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/Secondary Logical Unit - TypeP2 編

解説・手引・文法・操作書

3000-3-D80-10

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-F1M64-3241B uCosminexus TP1/NET/Secondary Logical Unit - TypeP2 07-50

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

■ 輸出時の注意

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

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

■ 発行

2017 年 7 月 3000-3-D80-10

■ 著作権

変更内容

変更内容（3000-3-D80-10） uCosminexus TP1/Message Control 07-51，uCosminexus TP1/NET/Library 07-51，uCosminexus TP1/NET/Secondary Logical Unit - TypeP2 07-50

追加・変更内容	変更個所
通信管理プログラムとホスト間の通信路の状態によって，ホストに対してコネクション確立要求を送信したり，ホストからコネクションの確立要求を待ち合わせたりする機能（INIT-SELF 送信制御）を追加した。	2.1.1，2.1.5， 6. システム定義 mcftalccn（コネクション定義の開始）
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。	2.1.1， 3. C 言語のライブラリ関数 C 言語のライブラリ関数の一覧， dc_mcf_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 ')
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。	3. C 言語のライブラリ関数 C 言語のライブラリ関数の一覧， dc_mcf_tactle， dc_mcf_tdctle， dc_mcf_tlsle， 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 ')
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

追加・変更内容	変更個所
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 ')
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
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
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。 • SLU - TypeP2 07-50 • SLU - TypeP2 07-00	付録 A
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，SLU - TypeP2 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 SLUTYPE-P プロトコルを使用するシステム管理者，システム設計者，およびプログラマを対象としています。また，オンラインや OpenTP1 システムの基礎的な知識を持っていて，次のマニュアルを理解されていることを前提としています。

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

■ マニュアルの構成

このマニュアルは，次に示す章と付録から構成されています。

第 1 章　概要

SLU - TypeP2 を使用した AP 間通信の概要について説明しています。

第 2 章　機能

SLU - TypeP2 のコネクション，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

SLU - TypeP2 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

SLU - TypeP2 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

SLU - TypeP2 に関連するユーザオウンコーディングインタフェース，および MCF イベントインタフェースについて説明しています。

第６章　システム定義

SLUTYPE-P 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 SLU - TypeP2 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

SLU - TypeP2 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

SLU - TypeP2 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

SLU - TypeP2 運用中に発生する障害と，SLU - TypeP2 の対応処理，およびメッセージの処理について説明しています。

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

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

SLU - TypeP2 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　理由コードおよびセンスコード一覧

障害通知イベントが発生した場合の理由コード，および-RSP 送信時のセンスコードについて説明しています。

付録 I　このマニュアルの参考情報

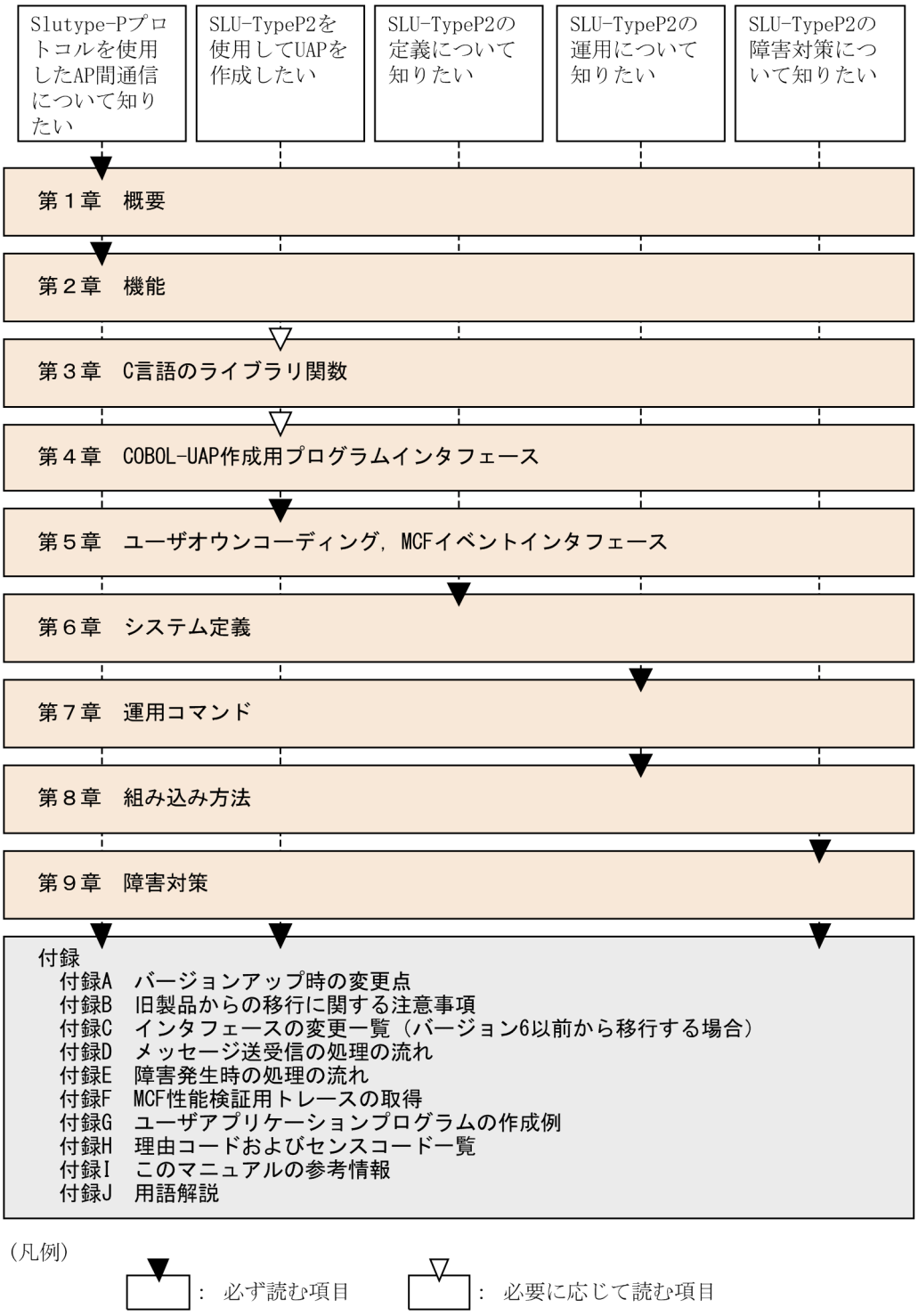
関連マニュアル，このマニュアルで使用している略語の意味などを説明しています。

付録 J　用語解説

SLU - TypeP2 で使用する用語について説明しています。

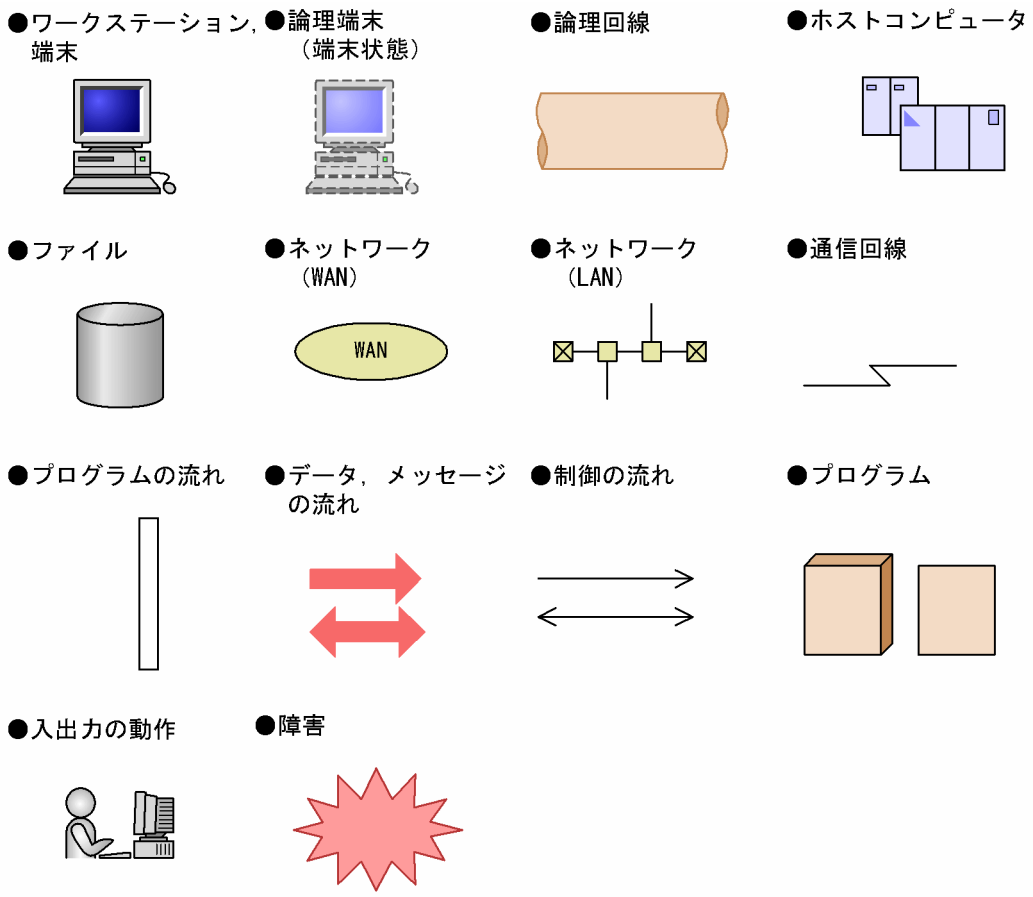
■ 読書手順

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）	この記号で区切られた項目は選択できることを示します。 (例) -t reply | request 　-t オプションに reply または request を指定できることを示します。 ただし，C 言語のインタフェースの説明でこの記号を使用した場合は，C 言語の文法規則に従います。
{　}	この記号で囲まれている複数の項目のうちから一つを選択できることを示します。

文法記述記号	意味
{　　}	（例）{DCMCFESI ｜ DCMCFEMI} 　DCMCFESI と DCMCFEMI のうち，どちらかを指定できることを示します。
＿ （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，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）
＜識別子＞	先頭がアルファベットの英数字列

構文要素記号	意味
＜記号名称＞	先頭が英字記号の英数字記号列
＜文字列＞	任意の文字の配列
＜パス名＞	記号名称，／，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

SLU - TypeP2 は，OpenTP1 システムを構成するプログラムの一つです。ホストコンピュータ，端末などを SLUTYPE-P プロトコルによって論理的に接続し，メッセージを送受信します。

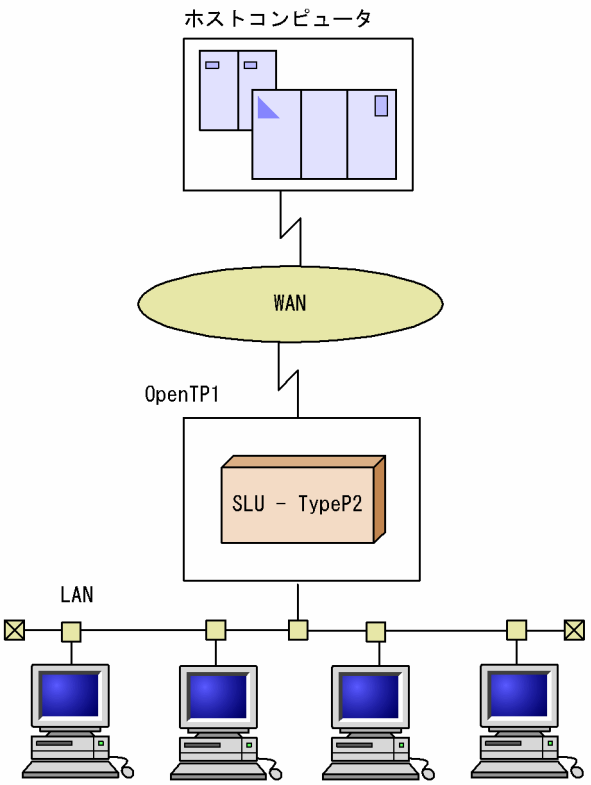
この章では，SLU - TypeP2 を使用したシステム間の通信（AP 間通信）の概要について説明します。

1.1　AP 間通信の概要

AP 間通信とは，異なるシステムにあるアプリケーションプログラム間でのメッセージ送受信をいいます。SLU - TypeP2 は，SLUTYPE-P プロトコルの 2 次局側として，1 次局側のホストシステムと AP 間通信をするプログラムです。SLU - TypeP2 を使用した AP 間通信では，相手システムで発生したトランザクションを自システムで処理したり，その結果を送信したりできます。

SLU - TypeP2 を使用したネットワーク構成の例を次の図に示します。

図 1-1　SLU - TypeP2 を使用したネットワーク構成の例



1.2　AP 間通信の形態

1.2.1　通信形態

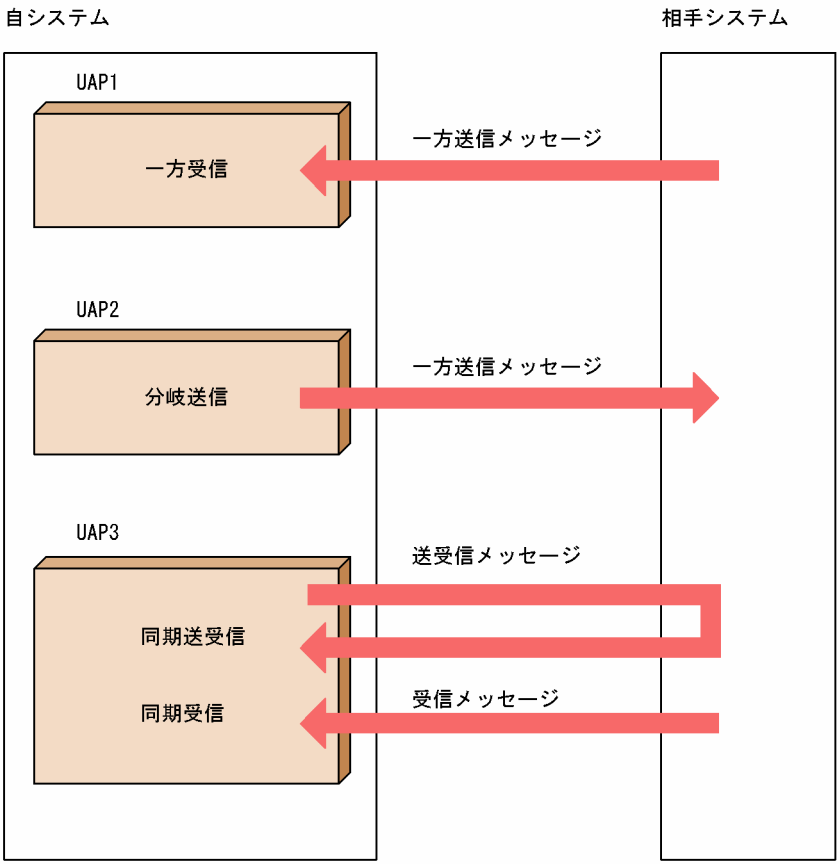
AP 間通信を使用すると，自システムで発生したトランザクションを相手システムで処理したり，その結果を受信したりできます。また，相手システムで発生したトランザクションを自システムで処理したり，その結果を送信したりできます。

