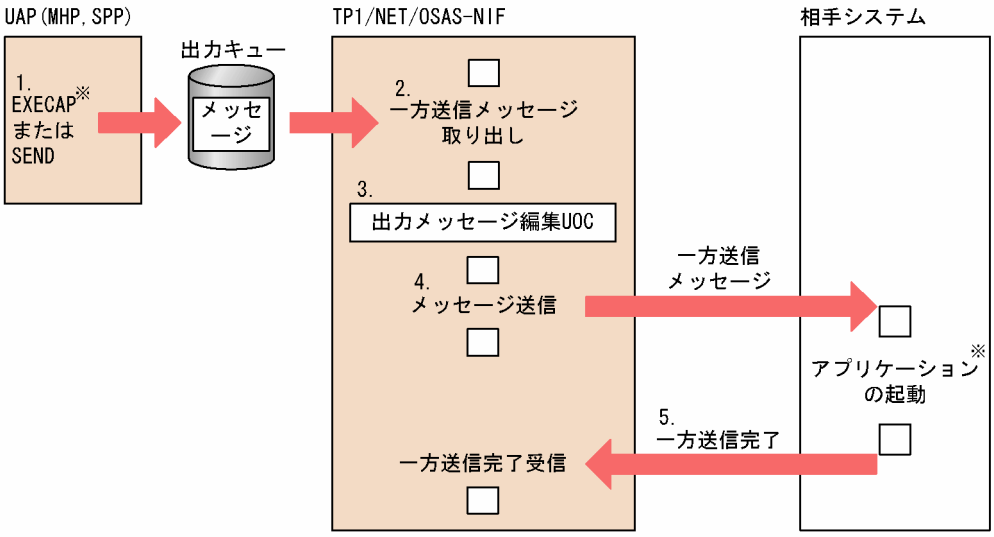
2. 入力メッセージ編集 UOC でメッセージを編集します。相手システムからの EXECAP 要求を受信した場合に，自システムのアプリケーションが指定されているときは，入力メッセージ編集 UOC は不要です。
3. TP1/NET/OSAS-NIF は入力キューへ問い合わせメッセージを書き込みます。
4. MHP を起動させます。MHP は RECEIVE 要求を発行してメッセージを受け取ります。ただし，相手システムからの EXECAP 要求を受信した場合は，EXECAP 要求の中で指定されたアプリケーションが起動されます。
5. MHP は，REPLY 要求を発行して出力キューへ応答メッセージを書き込みます。
6. TP1/NET/OSAS-NIF は，出力キューに書き込まれた応答メッセージを取り出します。
7. 出力メッセージ編集 UOC でメッセージを編集します。
8. 相手システムへ応答メッセージを送信します。

2.2.4 一方送信メッセージの送信

TP1/NET/OSAS-NIF が，相手システムへ一方送信メッセージを送信した場合の処理の流れを説明します。

一方送信メッセージ送信の処理の流れを次の図に示します。

図 2-20 一方送信メッセージ送信の処理の流れ



注※
EXECAP要求の場合，相手システムのアプリケーションを起動できます。

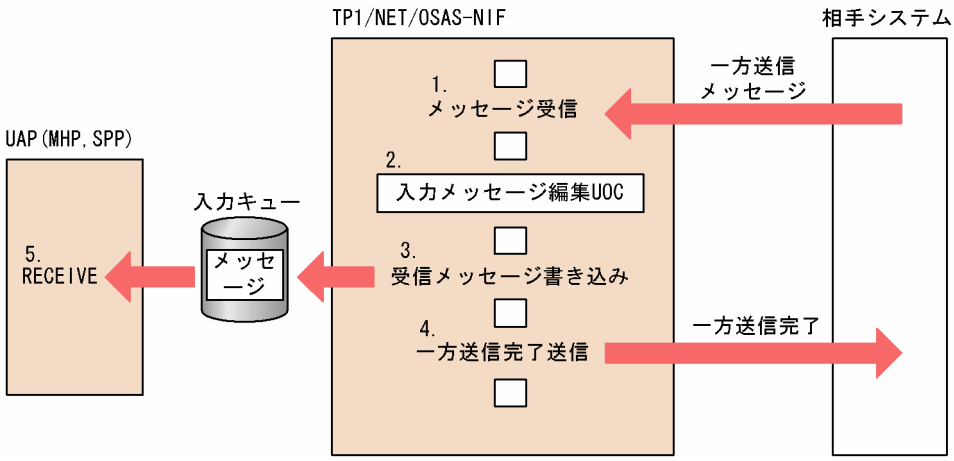
1. MHP または SPP から，send 型論理端末あてに一方送信メッセージを送信します。
2. TP1/NET/OSAS-NIF は出力キューに書き込まれたメッセージを取り出します。
3. 出力メッセージ編集 UOC でメッセージを編集します。
4. 相手システムへメッセージを送信します。
5. 相手システムからの一方送信完了を受信します。

2.2.5　一方送信メッセージの受信

TP1/NET/OSAS-NIF が，相手システムからの一方送信メッセージを受信した場合の処理の流れを説明します。

一方送信メッセージ受信の処理の流れを次の図に示します。

図 2-21　一方送信メッセージ受信の処理の流れ



1. 相手システムから receive 型論理端末あての一方送信メッセージを受信します。
2. 入力メッセージ編集 UOC でメッセージを編集します。
3. 受信したメッセージを入力キューに書き込みます。
4. 相手システムに一方送信完了を送信します。
5. MHP を起動させます。MHP は RECEIVE 要求を発行してメッセージを受け取ります。

2.3 アプリケーションプログラムの起動

TP1/NET/OSAS-NIF では，**アプリケーション起動機能**を使用できます。ここでは，アプリケーション起動機能を使用する場合に必要な準備と，使用方法について説明します。なお，アプリケーション起動機能の詳細については，マニュアル「OpenTP1 プログラム作成の手引」のアプリケーションプログラムの起動について説明している個所を参照してください。

2.3.1 アプリケーション起動機能を使用する場合に必要な MCF 通信プロセス

アプリケーション起動機能を使用する場合，相手システムとの通信を行うプロセスとは別の MCF のプロセスを使用します。TP1/NET/OSAS-NIF では，メッセージ送受信で使用する MCF のプロセスを**MCF 通信プロセス**，dc_mcf_execap 関数で使用する MCF のプロセスを**アプリケーション起動プロセス**といいます。

アプリケーション起動プロセスはアプリケーション起動環境定義に指定します。アプリケーション起動機能を使用する場合は，MCF 通信構成定義のアプリケーション起動環境定義を作成しておいてください。

なお，アプリケーション起動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。

2.3.2 アプリケーション起動機能の対象

アプリケーション起動機能の対象となるのは，自システムおよび相手システムです。

相手システムのアプリケーションを起動する場合には，次に示す 3 種類があります。

- SPP からのシステム間通信によってアプリケーションを起動する場合
- MHP からのシステム間通信によってアプリケーションを起動する場合
- システム間通信と自システム内のアプリケーション起動を併用する場合

それぞれの場合の定義例については，「アプリケーション起動機能を使用する場合に関連づける内容」を参照してください。

2.3.3 アプリケーション起動機能の使用方法

アプリケーション起動機能を使用する場合は，アプリケーションプログラムを起動する関数（dc_mcf_execap または CBLDCMCF('EXECAP ')）に，起動させたい MHP または SPP のアプリケーション名，および引き渡すメッセージのセグメントを指定します。

dc_mcf_execap 関数についての詳細は「dc_mcf_execap － アプリケーションプログラムの起動（C 言語）」を，CBLDCMCF('EXECAP ')関数についての詳細は「CBLDCMCF('EXECAP ') － アプリケーションプログラムの起動（COBOL 言語）」をそれぞれ参照してください。

2.4　フロー制御

フロー制御とは，メッセージ送信の信頼性を高めるため，メッセージを送信するときに，セグメント単位に通信相手への送信完了を確認しながら連続送信をする機能です。

TP1/NET/OSAS-NIF がメッセージを送信する場合は，必ずフロー制御をします。メッセージを受信する場合は，相手システムによって，フロー制御をする場合としない場合があります。フロー制御を次の図に示します。

図 2-22　フロー制御（問い合わせメッセージの送信と応答メッセージの受信）

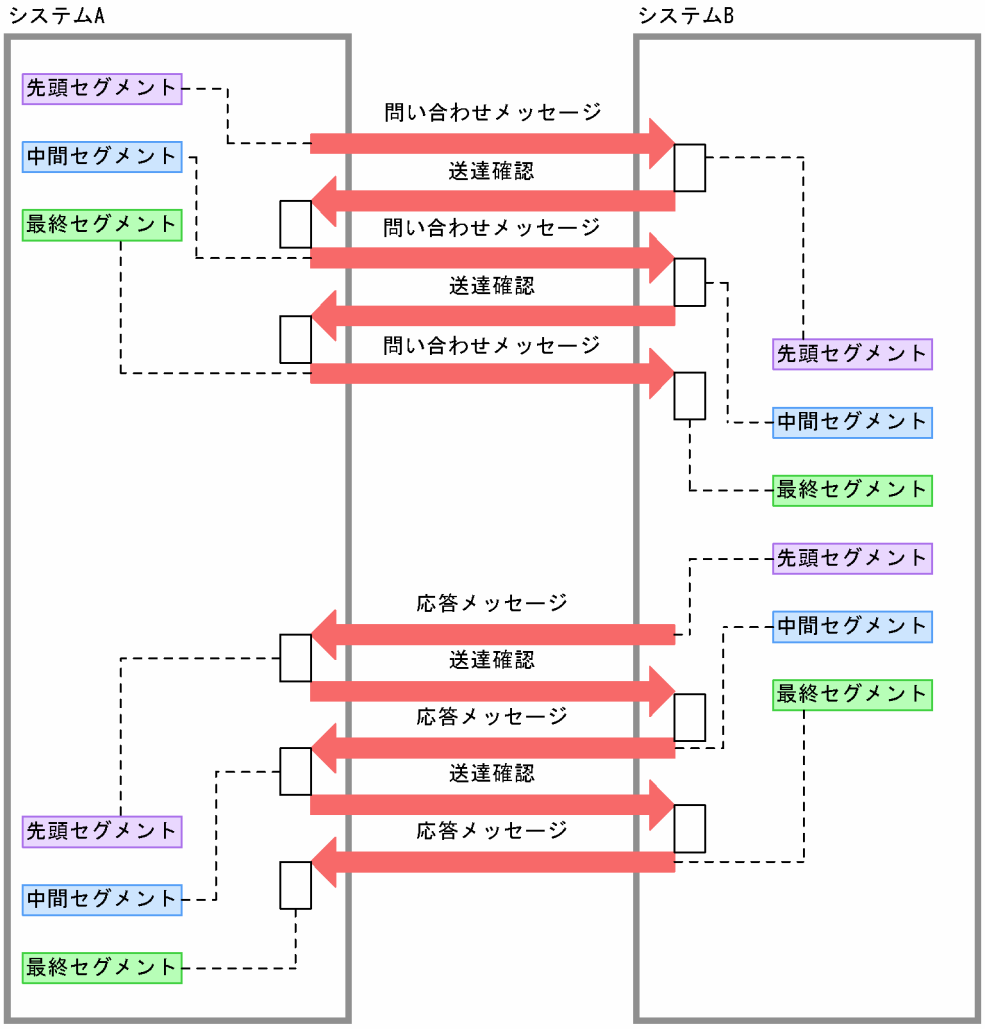
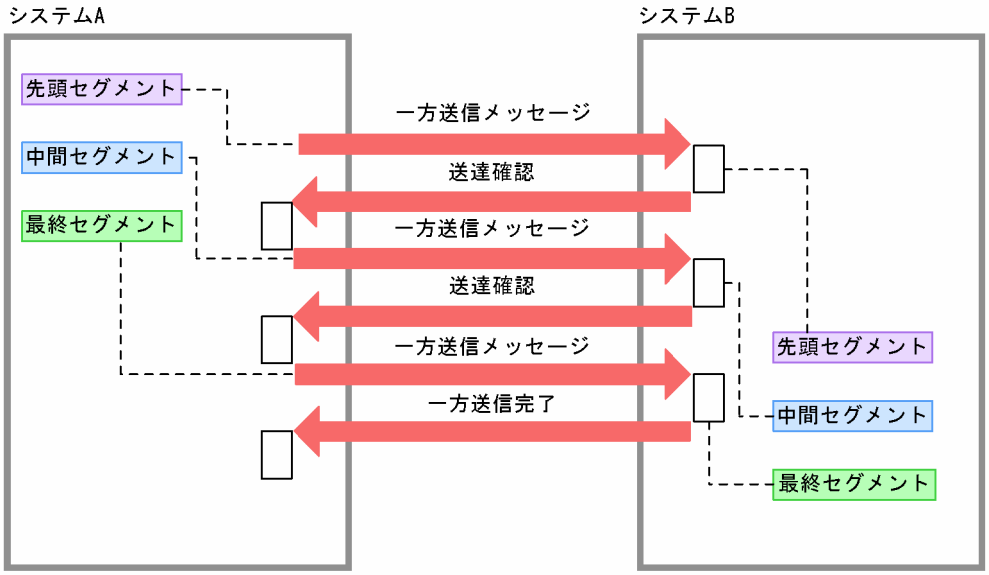


図 2-23　フロー制御（一方送信メッセージの送信と受信）



2.5　再送機能

TP1/NET/OSAS-NIF は，メッセージ送受信中に自システム内に障害が発生した場合，またはオンラインシステムが停止した場合，送信中だったメッセージを再度送信します。メッセージの再送のために，TP1/NET/OSAS-NIF は，メッセージに一連の通番を付けています。

2.5.1　メッセージ再送機能

仕掛り中メッセージの再送は，中断要因，キュー種別および論理端末の端末タイプによって異なります。

障害回復後の仕掛り中メッセージの再送を次の表に示します。

表 2-6　仕掛り中メッセージの再送

端末タイプ	中断要因								
	コネクション障害，オンライン停止，タイムアウト，NIF-REJECT 送信/受信		受信拒否受信		論理端末障害				左記以外
					mcftdctle 入力		バッファ不足		
	キュー種別		キュー種別		キュー種別		キュー種別		
	ディスク	メモリ	ディスク	メモリ	ディスク	メモリ	ディスク	メモリ	
request 型	○	×	○	×	○	×	○	×	×
request 型（同期型）	×	×	×	×	×	×	×	×	×
reply 型	△	×	×	×	―※1	―※1	△※2	×	×
send 型	○	×	○	×	○	×	○	×	×

（凡例）

○：再送します。

×：再送しません。

△：相手システムに依存します。

－：該当しません。

注※1

送信を完了してから論理端末を閉塞するため，再送は該当しません。

注※2

コネクションを解放するため，コネクション再確立後に再送されます。

ただし，次に示す場合は，メッセージの再送はしません。

- 送信メッセージが出力キューから消滅した場合
- 再送準備中に障害が発生した場合

次に，TP1/NET/OSAS-NIF による再送と，RESEND 要求による再送との違いを示します。

- TP1/NET/OSAS-NIF による再送：
  相手システムとのメッセージ送受信で障害が発生した場合に自動的に行われます。
- RESEND 要求による再送：
  相手システムの UAP がメッセージを受け取ったあとで，メッセージ損失が発生した場合に使用します。

2.5.2 NIF 通番

TP1/NET/OSAS-NIF は，UAP からのシステム間通信メッセージに一連の通番を付け，メッセージを管理し，再送機能に使用します。TP1/NET/OSAS-NIF がメッセージに付ける通番のことを，**NIF 通番**といいます。

TP1/NET/OSAS-NIF は，メッセージに NIF 通番を付け，メッセージの脱落や重複を防止したり，メッセージ送達の確認をしています。このため，相手システムでは，NIF/OSI プロトコルで定める通番機能，通番問い合わせ機能を，必ず使用してください。使用しない場合，コネクションを確立できません。

NIF 通番は，論理端末（エージェント）ごとに管理されています。

NIF 通番を参照することはできません。

また，TP1/NET/OSAS-NIF はオンライン再開始時に前回のオンライン停止時の NIF 通番を引き継ぐことができます。

オンライン再開始時に NIF 通番を引き継がない場合は，論理端末定義（mcftalcle -k）の quekind オペランドに memory を，論理端末定義（mcftalcle -d）の nugua オペランドに no を指定してください。

オンライン再開始時に NIF 通番を引き継がない指定をした場合，オンライン再開始時の NIF 通番はリセット状態となり，再送のための情報はなくなります。

2.6　タイマ監視

TP1/NET/OSAS-NIF は，相手システムと UAP とのメッセージ送受信でメッセージ送受信中の無応答を防止するため，タイマ監視をします。

2.6.1　相手システムとのメッセージ送受信に関するタイマ監視

TP1/NET/OSAS-NIF は，各種の送信メッセージに対する次のメッセージを受信するまでの時間を監視しています。

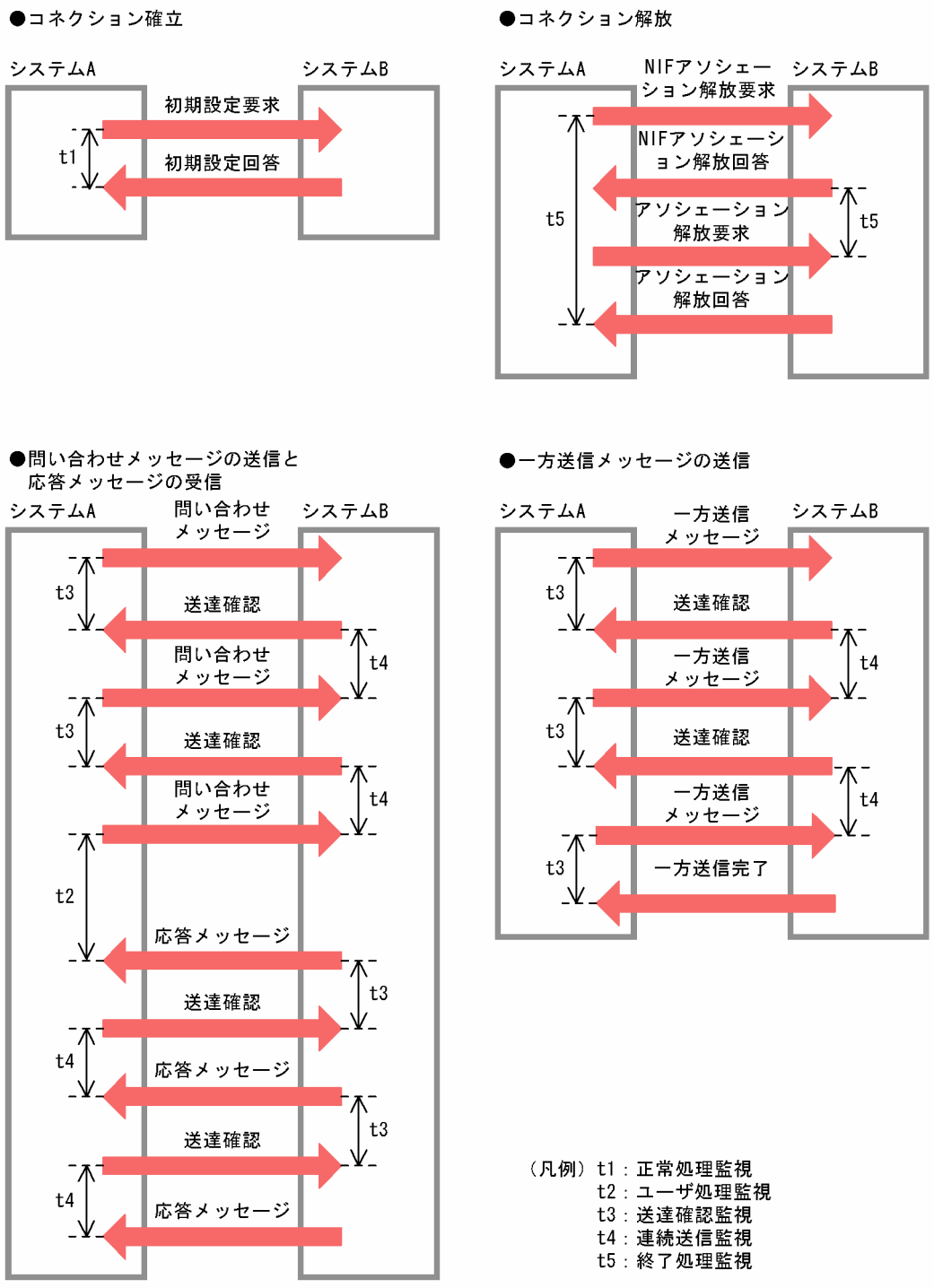
タイマ監視をするかどうか，および監視タイマ値は，コネクション定義（mcftalccn -v）で指定します。監視タイマ値に 0 を指定すると，時間監視はしません。

TP1/NET/OSAS-NIF の相手システムとのタイマ監視の種類と内容を次の表に示します。また，相手システムとのタイマ監視の範囲の例を図 2-24 に示します。

表 2-7　相手システムとのタイマ監視の種類と内容

タイマ監視の種別	内容	タイムアウト時の TP1/NET/OSAS-NIF の処理
正常処理監視（t1）	初期設定要求を送信してから，初期設定回答を受信するまでの時間の監視	コネクションの確立処理を中断します。
	論理端末の閉塞解除に関する各要求を送信してから，応答を受信するまでの時間の監視	論理端末の閉塞解除処理を中断します。
ユーザ処理監視（t2）	問い合わせメッセージの最終セグメントを送信してから応答メッセージの先頭セグメントを受信するまでの時間の監視	論理端末を閉塞します。 送信処理を中断します。
送達確認監視（t3）	一方送信メッセージの一つのセグメントを送信してから，送達確認または一方送信完了を受信するまでの時間の監視	論理端末を閉塞します。 送信処理を中断します。
	問い合わせメッセージまたは応答メッセージの，先頭セグメントまたは中間セグメントを送信してから送達確認を受信するまでの時間の監視	
連続送信監視（t4）	送達確認を送信してから，次のセグメントを受信するまでの時間の監視	論理端末を閉塞します。 受信メッセージを破棄します。
終了処理監視（t5）	NIF アソシエーション解放要求を送信してから，コネクションが解放するまでの時間の監視	コネクションを強制解放します。
	NIF アソシエーション解放回答を送信してから，アソシエーション解放要求を受信するまでの時間の監視	

図 2-24　相手システムとのタイマ監視の範囲の例



2.6.2　UAP とのメッセージ送受信に関するタイマ監視

TP1/NET/OSAS-NIF は，reply 型論理端末で問い合わせメッセージを受信して UAP を起動したあと，UAP から応答メッセージを受け付けるまで時間監視します。

タイマ監視をするかどうか，および監視タイマ値は，論理端末定義（mcftalcle -d）の rplytim オペランドで指定します。監視タイマ値に 0 を指定すると，時間監視はしません。

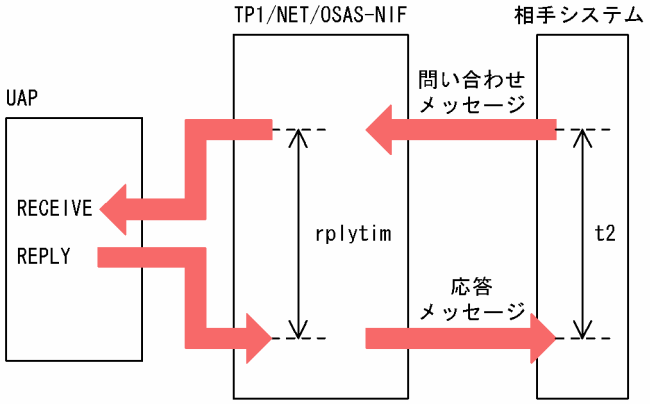
TP1/NET/OSAS-NIF の UAP とのタイマ監視の種類と内容を次の表に示します。また，UAP とのタイマ監視の範囲について図 2-25 に示します。

表 2-8　UAP とのタイマ監視の種類と内容

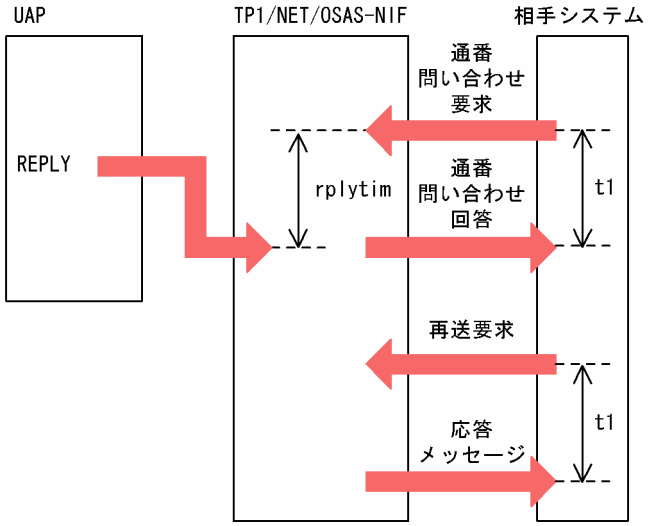
タイマ監視の種類と内容	内容	タイムアウト時の TP1/NET/OSAS-NIF の処理
応答監視タイマ	相手システムから問い合わせメッセージを受信して，入力キューへ登録後，UAP から応答メッセージの送信を受け付けるまでの時間の監視	論理端末を閉塞します。 送信処理を中断します。 応答メッセージは送信済みとして処理します。
	UAP からの応答メッセージの送信受け付け待ちの状態で，論理端末が閉塞したとき，論理端末の閉塞解除処理で，相手システムから通番問い合わせ要求を受信後，UAP から応答メッセージの送信を受け付けるまでの時間の監視	問い合わせメッセージは未受信として処理します。
	UAP からの応答メッセージの送信受け付け待ちの状態で，論理端末が閉塞したとき，論理端末の閉塞解除処理で，相手システムから再送された問い合わせメッセージを受信後 UAP から応答メッセージの送信を受け付けるまでの時間の監視	論理端末を閉塞します。 問い合わせメッセージの受信を拒否します。

図 2-25　UAP とのタイマ監視の範囲の例

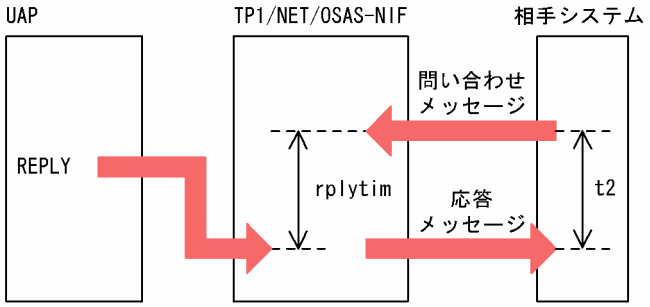
●問い合わせメッセージ受信



●通番問い合わせ要求受信



●再送問い合わせメッセージ受信



（凡例）t1：正常処理監視
t2：ユーザ処理監視
rplytim：応答監視

3 C 言語のライブラリ関数

この章では，TP1/NET/OSAS-NIF 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/OSAS-NIF 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_execap	アプリケーションプログラムの起動
dc_mcf_receive	メッセージの受信
dc_mcf_recvsync	同期型メッセージの後続セグメント受信
dc_mcf_reply	応答メッセージの送信
dc_mcf_resend	メッセージの再送
dc_mcf_send	メッセージの送信
dc_mcf_sendrecv	同期型メッセージの送受信
dc_mcf_tactcn	コネクションの確立
dc_mcf_tactle	論理端末の閉塞解除
dc_mcf_tdctcn	コネクションの解放
dc_mcf_tdctle	論理端末の閉塞
dc_mcf_tlscn	コネクションの状態取得
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_execap － アプリケーションプログラムの起動（C 言語）

形式

ANSI C，C++の形式

```
#include <dcmcf.h>
int dc_mcf_execap(DCLONG action, DCLONG commform, char *resv01,
                  DCLONG active, char *apnam, char *comdata,
                  DCLONG cdataleng)
```

K&R 版 C の形式

```
#include <dcmcf.h>
int dc_mcf_execap(action,commform,resv01,active,apnam,comdata,
                  cdataleng)
DCLONG          action;
DCLONG          commform;
char            *resv01;
DCLONG          active;
char            *apnam;
char            *comdata;
DCLONG          cdataleng;
```

機能

dc_mcf_execap 関数は，TP1/NET/OSAS-NIF の場合，相手システムのアプリケーションプログラムを起動します。

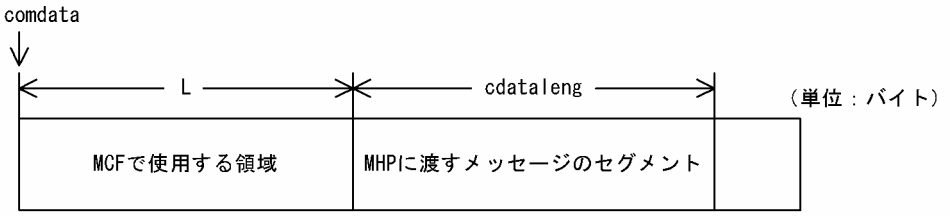
また，システム内の通信では UAP（SPP または MHP）から，apnam に設定したアプリケーション名の MHP を開始させます。

MHP に渡すメッセージの一つのセグメントの最大長は，32000 バイトです。

SPP から dc_mcf_execap 関数を呼び出す場合は，SPP がトランザクションとして処理していることと，その SPP のメイン関数で dc_mcf_open 関数を呼び出していることが前提です。

システム内のアプリケーションプログラムの起動については，マニュアル「OpenTP1 プログラム作成の手引」およびマニュアル「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。

開始させる MHP に渡すメッセージのセグメント形式を次に示します。L は，バッファ形式 1 の場合は 8 バイト，バッファ形式 2 の場合は 4 バイトです。



UAP で値を設定する引数

●action

開始させる MHP に渡すセグメントが論理メッセージの最終セグメントかどうかを，次の形式で設定します。

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

DCMCFESI

先頭セグメントまたは中間セグメントを渡す場合に設定します。この値を設定した dc_mcf_execap 関数を呼び出した場合は，そのあとに必ず action に DCMCFEMI を設定した dc_mcf_execap 関数を呼び出してください。

DCMCFEMI

最終セグメントを渡す場合，および論理メッセージが単一セグメントの場合に設定します。さらに，先頭セグメントまたは中間セグメントの引き渡し後，メッセージの引き渡しの終了を連絡する場合にもこの値を設定します。

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。

●resv01

ヌル文字列を設定します。

●active

0 を設定します。

●apnam

起動する MHP のアプリケーション名を設定します。アプリケーション名は最大 8 バイトの長さです。アプリケーション名の最後にはヌル文字を付けてください。

●comdata

起動する MHP に渡すセグメントの内容を設定します。先頭セグメントの引き渡し後，メッセージの引き渡しの終了を連絡する場合で，セグメントの内容がないときも必ず設定してください。

●cdataleng

起動する MHP に渡すセグメントの長さを設定します。

先頭セグメントの引き渡し後，メッセージの引き渡しの終了を連絡する場合で，セグメントの内容がないときは，0 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	メッセージキューへの入出力処理時に障害が発生しました。
		メッセージキューが閉塞されています。
		メッセージキューが割り当てられていません。
		cdataleng 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，apnam に設定した MHP を起動できません。
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	apnam に設定したアプリケーション名の MHP を開始しようとしましたが，開始しようとした MHP の管理テーブルを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_execap 関数を呼び出しました。
		＜ SPP の実行でリターンした場合＞ トランザクションの処理でない SPP から，dc_mcf_execap 関数を呼び出しました。
DCMCFRTN_72001	-13001	apnam に設定したアプリケーション名は，MCF で定義されていません。
		apnam に設定したアプリケーション名が間違っています。
		MCF マネジャ定義の通信サービス定義（mcfmcname）に，アプリケーション起動プロセス名または MCF 通信プロセス名を指定していません。
		アプリケーション起動プロセスまたは MCF 通信プロセスに対応するアプリケーション環境定義（mcfaenv -p）に，アプリケーション起動プロセス識別子を指定していません。
		アプリケーション環境定義（mcfaenv -p）で指定したアプリケーション起動プロセス識別子と，アプリケーション起動プロセスまたは MCF 通信プロセスの MCF 環境定義（mcftenv -s）で指定する識別子が一致していません。
		＜論理端末名称を指定してアプリケーションを起動する場合＞ • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の lname オペランドに論理端末を指定していません。

リターン値	リターン値（数値）	意味
DCMCFRTN_72001	-13001	• 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の lname オペランドに指定した論理端末を，MCF 通信プロセスの論理端末定義（mcftalcle）に定義していません。 • 起動先アプリケーションのアプリケーション属性定義に指定した論理端末が，request 型論理端末または send 型論理端末ではありません。 • 起動先アプリケーションのアプリケーション属性定義で指定した論理端末は，アプリケーション起動を使えません。
		＜コネクション ID を指定してアプリケーションを起動する場合＞ • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の cname オペランドにコネクション ID を指定していません。 • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の cname オペランドに指定したコネクション ID を，MCF 通信プロセスのコネクション定義（mcftalccn）に定義していません。 • MCF 通信プロセスの論理端末定義（mcftalcle）に，request 型論理端末を指定していません。
		＜ SPP からアプリケーションを起動する場合＞ • アプリケーション起動プロセス識別子を起動元の UAP のユーザサービス定義，またはユーザサービスデフォルト定義の mcf_psv_id オペランドに指定していません。 • 起動元の UAP のユーザサービス定義，またはユーザサービスデフォルト定義の mcf_psv_id オペランドに指定しているアプリケーション起動プロセス識別子が，アプリケーション起動プロセス，または MCF 通信プロセスの MCF 環境定義（mcftenv -s），およびアプリケーション環境定義（mcfaenv -p）で指定しているアプリケーション起動プロセス識別子と一致していません。 • 起動元の UAP のユーザサービス定義，またはユーザサービスデフォルト定義の mcf_mgrid オペランドに指定している MCF マネジャ識別子が，アプリケーション起動プロセスが属している MCF マネジャの識別子と一致していません。
DCMCFRTN_72005	-13005	＜ action で DCMCFESI を設定した場合＞ cdataleng に 0 バイト，またはマイナス値を設定しています。
DCMCFRTN_72007	-13007	dc_mcf_reply 関数をすでに呼び出した応答型（type=ans）の MHP から，応答型の MHP を dc_mcf_execap 関数で起動させています。
		dc_mcf_reply 関数をすでに呼び出した継続問い合わせ応答型（type=cont）の MHP から，継続問い合わせ応答型の MHP を dc_mcf_execap 関数で起動させています。
DCMCFTRN_72009	-13009	応答型（type=ans）の MHP から，dc_mcf_execap 関数で応答型の MHP を 2 回以上起動させています。
		継続問い合わせ応答型（type=cont）の MHP から，dc_mcf_execap 関数で継続問い合わせ応答型の MHP を 2 回以上起動させています。
DCMCFTRN_72011	-13011	応答型（type=ans）の MHP から，dc_mcf_execap 関数で応答型の MHP を 2 回以上起動させています。
		継続問い合わせ応答型（type=cont）でない MHP から，dc_mcf_execap 関数で継続問い合わせ応答型の MHP を 2 回以上起動させています。

リターン値	リターン値（数値）	意味
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_72026	-13026	action に設定したセグメント種別（DCMCFESI または DCMCFEMI を設定）の値が間違っています。
DCMCFRTN_72041	-13041	＜ action で DCMCFEMI を設定した場合＞ • cdataleng に 0 バイト，またはマイナス値を設定しています。 • action に DCMCFESI を設定した dc_mcf_execap 関数を呼び出さないで，メッセージの引き渡しの終了を連絡しています。
DCMCFRTN_77001	-18001	起動しようとするアプリケーションに対応する論理端末は，現在仕掛り中で使用できません。または，使用できる論理端末がありません。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_receive － メッセージの受信（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

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

機能

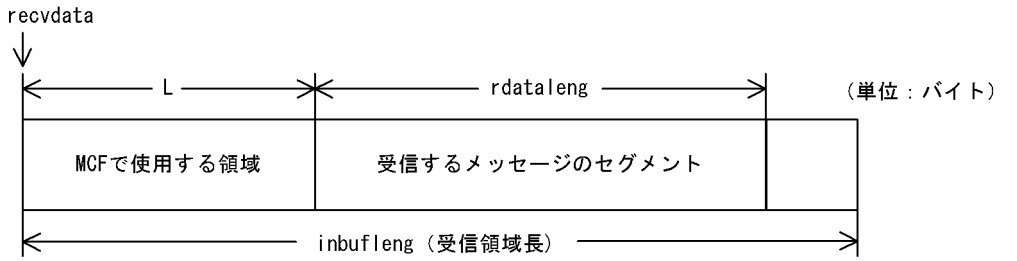
論理端末に届いたメッセージのうち，一つのセグメントを受信します。セグメントの数だけ dc_mcf_receive 関数を呼び出すと，一つの論理メッセージを受信できます。

受信できるメッセージの一つのセグメントの最大長は，65452 バイトです。

dc_mcf_recieve 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

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



UAP で値を設定する引数

●action

受信するセグメント，および使用するバッファ形式を，次の形式で設定します。

```
{DCMCFFRST｜DCMCFSEG}〔｜{DCMDFBUF1｜DCMDFBUF2}〕
```

DCMCFFRST

先頭セグメントを受信する場合や，論理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。

DCMDFBUF1

バッファ形式 1 を使用する場合に設定します。

DCMDFBUF2

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

受信したメッセージのセグメントが返されます。

●rdataleng

受信したメッセージの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，再び dc_mcf_receive 関数を呼び出した場合は，リターン値 DCMCFRTN_72000 が返されます。

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 関数を呼び出しました。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ました。
DCMCFRTN_72016	-13016	action に設定した値が間違っています。
		resv01 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。
DCMCFRTN_72025	-13025	action に設定したセグメント種別（DCMCFFRST または DCMCFSEG）の値が間違っています。
DCMCFRTN_72036	-13036	inbufleng の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

(凡例)

－：該当しません。

dc_mcf_recvsync － 同期型メッセージの後続セグメント受信（C 言語）

形式

ANSI C，C++の形式

```
#include<dcmcf.h>
int dc_mcf_recvsync (DCLONG action, DCLONG commform,
                     char *termnam,char *resv01,
                     char *recvdata,DCLONG *rdataleng,
                     DCLONG inbufleng, DCLONG *time,
                     DCLONG resv02)
```

K&R 版 C の形式

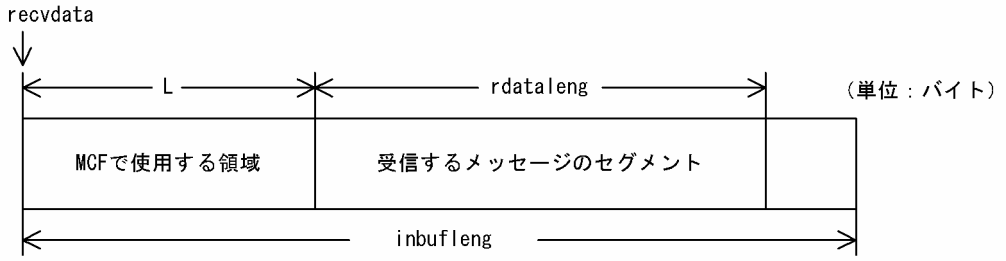
```
#include<dcmcf.h>
int dc_mcf_recvsync (action, commform, termnam, resv01,
                     recvdata, rdataleng, inbufleng,
                     time, resv02)
DCLONG action;
DCLONG commform;
char   *termnam;
char   *resv01;
char   *recvdata;
DCLONG *rdataleng;
DCLONG inbufleng;
DCLONG *time;
DCLONG resv02;
```

機能

相手システムから届いた論理メッセージ（同期型で受信）のうち，先頭セグメント以外の後続する一つのセグメントを受信します。先頭セグメントを受信する dc_mcf_sendrecv 関数のあとに，後続するセグメントの数だけ dc_mcf_recvsync 関数を発行することによって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，65452 バイトです。

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



UAP で値を設定する引数

●action

受信するセグメント，および使用するバッファ形式を次の形式で設定します。

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

DCMCFSEG

中間セグメントまたは最終セグメントの受信を示す DCMCFSEG 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

dc_mcf_sendrecv 関数の termnam パラメタで指定した入力元の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。論理端末名称の最後にはヌル文字を付けてください。

●resv01

NULL またはヌル文字列を設定します。

●recvdata

セグメントを受信する領域を設定します。

dc_mcf_recvsync 関数が終了すると，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。

●resv02

DCNOFLAGS を設定します。

OpenTP1 から値が返される引数

●recvdata

受信したメッセージのセグメントの内容が返されます。

●rdataleng

受信したメッセージのセグメントの長さが返されます。

●time

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

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	メッセージの最終セグメントを受信したあとで，その次のセグメントを受信する dc_mcf_recvsync 関数を呼び出しています。直前に呼び出した dc_mcf_recvsync 関数でメッセージはすべて受信しました。 このリターン値が返されたあとに，再び次のセグメントを受信する dc_mcf_recvsync 関数を呼び出した場合は，リターン値 DCMCFRTN_72000 が返されます。
DCMCFRTN_71108	-12108	メッセージ送受信に必要な管理テーブルが確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	＜ 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_recvsync 関数を呼び出しました。
		リターン値 DCMCFRTN_71001 が返されたあとに，再び次のセグメントを受信する dc_mcf_recvsync 関数を呼び出しました。
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_recvsync 関数を呼び出せない論理端末を設定しています。
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。

リターン値	リターン値 (数値)	意味
DCMCFRTN_73018	-14018	resv02 に設定した値が間違っています。
上記以外	−	プログラムの破壊などによる，予期しないエラーが発生しました。

(凡例)

−：該当しません。

dc_mcf_reply － 応答メッセージの送信（C 言語）

形式

ANSI C，C++の形式

```
#include<dcmcf.h>
int dc_mcf_reply (DCLONG action, DCLONG commform, char *resv01,
                  char *resv02, char *senddata, DCLONG sdataleng,
                  char *resv03, DCLONG opcd)
```

K&R 版 C の形式

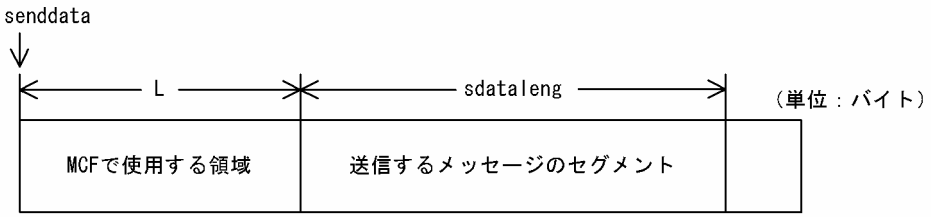
```
#include<dcmcf.h>
int dc_mcf_reply(action, commform, resv01, resv02, senddata,
                 sdataleng, resv03, opcd)
DCLONG action;
DCLONG commform;
char   *resv01;
char   *resv02;
char   *senddata;
DCLONG sdataleng;
char   *resv03;
DCLONG opcd;
```

機能

メッセージを入力した論理端末に送信する応答メッセージのうち，一つのセグメントを送信します。必要なセグメントの数だけ dc_mcf_reply 関数を発行することによって，一つの論理メッセージを送信できます。

送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

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



UAP で値を設定する引数

●action

送信するセグメント，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。

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

DCMCFESI

先頭セグメントまたは中間セグメントを送信する場合に設定します。

DCMCFEMI

最終セグメントを送信する場合や，論理メッセージが単一セグメントの場合に設定します。メッセージの送信の終了を連絡するために，最後は必ずこの値を設定してください。

DCMCFSEQ

出力通番を付ける場合に設定します。

ただし，非応答型のアプリケーションの場合は，DCMCFSEQ を指定しても，MCF は応答メッセージに出力通番を付けません。

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

DCNOFLAGS を設定します。

●resv01

ヌル文字列を設定します。

●resv02

NULL またはヌル文字列を設定します。

●senddata

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

先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときも，必ず設定してください。

●sdataleng

送信するセグメントの長さを設定します。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときは，0 を設定してください。

●resv03

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しています。
		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を確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_reply 関数を呼び出しました。 • 非応答型のアプリケーションからの問い合わせ応答をしない（UAP 共通定義（mcfmuap -c）の noansreply オペランドに no を指定）場合に，非応答型のアプリケーションから，dc_mcf_reply 関数を呼び出しました。
		＜ SPP の実行でリターンした場合＞ SPP では dc_mcf_reply 関数を呼び出せません。
DCMCFRTN_72001	-13001	入力元論理端末は，応答メッセージの送信をサポートしていません。
DCMCFRTN_72005	-13005	＜ action で DCMCFESI を設定した場合＞ sdataleng に 0 バイト，またはマイナス値を設定しています。
DCMCFRTN_72008	-13008	最終セグメントを送信する dc_mcf_reply 関数を呼び出したあとで，再び dc_mcf_reply 関数を呼び出しています。
		応答型のアプリケーションから dc_mcf_execap 関数を呼び出して応答型のアプリケーションを起動させたあとに，dc_mcf_reply 関数を呼び出しています。
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。

リターン値	リターン値（数値）	意味
DCMCFRTN_72016	-13016	resv01，resv02 または resv03 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72026	-13026	action に設定したセグメント種別（DCMCFESI または DCMCFEMI）の値が間違っています。
DCMCFRTN_72041	-13041	＜ action で DCMCFEMI を設定した場合＞ • sdataleng に 0 バイト，またはマイナス値を設定しています。 • action に DCMCFESI を設定した dc_mcf_reply 関数を呼び出さないで，メッセージの送信の終了を連絡しています。
DCMCFRTN_72045	-13045	継続問い合わせ応答型のアプリケーションはサポートしていません。
DCMCFRTN_72047	-13047	継続問い合わせ応答型のアプリケーションはサポートしていません。
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（一般送信，優先送信）

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（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設定してください。

UAP で値を設定する引数

●action

再送するセグメント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

優先の送信メッセージとして再送する場合に設定します。

DCMCFLAST

再送する対象のメッセージを検索するキーとして，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

メッセージの送信を示す，DCMCFOUT を設定します。

●rtermnam

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

●resv01

NULL を設定します。

●oseqid

再送するメッセージを検索するキーとして，以前に送信したメッセージの送信種別を設定します。次のどちらかの値を設定してください。

DCMCFRID_NORM

一般の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（一般の送信メッセージを対象）が設定されます。

●orgseq

再送するメッセージを検索するキーとして，以前に送信したメッセージの出力通番を設定します。action に 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

再送する対象のメッセージを検索するキーとして，以前に送信したメッセージの出力先の論理端末名称を設定します。論理端末名称は最大８バイトの長さです。論理端末名称の最後に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 オペランドに memory を指定）。 • 論理端末が閉塞しているなどの要因によって，送信メッセージが送信済みになっていません。 • 送信済みのメッセージがディスクキューに保持されていません（保持メッセージ数はメッセージキューサービス定義の quegrp コマンドの-m オプションで指定します）。
		otermnam で指定した論理端末に割り当てられているメッセージキューは，システム開始時に障害が発生したため，メモリキューを代用して縮退運転をしています。
DCMCFRTN_BUF_SHORT	-11905	再送するメッセージのセグメントの長さが，UAP 共通定義（mcfmuap -e）で指定した値を超えています。
DCMCFRTN_71002	-12002	メッセージキューへの出力処理中に障害が発生しました。
		メッセージキューが閉塞されています。
		メッセージキューが割り当てられていません。
		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。

リターン値	リターン値（数値）	意味
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	メッセージを再送しようとしましたが，再送先の管理テーブルを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 関数を呼び出しました。
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。
		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，および resv04 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72017	-13017	action に設定した出力通番の要否（DCMCFSEQ または DCMCFNSEQ）の値が間違っています。
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。
上記以外	−	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

−：該当しません。

注意事項

メッセージの再送時には，MCF マネジャ定義の UAP 共通定義（mcfmuap）の-e オプションおよび-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は，どのメッセージを再送するかは保証できません。

dc_mcf_send － 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

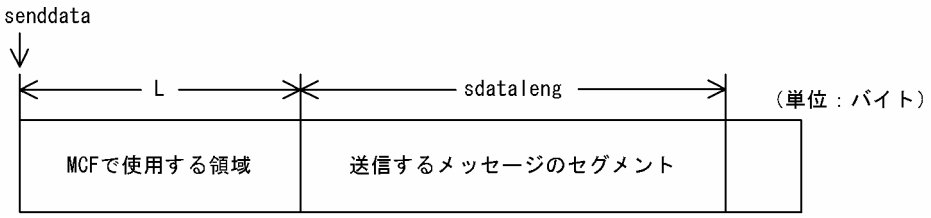
```
#include<dcmcf.h>
int dc_mcf_send(action, commform, termnam, resv01, senddata,
                sdataleng, resv02, opcd)
DCLONG    action;
DCLONG    commform;
char      *termnam;
char      *resv01;
char      *senddata;
DCLONG    sdataleng;
char      *resv02;
DCLONG    opcd;
```

機能

MCF で管理する論理端末に送信するメッセージの，一つのセグメントを送信要求します。必要なセグメントの数だけ，dc_mcf_send 関数を呼び出すと，一つの論理メッセージを送信できます。

送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

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



UAP で値を設定する引数

●action

送信するセグメント，一般か優先か，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。

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

DCMCFESI

先頭セグメントまたは中間セグメントを送信する場合に設定します。

DCMCFEMI

最終セグメントを送信する場合や，論理メッセージが単一セグメントの場合に設定します。メッセージの送信の終了を連絡するために，最後は必ずこの値を設定してください。

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領域を設定します。一つのセグメントで 32000 バイトまで送信できます。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときも，必ず設定してください。

●sdataleng

送信するセグメントの長さを設定します。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときは，0 を設定してください。

●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を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＞ トランザクションの処理でない SPP で，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。
		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_ 72005	-13005	＜ action で DCMCFESI を設定した場合＞ sdataleng に 0 バイト，またはマイナス値を設定しています。
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。

リターン値	リターン値（数値）	意味
DCMCFRTN_72016	-13016	resv01，resv02 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。
		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（DCMCFESI または DCMCFEMI）の値が間違っています。
DCMCFRTN_72041	-13041	＜ action で DCMCFEMI を設定した場合＞ • sdataleng に 0 バイト，またはマイナス値を設定しています。 • action に DCMCFESI を設定した dc_mcf_send 関数を呼び出さないで，メッセージの送信の終了を連絡しています。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_sendrecv － 同期型メッセージの送受信（C 言語）

形式

ANSI C，C++の形式

```
#include<dcmcf.h>
int dc_mcf_sendrecv (DCLONG action, DCLONG commform,
                     char *termnam, char *resv01,
                     char *senddata, DCLONG sdataleng,
                     char *recvdata, DCLONG *rdataleng,
                     DCLONG inbufleng, DCLONG *time,
                     DCLONG resv02)
```

K&R 版 C の形式

```
#include<dcmcf.h>
int dc_mcf_sendrecv(action, commform, termnam, resv01,
                    senddata, sdataleng, recvdata,
                    rdataleng, inbufleng, time, resv02)
DCLONG action;
DCLONG commform;
char   *termnam;
char   *resv01;
char   *senddata;
DCLONG sdataleng;
char   *recvdata;
DCLONG *rdataleng;
DCLONG inbufleng;
DCLONG *time;
DCLONG resv02;
```

機能

同期型でメッセージを送信したあと，同期型でメッセージを受信します。

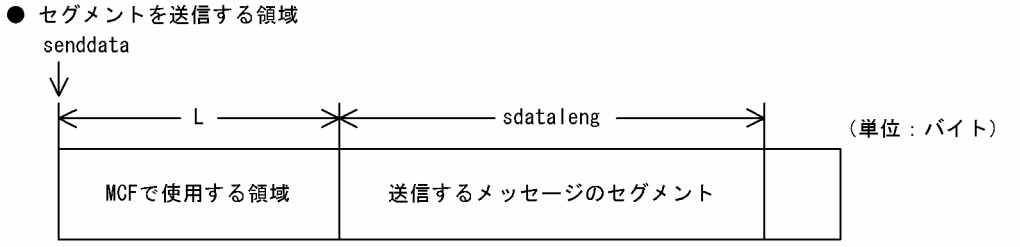
相手システムへ送る論理メッセージのうち，一つのセグメントを送信します。必要なセグメントの数だけ dc_mcf_sendrecv 関数を発行することによって，一つの論理メッセージを送信します。メッセージの最終セグメントを送信すると，dc_mcf_sendrecv 関数は相手システムからの応答を待ちます。応答が届くと，そのメッセージの先頭セグメントを受信します。

中間セグメントおよび最終セグメントを受信する場合は，dc_mcf_recvsync 関数を発行してください。

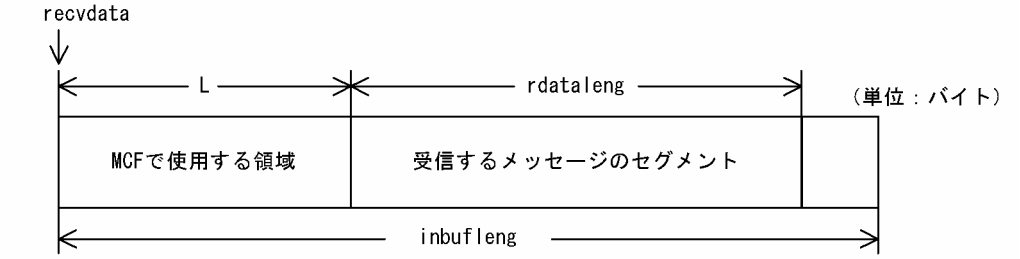
受信できるメッセージの一つのセグメントの最大長は，65452 バイトです。また，送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

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

● セグメントを送信する領域



● セグメントを受信する領域



UAP で値を設定する引数

●action

送信するセグメント，および使用するバッファ形式を次の形式で設定します。

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

DCMCFESI

先頭セグメントまたは中間セグメントを送信する場合に設定します。

DCMCFEMI

最終セグメントを送信する場合や，論理メッセージが単一セグメントの場合に設定します。メッセージの送信の終了を連絡するために，最後は必ずこの値を設定してください。この値を設定して dc_mcf_sendrecv 関数を発行すると，応答メッセージを待ちます。

DCMCFBUF1

バッファ形式 1 を使用する場合に設定します。

DCMCFBUF2

バッファ形式 2 を使用する場合に設定します。

●commform

同期型メッセージの送受信を示す DCMCFIO を設定します。

●termnam

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

●resv01

NULL またはヌル文字列を設定します。

●senddata

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

先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときも，必ず設定してください。

●sdataleng

送信するセグメントの長さを設定します。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときは，0 を設定してください。

●recvdata

セグメントを受信する領域を設定します。

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

処理終了後，recvdata には OpenTP1 から値が返ります。

●inbufleng

セグメントを受信する領域の長さを設定します。

●resv02

DCNOFLAGS を設定します。

OpenTP1 から値が返される引数

●recvdata

受信したメッセージの先頭セグメントの内容が返されます。

●rdataleng

受信したメッセージの先頭セグメントの長さが返されます。

●time

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

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71002	-12002	メッセージキューへの入出力処理時に障害が発生しました。
		メッセージキューが閉塞されています。
		メッセージキューが割り当てられていません。
		sdataleng に 32000 バイトを超える値を設定しています。
		MCF が終了処理中のため，メッセージの送受信を受け付けられません。
DCMCFRTN_71003	-12003	メッセージキューが満杯です。
DCMCFRTN_71004	-12004	メッセージを格納するバッファをメモリ上に確保できませんでした。
DCMCFRTN_71108	-12108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	＜ 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_sendrecv 関数を呼び出しています。
DCMCFRTN_72001	-13001	termnam に設定した論理端末名称が間違っています。
		termnam に設定した論理端末名称は，定義されていません。
		dc_mcf_sendrecv 関数を呼び出せない論理端末を設定しています。
DCMCFRTN_72005	-13005	＜ action で DCMCFESI を設定した場合＞ sdataleng に 0 バイト，またはマイナス値を設定しています。
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_72026	-13026	action に設定したセグメント種別（DCMCFESI または DCMCFEMI）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。

リターン値	リターン値（数値）	意味
DCMCFRTN_72041	-13041	＜ action で DCMCFEMI を設定した場合＞ • sdataleng に 0 バイト，またはマイナス値を設定しています。 • action に DCMCFESI を設定した dc_mcf_sendrecv 関数を呼び出さないで，メッセージの送信の終了を連絡しています。
DCMCFRTN_73001	-14001	同期型送受信監視時間（mcfmuap -t sndrcvtim）に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
		dc_mcf_sendrecv 関数を呼び出したあとで，論理端末の閉塞またはコネクションの解放が発生しました。
		出力先の論理端末で MCF 通信プロセスの内部障害が発生しました。
DCMCFRTN_73002	-14002	論理端末が閉塞しています。
DCMCFRTN_73004	-14004	メッセージの送信処理中にタイムアウトが発生しました。
DCMCFRTN_73005	-14005	メッセージの受信処理中にタイムアウトが発生しました。
DCMCFRTN_73008	-14008	運用コマンドまたは障害による論理端末閉塞処理中に，dc_mcf_sendrecv 関数を呼び出しました。
DCMCFRTN_73009	-14009	論理端末の閉塞解除の処理中に，dc_mcf_sendrecv 関数を呼び出しました。
DCMCFRTN_73010	-14010	入力メッセージ編集 UOC または出力メッセージ編集 UOC で障害が発生しました。
DCMCFRTN_73015	-14015	出力先の論理端末は，ほかの UAP で仕掛り中です。
DCMCFRTN_73018	-14018	resv02 に設定した値が間違っています。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_tactcn － コネクションの確立（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

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

機能

コネクションを確立します。

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

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

UAP で値を設定する引数

●action

確立するコネクションの指定方法を次の形式で設定します。

```
{DCMCFLE | DCMCFCN}
```

DCMCFLE

確立するコネクションを論理端末名称で指定するときに設定します。

DCMCFCN

確立するコネクションをコネクション ID で指定するときに設定します。

●cnopt

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

構造体の形式を次に示します。

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

- mcfid

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

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

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

  注※

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

- resv01

  領域をヌル文字で埋めます。

- idnam

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

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

  領域をヌル文字で埋めます。

●proinf，resv02，resv03，resv04

NULL を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tactcn 関数が受け付けられません。
DCMCFRTN_71002	-12002	MCF が終了処理中のため，dc_mcf_tactcn 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tactcn 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71007	-12007	指定されたコネクション ID は登録されていません。
DCMCFRTN_71008	-12008	指定された論理端末名称は登録されていません。
DCMCFRTN_71009	-12009	dc_mcf_tactcn 関数が，該当する MCF 通信プロセスではサポートされていません。
DCMCFRTN_71010	-12010	MCF 通信プロセスにコネクションの確立を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71011	-12011	コネクションが削除されているため，dc_mcf_tactcn 関数が受け付けられません。
DCMCFRTN_71014	-12014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループ名を指定しています。
DCMCFRTN_72050	-13050	action に未サポートのフラグを設定しています。
DCMCFRTN_72051	-13051	cnopt に NULL が設定されています。
DCMCFRTN_72052	-13052	proinf に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72054	-13054	resv03 に NULL が設定されていません。
DCMCFRTN_72055	-13055	resv04 に NULL が設定されていません。
DCMCFRTN_72060	-13060	action には DCMCFLE と DCMCFCN を同時に設定できません。
DCMCFRTN_72061	-13061	dcmcf_tactcnopt の mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72062	-13062	dcmcf_tactcnopt の resv01 がヌル文字で埋められていません。

リターン値	リターン値（数値）	意味
DCMCFRTN_72063	-13063	dcmcf_tactcnopt の idnam の先頭がヌル文字です。
DCMCFRTN_72064	-13064	dcmcf_tactcnopt の resv02 がヌル文字で埋められていません。
DCMCFRTN_72065	-13065	dcmcf_tactcnopt の resv03 がヌル文字で埋められていません。
DCMCFRTN_72066	-13066	dcmcf_tactcnopt の scnnam がヌル文字で埋められていません。
DCMCFRTN_72067	-13067	dcmcf_tactcnopt の resv04 がヌル文字で埋められていません。
DCMCFRTN_72068	-13068	dcmcf_tactcnopt の yournam がヌル文字で埋められていません。
DCMCFRTN_72069	-13069	dcmcf_tactcnopt の resv05 がヌル文字で埋められていません。
DCMCFRTN_72070	-13070	dcmcf_tactcnopt の hostnam がヌル文字で埋められていません。
DCMCFRTN_72071	-13071	dcmcf_tactcnopt の resv06 がヌル文字で埋められていません。
DCMCFRTN_72072	-13072	dcmcf_tactcnopt の resv07 がヌル文字で埋められていません。
DCMCFRTN_72073	-13073	dcmcf_tactleopt の idnam に設定された文字数が 9 以上です。
DCMCFRTN_72074	-13074	dcmcf_tactcnopt の idnam に設定された文字列中に不正な文字があります。

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/OSAS-NIF が正常に受け付けたことを意味します。このため，論理端末の閉塞解除が正常に完了したことを示すものではありません。

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

UAP で値を設定する引数

●action

閉塞解除する論理端末の指定方法を次の形式で設定します。

```
DCMCFLE
```

DCMCFLE

閉塞解除する論理端末を論理端末名称で指定するときに設定します。

●leopt

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

構造体の形式を次に示します。

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

- mcfid

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

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

  注※

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

- resv01

  領域をヌル文字で埋めます。

- idnam

  閉塞解除する論理端末の名称を設定します。論理端末名称は最大 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。

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

dc_mcf_tdctcn － コネクションの解放（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

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

機能

コネクションを解放します。

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

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

UAP で値を設定する引数

●action

解放するコネクションの指定方法を次の形式で設定します。

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

DCMCFLE

解放するコネクションを論理端末名称で指定するときに設定します。

DCMCFCN

解放するコネクションをコネクション ID で指定するときに設定します。

DCMCFFRC

コネクションを強制的に解放するときに設定します。

●cnopt

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

構造体の形式を次に示します。

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

- mcfid

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

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

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

  注※

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

- resv01

  領域をヌル文字で埋めます。

- idnam

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

- resv02，resv03，scnnam，resv04，resv05

  領域をヌル文字で埋めます。

●proinf，resv02，resv03，resv04

NULL を設定します。

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12001	MCF が開始処理中のため，dc_mcf_tdctcn 関数が受け付けられません。
DCMCFRTN_71002	-12002	MCF が終了処理中のため，dc_mcf_tdctcn 関数が受け付けられません。
DCMCFRTN_71004	-12004	dc_mcf_tdctcn 関数の処理中にメモリ不足が発生しました。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71007	-12007	指定されたコネクション ID は登録されていません。
DCMCFRTN_71008	-12008	指定された論理端末名称は登録されていません。
DCMCFRTN_71009	-12009	dc_mcf_tdctcn 関数が，該当する MCF 通信プロセスではサポートされていません。
DCMCFRTN_71010	-12010	MCF 通信プロセスにコネクションの解放を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71011	-12011	コネクションが削除されているため，dc_mcf_tdctcn 関数が受け付けられません。
DCMCFRTN_71014	-12014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループ名を指定しています。
DCMCFRTN_72050	-13050	action に未サポートのフラグを設定しています。
DCMCFRTN_72051	-13051	cnopt に NULL が設定されています。
DCMCFRTN_72052	-13052	proinf に NULL が設定されていません。
DCMCFRTN_72053	-13053	resv02 に NULL が設定されていません。
DCMCFRTN_72054	-13054	resv03 に NULL が設定されていません。
DCMCFRTN_72055	-13055	resv04 に NULL が設定されていません。
DCMCFRTN_72060	-13060	action には DCMCFLE と DCMCFCN を同時に設定できません。
DCMCFRTN_72061	-13061	dcmcf_tdctcnopt の mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72062	-13062	dcmcf_tdctcnopt の resv01 がヌル文字で埋められていません。

リターン値	リターン値（数値）	意味
DCMCFRTN_72063	-13063	dcmcf_tdctcnopt の idnam の先頭がヌル文字です。
DCMCFRTN_72064	-13064	dcmcf_tdctcnopt の resv02 がヌル文字で埋められていません。
DCMCFRTN_72065	-13065	dcmcf_tdctcnopt の resv03 がヌル文字で埋められていません。
DCMCFRTN_72066	-13066	dcmcf_tdctcnopt の scnnam がヌル文字で埋められていません。
DCMCFRTN_72067	-13067	dcmcf_tdctcnopt の resv04 がヌル文字で埋められていません。
DCMCFRTN_72069	-13069	dcmcf_tdctcnopt の resv05 がヌル文字で埋められていません。
DCMCFRTN_72073	-13073	dcmcf_tdctcnopt の idnam に設定された文字数が 9 以上です。
DCMCFRTN_72074	-13074	dcmcf_tdctcnopt の idnam に設定された文字列中に不正な文字があります。

dc_mcf_tdctle － 論理端末の閉塞（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

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

機能

論理端末を閉塞します。

なお，dc_mcf_tdctle 関数の正常終了は，論理端末の閉塞要求を TP1/NET/OSAS-NIF 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。

  注※

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

- resv01

  領域をヌル文字で埋めます。

- idnam

  閉塞する論理端末の名称を設定します。論理端末名称は最大 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。

リターン値	リターン値（数値）	意味
DCMCFRTN_71006	-12006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
DCMCFRTN_71008	-12008	指定された論理端末名称は登録されていません。
DCMCFRTN_71009	-12009	dc_mcf_tdctle 関数が，該当する MCF 通信プロセスではサポートされていません。
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_tlscn － コネクションの状態取得（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

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

機能

コネクションの状態を取得します。

UAP で値を設定する引数

●action

状態を取得するコネクションの指定方法を次の形式で設定します。

```
{DCMCFLE | DCMCFCN}
```

DCMCFLE

状態を取得するコネクションを論理端末名称で指定するときに設定します。

DCMCFCN

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

●cnopt

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

構造体の形式を次に示します。

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

- mcfid

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

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

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

  注※

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

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

- resv01

  領域をヌル文字で埋めます。

- idnam

  状態を取得するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は最大 8 バイトの長さです。論理端末名称，またはコネクション ID 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

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

処理終了後は，該当するコネクションの個数が返されます。

●inf

コネクション状態を格納する領域 dcmcf_cninf を設定します。

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

●resv04

NULL を設定します。

OpenTP1 から値が返される引数

●infcnt

この関数の対象となったコネクションの個数が返されます。

●inf

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

構造体の形式を次に示します。

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

- idnam

  要求したコネクションのコネクション ID が設定されます。コネクション ID は最大 8 バイトの長さです。コネクション ID の最後にはヌル文字が付けられます。

- resv01

  領域をヌル文字で埋めます。

- pnam

  要求したコネクションのプロトコル種別が設定されます。プロトコル種別の最後にはヌル文字が付けられます。

  NIF

  NIF/OSI プロトコル

- status

  要求したコネクションの状態として，次の値が設定されます。

  DCMCF_CNST_ACT

  確立状態

  DCMCF_CNST_ACT_B

  確立処理中状態

  DCMCF_CNST_DCT

  解放状態

DCMCF_CNST_DCT_B

解放処理中状態

- resv02

領域をヌル文字で埋めます。

リターン値

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

リターン値	リターン値（数値）	意味
DCMCFRTN_72060	-13060	action には DCMCFLE と DCMCFCN を同時に設定できません。
DCMCFRTN_72061	-13061	dcmcf_tlscnopt の mcfid に 0 未満または 240 以上の値が設定されています。
DCMCFRTN_72062	-13062	dcmcf_tlscnopt の resv01 がヌル文字で埋められていません。
DCMCFRTN_72063	-13063	dcmcf_tlscnopt の idnam の先頭がヌル文字です。
DCMCFRTN_72064	-13064	dcmcf_tlscnopt の resv02 がヌル文字で埋められていません。
DCMCFRTN_72065	-13065	dcmcf_tlscnopt の resv03 がヌル文字で埋められていません。
DCMCFRTN_72067	-13067	dcmcf_tlscnopt の resv04 がヌル文字で埋められていません。
DCMCFRTN_72073	-13073	dcmcf_tlscnopt の idnam に設定された文字数が 9 以上です。
DCMCFRTN_72074	-13074	dcmcf_tlscnopt の idnam に設定された文字列中に不正な文字があります。
DCMCFRTN_72076	-13076	infcnt に 1 が設定されていません。

dc_mcf_tlsle － 論理端末の状態取得（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

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

機能

論理端末の状態を取得します。

UAP で値を設定する引数

●action

状態を取得する論理端末の指定方法を次の形式で設定します。

```
DCMCFLE
```

DCMCFLE

状態を取得する論理端末を論理端末名称で指定するときに設定します。

●leopt

この関数の対象となった論理端末の情報を，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。

  注※

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

- resv01

  領域をヌル文字で埋めます。

- idnam

  状態を取得する論理端末の名称を設定します。論理端末名称は最大 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。

リターン値	リターン値（数値）	意味
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_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。

リターン値	リターン値（数値）	意味
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/OSAS-NIF 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/OSAS-NIF 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	'EXECAP△△'	アプリケーションプログラムの起動
	'RECEIVE△'	メッセージの受信
	'RECVSYNC'	同期型メッセージの後続セグメント受信
	'REPLY△△△'	応答メッセージの送信
	'RESEND△△'	メッセージの再送
	'SEND△△△△'	メッセージの送信
	'SENDRECV'	同期型メッセージの送受信
	'TACTCN△△'	コネクションの確立
	'TACTLE△△'	論理端末の閉塞解除
	'TDCTCN△△'	コネクションの解放
	'TDCTLE△△'	論理端末の閉塞
	'TLSCN△△△'	コネクションの状態取得
	'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。

なお，TP1/NET/OSAS-NIF では，通信相手システムのアプリケーションプログラムを起動することも，メッセージの送信に該当します。

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。

その他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/OSAS-NIF のメッセージ送受信の通信文で，通信記述項に指定できる句の指定要否を，次の表に示します。

表 4-3　通信記述項に指定できる句の指定要否

データ名を指定する句	データ領域の値の設定元							
	1.	2.	3.	4.	5.	6.	7.	8.
STATUS KEY	B	B	B	B	B	B	B	B
SYMBOLIC TERMINAL	B	U_1※	U_1	U_1	−	−	U_1	U_1
MESSAGE DATE	B	B	−	−	−	−	−	−
MESSAGE TIME	B	B	−	−	−	−	−	−
SYNCHRONOUS MODE	U_2	U_2	U_2	U_2	U_2	U_2	U_1	−
SWITCHING MODE	−	−	U_2	U_2	−	−	−	−
DETAIL MODE	−	−	U_2	U_2	U_2	U_2	−	−

（凡例）

1.：先頭セグメントの非同期受信（RECEIVE）

2.：中間，最終セグメントの非同期受信（RECEIVE）

3.：一方送信メッセージの先頭，中間セグメントの非同期送信（SEND）

4.：一方送信メッセージの単一，最終セグメントの非同期送信（SEND）

5.：応答メッセージの先頭，中間セグメントの非同期送信（SEND）

6.：応答メッセージの単一，最終セグメントの非同期送信（SEND）

7.：単一セグメントの同期送受信（SEND）

8.：アプリケーションプログラムの起動（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※

先頭メッセージ受信時の RECEIVE 文と同一の CD 句を用いた場合は省略できます。

データコミュニケーション機能と C 言語のライブラリ関数の対応

RECEIVE 文（データコミュニケーション機能）の指定と C 言語のライブラリ関数との対応を，次の表に示します。

表 4-4　RECEIVE 文（データコミュニケーション機能）の指定と C 言語のライブラリ関数との対応

FOR 句		SYNCHRONOUS MODE 句		対応する C 言語のライブラリ関数
INPUT	I-O	SYNC，または'1'	ASYNC，'0'，'△'，または省略	
○	－	－	○	dc_mcf_receive
－	○	－	○	
－	○	○	－	dc_mcf_recvsync※

（凡例）

○：指定あり

－：指定なし

注※

TP1/NET/OSAS-NIF のデータ操作言語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 PROGRAM	OUTPUT	I-O	SYNC，または'1'	ASYNC，'0'，'△'，または省略		
○	－	－	－	－	－	dc_mcf_execap
－	○	－	－	○	－	dc_mcf_send

FOR 句			SYNCHRONOUS MODE 句		BEFORE 句	対応する C 言語のライブラリ関数
OUTPUT PROGRAM	OUTPUT	I-O	SYNC，または'1'	ASYNC，'0'，'△'，または省略		
－	－	○	－	○	－	dc_mcf_reply
－	－	○	○	－	－	dc_mcf_sendsync※
－	－	○	○	－	○	dc_mcf_sendrecv

（凡例）

○：指定あり

－：指定なし

注※

TP1/NET/OSAS-NIF ではサポートしていない関数です。

CBLDCMCF('EXECAP ') ― アプリケーションプログラムの起動（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 ’EXECAP  ’.
   02 データ名B  PIC X(5).
   02 FILLER    PIC X(3).
   02 データ名C  PIC X(4) VALUE SPACE.
   02 データ名D  PIC X(4) VALUE SPACE.
   02 データ名E  PIC 9(8).
   02 データ名F  PIC 9(8).
   02 データ名G  PIC 9(9) COMP VALUE ZERO.
   02 データ名H  PIC X(4).
   02 データ名I  PIC X(4) VALUE SPACE.
   02 データ名J  PIC X(4) VALUE SPACE.
   02 データ名K  PIC X(4) VALUE SPACE.
   02 データ名L  PIC X(8) VALUE SPACE.
   02 データ名M  PIC X(4) VALUE SPACE.
   02 データ名N  PIC X(8).
   02 データ名01 PIC X(4) VALUE SPACE.
   02 データ名02 PIC 9(9) COMP VALUE ZERO.
   02 データ名03 PIC 9(9) COMP VALUE ZERO.
   02 データ名04 PIC X(1) VALUE SPACE.
   02 データ名05 PIC X(1).
   02 データ名P  PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名Q  PIC X(4) VALUE SPACE.
   02 データ名R  PIC X(8) VALUE SPACE.
   02 データ名S  PIC X(8) VALUE SPACE.
   02 データ名T  PIC X(6) VALUE SPACE.
   02 データ名U  PIC X(2) VALUE SPACE.
   02 データ名V  PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名W  PIC 9(x) COMP.
   02 データ名X  PIC X(x).
   02 データ名Y  PIC X(n).
```

機能

CBLDCMCF('EXECAP△△')は，TP1/NET/OSAS-NIF の場合，相手システムのアプリケーションプログラムを起動します。

また，システム内の通信では UAP（SPP または MHP）から，データ名 N に設定したアプリケーション名の MHP を開始させます。

MHP に渡すメッセージの一つのセグメントの最大長は，32000 バイトです。

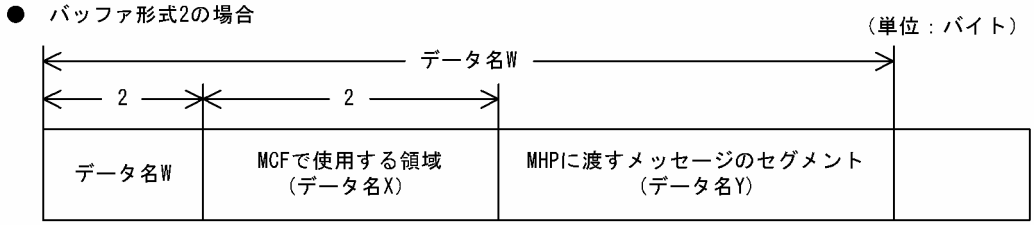
SPP から CBLDCMCF('EXECAP△△')を呼び出す場合は，SPP がトランザクションとして処理していることと，その SPP のメインプログラムで MCF 環境のオープン文を呼び出していることが前提です。

システム内のアプリケーションプログラムの起動については，マニュアル「OpenTP1 プログラム作成の手引」またはマニュアル「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。

開始させる MHP に渡すセグメントの領域（一意名 3 で示す領域）の形式を次に示します。

● バッファ形式1の場合

● バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

アプリケーションプログラムの起動を示す要求コード「VALUE 'EXECAP△△'」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

開始させる MHP に渡すセグメントが，論理メッセージの最終セグメントかどうかを設定します。次のどちらかの値を設定してください。

VALUE 'ESI△'

先頭セグメントまたは中間セグメントを渡す場合に設定します。この値を設定したCBLDCMCF('EXECAP△△')文を呼び出した場合は，そのあとに必ずデータ名 H に「VALUE 'EMI△'」を設定した CBLDCMCF('EXECAP△△')文を呼び出してください。

VALUE 'EMI△'

最終セグメントを渡す場合，および論理メッセージが単一セグメントの場合に設定します。さらに，先頭セグメントまたは中間セグメントの引き渡し後，メッセージの引き渡しの終了を連絡する場合にもこの値を設定してください。

●データ名 I，データ名 J，データ名 K，データ名 L，データ名 M

空白を設定します。

●データ名 N

起動する MHP のアプリケーション名を設定します。アプリケーション名は最大 8 バイトの長さです。8 バイトに満たない場合，アプリケーション名の後ろを空白で埋めてください。

●データ名 O1

空白を設定します。

●データ名 O2，データ名 O3

0 を設定します。

●データ名 O4

空白を設定します。

●データ名 O5

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 P

MCF で使用する領域です。

●データ名 Q，データ名 R，データ名 S，データ名 T，データ名 U

空白を設定します。

●データ名 V

MCF で使用する領域です。

●データ名 W

【バッファ形式 1 の場合】 PIC 9(9)

起動する MHP に渡すセグメントの長さを設定します。先頭セグメントの引き渡し後，メッセージの引き渡しの終了を連絡する場合で，セグメントの内容がないときは，0 を設定してください。

【バッファ形式 2 の場合】 PIC 9(4)

起動する MHP に渡すセグメントの長さ＋ 4 を設定します。

先頭セグメントの引き渡し後，メッセージの引き渡しの終了を連絡する場合で，セグメントの内容がないときは，4 を設定してください。

●データ名 X

【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 Y

起動する MHP に渡すセグメント内容を設定します。

先頭セグメントの引き渡し後，メッセージの引き渡しの終了を連絡する場合で，セグメントの内容がないときも必ず設定してください。

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入出力処理時に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	バッファ形式 1 の場合はデータ名 W 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 W 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，データ名 N に設定した MHP を起動できません。

ステータスコード	意味
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	MHP を開始しようとしましたが，開始しようとした MHP の管理テーブルを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('EXECAP△△')を呼び出しています。
	＜ SPP の実行でリターンした場合＞ トランザクションの処理でない SPP から，CBLDCMCF('EXECAP△△')を呼び出しています。
72001	データ名 N に設定したアプリケーション名は，MCF で定義されていません。
	データ名 N に設定したアプリケーション名が間違っています。
	MCF マネジャ定義の通信サービス定義（mcfmcname）に，アプリケーション起動プロセス名または MCF 通信プロセス名を指定していません。
	アプリケーション起動プロセス，または MCF 通信プロセスに対応するアプリケーション環境定義（mcfaenv -p）に，アプリケーション起動プロセス識別子を指定していません。
	アプリケーション環境定義（mcfaenv -p）で指定したアプリケーション起動プロセス識別子と，アプリケーション起動プロセス，または MCF 通信プロセスの MCF 環境定義（mcftenv -s）で指定する識別子が一致していません。
	＜論理端末名称を指定してアプリケーションを起動する場合＞ • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の lname オペランドに論理端末を指定していません。 • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の lname オペランドに指定した論理端末を，MCF 通信プロセスの論理端末定義（mcftalcle）に定義していません。 • 起動先アプリケーションのアプリケーション属性定義に指定した論理端末が，request 型論理端末または send 型論理端末ではありません。 • 起動先アプリケーションのアプリケーション属性定義で指定した論理端末は，アプリケーション起動を使えません。
	＜コネクション ID を指定してアプリケーションを起動する場合＞ • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の cname オペランドにコネクション ID を指定していません。 • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の cname オペランドに指定したコネクション ID を，MCF 通信プロセスのコネクション定義（mcftalccn）に定義していません。 • MCF 通信プロセスの論理端末定義（mcftalcle）に，request 型論理端末を指定していません。
	＜ SPP からアプリケーションを起動する場合＞ • アプリケーション起動プロセス識別子を起動元の UAP のユーザサービス定義，またはユーザサービスデフォルト定義の mcf_psv_id オペランドに指定していません。 • 起動元の UAP のユーザサービス定義，またはユーザサービスデフォルト定義の mcf_psv_id オペランドに指定しているアプリケーション起動プロセス識別子が，アプリケーション起動プロセス，または MCF

ステータスコード	意味
72001	通信プロセスの MCF 環境定義（mcftenv -s），およびアプリケーション環境定義（mcfaenv -p）で指定しているアプリケーション起動プロセス識別子と一致していません。 • 起動元の UAP のユーザサービス定義，またはユーザサービスデフォルト定義の mcf_mgrid オペランドに指定している MCF マネジャ識別子が，アプリケーション起動プロセスが属している MCF マネジャの識別子と一致していません。
72005	＜データ名 H で VALUE 'ESI△'を設定した場合＞ バッファ形式 1 の場合はデータ名 W 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 W に 0 から 4 バイト，またはマイナス値を設定しています。
72007	CBLDCMCF('REPLY△△△')をすでに呼び出した応答型（type=ans）の MHP から，応答型の MHP を CBLDCMCF('EXECAP△△')で起動させています。
	CBLDCMCF('REPLY△△△')をすでに呼び出した継続問い合わせ応答型（type=cont）の MHP から，継続問い合わせ応答型の MHP を CBLDCMCF('EXECAP△△')で起動させています。
72009	応答型（type=cont）の MHP から，CBLDCMCF('EXECAP△△')で応答型の MHP を 2 回以上起動させています。
	継続問い合わせ応答型（type=cont）の MHP から，CBLDCMCF('EXECAP△△')で継続問い合わせ応答型の MHP を 2 回以上起動させています。
72011	継続問い合わせ応答型（type=cont）でない MHP から，CBLDCMCF('EXECAP△△')で継続問い合わせ応答型の MHP を起動させています。
72016	データ名 O1，データ名 O2，データ名 O3 に設定した値が間違っています。
	データ名 P に設定した値が間違っています。
	データ名 V に設定した値が間違っています。
72024	データ名 Q 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	＜データ名 H で VALUE 'EMI△'を設定した場合＞ • バッファ形式 1 の場合はデータ名 W 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 W に 0 から 4 バイト，またはマイナス値を設定しています。 • データ名 H に VALUE 'ESI△'を設定した CBLDCMCF('EXECAP△△')を呼び出さないで，メッセージの引き渡しの終了を連絡しています。
77001	起動しようとするアプリケーションに対応する論理端末は，現在仕掛り中で使用できません。または，使用できる論理端末がありません。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('RECEIVE ') － メッセージの受信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A  PIC X(8) VALUE ’RECEIVE ’.
   02 データ名B  PIC X(5).
   02 FILLER    PIC X(3).
   02 データ名C  PIC X(4).
   02 データ名D  PIC X(4) VALUE SPACE.
   02 データ名E  PIC 9(8).
   02 データ名F  PIC 9(8).
   02 データ名G  PIC 9(9) COMP.
   02 データ名H  PIC X(4) VALUE SPACE.
   02 データ名I  PIC X(4) VALUE SPACE.
   02 データ名J  PIC X(4) VALUE SPACE.
   02 データ名K  PIC X(4) VALUE SPACE.
   02 データ名L  PIC X(8) VALUE SPACE.
   02 データ名M1 PIC X(4) VALUE SPACE.
   02 データ名M2 PIC X(8) VALUE SPACE.
   02 データ名M3 PIC X(4) VALUE SPACE.
   02 データ名M4 PIC 9(9) COMP VALUE ZERO.
   02 データ名M5 PIC 9(9) COMP VALUE ZERO.
   02 データ名M6 PIC X(1) VALUE SPACE.
   02 データ名M7 PIC X(1).
   02 データ名N  PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O  PIC X(4) VALUE SPACE.
   02 データ名P  PIC X(8).
   02 データ名Q  PIC X(8).
   02 データ名R  PIC X(8) VALUE SPACE.
   02 データ名T  PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名U  PIC 9(x) COMP.
   02 データ名V  PIC X(x).
   02 データ名W  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，65452 バイトです。

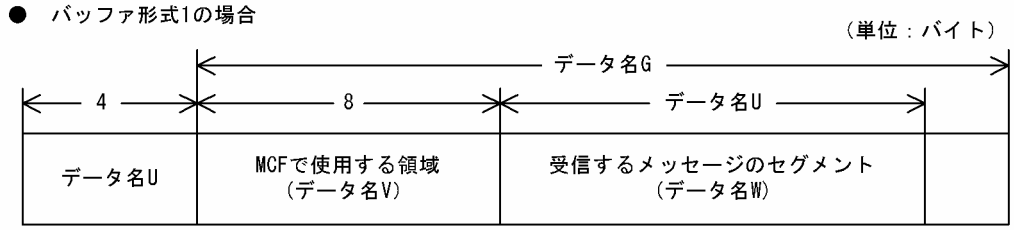
CBLDCMCF('RECEIVE△')で受信できるメッセージの種類を次に示します。

- 相手システムから送信されたメッセージ

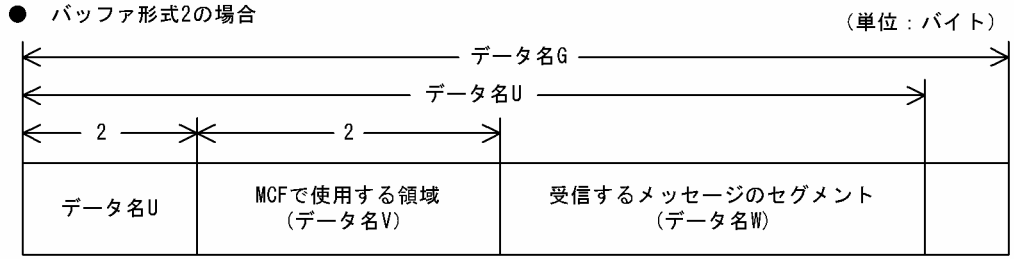
- MCF イベント
- アプリケーション起動で渡されたメッセージ

セグメントを受信する領域（一意名 3 で示す領域）の形式を次に示します。

● バッファ形式1の場合



● バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

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

●データ名 C

受信するセグメントを設定します。次のどちらかの値を設定してください。

VALUE 'FRST'

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

VALUE 'SEG△'

中間セグメントまたは最終セグメントを受信する場合に設定します。

●データ名 D

空白を設定します。

●データ名 G

セグメントを受信する領域の長さを設定します。

●データ名 H，データ名 I，データ名 J，データ名 K，データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O

空白を設定します。

●データ名 P

中間セグメントまたは最終セグメントを受信する場合は，入力元の論理端末名称を設定します。先頭セグメントの受信時に返された論理端末名称を設定してください。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない場合，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 から値が返ります。

●データ名 Q

MCF で使用する領域です。

●データ名 R

空白を設定します。

●データ名 T

MCF で使用する領域です。

●データ名 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 E

メッセージを受信した日付が YYYYMMDD（YYYY：西暦年　MM：月　DD：日）の形式で返されます。

●データ名 F

メッセージを受信した時刻が HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で返されます。

●データ名 P

先頭セグメントまたは単一セグメントを受信する場合，入力元の論理端末名称が返されます。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない場合，論理端末名称の後ろが空白で埋められます。

中間セグメントまたは最終セグメントを受信する場合，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 W

受信したセグメントが返されます。

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△')でメッセージをすべて受信しました。

ステータスコード	意味
71001	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，中間セグメントおよび最終セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出しています。先頭セグメントを受信する場合は，データ名 C に「VALUE 'FRST'」を設定して CBLDCMCF('RECEIVE△')を呼び出してください。 • ステータスコード 71001 が返されたあとに，再び CBLDCMCF('RECEIVE△')を呼び出しています。
	＜ SPP の実行でリターンした場合＞ SPP では CBLDCMCF('RECEIVE△')を呼び出せません。
72001	データ名 P に設定した論理端末名称が間違っています。
72013	データ名 G の指定値を超えるセグメントを受信しました。データ名 G の指定値を超えた部分は切り捨てられました。
	＜バッファ形式 2 の場合＞ 32763 バイトを超えるセグメントを受信しました。32763 バイトを超えた部分は切り捨てられました。
72016	データ名 D に設定した値が間違っています。
	データ名 N またはデータ名 T に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72025	データ名 C に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72036	データ名 G の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('RECVSYNC') － 同期型メッセージの後続セグメント受信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A  PIC X(8) VALUE ’RECVSYNC’.
   02 データ名B  PIC X(5).
   02 FILLER    PIC X(3).
   02 データ名C  PIC X(4) VALUE ’SEG ’.
   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) VALUE SPACE.
   02 データ名R  PIC X(8) VALUE SPACE.
   02 データ名T  PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名U  PIC 9(x) COMP.
   02 データ名V  PIC X(x).
   02 データ名Y  PIC X(n).
```

機能

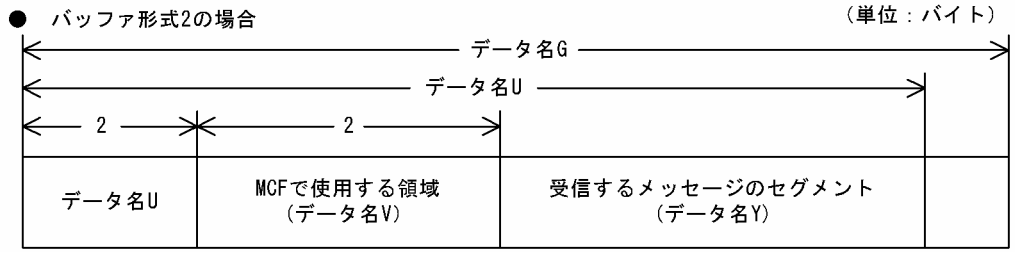
相手システムから届いた論理メッセージ（同期型で受信）のうち，先頭セグメント以外の後続する一つのセグメントを受信します。先頭セグメントを受信する CBLDCMCF('SENDRECV')のあとに，後続するセグメントの数だけ CBLDCMCF('RECVSYNC')を発行することによって，一つの論理メッセージを受信できます。

受信できるメッセージの一つのセグメントの最大長は，65452 バイトです。

セグメントを受信する領域（一意名 3 で示す領域）の形式を次に示します。

● バッファ形式1の場合

● バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

同期型メッセージの受信を示す要求コード「VALUE 'RECVSYNC'」を設定します。

●データ名 C

中間セグメントまたは最終セグメントの受信を示す「VALUE 'SEG△'」を設定します。

●データ名 D

空白を設定します。

●データ名 G

セグメントを受信する領域の長さを設定します。

●データ名 H，データ名 I，データ名 J，データ名 K，データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O

空白を設定します。

●データ名 P

入力元の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない場合，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 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 E

メッセージを受信した日付が YYYYMMDD（YYYY：西暦年　MM：月　DD：日）の形式で返されます。

●データ名 F

メッセージを受信した時刻が HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で返されます。

●データ名 U

【バッファ形式 1 の場合】 PIC 9(9)

受信したセグメントの長さが返されます。

【バッファ形式 2 の場合】 PIC X(2)

受信したセグメントの長さ＋ 4 が返されます。

●データ名 Y

受信した論理メッセージのセグメントの内容が返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	メッセージの最終セグメントを受信したあとで，その次のセグメントを受信する CBLDCMCF('RECVSYNC') を呼び出しています。直前に呼び出した CBLDCMCF('RECVSYNC')でメッセージはすべて受信しました。 このステータスコードが返されたあとに，再び次のセグメントを受信する CBLDCMCF('RECVSYNC')を呼び出した場合は，ステータスコード 72000 が返されます。
71002	MCF が終了処理中のため，受け付けられません。
71108	メッセージ送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 CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('RECVSYNC')を呼び出しました。
	ステータスコード 71001 が返されたあとに，再び CBLDCMCF('RECVSYNC')を呼び出しています。
72001	データ名 P に設定した論理端末名称が間違っています。
	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('RECVSYNC')を呼び出せない論理端末を設定しています。
72013	データ名 G の指定値を超えるセグメントを受信しました。データ名 G の指定値を超えた部分は切り捨てられました。
	＜バッファ形式 2 の場合＞ 32763 バイトを超えるセグメントを受信しました。32763 バイトを超えた部分は切り捨てられました。
72016	データ名 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。
	データ名 Q に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72025	データ名 C に設定した値が間違っています。

ステータスコード	意味
72028	データ名 A に設定した値が間違っています。
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。
73018	データ名 M5 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。

CBLDCMCF('REPLY ') － 応答メッセージの送信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A  PIC X(8) VALUE ’REPLY   ’.
   02 データ名B  PIC X(5).
   02 FILLER    PIC X(3).
   02 データ名C  PIC X(4) VALUE SPACE.
   02 データ名D  PIC X(4) VALUE SPACE.
   02 データ名E  PIC 9(8).
   02 データ名F  PIC 9(8).
   02 データ名G  PIC 9(9) COMP VALUE ZERO.
   02 データ名H  PIC X(4).
   02 データ名I  PIC X(4) VALUE SPACE.
   02 データ名J  PIC X(4) VALUE SPACE.
   02 データ名K  PIC X(4) VALUE SPACE.
   02 データ名L  PIC X(8) VALUE SPACE.
   02 データ名M1 PIC X(4) VALUE SPACE.
   02 データ名M2 PIC X(8) VALUE SPACE.
   02 データ名M3 PIC X(4) VALUE SPACE.
   02 データ名M4 PIC 9(9) COMP VALUE ZERO.
   02 データ名M5 PIC 9(9) COMP VALUE ZERO.
   02 データ名M6 PIC X(1) VALUE SPACE.
   02 データ名M7 PIC X(1).
   02 データ名N  PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O  PIC X(4) VALUE SPACE.
   02 データ名P  PIC X(8) VALUE SPACE.
   02 データ名Q  PIC X(8) VALUE SPACE.
   02 データ名R  PIC X(8) VALUE SPACE.
   02 データ名T  PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名U  PIC 9(x) COMP.
   02 データ名V  PIC X(x).
   02 データ名W  PIC X(n).
```

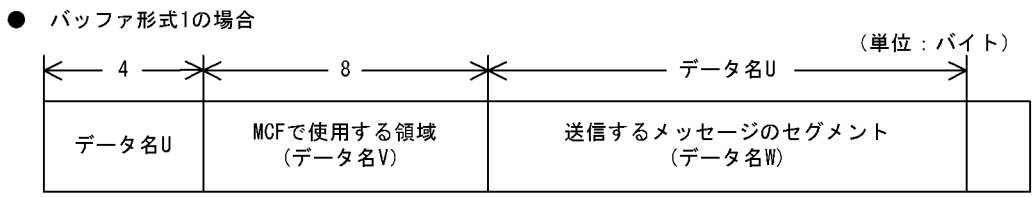
機能

メッセージを入力した論理端末に送信する応答メッセージのうち，一つのセグメントを送信要求します。必要なセグメントの数だけ CBLDCMCF('REPLY△△△')を発行することによって，一つの論理メッセージを送信できます。

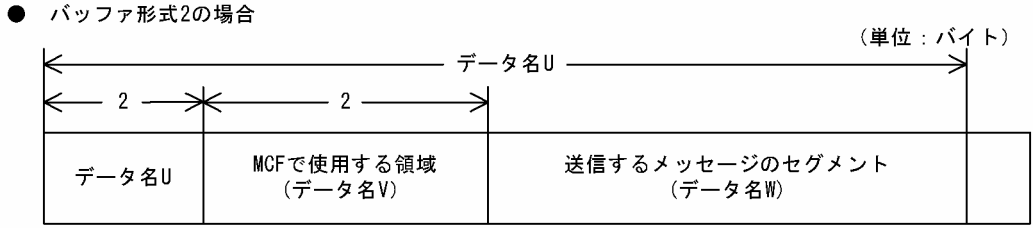
送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

セグメントを送信する領域（一意名 3 で示す領域）の形式を次に示します。

● バッファ形式1の場合



● バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

応答メッセージの送信を示す要求コード「VALUE 'REPLY△△△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E，データ名 F

MCF で使用する領域です。

●データ名 G

0 を設定します。

●データ名 H

送信するセグメントを設定します。次のどちらかの値を設定してください。

VALUE 'ESI△'

先頭セグメントまたは中間セグメントを送信する場合に設定します。

VALUE 'EMI△'

最終セグメントを送信する場合や，論理メッセージが単一セグメントの場合に設定します。メッセージの送信の終了を連絡するために，最後は必ずこの値を設定してください。

●データ名 I，データ名 J

空白を設定します。

●データ名 K

出力通番を付けるかどうかを設定します。

VALUE 'SEQ△'

出力通番を付ける場合に設定します。

ただし，非応答型のアプリケーションの場合は，VALUE 'SEQ△'を指定しても，MCF は応答メッセージに出力通番を付けません。

VALUE 'NSEQ'

出力通番を付けない場合に設定します。

空白

省略されたものとして，「VALUE 'NSEQ'」（出力通番を付けない）が設定されます。

●データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O，データ名 P，データ名 Q，データ名 R

空白を設定します。

●データ名 T

MCF で使用する領域です。

●データ名 U

【バッファ形式 1 の場合】 PIC 9(9)

送信するセグメントの長さを設定します。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときは，0 を設定してください。

【バッファ形式 2 の場合】 PIC 9(4)

送信するセグメントの長さ＋ 4 を設定します。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときは，4 を設定してください。

●データ名 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

●データ名 W

送信するセグメントの内容を設定します。一つのセグメントで 32000 バイトまで送信できます。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときも必ず設定してください。

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を確保できませんでした。

ステータスコード	意味
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，CBLDCMCF('REPLY△△△')を呼び出しました。 • 非応答型のアプリケーションからの問い合わせ応答をしない（UAP 共通定義（mcfmuap -c）の noansreply オペランドに no を指定）場合に，非応答型のアプリケーションから CBLDCMCF('REPLY△△△')を呼び出しました。
	＜SPP の実行でリターンした場合＞ SPP では CBLDCMCF('REPLY△△△')を呼び出せません。
72001	入力元論理端末は，応答メッセージの送信をサポートしていません。
72005	＜データ名 H で VALUE 'ESI△'を設定した場合＞ 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。
72008	最終セグメントを送信する CBLDCMCF('REPLY△△△')を呼び出したあとで，再び CBLDCMCF('REPLY△△△')を呼び出しています。
	応答型のアプリケーションから CBLDCMCF('EXECAP△△')を呼び出して応答型のアプリケーションを起動させたあとに，CBLDCMCF('REPLY△△△')を呼び出しています。
72016	データ名 N またはデータ名 T に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72017	データ名 K に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72020	データ名 I に設定した値が間違っています。
72026	データ名 H に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72041	＜データ名 H で VALUE 'EMI△'を設定した場合＞ • 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。 • データ名 H に VALUE 'ESI△'を設定した CBLDCMCF('REPLY△△△')を呼び出さないで，メッセージの送信の終了を連絡しています。
72045	継続問い合わせ応答型のアプリケーションはサポートしていません。
72047	継続問い合わせ応答型のアプリケーションはサポートしていません。
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（一般送信，優先送信）

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（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設定してください。

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

メッセージの送信を示す「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

再送する対象の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'」（一般の送信メッセージを対象）が設定されます。

●データ名 V

再送する対象の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，データ名 V に設定した値は無効になります。

空白

省略されたものとして，最終出力通番を持つメッセージを再送対象としないことが設定されます。

●データ名 X

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
70904	出力通番使用論理端末数（MCF マネジャ共通定義（mcfmcomn）の-n オプション）を省略，または 0 を指定しています。
	データ名 T，データ名 U，またはデータ名 V に設定した値が間違っています。
	再送するメッセージは次に示す理由によって，再送できるメッセージの条件を満たしていません。 • 再送対象とするメッセージの送信時に出力通番を付けていません。 • 出力メッセージの割り当て先にメモリキューを使用しています（論理端末定義（mcftalcle -k）の quekind オペランドに 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）。

ステータスコード	意味
70904	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	メッセージを再送しようとしましたが，再送先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('RESEND△△')を呼び出しています。 • 非トランザクション属性の MHP から CBLDCMCF('RECEIVE△')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△△')を呼び出しまています。
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。
	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，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，セグメントの最大長よりも大きな値を設定しておいてください。

-l オプション

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

CBLDCMCF('SEND ') － メッセージの送信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A  PIC X(8) VALUE ’SEND  ’.
   02 データ名B  PIC X(5).
   02 FILLER    PIC X(3).
   02 データ名C  PIC X(4) VALUE SPACE.
   02 データ名D  PIC X(4) VALUE SPACE.
   02 データ名E  PIC 9(8).
   02 データ名F  PIC 9(8).
   02 データ名G  PIC 9(9) COMP VALUE ZERO.
   02 データ名H  PIC X(4).
   02 データ名I  PIC X(4) VALUE SPACE.
   02 データ名J  PIC X(4).
   02 データ名K  PIC X(4).
   02 データ名L  PIC X(8) VALUE SPACE.
   02 データ名M1 PIC X(4) VALUE SPACE.
   02 データ名M2 PIC X(8) VALUE SPACE.
   02 データ名M3 PIC X(4) VALUE SPACE.
   02 データ名M4 PIC 9(9) COMP VALUE ZERO.
   02 データ名M5 PIC 9(9) COMP VALUE ZERO.
   02 データ名M6 PIC X(1) VALUE SPACE.
   02 データ名M7 PIC X(1).
   02 データ名N  PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O  PIC X(4) VALUE ’OUT ’.
   02 データ名P  PIC X(8).
   02 データ名Q  PIC X(8) VALUE SPACE.
   02 データ名R  PIC X(8) VALUE SPACE.
   02 データ名T  PIC X(28) VALUE LOW-VALUE.
01 一意名3.
 02 データ名U  PIC 9(x) COMP.
 02 データ名V  PIC X(x).
 02 データ名W  PIC X(n).
```

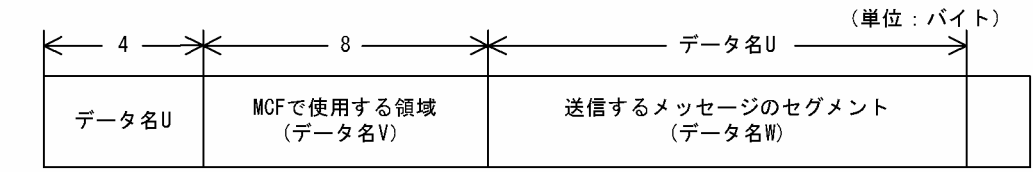
機能

MCF で管理する論理端末に送信するメッセージのうち，一つのセグメントを送信要求します。必要なセグメントの数だけ，CBLDCMCF('SEND△△△△')を発行することによって，一つの論理メッセージを送信できます。

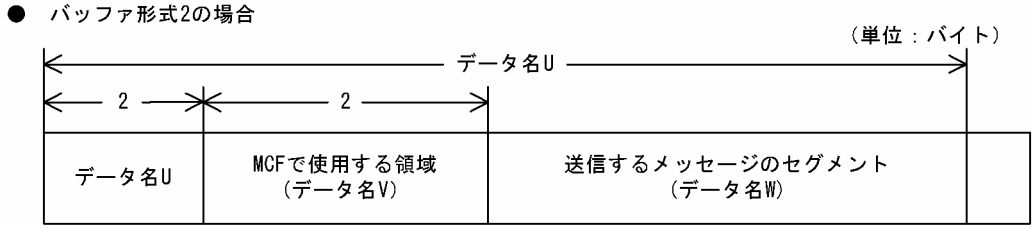
送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

送信するセグメントの領域（一意名 3 で示す領域）の形式を次に示します。

● バッファ形式1の場合



● バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

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

送信するセグメントを設定します。次のどちらかの値を設定してください。

VALUE 'ESI△'

先頭セグメントまたは中間セグメントを送信する場合に設定します。

VALUE 'EMI△'

最終セグメントを送信する場合や，論理メッセージが単一セグメントの場合に設定します。メッセージの送信の終了を連絡するために，最後は必ずこの値を設定してください。

●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。次のどれかを設定してください。

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。次のどれかを設定してください。

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'」（出力通番を付けない）が設定されます。

●データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O

一方送信を示す「VALUE 'OUT△'」を設定します。

●データ名 P

出力先の論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない場合，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を設定します。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときは，0 を設定してください。

【バッファ形式 2 の場合】 PIC 9(4)

送信するセグメントの長さ＋ 4 を設定します。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときは，4 を設定してください。

●データ名 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

●データ名 W

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

先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときも，必ず設定してください。

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を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。
72005	＜データ名 H で VALUE 'ESI△'を設定した場合＞ 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。
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，データ名 T に設定した値が間違っています。
72017	データ名 K に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72020	データ名 I に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72026	データ名 H に設定した値が間違っています。

ステータスコード	意味
72028	データ名 A に設定した値が間違っています。
72041	＜データ名 H で VALUE 'EMI△'を設定した場合＞ • 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。 • データ名 H に VALUE 'ESI△'を設定した CBLDCMCF('SEND△△△△')を呼び出さないで，メッセージの送信の終了を連絡しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('SENDRECV') － 同期型メッセージの送受信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3 一意名4
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A  PIC X(8) VALUE ’SENDRECV’.
   02 データ名B  PIC X(5).
   02 FILLER    PIC X(3).
   02 データ名C  PIC X(4) VALUE SPACE.
   02 データ名D  PIC X(4) VALUE SPACE.
   02 データ名E  PIC 9(8).
   02 データ名F  PIC 9(8).
   02 データ名G  PIC 9(9) COMP.
   02 データ名H  PIC X(4).
   02 データ名I  PIC X(4) VALUE SPACE.
   02 データ名J  PIC X(4) VALUE SPACE.
   02 データ名K  PIC X(4) VALUE SPACE.
   02 データ名L  PIC X(8) VALUE SPACE.
   02 データ名M1 PIC X(4) VALUE SPACE.
   02 データ名M2 PIC X(8) VALUE SPACE.
   02 データ名M3 PIC X(4) VALUE SPACE.
   02 データ名M4 PIC 9(9) COMP VALUE ZERO.
   02 データ名M5 PIC 9(9) COMP 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 ’IO  ’.
   02 データ名P  PIC X(8).
   02 データ名Q  PIC X(8) VALUE SPACE.
   02 データ名R  PIC X(8) VALUE SPACE.
   02 データ名T  PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名U  PIC 9(x) COMP.
   02 データ名V  PIC X(x).
   02 データ名W  PIC X(n).
01 一意名4.
   02 データ名X  PIC 9(x) COMP.
   02 データ名Y1 PIC X(x) VALUE SPACE.
   02 データ名Y2 PIC X(1).
   02 データ名Z  PIC X(n).
```

機能

同期型でメッセージを送信したあと，同期型でメッセージを受信します。

相手システムへ送る論理メッセージのうち，一つのセグメントを送信します。必要なセグメントの数だけCBLDCMCF('SENDRECV')を発行することによって，一つの論理メッセージを送信します。

メッセージの最終セグメントを送信すると，CBLDCMCF('SENDRECV')は相手システムからの応答を待ちます。応答が届くと，そのメッセージの先頭セグメントを受信します。

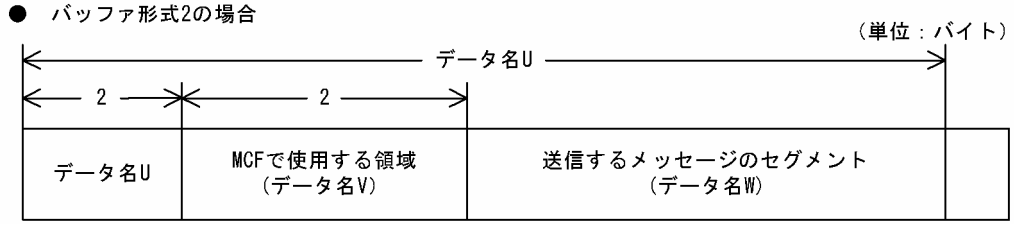
中間セグメントおよび最終セグメントを受信する場合はCBLDCMCF('RECVSYNC')を発行してください。

受信できるメッセージの一つのセグメントの最大長は，65452バイトです。また，送信できるメッセージの一つのセグメントの最大長は，32000バイトです。

セグメントを送信する領域（一意名3で示す領域）の形式を次に示します。

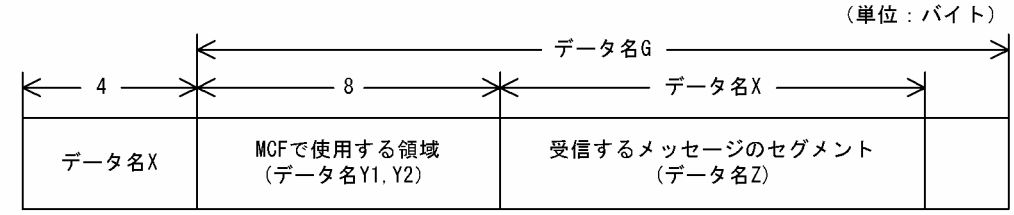
● バッファ形式1の場合

● バッファ形式2の場合

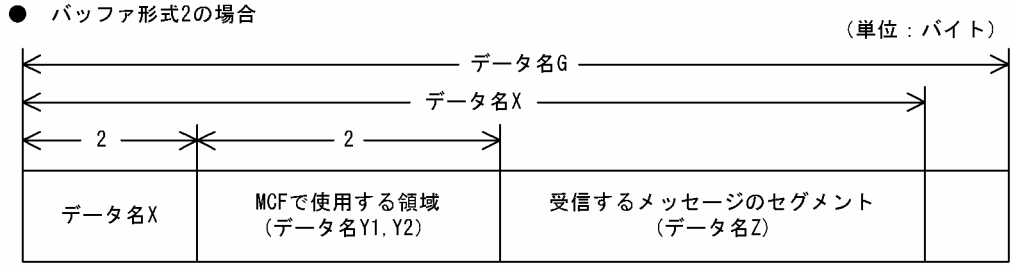


セグメントを受信する領域（一意名4で示す領域）の形式を次に示します。

● バッファ形式1の場合



● バッファ形式2の場合



UAP で値を設定するデータ領域

●データ名 A

同期型メッセージの送受信を示す要求コード「VALUE 'SENDRECV'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 G

セグメントを受信する領域の長さを設定します。

●データ名 H

送信するセグメントを設定します。次のどちらかの値を設定してください。

VALUE 'ESI△'

先頭セグメントまたは中間セグメントを送信する場合に設定します。

VALUE 'EMI△'

最終セグメントを送信する場合や，論理メッセージが単一セグメントの場合に設定します。メッセージの送信の終了を連絡するために，最後は必ずこの値を設定してください。この値を設定してCBLDCMCF('SENDRECV')を発行すると，相手システムからの応答を待ちます。

●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，データ名 M5

0 を設定します。

●データ名 M6

空白を設定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

省略されたものとして，「VALUE '1'」（バッファ形式 1）が設定されます。

●データ名 N

MCF で使用する領域です。

●データ名 O

同期型メッセージの送受信を示す「VALUE 'IO△△'」を設定します。

●データ名 P

メッセージを出力して応答を入力する論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない場合，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を設定します。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときは，0 を設定してください。

【バッファ形式 2 の場合】 PIC 9(4)

送信するセグメントの長さ＋ 4 を設定します。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときは，4 を設定してください。

●データ名 V

【バッファ形式 1 の場合】 PIC X(8)

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

●データ名 W

送信するセグメントの内容を設定します。一つのセグメントで 32000 バイトまで送信できます。先頭セグメントの送信後，メッセージの送信の終了を連絡する場合で，セグメントの内容がないときも，必ず設定してください。

●データ名 Y1

【バッファ形式 1 の場合】 PIC X(7)

【バッファ形式 2 の場合】 PIC X(1)

空白を設定します。

●データ名 Y2

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 E

メッセージを受信した日付が YYYYMMDD（YYYY：西暦年　MM：月　DD：日）の形式で返されます。

●データ名 F

メッセージを受信した時刻が HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で返されます。

●データ名 X

【バッファ形式 1 の場合】 PIC 9(9)

受信したセグメントの長さが返されます。

【バッファ形式 2 の場合】 PIC X(2)

受信したセグメントの長さ＋ 4 が返されます。

●データ名 Z

受信した論理メッセージの先頭セグメントの内容が返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71002	メッセージキューへの入出力処理時に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	バッファ形式 1 の場合はデータ名 U に 32000 バイトを超える値を設定しています。バッファ形式 2 の場合はデータ名 U に 32004 バイトを超える値を設定しています。
	MCF が終了処理中のため，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	メッセージ送信しようとしましたが，送信先に管理テーブルが確保できませんでした。

ステータスコード	意味
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，CBLDCMCF('SENDRECV')を呼び出しています。
72001	データ名 P に設定した論理端末名称が間違っています。
	データ名 P に設定した論理端末名称は，定義されていません。
	CBLDCMCF('SENDRECV')を呼び出せない論理端末を設定しています。
72005	＜データ名 H で VALUE 'ESI△'を設定した場合＞ 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。
72013	データ名 G の指定値を超えるセグメントを受信しました。データ名 G の指定値を超えた部分は切り捨てられました。
	＜バッファ形式 2 の場合＞ 32763 バイトを超えるセグメントを受信しました。32763 バイトを超えた部分は切り捨てられました。
72016	データ名 N またはデータ名 T に設定した値が間違っています。
	データ名 M4 に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72026	データ名 H に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72036	データ名 G の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
72041	＜データ名 H で VALUE 'EMI△'を設定した場合＞ • 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。 • データ名 H に VALUE 'ESI△'を設定した CBLDCMCF('SENDRECV')を呼び出さないで，メッセージの送信の終了を連絡しています。
73001	同期型送受信監視時間（mcfmuap -t sndrcvtim）に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
	CBLDCMCF('SENDRECV')を呼び出したあとで，論理端末の閉塞またはコネクションの解放が発生しました。
	出力先の論理端末で MCF 通信プロセスの内部障害が発生しました。
73002	論理端末が閉塞されています。
73004	メッセージの送信処理中にタイムアウトが発生しました。

ステータスコード	意味
73005	メッセージの受信処理中にタイムアウトが発生しました。
73008	運用コマンドまたは障害による論理端末閉塞処理中に，CBLDCMCF('SENDRECV')を呼び出しました。
73009	論理端末閉塞解除処理中に，CBLDCMCF('SENDRECV')を呼び出しました。
73010	入力メッセージ編集 UOC または出力メッセージ編集 UOC で障害が発生しました。
73015	出力先の論理端末は，ほかの UAP で仕掛り中です。
73018	データ名 M5 に設定した値が間違っています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('TACTCN ') － コネクションの確立（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TACTCN ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D1  PIC X(1)   VALUE SPACE.
   02 データ名D2  PIC X(1)   VALUE SPACE.
   02 データ名D3  PIC X(26)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F1  PIC X(8).
   02 データ名F2  PIC X(56)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.
```