SLU - TypeP2 を使用した AP 間通信の形態には，次の四つがあります。

- 一方受信
- 分岐送信
- 同期送受信
- 同期受信

SLU - TypeP2 を使用した AP 間通信の例を次の図に示します。

図 1-2　SLU - TypeP2 を使用した AP 間通信の例



(1) 一方受信

相手システムから AP 間通信の開始要求を受信する形態です。開始要求を受信すると，SLU - TypeP2 はメッセージ送受信のためのアプリケーションを起動します。

(2) 分岐送信

相手システムに対して，メッセージを送信する形態です。

(3) 同期送受信

自システムからメッセージを送信し，応答を受信する形態です。このとき，自システムからメッセージを送信後，相手システムからの応答を待ちます。応答を受信したときに，同期送受信を要求した UAP に制御を返します。

(4) 同期受信

同期送受信でメッセージが複数に分割されている場合，2 番目以降を受信する形態です。メッセージの先頭部分は同期送受信で受け取ります。メッセージの最終部分の受信が完了すると，同期受信を要求した UAP に制御を返します。

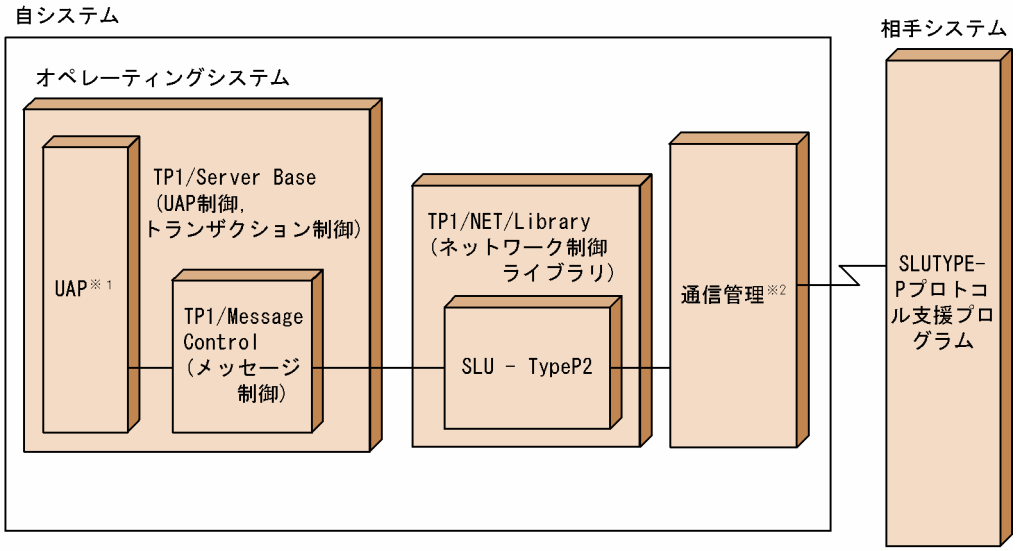
この形態は，同期送受信のメッセージが単一の場合は適用されません。

1.3　ソフトウェア構成の例

SLU - TypeP2 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。

SLU - TypeP2 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。また，SLU - TypeP2 が AP 間通信で使用するプロトコルおよび通信相手プログラムを表 1-1 に示します。

図 1-3　SLU - TypeP2 を組み込んだソフトウェア構成の例



注※1

SLU - TypeP2 で扱う UAP は，MHP および SPP です。UAP については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

注※2

XNF/AS を使用できます。

表 1-1　SLU - TypeP2 に適用する AP 間通信のプロトコル

プロトコル	通信相手
SLUTYPE-P プロトコル	IMS IBM ホストシステム

2 機能

一般に，AP 間通信をするときには，自システムと相手システムとの間であらかじめ通信上の規約（プロトコル）を決める必要があります。SLU - TypeP2 は，二つのシステムの間にコネクションという論理的通信路を設定し，メッセージを送受信します。

この章では，SLU - TypeP2 が設定するコネクションの確立方法，メッセージの種類と送受信の方法について説明します。

2.1　AP 間通信の仕組み

2.1.1　コネクションの確立と解放

SLU - TypeP2 では，相手システムとの間に論理的通信路（**コネクション**）を確立してメッセージを送受信します。コネクションは，SLUTYPE-P プロトコルのセションに対応します。

(1)　コネクションの確立

コネクションの確立には，次の五つがあります。

- ホストからの要求による確立
- オンライン開始・再開始時の自動確立
- 運用コマンド入力による手動確立
- API 発行による手動確立
- INIT-SELF 送信制御機能による確立

INIT-SELF 送信制御機能の詳細については，「2.1.5　INIT-SELF 送信制御」を参照してください。

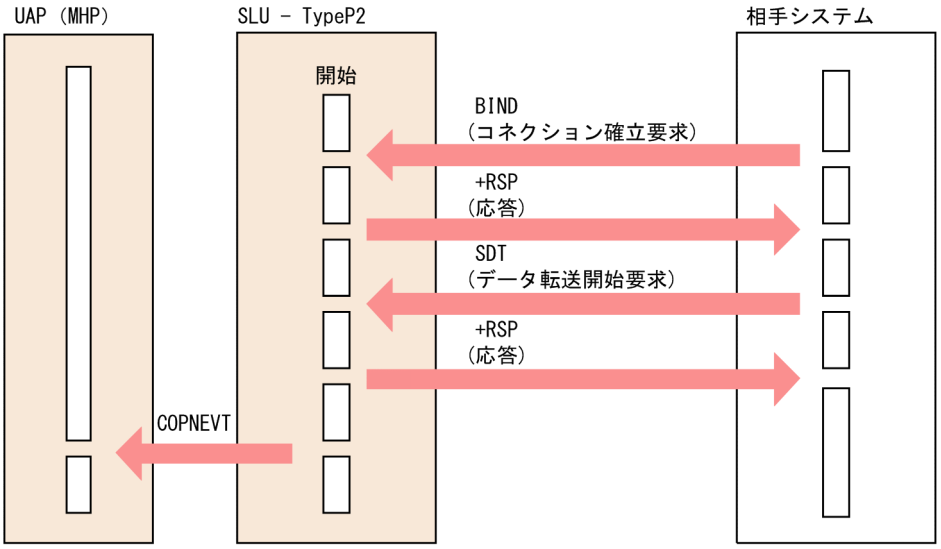
(a)　ホストからの要求による確立

ホスト（相手システム）からコネクションの確立の要求を受ける方法です。

起動種別がホスト起動の場合（コネクション定義（mcftalccn -k）に host を指定），SLU - TypeP2 はホストからの確立要求（BIND）を待ちます。コネクション確立後，すべての論理端末を閉塞解除し，UAP にコネクションの確立（COPNEVT）を通知します。

ホストからの要求による確立を次の図に示します。

図 2-1　ホストからの要求による確立



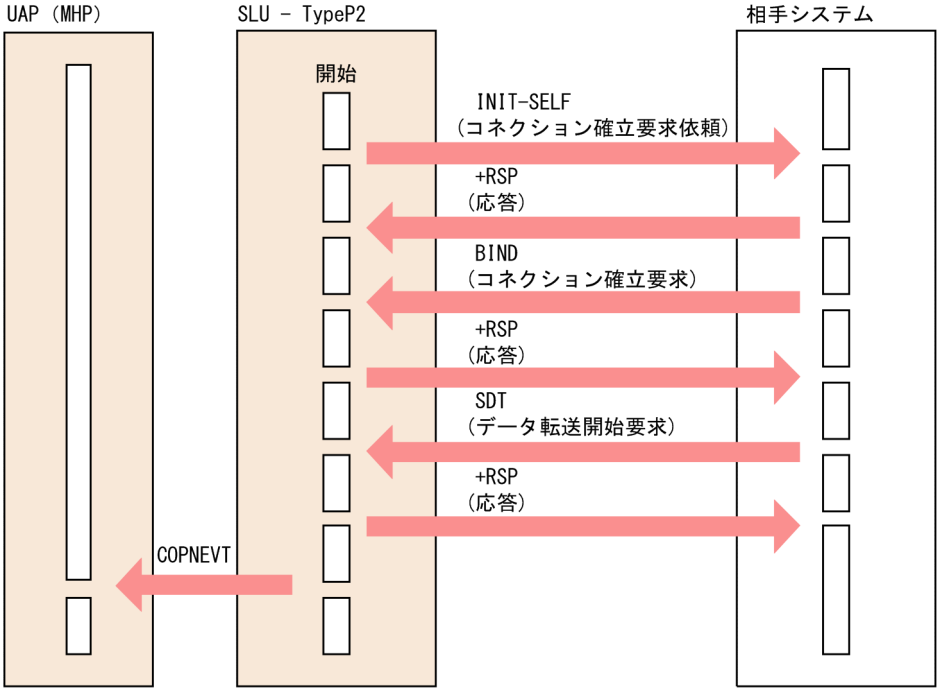
(b)　オンライン開始・再開始時の自動確立

次の場合，オンラインの開始・再開始時に SLU - TypeP2 はホストに対して確立要求を送信します。コネクション確立後，すべての論理端末を閉塞解除し，UAP にコネクションの確立（COPNEVT）を通知します。

- 起動種別が端末起動（コネクション定義（mcftalccn -k）に ws を指定）
- コネクションの確立方法が自動確立（コネクション定義（mcftalccn -i）に auto を指定）

オンライン開始・再開始時の自動確立を次の図に示します。

図 2-2　オンライン開始・再開始時の自動確立



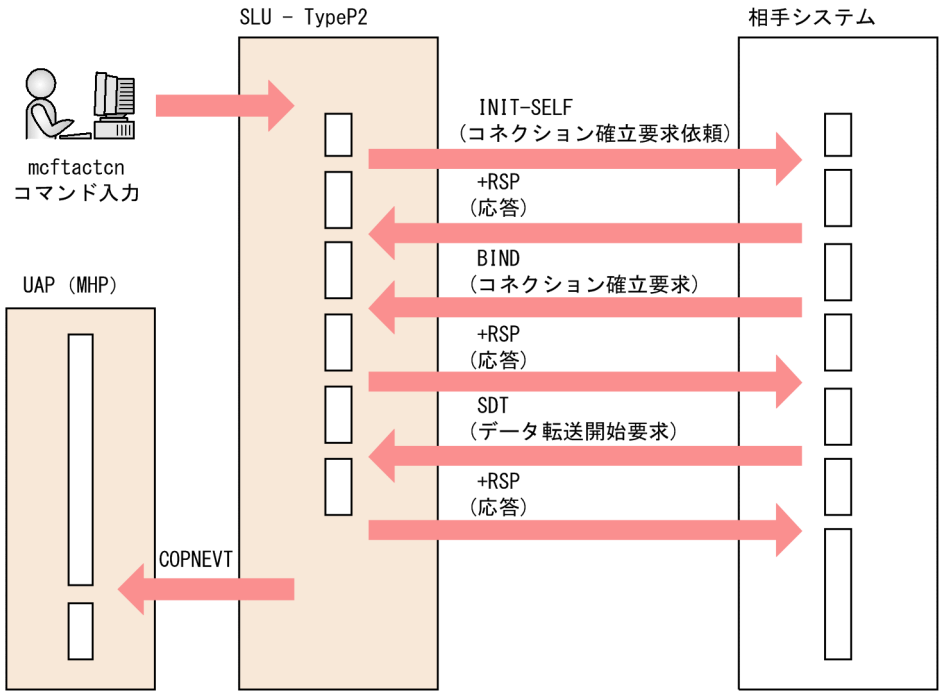
コネクション解放後に再確立する場合，運用コマンドを入力するか，API を発行する必要があります。

(c)　運用コマンド入力による手動確立

起動種別が端末起動の場合（コネクション定義（mcftalccn -k）に ws を指定），運用コマンド（mcftactcn）を入力することで，SLU - TypeP2 はホストに対して確立要求を送信します。コネクション確立後，すべての論理端末を閉塞解除し，UAP にコネクションの確立（COPNEVT）を通知します。

運用コマンド入力による手動確立を次の図に示します。

図 2-3　運用コマンド入力による手動確立

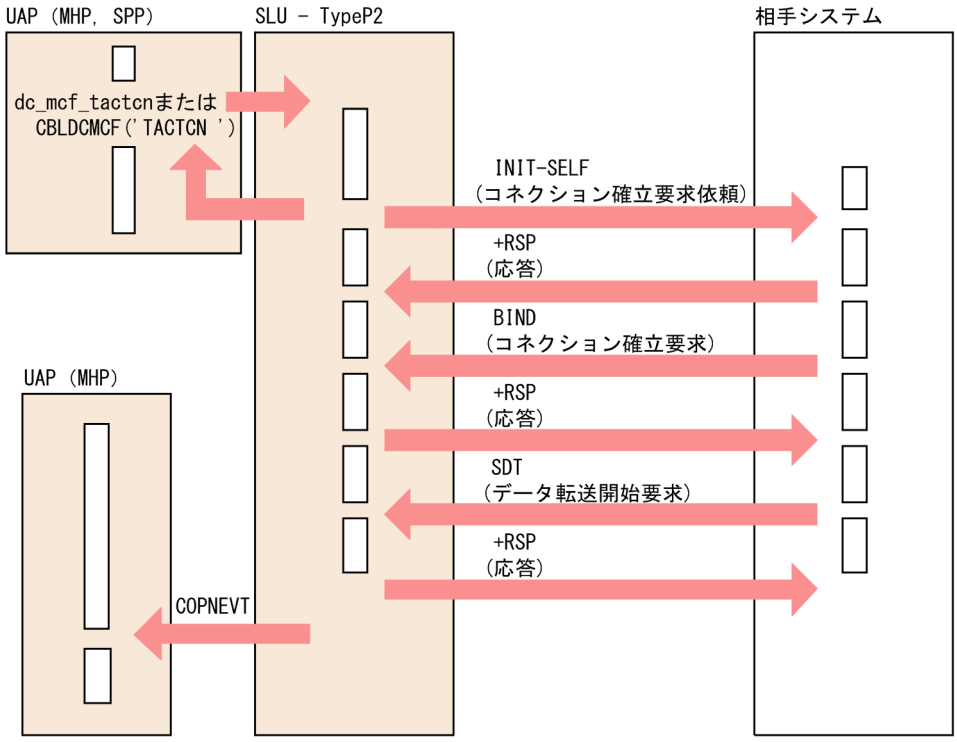


(d)　API 発行による手動確立

起動種別が端末起動の場合（コネクション定義（mcftalccn -k）に ws を指定），API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）を発行することで，SLU - TypeP2 はホストに対して確立要求を送信します。コネクション確立後，すべての論理端末を閉塞解除し，UAP にコネクションの確立（COPNEVT）を通知します。

API 発行による手動確立を次の図に示します。

図 2-4　API 発行による手動確立



(2)　コネクション確立時のチェック項目

ホストからの要求によってコネクションを確立する場合，SLU - TypeP2 はホストから受信した確立要求（BIND コマンド）のパラメタをチェックします。このチェックの結果，エラーが検出されると否定応答（センスコード=0821）が出力されます。否定応答（センスコード=0821）は，次のエラーが検出された場合に出力されます。

- BIND 指令中のセションパラメタが不正の場合
- 2 次局側でサポートしていない内容を含んでいる場合

SLU - TypeP2 で固有にチェックするパラメタの内容を次の表に示します。

表 2-1　パラメタのチェック内容

内容	位置（バイト）	長さ（バイト）	チェック基準値（16 進数字）
フォーマット，BIND 指令種別	1	1	$(01)_{16}$，または$(00)_{16}$
FM プロファイル番号	2	1	$(04)_{16}$
TS プロファイル番号	3	1	$(04)_{16}$
PLU プロトコル	4	1	$(b1)_{16}$
SLU プロトコル	5	1	$(b1)_{16}$

内容	位置（バイト）	長さ（バイト）	チェック基準値（16 進数字）
共通プロトコル	6	2	$(6080)_{16}$
SLU 送信最大 RU 長※	10	1	$(00)_{16}$，または MCF 通信構成定義の送信最大 RU 長（mcftalccn -r sndrusiz）の指定値
SLU 受信最大 RU 長※	11	1	$(00)_{16}$，または MCF 通信構成定義の受信最大 RU 長（mcftalccn -r rcvrusiz）の指定値
LU タイプ	14	1	$(00)_{16}$

注※

SLU 送信最大 RU 長と SLU 受信最大 RU 長は，相手システム（ホストシステム）で定義される最大 RU 長です。これらの値は，自システム（OpenTP1 システム）の MCF 通信構成定義（mcftalccn）の-r sndrusiz と-r rcvrusiz の指定値と，次に示す大小関係にある場合に有効となります。

- 自システムの送信最大 RU 長（-r sndrusiz）の指定値が，ホストシステムで指定されている SLU 送信最大 RU 長以下の場合
- 自システムの受信最大 RU 長（-r rcvrusiz）の指定値が，ホストシステムで指定されている SLU 受信最大 RU 長以上の場合

(3) コネクション確立時の再試行

SLU - TypeP2 は，コネクション確立時，コネクション定義の起動種別（mcftalccn -k）に端末起動（ws）が指定された場合，コネクション定義のコネクション確立再試行（mcftalccn -b）で指定された値に基づいてコネクション確立の再試行をします。

コネクション定義の詳細については，「mcftalccn（コネクション定義の開始）」を参照してください。

(4) コネクションの正常解放

コネクションの正常解放には，次の四つがあります。

- ホストからの正常解放
- オンライン終了時の正常解放
- 運用コマンド（mcftdctcn）の入力による正常解放
- API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）の発行による正常解放

コネクションが正常解放された場合，論理端末は自動的に閉塞され，SLU - TypeP2 は UAP にコネクションの解放（CCLSEVT）を通知します。なお，オンライン終了時は，CCLSEVT は通知されません。

コネクションの正常解放は，ユーザ間のデータ送受信が終了したあとに行ってください。

ホストからの正常解放を図 2-5 に，オンライン終了時の正常解放を図 2-6 に，運用コマンド入力による正常解放を図 2-7 に，API 発行による正常解放を図 2-8 に示します。

図 2-5　ホストからの正常解放

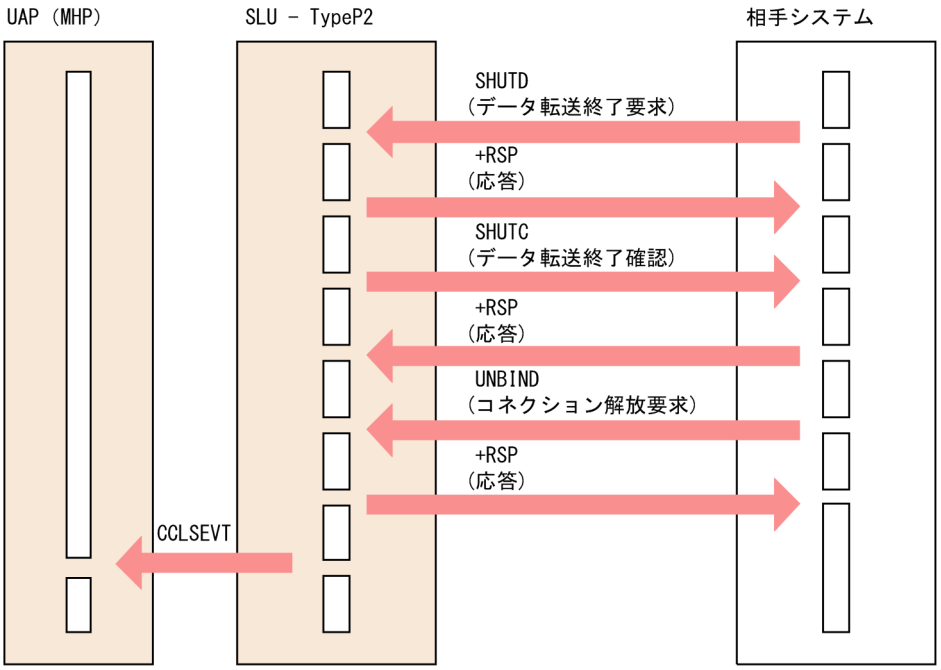
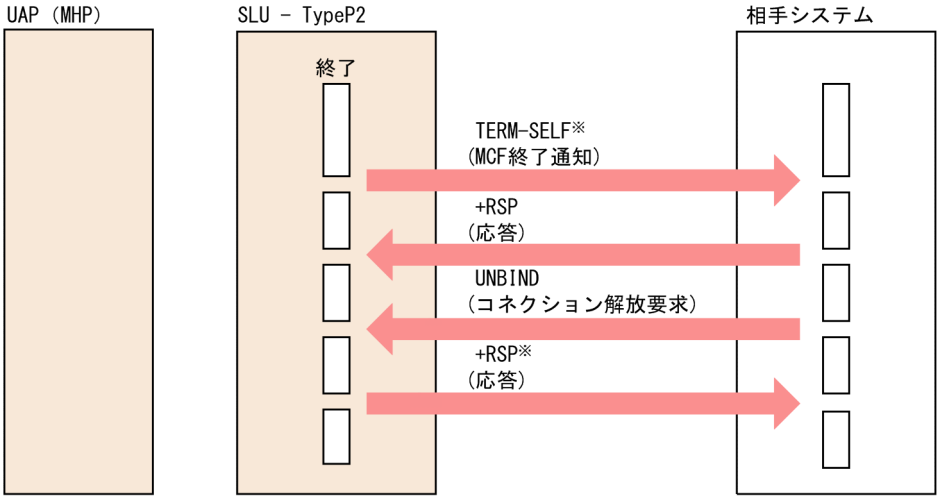


図 2-6　オンライン終了時の正常解放



注　オンライン終了時，CCLSEVTは通知されません。

注※　通信管理で発行されます。

図 2-7　運用コマンド入力による正常解放

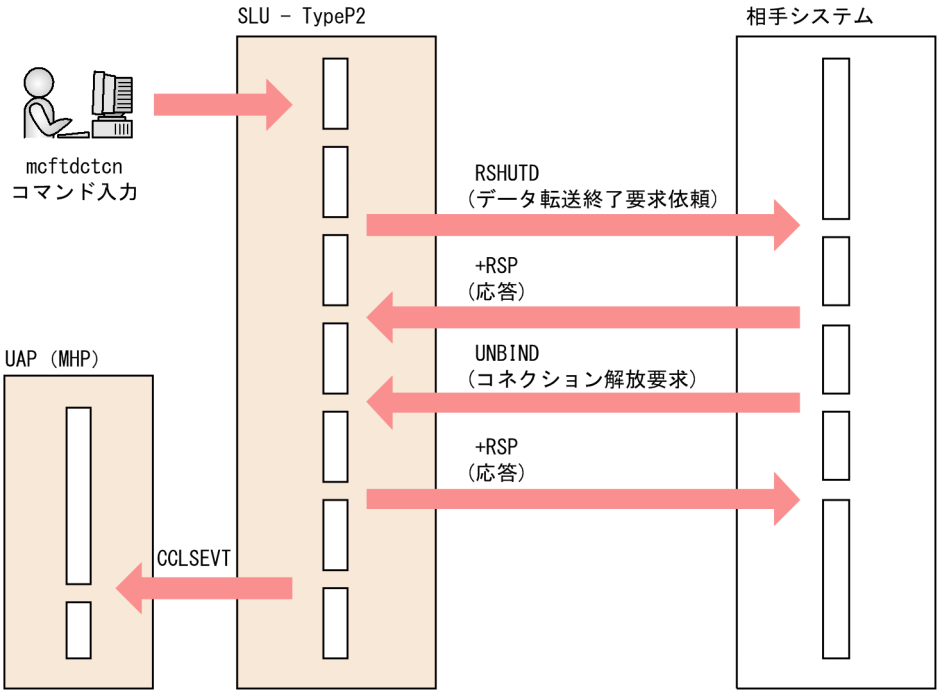
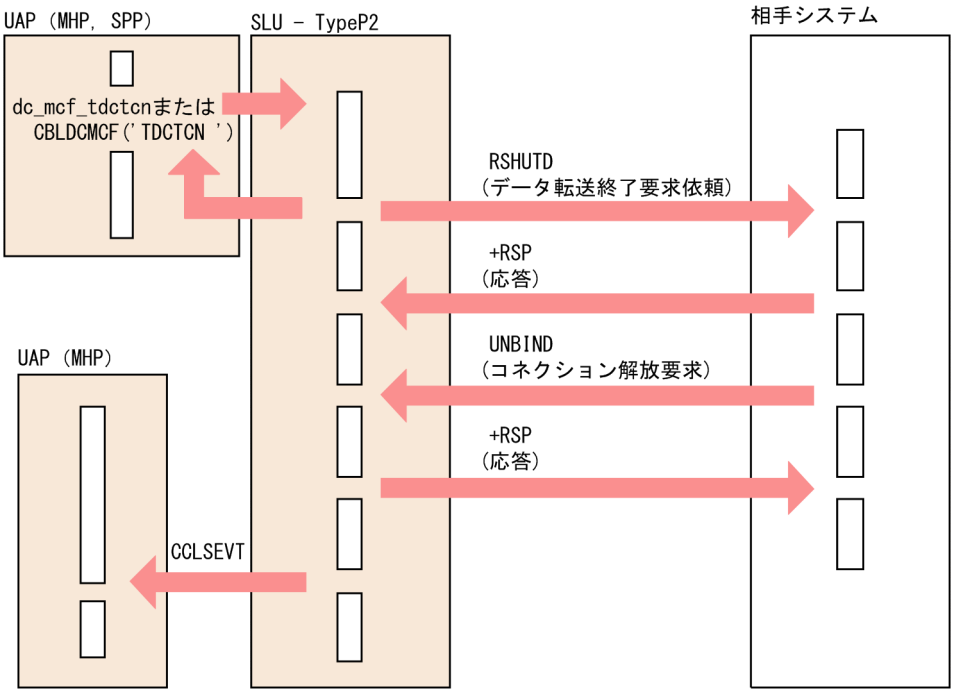


図 2-8　API 発行による正常解放



(5)　コネクションの強制解放

コネクションの強制解放には，次の四つがあります。

- 下位障害発生による強制解放
- 運用コマンド（mcftdctcn -f）の入力による強制解放

- 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△△')）の発行による強制解放
- バッファ不足などの内部障害による強制解放

注※

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。

コネクションが強制解放された場合，論理端末は自動的に閉塞され，SLU - TypeP2 は UAP にコネクションの解放（CERREVT）を通知します。下位障害，内部障害などの障害対策については，「9.　障害対策」を参照してください。

下位障害発生による強制解放を図 2-9 に，運用コマンド入力による強制解放を図 2-10 に，API 発行による強制解放を図 2-11 に，内部障害による強制解放を図 2-12 に示します。

図 2-9　下位障害発生による強制解放

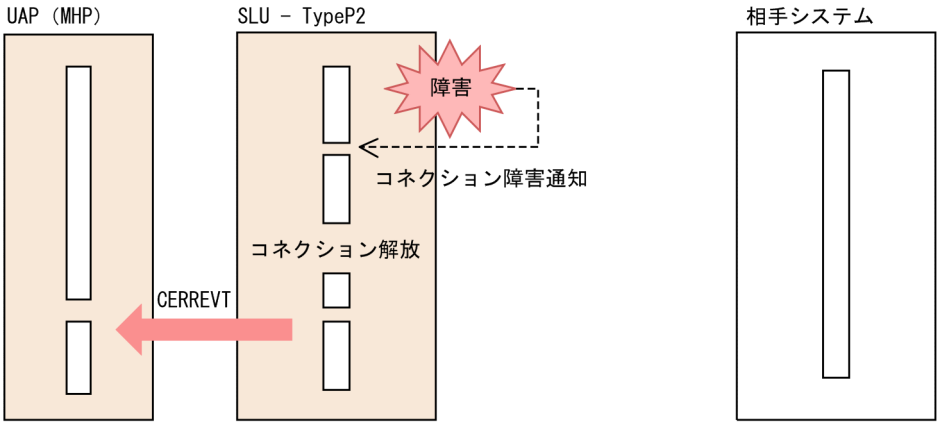
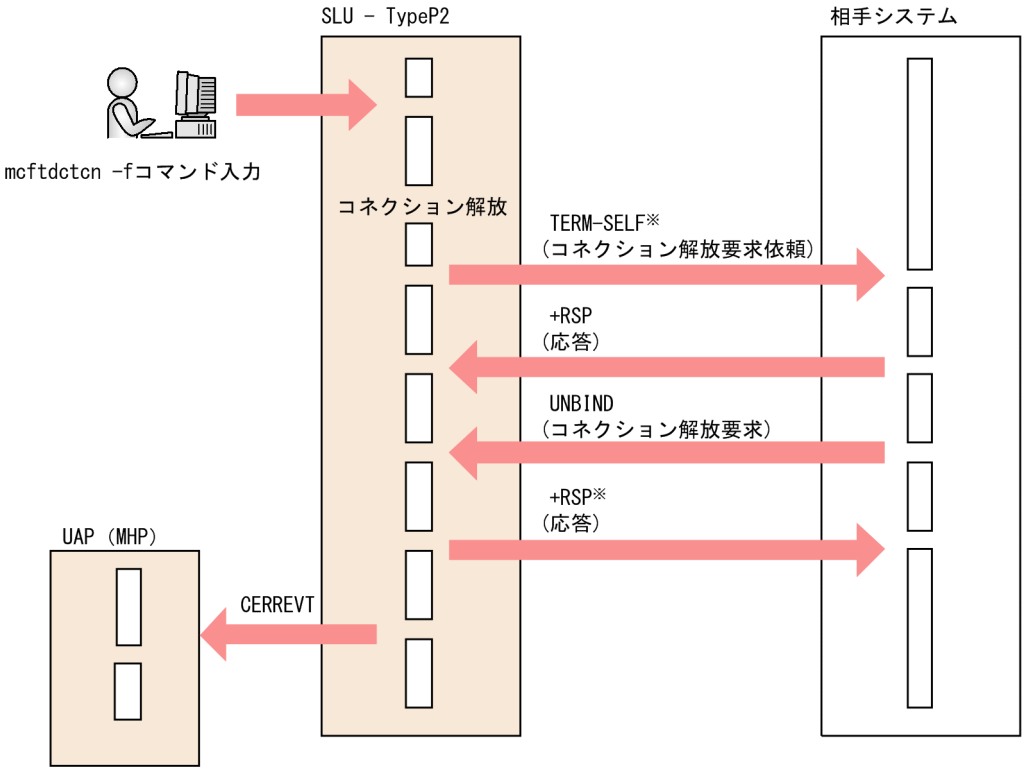