機能

コネクションを確立します。

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

CBLDCMCF('TACTCN△△')の呼び出し後にコネクションに関する何らかの処理をする場合は，CBLDCMCF('TLSCN△△△')を用いてコネクションの状態を確認してください。

UAP で値を設定するデータ領域

●データ名 A

コネクション確立を示す要求コード「VALUE 'TACTCN△△'」を設定します。

●データ名 C

確立するコネクションの指定方法を設定します。

VALUE 'LE△△'

確立するコネクションを論理端末名称で指定するときに設定します。

VALUE 'CN△△'

確立するコネクションをコネクション ID で指定するときに設定します。

空白

省略されたものとして，「VALUE 'LE△△'」（論理端末名称を指定）が設定されます。

●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

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

論理端末名称を使用してコネクションの確立を要求する場合は，無効となります。

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

注※

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

●データ名 F1

確立するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は最大 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，またはコネクション ID 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('TACTCN△△')が受け付けられません。
71002	MCF が終了処理中のため，CBLDCMCF('TACTCN△△')が受け付けられません。
71004	CBLDCMCF('TACTCN△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71007	指定されたコネクション ID は登録されていません。
71008	指定された論理端末名称は登録されていません。
71009	CBLDCMCF('TACTCN△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスにコネクションの確立を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
71011	コネクションが削除されているため，CBLDCMCF('TACTCN△△')が受け付けられません。
71014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループを指定しています。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に'LE△△'，'CN△△'，または空白以外の値が設定されています。
72059	データ名 D1，データ名 D2，またはデータ名 D3 に空白でない値が設定されています。
72061	データ名 E に 0 未満，または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。

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/OSAS-NIF 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。

注※

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

●データ名 F1

閉塞解除する論理端末の名称を設定します。論理端末名称は最大 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。

ステータスコード	意味
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。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D1，データ名 D2，またはデータ名 D3 に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。

CBLDCMCF('TDCTCN ') − コネクションの解放（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TDCTCN ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D1  PIC X(1).
   02 データ名D2  PIC X(1)   VALUE SPACE.
   02 データ名D3  PIC X(26)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F1  PIC X(8).
   02 データ名F2  PIC X(56)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.
```

機能

コネクションを解放します。

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

CBLDCMCF('TDCTCN△△')の呼び出し後にコネクションに関する何らかの処理をする場合は，CBLDCMCF('TLSCN△△△')を用いてコネクションの状態を確認してください。

UAP で値を設定するデータ領域

●データ名 A

コネクション解放を示す要求コード「VALUE 'TDCTCN△△'」を設定します。

●データ名 C

解放するコネクションの指定方法を設定します。

VALUE 'LE△△'

解放するコネクションを論理端末名称で指定するときに設定します。

VALUE 'CN△△'

解放するコネクションをコネクション ID で指定するときに設定します。

空白

省略されたものとして，「VALUE 'LE△△'」（論理端末名称を指定）が設定されます。

●データ名 D1

解放するコネクションの指定方法を設定します。

VALUE '1'

コネクションを強制的に解放します。

VALUE '0'

コネクションを正常に解放します。

空白

省略されたものとして，'0'（正常解放）が設定されます。

●データ名 D2，データ名 D3

空白を設定します。

●データ名 E

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

論理端末名称を使用してコネクションの解放を要求する場合は，無効となります。

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

注※

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

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

●データ名 F1

解放するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は最大 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，またはコネクション ID 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('TDCTCN△△')が受け付けられません。
71002	MCF が終了処理中のため，CBLDCMCF('TDCTCN△△')が受け付けられません。
71004	CBLDCMCF('TDCTCN△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。
71007	指定されたコネクション ID は登録されていません。
71008	指定された論理端末名称は登録されていません。
71009	CBLDCMCF('TDCTCN△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスにコネクションの解放を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
71011	コネクションが削除されているため，CBLDCMCF('TDCTCN△△')が受け付けられません。
71014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループ名を指定しています。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に'LE△△'，'CN△△'，または空白以外の値が設定されています。
72059	データ名 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。

ステータスコード	意味
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。
72075	データ名 D1 に 1，0，または空白以外の値が設定されています。

CBLDCMCF('TDCTLE ') － 論理端末の閉塞（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL　'CBLDCMCF'　USING　一意名1　一意名2
```

DATA DIVISION の指定

```
01　一意名1.
    02　データ名A　　PIC X(8)　　VALUE 'TDCTLE　'.
    02　データ名B　　PIC X(5).
    02　FILLER　　PIC X(3).
    02　データ名C　　PIC X(4)　　VALUE SPACE.
    02　データ名D1　PIC X(1)　　VALUE SPACE.
    02　データ名D2　PIC X(1)　　VALUE SPACE.
    02　データ名D3　PIC X(26)　VALUE SPACE.
    02　データ名E　　PIC 9(9)　　COMP.
    02　データ名F1　PIC X(8).
    02　データ名F2　PIC X(56)　VALUE SPACE.
    02　データ名G　　PIC X(8)　　VALUE SPACE.
    02　データ名H　　PIC X(8)　　VALUE SPACE.
    02　データ名I　　PIC X(144) VALUE SPACE.
    02　データ名J　　PIC X(184) VALUE SPACE.
    02　データ名K　　PIC 9(9)　　COMP VALUE ZERO.

01　一意名2.
    02　データ名L　　PIC 9(9)　　COMP VALUE ZERO.
```

機能

論理端末を閉塞します。

なお，CBLDCMCF('TDCTLE△△')の正常終了は，論理端末の閉塞要求を TP1/NET/OSAS-NIF 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。

注※

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

●データ名 F1

閉塞する論理端末の名称を設定します。論理端末名称は最大 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。

ステータスコード	意味
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。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D2，またはデータ名 D3 に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。
72075	データ名 D1 に空白でない値が設定されています。

CBLDCMCF('TLSCN ') － コネクションの状態取得（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL ’CBLDCMCF’ USING 一意名1 一意名2 一意名3
```

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TLSCN   ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D   PIC X(28)  VALUE SPACE.
   02 データ名E   PIC 9(9)   COMP.
   02 データ名F1  PIC X(8).
   02 データ名F2  PIC X(56)  VALUE SPACE.
   02 データ名G   PIC X(8)   VALUE SPACE.
   02 データ名H   PIC X(8)   VALUE SPACE.
   02 データ名I   PIC X(144) VALUE SPACE.
   02 データ名J   PIC X(184) VALUE SPACE.
   02 データ名K   PIC 9(9)   COMP VALUE ZERO.

01 一意名2.
   02 データ名L   PIC 9(9)   COMP VALUE ZERO.

01 一意名3.
   02 データ名M   PIC 9(9)   COMP.
   02 一意名4.
      03 データ名N   PIC X(8).
      03 データ名O   PIC X(4).
      03 データ名P   PIC X(4).
      03 データ名Q   PIC X(40)   VALUE LOW-VALUE.
```

機能

コネクションの状態を取得します。

UAP で値を設定するデータ領域

●データ名 A

コネクション状態取得を示す要求コード「VALUE 'TLSCN△△△'」を設定します。

●データ名 C

状態を取得するコネクションの指定方法を設定します。

VALUE 'LE△△'

状態を取得するコネクションを論理端末名称で指定するときに設定します。

VALUE 'CN△△'

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

空白

省略されたものとして，「VALUE 'LE△△'」（論理端末名称を指定）が設定されます。

●データ名 D

空白を設定します。

●データ名 E

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

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

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

注※

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

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

●データ名 F1

状態を取得するコネクションの論理端末名称，またはコネクション ID を設定します。論理端末名称，またはコネクション ID は最大 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，またはコネクション ID 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，該当するコネクションの個数が返されます。

●データ名 Q

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 M

この命令文の対象となったコネクションの個数が返されます。

●データ名 N

要求したコネクションのコネクション ID が設定されます。8 バイトに満たない場合，コネクション ID の後ろが空白で埋められます。

●データ名 O

要求したコネクションのプロトコル種別が設定されます。

VALUE 'NIF△'

NIF/OSI プロトコル

●データ名 P

要求したコネクションの状態として，次の値が設定されます。

VALUE 'ACT△'

確立状態

VALUE 'ACTB'

確立処理中状態

VALUE 'DCT△'

解放状態

VALUE 'DCTB'

解放処理中状態

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	MCF が開始処理中のため，CBLDCMCF('TLSCN△△△')が受け付けられません。
71004	CBLDCMCF('TLSCN△△△')の処理中にメモリ不足が発生しました。
71005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。
71006	内部障害が発生しました。原因については，メッセージログファイルを参照してください。

ステータスコード	意味
71007	指定されたコネクション ID は登録されていません。
71008	指定された論理端末名称は登録されていません。
71009	CBLDCMCF('TLSCN△△△')が，該当する通信プロセスではサポートされていません。
71010	MCF 通信プロセスにコネクションの状態取得を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。
71011	コネクションが削除されているため，CBLDCMCF('TLSCN△△△')が受け付けられません。
71014	TP1/NET/NCSB，または TP1/NET/X25-Extended の論理端末名称を指定しています。または TP1/NET/OSI-TP のコネクショングループ名を指定しています。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に'LE△△'，'CN△△'，または空白以外の値が設定されています。
72059	データ名 D に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。
72076	データ名 M に 1 以外の値が設定されています。

CBLDCMCF('TLSLE ') － 論理端末の状態取得（COBOL 言語）

形式

PROCEDURE DIVISION の指定

```
CALL  'CBLDCMCF'  USING  一意名1  一意名2  一意名3
```

DATA DIVISION の指定

```
01  一意名1.
    02  データ名A   PIC X(8)   VALUE 'TLSLE   '.
    02  データ名B   PIC X(5).
    02  FILLER      PIC X(3).
    02  データ名C   PIC X(4)   VALUE SPACE.
    02  データ名D   PIC X(28)  VALUE SPACE.
    02  データ名E   PIC 9(9)   COMP.
    02  データ名F1  PIC X(8).
    02  データ名F2  PIC X(56)  VALUE SPACE.
    02  データ名G   PIC X(8)   VALUE SPACE.
    02  データ名H   PIC X(8)   VALUE SPACE.
    02  データ名I   PIC X(144) VALUE SPACE.
    02  データ名J   PIC X(184) VALUE SPACE.
    02  データ名K   PIC 9(9)   COMP VALUE ZERO.

01  一意名2.
    02  データ名L   PIC 9(9)   COMP VALUE ZERO.

01  一意名3.
    02  データ名M   PIC 9(9)   COMP.
    02  一意名4.
        03  データ名N   PIC X(8).
        03  データ名O   PIC X(4)  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。

注※

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

●データ名 F1

状態を取得する論理端末の名称を設定します。論理端末名称は最大 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。

ステータスコード	意味
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C   PIC X(n).
```

PROCEDURE DIVISION（通信文）の指定

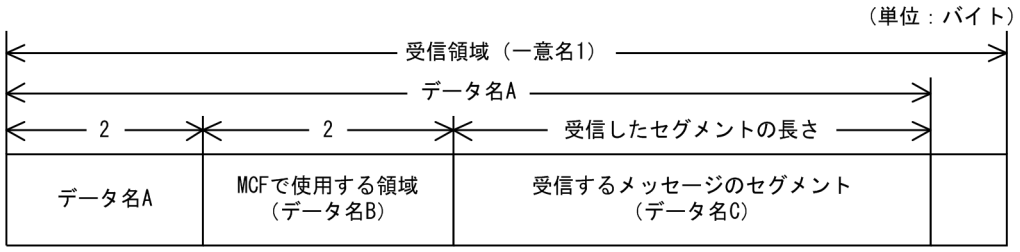
```
RECEIVE 通信記述名
     〔FIRST〕  SEGMENT
       INTO 一意名1.
```

機能

次に示す CALL インタフェースの機能を実現します。

- 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。



UAP で値を設定する項目

●FOR 句

次のどちらかの値を設定します。

INPUT

一方送信メッセージの受信

I-O

問い合わせメッセージの受信

●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ます。

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。

●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 C

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 RECEIVE 文を 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，FIRST を設定しないで RECEIVE 文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 RECEIVE 文を呼び出しています。直前に呼び出した 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。
	メッセージキューが割り当てられていません。
	MCF 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＞ • 非同期型のメッセージの先頭セグメントを受信する RECEIVE 文を呼び出す前に，中間セグメントまたは最終セグメントを受信する RECEIVE 文を呼び出しています。先頭セグメントを受信する場合は，FIRST を指定して RECEIVE 文を呼び出してください。

ステータスコード	意味
72000	• ステータスコード 71001 が返されたあとで，再び非同期型のメッセージを受信する RECEIVE 文を呼び出しています。
	＜ SPP の実行でリターンした場合＞ SPP では非同期型のメッセージを受信する RECEIVE 文を呼び出せません。
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 文を呼び出せ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	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 1（セグメントの内容を設定して送信する場合）

DATA DIVISION（通信記述項）の指定

```
CD 通信記述名
   FOR {OUTPUT PROGRAM | 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C   PIC X(n).
01  一意名3.
    02  データ名D   PIC 9(4)  COMP.
    02  データ名E   PIC X(2).
    02  データ名F   PIC X(n).
```

PROCEDURE DIVISION（通信文）の指定

```
SEND 通信記述名 FROM 一意名1
     〔WITH {ESI | EMI | 一意名2}〕
     〔BEFORE RECEIVING MESSAGE INTO 一意名3〕 .
```

形式 2（セグメントの内容を設定しないで，メッセージの送信の終了だけ連絡する場合）

DATA DIVISION（通信記述項）の指定

```
CD 通信記述名
   FOR {OUTPUT PROGRAM | OUTPUT | I-O}
    〔STATUS KEY IS データ名1〕
    〔SYMBOLIC TERMINAL IS データ名2〕
    〔SWITCHING MODE IS {NORMAL | PRIOR | データ名7}〕 .
```

PROCEDURE DIVISION（通信文）の指定

```
SEND 通信記述名 WITH EMI.
```

機能

次に示す CALL インタフェースの機能を実現します。

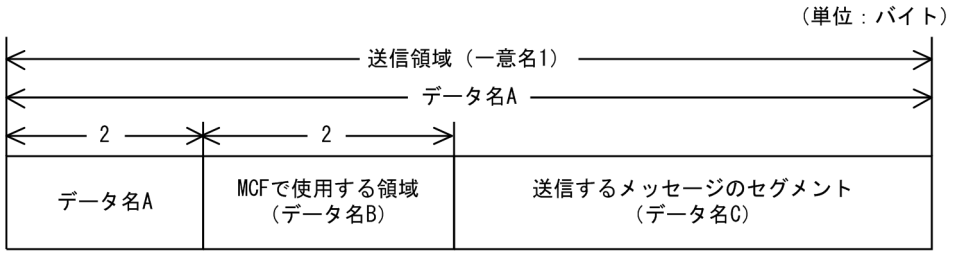
- アプリケーションプログラムの起動　CBLDCMCF('EXECAP△△')

- 応答メッセージの送信　CBLDCMCF('REPLY△△△')
- メッセージの送信　CBLDCMCF('SEND△△△△')
- 同期型メッセージの送受信　CBLDCMCF('SENDRECV')※

注※

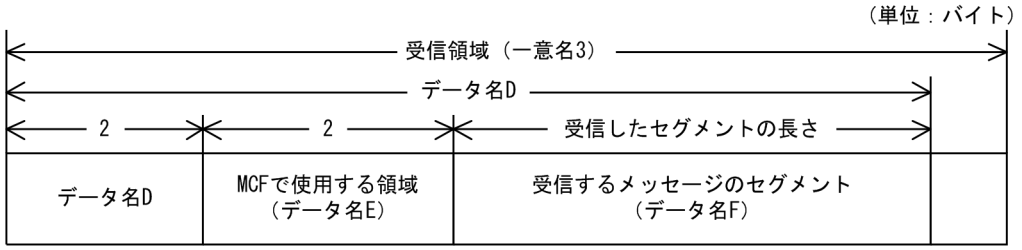
　単一セグメントだけ扱えます。

セグメントを送信する領域（一意名 1 で示す領域）の形式を次に示します。



同期型メッセージの送受信の場合，セグメントを受信する領域（一意名 3 で示す領域）も設定します。

セグメントを受信する領域の形式を次に示します。



UAP で値を設定する項目

●FOR 句

次のどれかを設定します。

OUTPUT PROGRAM

　アプリケーションプログラムの起動

OUTPUT

　メッセージの送信

I-O

　応答メッセージの送信，または同期型メッセージの送受信

●SYMBOLIC TERMINAL 句

FOR 句の設定内容によって，次のどれかを設定します。

OUTPUT PROGRAM

アプリケーション名を設定したデータ項目を設定します。データ名 2 に起動するアプリケーション名を設定します。アプリケーション名は最大 8 バイトの長さです。8 バイトに満たない場合，アプリケーション名の後ろを空白で埋めてください。

OUTPUT

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。

I-O

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受信

同期型のメッセージの送受信のとき設定します。

ASYNC

非同期型メッセージの送信

メッセージの送信または応答メッセージの送信のとき設定します。

データ名 6

次の値を設定したデータ項目

'0'または'△'：非同期型メッセージの送信

'1'：同期型メッセージの送受信

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

メッセージの送信の場合に，一般か優先かを設定します。

NORMAL

一般のメッセージ送信

PRIOR

優先のメッセージ送信

データ名 7

次の値を設定したデータ項目

'0'または'△'：一般のメッセージ送信

'1'：優先のメッセージ送信

省略した場合および非応答型のアプリケーションから応答メッセージを送信した場合は，NORMAL（一般のメッセージ送信）が設定されます。

●DETAIL MODE 句

非同期型メッセージの送信の場合に，出力通番を付けるかどうかを設定します。データ名 10 に次のどちらかを設定します。

データ名 10

次の値を設定したデータ項目

'0'または'△'：出力通番を付けます。

'1'：出力通番を付けません。

省略した場合および非応答型のアプリケーションから応答メッセージを送信した場合は，出力通番を付けません。

●データ名 A

送信するセグメントの長さ＋４を設定します。

●データ名 B

MCF で使用する領域です。

●データ名 C

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

●データ名 E

MCF で使用する領域です。

●WITH 句

送信するセグメントを設定します。非同期型のメッセージ送信の場合だけ設定します。

ESI

先頭セグメントまたは中間セグメントの送信

EMI

最終セグメントまたは単一セグメントの送信

一意名 2

次の値を設定したデータ項目

'1'：ESI（先頭セグメントまたは中間セグメントの送信）

'2'：EMI（最終セグメントまたは単一セグメントの送信）

省略した場合は，EMI（最終セグメントまたは単一セグメントの送信）が設定されます。

なお，同期型メッセージの送受信の場合は，EMI を指定するか，または指定を省略してください。

●BEFORE 句

同期型メッセージの送受信の場合，セグメントを受信するデータ項目（一意名 3）を設定します。メッセージの送信，または，応答メッセージの送信の場合，BEFORE 句を省略します。

OpenTP1 から値が返される項目

●STATUS KEY 句

ステータスコードを受け取りたい場合に設定します。省略した場合は，ステータスコードを受け取れません。データ名 1 にステータスコードが返されます。

●データ名 D

受信したセグメントの長さ＋ 4 が返されます。

●データ名 F

受信したセグメントの内容が返されます。

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を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 文を呼び出す前に，SEND 文を呼び出しています。 • 非応答型のアプリケーションからの問い合わせ応答をしない（UAP 共通定義（mcfmuap -c）の noansreply オペランドに no を指定）場合に，非応答型のアプリケーションから応答メッセージを送信しています。
	＜ SPP の実行でリターンした場合＞ トランザクションでない SPP の処理から，SEND 文を呼び出しています。 SPP では応答メッセージを送信する SEND 文を呼び出せません。

ステータスコード	意味
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 文を呼び出せない論理端末を設定しています。
	設定したアプリケーション名は，定義されていません。
	アプリケーション名が間違っています。
	MCF マネジャ定義の通信サービス定義（mcfmcname）に，アプリケーション起動プロセス名または MCF 通信プロセス名を指定していません。
	アプリケーション起動プロセス，または MCF 通信プロセスに対応するアプリケーション環境定義（mcfaenv -p）に，アプリケーション起動プロセス識別子を指定していません。
	アプリケーション環境定義（mcfaenv -p）で指定したアプリケーション起動プロセス識別子と，アプリケーション起動プロセス，または MCF 通信プロセスの MCF 環境定義（mcftenv -s）で指定する識別子が一致していません。
	＜論理端末名称を指定してアプリケーションを起動する場合＞ • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の lname オペランドに論理端末を指定していません。 • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の lname オペランドに指定した論理端末を，MCF 通信プロセスの論理端末定義（mcftalcle）に定義していません。 • 起動先アプリケーションのアプリケーション属性定義に指定した論理端末が，request 型論理端末または send 型論理端末ではありません。 • 起動先アプリケーションのアプリケーション属性定義で指定した論理端末はアプリケーション起動を使えません。
	＜コネクション ID を指定してアプリケーションを起動する場合＞ • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の cname オペランドにコネクション ID を指定していません。 • 起動先アプリケーションのアプリケーション属性定義（mcfaalcap -n）の cname オペランドに指定したコネクション ID を，MCF 通信プロセスのコネクション定義（mcftalccn）に定義していません。 • MCF 通信プロセスの論理端末定義（mcftalcle）に，request 型論理端末を指定していません。
	＜ SPP からアプリケーションを起動する場合＞ • アプリケーション起動プロセス識別子を起動元の UAP のユーザサービス定義，またはユーザサービスデフォルト定義の mcf_psv_id オペランドに指定していません。 • 起動元の UAP のユーザサービス定義，またはユーザサービスデフォルト定義の mcf_psv_id オペランドに指定しているアプリケーション起動プロセス識別子が，アプリケーション起動プロセス，または MCF 通信プロセスの MCF 環境定義（mcftenv -s），およびアプリケーション環境定義（mcfaenv -p）で指定しているアプリケーション起動プロセス識別子と一致していません。 • 起動元の UAP のユーザサービス定義，またはユーザサービスデフォルト定義の mcf_mgrid オペランドに指定している MCF マネジャ識別子が，アプリケーション起動プロセスが属している MCF マネジャの識別子と一致していません。
72005	＜ WITH 句に ESI を設定または WITH 句に 1 を設定した一意名 2 を設定した場合＞ 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。

ステータスコード	意味
72007	応答型（type=ans）の MHP で，応答メッセージを送信したあとで，応答型の MHP を起動しています。
	継続問い合わせ応答型（type=cont）の MHP で，応答メッセージを送信したあとで，継続問い合わせ応答型の MHP を起動しています。
72008	応答メッセージの最終セグメントを送信する SEND 文を呼び出したあとに，再び応答メッセージを送信する SEND 文を呼び出しています。
	SEND 文を呼び出して応答型（type=ans）の MHP を起動させたあとに，応答メッセージを送信する SEND 文を呼び出しています。
72009	応答型（type=ans）の MHP を 2 回以上起動しています。
	継続問い合わせ応答型（type=cont）の MHP を 2 回以上起動しています。
72011	応答型（type=ans）でない MHP から，応答型の MHP を起動させています。
	継続問い合わせ応答型（type=cont）でない MHP から，継続問い合わせ応答型の MHP を起動しています。
72013	データ名 F のサイズを超えるセグメントを受信しました。データ名 F のサイズを超えた部分は切り捨てられました。
	32763 バイトを超えるセグメントを受信しました。32763 バイトを超えた部分は切り捨てられました。
72017	DETAIL MODE 句に設定した値が間違っています。
72018	SWITCHING MODE 句に設定した値が間違っています。
72020	SYNCHRONOUS MODE 句に設定した値が間違っています。
72024	FOR 句に設定した値が間違っています。
72026	WITH 句に設定した値が間違っています。
72036	データ名 F のサイズが不足しています。5 バイト以上の領域を確保してください。
72041	＜ WITH 句を省略または WITH 句に EMI を設定または WITH 句に 2 を設定した一意名 2 を設定した場合＞ 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。
	＜メッセージの送信の終了を連絡した場合＞ WITH 句に ESI を設定または WITH 句に 1 を設定した一意名 2 を設定した SEND 文を呼び出さないで，メッセージの送信の終了を連絡しています。
72044	継続問い合わせ応答型のアプリケーションはサポートしていません。
72045	継続問い合わせ応答型のアプリケーションはサポートしていません。
72047	継続問い合わせ応答型のアプリケーションはサポートしていません。
73001	同期型送受信監視時間（mcfmuap -t sndrcvtim）に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
	SEND 文を呼び出したあとで，論理端末の閉塞またはコネクションの解放が発生しました。
	出力先の論理端末で MCF 通信プロセスの内部障害が発生しました。
73002	論理端末が閉塞されています。

ステータスコード	意味
73004	メッセージの送信処理中にタイムアウトが発生しました。
73005	メッセージの受信処理中にタイムアウトが発生しました。
73008	運用コマンドまたは障害による論理端末閉塞処理中に，SEND 文を呼び出しました。
73009	論理端末閉塞解除処理中に，SEND 文を呼び出しました。
73010	入力メッセージ編集 UOC または出力メッセージ編集 UOC で障害が発生しました。
73015	出力先の論理端末は，ほかの UAP で仕掛り中です。
73018	WAITING TIME 句に設定した値が間違っています。
77001	起動しようとするアプリケーションに対応する論理端末は，現在仕掛り中で使用できません。または，使用できる論理端末がありません。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

5 ユーザオウンコーディング，MCF イベントインタフェース

TP1/NET/OSAS-NIF 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。

TP1/NET/OSAS-NIF で使用できる UOC を次に示します。

1. 入力メッセージ編集 UOC
2. 出力メッセージ編集 UOC
3. 送信メッセージの通番編集 UOC

UOC は K&R 版 C，ANSI C および C++言語で作成します。UOC を作成する言語は，MCF メイン関数と同一言語で記述します。UOC を使用する場合は，あらかじめ UOC と MCF 提供ライブラリをリンケージする必要があります。

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の編集をする UOC です。

論理メッセージをユーザ任意の形式に変換したり，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したりできます。

TP1/NET/OSAS-NIF は，メッセージの最終セグメント受信後，入力メッセージ編集 UOC を起動します。ただし，MCF イベント通知時と，UAP からのシステム内部のアプリケーションプログラム起動時は起動しません。

入力メッセージ編集 UOC は，ユーザが作成する UOC のほかに，TP1/NET/OSAS-NIF が提供する標準 UOC があります。標準 UOC については，「(6)　標準 UOC」を参照してください。

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，必要に応じてコネクション定義（mcftalccn -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。

入力メッセージ編集 UOC のコーディングについては，「付録 H　UOC のコーディング例」を参照してください。

(1)　入力メッセージの編集

受信したメッセージが格納されている受信バッファおよび定義で指定した編集バッファをリスト形式で引き渡します。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は，UOC からのリターンコードによって，受信バッファに格納されたものを使用するか，または編集バッファに格納されたものを使用するかを選択できます。

(2) アプリケーション名の決定

該当する MCF に入力メッセージ編集 UOC が登録されている場合，論理メッセージの編集と同時にアプリケーション名を決定できます。

TP1/NET/OSAS-NIF は，相手システムからのメッセージの NIF ヘッダにあて先名称としてアプリケーション名が設定されている場合は，その情報を JIS コードに変換して UOC へ渡します。その情報は，プロトコル個別インタフェース領域アドレスのあて先名称領域（dcmnom_uoc の appname）に設定されます。ただし，アプリケーション名に英数字（大文字）以外の文字が設定されている場合，「¥0」を UOC へ渡します。

ユーザが作成する 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」の手前までに 1 から 8 バイトの識別子を設定します。先頭から 9 バイト目までに「¥0」がない場合，アプリケーション名を不正とし，不正アプリケーション名検出通知イベント（ERREVT1）を通知します。

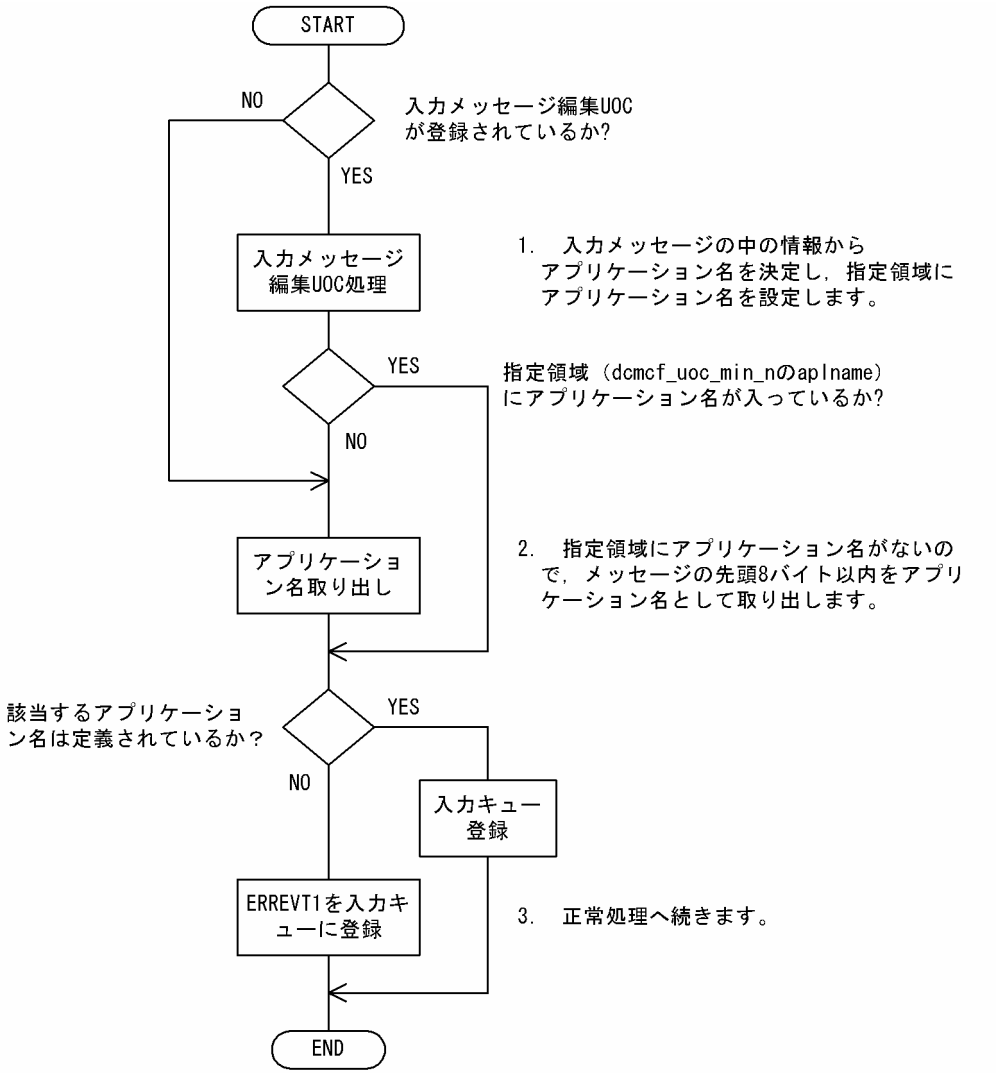
UOC でアプリケーション名を決定しない場合※，メッセージの先頭 8 バイト以内をアプリケーション名として使用します。ただし，この場合は JIS コードへの変換はしません。また，先頭から 9 バイト目までに空白がない場合，アプリケーション名を不正とし，不正アプリケーション名検出通知イベント（ERREVT1）を通知します。

注※

UOC が登録されていない，または，UOC で指定領域(dcmcf_uoc_min_n の aplname)を「¥0」で埋めた場合，UOC でアプリケーション名を決定しなかったと判断します。

アプリケーション名の決定の処理を次の図に示します。

図 5-1　アプリケーション名の決定の処理



(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，TP1/NET/OSAS-NIF は相手システムに受信拒否を送信し，該当する論理端末を閉塞します。なお，このとき 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をする UAP のアプリケーション名を設定します。また，MCF には正常リターンします。

(4)　UOC パラメタ不正の場合の処理

UOC で設定した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

MCF メイン関数で，作成した UOC の関数アドレスを指定します。入力メッセージ編集 UOC の関数アドレスは任意に決められます。UOC のオブジェクトファイルは，MCF メイン関数を翻訳・結合すると，TP1/NET/OSAS-NIF の実行形式ファイルに結合されて実行できる状態になります。MCF メイン関数の詳細については，「8.2　MCF メイン関数の作成」を参照してください。

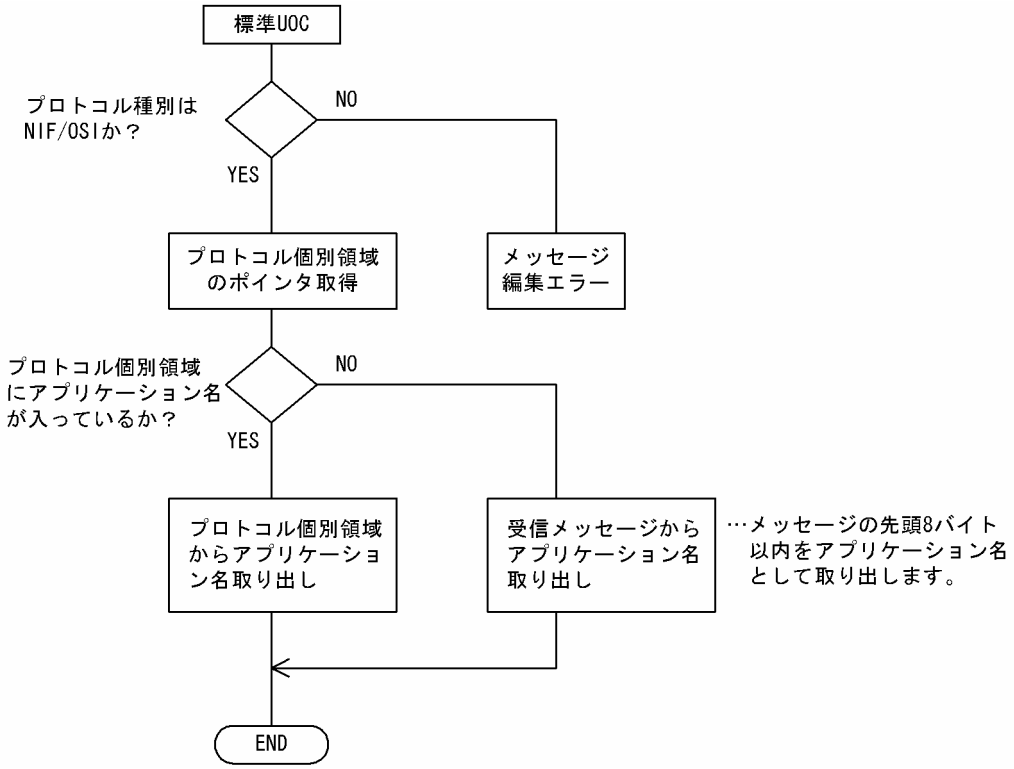
(6)　標準 UOC

TP1/NET/OSAS-NIF は，アプリケーション名を決定するための標準 UOC を提供しています。ユーザは，MCF メイン関数に標準 UOC のアドレスを指定すると，使用できます。関数名を次に示します。

関数名：dc_mcf_stduoc_msgin

標準 UOC の処理の概要を次の図に示します。

図 5–2　標準 UOC の処理の概要



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++ の形式

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

K&R 版 C の形式

```
#include <dcmcf.h>
#include <dcmnom.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;                   …プロトコル個別インタフェース
                                              領域アドレス
    DCLONG rtn_detail;                      …詳細リターンコード
    char   reserve3[8];                     …予備
} dcmcf_uoc_min_n;
```

(b) dcmcf_uocbuff_list_n（バッファリスト）の内容

```
typedef struct {
    DCLONG buf_num;                         …バッファ情報数
    DCLONG used_buf_num;                    …使用バッファ情報数
    char   reserve1[8];                     …予備
    dcmcf_uocbufinf_n buf_array[DCMCF_UOC_BUFF_MAX];
                                            …バッファ情報
} dcmcf_uocbuff_list_n;
```

(c) dcmcf_uocbufinf_n（バッファ情報）の内容

```
typedef struct {
    char   *buf_adr;                    …バッファアドレス
    DCULONG buf_size;                   …バッファ最大長
    DCULONG seg_size;                   …セグメント長
    char   reserve1[4];                 …予備
    dcmcfuoc_w_type buff_id;            …MCF内部情報
    DCMLONG buff_addr;                  …MCF内部情報
    char   reserve2[4];                 …予備
} dcmcf_uocbufinf_n;
```

(d) プロトコル個別インタフェース領域の内容

```
typedef struct {                         …プロトコル個別インタフェース領域
    char lename[16];                    …MCF内部情報
    char mapname[10];                   …MCF内部情報
    char reserve1[6];                   …予備
    char appname[10];                   …あて先名称領域
    char reserve2[6];                   …予備
} dcmnom_uoc;
```

(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_NF：NIF/OSI プロトコル

- le_name

  メッセージを入力した論理端末の名称が設定されます。

- rcv_prim

  受信サービスプリミティブとして，次の値が設定されます。

  - DCMCF_UOC_RCV_BRD：一方送信メッセージの受信
  - DCMCF_UOC_RCV_INQ：問い合わせメッセージの受信
  - DCMCF_UOC_RCV_REP：応答メッセージの受信
  - DCMCF_UOC_RCV_REP_SR：同期型応答メッセージの受信

    DCMCF_UOC_RCV_REP_SR が設定されている場合，アプリケーション名は無効です。

- buflist_adr

  受信用バッファリストのアドレスが設定されます。

- ebuflist_adr

  編集用バッファリストのアドレスが設定されます。

メッセージ編集バッファが未定義の場合，つまり，コネクション定義（mcftalccn -e）を省略した場合，ebuflist_adr にはヌル文字が設定されます。

- aplname

すべて「¥0」を設定します。

- pro_indv_ifa

プロトコル個別インタフェース領域アドレスが設定されます。

(b) dcmcf_uocbuff_list_n（バッファリスト）

- buf_num

バッファ情報の数が設定されます。

- 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，セグメント長が設定されます。

- buff_id，buff_addr

MCF で使用するパラメタです。

(d) プロトコル個別インタフェース

- lename，mapname

MCF で使用するパラメタです。

- appname

あて先名称が設定されます。TP1/NET/OSAS-NIF は，相手システムからきた EBCDIK コードを，自システムのコード体系に合わせて変換し，設定します。

相手システムからのあて先名称が設定されていなかった場合，またはあて先名称に英数字（大文字）以外のコードが設定されている場合は，すべて「¥0」を設定します。

(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 でリターンした場合に，MCF に渡されます。

MCF は，詳細リターンコードをメッセージログファイルに出力します。

詳細リターンコードは，-19999～-19000 の範囲で指定してください。

なお，標準 UOC は，詳細リターンコードを使用しません。標準 UOC でメッセージ不正を検出した場合は，詳細リターンコードは 0 になります。

(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。

(c) dcmcf_uocbufinf_n（バッファ情報）

- seg_size

セグメント長を設定します。

(6) リターン値

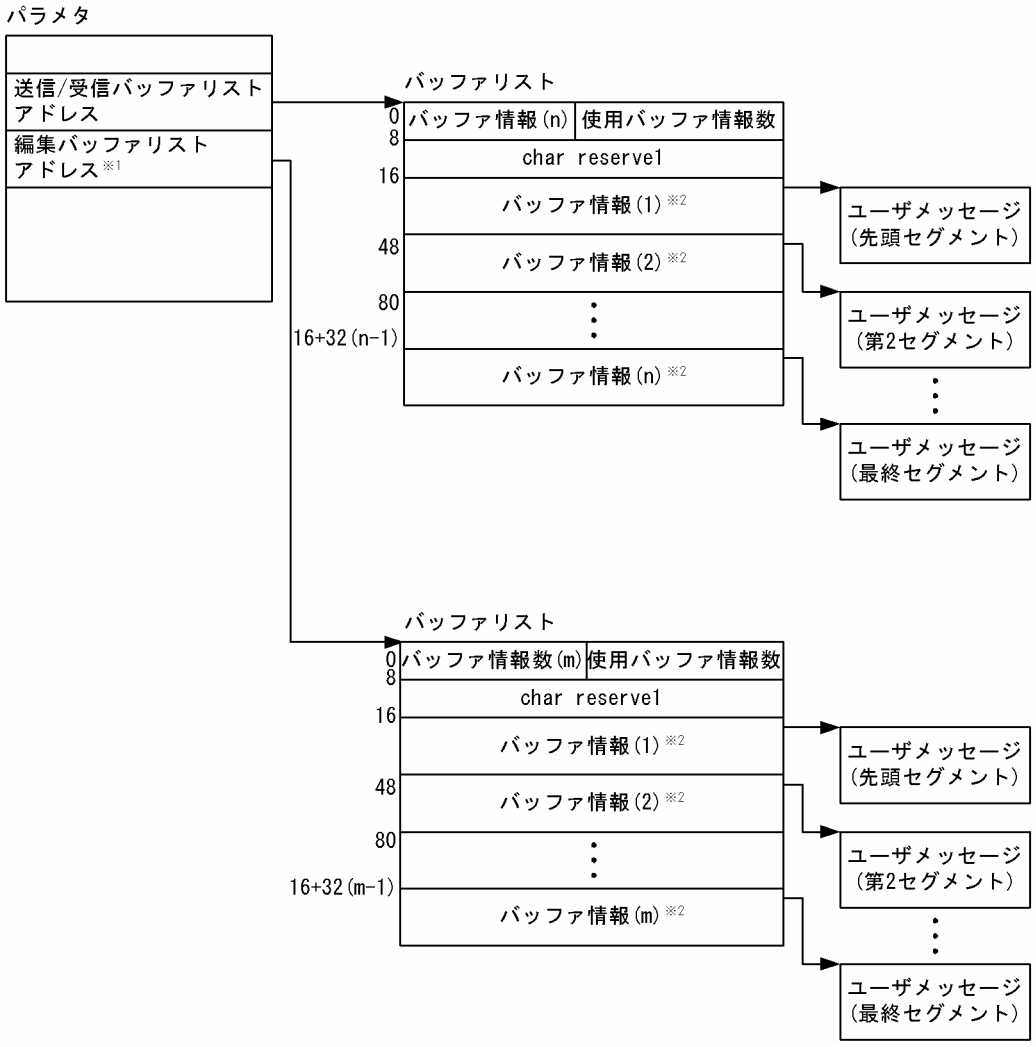
uoc_func()は次のコードでリターンしてください。

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 5-3　UOC インタフェース用のパラメタとバッファの関係



注※1

mcftalccn -e オプションを指定しなければヌル文字となり，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 32 バイトで次の形式をしています。

5.1.3　出力メッセージの編集

出力メッセージ編集 UOC は，送信する論理メッセージを編集する UOC です。出力メッセージ編集 UOC は，UAP が発行した送信メッセージを相手システムに送信する前に処理します。TP1/NET/OSAS-NIF は，出力キューから全セグメントを読み出すと，出力メッセージ編集 UOC を起動させます。

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，必要に応じてコネクション定義（mcftalccn -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定義で指定した編集バッファをリスト形式で引き渡します。UOC では，これらのバッファを使用して，出力メッセージの編集処理ができます。

また，UOC からのリターンコードで相手システムへ送信するメッセージとして，送信バッファに格納されたものを使用するか，または編集バッファに格納されたものを使用するかを選択できます。

(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，該当するメッセージを破棄し，論理端末を閉塞します。なお，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。

(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。

(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++　の形式

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

K&R 版 C　の形式

```
#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;                     …再送フラグ
    char   reserve2[1];                     …予備
    char   *pro_indv_ifa;                   …MCFが使用
    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

入力メッセージ編集 UOC インタフェースのバッファリストおよびバッファ情報の内容と同じです。「5.1.2(3)(b)　dcmcf_uocbuff_list_n（バッファリスト）の内容」および「5.1.2(3)(c)　dcmcf_uocbufinf_n（バッファ情報）の内容」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_NF：NIF/OSI プロトコル

- le_name

メッセージを出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。

  メッセージ編集バッファが未定義の場合，つまり，コネクション定義（mcftalccn -e）を省略した場合，ebuflist_adr にはヌル文字が設定されます。

- 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 の場合だけ有効です。

- msg_type

  メッセージ種別として，次の値が設定されます。ただし outputno_flag が DCMCF_UOC_OUTPUTNO_OK の場合だけ有効です。

  - 'o'：応答メッセージ
  - 'n'：一般送信メッセージ
  - 'p'：優先送信メッセージ
  - 's'：同期型の送信メッセージ

- outputno_flag

  メッセージ出力通番が有効であるかどうか，次の値が設定されます。

  - DCMCF_UOC_OUTPUTNO_OK：メッセージ出力通番有効
    （output_no および msg_type が有効になります）
  - DCMCF_UOC_OUTPUTNO_NG：メッセージ出力通番無効

- resend_flag

  次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）

入力メッセージ編集 UOC インタフェースのバッファリストおよびバッファ情報の内容と同じです。「5.1.2(4)(b) dcmcf_uocbuff_list_n（バッファリスト）」および「5.1.2(4)(c) dcmcf_uocbufinf_n（バッファ情報）」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 の範囲で指定してください。

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

入力メッセージ編集 UOC インタフェースのバッファリストおよびバッファ情報の内容と同じです。
「5.1.2(3)(b)　dcmcf_uocbuff_list_n（バッファリスト）の内容」および「5.1.2(3)(c)　dcmcf_uocbufinf_n（バッファ情報）の内容」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

UOC が正常リターンすると，TP1/NET/OSAS-NIF は，パラメタから取り出したアプリケーション名を EBCDIK コードに変換してメッセージを送信します。

(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 の場合と同じです。
「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

送信メッセージの通番編集 UOC は，受け取った出力通番を基に，ユーザ独自の処理をするための UOC です。

送信メッセージの通番編集 UOC を起動する場合，メッセージ送信の関数で，出力通番を付けるように設定してください。UOC は，UAP が先頭セグメントを送信する関数を発行したときに，MCF によって起動されます。したがって，この UOC でメッセージを編集する場合，先頭セグメントしか編集できません。

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。

(1)　形式

ANSI C，C++の形式

```
#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

```
#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 から値が渡される引数

- flags

  送信メッセージの通番編集 UOC がいつ呼ばれたかが渡されます。次の値が渡されます。

  - 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
  - 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

- termname

  送信先の論理端末名称が渡されます。

- sendno

  送信メッセージの出力通番が渡されます。

- sendid

  送信するメッセージ種別が渡されます。次のどちらかが渡されます。

  - 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渡されます。

- senddata

  送信メッセージの先頭セグメントのアドレスが渡されます。

(3) リターン値

リターン値	意味
DC_OK	正常リターン

(4) 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++の形式

```
#include <dcmcf.h>
int dc_mcf_regster(DCLONG flags, DCLONG (*uoc_addr)())
```

K&R 版 C の形式

```
#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項目

- flags

  DCMCF_SEND_UOC を設定します。

- uoc_addr

  flags に対応する UOC のアドレスを設定します。

(c) リターン値

リターン値	意味
DC_OK	正常に終了しました。
DCMCFER_INVALID_ARGS	引数の指定が間違っています。
DCMCFER_NOMEM	ローカルメモリが不足しています。

メイン関数への登録例を次に示します。

```
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();
}
```

5.1.7　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 では次に示す関数だけを使用できます。ほかの関数を使用した場合，TP1/NET/OSAS-NIF の動作に影響を与えるおそれがあるため使用しないでください。

- メモリ操作をする関数
  - データ領域管理（例：malloc，free）
  - 共有メモリ管理関数（例：shmctl，shmget，shmop）
  - メモリ操作（例：memory）
  - 文字列操作（例：string）
- 時間取得関数

(3)　UOC の異常処理

TP1/NET/OSAS-NIF の UOC で異常を検知した場合，MCF の所定のリターンコードを使用して，MCF に異常の発生を通知してください。UOC でプロセス終了となるシグナル，または abort()を発行すると，MCF が異常終了します。

(4)　UOC の実行タイミング

TP1/NET/OSAS-NIF 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，終了シーケンスと必ずしも同期が取れません。UAP より先に UOC が実行されたり，UAP がすべて終了してから UOC が呼ばれたりしても問題がないように作成してください。

(5)　UOC インタフェースパラメタの設定する項目

UOC パラメタの設定で，**ユーザが値を設定する項目**以外の項目について更新しないでください。

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。

この節では，TP1/NET/OSAS-NIF が通知する MCF イベントについて説明します。なお，MCF イベントの発生時は，入力メッセージの編集 UOC を呼び出しません。

MCF イベントの概要については，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/OSAS-NIF 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/OSAS-NIF 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。 • 入力メッセージ最大格納数を超過しました。 • 動的共用メモリが不足しました。 • キューファイルが満杯になりました。 • MHP のサービス，サービスグループ，またはアプリケーションが閉塞しています。 • スケジュール閉塞されているサービスグループの入力キューに未処理受信メッセージが残った状態で，OpenTP1 を正常終了または計画停止 A で終了しました。 • MHP のサービスグループ，またはアプリケーションがセキュア状態です。 • MHP で呼び出す receive 関数にセグメントを渡す前に，MHP が異常終了しました。 • アプリケーション名に相当する MHP のサービスがありません。 • ユーザサーバ未起動などによって，MHP の起動に失敗しました。 • DBMS の障害などによって，トランザクションの開始に失敗しました。	メッセージを廃棄したことを報告します。

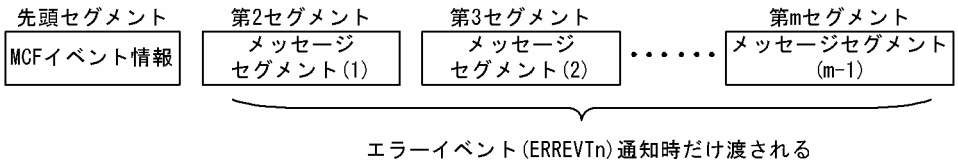
MCF イベント名	MCF イベントコード	発生した原因	MCF イベント処理用 MHP での処理の例
UAP 異常終了通知イベント	ERREVT3	MHP のセグメント受信関数に，セグメントを渡したあとに MHP の異常終了が発生しました。	UAP 異常終了時の対処障害メッセージを送信します。
未処理送信メッセージ廃棄通知イベント	ERREVTA	次の理由で未処理送信メッセージを破棄しました。 • MCF の正常終了処理時に，未処理送信メッセージの滞留時間監視の時間切れ（タイムアウト）が発生しました。 • 運用コマンド（mcftdlqle）の入力，または API（dc_mcf_tdlqle 関数もしくは CBLDCMCF('TDLQLE△△')）の発行によって，出力キューが削除されました。 • 閉塞されている論理端末の出力キューに未処理送信メッセージが残った状態で，dcstop コマンドが実行されました。	未処理送信メッセージを廃棄したことを報告します。
障害通知イベント	CERREVT	コネクション障害または論理端末障害が発生しました。	コネクションまたは論理端末に障害が発生したことを報告します。
状態通知イベント	COPNEVT	コネクションが確立しました。	コネクションが確立したことを報告します。
	CCLSEVT	コネクションが正常に解放されました。	コネクションが解放されたことを報告します。

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 セグメント以降に処理できなかったメッセージセグメントを最終セグメントまで設定します。

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 5-4 MCF イベント通知時のセグメント構成



MCF イベントは，作成した UAP が C 言語の場合と COBOL 言語の場合で，UAP へ渡される形式が異なります。

COBOL 言語を使用したエラーイベント（ERREVTn）の場合は，バッファ形式 1 とバッファ形式 2 とで，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，バッファ形式ごとに位置（バイト）を分けて説明しています。

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。C イベント（CxxxEVT）で使用する構造体は<dcmnom.h>で定義してあります。<dcmcf.h>，<dcmnom.h>の順で取り込んでください。