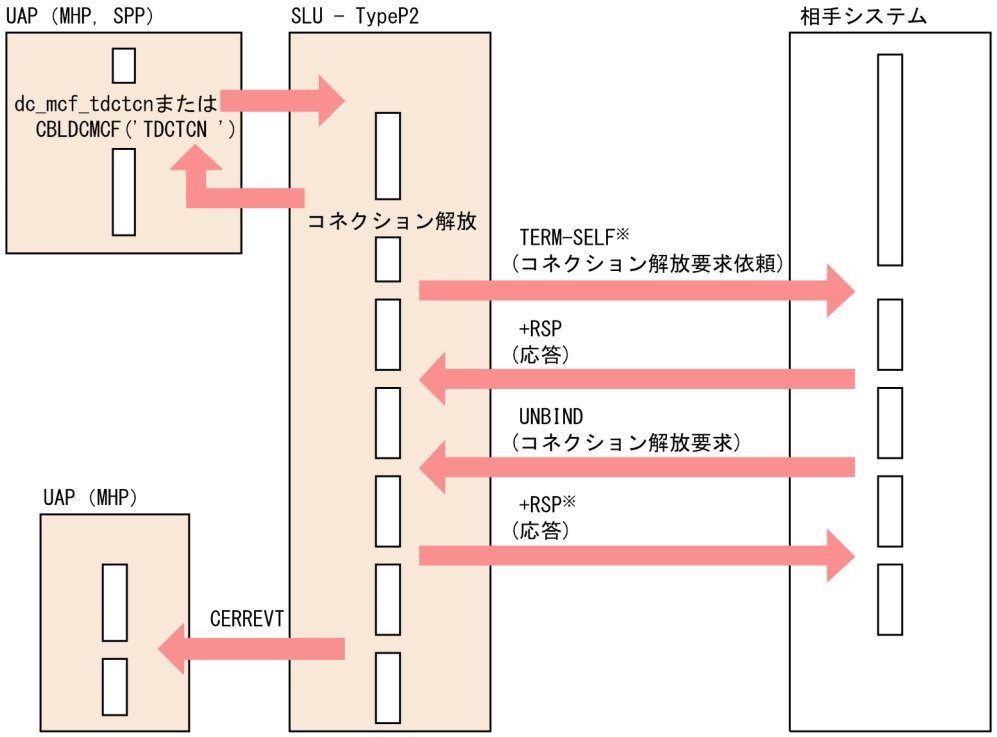
図 2-10　運用コマンド入力による強制解放



注※

通信管理で発行されます。

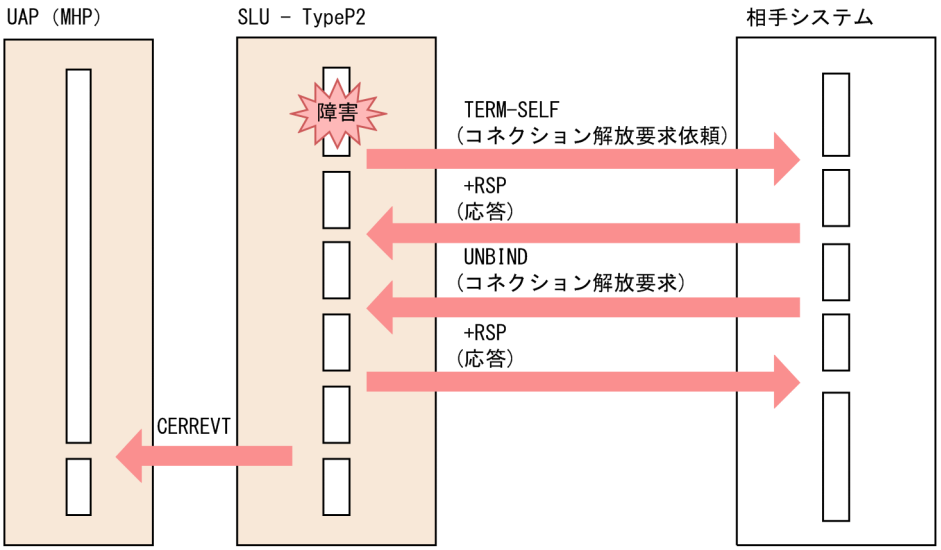
図 2-11　API 発行による強制解放



注※

通信管理で発行されます。

図 2-12　内部障害による強制解放



(6)　コネクション解放後の回復動作

システム定義のコネクション定義の起動種別（mcftalccn -k）に指定するオペランドによって，SLU - TypeP2 では解放したコネクションの回復動作が異なります。

コネクション定義の起動種別（mcftalccn -k）の指定内容と SLU - TypeP2 の動作を次の表に示します。

表 2-2　-k オプションの指定内容と SLU - TypeP2 の動作

-k オプションの指定内容	SLU - TypeP2 の動作
host（ホスト起動）	コネクション解放後，ホストからのコネクション確立要求を待ちます。
ws（端末起動）	コネクション解放後，ユーザからの運用コマンド（mcftactcn）の入力，または API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）の発行を待ちます。
auto（INIT-SELF 送信制御）	「2.1.5　INIT-SELF 送信制御」を参照してください。

なお，コマンド入力による手動回復なのか，システムによる自動回復なのかを示す回復動作情報は，SLU - TypeP2 が CERREVT または CCLSEVT に設定します。これらのイベントの種類および詳細については，「5.2　MCF イベントインタフェース」を参照してください。

コネクション定義の起動種別（mcftalccn -k）にホスト起動（host）が指定されたコネクションに対して運用コマンド（mcftactcn）が入力された場合，mcftactcn コマンドはエラーリターンして，メッセージログ（KFCA15342-E）を出力します。API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）が発行された場合，API はエラーリターンして，メッセージログ（KFCA11196-W）を出力します。

また，起動種別が端末起動の場合（コネクション定義（mcftalccn -k）に ws を指定），該当するコネクションに対して，次のどちらかの処理が実行されるまで，ホストからのコネクション確立要求は通信管理によって拒否されます。

- 運用コマンド（mcftactcn）が入力される
- API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）が発行される

2.1.2 コネクションと論理端末の関係

SLU - TypeP2 は，論理端末を通して，自システムの UAP とメッセージを送受信します。この論理端末は，SLU - TypeP2 と UAP との通信接点に当たります。

これに対してコネクションとは，SLU - TypeP2 が通信管理プログラムを介して，相手システムの UAP とメッセージを送受信するときに確立するものです。コネクションと論理端末の指定を対応させると，自システムと相手システムとの論理的通信路が確立でき，AP 間通信ができるようになります。コネクションと論理端末は，システム定義時に対応させます。

2.1.3 論理端末とアプリケーションの型の関係

SLU - TypeP2 で扱う論理端末の端末タイプには，次の三つがあります。

- send（送信型論理端末）
- receive（受信型論理端末）
- request（問い合わせ型論理端末）

アプリケーションは，ユーザが送受信データの中に指定したアプリケーション名をキーとして，一つの UAP（MHP）プロセスで実行されます。アプリケーションは，サービスの方式によって型が異なります。この型を MCF の属性の一つとして，システム定義時に指定します。SLU - TypeP2 で使用するアプリケーションの型は，非応答型（noans）です。

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

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

論理端末の端末タイプ	メッセージ	アプリケーションの型	UAP インタフェース	通信形態
send （送信型論理端末）	一方送信メッセージ	非応答型	send	分岐送信
			resend	
receive	一方送信メッセージ	非応答型	receive	一方受信

論理端末の端末タイプ	メッセージ	アプリケーションの型	UAP インタフェース	通信形態
（受信型論理端末）	一方送信メッセージ	非応答型	receive	一方受信
request （問い合わせ型論理端末）	問い合わせメッセージ	非応答型	send	分岐送信
			resend	
			sendrecv（セグメント送信時，先頭セグメント受信時）	同期送受信（同期問い合わせ応答）
			recvsync（後続セグメント受信時）	同期受信（同期問い合わせ応答）
			receive	一方受信

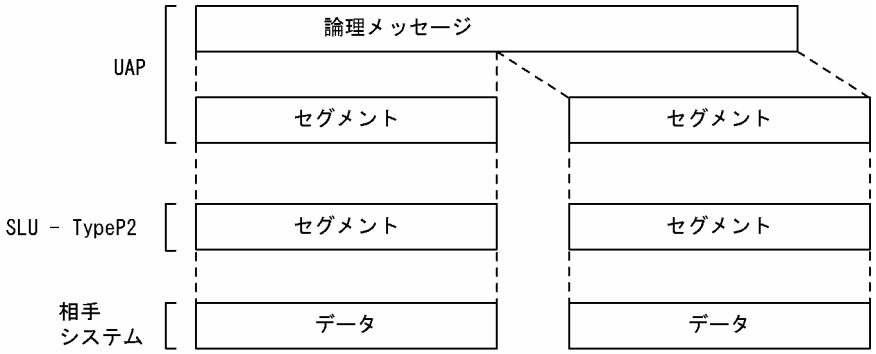
2.1.4　メッセージの分割と組み立て

SLU - TypeP2 と UAP との間では，メッセージを**セグメント**と呼ばれる単位に分割して扱います。一つの業務で処理するメッセージを**論理メッセージ**といいます。論理メッセージは，一つまたは複数のセグメントで構成されます。

UAP で送受信命令を発行すると，一つのセグメントを受信または送信します。論理メッセージが複数のセグメントで構成される場合，セグメントの数だけ送受信命令を発行してください。

セグメントと論理メッセージの関係を次の図に示します。

図 2-13　セグメントと論理メッセージ



また，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。

メッセージの形式，パラメタの詳細，データ名などについては，「3.　C 言語のライブラリ関数」または「4.　COBOL-UAP 作成用プログラムインタフェース」を参照してください。

2.1.5　INIT-SELF 送信制御

INIT-SELF 送信制御は，通信管理プログラムとホスト間の通信路の状態によって，ホストに対してコネクション確立要求を送信したり，ホストからコネクションの確立要求を待ち合わせたりする機能です。

この機能を使用すると，ホストに対してコネクション確立要求を送信する回数が減るため，ホストの負荷を軽減できます。

(1)　コネクションの確立

INIT-SELF 送信制御を使用すると，SLU - TypeP2 はオンラインの開始・再開始時およびコネクションの解放時にコネクションを確立します。

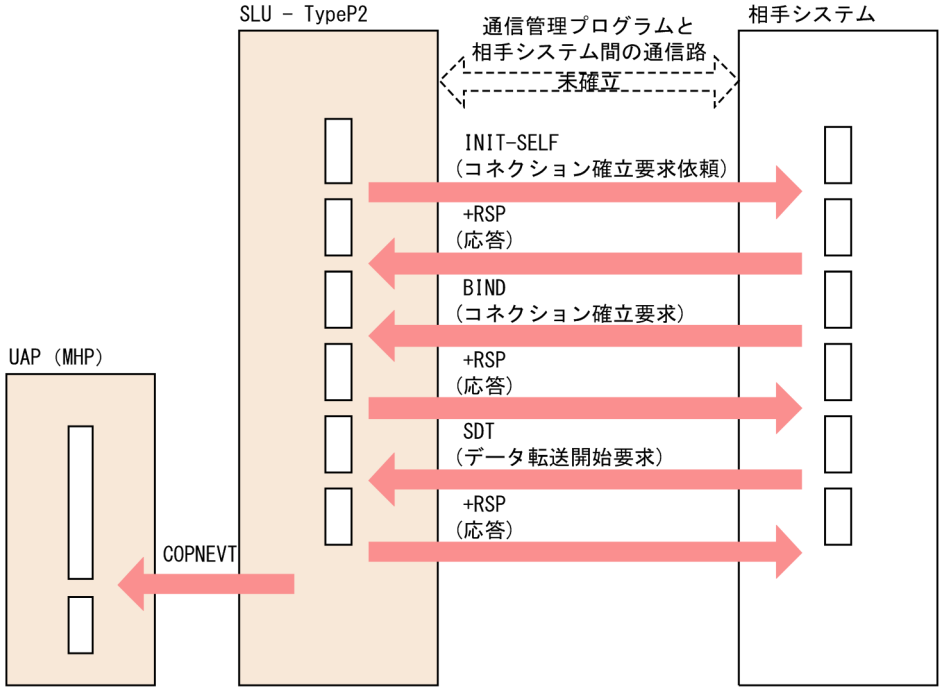
コネクションの確立の詳細を以降に説明します。

(a)　INIT-SELF を送信する場合のコネクションの確立

通信管理プログラムとホスト間の通信路が未確立の場合，SLU - TypeP2 はホストに対して確立要求を送信します。コネクション確立後，すべての論理端末を閉塞解除し，UAP にコネクションの確立（COPNEVT）を通知します。

INIT-SELF を送信する場合のコネクションの確立を次の図に示します。

図 2-14　INIT-SELF を送信する場合のコネクションの確立

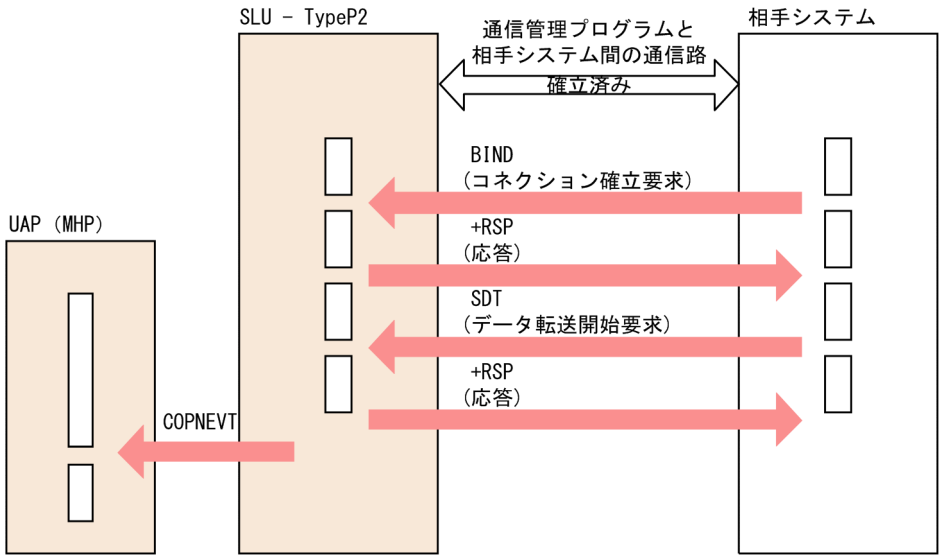


(b)　INIT-SELF を送信しない場合のコネクションの確立

通信管理プログラムとホスト間の通信路が確立済みの場合，SLU - TypeP2 はホストからの確立要求を待ちます。コネクション確立後，すべての論理端末を閉塞解除し，UAP にコネクションの確立（COPNEVT）を通知します。