各 MCF イベントの共通ヘッダと，各イベントの MCF 情報の形式を次に示します。

(1) MCF イベントの共通ヘッダ

(a) 形式

```
struct dc_mcf_evtheader {
  char   mcfevt_name[9] ;              … MCFイベントコード
  char   le_name[16] ;                 … 論理端末名称
  char   cn_name[9] ;                  … コネクション名
  unsigned char format_kind;           … MCF使用領域
  char   reserve01 ;                   … 予備
  DCLONG time ;                        … メッセージ入力時刻
} ;
```

(b) MCF イベントとして設定される項目

- le_name

  メッセージを入力した論理端末名称が設定されます。

  ERREVT2 または ERREVT3 で，次に示す場合は，'*'が設定されます。

  - SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合
  - 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動した MHP で，障害が発生した場合

  ERREVTA の場合は，メッセージを出力する論理端末名称が設定されます。

  論理端末障害の CERREVT の場合は，障害が発生した論理端末名称が設定されます。

  COPNEVT，CCLSEVT およびコネクション障害の CERREVT の場合は，不定です。

- cn_name

  コネクション名が設定されます。

  ERREVT2 または ERREVT3 で，次に示す場合は，'*'が設定されます。

- SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合
- 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動した MHP で，障害が発生した場合

- time

メッセージを入力した時刻が，1970 年 1 月 1 日 0 時 0 分 0 秒からの通算の秒数で設定されます。

(2) ERREVT1

(a) 形式

```
struct dc_mcf_evt1_type {
  struct dc_mcf_evtheader  evtheader ; … MCFイベント共通ヘッダ
  char reserve01[12] ;                 … 予備
  char reserve02[10] ;                 … 予備
  char reserve03[2] ;                  … 予備
  char ap_name[10] ;                   … アプリケーション名
                                          （メッセージに対応する
                                          アプリケーション名）
  char reserve04[2] ;                  … 予備
} ;
```

(b) MCF イベントとして設定される項目

- ap_name

次に示すどれかが設定されます。

- 形式不正の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

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。

(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。

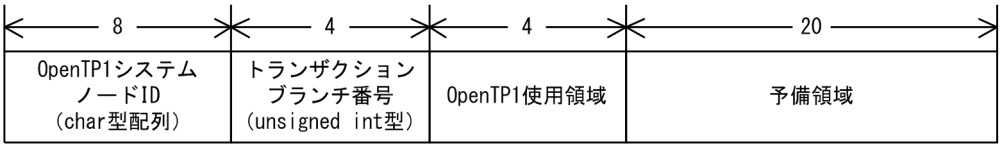
- serv_grp_name

  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 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
                                          （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。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

コネクション障害または論理端末障害発生時に，相手システムから受信した障害報告の利用者データがある場合，CERREVT の第 2 セグメントに設定します。

(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
  DCLONG err_reason1 ;                   … 理由コード1
  DCLONG err_reason2 ;                   … 理由コード2
  char   reserve1[44] ;                  … 予備
} dcmnom_cerrevt ;
```

(b) MCF イベントとして設定される項目

- err_fact

  障害の発生要因が設定されます。

  - 0x30：コネクション障害発生
  - 0x31：論理端末障害発生

- err_reason1，err_reason2

  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。

(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[56] ;                   … 予備
} dcmnom_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。

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備	20	16	20	−	−
アプリケーション名	40	36	8	英数字	次に示すどれかが設定されます。 • 形式不正となったアプリケーション名 • 定義されていないアプリケーション名
予備	48	44	8	−	−
予備	56	52	8	−	−
予備	64	60	8	−	−
コネクション名	72	68	8	英数字	コネクション名です。
予備	80	76	16	−	−
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。 YYYYMMDD の形式です。 YYYY：西暦の年 MM：月 DD：日
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。 HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
予備	112	108	16	−	−

（凡例）

−：該当しません。または，使用されません。

表 5-3　COBOL 言語の MCF イベント情報の内容（ERREVT2）

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備（形式 1 のときだけ）	0	−	2	−	−
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。

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	メッセージを入力した論理端末名称です。 次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	20	16	20	−	−
アプリケーション名	40	36	8	英数字	エラーになった UAP のアプリケーション名が設定されます。
予備	48	44	8	−	−
予備	56	52	8	−	−
予備	64	60	8	−	−
コネクション名	72	68	8	英数字	コネクション名です。 次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	80	76	16	−	−
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。 YYYYMMDD の形式です。 YYYY：西暦の年 MM：月 DD：日
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。 HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
理由コード※	112	108	4	外部 10 進数字	理由コードが設定されます。
予備	116	112	12	−	−

（凡例）

－：該当しません。または，使用されません。

注※

理由コードの内容については，「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。 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	－	－
予備	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	英数字	コネクション名です。 次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	80	76	16	－	－
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。 YYYYMMDD の形式です。

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
メッセージが入力された日付	96	92	8	外部 10 進数字	YYYY：西暦の年 MM：月 DD：日
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。 HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
予備	112	108	16	−	−
サービス名	128	124	31	英数字	異常が発生した MHP 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 MHP のサービスグループ名です。
予備	191	187	1	−	−
トランザクションブランチ ID（BID）	192	188	36	英数字	異常が発生したトランザクションの BID です。 トランザクションブランチ ID の形式については，表 5-5 を参照してください。
予備	228	224	28	−	−

（凡例）

−：該当しません。または，使用されません。

表 5-5　トランザクションブランチ ID の形式

項目	位置（バイト）	長さ（バイト）	属性
OpenTP1 システムノード ID	0	8	英数字
トランザクションブランチ番号	8	4	2 進数字
OpenTP1 使用領域	12	4	−
予備	16	20	−

（凡例）

−：該当しません。または，使用されません。

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	−	−
予備	48	44	8	−	MCF が使用します。
アプリケーション名	56	52	8	英数字	正常終了したメッセージのアプリケーション名（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，空白が設定されます）。
予備	64	60	8	−	−
コネクション名	72	68	8	英数字	コネクション名です。
予備	80	76	16	−	−
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。 YYYYMMDD の形式です。 YYYY：西暦の年 MM：月 DD：日
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。 HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
予備	112	108	16	−	−
予備	128	124	31	−	−
予備	159	155	1	−	−
予備	160	156	31	−	−

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
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	MCF が使用します。※1
予備	16	8	−	−
入力元コネクション名	24	8	英数字	コネクション名が設定されます。
メッセージが入力された日付	32	8	外部 10 進数字	CERREVT 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	40	8	外部 10 進数字	CERREVT 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。
障害要因コード	48	4	2 進数字	障害要因が設定されます。 $(00000030)_{16}$：コネクション障害 $(00000031)_{16}$：論理端末障害
理由コード 1※2	52	4	2 進数字	理由コード 1 が設定されます。
理由コード 2※2	56	4	2 進数字	理由コード 2 が設定されます。
予備	60	44	−	−

（凡例）

−：該当しません。または，使用されません。

注※1

論理端末障害の場合，障害の発生した論理端末名称が設定されます。

注※2

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	MCF が使用します。
予備	16	8	－	－
入力元コネクション名	24	8	英数字	コネクション名が設定されます。
メッセージが入力された日付	32	8	外部 10 進数字	COPNEVT，CCLSEVT 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	40	8	外部 10 進数字	COPNEVT，CCLSEVT 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	48	56	－	－

（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

NIF/OSI プロトコルを使用する場合の TP1/NET/OSAS-NIF のシステム定義，自システムの通信管理プログラム（XNF/AS）と関連づける内容，相手システムとの定義の関係およびシステム定義例について説明します。

TP1/NET/OSAS-NIF の定義の概要

TP1/NET/OSAS-NIF 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/OSAS-NIF に固有の定義を説明します。

使用する定義ファイル

MCF および TP1/NET/OSAS-NIF 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 プロトコル固有定義ソースファイル	プロトコルごとの実行環境
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，オペランドを使用して作成します。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/OSAS-NIF に固有の定義があるものを次に示します。

- MCF マネジャ定義
- MCF 通信構成定義
- MCF アプリケーション定義

この章では，TP1/NET/OSAS-NIF に固有，または関連する定義コマンド，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 コマンド（バッファグループ定義）の length オペランド，count オペランドの指定値については，「mcftalccn（コネクション定義の開始）」の注意事項を参照してください。

TP1/NET/OSAS-NIF 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/OSAS-NIF 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コマンドについて説明します。TP1/NET/OSAS-NIF 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。

通信定義の内容の関連づけ

NIF/OSI プロトコルを使用して相手システムと通信するためには，TP1/NET/OSAS-NIF のシステム定義内容を自システムの通信管理プログラムや，相手システムの通信定義と関連づける必要があります。

自システムの通信管理プログラム XNF/AS と関連づける内容については，「自システムの通信管理プログラム（XNF/AS）と関連づける内容」で説明します。さらに，相手システム（TMS-4V/SP または XDM/DCCM3）のネットワーク定義と関連づける内容については，「相手システムの通信定義と関連づける内容」で説明します。

TP1/NET/OSAS-NIF 固有のシステム定義の種類

TP1/NET/OSAS-NIF に固有の定義を次の表に示します。

表 6-2　TP1/NET/OSAS-NIF 固有の定義の種類

定義名		コマンド	オプション・オペランド		定義内容	指定値((値範囲))《省略時解釈値》
MCF マネジャ定義		mcfmcomn	-l	−	MCF メッセージ回復用作業領域長	符号なし整数((0～524288))《0》
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定義できます。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	mcftpcmn（プロトコル共通定義）指定数：1	-e	termactb	コネクション確立処理中のオンライン終了時動作	《wait》 | cont
			-r	autoonln	着呼型コネクションの自動受付開始機能を使用するかどうかを指定	yes | 《no》
				errdetect	過着呼による障害検出機能を使用するかどうかを指定	yes | 《no》
		mcftalccn（コネクション定義の開始）指定数：1～512※	-c	−	コネクション ID	1～8 文字の識別子
			-p	−	プロトコルの種別	onf
			-n	−	自システムの PSAP アドレス	1～142 文字の 16 進数字
			-q	−	相手システムの PSAP アドレス	1～186 文字の 16 進数字
			-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))
			-m	mode	使用する通信管理	xnfas
			-t	−	コネクションの種別	《int》 | rsp
			-i	−	コネクションの確立方法	auto | 《manual》
			-b	bretry	コネクション確立障害時の確立再試行をするかどうかを指定	《yes》 | no
				bretrycnt	コネクション確立障害時の確立再試行回数	符号なし整数((0～65535))《0》（単位：回）
				bretryint	コネクション確立障害時の確立再試行間隔	符号なし整数((0～2550))《60》（単位：秒）

定義名		コマンド	オプション・オペランド		定義内容	指定値((値範囲))《省略時解釈値》
MCF通信構成定義	プロトコル固有定義	mcftalccn（コネクション定義の開始）指定数：1～512※	-v	tim1	正常処理監視タイマ値	符号なし整数((0，10～8191))《60》（単位：秒）
				tim2	ユーザ処理監視タイマ値	符号なし整数((0，10～8191))《60》（単位：秒）
				tim3	送達確認監視タイマ値	符号なし整数((0，10～8191))《60》（単位：秒）
				tim4	連続送信監視タイマ値	符号なし整数((0，10～8191))《60》（単位：秒）
				tim5	終了処理監視タイマ値	符号なし整数((0，10～65535))《300》（単位：秒）
			-o	ownsid	自システムのシステムID	16進数字((0001～ffff))
				otrsid	相手システムのシステムID	16進数字((0001～ffff))
			-y	nummax	NIF通番最大値	符号なし整数((1～4294967295))《32767》
			-x	embed	コネクションの確立方式	《yes》 | no
				agentnum	エージェント番号のオクテット数指定方式	《asn1》 | fixed
			-z	slot	仮想スロット番号	符号なし整数((0～65535))
		mcftalcle（論理端末定義）指定数1～33（重畳型）1～2048（非重畳型）	-l	－	論理端末名称	1～8文字の識別子
			-t	－	論理端末の端末タイプ	request | reply | send | receive
			-m	mmsgcnt	メモリ出力メッセージ最大格納数	符号なし整数((0～65535))《0》
				dmsgcnt	ディスク出力メッセージ最大格納数	符号なし整数((0～65535))《0》
			-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
			-r	repr	代表論理端末にするかどうかを指定	yes | 《no》
			-d	sync	同期型を使用するかどうかを指定	yes | 《no》
				nugua	NIF通番を引き継ぐかどうかを指定	《yes》 | no
				rplytim	応答監視タイマ値	符号なし整数((0～8191))《0》（単位：秒）
		mcftalced（コネクショ	－	－	コネクション定義の終了	－

定義名		コマンド	オプション・オペランド		定義内容	指定値((値範囲))《省略時解釈値》
MCF通信構成定義	プロトコル固有定義	ン定義の終了）指定数：mcftalccnと同数	－	－	コネクション定義の終了	－
MCFアプリケーション定義		mcfaalcap（アプリケーション属性定義）	-n	lname	論理端末名称	1～8文字の識別子
				cname	コネクションID	1～8文字の識別子

（凡例）

－：該当しません。

注※

定義できる最大値です。同時接続できる最大数を求める計算式を次に示します。なお，ソケット用ファイル記述子の最大数については，「システムサービス共通情報定義」を参照してください。

同時接続数≦256-（定義コネクション数×2＋ソケット用ファイル記述子の最大数＋MCFメイン関数でユーザが使用するファイル記述子数＋30）

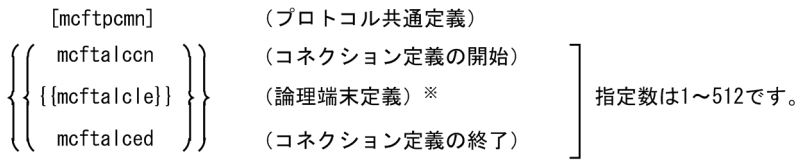
注

TP1/NET/OSAS-NIF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，およびオペランドがあります。それらについては，マニュアル「OpenTP1 システム定義」を参照してください。

定義の指定順序

TP1/NET/OSAS-NIF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。

図 6-1　TP1/NET/OSAS-NIFのプロトコル固有定義コマンドの指定順序



注※

mcftalccnとmcftalcleの指定は，1対n（n：重畳型の場合は1～33，非重畳型の場合は1～2048）です。

mcfmcomn（MCF マネジャ共通定義）

形式

```
mcfmcomn                :
          〔-l MCFメッセージ回復用作業領域長〕
                        :
```

機能

メッセージの回復を保証するための作業領域長を定義します。

オプション

● -l MCF メッセージ回復用作業領域長～〈符号なし整数〉((0～524288))《0》

TP1/NET/OSAS-NIF を使用する場合，メッセージの回復を保証するための作業領域長を指定します。

作業領域長の計算式を次に示します。

```
↑論理端末数×192＋192↑1024＋256
```

（凡例） ↑ ↑1024：1024 ごとに切り上げます。

指定例

論理端末数が六つの場合の MCF マネジャ共通定義（mcfmcomn）の例を次に示します。

```
###   MCFマネジャ共通定義
mcfmcomn  -n   300                ¥
          -p   100                ¥
          -l   1600
```

mcftpcmn（プロトコル共通定義）

形式

```
〔mcftpcmn    〔-e ”〔termactb=wait|cont〕 ”〕
              〔-r ”〔autoonln=yes|no〕
                   〔errdetect=yes|no〕 ”〕 〕
```

機能

TP1/NET/OSAS-NIF に関する環境を定義します。

オプション

● -e

（オペランド）

termactb=wait ｜ cont　～《wait》

コネクション確立処理中のオンライン終了時動作を指定します。

wait

コネクション確立処理が完了するまで終了処理を待ち合わせます。

cont

コネクション確立処理を中断し，終了処理を続行します。

● -r

（オペランド）

autoonln=yes ｜ no～《no》

着呼型コネクションの自動受付開始機能を使用するかどうかを指定します。

yes

着呼型コネクションの自動受付開始機能を使用します。

no

着呼型コネクションの自動受付開始機能を使用しません。

yes を指定した場合，プロトコル共通定義（mcftpcmn -e）の termactb オペランドに cont を指定する必要があります。wait を指定，または省略した場合，定義オブジェクト生成時にエラーとなります。

errdetect=yes ｜ no～《no》

過着呼による障害検出機能を使用するかどうかを指定します。

yes

過着呼による障害検出機能を使用します。

no

過着呼による障害検出機能を使用しません。

yes を指定した場合，自システムの PSAP アドレス（mcftalccn -n）が同一のコネクションで，次のオプションおよびオペランドの指定値を一致させる必要があります。指定値が一致していない場合，定義オブジェクト生成時にエラーとなります。

- メッセージ送信用バッファのグループ番号（mcftalccn -g sndbuf）
- メッセージ受信用バッファのグループ番号（mcftalccn -g rcvbuf）
- コネクション確立の発呼と着呼の識別（mcftalccn -t）

no を指定，または，省略した場合，すべての着呼型のコネクションの自システムの PSAP アドレスを一意にしてください。自システムの PSAP アドレスが重複している着呼型コネクションが存在する場合，相手システムからの確立要求を拒否することがあります。

mcftalccn（コネクション定義の開始）

形式

```
mcftalccn -c コネクションID
          -p onf
          -n x'自システムのPSAPアドレス'
          -q x'相手システムのPSAPアドレス'
          -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"〕
          -m "mode=xnfas"
          〔-t int｜rsp〕
          〔-i auto｜manual〕
          〔-b "〔bretry=yes｜no〕
                〔bretrycnt=コネクション確立障害時の確立再試行回数〕
                〔bretryint=コネクション確立障害時の確立再試行間隔〕"〕
          〔-v "〔tim1=正常処理監視タイマ値〕
                〔tim2=ユーザ処理監視タイマ値〕
                〔tim3=送達確認監視タイマ値〕
                〔tim4=連続送信監視タイマ値〕
                〔tim5=終了処理監視タイマ値〕"〕
          -o "ownsid=x'自システムID'
              otrsid=x'相手システムID'"
          〔-y "〔nummax=NIF通番最大値〕"〕
          〔-x "〔embed=yes｜no〕
                〔agentnum=asn1｜fixed〕"〕
          -z "slot=仮想スロット番号"
```

機能

コネクションに関する環境を定義します。

オプション

● -c コネクション ID　～〈識別子〉((1～8 文字))

OpenTP1 システム内で，一意となるコネクション ID を指定します。

● -p onf

プロトコルの種別を指定します。

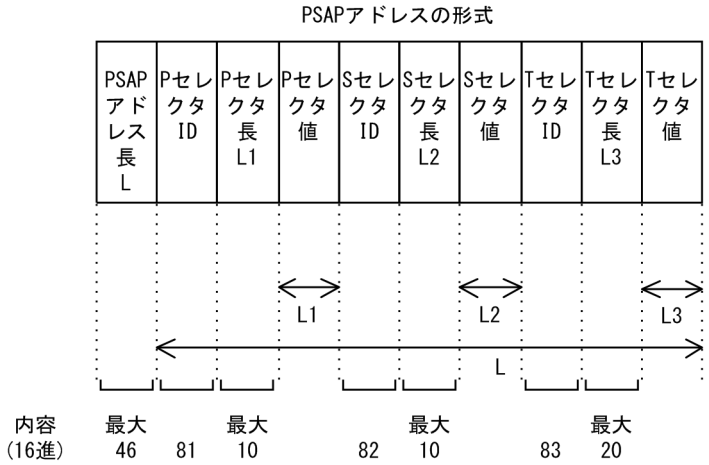
onf

NIF/OSI プロトコル

● -n x'自システムの PSAP アドレス'　～〈1～142 文字の 16 進数字〉

自システムの PSAP アドレスを指定します。

自システムの PSAP アドレスの形式を次の図に示します。

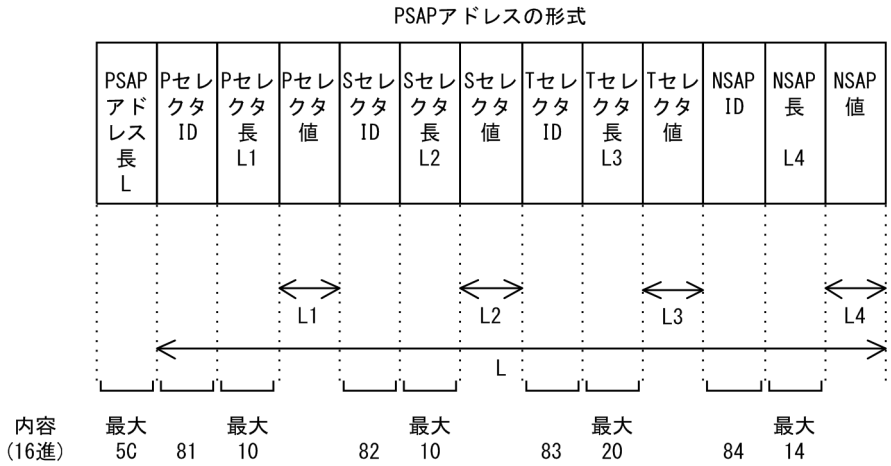


ほかのコネクションが使用する PSAP アドレスと重複しない PSAP アドレスを指定してください。

● -q x'相手システムの PSAP アドレス'　～〈1～186 文字の 16 進数字〉

相手システムの PSAP アドレスを指定します。

相手システムの PSAP アドレスの形式を次に示します。なお，NSAP 値の形式については，マニュアル「通信管理 XNF/AS NSAP アドレス概説編」を参照してください。



-n オプションで指定する自システムの PSAP アドレスと，-q オプションで指定する相手システムの PSAP アドレスの組み合わせは，ほかのコネクションが使用する組み合わせと重複しないでください。

● -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))

メッセージ送信用バッファのグループ番号を指定します。

バッファグループ定義（mcftbuf -g）の groupno オペランドに指定されているバッファグループ番号を指定してください。

rcvbuf=メッセージ受信用バッファグループ番号　～〈符号なし整数〉((1～512))

メッセージ受信用バッファのグループ番号を指定します。

バッファグループ定義（mcftbuf -g）の groupno オペランドに指定されているバッファグループ番号を指定してください。

● -e

（オペランド）

msgbuf=メッセージ編集用バッファグループ番号　～〈符号なし整数〉((1～512))

入力，出力メッセージ編集 UOC 呼び出し時に，メッセージ編集用として使用するバッファグループ番号を指定します。

このオペランドを省略した場合は，メッセージ編集用バッファは確保されません。メッセージ編集用バッファグループ番号は，バッファグループ定義（mcftbuf -g）の groupno オペランドで指定するバッファグループ番号を指定してください。

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

入力，出力メッセージ編集 UOC 呼び出し時に，メッセージ編集用として使用するバッファの数を指定します。

msgbuf オペランドで指定するメッセージ編集用バッファグループ番号に対応するバッファグループ定義（mcftbuf -g）の count オペランドおよび extend オペランドで指定するバッファ数の中から，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。

メッセージ編集用バッファグループ番号に対応する mcftbuf の count オペランドおよび extend オペランドには，すべての論理端末で入力メッセージ編集 UOC および出力メッセージ編集 UOC が，同時に動作した場合を考慮した面数を指定してください。

また，このオペランドの指定は，バッファグループ定義（mcftbuf -g）の count オペランドおよび extend オペランドで指定されたバッファ数の合計値を超える指定はできません。

msgbuf オペランドを省略した場合は，このオペランドの指定は無効です。

msgbuf オペランドを指定した場合，このオペランドを省略できません。省略した場合，定義オブジェクト生成時に KFCA11519-E メッセージを出力してエラーとなります。

● -m

（オペランド）

mode=xnfas

使用する通信管理を次の値で指定します。

このオペランドは必ず指定してください。

xnfas

XNF/AS

● -t int | rsp　～《int》

コネクション確立時の発呼と着呼の識別を指定します。

int

自システムが発呼側

rsp

自システムが着呼側

● -i auto | manual 〜《manual》

オンライン開始時にコネクションを自動的に確立するかどうかを指定します。

auto

オンライン開始時にコネクションを自動的に確立します。

manual

MCF 起動後，コネクションを確立します。コネクションの確立は，運用コマンド（mcftactcn）の入力，または API（dc_mcf_tactcn 関数もしくは CBLDCMCF('TACTCN△△')）の発行で行います。

● -b

(オペランド)

bretry=yes | no 〜《yes》

コネクション確立時に障害が発生した場合，コネクション確立再試行をするかどうかを指定します。

yes

コネクション確立再試行をします。

no

コネクション確立再試行をしません。

bretrycnt=コネクション確立障害時の確立再試行回数 〜〈符号なし整数〉((0〜65535))《0》(単位：回)

コネクションの確立再試行をする場合の確立再試行回数を指定します。このオペランドを省略した場合，または 0 を指定した場合，確立再試行を無限回繰り返します。

bretry オペランドで no を指定した場合，bretrycnt オペランドの指定は無効になります。

bretryint=コネクション確立障害時の確立再試行間隔 〜〈符号なし整数〉((0〜2550))《60》(単位：秒)

コネクションの確立再試行をする場合の確立再試行間隔を指定します。0 を指定した場合，直ちにコネクションの確立再試行をします。

bretry オペランドで no を指定した場合，bretryint オペランドの指定は無効になります。

注意事項

再試行間隔の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔で再試行するかどうかをチェックします。このため，このオペランドで指定した再試行間隔と実際に再試行する時間には秒単位の誤差が生じます。

● **-v**

（オペランド）

tim1=正常処理監視タイマ値　～〈符号なし整数〉((0,10～8191))《60》（単位：秒）

正常処理監視タイマ値を指定します。

0 を指定した場合，正常処理監視をしません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。

tim2=ユーザ処理監視タイマ値　～〈符号なし整数〉((0,10～8191))《60》（単位：秒）

ユーザ処理監視タイマ値を指定します。

0 を指定した場合，ユーザ処理監視をしません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。

tim3=送達確認監視タイマ値　～〈符号なし整数〉((0,10～8191))《60》（単位：秒）

送達確認監視タイマ値を指定します。

0 を指定した場合，送達確認監視をしません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。

tim4=連続送信監視タイマ値　～〈符号なし整数〉((0,10～8191))《60》（単位：秒）

連続送信監視タイマ値を指定します。

0 を指定した場合，連続送信監視をしません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。

tim5=終了処理監視タイマ値　～〈符号なし整数〉((0，10～65535))《300》（単位：秒）

終了処理監視タイマ値を指定します。

0 を指定した場合，終了処理監視をしません。

注意事項

監視時間の精度は秒単位です。また，タイマ定義（mcfttim -t）の btim オペランドで指定する時間の間隔でタイムアウトが発生したかどうかを監視しています。このため，このオペランドで指定した監視時間と実際にタイムアウトを検出する時間には秒単位の誤差が生じます。そのため，タイミングによっては，指定した監視時間よりも短い時間でタイムアウトすることがあります。

● -o

（オペランド）

ownsid=x'自システム ID'　～〈16 進数字〉((0001～ffff))

自システムのシステム ID をネットワーク内で一意に指定します。

otrsid=x'相手システム ID'　～〈16 進数字〉((0001～ffff))

相手システムのシステム ID をネットワーク内で一意に指定します。

● -y

（オペランド）

nummax=NIF 通番最大値　～〈符号なし整数〉((1～4294967295))《32767》

通番管理機能で使用する NIF 通番の最大値を指定します。

● -x

（オペランド）

embed=yes | no　～《yes》

コネクションの確立方式を指定します。

ただし，-t オプションで rsp を指定した場合，このオペランドの指定は無効になります。

yes

重畳型（OSI 構造の層の中の ACSE と NIF/OSI を同時に確立します。重畳型で接続できる端末の台数は，1～33 台です）

no

非重畳型（OSI 構造の層の中の ACSE を確立してから NIF/OSI を確立します。非重畳型で接続できる端末の台数は，1～2048 台です）

agentnum=asn1 | fixed　～《asn1》

NIF ヘッダに設定するエージェント番号のオクテット数を指定します。

このオペランドの指定を省略した場合，asn1 に設定されます。

asn1

ASN.1 のエンコード規則に従い，最小のオクテット数に設定します。

fixed

2 オクテット固定で設定します。

● -z

（オペランド）

slot=仮想スロット番号　～〈符号なし整数〉((0～65535))

仮想スロット番号を指定します。

XNF/AS の通信構成定義で指定する仮想スロット番号を指定してください。

このオペランドは必ず指定してください。

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（mcftbuf）で指定したバッファグループ番号に対応させてください。mcftbuf 定義コマンドでコネクションごとに割り当てる資源の量を次の表に示します。バッファグループ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。

表 6-3　mcftbuf 定義コマンドでコネクションごとに割り当てる資源の量

mcftalccn コマンド		バッファグループ定義　mcftbuf コマンド　-g オプション	
		length　※1	count + extend　※2
-g	sndbuf	論理端末数×10 または コネクション間で最大のメッセージ長どちらかの最大値	論理端末数×（最大セグメント分割数＋ 1）
	rcvbuf	論理端末数×10 または コネクション間で最大のメッセージ長どちらかの最大値	論理端末数×（最大セグメント分割数＋ 1）＋ 2※3
-e	msgbuf	ユーザ任意値（入力メッセージ編集 UOC 出力メッセージ編集 UOC で編集したメッセージを格納できるサイズ）	編集用バッファ数

注※1

length オペランドに指定できる最大値-256 未満の値を指定してください。また，複数コネクションでバッファグループを共用する場合，コネクション間での最大の値を指定します。

注※2

複数コネクションでバッファグループを共用する場合，コネクションごとの和を指定します。

注※3

過着呼による障害検出を使用する場合，着呼型のコネクションごとに加算してください。

TP1/NET/OSAS-NIF は，オンライン開始時に送受信バッファが不足していないかチェックします。バッファが不足している場合，KFCA13525-W メッセージが出力されます。

なお，オンライン開始時のチェックでは，送受信するメッセージが単一セグメント（最大セグメント分割数が 0）と仮定します。このため，複数セグメントのメッセージを送受信するのに，表 6-3 に示した値よ

り小さい値をバッファグループ定義に指定すると，オンライン開始時に KFCA13525-W メッセージが出力されないで，オンライン中にバッファ不足が発生することがあるので，注意してください。

mcftalcle（論理端末定義）

形式

```
mcftalcle -l 論理端末名称
          -t request | reply | send | receive
         〔-m ”〔mmsgcnt=メモリ出力メッセージ最大格納数〕
              〔dmsgcnt=ディスク出力メッセージ最大格納数〕 ”〕
          -k ”〔quekind=memory | disk〕
              〔quegrpid=キューグループID〕 ”
         〔-o ”〔aj=yes | no〕 ”〕
         〔-r ”〔repr=yes | no〕 ”〕
         〔-d ”〔sync=yes | no〕
              〔nugua=yes | no〕
              〔rplytim=応答監視タイマ値〕 ”〕
```

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。

● -t request | reply | send | receive

この論理端末の端末タイプを指定します。

request
: request 型論理端末または request 型論理端末（同期型）

reply
: reply 型論理端末

send
: send 型論理端末

receive
: receive 型論理端末

なお，request 型論理端末（同期型）を指定する場合，-d オプションの sync オペランドに yes を指定してください。

● -m

（オペランド）

mmsgcnt=メモリ出力メッセージ最大格納数　～〈符号なし整数〉((0～65535))《0》

メモリキューで待ち合わせ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，それ以降 UAP からの送信要求（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 を指定した場合，または省略した場合，メモリキューで待ち合わせをする出力メッセージの数は指定可能な最大数（65535）となります。ただし，実際に待ち合わせをできる出力メッセージ数は動的共用メモリの容量に依存します。

dmsgcnt=ディスク出力メッセージ最大格納数　～〈符号なし整数〉((0～65535))《0》

ディスクキューで待ち合わせをする出力メッセージの最大格納数を指定します。

出力メッセージの待ち合わせ数が指定した最大数になると，それ以後 UAP からの送信要求（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 を指定した場合，または省略した場合，ディスクキューで待ち合わせをする出力メッセージの数は指定可能な最大数（65535）となります。ただし，実際に待ち合わせをできる出力メッセージ数はメッセージキューファイルの容量に依存します。

● -k

（オペランド）

quekind=memory | disk　～《disk》

送受信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。

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をする出力メッセージに使用するキューグループ ID を指定します。

入出力キュー定義（mcfmqgid）で指定するキューグループ ID（キューグループ種別は 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 を指定した場合は省略できません。

● -o

（オペランド）

aj=yes | no　～《yes》

送信完了時の情報を取得するかどうかを指定します。

yes

取得します。

no

取得しません。

● -r

（オペランド）

repr=yes | no　～《no》

該当する論理端末を代表論理端末とするかどうかを指定します。

代表論理端末は，コネクションに関するイベントを受信するための論理端末です。

一つのコネクション内に複数の論理端末がある場合，一つの論理端末定義でだけ，yes を指定できます。このオペランドの指定がすべて no の場合，最初に定義した論理端末が代表論理端末になります。

yes

該当する論理端末定義の論理端末を代表論理端末とします。

no

該当する論理端末定義の論理端末を代表論理端末としません。

● -d

（オペランド）

sync=yes | no　～《no》

request 型論理端末を同期型として使用するかどうかを指定します。-t オプションに request を指定した場合だけ有効です。

このオペランドの指定を省略した場合，no に設定されます。

yes

同期型として使用します。

no

同期型として使用しません。

nugua=yes | no　～《yes》

MCF の再開始時に，NIF 通番を引き継ぐかどうかを指定します。

-k オプションの quekind オペランドに memory を指定した場合，有効となります。

このオペランドの指定を省略した場合，yes に設定されます。

yes

NIF 通番を引き継ぎます。

no

NIF 通番を引き継ぎません。

なお，no を指定した場合，MCF の再開始時には，NIF 通番はリセット状態となり，再送のための情報は失われます。

rplytim=応答監視タイマ値　　～〈符号なし整数〉((0～8191))《0》（単位：秒）

問い合わせメッセージを受信して，reply 型論理端末で起動した MHP からの応答メッセージの送信を受け付けるまでの監視時間を秒単位で指定します。-t オプションに reply を指定した場合だけ有効です。

次に示す契機から応答メッセージの送信を受け付けるまでの時間を監視します。

- 相手システムから問い合わせメッセージを受信して入力キューへ登録したとき
- MHP からの応答メッセージ送信待ちの状態で，相手システムから通番問い合わせ要求を受信したとき
- MHP からの応答メッセージを送信待ちの状態で，相手システムから再送された問い合わせメッセージを受信したとき

問い合わせメッセージを受信して，監視時間内に MHP からの応答メッセージ送信を受け付けない場合，該当するメッセージの送受信の処理を打ち切り，次メッセージを処理します。

通番問い合わせ要求または再送された問い合わせメッセージを受信して，監視時間内に MHP から応答メッセージの送信を受け付けない場合，該当するメッセージは未受信として処理します。

このオペランドの指定を省略した場合，または 0 を指定した場合，応答監視はしません。

なお，監視時間の指定はコネクション定義（mcftalccn -v）の tim1 オペランドまたは tim2 オペランドの指定値より小さな値を指定してください。

注意事項

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

mcftalced（コネクション定義の終了）

形式

```
mcftalced
```

機能

コネクションの定義の終了を示します。

オプション

ありません。

mcfaalcap（アプリケーション属性定義）

形式

```
mcfaalcap -n ”name=相手システムのアプリケーション名”
                :
              [lname=自システムの論理端末名称]
              [cname=コネクションID]
                :
```

機能

EXECAP（相手システムのアプリケーションの起動）要求を発行する場合に，相手システムのアプリケーション名と次に示すどちらかをあわせて指定します。

- 自システムの論理端末名称
- 自システムのコネクション ID

オプション

● -n

（オペランド）

name=相手システムのアプリケーション名

EXECAP 要求を発行するときの相手システムのアプリケーション名を指定します。ただし，アプリケーション名には英小文字および_（アンダスコア）は指定できません。

lname=自システムの論理端末名称

EXECAP 要求を発行する場合に，name オペランドで指定した相手システムのアプリケーションとメッセージの送受信をする自システムの論理端末名称を指定します。

論理端末名称は，TP1/NET/OSAS-NIF の論理端末定義（mcftalcle）で定義した request 型論理端末または send 型論理端末を指定してください。

cname=自システムのコネクション ID

EXECAP 要求を発行する場合に，name オペランドで指定した相手システムのアプリケーションと通信する自システムのコネクション ID を指定します。コネクション下には，一つ以上の request 型論理端末を定義してください。

ただし，type オペランドで ans または cont を指定したときだけ指定できます。-g オプションの type オペランドに SPP を指定したときは指定できません。

注意事項

EXECAP 要求で相手システムのアプリケーションを起動する場合，次に示す項目に注意してください。

1. TP1/NET/OSAS-NIF の MCF 通信構成定義に対応する mcfaalcap コマンドで次の項目を指定してください。ただし，自システム内でアプリケーション起動機能を使用する場合は，アプリケーション起動プロセスの MCF 通信構成定義に対応する mcfaalcap コマンドで次の項目を指定してください。
   - 相手システムのアプリケーション名
   - 論理端末名称またはコネクション ID

   アプリケーション名には英小文字および_（アンダスコア）は指定できません。論理端末名称を指定する場合，request 型論理端末または send 型論理端末の論理端末名称を指定してください。コネクション ID を指定する場合，コネクション下に一つ以上の request 型論理端末を定義してください。
2. 相手システム側で起動するアプリケーションを定義する場合，サービスグループ名，サービス名は意味を持ちません。任意の名称を指定してください。
3. mcfaalcap -g quekind で指定するメッセージの割り当て先は，必ず disk を指定してください。quekind オペランドの指定を省略した場合，またはディスクキューからの縮退処理によってメモリキューを使用している場合は，メッセージの再送はしません。

EXECAP 要求を発行する起動元のアプリケーションプログラムと，起動先のアプリケーション属性定義との関係を次の表に示します。

次の表に示す以外の関係で，EXECAP 要求を発行した場合はエラーとなります。

表 6-4　起動元のアプリケーションプログラムと起動先のアプリケーション属性定義との関係

起動元のアプリケーションプログラム	起動先のアプリケーション属性定義（mcfaalcap）		
	−n		
	lname	cname	type
任意	send 型論理端末を指定します。	指定しません。	noans
ans 型 MHP	request 型論理端末を指定します。		ans
noans 型 MHP または SPP			
cont 型 MHP			cont
ans 型 MHP	指定しません。	request 型論理端末が定義してあるコネクション ID を指定します。	ans
noans 型 MHP または SPP			
cont 型 MHP			cont

指定例

```
###   アプリケーション属性定義(OSASNF)
mcfaalcap   -n  ”name=APL01                ¥
                 kind=user                 ¥
                 type=noans                ¥
                 msgcnt=100                ¥
                 lname=NFLE01”             ¥
            -g  ”servgrpn=srvgrp01         ¥
```

```
                quegrpid=quegrp02        ¥
                quekind=disk"            ¥
          -v    "servname=srv01"
mcfaalcap -n    "name=APL02              ¥
                kind=user                ¥
                type=ans                 ¥
                msgcnt=100               ¥
                cname=cnct01"            ¥
          -g    "servgrpn=srvgrp01       ¥
                quegrpid=quegrp02        ¥
                quekind=disk"            ¥
          -v    "servname=srv02"
```

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/OSAS-NIF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

● module="TP1/NET/OSAS-NIF 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 通信プロセス同士および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/OSAS-NIF 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/OSAS-NIF 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 システム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/OSAS-NIF の場合，コネクションの総数を 3 倍した値になります。実際に通信を行うコネクションの総数ではありませんので，注意してください。

注※2

MCF 通信プロセスが扱う定義ファイルなどの数の最大値です。

自 OpenTP1 内の MCF 通信プロセスごとに計算し，その結果の中で最大値が 500 より大きい場合は，その値を指定します。500 以下の場合は，500 を指定します。指定値を超えてファイルのアクセスが発生した場合，その超過分は，ソケット用ファイル記述子使用数として扱われます。この場合，「max_socket_descriptors オペランドの指定値－max_open_fds オペランドの指定値の超過分」が，実際のソケット用ファイル記述子の最大数になりますので，ご注意ください。

max_socket_descriptors オペランドと max_open_fds オペランドには次の条件を満たす値を指定してください。

```
（「max_socket_descriptorsオペランドの指定値」
＋「max_open_fdsオペランドの指定値」）≦4096
```

ただし，TP1/NET/OSAS-NIF 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 2048 です。

TP1/NET/OSAS-NIF 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/OSAS-NIF 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/OSAS-NIF 固有の出力情報や取得タイミングについては，「付録 F　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，上記の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

```
mcfosnf -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/OSAS-NIF 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/OSAS-NIF 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

形式

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

機能

MCF 通信構成定義のプロトコル（TP1/NET/OSAS-NIF）固有定義オブジェクト，または，これと共通定義オブジェクトとを結合したオブジェクトを解析し，定義ソースのイメージで標準出力します。

解析対象が不正であった場合，オブジェクトの解析および出力ができないことがあります。

オプション

● -i 〔パス名〕 解析対象オブジェクトファイル名　～〈パス名〉〈1～8 文字の英数字〉

解析する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

定義オブジェクトの解析後の出力例を次に示します。

なお，値を指定していない場合は，省略値が表示されます。

```
#####################################################################
      MCF communication configuration definition
                 OSAS-NIF definition
#####################################################################
   OBJECT FILE NAME : XXXXXXXX
   VV-RR            : xx-xx
   DATE             : yyyy-mm-dd hh:mm:ss
#####################################################################

mcftalccn
  -c           = cnct01
  -p           = onf
  -n           = x'0a81008202000283020003'
  -q           = x'178100820200038302000384 0b490001020000000000fe01'
  -g sndbuf    = 1
  -g rcvbuf    = 2
  -e msgbuf    = 3
  -e count     = 14
  -m mode      = xnfas
  -t           = int
  -i           = manual
  -b bretry    = yes
  -b bretrycnt = 20
  -b bretryint = 5
  -v tim1      = 60
  -v tim2      = 60
  -v tim3      = 60
```

```
  -v tim4      = 60
  -v tim5      = 60
  -o ownsid    = x'0001'
  -o otrsid    = x'ffff'
  -y nummax    = 20
  -x embed     = yes
  -x agentnum  = asn1
* -x rnodetr   = yes
* -x acname    = x'816fa973'
* -x timset    = yes
* -x userpi    = e1
  -z slot      = 1

mcftalcle
  -l           = NFLE01
  -t           = send
  -m mmsgcnt   = 0
  -m dmsgcnt   = 0
  -k quekind   = disk
  -k quegrpid  = quegrp01
  -o aj        = yes
  -r repr      = no
  -d sync      = no
  -d nugua     = yes
  -d rplytim   = 0
* -d ctlsend   = no
* -d nuupmode  = normal

mcftalcle
  -l           = NFLE02
  -t           = request
  -m mmsgcnt   = 0
  -m dmsgcnt   = 0
  -k quekind   = disk
  -k quegrpid  = quegrp01
  -o aj        = yes
  -r repr      = no
  -d sync      = no
  -d nugua     = yes
  -d rplytim   = 0
* -d ctlsend   = no
* -d nuupmode  = normal

mcftalced

########################## End Of File ############################
```

注

先頭に*が付いている行は，限定公開機能の定義です。

メッセージキューサービス定義

TP1/NET/OSAS-NIF を使用する場合，メッセージの再読み出しをするため，メッセージキューサービス定義（quegrp）の-m オプションに，メッセージキューファイルに保持するメッセージ数を指定してください。

メッセージキューサービス定義（quegrp）の例を次に示します。

```
#que : メッセージキューサービス定義
quegrp -g quegrp01 -f /dev/rdsk/rhd011/quef01 -n 256 -m 1
quegrp -g quegrp02 -f /dev/rdsk/rhd011/quef02 -n 256 -m 1
set que_xidnum   = 256
```

メッセージキューサービス定義については，マニュアル「OpenTP1 システム定義」を参照してください。

自システムの通信管理プログラム（XNF/AS）と関連づける内容

相手システムとの接続時に使用したい機能によって，使用する通信管理プログラムは次のとおり異なります。

- OSI 拡張機能を使用する場合
  XNF/AS/BASE
  XNF/AS/OSI Extension
- 自局 IP アドレス指定機能を使用する場合
  XNF/AS/BASE
  XNF/AS/OSI Extension
  XNF/AS/OSI Extension/Cluster
- OSI 拡張高信頼化機能を使用する場合
  XNF/AS/BASE
  XNF/AS/OSI Extension
  XNF/AS/Host Adaptor

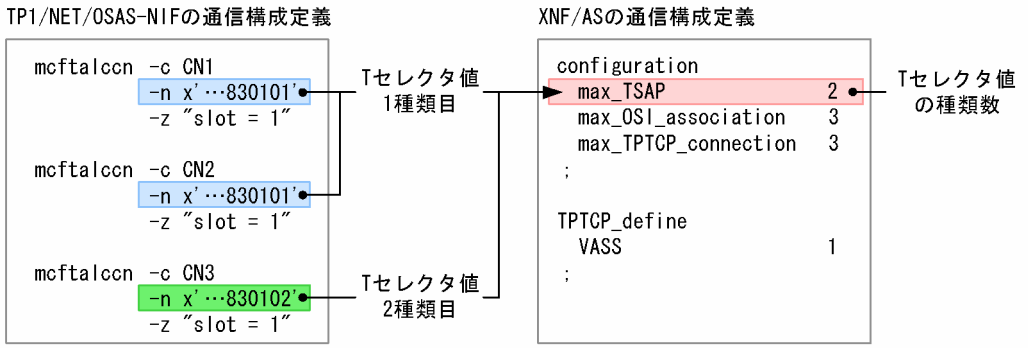
通信管理プログラムの詳細については，マニュアル「通信管理 XNF/AS 解説・運用編」およびマニュアル「通信管理 XNF/AS 構成定義編」を参照してください。

どの通信管理プログラムの機能を使用する場合でも，コネクション定義（mcftalccn）で指定する定義の内容と，XNF/AS のシステム定義時に指定する定義の内容とを関連づける必要があります。関連づける必要のある項目を次に示します。

- コネクション数
- T セレクタ値の種類数※
- 仮想スロット番号

注※

例えば，次の図に示すように三つの mcftalccn コマンドを定義した場合は，T セレクタ値の種類数である 2 を XNF/AS の通信構成定義と関連づけます。



関連づける項目は，使用する機能によって異なります。使用する機能別に，次に示します。

OSI 拡張機能を使用する場合

OSI 拡張機能を使用する場合は，次の表に示すとおりに定義を関連づけてください。

表 6-5　OSI 拡張機能を使用する場合の XNF/AS の定義との関連づけ

項番	TP1/NET/OSAS-NIF での定義内容		XNF/AS での定義内容	
	コマンド	内容	定義文	オペランド
1	mcftalccn	コネクション数（定義数）	configuration	max_OSI_association※ max_TPTCP_connection※
2	mcftalccn	T セレクタ値の種類数（-n オプションの T セレクタ値）		max_TSAP
3	mcftalccn	仮想スロット番号（-z オプションの slot オペランド）	TPTCP_define	VASS

注※

過着呼による障害検出を使用する場合，次の計算式の値を設定します。

着呼型コネクション数×3 ＋発呼型コネクション数

自局 IP アドレス指定機能を使用する場合

自局 IP アドレス指定機能を使用する場合は，次の表に示すとおりに定義を関連づけてください。

表 6-6　自局 IP アドレス指定機能を使用する場合の XNF/AS の定義との関連づけ

項番	TP1/NET/OSAS-NIF での定義内容		XNF/AS での定義内容	
	コマンド	内容	定義文	オペランド
1	mcftalccn	コネクション数（定義数）	configuration	max_OSI_association※ max_TPTCP_connection※
2		T セレクタ値の種類数（-n オプションの T セレクタ値）		max_TSAP
3		仮想スロット番号（-z オプションの slot オペランド）	TPTCP_slot	VASS

注※

過着呼による障害検出を使用する場合，次の計算式の値を設定します。

着呼型コネクション数×3 ＋発呼型コネクション数

OSI 拡張高信頼化機能を使用する場合

OSI 拡張高信頼化機能を使用する場合は，次の表に示すとおりに定義を関連づけてください。

表 6–7　OSI 拡張高信頼化機能を使用する場合の XNF/AS の定義との関連づけ

項番	TP1/NET/OSAS-NIF での定義内容		XNF/AS での定義内容	
	コマンド	内容	定義文	オペランド
1	mcftalccn	コネクション数（定義数）	configuration	max_OSI_association※ max_TPTCP_connection※
2		T セレクタ値の種類数（-n オプションの T セレクタ値）		max_TSAP
3		仮想スロット番号（-z オプションの slot オペランド）	TPTCP_VC	VASS

注※

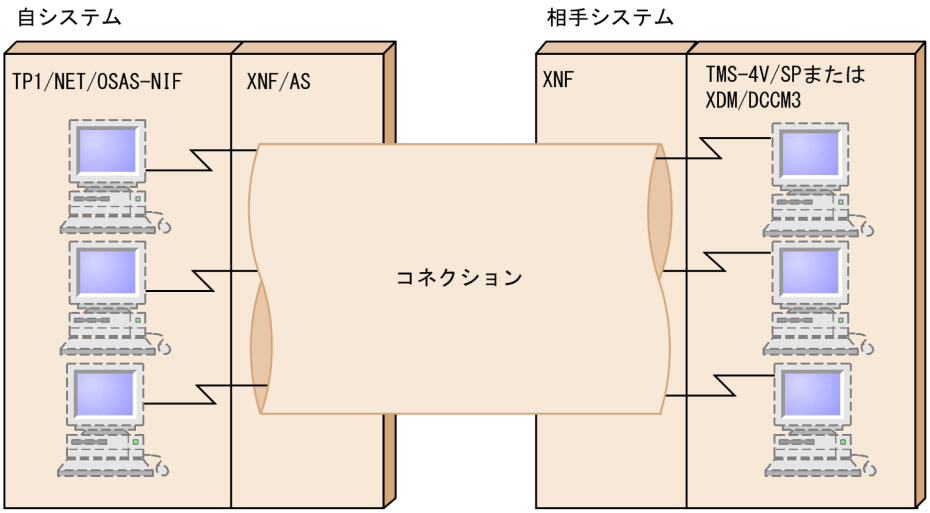
過着呼による障害検出を使用する場合，次の計算式の値を設定します。

着呼型コネクション数×3 ＋発呼型コネクション数

相手システムの通信定義と関連づける内容

TP1/NET/OSAS-NIF を使用する場合のネットワーク構成の例を次の図に示します。

図 6-2　ネットワーク構成の例



この例の場合，TP1/NET/OSAS-NIF は，XNF，および TMS-4V/SP または XDM/DCCM3 の定義と関連づける必要があります。

XNF の定義と関連づける内容

TP1/NET/OSAS-NIF のシステム定義と XNF の定義との関係を以降の図に示します。XNF の定義については，マニュアル「XNF TCP/IP 接続機能 XNF/TCP 定義とコマンド」を参照してください。

図 6-3　XNF の定義との関係（OSI 拡張機能または自局 IP アドレス指定機能を使用する場合）

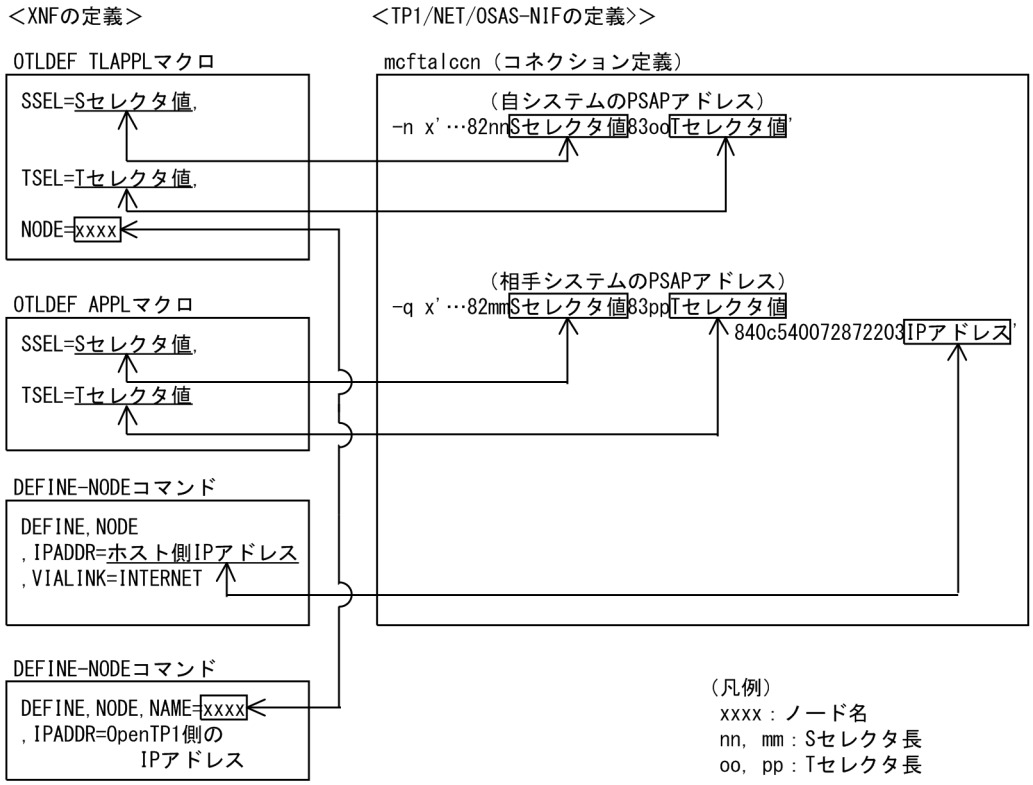
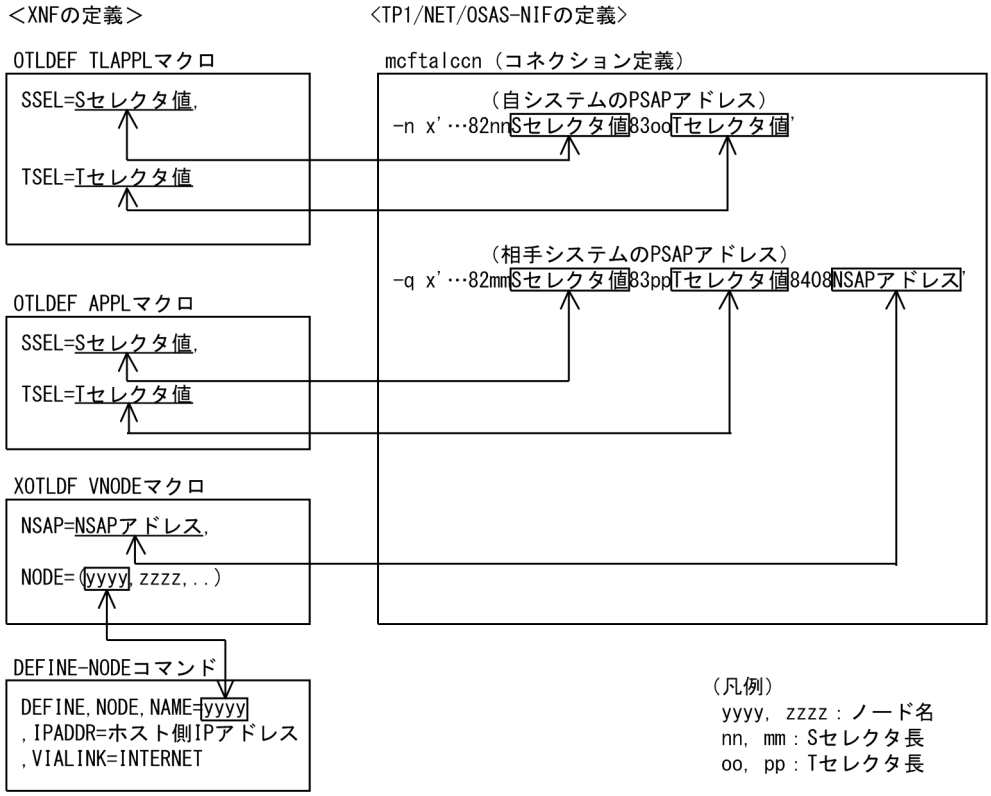


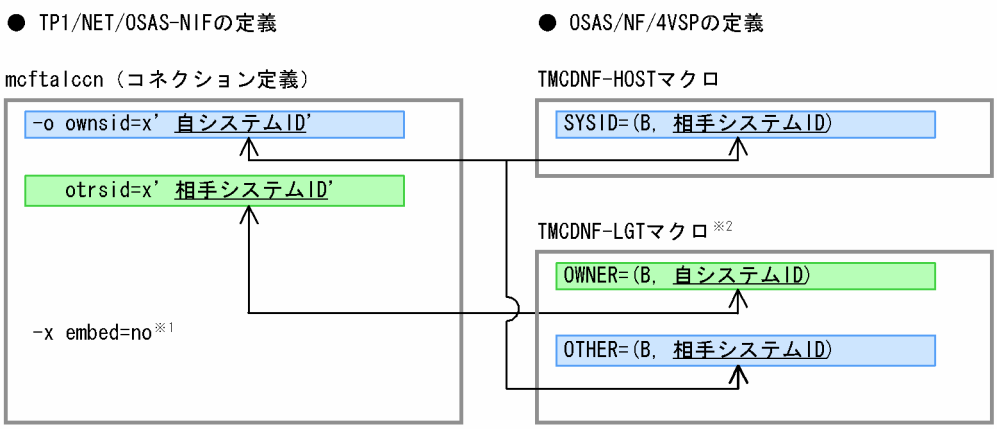
図 6-4　XNF の定義との関係（OSI 高信頼化機能を使用する場合）



TMS-4V/SP システム定義と関連づける内容

相手システムが TMS-4V/SP システムの場合の，TP1/NET/OSAS-NIF と OSAS/NF/4VSP の定義の関係を次の図に示します。OSAS/NF/4VSP の定義については，マニュアル「VOS3 TMS-4V/SP OSAS/AP 間通信サービス-NF OSAS/NF/4VSP」を参照してください。

図 6-5　OSAS/NF/4VSP の定義との関係



注※1

相手システムが TMS-4V/SP システムの場合，コネクションの確立方式には非重畳型を指定してください。

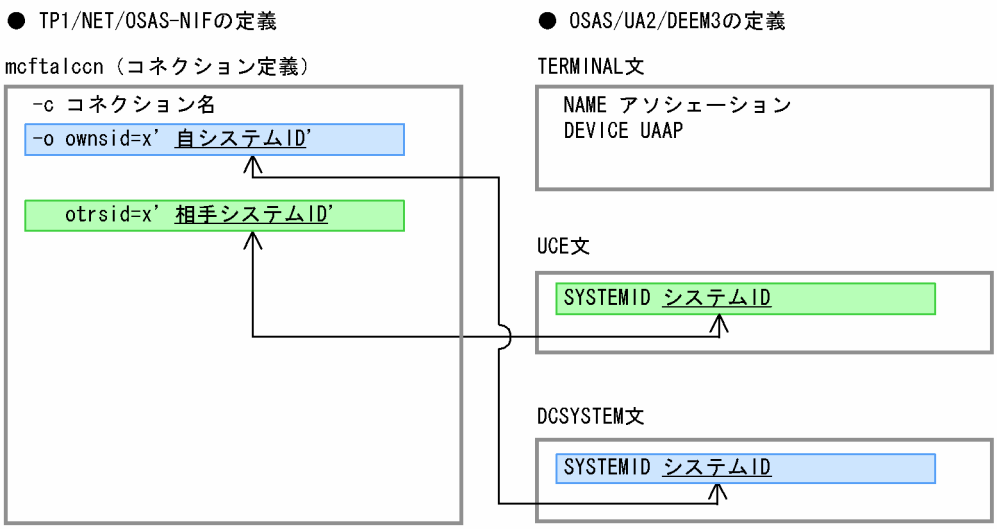
注※2

相手システムが TP1/NET/OSAS-NIF の場合，ホストノードに対して複数の実ノードを定義することはできません。

XDM/DCCM3 システム定義と関連づける内容

相手システムが XDM/DCCM3 システムの場合の TP1/NET/OSAS-NIF と OSAS/UA2/DCCM3 の定義の関係を次の図に示します。OSAS/UA2/DCCM3 の定義については，マニュアル「VOS3 OSI アプリケーション共通機能/AP 間通信サービス/DCCM3 OSAS/UA2/DCCM3」を参照してください。

図 6-6　OSAS/UA2/DCCM3 の定義との関係



アプリケーション起動機能を使用する場合に関連づける内容

EXECAP 要求によってアプリケーションを起動する場合，自システム内の次のプロセス間で定義する内容を関連づける必要があります。

- SPP または MHP
- アプリケーション起動プロセス
- MCF 通信プロセス

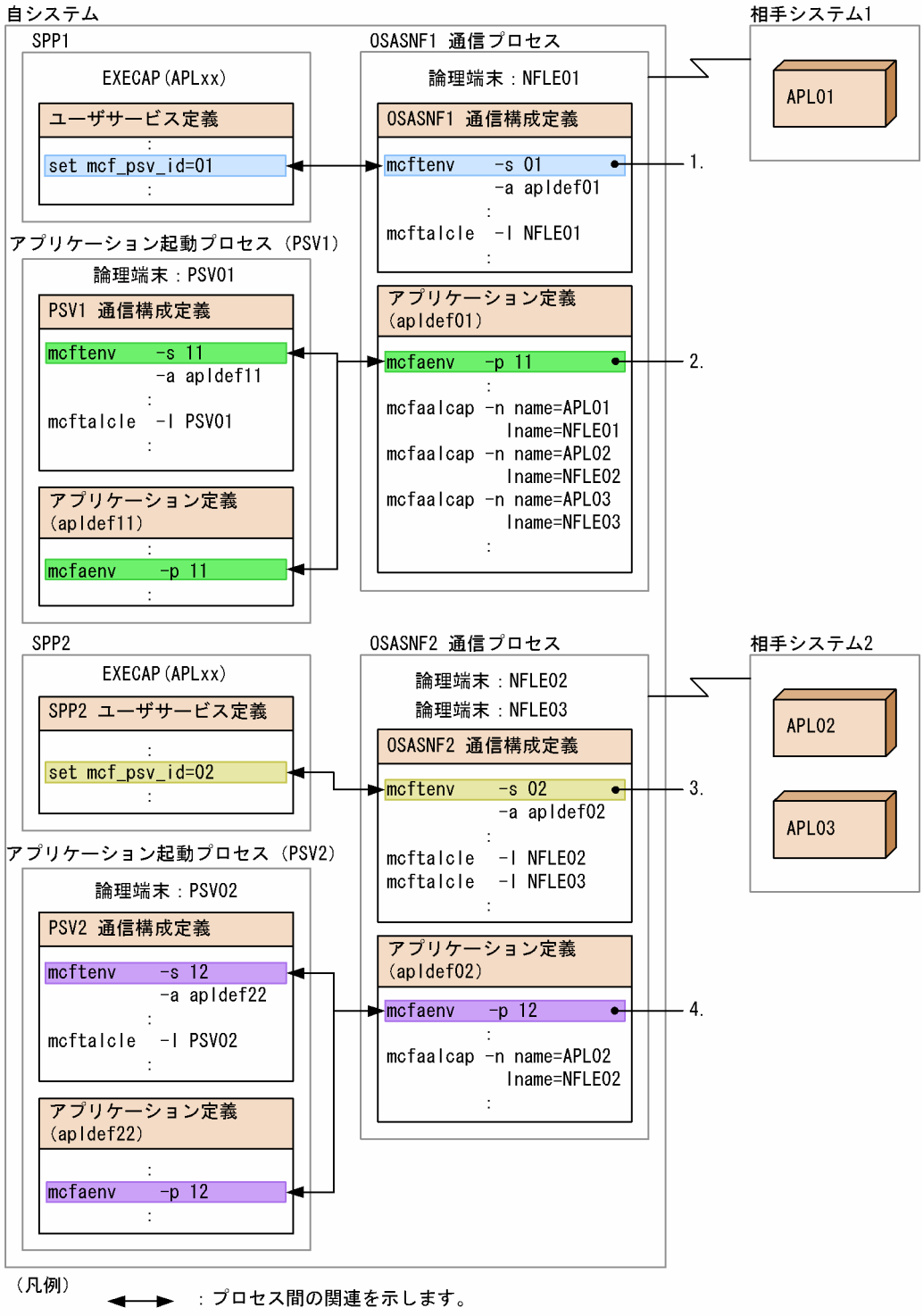
ここでは，次に示す三つのパターンで，それぞれの定義をどのように関連づけるのか説明します。

- SPP からのシステム間通信によってアプリケーションを起動する場合
- MHP からのシステム間通信によってアプリケーションを起動する場合
- システム間通信と自システム内のアプリケーション起動を併用する場合

SPP からのシステム間通信によってアプリケーションを起動する場合

SPP から EXECAP 要求で相手システムのアプリケーションを起動する場合は，次の図に示すように定義を関連づけます。

図 6-7　SPP からのシステム間通信によってアプリケーションを起動する場合の定義例



1. SPP1 のユーザサービス定義のアプリケーション起動プロセス識別子（set mcf_psv_id）には，OSASNF1 通信プロセスの MCF 環境定義（mcftenv -s）の識別子を指定します。
2. OSASNF1 通信プロセスのアプリケーション環境定義のアプリケーション起動プロセス識別子（mcfaenv -p）には，アプリケーション起動プロセス（PSV1）の MCF 環境定義（mcftenv -s）の識別子を指定します。

3. SPP2 のユーザサービス定義のアプリケーション起動プロセス識別子（set mcf_psv_id）には，OSASNF2 通信プロセスの MCF 環境定義（mcftenv -s）の識別子を指定します。
4. OSASNF2 通信プロセスのアプリケーション環境定義のアプリケーション起動プロセス識別子（mcfaenv -p）には，アプリケーション起動プロセス（PSV2）の MCF 環境定義（mcftenv -s）の識別子を指定します。

この定義例の場合，各 SPP からアプリケーション起動を要求すると，次の表に示すような結果になります。

表 6-8　SPP からのアプリケーション起動の結果

SPP	起動するアプリケーション	アプリケーション起動の可否	アプリケーション起動の結果
SPP1	APL01	○	SPP1 のユーザサービス定義に指定したアプリケーション起動プロセス識別子と一致する OSASNF1 通信プロセスの，アプリケーション定義（apldef01）の APL01 の論理端末（NFLE01）に要求を出すため，OSASNF1 の相手システム 1 で起動されます。
	APL02	○	SPP1 のユーザサービス定義に指定したアプリケーション起動プロセス識別子と一致する OSASNF1 通信プロセスの，アプリケーション定義（apldef01）の APL02 の論理端末（NFLE02）に要求を出すため，OSASNF2 の相手システム 2 で起動されます。
	APL03	○	SPP1 のユーザサービス定義に指定したアプリケーション起動プロセス識別子と一致する OSASNF1 通信プロセスの，アプリケーション定義（apldef01）の APL03 の論理端末（NFLE03）に要求を出すため，OSASNF2 の相手システム 2 で起動されます。
SPP2	APL01	×	SPP2 のユーザサービス定義に指定したアプリケーション起動プロセス識別子と一致する OSASNF2 通信プロセスの，アプリケーション定義（apldef02）に APL01 の定義がないため，起動要求がエラーになります。
	APL02	○	SPP2 のユーザサービス定義に指定したアプリケーション起動プロセス識別子と一致する OSASNF2 通信プロセスの，アプリケーション定義（apldef02）の APL02 の論理端末（NFLE02）に要求を出すため，OSASNF2 の相手システム 2 で起動されます。
	APL03	×	SPP2 のユーザサービス定義に指定したアプリケーション起動プロセス識別子と一致する OSASNF2 通信プロセスの，アプリケーション定義（apldef02）に APL03 の定義がないため，起動要求がエラーになります。

（凡例）

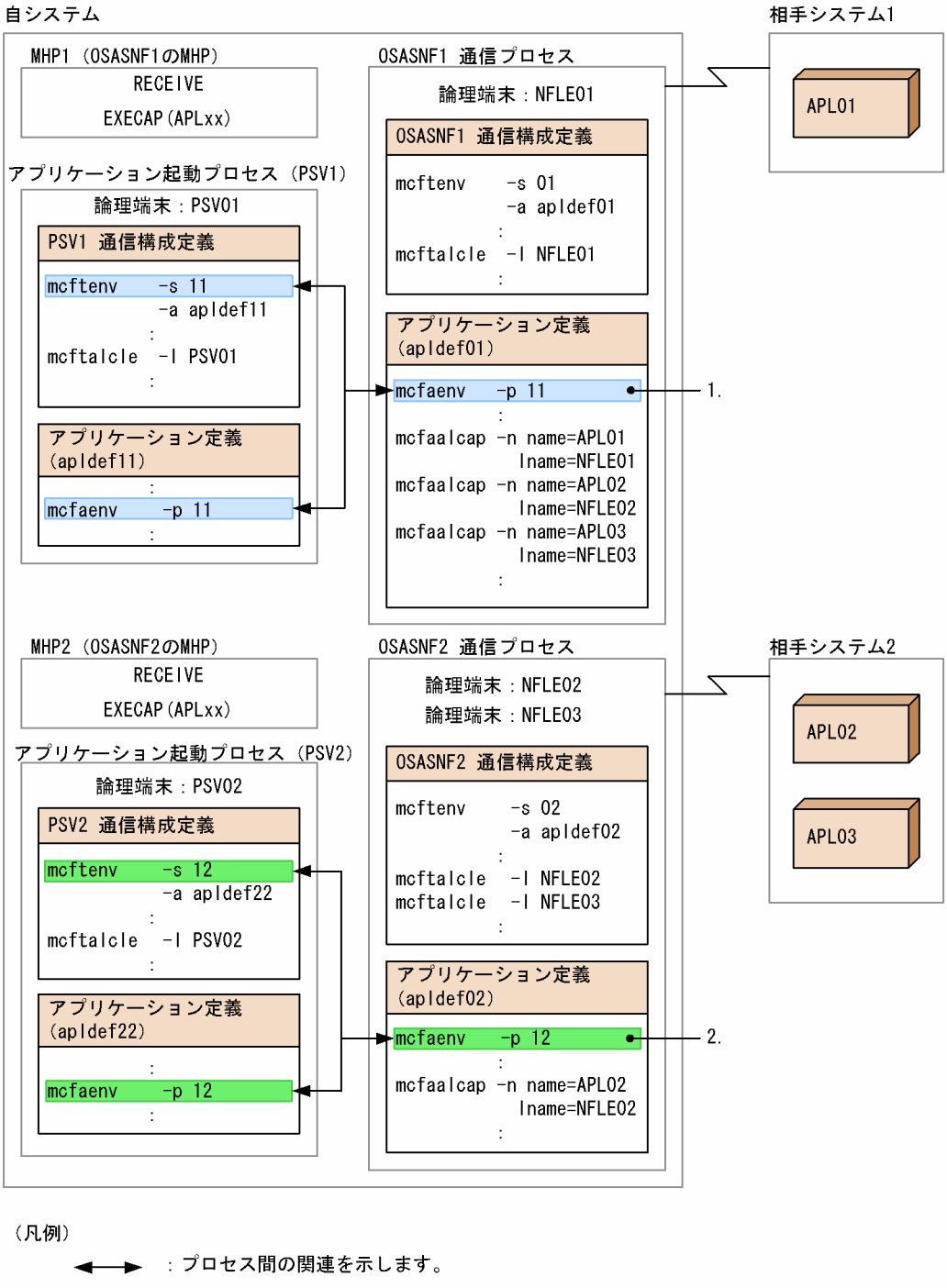
○：起動できます。

×：起動できません。

MHP からのシステム間通信によってアプリケーションを起動する場合

MHP から EXECAP 要求で相手システムのアプリケーションを起動する場合は，次の図に示すように定義を関連づけます。

図 6-8　MHP からのシステム間通信によってアプリケーションを起動する場合の定義例



1. OSASNF1 通信プロセスのアプリケーション環境定義のアプリケーション起動プロセス識別子（mcfaenv -p）には，アプリケーション起動プロセス（PSV1）の MCF 環境定義（mcftenv -s）の識別子を指定します。
2. OSASNF2 通信プロセスのアプリケーション環境定義のアプリケーション起動プロセス識別子（mcfaenv -p）には，アプリケーション起動プロセス（PSV2）の MCF 環境定義（mcftenv -s）の識別子を指定します。

この定義例の場合，各 MHP からアプリケーション起動を要求すると，次の表に示すような結果になります。

表 6-9　MHP からのアプリケーション起動の結果

MHP	起動するアプリケーション	アプリケーション起動の可否	アプリケーション起動の結果
MHP1（OSASNF1 の MHP）	APL01	○	MHP1 を起動した OSASNF1 通信プロセスのアプリケーション定義（apldef01）の APL01 の論理端末（NFLE01）に要求を出すため，OSASNF1 の相手システム 1 で起動されます。
	APL02	○	MHP1 を起動した OSASNF1 通信プロセスのアプリケーション定義（apldef01）の APL02 の論理端末（NFLE02）に要求を出すため，OSASNF2 の相手システム 2 で起動されます。
	APL03	○	MHP1 を起動した OSASNF1 通信プロセスのアプリケーション定義（apldef01）の APL03 の論理端末（NFLE03）に要求を出すため，OSASNF2 の相手システム 2 で起動されます。
MHP2（OSASNF2 の MHP）	APL01	×	MHP2 を起動した OSASNF2 通信プロセスのアプリケーション定義（apldef02）に APL01 の定義がないため，起動要求がエラーになります。
	APL02	○	MHP2 を起動した OSASNF2 通信プロセスのアプリケーション定義（apldef02）の APL02 の論理端末（NFLE02）に要求を出すため，OSASNF2 の相手システム 2 で起動されます。
	APL03	×	MHP2 を起動した OSASNF2 通信プロセスのアプリケーション定義（apldef02）に APL03 の定義がないため，起動要求がエラーになります。

（凡例）

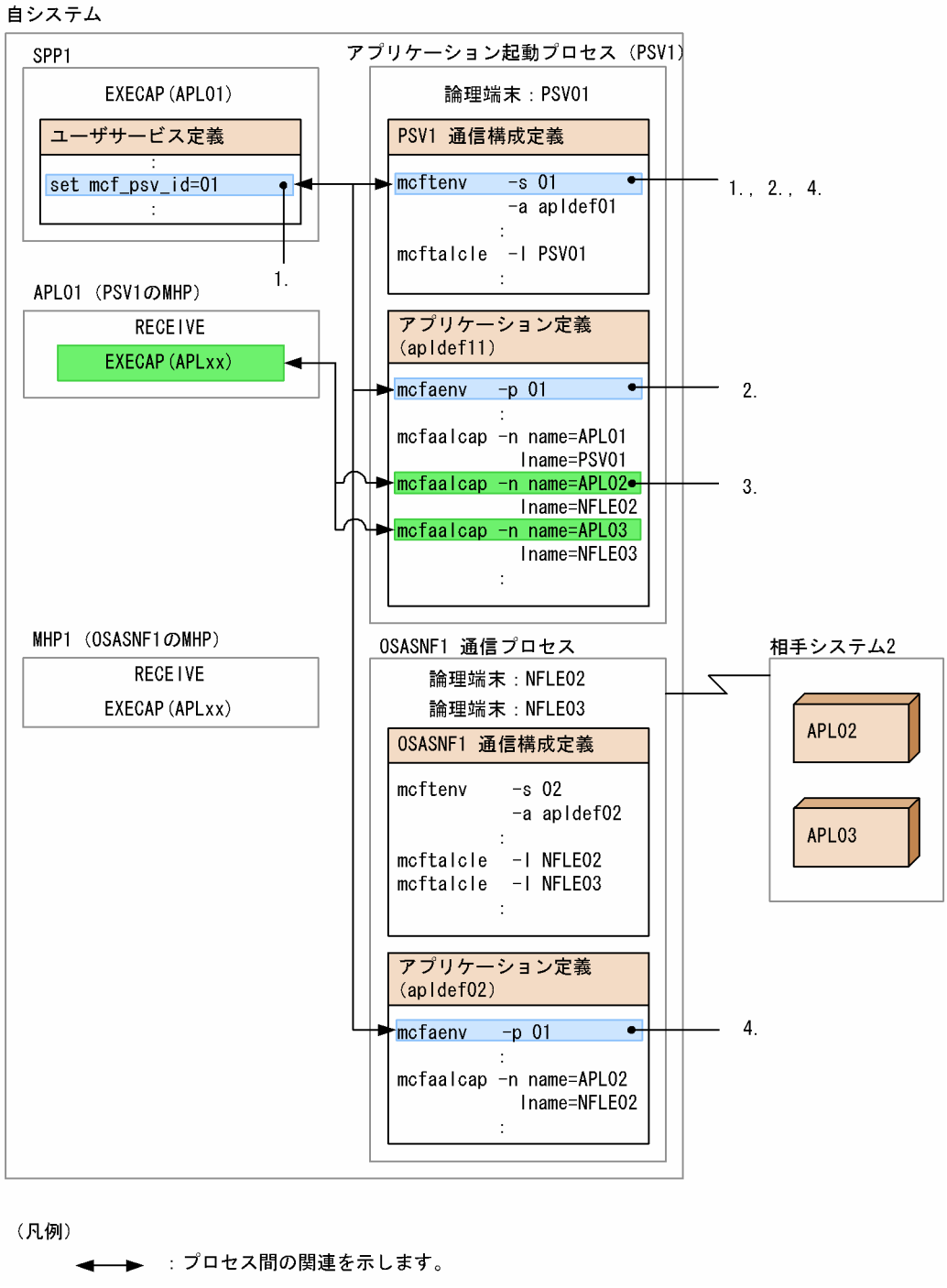
○：起動できます。

×：起動できません。

システム間通信と自システム内のアプリケーション起動を併用する場合

自システム内でのアプリケーション起動によって UAP を起動したあと，その UAP でシステム間通信を行ってアプリケーション起動要求を行う場合は，次の図に示すように定義を関連づけます。

図 6-9　システム間通信と自システム内アプリケーション起動の併用する場合の定義例



1. 自システム内のアプリケーション起動の場合，SPP1 のユーザサービス定義のアプリケーション起動プロセス識別子（set mcf_psv_id）には，アプリケーション起動プロセス（PSV1）の MCF 環境定義（mcftenv -s）の識別子を指定します。
2. アプリケーション起動プロセス（PSV1）のアプリケーション環境定義のアプリケーション起動プロセス識別子（mcfaenv -p）には，アプリケーション起動プロセス（PSV1）の MCF 環境定義（mcftenv -s）の識別子を指定します。

3. 自システム内でのアプリケーション起動によって起動した UAP から，さらにシステム間通信によるアプリケーション起動要求を行う場合，アプリケーション起動プロセス（PSV1）の MCF アプリケーション定義に，起動するアプリケーションを定義します。※
4. OSASNF1 通信プロセスのアプリケーション環境定義のアプリケーション起動プロセス識別子（mcfaenv -p）には，アプリケーション起動プロセス（PSV1）の MCF 環境定義（mcftenv -s）の識別子を指定します。

注※

OSASNF1 通信プロセスの論理端末名も設定してください。

この定義例の場合，各 UAP からアプリケーション起動を要求すると，次の表に示すような結果になります。

表 6–10　UAP からのアプリケーション起動の結果

UAP	起動するアプリケーション	アプリケーション起動の可否	アプリケーション起動の結果
SPP1	APL01	○	SPP1 ユーザサービス定義に指定したアプリケーション起動プロセス識別子と一致するアプリケーション起動プロセス（PSV1）の，アプリケーション定義（apldef01）の APL01 の論理端末（PSV01）に要求を出すため，自システムで起動されます。
	APL02	○	SPP1 ユーザサービス定義に指定したアプリケーション起動プロセス識別子と一致するアプリケーション起動プロセス（PSV1）の，アプリケーション定義（apldef01）の APL02 の論理端末（NFLE02）に要求を出すため，OSASNF1 の相手システム 2 で起動されます。
	APL03	○	SPP1 ユーザサービス定義に指定したアプリケーション起動プロセス識別子と一致するアプリケーション起動プロセス（PSV1）の，アプリケーション定義（apldef01）の APL03 の論理端末（NFLE03）に要求を出すため，OSASNF1 の相手システム 2 で起動されます。
APL01（PSV1 の MHP）	APL01	○	APL01 を起動したアプリケーション起動プロセス（PSV01）の，アプリケーション定義（apldef01）の APL01 の論理端末（PSV01）に要求を出すため，自システムで起動されます。
	APL02	○	APL01 を起動したアプリケーション起動プロセス（PSV01）の，アプリケーション定義（apldef01）の APL02 の論理端末（NFLE02）に要求を出すため，OSASNF1 の相手システム 2 で起動されます。
	APL03	○	APL01 を起動したアプリケーション起動プロセス（PSV01）の，アプリケーション定義（apldef01）の APL03 の論理端末（NFLE03）に要求を出すため，OSASNF1 の相手システム 2 で起動されます。

UAP	起動するアプリケーション	アプリケーション起動の可否	アプリケーション起動の結果
MHP1（OSASNF1 の MHP）	APL01	×	MHP1 を起動した通信プロセス（OSASNF1）のアプリケーション定義（apldef02）に APL01 の定義がないため，起動要求がエラーになります。
	APL02	○	MHP1 を起動した通信プロセス（OSASNF1）のアプリケーション定義（apldef02）の，APL02 の論理端末（NFLE02）に要求を出すため，OSASNF1 の相手システム 2 で起動されます。
	APL03	×	MHP1 を起動した通信プロセス（OSASNF1）のアプリケーション定義（apldef02）に APL03 の定義がないため，起動要求がエラーになります。

（凡例）

○：起動できます。

×：起動できません。

MCF トレースファイルの見積もり式

ここでは，トレース情報量の見積もり式，トレース情報が失われる経過時間の見積もり式，および具体的な見積もりの例について説明します。

トレース情報量の見積もり式

1 秒当たりに取得する MCF トレースファイルの，トレース情報量の見積もり式を次に示します。

```
1秒当たりのトレース情報量（単位：バイト）
                              ＝ A×B＋C×D＋E×F
```

- A：コネクションの確立と解放時に取得するトレース情報量（単位：バイト）
  - 27500 ＋ 2600×コネクション配下の論理端末数
- B：1 秒当たりのコネクション確立および解放の回数※
- C：メッセージ送信時に取得するトレース情報量（単位：バイト）
  - 問い合わせメッセージまたは応答メッセージの送信の場合：7000 ＋（7000×（送信セグメント数－1））
  - 一方送信メッセージの送信の場合：9000 ＋（7000×（送信セグメント数－1））
- D：1 秒当たりのメッセージ送信回数※
- E：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）
  - 問い合わせメッセージまたは応答メッセージの受信の場合：6000 ＋（6500×（送信セグメント数－1））
  - 一方送信メッセージの受信の場合：8000 ＋（6500×（送信セグメント数－1））
- F：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。

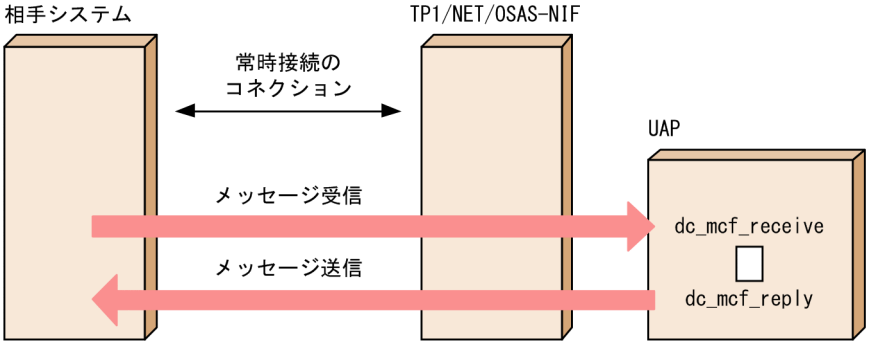
```
経過時間（秒） ＝ F×G×H／1秒当たりのトレース情報量（単位：バイト）
```

- 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，問い合わせメッセージの受信と応答メッセージの送信を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
論理端末数	1 つ
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
送信セグメント数	1 つ
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

```
((27500＋2600×1)×0※) ＋ ((7000＋7000×0)×(120／60)) ＋ ((6000＋6500×0)×(120／60))
＝ 26000
```

1 秒当たりのトレース情報量は，26000 バイトとなります。

注※

コネクションが常時接続であるため，1 秒当たりのコネクション確立および解放の回数は 0 として計算してください。

トレース情報が失われる経過時間の見積もり

```
204800×100×3／26000 ＝ 2363.1
```

トレース情報が失われる経過時間は，2363.1 秒（約 39 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。

IP アドレスの変更

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。

IP アドレスを変更する場合に見直しが必要な定義

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 6–11　IP アドレスを変更する場合に見直しが必要な定義の一覧

定義ファイル名	定義	見直しが必要な条件
MCF 通信構成定義	mcftalccn -q※1	OSI 拡張機能を使用する場合（OSI 拡張高信頼化機能を除く）
XNF の構成定義文	TPTCP_slot の IP_address※2	自局 IP アドレス指定機能を使用する場合

注※1

OSI 拡張機能を使用する場合，NSAP アドレスに IP アドレスが含まれています。NSAP アドレスの形式については，マニュアル「通信管理 XNF/AS NSAP アドレス概説編」を参照してください。

注※2

詳細については，マニュアル「通信管理 XNF/AS 構成定義編」を参照してください。

IP アドレスの変更手順

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. XNF/AS を停止します。
3. MCF 通信構成定義および XNF の構成定義文について，変更前の IP アドレスを grep コマンドで検索します。
4. 検索の結果，変更前の IP アドレスが見つかった場合には，変更します。
5. MCF 通信構成定義を変更した場合，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。XNF の構成定義文を変更した場合，ゼネレーションを行います。

コネクション（論理端末）の追加

コネクション（論理端末）を追加する場合に見直す必要のある定義の一覧，および再見積もりが発生する条件を次の表に示します。

表 6-12　コネクション（論理端末）を追加する場合に見直しが必要な定義の一覧

定義ファイル名	定義	再見積もりが発生する条件
システム環境定義	static_shmpool_size※1	無条件に再見積もりが必要
	dynamic_shmpool_size※1	同時に送受信するメッセージ数が増加する場合
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 -l	無条件に再見積もりが必要
	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，「mcftalccn（コネクション定義の開始）」の注意事項とマニュアル「OpenTP1 システム定義」の「mcftbuf」の説明を参照してください。

注※5

詳細については，「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 6-13　コネクション（論理端末）を追加する場合に見直しが必要な OpenTP1 ファイルの一覧

OpenTP1 ファイル	再見積もりが発生する条件
ステータスファイル	次に示すどれかの条件の場合，再見積もりが必要 • メッセージ出力通番を使用する場合 • 拡張予約定義を定義している場合 • 状態を引き継ぐ論理端末が増える場合

OpenTP1 ファイル	再見積もりが発生する条件
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/OSAS-NIF 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/OSAS-NIF を使用したシステム定義の例を示します。TP1/NET/OSAS-NIF のシステム構成例を次の図に示します。

図 6-10　TP1/NET/OSAS-NIF のシステム構成例

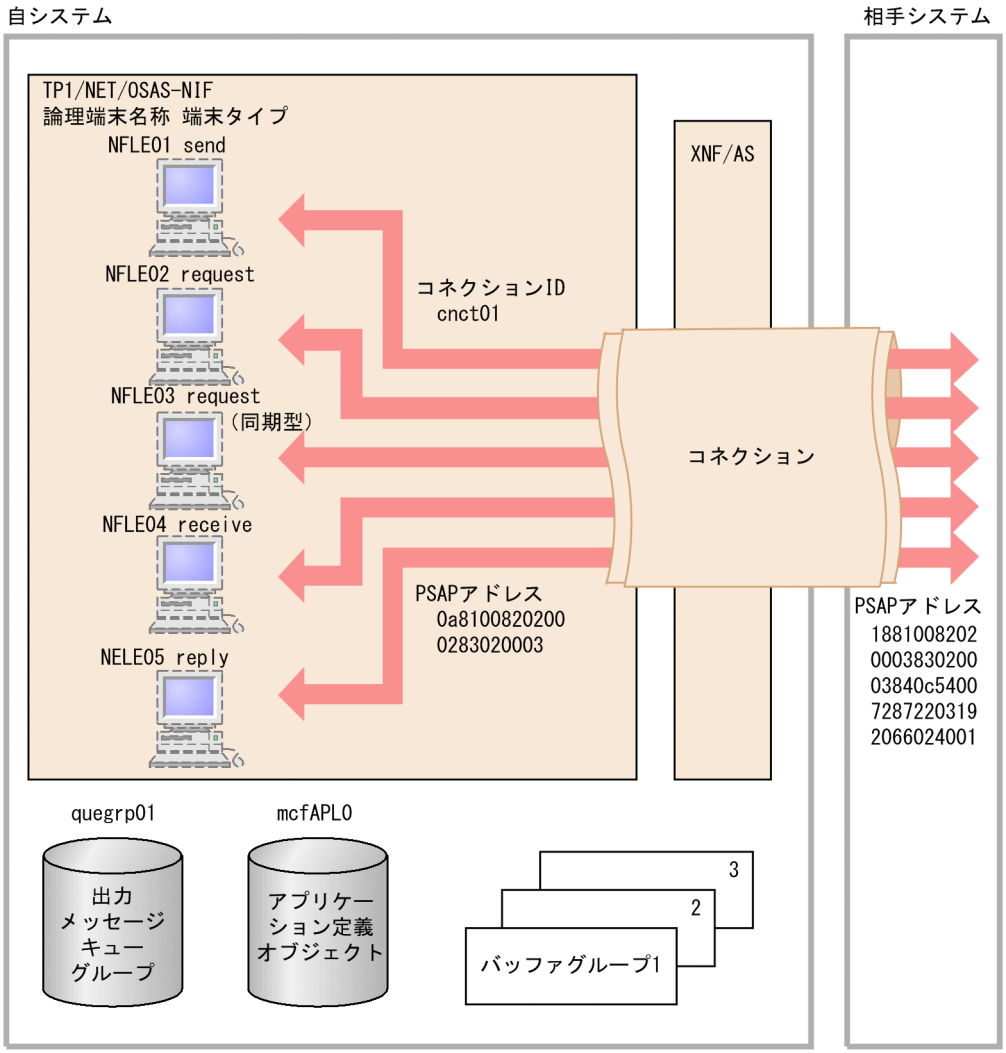


図 6-10 のシステム構成例のコーディング例を次に示します。このコーディング例は，/BeTRAN/examples/mcf/OSASNIF/conf/com_d のファイルで提供しています。

```
### MCF通信構成定義 TP1/NET/OSAS-NIFプロトコル固有定義                ###
#
### コネクション定義の開始(cnct01)
mcftalccn   -c  cnct01                                          ¥
            -p  onf                                             ¥
            -n  x'0a81008202000283020003'                       ¥
            -g "sndbuf=1                                        ¥
                rcvbuf=2"                                       ¥
            -e "msgbuf=3                                        ¥
                count=14"                                       ¥
            -m "mode=xnfas"                                     ¥
```

```
            -v "tim5=60"                                                    ¥
            -o "ownsid=x'0001'                                              ¥
                otrsid=x'ffff'"                                             ¥
            -b "bretrycnt=20                                                ¥
                bretryint=5"                                                ¥
            -z "slot=1"                                                     ¥
            -q x'1881008202000383020003840c5400728722031920660240001'
#
###  論理端末定義
     mcftalcle   -l  NFLE01                                                 ¥
                 -t  send                                                   ¥
                 -k "quegrpid=quegrp01"
#
###  論理端末定義
     mcftalcle   -l  NFLE02                                                 ¥
                 -t  request                                                ¥
                 -k "quegrpid=quegrp01"
#
###  論理端末定義
     mcftalcle   -l  NFLE03                                                 ¥
                 -t  request                                                ¥
                 -k "quegrpid=quegrp01"                                     ¥
                 -d "sync=yes"
#
###  論理端末定義
     mcftalcle   -l  NFLE04                                                 ¥
                 -t  receive                                                ¥
                 -k "quegrpid=quegrp01"
#
###  論理端末定義
     mcftalcle   -l  NFLE05                                                 ¥
                 -t  reply                                                  ¥
                 -k "quegrpid=quegrp01"                                     ¥
                 -d "rplytim=30"
#
#-------------------------------------------------------------------------#
### コネクション定義の終了(cnct01)
mcftalced
#
```

7 運用コマンド

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

TP1/NET/OSAS-NIF の運用コマンド

ここでは，TP1/NET/OSAS-NIF 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/OSAS-NIF で使用する運用コマンドを次の表に示します。

表 7-1　TP1/NET/OSAS-NIF で使用する運用コマンド

機能		コマンド名称	定義中に組み込む	オフライン中に実行	オンライン中に実行	UAP から実行
コネクション管理	コネクションの確立	mcftactcn	×	×	○	○
	コネクションの解放	mcftdctcn	×	×	○	○
	コネクションの状態表示	mcftlscn	×	×	○	○
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。

また，MCF が標準的に提供しているコマンドのうち，TP1/NET/OSAS-NIF では次に示すコマンドは使用できないので，ご注意ください。

- mcftspqle
- mcftendct
- mcfadltap

mcftactcn（コネクションの確立）

形式

```
mcftactcn 〔-s MCF通信プロセス識別子〕 -c コネクションID
```

機能

コネクションを確立します。

オプション

● -s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象のコネクションを制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftactcn コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

● -c コネクション ID　～〈1～8 文字の識別子〉

確立するコネクションのコネクション ID を指定します。

コネクション ID は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数指定するときは，引用符（"）で囲んで，コネクション ID とコネクション ID との間を空白で区切ります。同一コネクション ID は重複して指定できません。

また，コネクション ID は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外のコネクション ID を混在して指定できません。一括指定の場合も，引用符（"）で囲みます。

*：すべてのコネクションを確立します。

先行文字列*：先行文字列で始まるすべてのコネクションを確立します。

〈複数指定の例〉　cnn1，cnn2，cnn3 を指定する場合

```
-c "cnn1△cnn2△cnn3"
```

〈一括指定の例〉　cnn で始まるすべてのコネクションを指定する場合

```
-c "cnn*"
```

出力メッセージ

出力メッセージ ID	内容	出力先
KFCA10350-I	mcftactcn コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftactcn コマンド入力元への応答を失敗しました。	メッセージログファイル
KFCA10371-I	mcftactcn コマンドを正常に受け付けました。	標準出力
KFCA10373-E	mcftactcn コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10391-E	mcftactcn コマンドはサポートされていません。	標準エラー出力
KFCA10500-I	ヘルプメッセージ	標準出力
KFCA13530-E	コマンドを無視しました。	標準エラー出力
KFCA13531-E	コマンドが失敗しました。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftactcn コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftactle（論理端末の閉塞解除）

形式

```
mcftactle 〔-s MCF通信プロセス識別子〕 〔-c コネクションID〕
          -l 論理端末名称
```

機能

論理端末の閉塞を解除します。

オプション

● -s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象の論理端末を制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftactle コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

● -c コネクション ID　～〈1～8 文字の識別子〉

閉塞解除したい論理端末に対応するコネクションのコネクション ID を指定します。

コネクション ID は複数指定できません。また，一括指定もできません。

● -l 論理端末名称　～〈1～8 文字の識別子〉

閉塞解除する論理端末の名称を指定します。

-c オプションを指定した場合は，指定したコネクション ID に対応する論理端末名称を指定します。

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。一括指定の場合も，引用符（"）で囲みます。

*：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名称を指定する場合

```
-l ”len*”
```

出力メッセージ

出力メッセージ 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
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10382-E	指定した論理端末は登録されていません。	標準エラー出力
KFCA10390-E	指定されたコネクション ID と論理端末名称の対応が正しくありません。	標準エラー出力
KFCA10391-E	mcftactle コマンドはサポートされていません。	標準エラー出力
KFCA10395-E	指定したコネクションには未接続の論理端末名称が指定されています。	標準エラー出力
KFCA10503-I	ヘルプメッセージ	標準出力
KFCA13530-E	コマンドを無視しました。	標準エラー出力
KFCA13531-E	コマンドが失敗しました。	標準エラー出力

出力メッセージ ID	内容	出力先
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftactle コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftdctcn（コネクションの解放）

形式

```
mcftdctcn 〔-s MCF通信プロセス識別子〕 -c コネクションID 〔-f〕
```

機能

コネクションを解放します。

-f オプションを指定した場合は，該当コネクションが仕掛り中の場合でも，強制的に解放します。コネクション解放後，CERREVT を通知します。

-f オプションを指定しない場合は，該当コネクションの仕掛り完了を待ってから，解放します。コネクション解放後，CCLSEVT を通知します。

なお，仕掛り中の処理は，次のように扱います。

- 受信仕掛り中の場合，受信途中のメッセージを破棄します。
- 非同期 SEND（dc_mcf_send）の仕掛り中の場合，送信処理を中断し，仕掛り中のメッセージを出力キューに戻します。出力キューに戻したメッセージは，次回のコネクション確立時に再送されます。
- 同期送受信（dc_mcf_sendrecv）仕掛り中の場合，送受信処理を中断し，エラーコード DCMCFRTN_73020 でリターンします。

オプション

● -s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象のコネクションを制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftdctcn コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

● -c コネクション ID　～〈1～8 文字の識別子〉

解放するコネクションのコネクション ID を指定します。

コネクション ID は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数指定するときは，引用符（"）で囲んで，コネクション ID とコネクション ID との間を空白で区切ります。同一コネクション ID は，重複して指定できません。

また，コネクション ID は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外のコネクション ID を混在して指定できません。一括指定の場合も，引用符（"）で囲みます。

*：すべてのコネクションを解放します。

先行文字列*：先行文字列で始まるすべてのコネクションを解放します。

〈複数指定の例〉 cnn1，cnn2，cnn3 を指定する場合

```
-c "cnn1△cnn2△cnn3"
```

〈一括指定の例〉 cnn で始まるすべてのコネクションを指定する場合

```
-c "cnn*"
```

● -f

該当するコネクションを強制的に解放します。

出力メッセージ

出力メッセージ ID	内容	出力先
KFCA10350-I	mcftdctcn コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftdctcn 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
KFCA10371-I	mcftdctcn コマンドを正常に受け付けました。	標準出力
KFCA10373-E	mcftdctcn コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10391-E	mcftdctcn コマンドはサポートされていません。	標準エラー出力

出力メッセージ ID	内容	出力先
KFCA10501-I	ヘルプメッセージ	標準出力
KFCA13530-E	コマンドを無視しました。	標準エラー出力
KFCA13531-E	コマンドが失敗しました。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftdctcn コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftdctle（論理端末の閉塞）

形式

```
mcftdctle 〔-s MCF通信プロセス識別子〕 〔-c コネクションID〕
          -l 論理端末名称
```

機能

論理端末を強制的に閉塞します。

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。

● -c コネクション ID　～〈1～8 文字の識別子〉

閉塞したい論理端末に対応するコネクションのコネクション ID を指定します。

コネクション ID は複数指定できません。また，一括指定もできません。

● -l 論理端末名称　～〈1～8 文字の識別子〉

閉塞する論理端末の名称を指定します。

-c オプションを指定した場合は，指定したコネクション ID に対応する論理端末名称を指定します。

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。一括指定の場合も，引用符（"）で囲みます。

*：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名称を指定する場合

```
-l ”len*”
```

出力メッセージ

出力メッセージ 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
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10382-E	指定した論理端末は登録されていません。	標準エラー出力
KFCA10390-E	指定されたコネクション ID と論理端末名称の対応が正しくありません。	標準エラー出力
KFCA10391-E	mcftdctle コマンドはサポートされていません。	標準エラー出力
KFCA10395-E	指定したコネクションには未接続の論理端末名称が指定されています。	標準エラー出力
KFCA10504-I	ヘルプメッセージ	標準出力
KFCA13530-E	コマンドを無視しました。	標準エラー出力
KFCA13531-E	コマンドが失敗しました。	標準エラー出力

出力メッセージ ID	内容	出力先
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- receive 型論理端末および reply 型論理端末で，メッセージの受信仕掛り中にこのコマンドを実行した場合

  受信仕掛り中のメッセージを破棄します。以降の受信メッセージは入力キューに登録しません。
- request 型論理端末で，メッセージの受信仕掛り中にこのコマンドを実行した場合

  メッセージ受信完了を待ってから論理端末を閉塞します。
- メッセージの送信仕掛り中にこのコマンドを実行した場合

  メッセージの送信処理が中断されます。出力キューにディスクキューを使用しているときは，UAP からの送信メッセージは出力キュー上に格納されます。出力キューにメモリキューを使用しているときは，UAP からの送信メッセージは破棄されます。
- mcftdctle コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftlscn（コネクションの状態表示）

形式

```
mcftlscn 〔-s MCF通信プロセス識別子〕 -c コネクションID 〔-d〕
```

機能

コネクションの状態を標準出力に出力します。

オプション

● -s MCF 通信プロセス識別子　～〈数字（0～9），a～f〉((01～ef))

処理対象のコネクションを制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数指定できません。

このオプションの指定を省略すると，すべての MCF 通信サービスに対して，mcftlscn コマンドを実行します。したがって，MCF 通信サービスを検索するオーバヘッドが，運用コマンドの処理に加わります。

MCF 通信サービスが多い構成や運用コマンドを多数入力する運用を行う場合は，-s オプションで，MCF 通信プロセス識別子を指定する運用設計を行ってください。

● -c コネクション ID　～〈1～8 文字の識別子〉

状態を表示するコネクションのコネクション ID を指定します。

コネクション ID は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数指定するときは，引用符（"）で囲んで，コネクション ID とコネクション ID との間を空白で区切ります。同一コネクション ID は，重複して指定できません。

また，コネクション ID は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外のコネクション ID を混在して指定できません。一括指定の場合も，引用符（"）で囲みます。

*：すべてのコネクションの状態を表示します。

先行文字列*：先行文字列で始まるすべてのコネクションの状態を表示します。

〈複数指定の例〉　cnn1，cnn2，cnn3 を指定する場合

```
-c "cnn1△cnn2△cnn3"
```

〈一括指定の例〉　cnn で始まるすべてのコネクションを指定する場合

```
-c "cnn*"
```

● -d

コネクションの状態と該当するコネクションに対応する論理端末の情報を表示します。

このオプションの指定を省略すると，コネクションの状態だけを表示します。

出力形式

```
mmm cccccccc ppp sssss dddd
    llllllll ttt nn
```

注

-d オプションを指定しないで mcftlscn コマンドを実行した場合は，「mmm cccccccc ppp sssss dddd」の行だけ出力されます。

- mmm：MCF 識別子
- cccccccc：コネクション ID
- ppp：プロトコル種別
  NIF…NIF プロトコル
- sssss：コネクション状態
  ACT…確立
  ACT/B…確立処理中
  DCT…解放
  DCT/B…解放処理中
- dddd：詳細ステータス（保守情報）
- llllllll：論理端末名称
- ttt：論理端末の端末タイプ
  REQ…request 型
  RLY…reply 型
  SND…send 型
  REV…receive 型
- nn：エージェント番号

出力メッセージ

出力メッセージ ID	内容	出力先
KFCA10350-I	mcftlscn コマンドが入力されました。	標準出力
KFCA10351-E	MCF 開始処理中です。	標準エラー出力
KFCA10352-E	MCF 終了処理中です。	標準エラー出力

出力メッセージ ID	内容	出力先
KFCA10353-W	入力形式が誤っています。	標準エラー出力
KFCA10354-E	メモリ不足です。	メッセージログファイル，または標準エラー出力
KFCA10355-W	引数の指定が誤っています。	標準エラー出力
KFCA10356-E	プロセス間でタイムアウトが発生しました。	標準エラー出力
KFCA10357-E	MCF 内でタイムアウトが発生しました。	標準エラー出力
KFCA10358-E	内部関数のエラーが発生しました。	メッセージログファイル，または標準エラー出力
KFCA10359-W	mcftlscn コマンド入力元への応答を失敗しました。	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
KFCA10361-I	標準情報を表示します。	標準出力
KFCA10362-I	詳細情報を表示します。	標準出力
KFCA10369-I	状態表示を終了します。	標準出力
KFCA10373-E	mcftlscn コマンドが異常終了しました。	標準エラー出力
KFCA10380-E	相手プロセスの検索に失敗しました。	標準エラー出力
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10391-E	mcftlscn コマンドはサポートされていません。	標準エラー出力
KFCA10502-I	ヘルプメッセージ	標準出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

mcftlsle（論理端末の状態表示）

形式

```
mcftlsle 〔-s MCF通信プロセス識別子〕 〔-c コネクションID〕
         -l 論理端末名称 〔-q〕
```

機能

論理端末の状態を標準出力に出力します。

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。

● -c コネクション ID　～〈1～8 文字の識別子〉

状態を表示したい論理端末に対応するコネクションのコネクション ID を指定します。

コネクション ID は複数指定できません。また，一括指定もできません。

● -l 論理端末名称　～〈1～8 文字の識別子〉

状態を表示する論理端末の名称を指定します。

-c オプションを指定した場合，指定したコネクション ID に対応する論理端末名称を指定します。

論理端末名称は，一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数指定するときは，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。一括指定の場合も，引用符（"）で囲みます。

*：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名称を指定する場合

```
-l ”len*”
```

● -q

指定した論理端末に対応する出力キューの保留状態を表示します。

このオプションの指定を省略すると，論理端末に対応する出力キューの保留状態は表示しません。

出力形式

```
mmm llllllll sss〔tttt〕
    SYNC xxxxxxxxxx yyyyyyyyyy zzzzzzzzzz
      IO     :           :         :
    PRIO     :           :         :
    NORM     :           :         :
    iii ooo
```

- mmm：MCF 識別子
- llllllll：論理端末名称
- sss：論理端末状態
  ACT…閉塞解除状態
  DCT…閉塞状態
- 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 オプション指定時だけ表示）
  NOH…保留解除
  HLD…保留

- ooo：出力キューの出力の保留状態（-q オプション指定時だけ表示）
  NOH…保留解除
  HLD…保留

出力メッセージ

出力メッセージ 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
KFCA10381-E	指定したコネクションは登録されていません。	標準エラー出力
KFCA10382-E	指定した論理端末は登録されていません。	標準エラー出力
KFCA10390-E	指定されたコネクション ID と論理端末名称の対応が正しくありません。	標準エラー出力
KFCA10391-E	mcftlsle コマンドはサポートされていません。	標準エラー出力
KFCA10395-E	指定したコネクションには未接続の論理端末名称が指定されています。	標準エラー出力
KFCA10505-I	ヘルプメッセージ	標準出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

8 組み込み方法

TP1/NET/OSAS-NIF を OpenTP1 システムへ組み込む方法について説明します。

8.1　TP1/NET/OSAS-NIF の組み込みの流れ

TP1/NET/OSAS-NIF 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/OSAS-NIF 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/OSAS-NIF 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。

また，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，「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/OSAS-NIF は，OpenTP1 プロセスサービスによって起動されます。

TP1/NET/OSAS-NIF を起動するには，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/OSAS-NIF 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，mcfplosasnf 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 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（K&R 版 C，ANSI C または 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 および図 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。

- 図 8-1（ANSI C，C++）：/BeTRAN/examples/mcf/OSASNIF/cmlib/ansi/com.c
- 図 8-2（K&R 版 C）：/BeTRAN/examples/mcf/OSASNIF/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）

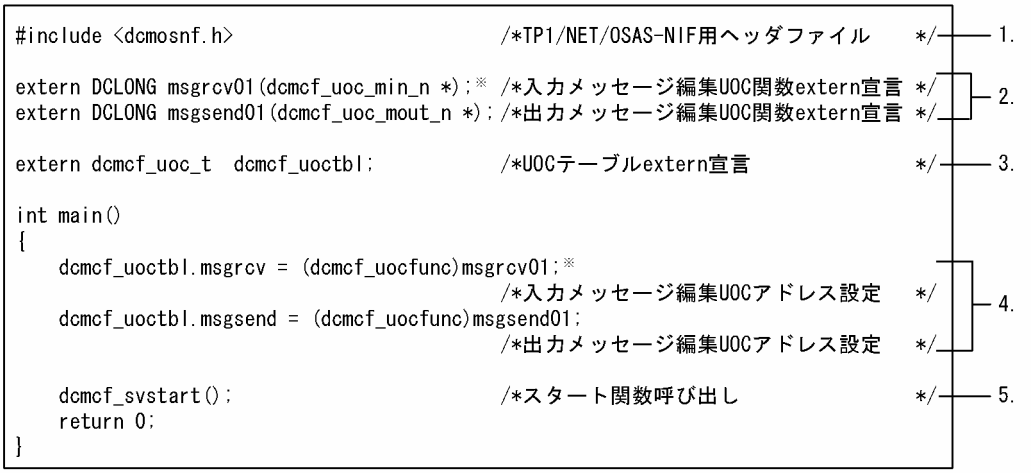
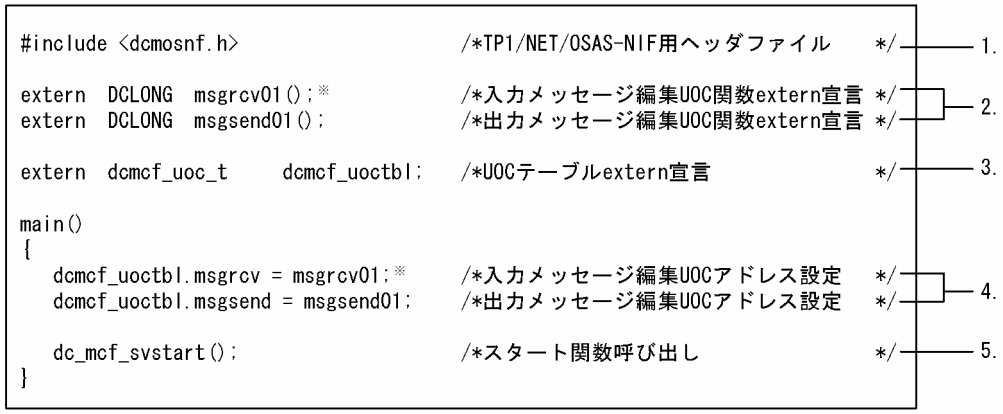


図 8-2　MCF メイン関数のコーディング概要（K&R 版 C の場合）



注※

TP1/NET/OSAS-NIF 提供の標準 UOC を使用する場合は，「msgrcv01」の代わりに「dc_mcf_stduoc_msgin」をコーディングしてください。さらに，下記のように extern 宣言を行ってください。

ANSI C の場合

```
extern DCLONG dc_mcf_stduoc_msgin(dcmcf_uoc_min_n *);
```

C++の場合

```
extern "C"{ extern DCLONG dc_mcf_stduoc_msgin(dcmcf_uoc_min_n *); }
```