INIT-SELF を送信しない場合のコネクションの確立を次の図に示します。

図 2-15　INIT-SELF を送信しない場合のコネクションの確立

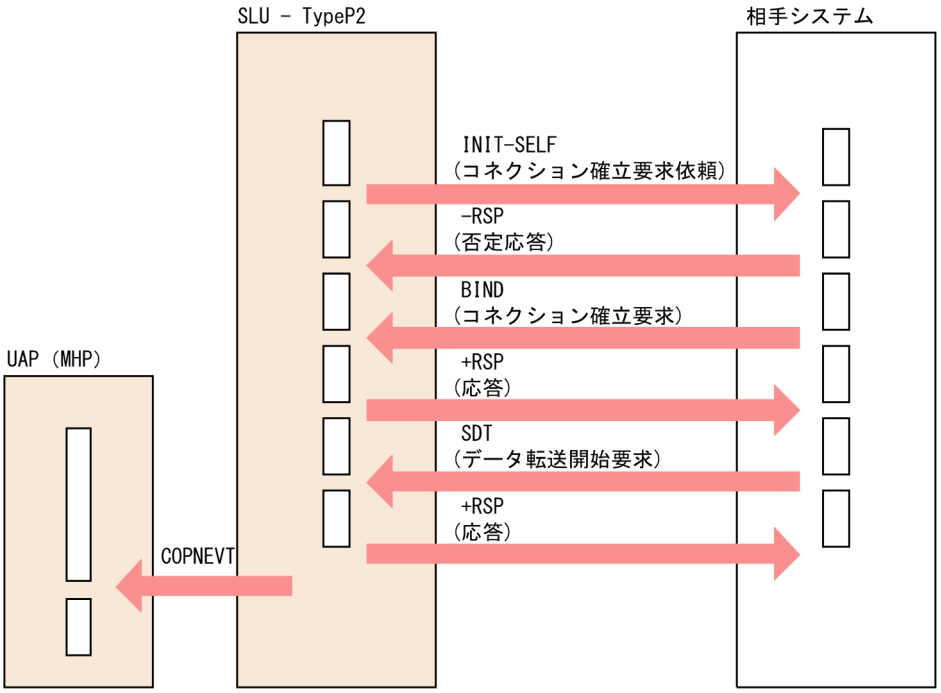


(c)　INIT-SELF の否定応答を受信した場合

INIT-SELF の否定応答を受信した場合，SLU - TypeP2 はホストからの確立要求を待ちます。コネクション確立後，すべての論理端末を閉塞解除し，UAP にコネクションの確立（COPNEVT）を通知します。

INIT-SELF の否定応答受信を次の図に示します。

図 2-16　INIT-SELF の否定応答受信

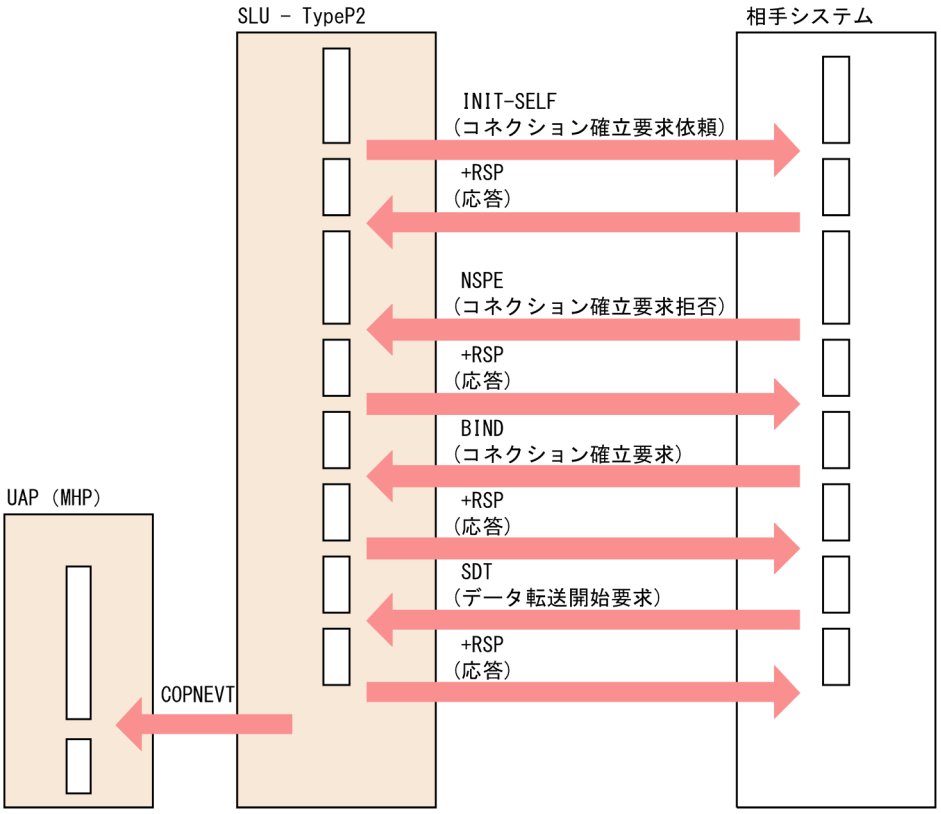


(d)　NSPE を受信した場合

NSPE を受信した場合，SLU - TypeP2 はホストからの確立要求を待ちます。コネクション確立後，すべての論理端末を閉塞解除し，UAP にコネクションの確立（COPNEVT）を通知します。

NSPE 受信を次の図に示します。

図 2-17　NSPE 受信



(e)　通信管理プログラムとホスト間の通信路の確立に失敗した場合

通信管理プログラムとホスト間の通信路の確立の失敗した場合，SLU - TypeP2 は確立するまで無限に再試行します。

確立の再試行間隔は，コネクション確立障害時の確立再試行間隔（mcftalccn -b bretryint）で指定します。

ただし，次の場合は直ちに再試行します。

- コネクション確立障害時の確立再試行間隔（mcftalccn -b bretryint）に 0 を指定
- コネクション定義（mcftalccn -b）の bretry オペランドに no を指定
- 送信バッファ面数の不足を検出
- BIND のセションパラメタの不一致または不正を検出

(2)　コネクション解放後の回復動作

INIT-SELF 送信制御を使用する場合のコネクション解放後の回復動作の詳細を以降に説明します。

(a)　ホストからの解放

ホストから正常に解放された場合，SLU - TypeP2 は CCLSEVT を通知したあと，コネクションを再確立します。通信管理プログラムとホスト間の通信路の状態によって，ホストに対してコネクション確立要求を送信したり，ホストからコネクションの確立要求を待ち合わせたりします。

(b) オンライン終了時の解放

オンラインが終了した場合，SLU - TypeP2 はコネクションを解放します。このとき，CCLSEVT の通知も，コネクションの再確立もしません。

(c) 運用コマンドまたは API による正常解放

運用コマンドまたは API によって正常解放された場合，SLU - TypeP2 は CCLSEVT を通知したあと，コネクションを再確立します。通信管理プログラムとホスト間の通信路の状態によって，ホストに対してコネクション確立要求を送信したり，ホストからコネクションの確立要求を待ち合わせたりします。

(d) 下位層障害発生による強制解放

下位層障害によって強制解放された場合，SLU - TypeP2 は CERREVT を通知したあと，コネクションを再確立します。通信管理プログラムとホスト間の通信路の状態によって，ホストに対してコネクション確立要求を送信したり，ホストからコネクションの確立要求を待ち合わせたりします。

(e) 運用コマンドまたは API による強制解放

運用コマンドまたは API によって強制解放された場合，SLU - TypeP2 は CERREVT を通知したあと，コネクションを再確立します。通信管理プログラムとホスト間の通信路の状態によって，ホストに対してコネクション確立要求を送信したり，ホストからコネクションの確立要求を待ち合わせたりします。

(f) 内部障害による解放

内部障害によって強制解放された場合，SLU - TypeP2 は CERREVT を通知したあと，コネクションを再確立します。通信管理プログラムとホスト間の通信路の状態によって，ホストに対してコネクション確立要求を送信したり，ホストからコネクションの確立要求を待ち合わせたりします。

(3) 注意事項

- コネクションの再確立時にホストからのコネクション確立要求を待ち合わせる場合，SLU - TypeP2 は通信管理プログラムとの接点を一時的に切り離します。SLU - TypeP2 と通信管理プログラムの接点が切り離されているときにホストからコネクション確立要求を受信すると，通信管理プログラムが否定応答を送信します。コネクション確立要求の否定応答を受信した場合，再度コネクション確立要求を送信してください。

2.2　AP 間通信メッセージの送受信

SLU - TypeP2 では，次に示すメッセージの送受信をします。

- 問い合わせメッセージの送信と応答メッセージの受信
- 一方送信メッセージの送信と受信

2.2.1　問い合わせメッセージの送信と応答メッセージの受信

ホスト（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，応答を同期型のメッセージで受信します。

ホストへ問い合わせメッセージを送信するとき，UAP から sendrecv 関数を呼び出します。問い合わせメッセージが複数のセグメントで構成される場合，セグメントの数だけ sendrecv 関数を呼び出す必要があります。UAP がメッセージのセグメントを送信し終わると，SLU - TypeP2 は，ホストからの応答を待ちます。

ホストからの応答メッセージを受信すると，SLU - TypeP2 はメッセージの先頭セグメントを UAP に返します。応答メッセージが複数のセグメントで構成される場合，UAP は recvsync 関数を呼び出して後続セグメントを受信できます。

SLU - TypeP2 では，問い合わせ応答のデータ送受信時，ホストへ送信する応答識別（RQD または RQE）を指定することで，送信に対する応答（+RSP または-RSP）が必要かどうかを決定できます。応答を不要と指定すると，エラーが発生した場合だけ応答（-RSP）が送信されます。応答識別の指定については，「mcftalccn（コネクション定義の開始）」の-d オプションを参照してください。

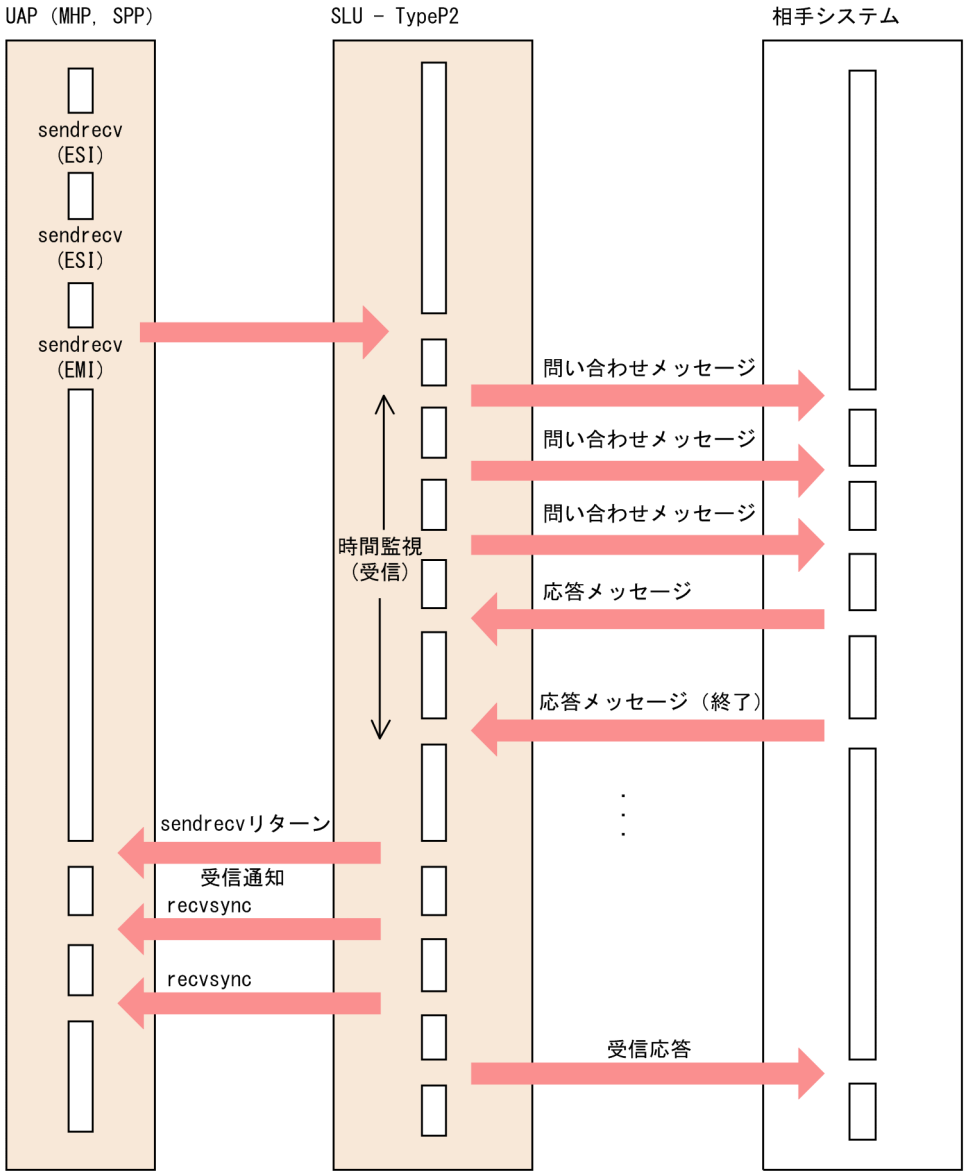
また，ユーザの指定によっては，問い合わせメッセージ送信完了から応答メッセージ受信までの間，時間監視ができます。監視時間内に応答がない場合，同期送受信要求がエラーリターンします。

ホストからの応答メッセージがダミーメッセージ（0 バイトデータ）の場合，データは UAP に通知されません。

自システムから複数セグメントメッセージを送信する場合，先頭または中間セグメントに対して否定応答（-RSP）を返さないで，最終セグメントに対して否定応答（-RSP）を返す相手システムと接続してください。

問い合わせメッセージの送信と応答メッセージの受信を次の図に示します。

図 2-18　問い合わせメッセージの送信と応答メッセージの受信



2.2.2　一方送信メッセージの送信と受信

一方送信メッセージを送信し，また，ホストから一方送信メッセージを受信します。

ホストへ一方送信メッセージを送信するとき，UAP から send 関数を呼び出します。一方送信メッセージが複数のセグメントで構成される場合，セグメントの数だけ send 関数を呼び出してください。一方送信メッセージの送信が完了すると，SLU - TypeP2 は，出力キューにある一方送信メッセージを送信済みとします。