K&R 版 C の場合

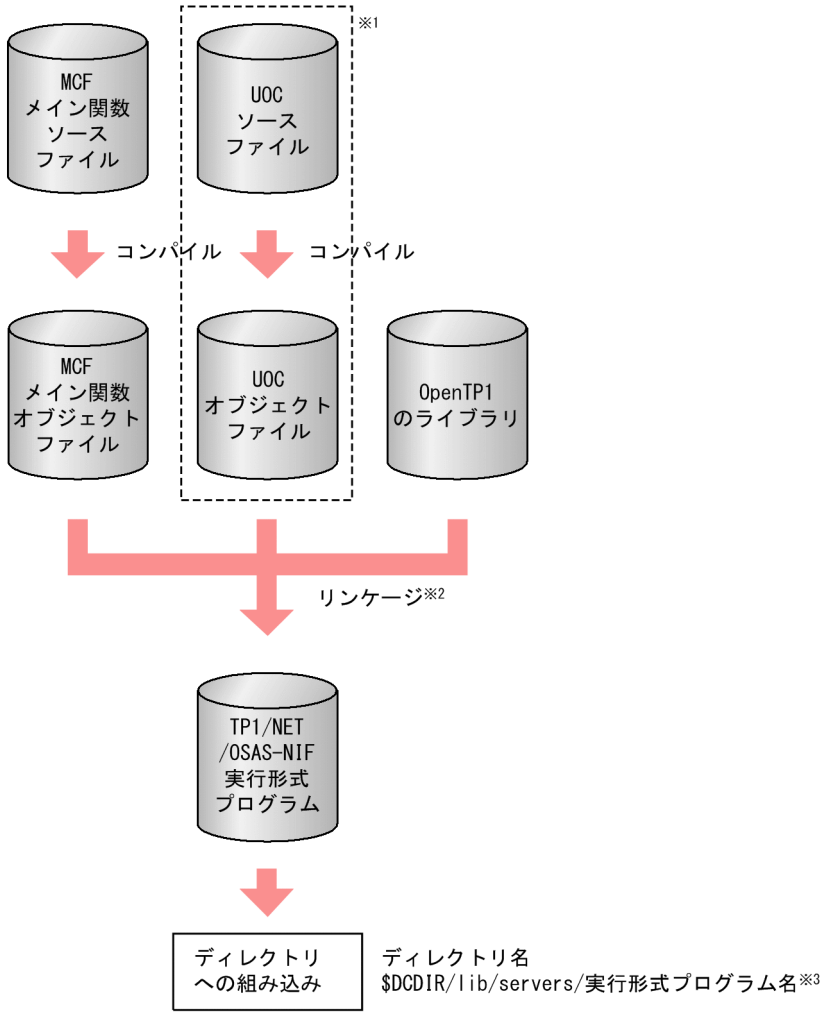
```
extern DCLONG dc_mcf_stduoc_msgin();
```

1. TP1/NET/OSAS-NIF 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。
   これらの UOC を使用する場合だけ，コーディングしてください。
3. UOC テーブルを extern 宣言します。UOC 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。
   2.の UOC を使用する場合だけ，コーディングしてください。
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。

```
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 */
```

   これらの UOC を使用する場合だけ，コーディングしてください。
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



注※1

UOC を使用しない場合は，必要ありません。

注※2

mcfplosasnf コマンドでリンケージします。

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

注※3

TP1/NET/OSAS-NIF 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

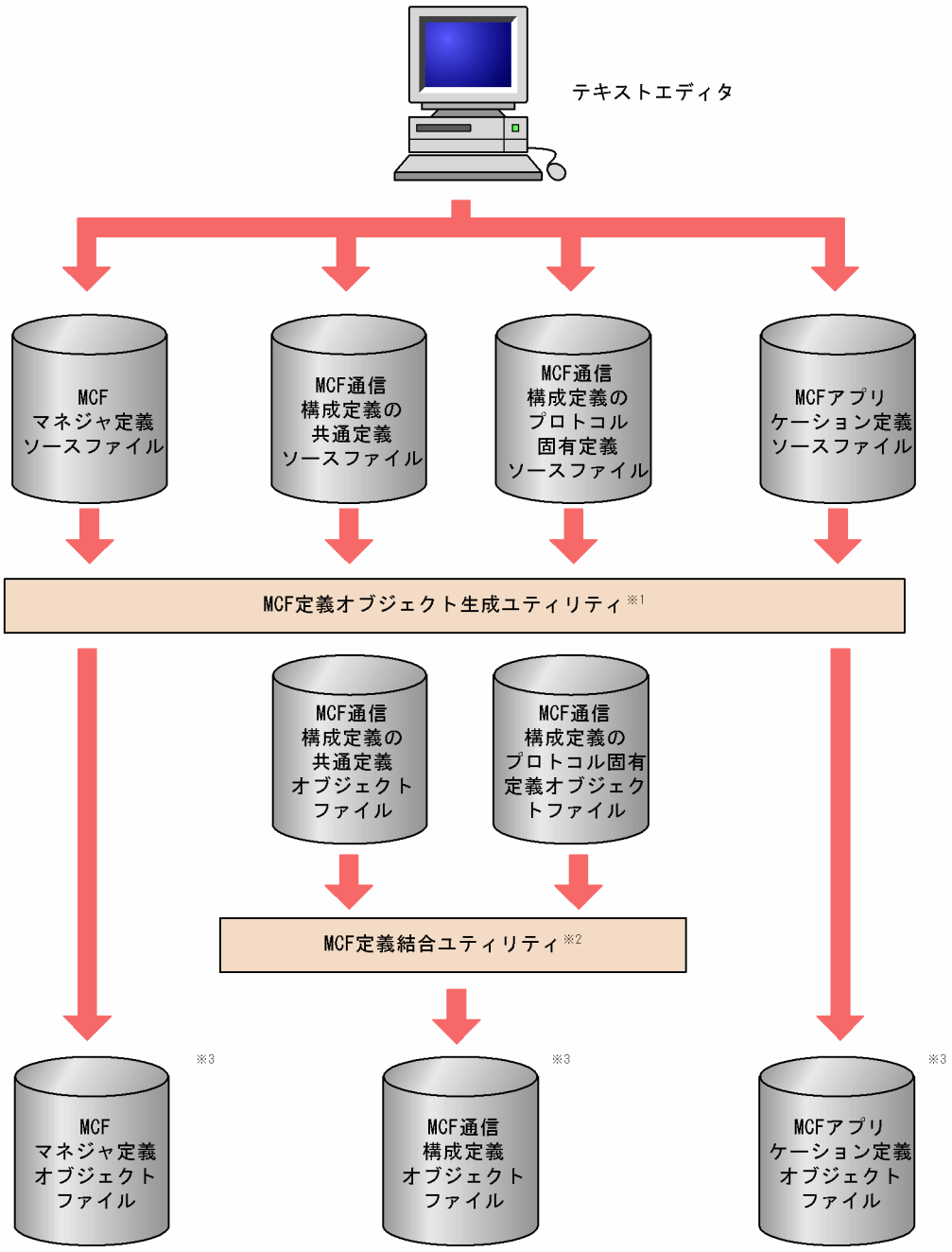
定義オブジェクトファイルを次の手順で生成します。

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. OS のテキストエディタを使用して，MCF の定義ファイルから，次に示す定義ソースファイルを作成します。
   - MCF マネジャ定義ソースファイル
   - MCF 通信構成定義の共通定義ソースファイル
   - MCF 通信構成定義の TP1/NET/OSAS-NIF 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/OSAS-NIF 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ます。

定義オブジェクトファイルの作成方法の概要を次の図に示します。

図 8-4　定義オブジェクトファイルの作成方法の概要



注※1

次に示すコマンドで生成します。

```
mcfXXXX△-i△〔パス名〕入力ファイル名
        △-o△〔パス名〕出力オブジェクトファイル名
```

mcfXXXX は，ソースファイルごとに異なります。

- mcfmngr：MCF マネジャ定義ソースファイル
- mcfcomn：MCF 通信構成定義のソースファイル
- mcfosnf：MCF 通信構成定義のプロトコル（TP1/NET/OSAS-NIF）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 mcfosnf コマンドについては「MCF 定義オブジェクト生成ユティリティの起動」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/OSAS-NIF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

TP1/NET/OSAS-NIF の障害対策について説明します。

9.1　障害の種類と対応処理

TP1/NET/OSAS-NIF で発生する障害の種類と対応処理を次の表に示します。

表 9-1 で記号で示すユーザの対応処理の詳細を表 9-2 に示します。該当する A～N の記号で検索してください。

表 9-1　障害の種類と対応処理

障害の種類	障害内容	TP1/NET/OSAS-NIF の処理	ユーザの対応処理
コネクション障害	コネクションの強制解放（mcftdctcn -f コマンド入力）	1. コネクション障害を通知するメッセージログ（KFCA13502-E）を出力します。 2. 該当するコネクション下の論理端末を閉塞します。 3. CERREVT を起動します。 4. コネクションを解放します。	A
	下位層障害	同上	B
	コネクション確立処理の失敗	1. コネクション障害を通知するメッセージログ（KFCA13502-E）を出力します。 2. CERREVT を起動します。	B
論理端末障害	論理端末層検出異常	1. メッセージ受信中の障害の場合，論理端末の端末タイプごとに次に示す処理をします。 ●reply 型，receive 型 (a)受信メッセージを破棄します。 (b)受信失敗を通知するメッセージログ（KFCA13506-E）を出力します。 (c)相手システムへ受信拒否を送信します。 ●request 型，request 型（同期型） (a)受信メッセージを破棄します。 (b)受信失敗を通知するメッセージログ（KFCA13506-E）を出力します。 2. メッセージ送信中の障害の場合は次に示す処理をします。 (a)送信メッセージを破棄します。 (b)送信中断を通知するメッセージログ（KFCA13508-E）を出力します。 (c)相手システムへ送信中断を送信します。 3. 障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 4. CERREVT を起動します。 5. 論理端末を閉塞します。	C
	論理端末閉塞解除失敗	1. 障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 2. CERREVT を起動します。	C

障害の種類	障害内容	TP1/NET/OSAS-NIF の処理	ユーザの対応処理
論理端末障害	mcftdctle コマンド入力	1. 論理端末の端末タイプごとに次に示す処理をします。 ●request 型，request 型（同期型），send 型 (a)メッセージ送信中の場合 ・メッセージ送信を中断します。 ・送信中断を通知するメッセージログ（KFCA13508-E）を出力します。 ・相手システムへ送信中断を送信します。 ●reply 型，receive 型 (a)メッセージ受信中の場合 ・受信メッセージを破棄します。 ・受信失敗を通知するメッセージログ（KFCA13506-E）を出力します。 ・相手システムへ受信拒否を送信します。 2. 障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 3. CERREVT を起動します。 4. 論理端末を閉塞します。	D
入力メッセージ編集 UOC の障害	UOC エラー	1. UOC エラー（KFCA10611-E），UOC パラメタ不正（KFCA10620-E）またはアプリケーション名取得失敗（KFCA10610-E）を通知するメッセージログを出力します。 2. 以降の処理は「論理端末障害」の「論理端末層検出異常」と同じです。	F
入力キュー，スケジューラ，ネームサーバまたは RPC の障害	・UAP 閉塞 ・アプリケーション名不正 ・入力キュー書き込み障害	1. 入力キュー障害を通知するメッセージログ（KFCA10604-E）を出力します。 2. 入力メッセージの種類に従い，次に示す処理をします。	−
		●受信メッセージ（エラーイベント定義） ERREVT1 または ERREVT2 を起動します。	E
		●受信メッセージ（エラーイベント未定義） 以降の処理は「論理端末障害」の「論理端末層検出異常」と同じです。	F
		●ERREVT1，ERREVT2 以降の処理は「論理端末障害」の「論理端末層検出異常」と同じです。	F
		●ERREVT3，ERREVTA，COPNEVT，CCLSEVT，CERREVT メッセージを破棄します。	E
出力キュー，出力メッセージ編集 UOC 障害	・出力キュー読み込み障害 ・UOC エラー	1. UOC エラー（KFCA10611-E），UOC パラメタ不正（KFCA10620-E）または出力キュー障害（KFCA10605-E）を通知するメッセージログを出力します。 2. 論理端末障害の端末タイプごとに次に示す処理をします。 ●request 型，request 型（同期型），send 型 以降の処理は「論理端末障害」の「論理端末層検出異常」と同じです。	F

障害の種類	障害内容	TP1/NET/OSAS-NIF の処理	ユーザの対応処理
出力キュー，出力メッセージ編集 UOC 障害	・出力キュー読み込み障害 ・UOC エラー	●reply 型 以降の処理は「UAP 異常」の「ERREVT3 未定義」と同じです。	F
出力キュー障害	・メッセージ送信済み障害 ・メッセージリセット障害	1. メッセージ送信完了処理で障害を通知するメッセージログ（KFCA10617-E）を出力します。 2. 障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 3. CERREVT を起動します。 4. 論理端末を閉塞します。	F
送信バッファ障害（通常メッセージ）	・バッファ長不足 ・バッファ数不足	1. メッセージ出力障害を通知するメッセージログ（KFCA10605-E）を出力します。 2. 論理端末の端末タイプごとに次に示す処理をします。	−
		●request 型，request 型（同期型），send 型 (a)メッセージ送信を中断します。 (b)送信中断を通知するメッセージログ（KFCA13508-E）を出力します。 (c)障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 (d)CERREVT を起動します。 (e)論理端末を閉塞します。	F
		●reply 型 (a)メッセージ送信を中断します。 (b)送信中断を通知するメッセージログ（KFCA13508-E）を出力します。 (c)障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 (d)CERREVT を起動します。 (e)論理端末を閉塞します。 (f)以降の処理は「コネクション障害」の「下位層障害」と同じです。	G
受信バッファ障害	・バッファ長不足 ・バッファ数不足	1. 受信バッファ不足を通知するメッセージログ（KFCA13577-E）を出力します。 2. コネクション確立済みの場合は次に示す処理をします。 (a)NIF アソシエーション解放を通知するメッセージログ（KFCA13551-I）を出力します。 (b)以降の処理は「コネクション障害」の「下位層障害」と同じです。 3. コネクション確立中の場合は次に示す処理をします。 (a)NIF アソシエーション確立失敗を通知するメッセージログ（KFCA13579-E）を出力します。 (b)以降の処理は「コネクション障害」の「コネクション確立処理の失敗」と同じです。	G

障害の種類	障害内容	TP1/NET/OSAS-NIF の処理	ユーザの対応処理
受信バッファ障害	・バッファ不足	1. 受信バッファサイズ値超過を通知するメッセージログ（KFCA13583-E）を出力します。 2. 以降の処理は「バッファ長不足，バッファ数不足」の 2.および 3.と同じです。	G
送信バッファ障害（制御メッセージ）	・バッファ長不足 ・バッファ数不足	1. 送信バッファ不足を通知するメッセージログ（KFCA13576-E）を出力します。 2. コネクション確立済みの場合は次に示す処理をします。 (a)NIF アソシエーション解放を通知するメッセージログ（KFCA13551-I）を出力します。 (b)以降の処理は「コネクション障害」の「下位層障害」と同じです。 3. コネクション確立中の場合，以降の処理は「NIF アソシエーション確立失敗」と同じです。	G
	・バッファ不足	1. 送信バッファサイズ値超過を通知するメッセージログ（KFCA13582-E）を出力します。 2. 以降の処理は「バッファ長不足，バッファ数不足」の 2.および 3.と同じです。	G
編集バッファ障害	・バッファ長不足	－	H
	・バッファ数不足	1. 入力メッセージの場合は次に示す処理をします。 (a)アプリケーション名取得失敗を通知するメッセージログ（KFCA10610-E）を出力します。 (b)論理端末の端末タイプごとに，次に示す処理をします。	－
		●request 型 (a)受信メッセージを破棄します。 (b)受信失敗を通知するメッセージログ（KFCA13506-E）を出力します。 (c)障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 (d)CERREVT を起動します。 (e)論理端末を閉塞します。 (f)以降の処理は「コネクション障害」の「下位層障害」と同じです。	G
		●request 型（同期型），reply 型，receive 型 「論理端末障害」の「論理端末層検出異常」と同じです。	F
		1. 出力メッセージの場合は，「送信バッファ障害（通常メッセージ）」と同じです。	F，G
UAP 型不一致	・入力メッセージと論理端末の端末タイプの不一致	1. アプリケーション型不正を通知するメッセージログ（KFCA13522-E）を出力します。 2. 受信メッセージを破棄します。 3. 相手システムへ受信拒否を送信します。 4. 障害発生を通知するメッセージログ（KFCA13503-E）を出力します。	F

障害の種類	障害内容	TP1/NET/OSAS-NIF の処理	ユーザの対応処理
UAP 型不一致	・入力メッセージと論理端末の端末タイプの不一致	5. CERREVT を起動します。 6. 論理端末を閉塞します。	F
UAP 障害	・UAP 異常終了 ・rollback 発行 など	1. ERREVT3 を起動します。 2. request 型論理端末（同期型）を使用している場合は「論理端末障害」の「論理端末層検出異常」と同じです。	E
	・ERREVT1 ERREVT2 ERREVT3 障害 ・ERREVT3 未定義	1. reply 型論理端末を使用している場合は次に示す処理をします。 (a)送信中断を通知するメッセージログ（KFCA13508-E）を出力します。 (b)相手システムへ UAP 異常を送信します。 (c)障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 (d)CERREVT を起動します。 (e)論理端末を閉塞します。	F
	・UAP 異常受信	1. request 型論理端末を使用している場合は次に示す処理をします。 (a)受信失敗を通知するメッセージログ（KFCA13506-E）を出力します。 (b)障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 (c)CERREVT を起動します。 (d)論理端末を閉塞します。	F
プロトコル障害	・シーケンスエラーなど	1. プロトコル違反を通知するメッセージログ（KFCA13573-E）を出力します。 2. コネクション確立済みの場合は次に示す処理をします。 (a)NIF アソシエーション解放を通知するメッセージログ（KFCA13551-I）を出力します。 (b)以降の処理は「コネクション障害」の「下位層障害」と同じです。 3. コネクション確立中の場合は「コネクション障害」の「コネクション確立処理の失敗」と同じです。	B
アソシエーション層プロトコル障害	・シーケンスエラー ・タイムアウト発生	1. アソシエーション層障害を通知するメッセージログ（KFCA13586-E）を出力します。 2. コネクション確立済みの場合は「プロトコル障害」の 2.と同じです。 3. コネクション確立中に障害が発生して，再試行不可能またはリトライオーバになった場合は，「プロトコル障害」の 2.と同じ処理をします。	B
不正データ受信	・未支援データ受信 ・不正データ受信	1. 不正データ受信を通知するメッセージログ（KFCA13570-E）を出力します。 2. コネクション確立中の場合は「プロトコル障害」の 3.と同じです。 3. コネクション解放中の場合は「プロトコル障害」の 2.と同じです。 4. メッセージ送受信中の場合は次に示す処理をします。	1.：－ 2.～ 3.：B 4.～ 5.：M

障害の種類	障害内容	TP1/NET/OSAS-NIF の処理	ユーザの対応処理
不正データ受信	・未支援データ受信 ・不正データ受信	(a)NIF-REJECT 送信を通知するメッセージログ（KFCA13569-E）を出力します。 (b)以降の処理は「処理中断連絡受信」の「NIF-REJECT 受信」の 2.～6.と同じです。 5. 論理端末閉塞解除処理中の場合は次に示す処理をします。 (a)NIF-REJECT 送信を通知するメッセージログ（KFCA13569-E）を出力します。 (b)以降の処理は「論理端末障害」の「論理端末閉塞解除失敗」と同じです。	1.：－ 2.～ 3.：B 4.～ 5.：M
緊急解放要求受信	・NIF-ABORT 受信	1. ABORT 受信を通知するメッセージログ（KFCA13571-E）を出力します。 2. コネクション確立済みの場合は「プロトコル障害」の 2.と同じです。 3. コネクション確立中に障害が発生して，再試行不可能またはリトライオーバになった場合は，「プロトコル障害」の 2.と同じ処理をします。	B
	・A-ABORT 受信 ・A-P-ABORT 受信	1.「アソシエーション層プロトコル障害」と同じです。	B
処理中断連絡受信	・NIF-REJECT 受信	1. NIF-REJECT 受信を通知するメッセージログ（KFCA13572-E）を出力します。 2. メッセージ送信中の場合，論理端末の端末タイプごとに次に示す処理をします。 ●request 型，reply 型，send 型 (a)メッセージ送信を中断します。 (b)送信中断を通知するメッセージログ（KFCA13508-E）を出力します。 ●request 型（同期型） (a) 送信メッセージを破棄します。 (b)送信中断を通知するメッセージログ（KFCA13508-E）を出力します。 3. メッセージ受信中の場合は次に示す処理をします。 (a)受信メッセージを破棄します。 (b)受信失敗を通知するメッセージログ（KFCA13506-E）を出力します。 4. 障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 5. CERREVT を起動します。 6. 論理端末を閉塞します。	M
	・受信拒否受信	1. 論理端末の端末タイプごとに次に示す処理をします。 ●request 型，send 型 (a)メッセージ送信を中断します。	N

障害の種類	障害内容	TP1/NET/OSAS-NIF の処理	ユーザの対応処理
処理中断連絡受信	・受信拒否受信	(b)送信中断を通知するメッセージログ（KFCA13508-E）を出力します。 ●request 型（同期型），reply 型 (a)送信メッセージを破棄します。 (b)送信中断を通知するメッセージログ（KFCA13508-E）を出力します。 2. 障害発生を通知するメッセージログ（KFCA13503-E）を出力します。 3. CERREVT を起動します。 4. 論理端末を閉塞します。	N
	・送信中断受信	1.「NIF-REJECT 受信」の 3.～6.と同じです。	N
タイムアウト	・t1，t2，t3，t4，t5 タイムアウト発生	1. タイムアウト発生を通知するメッセージログ（KFCA13580-E）を出力します。 2. コネクション確立中の場合は「緊急解放要求受信」の 3.と同じです。 3. コネクション解放中の場合は「プロトコル障害」の 2.と同じです。 4. メッセージ送受信中の場合は「不正データ受信」の 4.と同じです。 5. 論理端末閉塞解除処理中の場合は「不正データ受信」の 5.と同じです。	1.～3.：B 4.～5.：M
	・rplytim タイムアウト発生	1. 問い合わせメッセージ受信の場合は「UAP 障害」の「ERREVT3 未定義」と同じです。 2. 再送問い合わせメッセージ受信の場合は「論理端末障害」の「論理端末層検出異常」と同じです。	F
下位層障害	・XNF マクロエラー ・切断受信	1.「アソシエーション層プロトコル障害」と同じです。	B
内部インタフェース領域確保失敗	・共用メモリ不足	1. 共用メモリ不足を通知するメッセージログ（KFCA13578-E）を出力します。 2.「プロトコル障害」の 2.および 3.と同じです。	G
	・ローカルメモリ不足	1. ローカルメモリ不足を通知するメッセージログ（KFCA13575-E）を出力します。 2. 以降の処理は「プロトコル障害」の 2.および 3.と同じです。	G
	・ローカルメモリバッファ取得失敗	1. ローカルメモリバッファ取得失敗を通知するメッセージログ（KFCA13518-E）を出力します。 2. 処理を終了します。	E
着呼受付開始失敗	・XNF 終了中 ・受信バッファ不足	●着呼型コネクションの自動受付開始および過着呼による障害検出を使用しない場合 1. アソシエーション層障害を通知するメッセージログ（KFCA13586-E）を出力します。 2. CERREVT を起動します。	B

障害の種類	障害内容	TP1/NET/OSAS-NIF の処理	ユーザの対応処理
着呼受付開始失敗	・XNF 終了中 ・受信バッファ不足	●着呼型コネクションの自動受付開始または過着呼による障害検出を使用する場合 1. アソシエーション層障害を通知するメッセージログ（KFCA13586-E）を出力します。 2. 着呼受付開始に成功するまで 1 秒間隔で再試行します。	I
NIF アソシエーション確立失敗	・相手システムの PSAP アドレス未定義	●着呼型コネクションの自動受付開始および過着呼による障害検出を使用しない場合 1. NIF アソシエーション確立失敗を通知するメッセージログ（KFCA13579-E）を出力します。 2. 相手システムへ確立拒否を送信します。 3. CERREVT を起動します。	G
		●着呼型コネクションの自動受付開始を使用し，過着呼による障害検出を使用しない場合 1. NIF アソシエーション確立失敗を通知するメッセージログ（KFCA13579-E）を出力します。 2. 相手システムへ確立拒否を送信します。 3. CERREVT を起動します。	E
		●過着呼による障害検出を使用する場合 1. NIF アソシエーション確立失敗を通知するメッセージログ（KFCA13579-E）を出力します。 2. 相手システムへ確立拒否を送信します。	E
	・構成不一致	「NIF アソシエーション確立失敗」の「相手システムの PSAP アドレス未定義」と同じです。	G※ E※
開始処理失敗	・共用メモリ不足 ・ローカルメモリ不足 ・論理端末の窓口登録失敗 など	1. イニシャライズ処理打ち切りを通知するメッセージログ（KFCA13519-E，KFCA13520-E，KFCA13584-E）を出力します。 2. MCF プロセスが異常終了します。	J
再送処理失敗	・再送失敗	1. 再送処理失敗を通知するメッセージログ（KFCA13509-E）を出力します。 2. 再送処理を終了します。 3. 論理端末を閉塞解除します。	I
	・再送メッセージ受信失敗	1. 再送メッセージ受信失敗を通知するメッセージログ（KFCA13510-E）を出力します。 2. 再送処理を終了します。 3. 論理端末を閉塞解除します。	I
MCF の障害	・内部論理矛盾など	1. 内部矛盾を通知するメッセージログ（KFCA13547-E，KFCA13549-E，KFCA13574-E）を出力します。 2. メモリダンプを出力します。	K

障害の種類	障害内容	TP1/NET/OSAS-NIF の処理	ユーザの対応処理
MCF の障害	・内部論理矛盾など	3. 必要に応じ，MCF プロセス異常終了，コネクション解放，論理端末閉塞，処理続行します。	K
	・MCF プログラムエラー	1. OS によるコアダンプを出力します。 2. MCF プロセスを異常終了します。	K
	・UOC プログラムエラー	1.「MCF プログラムエラー」と同じです。	L

（凡例）

－：対応処理はありません。

注※

●着呼型コネクションの自動受付開始を使用しない場合：G

●着呼型コネクションの自動受付開始を使用する場合：E

表 9-2　ユーザの対応処理の詳細

ユーザの対応処理	ユーザの対応処理の詳細	採取資料			
		L	C	D	T
A	1. 次のどちらかの方法で再度コネクションを確立してください。 ・運用コマンド（mcftactcn）を入力する ・API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）を発行する	－	－	－	－
B	1. メッセージログで原因を調査してください。 2. 原因不明の場合は，資料を採取し保守員に連絡してください。 3. 原因を取り除いてください。 4. 次のどちらかの方法で再度コネクションを確立してください。 ・運用コマンド（mcftactcn）を入力する ・API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）を発行する	○	－	－	○
C	1. メッセージログで原因を調査してください。 2. 原因不明の場合は，資料を採取し保守員に連絡してください。 3. 原因を取り除いてください。 4. 次のどちらかの方法で再度論理端末を閉塞解除してください。 ・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	○	－	－	○
D	1. 次のどちらかの方法で再度論理端末を閉塞解除してください。 ・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	－	－	－	－
E	1. メッセージログで原因を調査してください。 2. 原因を取り除いてください。	－	－	－	－
F	1. メッセージログで原因を調査してください。	－	－	－	－

ユーザの対応処理	ユーザの対応処理の詳細	採取資料			
		L	C	D	T
F	2. 原因を取り除いてください。 3. 次のどちらかの方法で再度論理端末を閉塞解除してください。 ・運用コマンド（mcftactcn）を入力する ・API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）を発行する	−	−	−	−
G	1. メッセージログで原因を調査してください。 2. 原因を取り除いてください。 3. 次のどちらかの方法で再度コネクションを確立してください。 ・運用コマンド（mcftactcn）を入力する ・API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）を発行する	−	−	−	−
H	1. 必要に応じて UOC をリターンしてください。	−	−	−	−
I	1. メッセージログで原因を調査してください。 2. 原因不明の場合は，資料を採取し保守員に連絡してください。 3. 原因を取り除いてください。	○	−	−	○
J	1. メッセージログで原因を調査してください。 2. 原因を取り除いてください。 3. MCF を再開始してください。	−	−	−	−
K	1. 資料を採取して，保守員に連絡してください。 2. MCF を再開始してください。	○	○	○	○
L	1. 原因を取り除いてください。 2. MCF を再開始してください。	−	−	−	−
M	1. メッセージログで原因を調査してください。 2. 原因不明の場合は，資料を採取し保守員に連絡してください。 3. 原因を取り除いてください。 4. 次のどれかの方法で再度論理端末を閉塞解除してください。 ・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 ・相手システムの障害復旧をする	○	−	−	○
N	1. メッセージログで原因を調査してください。 2. 原因を取り除いてください。 3. 次のどれかの方法で再度論理端末を閉塞解除してください。 ・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 ・相手システムの障害復旧をする	−	−	−	−

（凡例）

L：メッセージログファイル

C：コアファイルおよび実行形式プログラムファイル

D：共用メモリダンプファイル

T：MCF イベントトレースファイル

○：採取する資料を示します。

－：該当しません。

9.2　通信形態と障害の処理

TP1/NET/OSAS-NIF でメッセージ処理中に発生する障害の発生個所と，TP1/NET/OSAS-NIF の処理を通信形態ごとに示します。

9.2.1　問い合わせ応答形態の障害

(1)　非応答型 UAP からの問い合わせメッセージ送信時の障害

非応答型 UAP からの問い合わせメッセージ送信時の障害発生個所を次の図に示します。また，非応答型 UAP からの問い合わせメッセージ送信時の障害の処理を表 9-3 に示します。該当する数字を参照してください。

図 9-1　非応答型 UAP からの問い合わせメッセージ送信時の障害発生個所

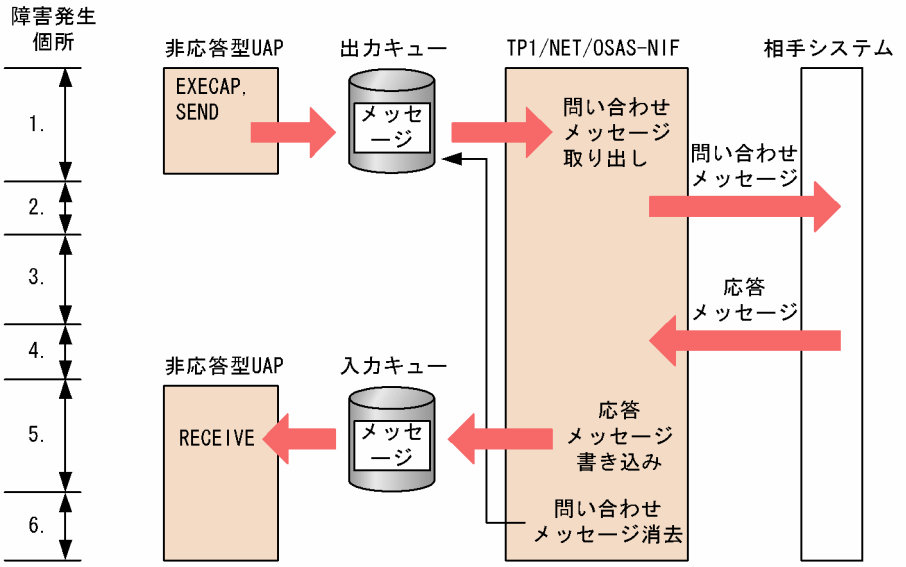


表 9-3　非応答型 UAP からの問い合わせメッセージ送信時の障害の処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	UAP でできる処理
1.	出力メッセージ編集 UOC 障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−	−
	出力キュー障害	同上	破棄	−	−
	バッファ取得障害	同上	保留	−	−
2.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。	保留	−	CERREVT 処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	UAP でできる処理
2.	コネクション障害	• コネクションを解放します。	保留	−	CERREVT 処理
	mcftdctle 入力	• メッセージログを出力します。 • 相手システムへ送信中断を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	保留	−	CERREVT 処理
	受信拒否受信	同上	保留	−	CERREVT 処理
	t3 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	保留	−	CERREVT 処理
	NIF-REJECT 受信	同上	保留	−	CERREVT 処理
3.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	保留	−	CERREVT 処理
	t2 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	保留	−	CERREVT 処理
	UAP 異常受信	同上	破棄	受信済み	CERREVT 処理
4.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	保留	破棄・未受信	CERREVT 処理
	t4 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	保留	破棄・未受信	CERREVT 処理
	NIF-REJECT 受信	同上	保留	破棄・未受信	CERREVT 処理
	送信中断受信	同上	破棄	破棄・受信済み	CERREVT 処理
5.	入力メッセージ編集 UOC 障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	破棄・受信済み	−
	UAP 型不一致	同上	破棄	破棄・受信済み	−

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	UAP でできる処理
5.	アプリケーション名取得障害	• ERREVT1 または ERREVT2 を起動します（起動できない場合は，メッセージログを出力後，CERREVT を起動して，論理端末を閉塞します）。	送信完了	ERREVT1 または ERREVT2 で受信（起動できない場合は破棄）	ERREVT1 または ERREVT2 処理
	入力キュー障害	同上	送信完了	ERREVT1 または ERREVT2 で受信（起動できない場合は破棄）	ERREVT1 または ERREVT2 処理
	バッファ取得障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	保留	破棄・未受信	CERREVT 処理
6.	メッセージ送信完了障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	－	－	CERREVT 処理

（凡例）

－：該当しません。

(2)　応答型 UAP からの問い合わせメッセージ送信時の障害

応答型 UAP からの問い合わせメッセージ送信時の障害発生個所を次の図に示します。また，応答型 UAP からの問い合わせメッセージ送信時の障害の処理を表 9-4 に示します。該当する数字を参照してください。

図 9-2　応答型 UAP からの問い合わせメッセージ送信時の障害発生個所

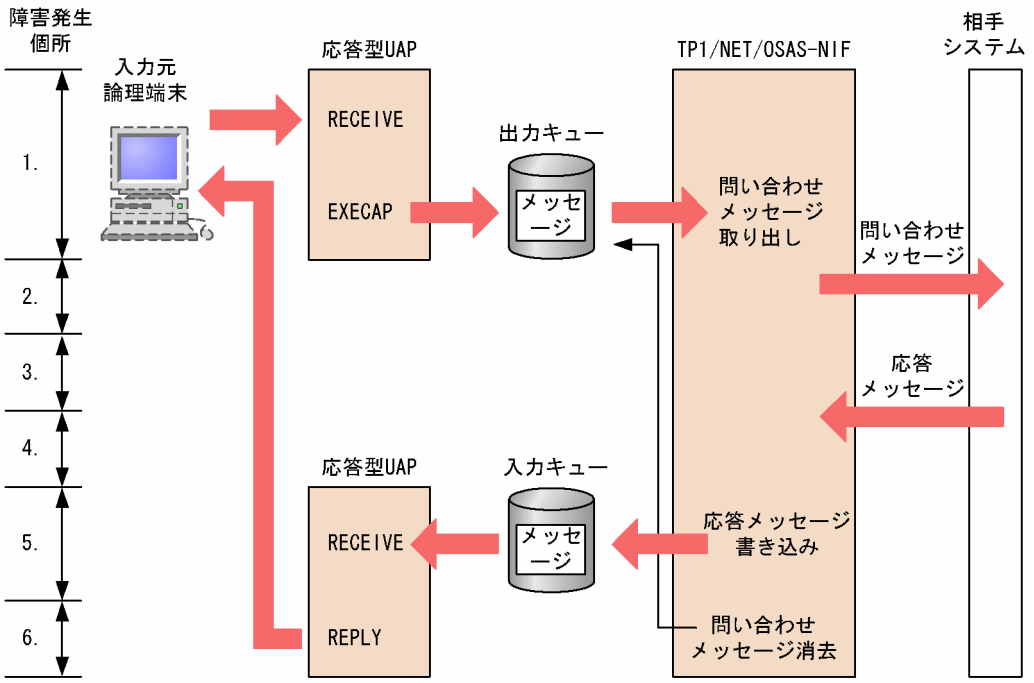


表 9-4　非応答型 UAP からの問い合わせメッセージ送信時の障害の処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	入力元論理端末への UAP 異常報告	UAP でできる処理
1.	出力メッセージ編集 UOC 障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−	あり	−
	出力キュー障害	同上	破棄	−	あり	−
	バッファ取得障害	同上	保留	−	−	−
2.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	保留	−	−	CERREVT 処理
	mcftdctle 入力	• メッセージログを出力します。 • 相手システムへ送信中断を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	保留	−	−	CERREVT 処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	入力元論理端末への UAP 異常報告	UAP でできる処理
2.	受信拒否受信	同上	保留	−	−	CERREVT 処理
	t3 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	保留	−	−	CERREVT 処理
	NIF-REJECT 受信	同上	保留	−	−	CERREVT 処理
3.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	保留	−	−	CERREVT 処理
	t2 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	保留	−	−	CERREVT 処理
	UAP 異常受信	同上	破棄	受信済み	あり	CERREVT 処理
4.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	保留	破棄・未受信	−	CERREVT 処理
	t4 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	保留	破棄・未受信	−	CERREVT 処理
	NIF-REJECT 受信	同上	保留	破棄・未受信	−	CERREVT 処理
	送信中断受信	同上	破棄	破棄・受信済み	あり	CERREVT 処理
5.	入力メッセージ編集 UOC 障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	破棄・受信済み	あり	−
	UAP 型不一致	同上	破棄	破棄・受信済み	あり	−

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	入力元論理端末への UAP 異常報告	UAP でできる処理
5.	アプリケーション名取得障害	• ERREVT1 を起動します（起動できない場合は，メッセージログを出力後，CERREVT を起動して，論理端末を閉塞します）。	送信完了	ERREVT1 で受信（起動できない場合は破棄）	あり	ERREVT1 処理
	入力キュー障害	• ERREVT2 を起動します（起動できない場合は，メッセージログを出力後，CERREVT を起動して，論理端末を閉塞します）。	送信完了	ERREVT2 で受信（起動できない場合は破棄）	−	ERREVT2 処理 REPLY 要求で入力元論理端末へメッセージ送信
	バッファ取得障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	保留	破棄・未受信	−	CERREVT 処理
6.	メッセージ送信完了障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	−	−	CERREVT 処理

（凡例）

−：該当しません。

(3) 応答メッセージ送信時の障害

応答メッセージ送信時の障害発生個所を次の図に示します。また，応答メッセージ送信時の障害の処理を表 9-5 に示します。該当する数字を参照してください。

図 9-3　応答メッセージ送信時の障害発生個所

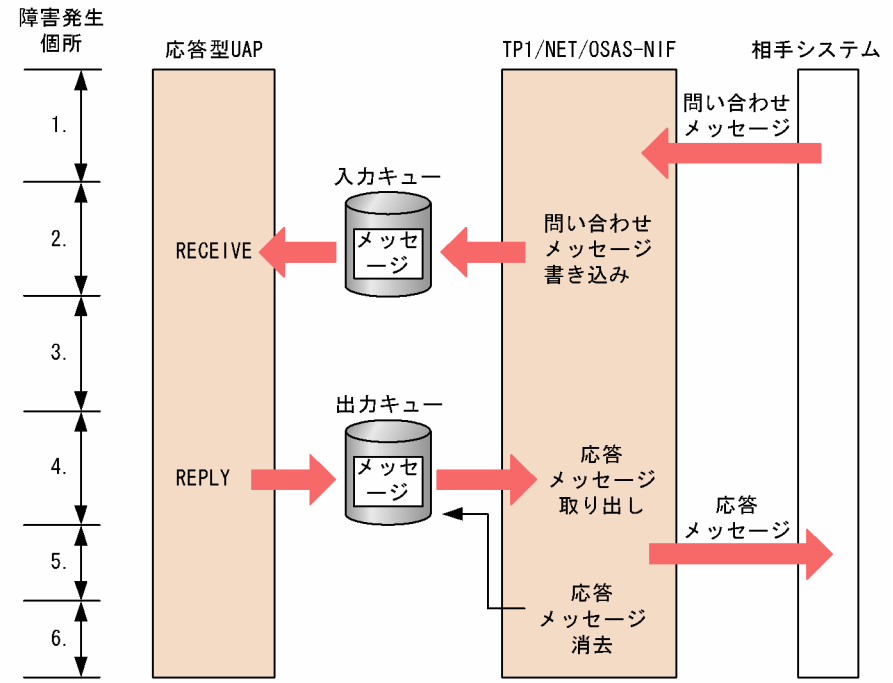


表 9-5　応答メッセージ送信時の障害の処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	UAP でできる処理
1.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	破棄	−	CERREVT 処理
	mcftdctle 入力	• メッセージログを出力します。 • 相手システムへ受信拒否を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−	CERREVT 処理
	t4 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−	CERREVT 処理
	NIF-REJECT 受信	同上	破棄	−	CERREVT 処理
	送信中断受信	同上	破棄	−	CERREVT 処理
2.	入力メッセージ編集 UOC 障害	• メッセージログを出力します。 • 相手システムへ受信拒否を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−	−

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	UAP でできる処理
2.	UAP 型不一致	同上	破棄	－	－
	バッファ取得障害	同上	破棄	－	－
	アプリケーション名取得障害	• ERREVT1 または ERREVT2 起動します（起動できない場合は，メッセージログを出力後，相手システムへ受信拒否を送信してから，CERREVT を起動して，論理端末を閉塞します)。	ERREVT1 または ERREVT2 で受信（起動できない場合は破棄）	－	ERREVT1 または ERREVT2 処理 REPLY 要求で相手システムへメッセージ送信
	入力キュー障害	同上	ERREVT1 または ERREVT2 で受信（起動できない場合は破棄）	－	ERREVT1 または ERREVT2 処理 REPLY 要求で相手システムへメッセージ送信
3.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	－	未送信	CERREVT 処理
	NIF-REJECT 受信	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	－	未送信	CERREVT 処理
	rplytim タイムアウト発生	• メッセージログを出力します。 • 相手システムへ UAP 異常を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	－	送信済み	－
	UAP 異常終了	同上	－	送信済み	－
4.	出力メッセージ編集 UOC 障害	• メッセージログを出力します。 • 相手システムへ UAP 異常を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	－	送信済み	－
	出力キュー障害	同上	－	送信済み	－
	バッファ取得障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	－	未送信	CERREVT 処理
5.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。	－	未送信	CERREVT 処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	UAP でできる処理
5.	コネクション障害	• コネクションを解放します。	−	未送信	CERREVT 処理
	受信拒否受信	• メッセージログを出力します。 • 相手システムへ送信中断を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	送信済み	−
	t3 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	未送信	−
	NIF-REJECT 受信	同上	−	未送信	−
6.	メッセージ送信完了障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	−	CERREVT 処理

(凡例)

−：該当しません。

9.2.2 同期型問い合わせ応答形態の障害

同期型の問い合わせメッセージ送信時の障害発生個所を次の図に示します。また，同期型の問い合わせメッセージ送信時の障害の処理を表 9-6 に示します。該当する数字を参照してください。

図 9-4 同期型の問い合わせメッセージ送信時の障害発生個所

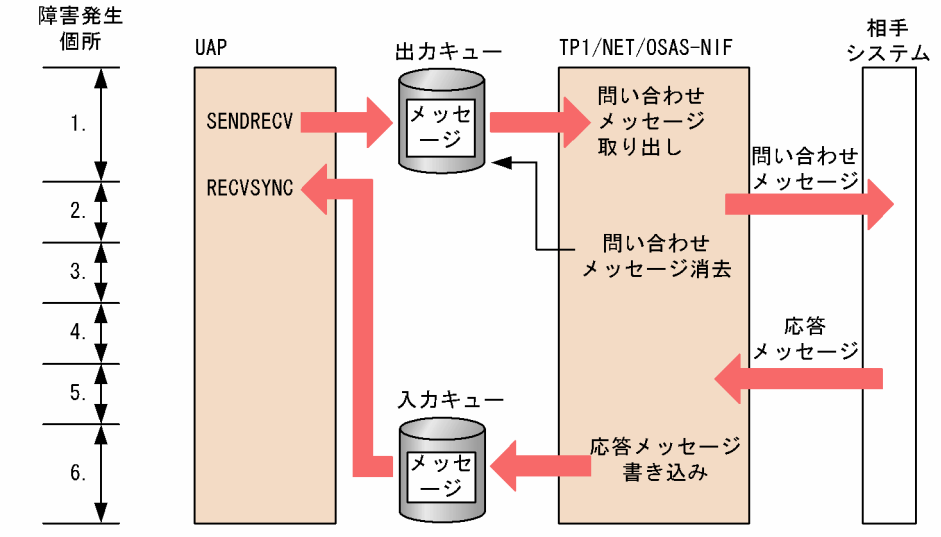


表 9-6　同期型の問い合わせメッセージ送信時の障害の処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	UAP でできる処理
1.	出力メッセージ編集 UOC 障害	• メッセージログを出力します。 • UAP へエラーリターンします。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−	UAP 処理
	出力キュー障害	同上	破棄	−	UAP 処理
	バッファ取得障害	同上	破棄	−	UAP 処理
2.	コネクション障害	• メッセージログを出力します。 • UAP へエラーリターンします。 • CERREVT を起動します。 • コネクションを解放します。	破棄	−	UAP 処理
	mcftdctle 入力	• メッセージログを出力します。 • UAP へエラーリターンします。 • 相手システムへ送信中断を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−	UAP 処理
	受信拒否受信	同上	破棄	−	UAP 処理
	t3 タイムアウト発生	• メッセージログを出力します。 • UAP へエラーリターンします。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−	UAP 処理
	NIF-REJECT 受信	同上	破棄	−	UAP 処理
	UAP 異常終了	• ERREVT2 または ERREVT3 を起動します。 • メッセージログを出力します。 • 相手システムへ送信中断を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−	ERREVT2 または ERREVT3 処理
3.	メッセージ送信完了障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	UAP で受信	CERREVT 処理
4.	コネクション障害	• メッセージログを出力します。	−	受信済み	UAP 処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	UAP でできる処理
4.	コネクション障害	• UAP へエラーリターンします。 • CERREVT を起動します。 • コネクションを解放します。	−	受信済み	UAP 処理
	t2 タイムアウト発生	• メッセージログを出力します。 • UAP へエラーリターンします。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	受信済み	UAP 処理
	UAP 異常受信	同上	−	受信済み	UAP 処理
	UAP 異常終了	• ERREVT2 または ERREVT3 を起動します。 • メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	破棄・受信済み	ERREVT2 または ERREVT3 処理
5.	コネクション障害	• メッセージログを出力します。 • UAP へエラーリターンします。 • CERREVT を起動します。 • コネクションを解放します。	−	破棄・受信済み	UAP 処理
	t2 タイムアウト発生	• メッセージログを出力します。 • UAP へエラーリターンします。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	破棄・受信済み	UAP 処理
	NIF-REJECT 受信	同上	−	破棄・受信済み	UAP 処理
	送信中断受信	同上	−	破棄・受信済み	UAP 処理
	UAP 異常終了	• ERREVT2 または ERREVT3 を起動します。 • メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	破棄・受信済み	ERREVT2 または ERREVT3 処理
6.	入力メッセージ編集 UOC 障害	• メッセージログを出力します。 • UAP へエラーリターンします。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	破棄・受信済み	UAP 処理
	バッファ取得障害	同上	−	破棄・受信済み	UAP 処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	問い合わせメッセージの扱い	応答メッセージの扱い	UAP でできる処理
6.	入力キュー障害	同上	−	破棄・受信済み	UAP 処理
	応答エラー	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	−	CERREVT 処理

(凡例)

−：該当しません。

9.2.3 分岐送信形態の障害

一方送信メッセージ送信時の障害発生個所を次の図に示します。また，一方送信メッセージ送信時の障害の処理を表 9-7 に示します。該当する数字を参照してください。

図 9-5 一方送信メッセージ送信時の障害発生個所

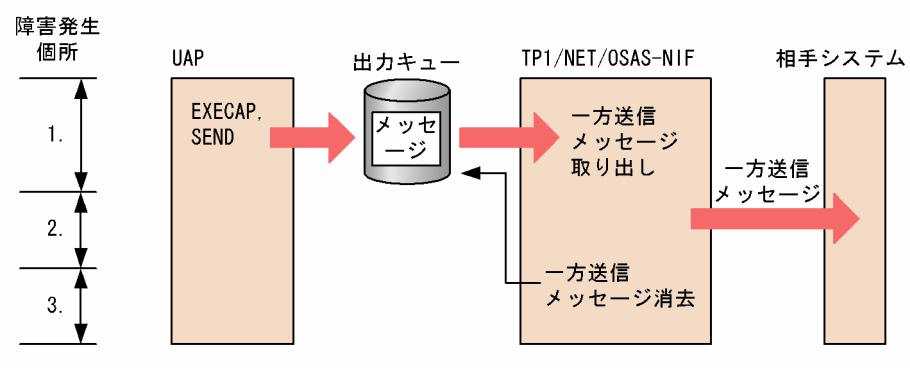


表 9-7 一方送信メッセージ送信時の障害の処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	送信メッセージの扱い	UAP でできる処理
1.	出力メッセージ編集 UOC 障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−
	出力キュー障害	同上	破棄	−
	バッファ取得障害	同上	保留	−
2.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	保留	CERREVT 処理
	mcftdctle 入力	• メッセージログを出力します。 • 相手システムへ送信中断を送信します。 • CERREVT を起動します。	保留	CERREVT 処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	送信メッセージの扱い	UAP でできる処理
2.	mcftdctle 入力	• 論理端末を閉塞します。	保留	CERREVT 処理
	受信拒否受信	同上	保留	CERREVT 処理
	t3 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	保留	CERREVT 処理
	NIF-REJECT 受信	同上	保留	CERREVT 処理
3.	メッセージ送信完了障害	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	−	CERREVT 処理

(凡例)

−：該当しません。

9.2.4 一方受信形態の障害

一方送信メッセージ受信時の障害発生個所を次の図に示します。また，一方送信メッセージ受信時の障害の処理を表 9-8 に示します。該当する数字を参照してください。

図 9-6 一方送信メッセージ受信時の障害発生個所

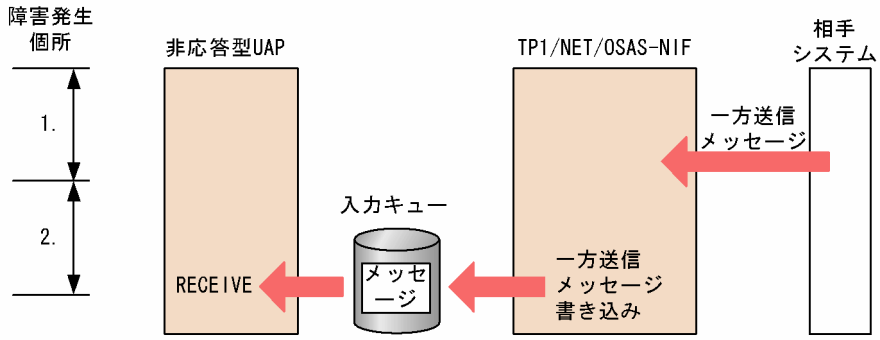


表 9-8 一方送信メッセージ受信時の障害の処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	受信メッセージの扱い	UAP でできる処理
1.	コネクション障害	• メッセージログを出力します。 • CERREVT を起動します。 • コネクションを解放します。	破棄	CERREVT 処理

発生個所	内容	TP1/NET/OSAS-NIF の処理	受信メッセージの扱い	UAP でできる処理
1.	mcftdctle 入力	• メッセージログを出力します。 • 相手システムへ受信拒否を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	CERREVT 処理
	t4 タイムアウト発生	• メッセージログを出力します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	CERREVT 処理
	NIF-REJECT 受信	同上	破棄	CERREVT 処理
	送信中断受信	同上	破棄	CERREVT 処理
2.	入力メッセージ編集 UOC 障害	• メッセージログを出力します。 • 相手システムへ受信拒否を送信します。 • CERREVT を起動します。 • 論理端末を閉塞します。	破棄	−
	UAP 型不一致	同上	破棄	−
	バッファ取得障害	同上	破棄	−
	アプリケーション名取得障害	• ERREVT1 または ERREVT2 を起動します（起動できない場合は，メッセージログを出力後，相手システムへ受信拒否を送信してから，CERREVT を起動し，論理端末を閉塞します）。	ERREVT1 または ERREVT2 で受信（起動できない場合破棄）	ERREVT1 または ERREVT2 処理
	入力キュー障害	同上	ERREVT1 または ERREVT2 で受信（起動できない場合破棄）	ERREVT1 または ERREVT2 処理

（凡例）

−：該当しません。

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/OSAS-NIF 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/OSAS-NIF 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 • SYNCHRONOUS MODE 句の設定値に ASYNC を追加 • SYNCHRONOUS MODE 句のデータ名 6 の設定値に 0，空白を追加
	定義	• なし
	コマンド	• mcfosnf コマンド • -r オプション
変更		なし
削除		なし

TP1/NET/OSAS-NIF 07-50 での動作の変更点はありません。

TP1/NET/OSAS-NIF 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/OSAS-NIF 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/OSAS-NIF 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	プロトコル共通定義（mcftpcmn） • -e オプションの termactb オペランド • -r オプションの autoonln オペランド • -r オプションの errdetect オペランド

種別	分類	内容
追加	コマンド	• なし
変更		なし
削除		なし

TP1/NET/OSAS-NIF 07-01 での動作の変更点はありません。

TP1/NET/OSAS-NIF 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/OSAS-NIF 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/OSAS-NIF 07-00 での動作の変更点はありません。

TP1/NET/OSAS-NIF 07-00 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

旧製品から移行する場合の注意事項を示します。

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。

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	ソースファイルを変更しないで使用できます。※
	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	ソースファイルを変更しないで使用できます。※
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が，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 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_execap	3. dc_mcf_execap － アプリケーションプログラムの起動（C 言語）
	dc_mcf_receive	3. dc_mcf_receive － メッセージの受信（C 言語）
	dc_mcf_recvsync	3. dc_mcf_recvsync － 同期型メッセージの後続セグメント受信（C 言語）
	dc_mcf_reply	3. dc_mcf_reply － 応答メッセージの送信（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 － メッセージの送信（C 言語）
	dc_mcf_sendrecv	3. dc_mcf_sendrecv － 同期型メッセージの送受信（C 言語）
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_execap － アプリケーションプログラムの起動

(a)　ANSI C，C++の形式

バージョン 6 以前

```
#include <dcmcf.h>
int  dc_mcf_execap(long action,
                   long commform,
                   char *resv01,
                   long active,
```

バージョン 7

```
#include <dcmcf.h>
int  dc_mcf_execap(DCLONG action,
                   DCLONG commform,
                   char *resv01,
                   DCLONG active,
```

バージョン 6 以前	バージョン 7
char *apnam, char *comdata, long cdataleng)	char *apnam, char *comdata, DCLONG cdataleng)

(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_execap(action, commform, resv01, active, apnam, comdata, cdataleng) long action; long commform; char *resv01; long active; char *apnam; char *comdata; long cdataleng;	#include <dcmcf.h> int dc_mcf_execap(action, commform, resv01, active, apnam, comdata, cdataleng) DCLONG action; DCLONG commform; char *resv01; DCLONG active; char *apnam; char *comdata; DCLONG cdataleng;

(2) dc_mcf_receive － メッセージの受信

(a) ANSI C，C++の形式

バージョン 6 以前	バージョン 7
#include <dcmcf.h> int dc_mcf_receive(long action, long commform, char *termnam, char *resv01, char *recvdata, long *rdataleng, long inbufleng, long *time)	#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;	#include <dcmcf.h> int dc_mcf_receive(action, commform, termnam, resv01, recvdata, rdataleng, inbufleng, time) DCLONG action; DCLONG commform; char *termnam; char *resv01;

バージョン 6 以前	バージョン 7
char *recvdata; long *rdataleng; long inbufleng; long *time;	char *recvdata; DCLONG *rdataleng; DCLONG inbufleng; DCLONG *time;