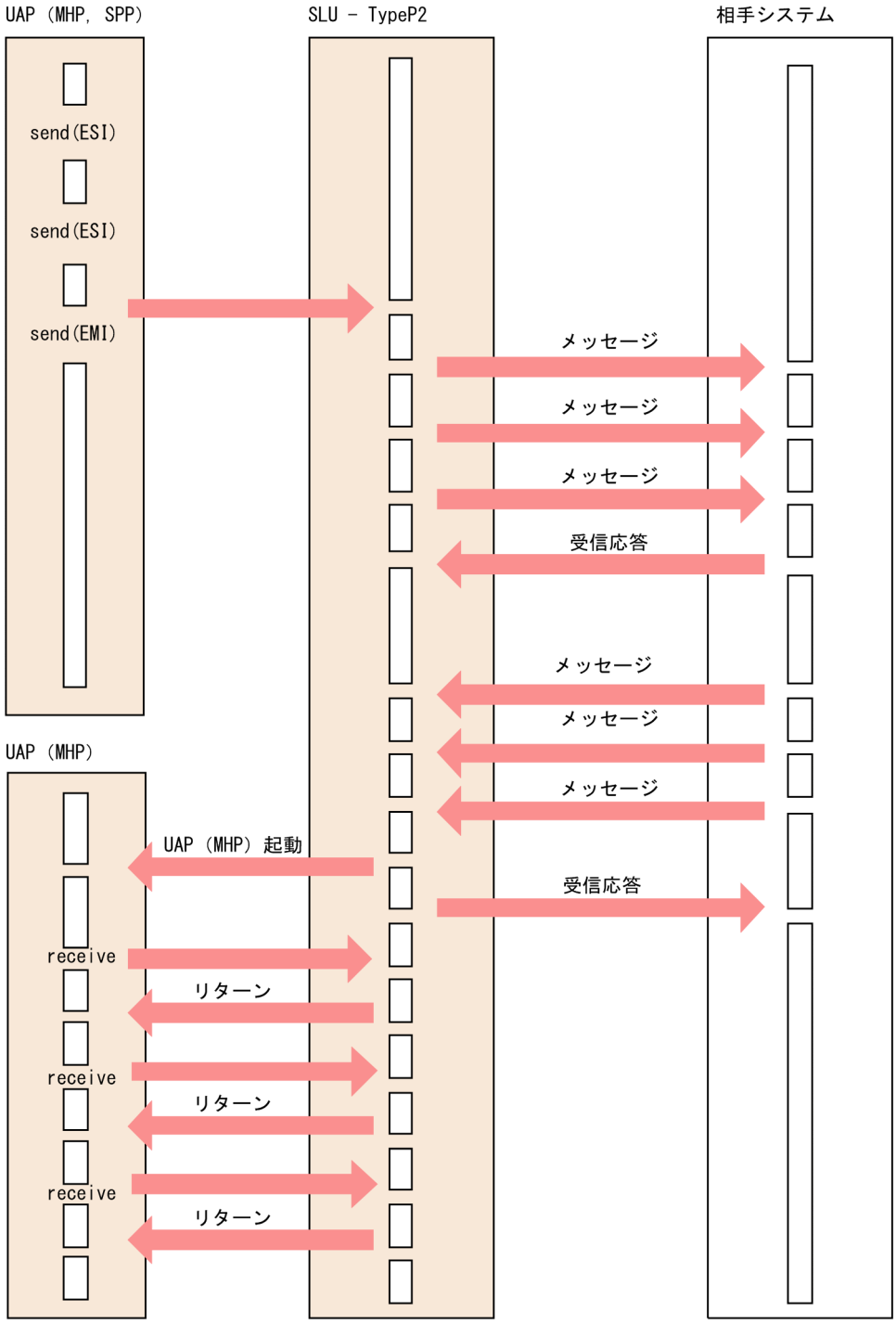
ホストからの応答も，一方送信メッセージとして受信します。一方送信メッセージを受信すると，SLU - TypeP2 は，メッセージに対応するアプリケーションを起動します。アプリケーションに該当する UAP は，receive 関数を呼び出してメッセージを受信できます。一方送信メッセージが複数のセグメントで構成される場合，セグメントの数だけ receive 関数を呼び出してください。

ホストから受信した一方送信メッセージがダミーメッセージ（0 バイトデータ）の場合，データは UAP に通知されません。

自システムから複数セグメントメッセージを送信する場合，先頭または中間セグメントに対して否定応答（-RSP）を返さないで，最終セグメントに対して否定応答（-RSP）を返す相手システムと接続してください。

一方送信メッセージの送信と受信を次の図に示します。

図 2-19　一方送信メッセージの送信と受信

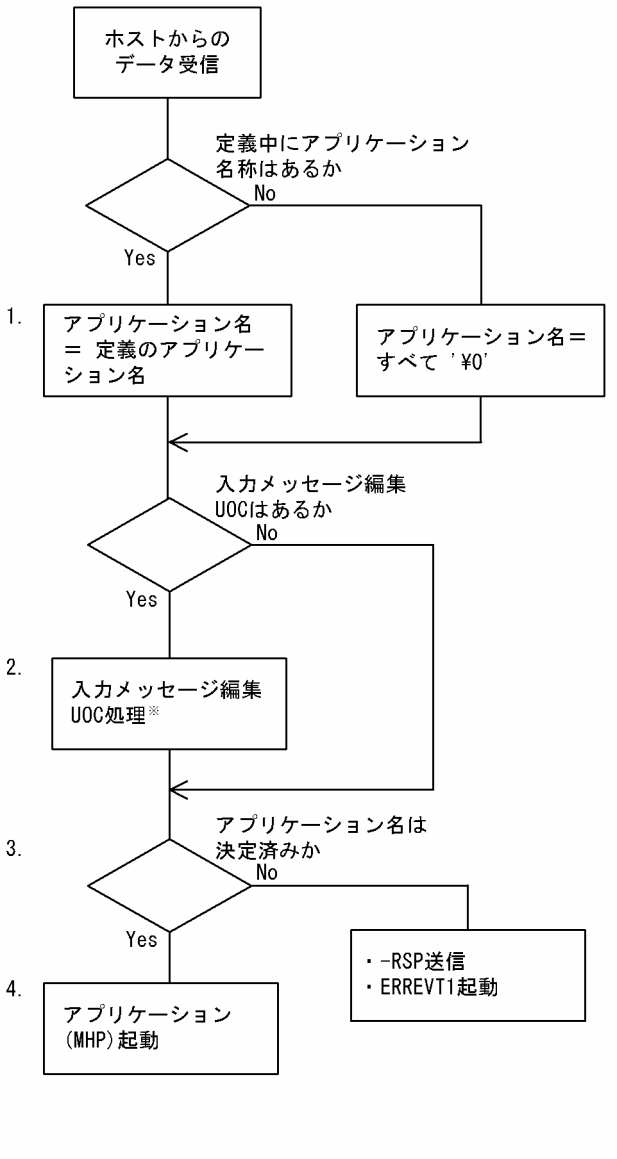


2.2.3　アプリケーション名の決定

SLU - TypeP2 では，システム定義時に，論理端末定義（mcftalcle）の-v オプションで指定された値を，ホストからの一方送信メッセージ受信時に，アプリケーション名とします。論理端末定義で指定を省略した場合は，入力メッセージ編集 UOC を使用してアプリケーション名を決定する必要があります。入力メッセージ編集 UOC については，「5.1.2　入力メッセージ編集 UOC インタフェース」を参照してください。

アプリケーション名決定の流れを次の図に示します。

図 2-20　アプリケーション名決定の流れ



注※

入力メッセージ編集 UOC によるアプリケーション名の決定については，「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」を参照してください。

1. 入力メッセージの中の情報からアプリケーション名を決定します。
2. 入力メッセージがある場合，入力メッセージ編集 UOC による処理をします。

3. アプリケーション名が決定しているかどうかを判定します。

4. 決定されたアプリケーション名に基づいてアプリケーションを起動します。

3 C 言語のライブラリ関数

この章では，SLU - TypeP2 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

SLU - TypeP2 で使用する C 言語のライブラリ関数の一覧を，次の表に示します。

表 3-1　C 言語のライブラリ関数の一覧

関数名	機能
dc_mcf_receive	一方送信メッセージの受信
dc_mcf_recvsync	同期型の応答メッセージの受信
dc_mcf_resend	メッセージの再送
dc_mcf_send	一方送信メッセージの送信
dc_mcf_sendrecv	同期型の問い合わせ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_receive － 一方送信メッセージの受信（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

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

機能

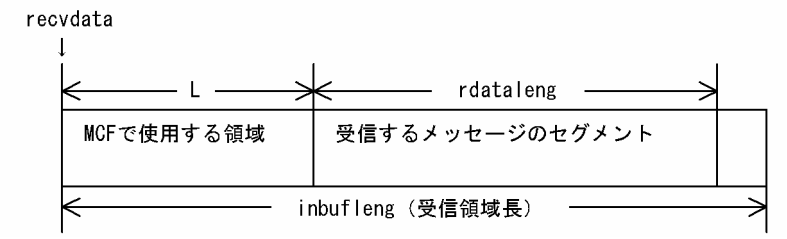
論理端末に届いたメッセージのうち，一つのセグメントを受信します。セグメントの数だけ dc_mcf_receive 関数を呼び出すと，一つの論理メッセージを受信できます。

受信できるメッセージの一つのセグメントの最大長は，32767 バイトです。

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

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

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}〔｜{DCMCFBUF1｜DCMCFBUF2}〕
```

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。

DCMCFBUF1

バッファ形式 1 を使用する場合に設定します。

DCMCFBUF2

バッファ形式 2 を使用する場合に設定します。

●commform

DCNOFLAGS を設定します。

●termnam

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

処理終了後，termnam には OpenTP1 から値が返ります。

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

中間セグメントまたは最終セグメントを受信した場合，値は返されません。

●resv01

NULL またはヌル文字列を設定します。

●recvdata

セグメントを受信する領域を設定します。

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

処理終了後，recvdata には OpenTP1 から値が返ります。

●inbufleng

セグメントを受信する領域の長さを設定します。

OpenTP1 から値が返される引数

●termnam

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

中間セグメントまたは最終セグメントを受信する場合は，ここで返された論理端末名称を，termnam に設定してください。

●recvdata

受信したセグメントの内容が返されます。

●rdataleng

受信したセグメントの長さが返されます。

●time

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

リターン値

リターン値	リターン値 (数値)	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71000	-12000	先頭セグメントを受信する dc_mcf_receive 関数を 2 回以上呼び出しています。中間セグメントまたは最終セグメントを受信する場合は，action に DCMCFSEG を設定して dc_mcf_receive 関数を呼び出してください。
DCMCFRTN_71001	-12001	メッセージの最終セグメントを受信したあとで，次のセグメントを受信する dc_mcf_receive 関数を呼び出しています。直前に呼び出した dc_mcf_receive 関数でメッセージはすべて受信しました。このリターン値が返されたあとに，再び dc_mcf_receive 関数を呼び出した場合は，リターン値 DCMCFRTN_72000 が返されます。
DCMCFRTN_71002	-12002	メッセージキューからの入力処理中に障害が発生しました。
		メッセージキューが閉塞されています。
DCMCFRTN_72000	-13000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，中間セグメントまたは最終セグメントを受信する dc_mcf_receive 関数を呼び出しています。先頭セグメントを受信する場合は，action に DCMCFFRST を設定して dc_mcf_receive 関数を呼び出してください。

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

（凡例）

－：該当しません。

dc_mcf_recvsync － 同期型の応答メッセージの受信（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

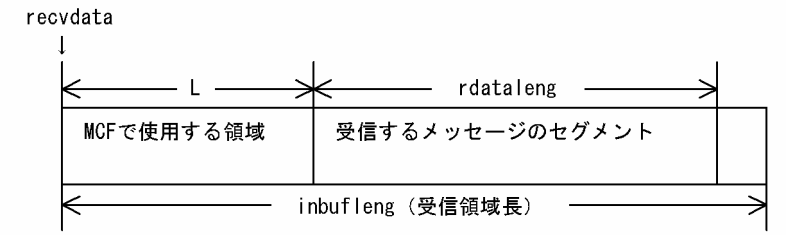
```
#include <dcmcf.h>
int dc_mcf_recvsync(action, commform, termnam, resv01,
                    recvdata, rdataleng, inbufleng,
                    time, resv02)
DCLONG      action;
DCLONG      commform;
char        *termnam;
char        *resv01;
char        *recvdata;
DCLONG      *rdataleng;
DCLONG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

機能

相手システムから同期型で受信したメッセージのうち，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，32767 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

中間セグメントおよび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

受信したセグメントの内容が返されます。

●rdataleng

受信したセグメントの長さが返されます。

●time

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

リターン値

リターン値	リターン値（数値）	意味
DCMCFRTN_00000	0	正常に終了しました。
DCMCFRTN_71001	-12000	メッセージの最終セグメントを受信したあとで，次のセグメントを受信する dc_mcf_recvsync 関数を呼び出しています。直前に呼び出した dc_mcf_recvsync 関数でメッセージはすべて受信しました。 このリターン値が返されたあとに，再び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	メッセージ受信に必要な管理テーブルが確保できませんでした。
		プロセスのローカルメモリが不足しています。
DCMCFRTN_72000	-13000	先頭セグメントを受信する dc_mcf_receive 関数を呼び出す前に，dc_mcf_recvsync 関数を呼び出しています。
		リターン値 DCMCFRTN_71001 が返されたあとに，再び dc_mcf_recvsync 関数を呼び出しています。
DCMCFRTN_72001	-13001	termnam に設定した論理端末名称が間違っています。
		termnam に設定した論理端末名称は，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られました。
DCMCFRTN_72016	-13016	action に設定した値が間違っています。
		resv01 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。
DCMCFRTN_72025	-13025	action に設定したセグメント種別（DCMCFFRST または DCMCFSEG）の値が間違っています。
DCMCFRTN_72036	-13036	inbufleng の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
上記以外	−	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_resend － メッセージの再送（C 言語）

形式

ANSI C，C++ の形式

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

K&R 版 C の形式

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

機能

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

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

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

UAP で値を設定する引数

●action

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

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

DCMCFSEQ

再送するメッセージに出力通番を付け直す場合に設定します。

DCMCFNSEQ

再送するメッセージに出力通番を付け直さない場合に設定します。

DCMCFNORM

一般の一方送信メッセージとして再送する場合に設定します。

DCMCFPRIO

優先の一方送信メッセージとして再送する場合に設定します。

DCMCFLAST

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

●commform

一方送信を示す，DCMCFOUT を設定します。

●rtermnam

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

●resv01

NULL を設定します。

●oseqid

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

DCMCFRID_NORM

一般の一方送信メッセージを対象とする場合に設定します。

DCMCFRID_PRIO

優先の一方送信メッセージを対象とする場合に設定します。

省略した場合は，DCMCFRID_NORM（一般の一方送信メッセージを対象）が設定されます。
DCMCFRID_PRIO を設定した場合は，otermnam に出力先の論理端末名称を設定できません。

●orgseq

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

●otermnam

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

●resv02，resv03，resv04

NULL を設定します。

●opcd

DCNOFLAGS を設定します。

リターン値

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

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

（凡例）

－：該当しません。

注意事項

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

-e オプション

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

-l オプション

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

dc_mcf_send － 一方送信メッセージの送信（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

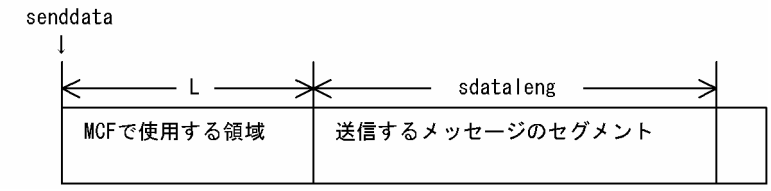
```
#include<dcmcf.h>
int dc_mcf_send (action, commform, termnam, resv01,
                 senddata,sdataleng, resv02, opcd)
DCLONG   action;
DCLONG   commform;
char     *termnam;
char     *resv01;
char     *senddata;
DCLONG   sdataleng;
char     *resv02;
DCLONG   opcd;
```

機能

相手システムへ送る一方送信メッセージのうち，一つのセグメントを送信します。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，優先か一般か，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場合も，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場合は，0 を設定します。

●resv02

NULL またはヌル文字列を設定します。

●opcd

DCNOFLAGS を設定します。

リターン値

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

リターン値	リターン値 (数値)	意味
DCMCFRTN_72016	-13016	action に設定したメッセージ種別（DCMCFNORM または DCMCFPRIO）の値が間違っています。
		action に設定した値が間違っています。
		opcd に設定した値が間違っています。
		resv01 または resv02 に設定した値が間違っています。
		引数に設定した値に間違いがあります。
DCMCFRTN_72017	-13017	action に設定した出力通番の要否（DCMCFSEQ または DCMCFNSEQ）の値が間違っています。
DCMCFRTN_72024	-13024	commform に設定した値が間違っています。
DCMCFRTN_72026	-13026	action に設定したセグメント種別（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 － 同期型の問い合わせメッセージの送受信（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