(3) dc_mcf_recvsync － 同期型メッセージの後続セグメント受信

(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_recvsync(long action, long commform, char *termnam, char *resv01, char *recvdata, long *rdataleng, long inbufleng, long *time, long resv02)	#include <dcmcf.h> int dc_mcf_recvsync(DCLONG action, DCLONG commform, char *termnam, char *resv01, char *recvdata, DCLONG *rdataleng, DCLONG inbufleng, DCLONG *time, DCLONG resv02)

(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_recvsync(action, commform, termnam, resv01, recvdata, rdataleng, inbufleng, time, resv02) long action; long commform; char *termnam; char *resv01; char *recvdata; long *rdataleng; long inbufleng; long *time; long resv02;	#include <dcmcf.h> int dc_mcf_recvsync(action, commform, termnam, resv01, recvdata, rdataleng, inbufleng, time, resv02) DCLONG action; DCLONG commform; char *termnam; char *resv01; char *recvdata; DCLONG *rdataleng; DCLONG inbufleng; DCLONG *time; DCLONG resv02;

(4) dc_mcf_reply － 応答メッセージの送信

(a) ANSI C，C++の形式

バージョン 6 以前	バージョン 7
#include <dcmcf.h> int dc_mcf_reply(long action, long commform,	#include <dcmcf.h> int dc_mcf_reply(DCLONG action, DCLONG commform,

バージョン 6 以前	バージョン 7
char *resv01, char *resv02, char *senddata, long sdataleng, char *resv03, long opcd)	char *resv01, char *resv02, char *senddata, DCLONG sdataleng, char *resv03, 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_reply(action, commform, resv01, resv02, senddata, sdataleng, resv03, opcd) long action; long commform; char *resv01; char *resv02; char *senddata; long sdataleng; char *resv03; long opcd;	#include <dcmcf.h> int dc_mcf_reply(action, commform, resv01, resv02, senddata, sdataleng, resv03, opcd) DCLONG action; DCLONG commform; char *resv01; char *resv02; char *senddata; DCLONG sdataleng; char *resv03; DCLONG opcd;

(5) 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,	#include <dcmcf.h> int dc_mcf_resend(action, commform, rtermnam, resv01,

バージョン 6 以前

```
                    oseqid,
                    orgseq,
                    otermnam,
                    resv02,
                    resv03,
                    resv04,
                    opcd)
long        action;
long        commform;
char        *rtermnam;
char        *resv01;
long        oseqid;
long        orgseq;
char        *otermnam;
char        *resv02;
char        *resv03;
char        *resv04;
long        opcd;
```

バージョン 7

```
                    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;
```

(6) dc_mcf_send － 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;
```

(7) dc_mcf_sendrecv － 同期型のメッセージの送受信

(a) ANSI C，C++の形式

バージョン６以前

```
#include <dcmcf.h>
int  dc_mcf_sendrecv(long action,
                     long commform,
                     char *termnam,
                     char *resv01,
                     char *senddata,
                     long sdataleng,
                     char *recvdata,
                     long *rdataleng,
                     long inbufleng,
                     long *time,
                     long resv02)
```

バージョン７

```
#include <dcmcf.h>
int  dc_mcf_sendrecv(DCLONG action,
                     DCLONG commform,
                     char *termnam,
                     char *resv01,
                     char *senddata,
                     DCLONG sdataleng,
                     char *recvdata,
                     DCLONG *rdataleng,
                     DCLONG inbufleng,
                     DCLONG *time,
                     DCLONG resv02)
```

(b) K&R 版 C の形式

バージョン６以前

```
#include <dcmcf.h>
int  dc_mcf_sendrecv(action,
                     commform,
                     termnam,
                     resv01,
                     senddata,
                     sdataleng,
                     recvdata,
                     rdataleng,
                     inbufleng,
                     time,
                     resv02)
long       action;
long       commform;
char       *termnam;
char       *resv01;
char       *senddata;
long       sdataleng;
char       *recvdata;
long       *rdataleng;
long       inbufleng;
long       *time;
long       resv02;
```

バージョン７

```
#include <dcmcf.h>
int  dc_mcf_sendrecv(action,
                     commform,
                     termnam,
                     resv01,
                     senddata,
                     sdataleng,
                     recvdata,
                     rdataleng,
                     inbufleng,
                     time,
                     resv02)
DCLONG     action;
DCLONG     commform;
char       *termnam;
char       *resv01;
char       *senddata;
DCLONG     sdataleng;
char       *recvdata;
DCLONG     *rdataleng;
DCLONG     inbufleng;
DCLONG     *time;
DCLONG     resv02;
```

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

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

バージョン 7

```
#include <dcmcf.h>
#include <dcmnom.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

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

dcmcf_uoc_min_n *parm;
```

バージョン 7

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

dcmcf_uoc_min_n *parm;
```

(b)　パラメタの内容

dcmcf_uoc_min_n の内容

バージョン 6 以前

```
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;
```

バージョン 7

```
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

```
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;
```

バージョン 7

```
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	バージョン 7
#include <dcmcfuoc.h> long uoc_func(dcmcf_uoc_mout_n *parm)	#include <dcmcfuoc.h> DCLONG uoc_func(dcmcf_uoc_mout_n *parm)

K&R 版 C の形式

バージョン 6 以前	バージョン 7
#include <dcmcfuoc.h> long uoc_func(parm) dcmcf_uoc_mout_n *parm;	#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	バージョン 7

typedef struct {
    long pro_kind;
    char le_name[9];
    char reserve1[7];
    dcmcf_uocbuff_list_n *buflist_adr;
    dcmcf_uocbuff_list_n *ebuflist_adr;
    long output_no;
    char msg_type;
    char outputno_flag;
    char resend_flag;
    char reserve2[1];
    char *pro_indv_ifa;
    long rtn_detail;
    char reserve3[20];
} dcmcf_uoc_mout_n;
	typedef struct {
	    DCLONG pro_kind;
	    char le_name[9];
	    char reserve1[7];
	    dcmcf_uocbuff_list_n *buflist_adr;
	    dcmcf_uocbuff_list_n *ebuflist_adr;
	    DCLONG output_no;
	    char msg_type;
	    char outputno_flag;
	    char resend_flag;
	    char reserve2[1];
	    char *pro_indv_ifa;
	    DCLONG rtn_detail;
	    char reserve3[20];
	} dcmcf_uoc_mout_n;

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	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];
    long user_leng;
    char user_data[16];
    char reserve07[16];
} ;
	struct dc_mcf_evta_type {
	    struct dc_mcf_evtheader evtheader;
	    char reserve01[12];
	    char map_name[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	バージョン 7

typedef struct {
    struct dc_mcf_evtheader header;
    long err_fact;
    long err_reason1;
    long err_reason2;
    char reserve1[44];
} dcmnom_cerrevt;
	typedef struct {
	    struct dc_mcf_evtheader header;
	    DCLONG err_fact;
	    DCLONG err_reason1;
	    DCLONG err_reason2;
	    char reserve1[44];
	} dcmnom_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 <dcmosnf.h>                          /*TP1/NET/OSAS-NIF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   */

    dcmcf_svstart();                       /*スタート関数呼び出し                */
    return 0;
}
```

(b) バージョン７

```
#include <dcmosnf.h>                          /*TP1/NET/OSAS-NIF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   */

    dcmcf_svstart();                       /*スタート関数呼び出し                */
    return 0;
}
```

(2)　K&R 版 C の場合

(a)　バージョン 6 以前

```
#include <dcmosnf.h>                    /*TP1/NET/OSAS-NIF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                */

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();                    /*スタート関数呼び出し                */
}
```

(b)　バージョン 7

```
#include <dcmosnf.h>                    /*TP1/NET/OSAS-NIF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

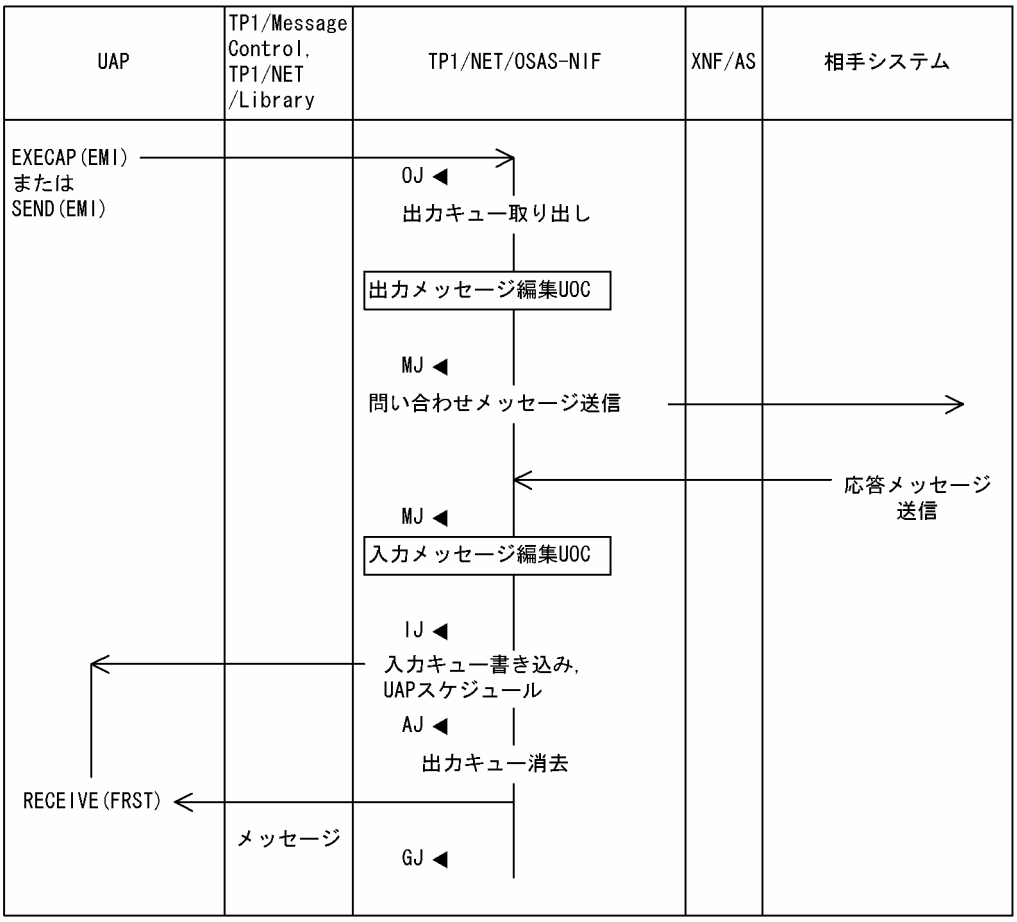
メッセージを送受信するときのデータの流れ，およびジャーナルの取得タイミングについて次に示します。

付録 D.1　問い合わせメッセージの送信

問い合わせメッセージを送信するときの処理の流れを，図 D-1，図 D-2 に示します。

(1)　単一セグメントの場合

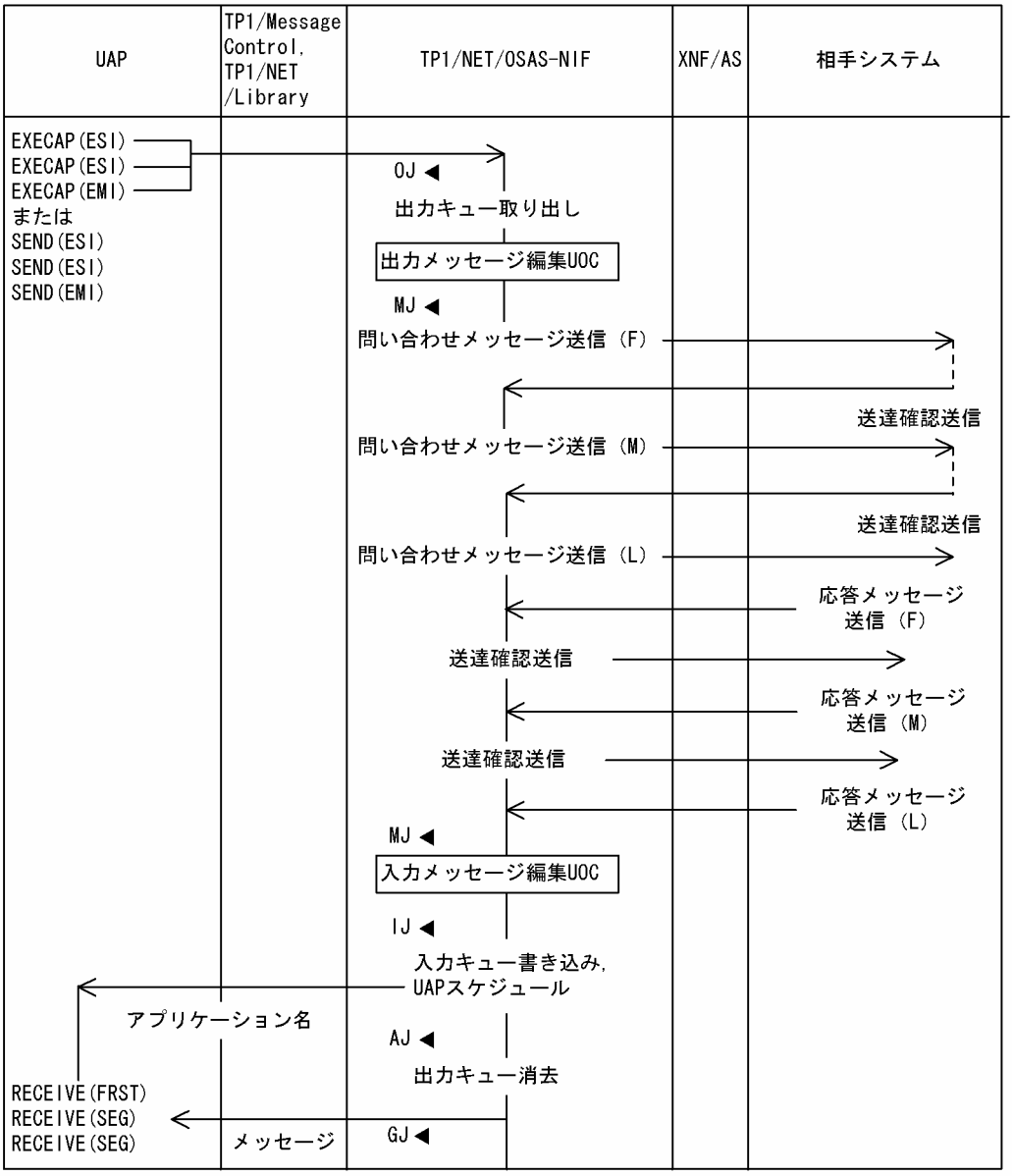
図 D-1　問い合わせメッセージの送信と応答メッセージの受信の処理の流れ（単一セグメントの場合）



（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
IJ◀：メッセージ入力ジャーナル取得
AJ◀：メッセージ送信完了ジャーナル取得
GJ◀：メッセージ受信ジャーナル取得

(2)　セグメント分割送信とセグメント組み立て受信の場合

図 D-2　問い合わせメッセージの送信と応答メッセージの受信の処理の流れ（セグメント分割送信とセグメント組み立て受信の場合）



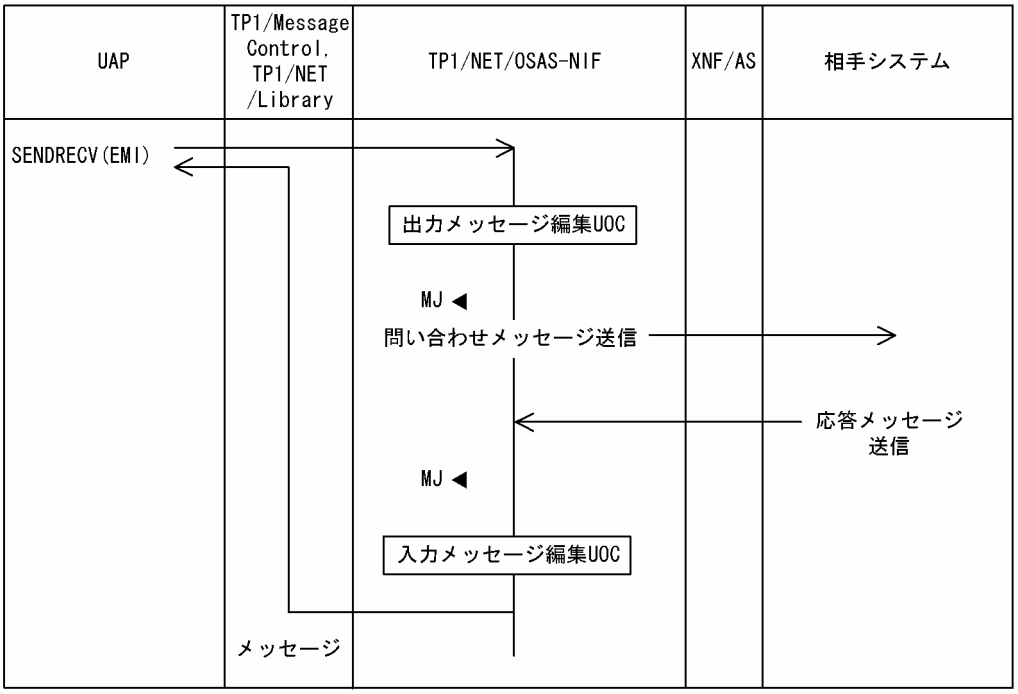
（凡例）　OJ◀：メッセージ出力ジャーナル取得
MJ◀：メッセージジャーナル取得
IJ◀：メッセージ入力ジャーナル取得
AJ◀：メッセージ送信完了ジャーナル取得
GJ◀：メッセージ受信ジャーナル取得

付録 D.2　同期型の問い合わせメッセージの送信

同期型の問い合わせメッセージを送信するときの処理の流れを，図 D-3，図 D-4 に示します。

(1) 単一セグメントの場合

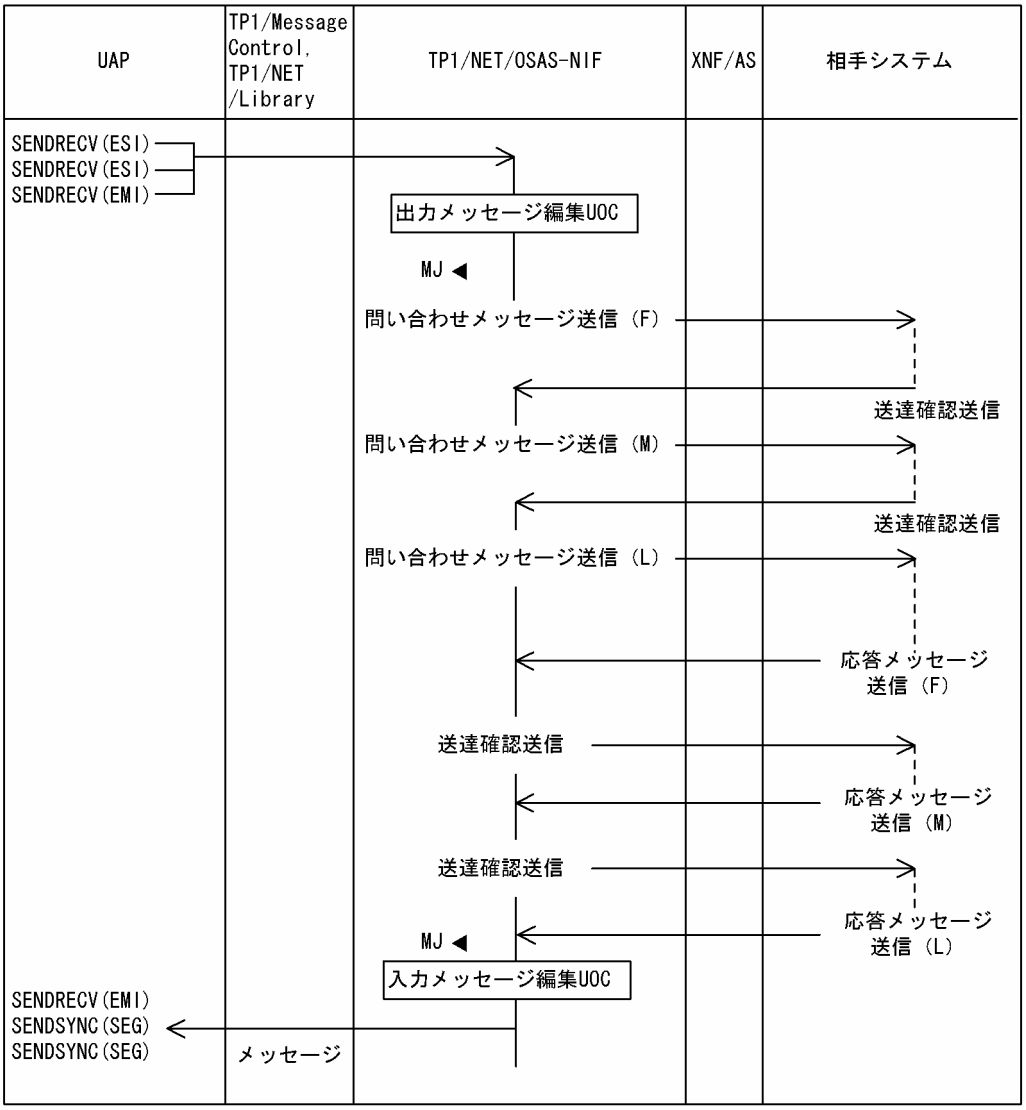
図 D-3 同期型の問い合わせメッセージの送信と応答メッセージの受信の処理の流れ（単一セグメントの場合）



（凡例） MJ◀：メッセージジャーナル取得

(2)　セグメント分割送信とセグメント組み立て受信の場合

図 D-4　同期型の問い合わせメッセージの送信と応答メッセージの受信の処理の流れ（セグメント分割送信とセグメント組み立て受信の場合）



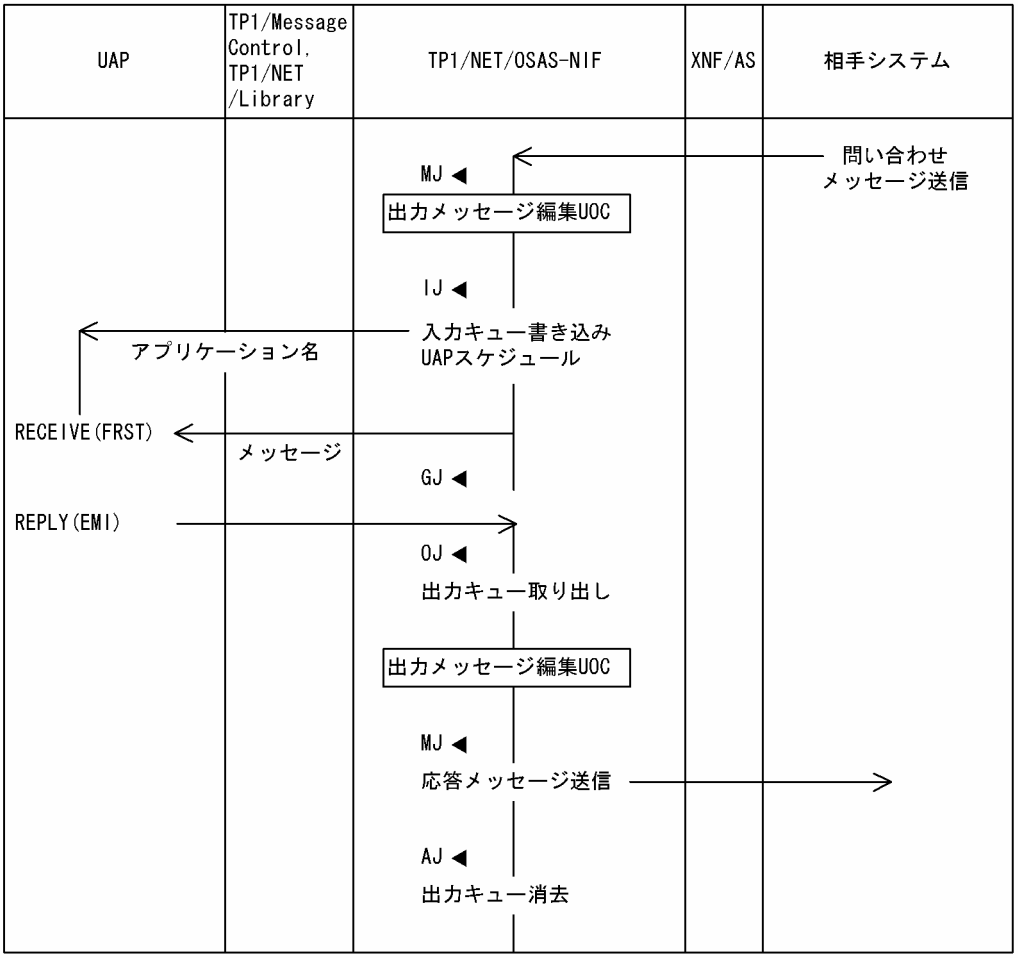
（凡例）　MJ◀：メッセージジャーナル取得

付録 D.3　応答メッセージの送信

応答メッセージを送信するときの処理の流れを，図 D-5，図 D-6 に示します。

(1) 単一セグメントの場合

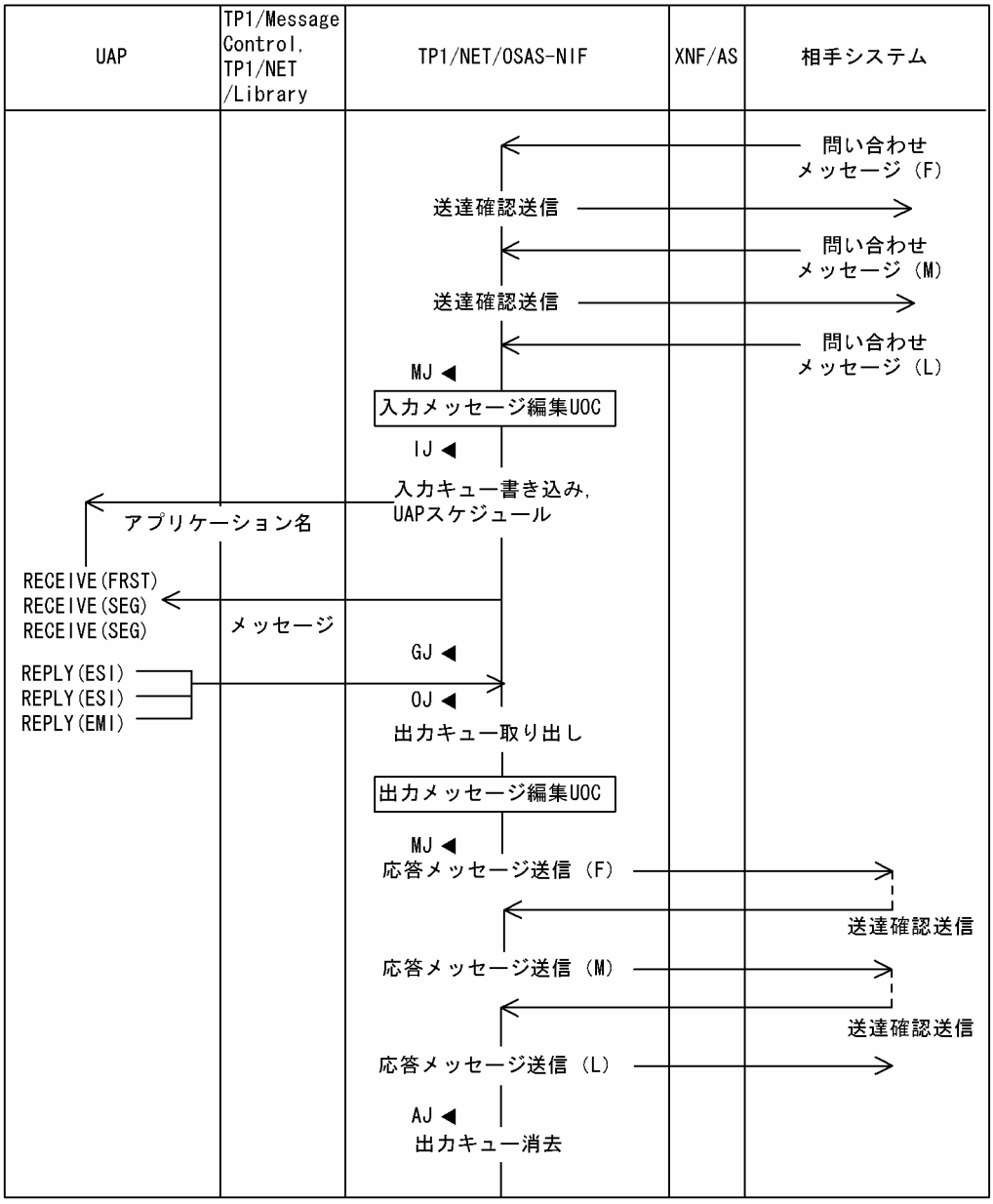
図 D-5 問い合わせメッセージの受信と応答メッセージの送信の処理の流れ（単一セグメントの場合）



（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
OJ◀：メッセージ出力ジャーナル取得
AJ◀：メッセージ送信完了ジャーナル取得

(2)　セグメント組み立て受信とセグメント分割送信の場合

図 D-6　問い合わせメッセージの受信と応答メッセージの送信の処理の流れ（セグメント組み立て受信とセグメント分割送信の場合）



（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

OJ◀：メッセージ出力ジャーナル取得

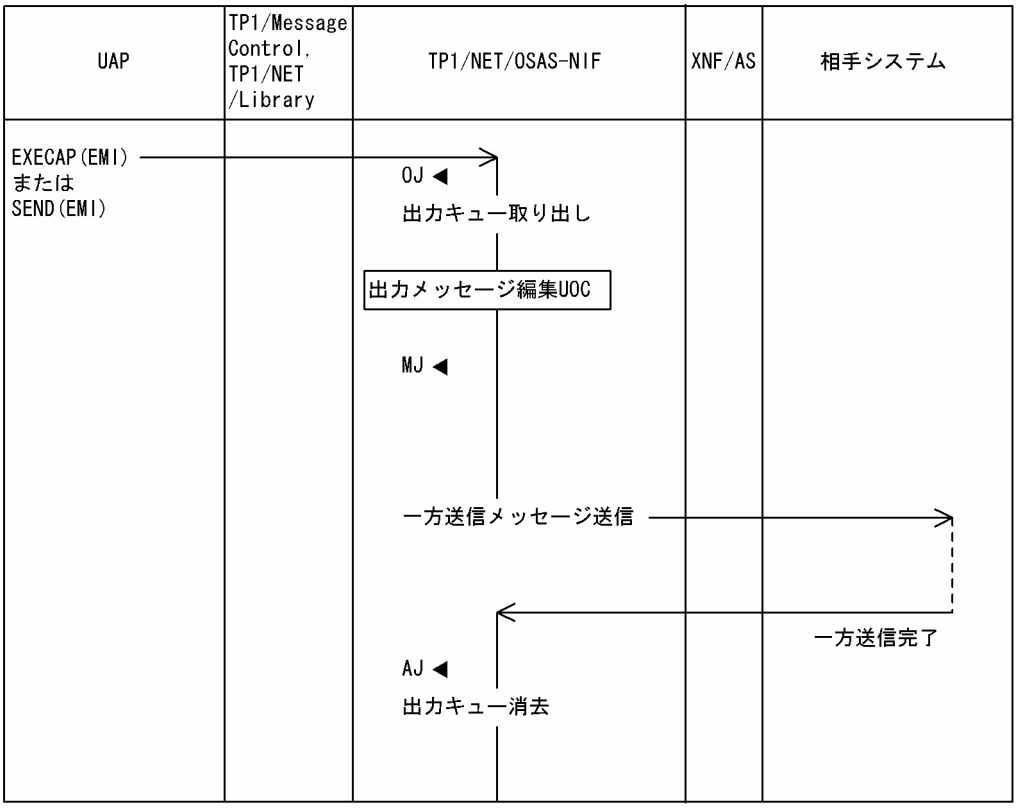
AJ◀：メッセージ送信完了ジャーナル取得

付録 D.4　一方送信メッセージの送信

一方送信メッセージを送信するときの処理の流れを，図 D-7，図 D-8 に示します。

(1) 単一セグメントの場合

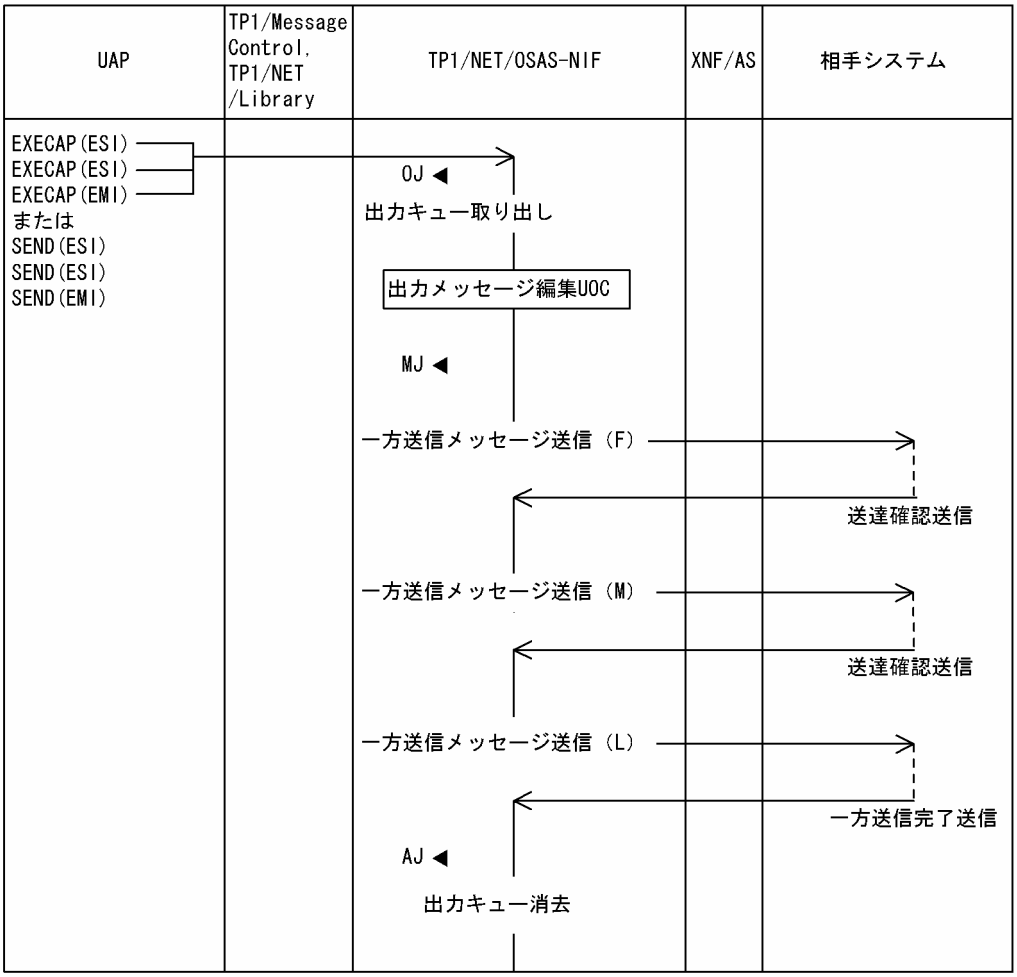
図 D-7 一方送信メッセージ送信の処理の流れ（単一セグメントの場合）



（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

(2)　セグメント分割送信の場合

図 D-8　一方送信メッセージの送信処理の流れ（セグメント分割送信の場合）



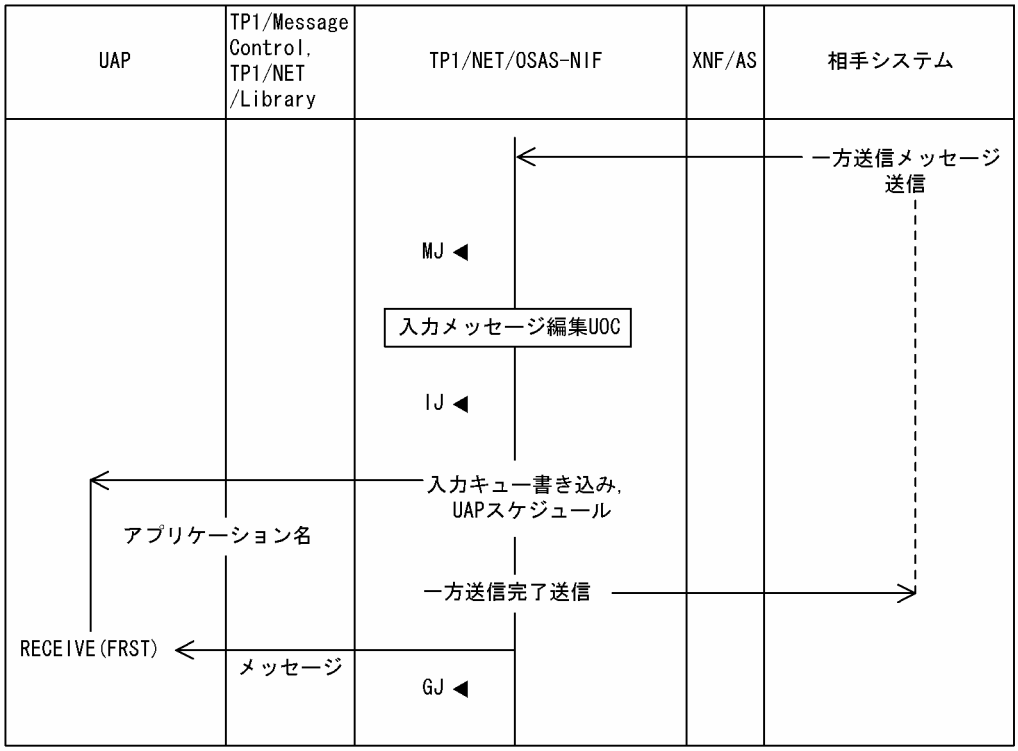
（凡例）　OJ◀：メッセージ出力ジャーナル取得
　　MJ◀：メッセージジャーナル取得
　　AJ◀：メッセージ送信完了ジャーナル取得

付録 D.5　一方送信メッセージの受信

一方送信メッセージを受信するときの処理の流れを，図 D-9，図 D-10 に示します。

(1) 単一セグメントの場合

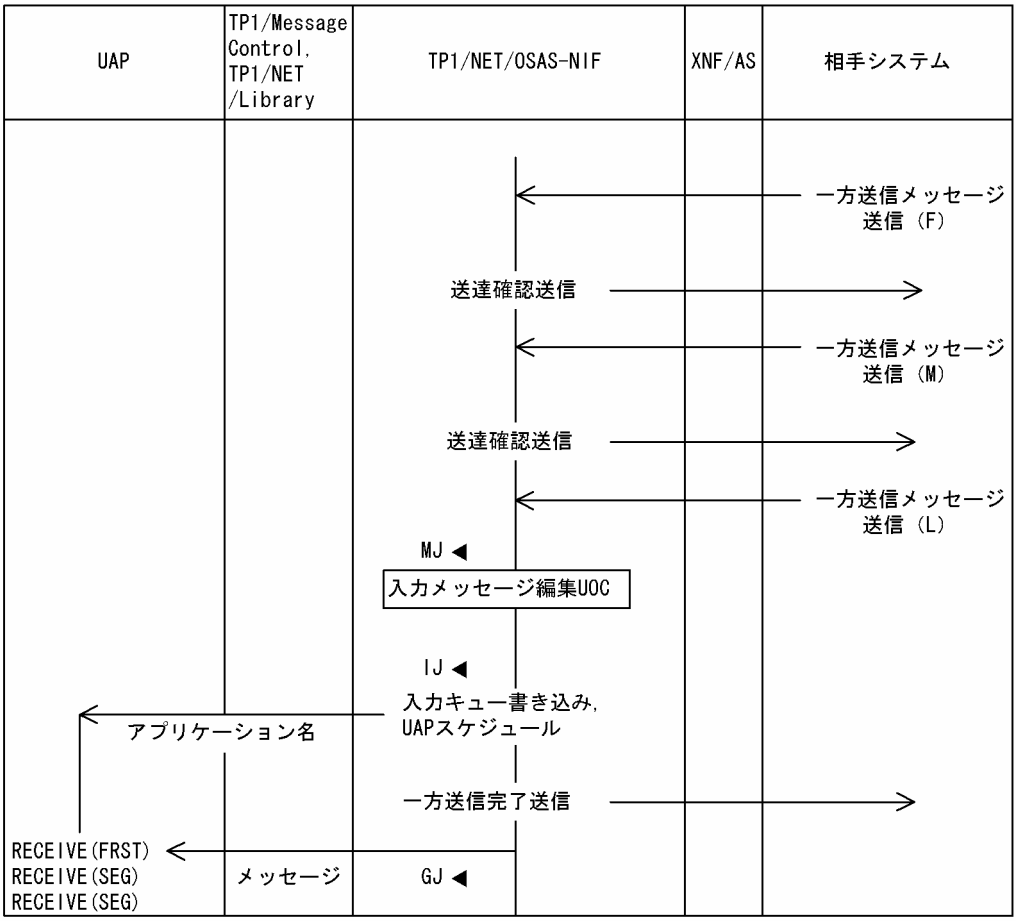
図 D-9 一方送信メッセージ受信の処理の流れ（単一セグメントの場合）



（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

(2)　セグメント組み立て受信の場合

図 D-10　一方送信メッセージ受信の処理の流れ（セグメント組み立て受信の場合）



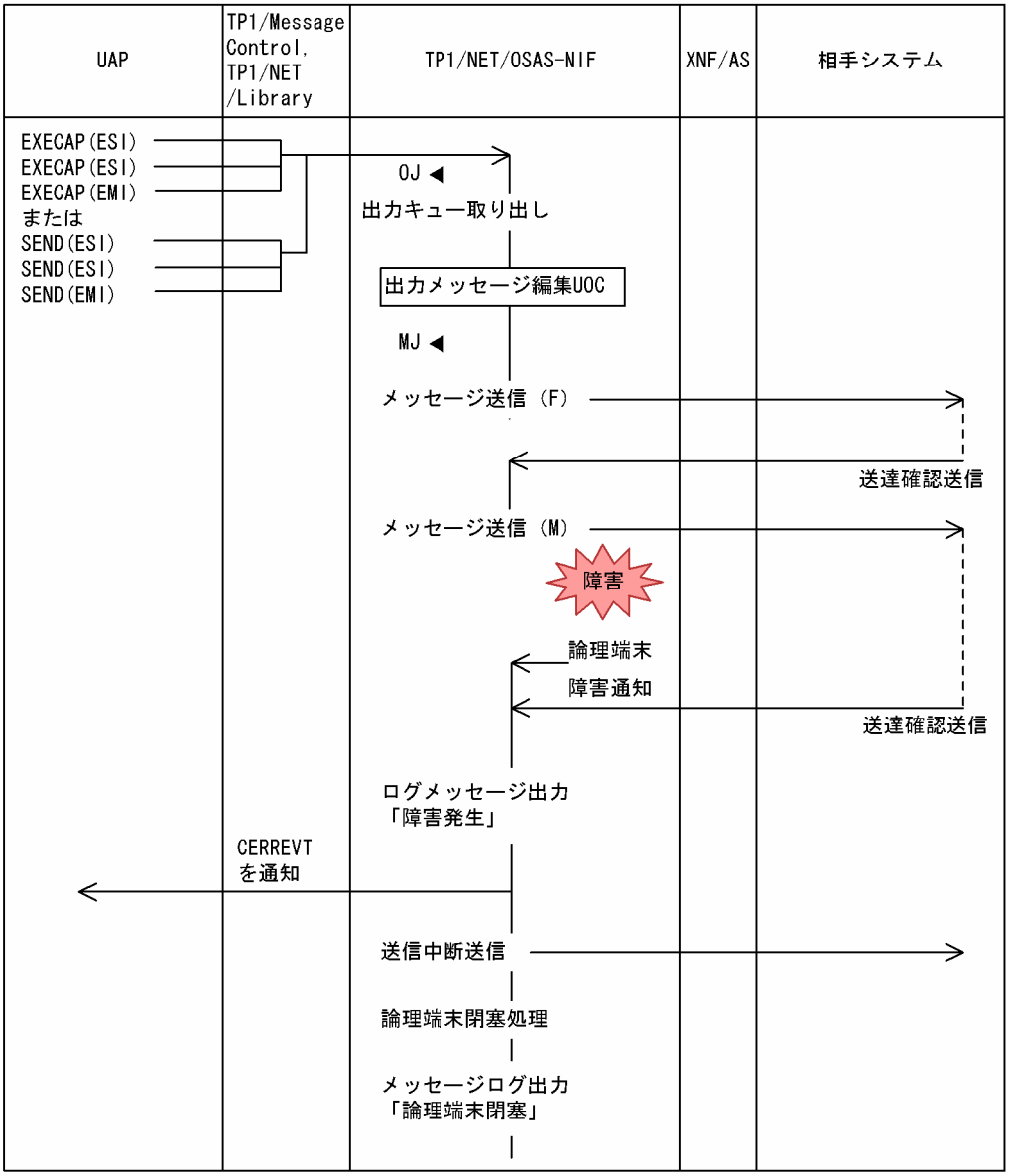
（凡例）　MJ◀：メッセージジャーナル取得
　　IJ◀：メッセージ入力ジャーナル取得
　　GJ◀：メッセージ受信ジャーナル取得

付録 E　障害発生時の処理の流れ

メッセージの送受信中に，障害が発生した場合の処理の流れを，図 E-1～図 E-4 に示します。

付録 E.1　メッセージ送信時の論理端末障害

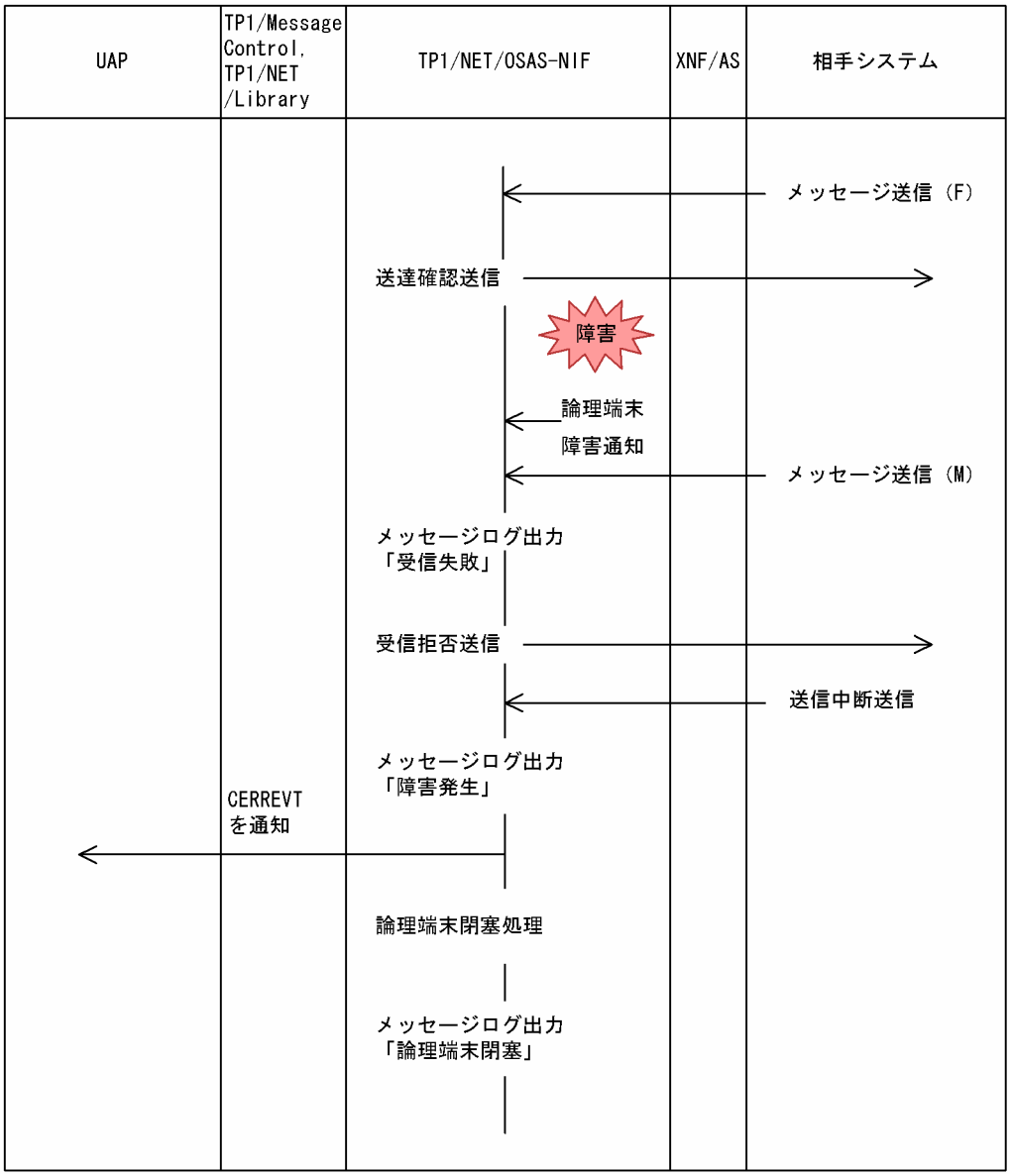
図 E-1　メッセージ送信時の論理端末障害



（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

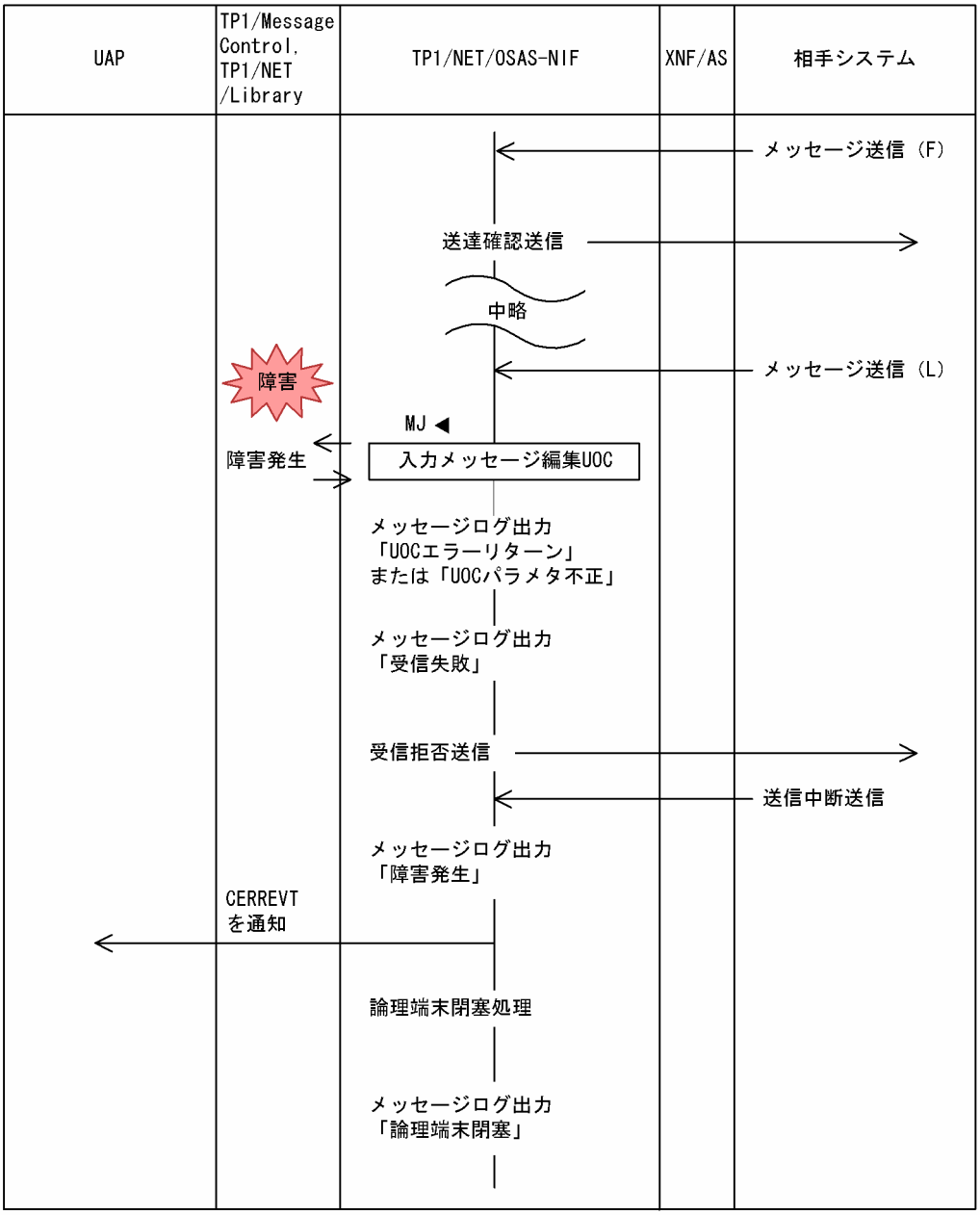
付録 E.2　メッセージ受信時の論理端末障害

図 E-2　メッセージ受信時の論理端末障害



付録 E.3　入力メッセージ編集 UOC エラー

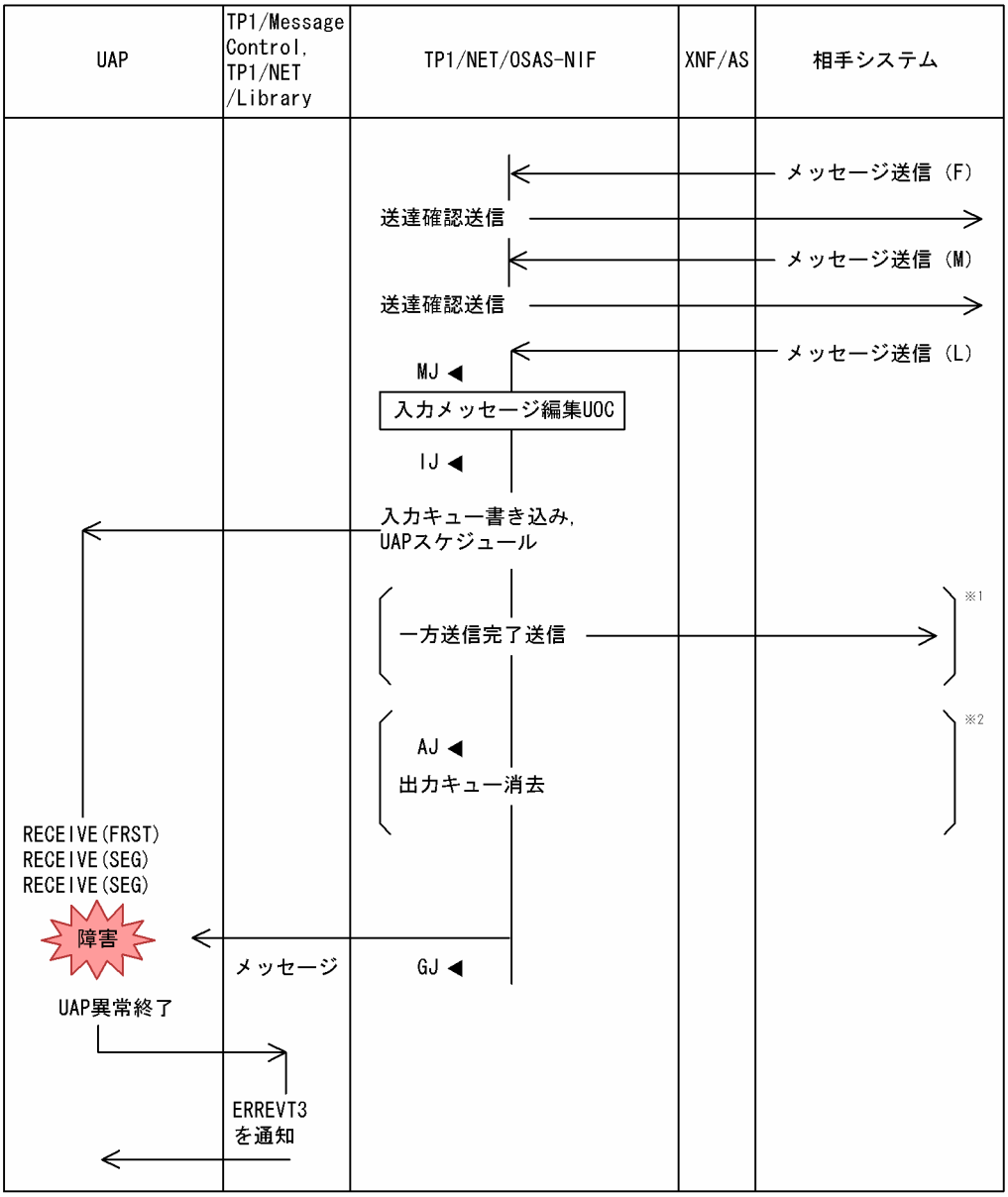
図 E-3　入力メッセージ編集 UOC エラー



（凡例）　MJ ◀：メッセージジャーナル取得

付録 E.4　UAP 異常終了

図 E-4　UAP 異常終了



（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
AJ ◀：メッセージ送信完了ジャーナル取得
GJ ◀：メッセージ受信ジャーナル取得