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

機能

同期型でメッセージを送信したあと，同期型でメッセージを受信します。

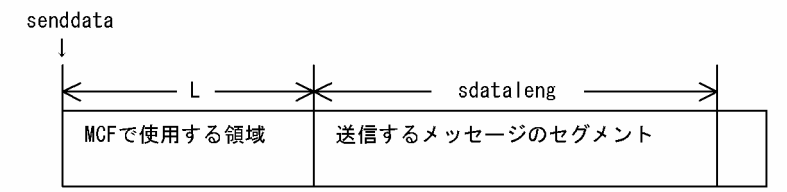
相手システムへ送るメッセージのうち，一つのセグメントを送信します。セグメントの数だけ dc_mcf_sendrecv 関数を呼び出すと，一つの論理メッセージを送信できます。

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。中間セグメントまたは最終セグメントを受信する場合は，dc_mcf_recvsync 関数を呼び出してください。

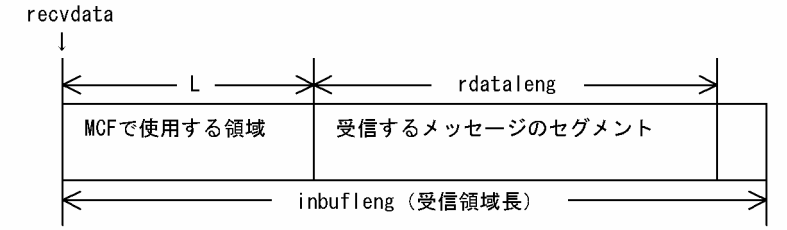
受信できるメッセージの一つのセグメントの最大長は，32767 バイトです。また，送信できるメッセージの一つのセグメントの最大長は，32000 バイトです。

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

●セグメントを送信する領域



●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 関数を呼び出すと，相手システムからの応答を待ちます。

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。

●recvdata

セグメントを受信する領域を設定します。

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

処理終了後，recvdata には OpenTP1 から値が返ります。

●inbufleng

セグメントを受信する領域の長さを設定します。

●watchtime

dc_mcf_sendrecv 関数を呼び出してから終了するまでの最大時間を設定します。

0 を設定した場合，MCF マネジャ定義の UAP 共通定義で指定した同期型送受信監視時間（mcfmuap -t sndrcvtim）が設定されます。負の値を設定した場合は，時間を監視しません。

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

OpenTP1 から値が返される引数

●recvdata

受信したセグメントの内容が返されます。

●rdataleng

受信したセグメントの長さが返されます。

●time

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

リターン値

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

リターン値	リターン値 (数値)	意味
DCMCFRTN_72036	-13036	inbufleng の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
DCMCFRTN_72041	-13041	＜ 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_72073	-13073	非同期型のメッセージの送信処理が仕掛り中です。
DCMCFRTN_73001	-14001	watchtime に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
		出力キューからのメッセージの読み込み時に障害が発生しました。
		相手システムから否定応答（-RSP）を受信しました。
		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_73005	-14005	watchtime に設定した時間が経過しましたが，論理端末からの応答がありません。
		応答ダミーデータ（空白メッセージ）を受信しました。
DCMCFRTN_73008	-14008	論理端末が閉塞中，または MCF が終了処理中に，dc_mcf_sendrecv 関数を呼び出しました。
DCMCFRTN_73010	-14010	入力または出力メッセージ編集 UOC で障害が発生しました。
		メッセージの読み込み時に障害が発生しました。
		メッセージの編集エラーが発生しました。
DCMCFRTN_73015	-14015	出力先の論理端末は，ほかの UAP で仕掛り中です。
DCMCFRTN_73018	-14018	watchtime に設定した値が間違っています。
DCMCFRTN_73020	-14020	出力先の論理端末は停止しています。
上記以外	－	プログラムの破壊などによる，予期しないエラーが発生しました。

（凡例）

－：該当しません。

dc_mcf_tactcn － コネクションの確立（C 言語）

形式

ANSI C，C++の形式

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

K&R 版 C の形式

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

機能

コネクションを確立します。

なお，dc_mcf_tactcn 関数の正常終了は，コネクション確立要求を SLU - TypeP2 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。
DCMCFRTN_72063	-13063	dcmcf_tactcnopt の idnam の先頭がヌル文字です。

リターン値	リターン値 (数値)	意味
DCMCFRTN_72064	-13064	dcmcf_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 SLU - TypeP2 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 SLU - TypeP2 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。
DCMCFRTN_72063	-13063	dcmcf_tdctcnopt の idnam の先頭がヌル文字です。

リターン値	リターン値 (数値)	意味
DCMCFRTN_72064	-13064	dcmcf_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 SLU - TypeP2 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。

  SL2

  SLUTYPE-P プロトコル（2 次局）

- 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。
DCMCFRTN_71005	-12005	通信障害が発生しました。原因については，メッセージログファイルを参照してください。

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

4 COBOL-UAP 作成用プログラムインタフェース

この章では，SLU - TypeP2 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

SLU - TypeP2 で使用する COBOL-UAP 作成用プログラムインタフェースについて，COBOL 言語，およびデータ操作言語に分けて説明します。

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

COBOL 言語のプログラムインタフェース

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

COBOL 言語のプログラムインタフェースの一覧を，次の表に示します。

表 4-1　COBOL 言語のプログラムインタフェースの一覧

プログラム名	データ名 A に設定する要求コード	機能
CBLDCMCF	'RECEIVE△'	一方送信メッセージの受信
	'RECVSYNC'	同期型の応答メッセージの受信
	'RESEND△△'	メッセージの再送
	'SEND△△△△'	一方送信メッセージの送信
	'SENDRECV'	同期型の問い合わせ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。

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

SLU - TypeP2 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.
STATUS KEY	B	B	B	B	B
SYMBOLIC TERMINAL	B	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_1
SWITCHING MODE	−	−	U_2	U_2	−
DETAIL MODE	−	−	U_2	U_2	−
WAITING TIME	−	−	−	−	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.：単一セグメントの同期送受信（SEND）

B：OpenTP1 から値が返されます。省略できます。

U_1：UAP で値を設定します。省略できません。

U_2：UAP で値を設定します。省略できます。

－：該当しません。設定しても無効です。

注※

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※

SLU - TypeP2 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	I-O	SYNC，または'1'	ASYNC，'0'，'△'，または省略		
○	－	－	○	－	dc_mcf_send
－	○	－	○	－	dc_mcf_reply※1，※2
－	○	○	－	－	dc_mcf_sendsync※1
－	○	○	－	○	dc_mcf_sendrecv

（凡例）

○：指定あり

－：指定なし

注※1

SLU - TypeP2 ではサポートしていない関数です。

注※2

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

CBLDCMCF('RECEIVE ') － 一方送信メッセージの受信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

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

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE ’RECEIVE ’.
   02 データ名B   PIC X(5).
   02 FILLER     PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP.
   02 データ名H   PIC X(4)  VALUE SPACE.
   02 データ名I   PIC X(4)  VALUE SPACE.
   02 データ名J   PIC X(4)  VALUE SPACE.
   02 データ名K   PIC X(4)  VALUE SPACE.
   02 データ名L   PIC X(8)  VALUE SPACE.
   02 データ名M1  PIC X(4)  VALUE SPACE.
   02 データ名M2  PIC X(8)  VALUE SPACE.
   02 データ名M3  PIC X(4)  VALUE SPACE.
   02 データ名M4  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M6  PIC X(1)  VALUE SPACE.
   02 データ名M7  PIC X(1).
   02 データ名N   PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O   PIC X(4)  VALUE SPACE.
   02 データ名P   PIC X(8).
   02 データ名Q   PIC X(8).
   02 データ名R   PIC X(8)  VALUE SPACE.
   02 データ名T   PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名U   PIC 9(x)  COMP.
   02 データ名V   PIC X(x).
   02 データ名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，32767 バイトです。

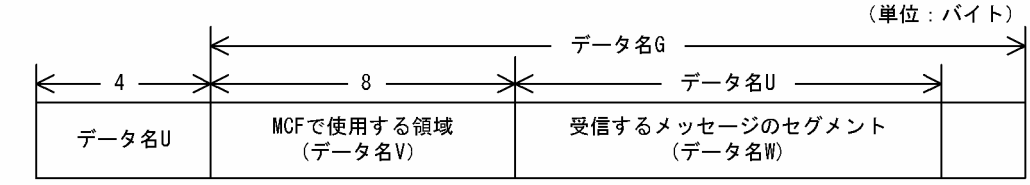
CBLDCMCF('RECEIVE△')で受信できるメッセージの種類を次に示します。

- 相手システムから送信されたメッセージ

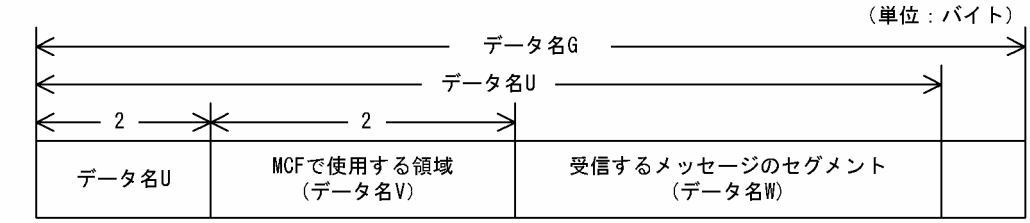
- MCF イベント
- アプリケーション起動で渡されたメッセージ

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 から値が返されます。

●データ名 Q

MCF で使用する領域です。

●データ名 R

空白を設定します。

●データ名 T

MCF で使用する領域です。

●データ名 V

【バッファ形式 1 の場合】 `PIC X(8)`

【バッファ形式 2 の場合】 PIC X(2)

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

●データ名 E

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

●データ名 F

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

●データ名 P

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

中間セグメントまたは最終セグメントを受信する場合，ここで返された論理端末名称をデータ名 P に設定してください。

●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の内容が返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71000	先頭セグメントを受信する CBLDCMCF('RECEIVE△')を 2 回以上呼び出しています。中間セグメントまたは最終セグメントを受信する場合は，データ名 C に「VALUE 'SEG△'」を設定して CBLDCMCF('RECEIVE△')を呼び出してください。
71001	メッセージの最終セグメントを受信したあとで，次のセグメントを受信する CBLDCMCF('RECEIVE△')を呼び出しています。直前に呼び出した CBLDCMCF('RECEIVE△')でメッセージはすべて受信しました。こ

ステータスコード	意味
71001	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。先頭セグメントを呼び出す場合は，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') －同期型の応答メッセージの受信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

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

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE ’RECVSYNC’.
   02 データ名B   PIC X(5).
   02 FILLER     PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP.
   02 データ名H   PIC X(4)  VALUE SPACE.
   02 データ名I   PIC X(4)  VALUE SPACE.
   02 データ名J   PIC X(4)  VALUE SPACE.
   02 データ名K   PIC X(4)  VALUE SPACE.
   02 データ名L   PIC X(8)  VALUE SPACE.
   02 データ名M1  PIC X(4)  VALUE SPACE.
   02 データ名M2  PIC X(8)  VALUE SPACE.
   02 データ名M3  PIC X(4)  VALUE SPACE.
   02 データ名M4  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5  PIC 9(9)  COMP 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).
```

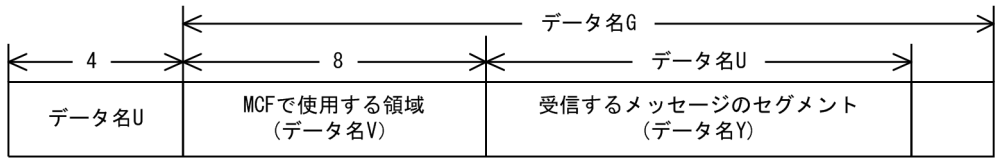
機能

相手システムから届いた同期型の受信メッセージのうち，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 CBLDBMCF('SENDRECV')のあとに，後続するセグメントの数だけ CBLDCMCF('RECVSYNC')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，32767 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　　（単位：バイト）



●バッファ形式２の場合　　（単位：バイト）

UAP で値を設定するデータ領域

●データ名 A

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

●データ名 C

メッセージの後続セグメントを受信する「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 を使用することを設定します。

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 9(4)

受信したセグメントの長さ＋ 4 が返されます。

●データ名 Y

受信したセグメントの内容が返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71001	メッセージの最終セグメントを受信したあとで，次のセグメントを受信する CBLDCMCF('RECVSYNC') を呼び出しています。直前に呼び出した CBLDCMCF('RECVSYNC')でメッセージはすべて受信しました。 このステータスコードが返されたあとに，再び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。
71108	メッセージ受信に必要な管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('RECVSYNC')を呼び出しています。
	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 Q に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72025	データ名 C に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。

ステータスコード	意味
72036	データ名 G の指定値が不足しています。バッファ形式 1 の場合は 9 バイト以上，バッファ形式 2 の場合は 5 バイト以上の領域を確保してください。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

CBLDCMCF('RESEND ') － メッセージの再送（COBOL 言語）

形式

PROCEDURE DIVISION の指定

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

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE 'RESEND  '.
   02 データ名B   PIC X(5).
   02 FILLER     PIC X(3).
   02 データ名C   PIC X(4)  VALUE SPACE.
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP VALUE ZERO.
   02 データ名H   PIC X(4)  VALUE SPACE.
   02 データ名I   PIC X(4)  VALUE SPACE.
   02 データ名J   PIC X(4).
   02 データ名K   PIC X(4).
   02 データ名L   PIC X(8)  VALUE SPACE.
   02 データ名M1  PIC X(4)  VALUE SPACE.
   02 データ名M2  PIC X(8)  VALUE SPACE.
   02 データ名M3  PIC X(4)  VALUE SPACE.
   02 データ名M4  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M6  PIC X(1)  VALUE SPACE.
   02 データ名M7  PIC X(1)  VALUE SPACE.
   02 データ名N   PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O   PIC X(4)  VALUE 'OUT '.
   02 データ名P   PIC X(8).
   02 データ名Q   PIC X(8)  VALUE SPACE.
   02 データ名R   PIC X(8)  VALUE SPACE.
   02 データ名S   PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名T   PIC X(8).
   02 データ名U   PIC X(4).
   02 データ名V   PIC 9(9)  COMP.
   02 データ名W   PIC X(4).
   02 データ名X   PIC X(12) VALUE LOW-VALUE.
```

機能

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

- 出力先の論理端末名称

- メッセージ出力通番
- メッセージ種別（一般の一方送信，優先の一方送信）

対象としたメッセージが以前に送信されていない場合は，CBLDCMCF('RESEND△△')はステータスコード 70904 を返します。また，メッセージキュー（ディスクキュー）内に対象のメッセージがない場合もステータスコード 70904 を返します。このため，使用するメッセージキューの種別ではディスクキューを指定するとともに，メッセージキューの大きさの定義は余裕を持った値を指定してください。

UAP で値を設定するデータ領域

●データ名 A

メッセージの再送を示す要求コード「VALUE 'RESEND△△'」を設定します。

●データ名 C，データ名 D

空白を設定します。

●データ名 E，データ名 F

MCF で使用する領域です。

●データ名 G

0 を設定します。

●データ名 H，データ名 I

空白を設定します。

●データ名 J

一般として再送するか優先として再送するかを設定します。

VALUE 'NORM'

一般の一方送信メッセージとして再送する場合に設定します。

VALUE 'PRIO'