注※1

一方送信メッセージ受信の場合だけ該当します。

注※2

応答メッセージ受信の場合だけ該当します。

付録 F　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 F.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)　メッセージ送受信時

論理端末単位に相手システムと送受信するメッセージの情報を取得します。MCF 固有情報のダンプ出力情報を，以降の表に示します。

表 F-1　メッセージ送受信時の MCF 固有情報のダンプ出力情報

イベント ID	オフセット				
	0x0000～0x0003	0x0004～0x0007	0x0008～0x000f	0x0010～0x0017	0x0018～0x001f
0xa000	MCF 識別子	スレッド ID	－	入力元論理端末名	プロトコルデータ名称
0xa001			－	出力先論理端末名	

（凡例）

－：情報を取得しません。

表 F-2　プロトコルデータ名称の出力情報

プロトコルデータの種類		プロトコルデータ名称出力情報
NIF-INVOKE	問い合わせ	"INV(REQ)"
	応答	"INV(RPL)"
	一方送信	"INV(SND)"
	イニシャル要求	"INV(IRQ)"
	再送要求	"INV(RRQ)"
	再送回答	"INV(RRS)"
	通知	"INV(IND)"
	通番問い合わせ要求	"INV(NRQ)"
	通番問い合わせ回答	"INV(NRS)"

プロトコルデータの種類	プロトコルデータ名称出力情報
NIF-RESULT	"N-RESULT"
NIF-REJECT	"N-REJECT"
NIF-ERROR	"N-ERROR "

(2) UOC 呼び出し時

TP1/NET/OSAS-NIF 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 F-3 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 F-4 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"

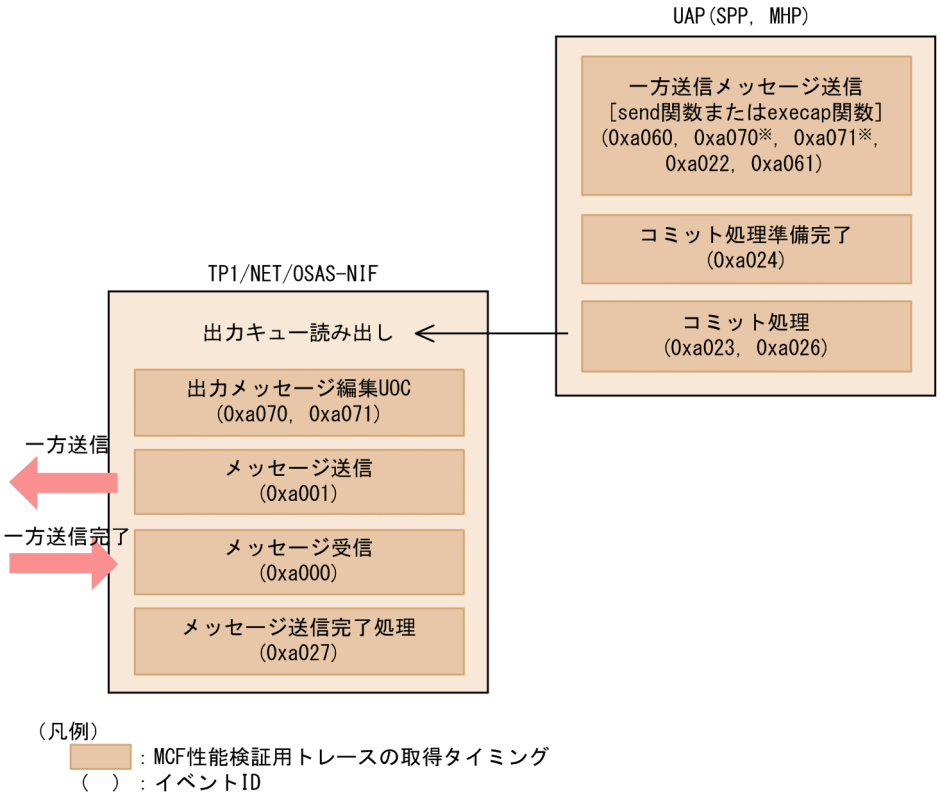
付録 F.2 MCF 性能検証用トレースの取得タイミング

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 F-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



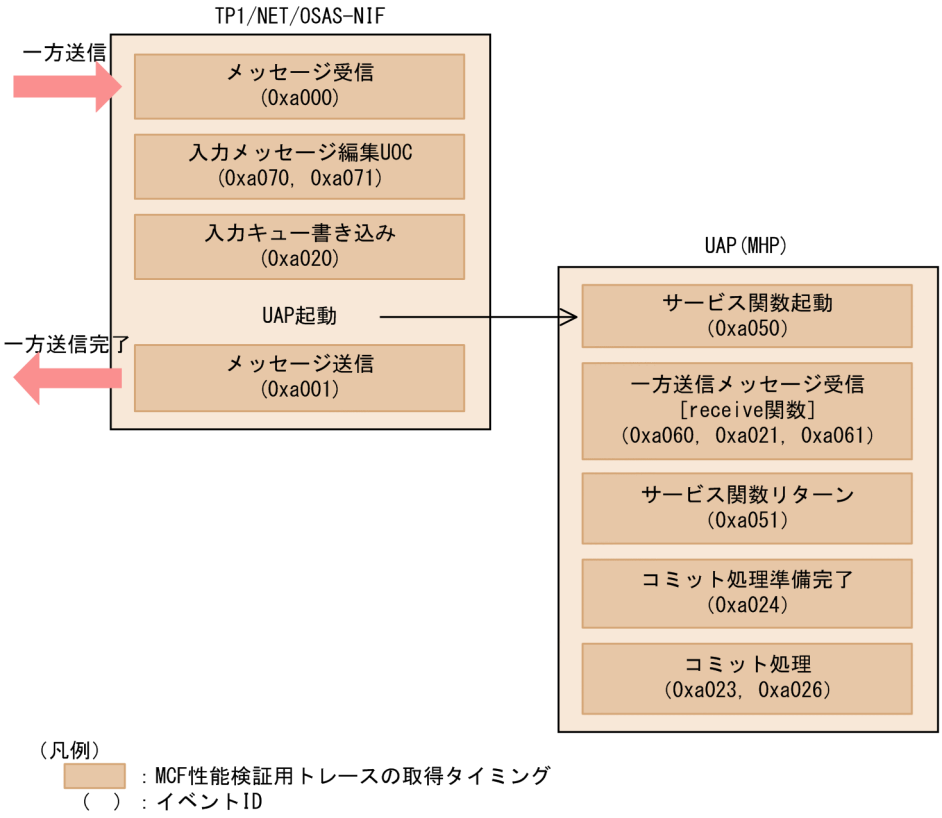
注※

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。

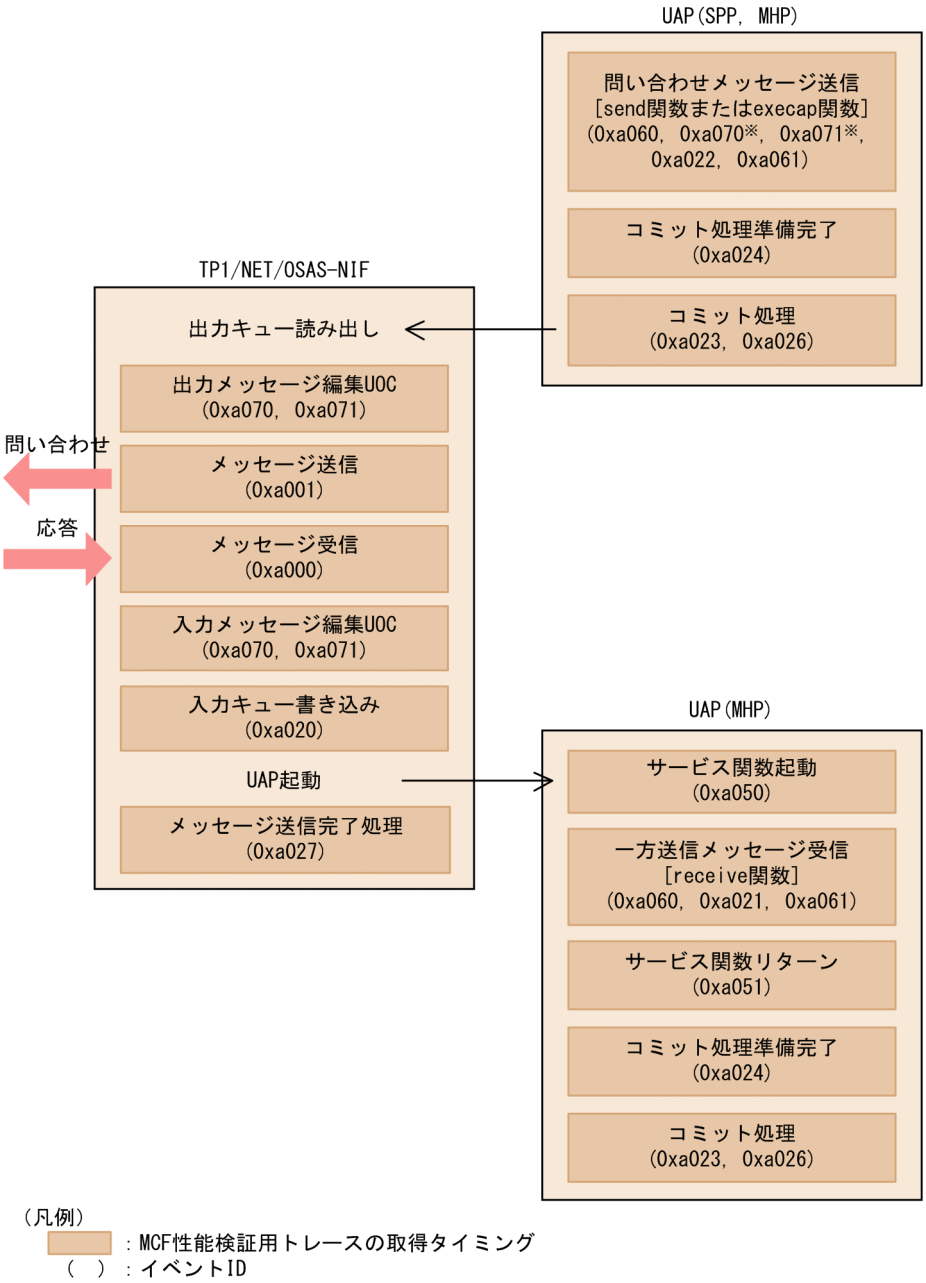
図 F-2　一方送信メッセージ受信時の MCF 性能検証用トレースの取得タイミング



(3)　問い合わせメッセージ送信時

問い合わせ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 F-3　問い合わせ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



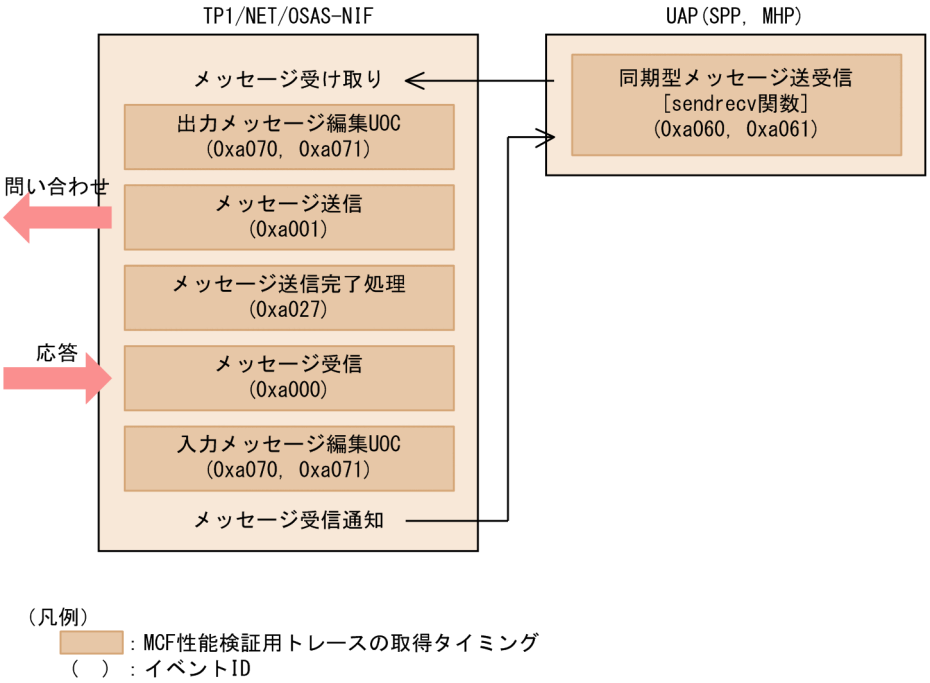
注※

送信メッセージの通番編集 UOC 使用時に該当します。

(4)　同期型の問い合わせメッセージ送受信時

同期型の問い合わせメッセージ送受信時の MCF 性能検証用トレースの取得タイミングについて，次の図に示します。

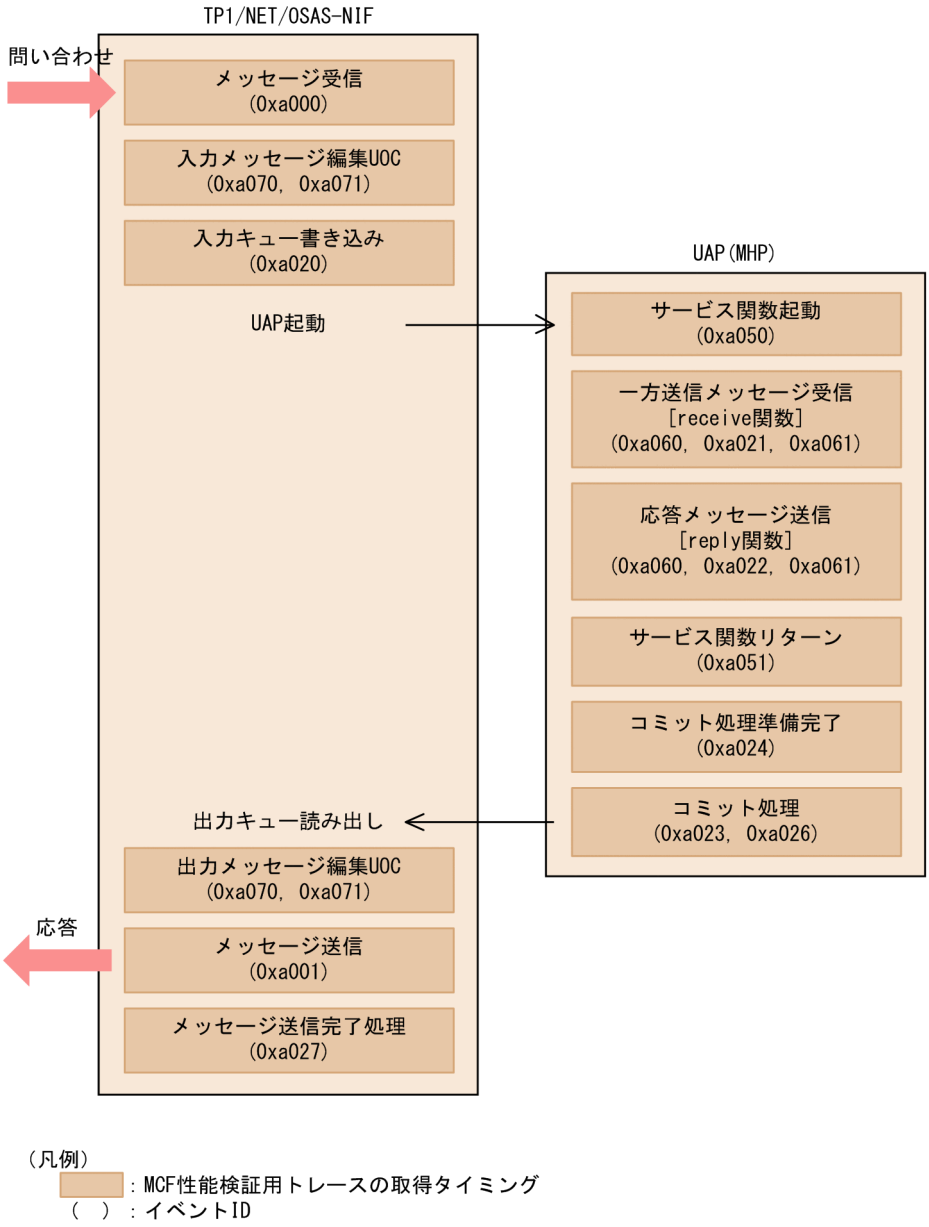
図 F-4　同期型の問い合わせメッセージ送受信時の MCF 性能検証用トレースの取得タイミング



(5)　問い合わせメッセージ受信時

問い合わせメッセージ受信時の MCF 性能検証用トレースの取得タイミングについて，次の図に示します。

図 F-5　問い合わせメッセージ受信時の MCF 性能検証用トレースの取得タイミング



付録 F.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 F-5　MCF 性能検証用トレースの取得量

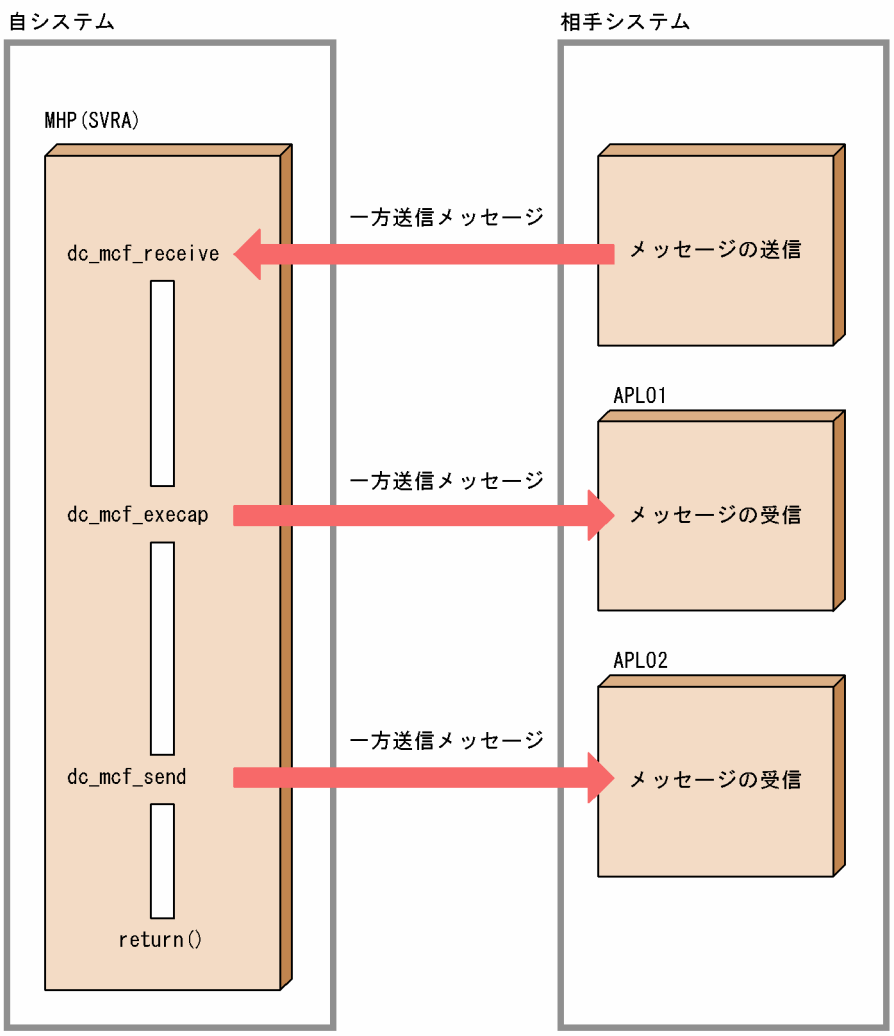
メッセージの送受信形態	トレース取得量 (単位：キロバイト)
一方送信メッセージの送信	1.9
一方送信メッセージの受信	2.3
問い合わせメッセージの送信	3.9

メッセージの送受信形態	トレース取得量 (単位：キロバイト)
同期型の問い合わせメッセージの送信	1.2
問い合わせメッセージの受信	3.0

付録 G　ユーザアプリケーションプログラムの作成例

UAP の作成例の処理の流れを次の図に示します。

図 G-1　UAP の作成例の処理の流れ



付録 G.1　コーディング例

(1)　C 言語

(a)　ANSI C，C++の場合

```
/*
 * MHP SERVICE FUNCTION
 */

#include <string.h>
#include <stdio.h>
#include <sys/types.h>
```

```
#include <dcmcf.h>
#include <dcrpc.h>

void SVRA(void)
{
     DCLONG action ;
     DCLONG commform ;
     DCLONG opcd ;
     DCLONG active ;
     char   recvdata[1024] ;
     DCLONG rdataleng ;
     DCLONG time ;
     DCLONG inbufleng ;
     int    rtn_cod ;
     DCLONG cdataleng ;
     char   termnam[10] ;
     struct dataform { char mcfuse[8] ;
                       char trnname[9] ;
                       char data[23] ;
                     } ;
     static struct dataform execdata ;
     static struct dataform senddata ;
     static char resv01[9] ;
     static char resv02[9] ;
     static char apnam[9] ;

     strncpy(execdata.mcfuse, "        ", 8) ;
     strncpy(execdata.trnname, "         ", 9) ;
     strcpy(execdata.data, "SRVA EXECAP DATA") ;
     strncpy(senddata.mcfuse, "        ", 8) ;
     senddata.trnname[0] = 0xc1 ;  /* A */
     senddata.trnname[1] = 0xd7 ;  /* P */
     senddata.trnname[2] = 0xd3 ;  /* L */
     senddata.trnname[3] = 0xf0 ;  /* 0 */
     senddata.trnname[4] = 0xf2 ;  /* 2 */
     senddata.trnname[5] = 0x40 ;  /*   */
     senddata.trnname[6] = 0x40 ;  /*   */
     senddata.trnname[7] = 0x40 ;  /*   */
     senddata.trnname[8] = 0x40 ;  /*   */
     strcpy(senddata.data, "SRVA SEND   DATA") ;
     strcpy(execdata.mcfuse, "" ) ;
     strcpy(resv02, "") ;
     strcpy(apnam, "APL01") ;

/*
 * MCF-RECEIVE(MESSAGE RECEIVING)
 */

     action = DCMCFFRST ;
     commform = DCNOFLAGS ;
     inbufleng = sizeof(recvdata) ;
     rtn_cod = dc_mcf_receive(action, commform, termnam, resv01,
                              recvdata, &rdataleng, inbufleng, &time) ;
     if(rtn_cod != DCMCFRTN_00000) {
/*
 * MCF-ROLLBACK(ERROR PROCESSING)
 */
     rtn_cod = dc_mcf_rollback(DCMCFNRTN) ;
```

```
    }
/*
 * MCF-EXECAP(APPICATION PROGRAM BOOTING)
 */
    action = DCMCFEMI|DCMCFJUST ;
    commform = DCNOFLAGS ;
    active = 0 ;
    cdataleng = 25 ;
    rtn_cod = dc_mcf_execap(action, commform, resv01, active,
                            apnam, (char*)&execdata, cdataleng) ;
    if(rtn_cod != DCMCFRTN_00000) {
/*
 * MCF-ROLLBACK(ERROR PROCESSING)
 */
    rtn_cod = dc_mcf_rollback(DCMCFNRTN) ;
    }
/*
 * MCF-SEND(MESSAGE SENDING)
 */
    action = DCMCFEMI ;
    commform = DCMCFOUT ;
    strcpy(termnam, "NFLE02");
    opcd = DCNOFLAGS ;
    cdataleng = 25 ;
    rtn_cod = dc_mcf_send(action, commform, termnam, resv01,
                          (char *)&senddata, cdataleng, resv02, opcd) ;
    if(rtn_cod != DCMCFRTN_00000) {
/*
 * MCF ROLLBACK(ERROR PROCESSING)
 */
    rtn_cod = dc_mcf_rollback(DCMCFNRTN) ;
    }
}
```

(b) K&R 版 C の場合

```
/*
 * MHP SERVICE FUNCTION
 */
#include <stdio.h>
#include <sys/types.h>
#include <dcmcf.h>
#include <dcrpc.h>

SVRA()
{
    DCLONG action ;
    DCLONG commform ;
    DCLONG opcd ;
    DCLONG active ;
    char   recvdata[1024] ;
    DCLONG rdataleng ;
    DCLONG time ;
    DCLONG inbufleng ;
    int    rtn_cod ;
    DCLONG cdataleng ;
    char   termnam[10] ;
```

```
     struct dataform { char mcfuse[8] ;
                       char trnname[9] ;
                       char data[23] ;
                     } ;
     static struct dataform execdata ;
     static struct dataform senddata ;
     static char resv01[9] ;
     static char resv02[9] ;
     static char apnam[9] ;

     strncpy(execdata.mcfuse, "        ", 8) ;
     strncpy(execdata.trnname, "         ", 9) ;
     strcpy(execdata.data, "SRVA EXECAP DATA") ;
     strncpy(senddata.mcfuse, "        ", 8) ;
     senddata.trnname[0] = 0xc1 ;  /* A */
     senddata.trnname[1] = 0xd7 ;  /* P */
     senddata.trnname[2] = 0xd3 ;  /* L */
     senddata.trnname[3] = 0xf0 ;  /* 0 */
     senddata.trnname[4] = 0xf2 ;  /* 2 */
     senddata.trnname[5] = 0x40 ;  /*   */
     senddata.trnname[6] = 0x40 ;  /*   */
     senddata.trnname[7] = 0x40 ;  /*   */
     senddata.trnname[8] = 0x40 ;  /*   */
     strcpy(senddata.data, "SRVA SEND   DATA") ;
     strcpy(execdata.mcfuse, "" ) ;


     strcpy(resv02, "") ;
     strcpy(apnam, "APL01") ;

/*
 * MCF-RECEIVE(MESSAGE RECEIVING)
 */
     action = DCMCFFRST ;
     commform = DCNOFLAGS ;
     inbufleng = sizeof(recvdata) ;
     rtn_cod = dc_mcf_receive(action, commform, termnam, resv01, recvdata,
                              &rdataleng, inbufleng, &time) ;
     if(rtn_cod != DCMCFRTN_00000) {
/*
 * MCF-ROLLBACK(ERROR PROCESSING)
 */
     rtn_cod = dc_mcf_rollback(DCMCFNRTN) ;
     }
/*
 * MCF-EXECAP(APPLICATION PROGRAM BOOTING)
 */
     action = DCMCFEMI|DCMCFJUST ;
     commform = DCNOFLAGS ;
     active = 0 ;
     cdataleng = 25 ;
     rtn_cod = dc_mcf_execap(action, commform, resv01, active,
                             apnam, (char*)&execdata, cdataleng) ;
     if(rtn_cod != DCMCFRTN_00000) {
/*
 * MCF-ROLLBACK(ERROR PROCESSING)
 */
     rtn_cod = dc_mcf_rollback(DCMCFNRTN) ;
```

```
    }
/*
 * MCF-SEND(MESSAGE SENDING)
 */
    action = DCMCFEMI ;
    commform = DCMCFOUT ;
    strcpy(termnam, "NFLE02");
    opcd = DCNOFLAGS ;
    cdataleng = 25 ;
    rtn_cod = dc_mcf_send(action, commform, termnam, resv01,
                          (char *)&senddata, cdataleng, resv02, opcd) ;
    if(rtn_cod != DCMCFRTN_00000) {
/*
 * MCF ROLLBACK(ERROR PROCESSING)
 */
    rtn_cod = dc_mcf_rollback(DCMCFNRTN) ;
    }
}
```

(2) COBOL 言語

```
*****************************************************************
*    MHP SERVICE PROGRAM                                        *
*****************************************************************

 IDENTIFICATION DIVISION.

 PROGRAM-ID. SVRA.

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

*****************************************************************
*   WORKING STORAGE                                             *
*****************************************************************

 DATA DIVISION.
 WORKING-STORAGE SECTION.

*****************************************************************
*    DATA AREA FOR MCF-RECEIVE                                  *
*****************************************************************

 01  RECV-PARM1.
   02  RECV-A            PIC X(8) VALUE 'RECEIVE '.
   02  RECV-B            PIC X(5).
   02  FILLER            PIC X(3).
   02  RECV-C            PIC X(4) VALUE 'FRST'.
   02  RECV-D            PIC X(4) VALUE SPACE.
   02  RECV-E            PIC 9(8).
   02  RECV-F            PIC 9(8).
   02  RECV-G            PIC 9(9) COMP VALUE 1024.
   02  RECV-H            PIC X(4) VALUE SPACE.
   02  RECV-I            PIC X(4) VALUE SPACE.
   02  RECV-J            PIC X(4) VALUE SPACE.
   02  RECV-K            PIC X(4) VALUE SPACE.
```

```
   02  RECV-L            PIC X(8) VALUE SPACE.
   02  RECV-M1           PIC X(4) VALUE SPACE.
   02  RECV-M2           PIC X(8) VALUE SPACE.
   02  RECV-M3           PIC X(4) VALUE SPACE.
   02  RECV-M4           PIC 9(9) COMP VALUE ZERO.
   02  RECV-M5           PIC 9(9) COMP VALUE ZERO.
   02  RECV-M6           PIC X(1) VALUE SPACE.
   02  RECV-M7           PIC X(1) VALUE SPACE.
   02  RECV-N            PIC X(14) VALUE LOW-VALUE.
 01  RECV-PARM2.
   02  RECV-O            PIC X(4) VALUE SPACE.
   02  RECV-P            PIC X(8).
   02  RECV-Q            PIC X(8) VALUE SPACE.
   02  RECV-R            PIC X(8) VALUE SPACE.
   02  RECV-T            PIC X(28) VALUE LOW-VALUE.
 01  RECV-PARM3.
   02  RECV-U            PIC 9(9) COMP.
   02  RECV-V            PIC X(8).
   02  RECV-W            PIC X(1024).

****************************************************************
*    DATA AREA FOR MCF-EXECAP                                  *
****************************************************************

 01  EXEC-PARM1.
   02  EXEC-A            PIC X(8) VALUE 'EXECAP  '.
   02  EXEC-B            PIC X(5).
   02  FILLER            PIC X(3).
   02  EXEC-C            PIC X(4) VALUE SPACE.
   02  EXEC-D            PIC X(4) VALUE SPACE.
   02  EXEC-E            PIC 9(8).
   02  EXEC-F            PIC 9(8).
   02  EXEC-G            PIC 9(9) COMP VALUE ZERO.
   02  EXEC-H            PIC X(4) VALUE 'EMI '.
   02  EXEC-I            PIC X(4) VALUE SPACE.
   02  EXEC-J            PIC X(4) VALUE SPACE.
   02  EXEC-K            PIC X(4) VALUE SPACE.
   02  EXEC-L            PIC X(8) VALUE SPACE.
   02  EXEC-M            PIC X(4) VALUE SPACE.
   02  EXEC-N            PIC X(8) VALUE 'APL01   '.
   02  EXEC-O1           PIC X(4) VALUE SPACE.
   02  EXEC-O2           PIC 9(9) COMP VALUE ZERO.
   02  EXEC-O3           PIC 9(9) COMP VALUE ZERO.
   02  EXEC-O4           PIC X(1) VALUE SPACE.
   02  EXEC-O5           PIC X(1) VALUE SPACE.
   02  EXEC-P            PIC X(14) VALUE LOW-VALUE.
 01  EXEC-PARM2.
   02  EXEC-Q            PIC X(4) VALUE SPACE.
   02  EXEC-R            PIC X(8) VALUE SPACE.
   02  EXEC-S            PIC X(8) VALUE SPACE.
   02  EXEC-T            PIC X(6) VALUE SPACE.
   02  EXEC-U            PIC X(2) VALUE SPACE.
   02  EXEC-V            PIC X(28) VALUE LOW-VALUE.
 01  EXEC-PARM3.
   02  EXEC-W            PIC 9(9) COMP VALUE 25.
   02  EXEC-X            PIC X(8).
   02  EXEC-Y1           PIC X(9) VALUE SPACE.
   02  EXEC-Y2           PIC X(16) VALUE 'SVRA EXECAP DATA'.
```

```
*****************************************************************
*    DATA AREA FOR MCF-SEND                                     *
*****************************************************************

 01  SEND-PARM1.
   02  SEND-A            PIC X(8) VALUE 'SEND    '.
   02  SEND-B            PIC X(5).
   02  FILLER            PIC X(3).
   02  SEND-C            PIC X(4) VALUE SPACE.
   02  SEND-D            PIC X(4) VALUE SPACE.
   02  SEND-E            PIC 9(8).
   02  SEND-F            PIC 9(8).
   02  SEND-G            PIC 9(9) COMP VALUE ZERO.
   02  SEND-H            PIC X(4) VALUE 'EMI '.
   02  SEND-I            PIC X(4) VALUE SPACE.
   02  SEND-J            PIC X(4) VALUE SPACE.
   02  SEND-K            PIC X(4) VALUE SPACE.
   02  SEND-L            PIC X(8) VALUE SPACE.
   02  SEND-M1           PIC X(4) VALUE SPACE.
   02  SEND-M2           PIC X(8) VALUE SPACE.
   02  SEND-M3           PIC X(4) VALUE SPACE.
   02  SEND-M4           PIC 9(9) COMP VALUE ZERO.
   02  SEND-M5           PIC 9(9) COMP VALUE ZERO.
   02  SEND-M6           PIC X(1) VALUE SPACE.
   02  SEND-M7           PIC X(1) VALUE SPACE.
   02  SEND-N            PIC X(14) VALUE LOW-VALUE.
 01  SEND-PARM2.
   02  SEND-O            PIC X(4) VALUE 'OUT '.
   02  SEND-P            PIC X(8) VALUE 'NFLE02  '.
   02  SEND-Q            PIC X(8) VALUE SPACE.
   02  SEND-R            PIC X(8) VALUE SPACE.
   02  SEND-T            PIC X(28) VALUE LOW-VALUE.
 01  SEND-PARM3.
   02  SEND-U            PIC 9(9) COMP VALUE 25.
   02  SEND-V            PIC X(8).
   02  SEND-W1           PIC X(9) VALUE X'C1D7D3F0F240404040'.
   02  SEND-W2           PIC X(16) VALUE 'SVRA SEND   DATA'.

*****************************************************************
*    DATA AREA FOR MCF-ROLLBACK                                 *
*****************************************************************

 01  RBK-PARM1.
   02  RBK-A            PIC X(8) VALUE 'ROLLBACK'.
   02  RBK-B            PIC X(5).
   02  FILLER           PIC X(3).
   02  RBK-C            PIC X(4) VALUE 'NRTN'.
   02  RBK-D            PIC X(12) VALUE LOW-VALUE.

 PROCEDURE DIVISION.

*****************************************************************
*    MCF-RECEIVE(RECEIVE OF MESSAGE)                            *
*****************************************************************

   CALL 'CBLDCMCF' USING RECV-PARM1 RECV-PARM2 RECV-PARM3.
   IF RECV-B IS NOT EQUAL TO '00000'
```

```
*****************************************************************
*    MCF-ROLLBACK(ERROR PROCESSING)                             *
*****************************************************************

   CALL 'CBLDCMCF' USING RBK-PARM1.

*****************************************************************
*    MCF-EXECAP(APPLICATION PROGRAM BOOTING)                    *
*****************************************************************

   CALL 'CBLDCMCF' USING EXEC-PARM1 EXEC-PARM2 EXEC-PARM3.
   IF EXEC-B IS NOT EQUAL TO '00000'

*****************************************************************
*    MCF-ROLLBACK(ERROR PROCESSING)                             *
*****************************************************************

   CALL 'CBLDCMCF' USING RBK-PARM1.

*****************************************************************
*    MCF-SEND(SEND OF MESSAGE)                                  *
*****************************************************************

   CALL 'CBLDCMCF' USING SEND-PARM1 SEND-PARM2 SEND-PARM3.
   IF SEND-B IS NOT EQUAL TO '00000'

*****************************************************************
*    MCF-ROLLBACK(ERROR PROCESSING)                             *
*****************************************************************

   CALL 'CBLDCMCF' USING RBK-PARM1.

*****************************************************************
*    END PROCESSING                                             *
*****************************************************************

   EXIT PROGRAM.
```

(3) データ操作言語

```
*****************************************************************
*    MHP SERVICE PROGRAM                                        *
*****************************************************************

 IDENTIFICATION DIVISION.

 PROGRAM-ID. SVRA.

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.

*****************************************************************
*    WORKING STORAGE                                            *
*****************************************************************
```

```
 DATA DIVISION.
 WORKING-STORAGE SECTION.

*****************************************************************
*    AREA FOR MESSAGE RECEIVING                                 *
*****************************************************************

  01  RECV-AREA.
    02  RE-DATALENG       PIC 9(4) COMP.
    02  RE-RSV1           PIC X(2).
    02  RE-DATA           PIC X(1024).

*****************************************************************
*    AREA FOR APPLICATION BOOTING MESSAGE                       *
*****************************************************************

  01  SEND-PRO-AREA.
    02  PRO-DATALENG       PIC 9(4) COMP VALUE 29.
    02  PRO-RSV1           PIC X(2).
    02  PRO-DATA1          PIC X(9) VALUE SPACE.
    02  PRO-DATA2          PIC X(16) VALUE 'SVRA EXECAP DATA'.

*****************************************************************
*    AREA FOR MESSAGE SENDING                                   *
*****************************************************************

  01  SEND-AREA.
    02  SE-DATALENG       PIC 9(4) COMP VALUE 29.
    02  SE-RSV1           PIC X(2).
    02  SE-DATA1          PIC X(9) VALUE X'C1D7D3F0F240404040'.
    02  SE-DATA2          PIC X(16) VALUE 'SVRA SEND   DATA'.

 COMMUNICATION SECTION.

*****************************************************************
*    RECEIVE OF MESSAGE (COMMUNICATION DESCRIPTION TERM)        *
*****************************************************************

 CD RECV-INF
   FOR INPUT
   STATUS KEY IS        RE-STATUS
   SYMBOLIC TERMINAL IS RE-TERMNAM
   MESSAGE DATE IS      RE-DATE
   MESSAGE TIME IS      RE-TIME.

*****************************************************************
* APPLICATION PROGRAM BOOTING (COMMUNICATION DESCRIPTON TERM) *
*****************************************************************

 CD SEND-PRO
   FOR OUTPUT PROGRAM
   STATUS KEY IS        SE-STATUS-PRO
   SYMBOLIC TERMINAL IS SE-TERMNAM-PRO.

*****************************************************************
*    SEND OF MESSAGE (COMMUNICATIO DESCRIPTION TERM)            *
*****************************************************************
```

```
 CD SEND-INF
   FOR OUTPUT
   STATUS KEY IS        SE-STATUS
   SYMBOLIC TERMINAL IS SE-TERMNAM.

 PROCEDURE DIVISION.

****************************************************************
*    RECEIVE OF MESSAGE (COMMUNICATION DESCRIPTION SENTENCE)  *
****************************************************************

   MOVE 1028 TO RE-DATALENG.
   RECEIVE RECV-INF
           FIRST SEGMENT
           INTO RECV-AREA.
   IF RE-STATUS IS NOT EQUAL '00000'

****************************************************************
*    PARTIAL RECOVERY                                          *
****************************************************************

   ROLLBACK WITH STOPPING.

****************************************************************
*    APPLICATION PROGRAM BOOTING (COMMUNICATION SENTENCE)     *
****************************************************************

   MOVE 'APL01   ' TO SE-TERMNAM-PRO.
   SEND SEND-PRO
        FROM SEND-PRO-AREA
        WITH EMI.
   IF SE-STATUS-PRO IS NOT EQUAL '00000'

****************************************************************
*    PARTIAL RECOVERY                                          *
****************************************************************

   ROLLBACK WITH STOPPING.

****************************************************************
*    SEND OF MESSAGE (COMMUNICATION SENTENCE)                  *
****************************************************************

   MOVE 'NFLE02  ' TO SE-TERMNAM.
   SEND SEND-INF
        FROM SEND-AREA
        WITH EMI.
   IF SE-STATUS IS NOT EQUAL '00000'

****************************************************************
*    PARTIAL RECOVERY                                          *
****************************************************************

   ROLLBACK WITH STOPPING.

****************************************************************
*    END PROCESSING                                            *
****************************************************************
```

```
    EXIT PROGRAM.
```

付録 G.2　提供するサンプルコーディング

ここでは，TP1/NET/OSAS-NIF が提供するサンプルコーディングの格納場所について言語ごとに示します。

(1)　C 言語

- ANSI C，C++の場合
  /BeTRAN/examples/mcf/OSASNIF/aplib/ansi/ap.c
- K&R 版 C の場合
  /BeTRAN/examples/mcf/OSASNIF/aplib/c/ap.c

(2)　COBOL 言語

/BeTRAN/examples/mcf/OSASNIF/aplib/cobol/ap.cbl

(3)　データ操作言語

/BeTRAN/examples/mcf/OSASNIF/aplib/dml/ap.cbl

付録 H　UOC のコーディング例

TP1/NET/OSAS-NIF の入力メッセージ編集 UOC のコーディング例（K&R 版 C）を次に示します。このコーディング例は，/BeTRAN/examples/mcf/OSASNIF/cmlib/c/uoc.c のファイルで提供しています。

```
#include <stdio.h>
#include <dcmcf.h>
#include <dcmnom.h>
#include <dcmcfuoc.h>

#define ERR_PRT_KIND    -19001L

DCLONG dc_mcf_stduoc_msgin(param)

/* ADDRESS OF EDITING AREA FOR INPUT MESSAGE EDITING UOC       */
dcmcf_uoc_min_n *param ;

{
/*******************************/
/* DECLARATION OF ARGUMENT     */
/*******************************/

dcmnom_uoc *ifa_ptr ; /* POINTER OF PROTOCOL INDIVIDUAL INTERFACE */
DCLONG n ;
 if(param->pro_kind != DCMCF_UOC_PRO_NF){ /* CHECK OF PROTOCOL KIND */
     param->rtn_detail=ERR_PRT_KIND;
     return(DCMCF_UOC_MSG_NG);            /* MESSAGE EDITING ERROR */
  }
 ifa_ptr = (dcmnom_uoc *)param->pro_indv_ifa ;
        /* POINTER AQUISITION OF PROTOCOL INDIVIDUAL AREA BY CASTING */
 if(ifa_ptr->appname[0] != 0){
   strcpy(param->aplname,ifa_ptr->appname);
  }
 else{
       param->aplname[8] = 0 ;
       for(n = 0 ; n < 9 ; n++){
/* NULL TERMINATING IF SPACE, AND MAKING NORMAL DECISION OF AP NAME */
         if(param->buflist_adr->buf_array[0].buf_adr[n] == ' '){
             param->aplname[n] = 0 ;
             break;
         }
         param->aplname[n] =
                     param->buflist_adr->buf_array[0].buf_adr[n] ;
     }
 }
 param->buflist_adr->used_buf_num = param->buflist_adr->buf_num ;
 return(DCMCF_UOC_MSG_OK_RCV);
 }
```

付録 I　理由コード一覧

付録 I.1　ERREVT2 の理由コード

ERREVT2 通知時の理由コードを次の表に示します。

表 I-1　ERREVT2 の理由コード

C 言語の理由コード（16 進数字）	COBOL 言語の理由コード（外部 10 進数字）	ERREVT2 の通知理由
DCMCF_NO_SERV (0010)	0010	アプリケーション名に相当する MHP のサービスがありません。
DCMCF_SCD_ERR (0020)	0020	ユーザサーバ未起動などによって，MHP の起動に失敗しました。
DCMCF_QUE_BUF_ERR (0030)	0030	動的共用メモリが不足しました。
DCMCF_QUE_FIL_OVER (0031)	0031	キューファイルが満杯になりました。
DCMCF_QUE_LIMIT_OVER (0032)	0032	入力メッセージ最大格納数を超過しました。
DCMCF_QUE_IO_ERR (0033)	0033	入力キューに障害が発生しました。
DCMCF_AP_CLOSE (0040)	0040	MHP のアプリケーションが閉塞しています。
DCMCF_AP_SECURE (0041)	0041	MHP のアプリケーションがセキュア状態です。
DCMCF_SERV_CLOSE (0042)	0042	• MHP のサービスまたはサービスグループが閉塞しています。 • スケジュール閉塞されているサービスグループの入力キューに未処理受信メッセージが残った状態で，OpenTP1 を正常終了または計画停止 A で終了しました。
DCMCF_SERV_SECURE (0043)	0043	MHP のサービスグループがセキュア状態です。
DCMCF_ABNORMAL_END (0050)	0050	• MHP で呼び出す receive 関数にセグメントを渡す前に，MHP が異常終了しました。 • DBMS の障害などによって，トランザクションの開始に失敗しました。

付録 I.2　CERREVT の理由コード

CERREVT の理由コードを次の表に示します。

表 I-2　CERREVT の理由コード

理由コード 1 (16 進数字)	理由コード 2 (16 進数字)	発生条件	発生個所
DCMNOM_RSN1_MCF (00000001)	DCMNOM_RSN2_ITQ (00000001)	メッセージ入力障害	論理端末
	DCMNOM_RSN2_OTGET (00000002)	送信メッセージ取得障害	論理端末
	DCMNOM_RSN2_OTCMP (00000003)	メッセージ送信完了処理障害	論理端末
	DCMNOM_RSN2_DCTLE (00000004)	運用コマンドまたは API による論理端末の閉塞	論理端末
	DCMNOM_RSN2_NOBUF (00000005)	バッファの取得失敗	論理端末
	DCMNOM_RSN2_DCTCN (00000006)	運用コマンドまたは API によるコネクション強制解放	コネクション
	DCMNOM_RSN2_LEERR (00000007)	論理端末障害によるコネクション強制解放	コネクション
	DCMNOM_RSN2_SYCER (00000008)	UAP への同期リターン失敗	論理端末
DCMNOM_RSN1_ERR (00000002)	DCMNOM_RSN2_CNERR (00000000)	下位層障害	コネクション，論理端末
	DCMNOM_RSN2_TIMEOUT (00000001)	タイムアウト発生	コネクション，論理端末
	DCMNOM_RSN2_RCV_RJT (00000002)	受信拒否受信による論理端末閉塞	論理端末
	DCMNOM_RSN2_RCV_STOP (00000003)	送信中断受信による論理端末閉塞	論理端末
	DCMNOM_RSN2_UERR (00000004)	UAP 異常受信による論理端末閉塞	論理端末
DCMNOM_RSN1_UOC (00000003)	詳細リターンコード (UOC からのリターンコード)	ユーザ（メッセージ編集 UOC）検出障害	コネクション，論理端末
DCMNOM_RSN1_UAP (00000004)	DCMNOM_RSN2_SND_ERR (00000000)	エラー送信による論理端末閉塞	論理端末

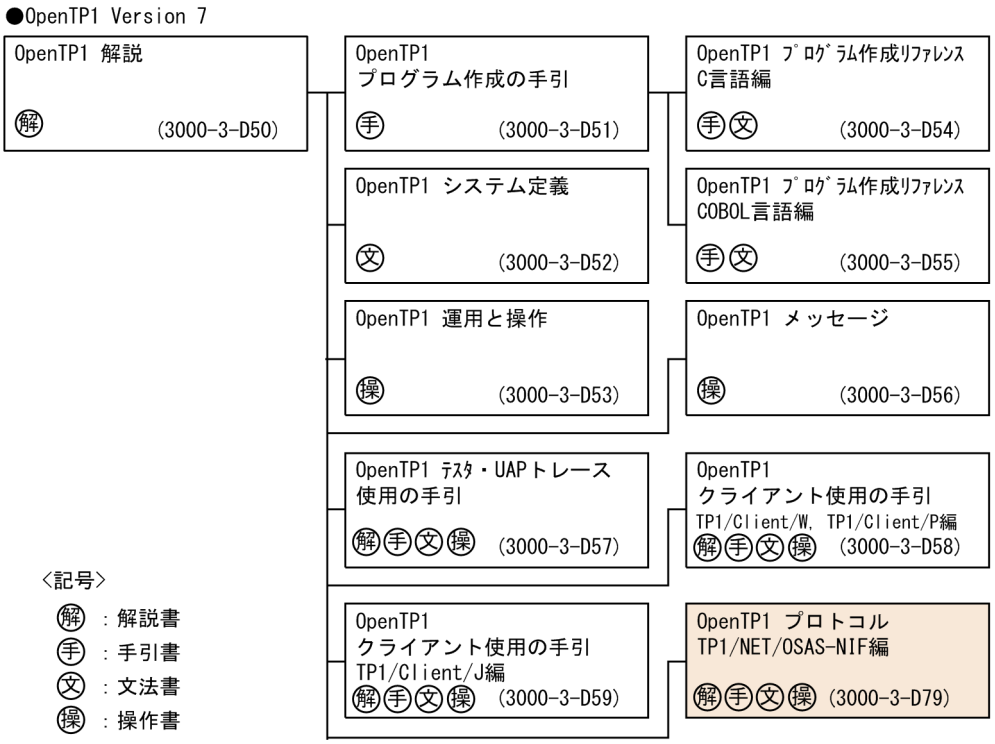
理由コード 1 (16 進数字)	理由コード 2 (16 進数字)	発生条件	発生個所
DCMNOM_RSN1_UAP (00000004)	DCMNOM_RSN2_MSGERR (00000001)	UAP 送信メッセージ編集失敗による論理端末閉塞	論理端末
	DCMNOM_RSN2_UAPAB (00000002)	UAP 異常またはアプリケーション型不正による論理端末閉塞	論理端末
	DCMNOM_RSN2_VRERR (00000003)	前提バージョン不一致による論理端末閉塞	論理端末
	DCMNOM_RSN2_RTIMEOUT (00000004)	応答監視タイムアウトによる論理端末閉塞	論理端末
上記以外		上記以外の障害	不定

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

このマニュアルの関連マニュアルを次に示します。必要に応じてお読みください。

(1)　OpenTP1 関連



(2)　通信管理関連

- 通信管理 XNF/AS 解説・運用編（3000-3-B61）
- 通信管理 XNF/AS 構成定義編（3000-3-B62）
- 通信管理 XNF/AS NSAP アドレス概説編（3000-3-B43）

(3)　相手システム（VOS3）関連

- TMS-4V/SP OSAS/AP 間通信サービス-NF OSAS/NF/4VSP（6190-6-127）
- OSI アプリケーション共通機能/AP 間通信サービス/DCCM3 OSAS/UA2/DCCM3（6190-6-371）
- XNF TCP/IP 接続機能 XNF/TCP 定義とコマンド（6190-3-584）

付録 J.2　このマニュアルでの表記

(1)　製品名

このマニュアルでの表記と正式名称を次の表に示します。

表記		製品名
TP1/Message Control		uCosminexus TP1/Message Control
TP1/NET/Library		uCosminexus TP1/NET/Library
TP1/NET/OSAS-NIF		uCosminexus TP1/NET/OSAS-NIF
UNIX	AIX	AIX V6.1
		AIX V7.1
		AIX V7.2

(2)　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）	'~'（チルダ）

付録 J.3　英略語

このマニュアルで使用する英略語を次に示します。

英略語	英字での表記
ACSE	Association Control Service Element
AT	Agent
ATM	Agent Manager
HNA	Hitachi Network Architecture
LE	Logical Entity
MCF	Message Control Facility

英略語	英字での表記
MHP	Message Handling Program
NIF/OSI	Network Interface Feature/Open Systems Interconnection
OS	Operating System
RPC	Remote Procedure Call
SPP	Service Providing Program
TMS-4V/SP	Transaction Management System-4V/System Product
UAP	User Application Program

付録 J.4　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，メッセージの種類で異なります。非同期型の問い合わせメッセージの場合は，応答メッセージの最終セグメントを受信した直後です。一方送信メッセージの場合は，一方送信メッセージの最終セグメントを送信したあと，相手システムから一方送信完了を受信した直後です。応答メッセージの場合は，応答メッセージの最終セグメントを送信した直後です。

GJ（メッセージ受信ジャーナル）

MCF が取得する統計用の履歴情報の一つです。入力メッセージサイズ，入力論理端末名などで構成されます。

GJ の取得タイミングは，非同期受信関数を発行して，MHP が受信メッセージを入力キューから取り出した直後です。

IJ（メッセージ入力ジャーナル）

MCF が取得する統計用の履歴情報の一つです。メッセージ入力通番，入力論理端末名，入力メッセージ種別，および入力メッセージなどで構成されます。

IJ の取得タイミングは，相手システムから受信したメッセージを入力キューに格納する直前です。

MJ（メッセージジャーナル）

MCF が取得する統計用の履歴情報の一つです。入力論理端末名または，出力論理端末名，メッセージジャーナル種別，入力または出力メッセージ（入力メッセージ編集前のデータ，または出力メッセージ編集後のデータ）などで構成されます。

MJ の取得タイミングは，入力メッセージの場合，入力メッセージ編集 UOC を呼び出す直前です。また，出力メッセージの場合，出力メッセージ編集 UOC を呼び出した直後です。

OJ（メッセージ出力ジャーナル）

MCF が取得する統計用の履歴情報の一つです。メッセージ出力通番，出力論理端末名，出力メッセージ種別，出力メッセージなどで構成されます。

OJ の取得タイミングは，非同期送信関数を発行して，UAP が送信メッセージを出力キューに格納した直後です。

（ア行）

エージェント（AT）

NIF/OSI プロトコルで規定されているサービス要素です。

エージェントマネジャ（ATM）

コネクションの管理およびエージェントの初期化や管理をするエージェントです。

（カ行）

コネクション

通信相手システムとの論理的な通信路です。TP1/NET/OSAS-NIF の通信管理側の通信接点であり，TP1/NET/OSAS-NIF と通信管理はコネクション単位に送受信をします。

（ラ行）

論理端末（LE）

TP1/NET/OSAS-NIF の UAP 側の通信接点であり，TP1/NET/OSAS-NIF と UAP は論理端末単位に送受信をします。

索引

Q

R

S

T

U

く

こ

さ

し

せ

そ

み

め

ゆ

よ

り

れ

ろ