優先の一方送信メッセージとして再送する場合に設定します。

空白

省略されたものとして，「VALUE 'NORM'」（一般の一方送信メッセージとして再送）が設定されます。

●データ名 K

再送するメッセージに出力通番を付け直すかどうかを設定します。

VALUE 'SEQ△'

再送するメッセージに出力通番を付け直す場合に設定します。

VALUE 'NSEQ'

再送するメッセージに出力通番を付け直さない場合に設定します。

空白

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

●データ名 L，データ名 M1，データ名 M2，データ名 M3

空白を設定します。

●データ名 M4，データ名 M5

0 を設定します。

●データ名 M6，データ名 M7

空白を設定します。

●データ名 N

MCF で使用する領域です。

●データ名 O

一方送信を示す「VALUE 'OUT△'」を設定します。

●データ名 P

出力先の論理端末名称を設定します。論理端末名称は最大８バイトの長さです。８バイトに満たない場合，論理端末名称の後ろを空白で埋めてください。

●データ名 Q，データ名 R

空白を設定します。

●データ名 S

MCF で使用する領域です。

●データ名 T

再送するメッセージを検索するキーとして，以前に送信したメッセージの出力先の論理端末名称を設定します。論理端末名称は最大８バイトの長さです。８バイトに満たない場合，論理端末名称の後ろを空白で埋めてください。

●データ名 U

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

VALUE 'NORM'

一般の一方送信メッセージを対象とする場合に設定します。

VALUE 'PRIO'

優先の一方送信メッセージを対象とする場合に設定します。

空白

省略されたものとして，「VALUE 'NORM'」（一般の一方送信メッセージを対象）が設定されます。

VALUE 'PRIO'を設定した場合は，データ名 T に出力先の論理端末名称を設定できません。

●データ名 V

再送するメッセージを検索するキーとして，以前に送信したメッセージの出力通番を設定します。データ名 W に「VALUE 'LAST'」を設定した場合は，ここに設定した値は無効となります。

●データ名 W

最終出力通番を持つメッセージを再送するかどうかを設定します。

VALUE 'LAST'

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

空白

データ名 V で設定した出力通番を持つメッセージを再送する場合に設定します。

●データ名 X

MCF で使用する領域です。

OpenTP1 から値が返されるデータ領域

●データ名 B

ステータスコードが，5 けたの数字で返されます。

ステータスコード

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

ステータスコード	意味
70904	• 送信済みのメッセージがディスクキューに保持されていません（保持メッセージ数はメッセージキューサービス定義の quegrp コマンドの-m オプションで指定します）。
	データ名 T で指定した論理端末に割り当てられているメッセージキューは，システム開始時に障害が発生したため，メモリキューを代用して縮退運転をしています。
70905	再送するメッセージのセグメントの長さが，UAP 共通定義（mcfmuap -e）で指定した値を超えています。
71002	メッセージキューへの出力処理中に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	MCF が終了処理中のため，メッセージの再送を受け付けられません。
71003	メッセージキューが満杯です。
71004	メッセージキューから取り出したメッセージを格納するバッファ（作業領域）をメモリ上に確保できませんでした。
71108	メッセージを再送しようとしましたが，再送先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	＜ MHP の実行でリターンした場合＞ • 先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('RESEND△△')を呼び出しています。 • 非トランザクション属性の MHP から，CBLDCMCF('RESEND△△')を呼び出しています。
	＜ SPP の実行でリターンした場合＞ トランザクションでない SPP の処理から，CBLDCMCF('RESEND△△')を呼び出しています。
72001	データ名 P またはデータ名 T に設定した論理端末名称が間違っています。
	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，またはデータ名 X に設定した値が間違っています。
72017	データ名 K に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

注意事項

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

-e オプション

-e オプションでは，CBLDCMCF('RESEND△△')で使用する作業領域の大きさを指定します。再送するメッセージのセグメントがこの作業領域より大きい場合，CBLDCMCF('RESEND△△')はメッセージを再送しないで，ステータスコード 70905 を返します。このため，-e オプションでは，セグメントの最大長よりも大きな値を設定しておいてください。

-l オプション

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

CBLDCMCF('SEND ') － 一方送信メッセージの送信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

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

DATA DIVISION の指定

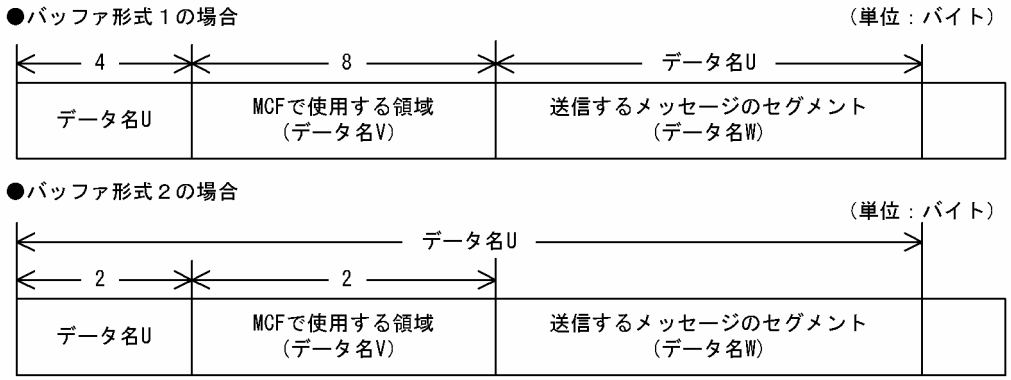
```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE 'SEND    '.
   02 データ名B   PIC X(5).
   02 FILLER    PIC X(3).
   02 データ名C   PIC X(4)  VALUE SPACE.
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP VALUE ZERO.
   02 データ名H   PIC X(4).
   02 デー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

機能

相手システムへ送る一方送信メッセージのうち，一つのセグメントを送信します。セグメントの数だけCBLDCMCF('SEND△△△△')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（一意名 3 で示す領域）の形式を示します。



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。次のどちらかを設定します。

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。

VALUE 'NORM'

一般の一方送信メッセージとして送信する場合に設定します。

VALUE 'PRIO'

優先の一方送信メッセージとして送信する場合に設定します。

空白

省略されたものとして，「VALUE 'NORM'」（一般の一方送信メッセージとして送信）が設定されます。

●データ名 K

出力通番を付けるかどうかを設定します。

VALUE 'SEQ△'

出力通番を付ける場合に設定します。

VALUE 'NSEQ'

出力通番を付けない場合に設定します。

空白

省略されたものとして，「VALUE 'NSEQ'」（出力通番を付けない）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。

ステータスコード	意味
71002	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	バッファ形式 1 の場合はデータ名 U に 32000 バイトを超える値を設定しています。バッファ形式 2 の場合はデータ名 U に 32004 バイトを超える値を設定しています。
	MCF が終了処理中のため，メッセージの送信を受け付けられません。
71003	メッセージキューが満杯です。
71004	メッセージを格納するバッファをメモリ上に確保できませんでした。
71108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	＜ MHP の実行でリターンした場合＞ 先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('SEND△△△△')を呼び出しています。
	＜ SPP の実行でリターンした場合＞ トランザクションでない SPP の処理から，CBLDCMCF('SEND△△△△')を呼び出しています。
72001	データ名 P に設定した論理端末名称が間違っています。
	データ名 P に設定した論理端末名称は，定義されていません。
	CBLDCMCF('SEND△△△△')を呼び出せない論理端末を設定しています。
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 N またはデータ名 T に設定した値が間違っています。
	データ名 J に設定した値が間違っています。
	データ名 M1 に設定した値が間違っています。
	データ名 M7 に設定した値が間違っています。
72017	データ名 K に設定した値が間違っています。
72019	データ名 M6 に設定した値が間違っています。
72020	データ名 I に設定した値が間違っています。
72024	データ名 O に設定した値が間違っています。
72026	データ名 H に設定した値が間違っています。
72028	データ名 A に設定した値が間違っています。
72041	＜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。

ステータスコード	意味
72041	• 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') － 同期型の問い合わせメッセージの送受信（COBOL 言語）

形式

PROCEDURE DIVISION の指定

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

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)  VALUE ’SENDRECV’.
   02 データ名B   PIC X(5).
   02 FILLER     PIC X(3).
   02 データ名C   PIC X(4)  VALUE SPACE.
   02 データ名D   PIC X(4)  VALUE SPACE.
   02 データ名E   PIC 9(8).
   02 データ名F   PIC 9(8).
   02 データ名G   PIC 9(9)  COMP.
   02 データ名H   PIC X(4).
   02 データ名I   PIC X(4)  VALUE SPACE.
   02 データ名J   PIC X(4)  VALUE SPACE.
   02 データ名K   PIC X(4)  VALUE SPACE.
   02 データ名L   PIC X(8)  VALUE SPACE.
   02 データ名M1  PIC X(4)  VALUE SPACE.
   02 データ名M2  PIC X(8)  VALUE SPACE.
   02 データ名M3  PIC X(4)  VALUE SPACE.
   02 データ名M4  PIC 9(9)  COMP VALUE ZERO.
   02 データ名M5  PIC 9(9)  COMP※.
   02 データ名M6  PIC X(1)  VALUE SPACE.
   02 データ名M7  PIC X(1).
   02 データ名N   PIC X(14) VALUE LOW-VALUE.
01 一意名2.
   02 データ名O   PIC X(4)  VALUE ’IO  ’.
   02 データ名P   PIC X(8).
   02 データ名Q   PIC X(8)  VALUE SPACE.
   02 データ名R   PIC X(8)  VALUE SPACE.
   02 データ名T   PIC X(28) VALUE LOW-VALUE.
01 一意名3.
   02 データ名U   PIC 9(x)  COMP.
   02 データ名V   PIC X(x).
   02 データ名W   PIC X(n).
01 一意名4.
   02 データ名X   PIC 9(x)  COMP.
   02 データ名Y1  PIC X(x)  VALUE SPACE.
   02 データ名Y2  PIC X(1).
   02 データ名Z   PIC X(n).
```

注※

負の値を設定する場合，「PIC S9(9) COMP」としてください。

機能

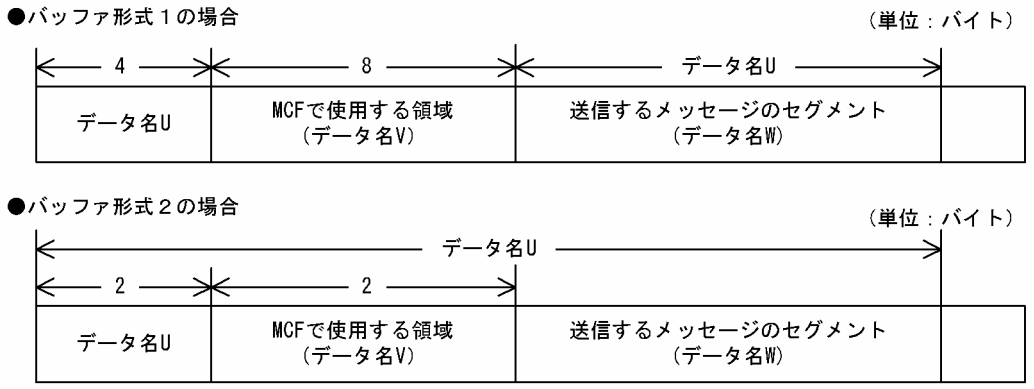
同期型でメッセージを送信したあと，同期型でメッセージを受信します。

相手システムへ送るメッセージのうち，一つのセグメントを送信します。セグメントの数だけCBLDCMCF('SENDRECV')を呼び出すと，一つの論理メッセージを送信できます。

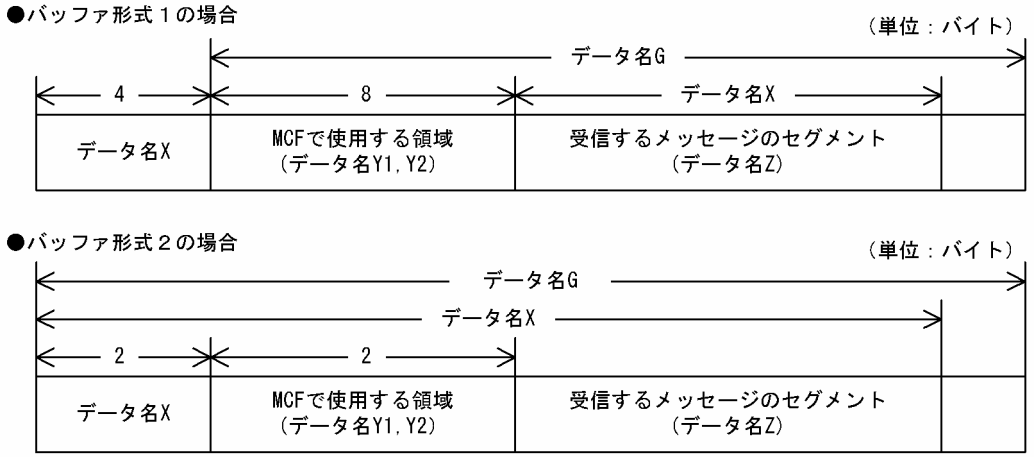
メッセージの最終セグメントを送信すると，CBLDCMCF('SENDRECV')は相手システムからの応答を待ちます。応答が届くと，そのメッセージの先頭セグメントを受信します。中間セグメントまたは最終セグメントを受信する場合は，CBLDCMCF('RECVSYNC')を呼び出してください。

受信できるメッセージの一つのセグメントの最大長は，32000 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）の形式を示します。



セグメントを受信する領域（一意名 4 で示す領域）の形式を示します。



UAP で値を設定するデータ領域

●データ名 A

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

●データ名 C，データ名 D

空白を設定します。

●データ名 G

セグメントを受信する領域の長さを設定します。

●データ名 H

送信するセグメントを設定します。次のどちらか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') を呼び出すと，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

0 を設定します。

●データ名 M5

CBLDCMCF('SENDRECV')を呼び出してから終了するまでの最大時間を設定します。

0 を設定した場合，MCF マネジャ定義の UAP 共通定義で指定した同期型送受信監視時間（mcfmuap -t sndrcvtim）が設定されます。負の値を設定した場合は，時間を監視しません。

注意事項

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

●データ名 M6

空白を指定します。

●データ名 M7

使用するバッファ形式を設定します。

VALUE '1'

バッファ形式 1 を使用する場合に設定します。

VALUE '2'

バッファ形式 2 を使用する場合に設定します。

空白

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

●データ名 N

MCF で使用する領域です。

●データ名 O

同期型のメッセージの送受信を示す「VALUE 'IO△△'」を設定します。

●データ名 P

メッセージを出力して応答を入力する論理端末名称を設定します。論理端末名称は最大 8 バイトの長さです。8 バイトに満たない場合，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 9(4)

受信したセグメントの長さ＋ 4 が返されます。

●データ名 Z

受信したセグメントの内容が返されます。

ステータスコード

ステータスコード	意味
00000	正常に終了しました。
71002	メッセージキューへの入出力処理時に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。

ステータスコード	意味
71002	バッファ形式 1 の場合はデータ名 U に 32000 バイトを超える値を設定しています。バッファ形式 2 の場合はデータ名 U に 32004 バイトを超える値を設定しています。
	MCF が終了処理中のため，メッセージの送受信を受け付けられません。
71003	メッセージキューが満杯です。
71004	メッセージを格納するバッファをメモリ上に確保できませんでした。
71108	メッセージを送信しようとしましたが，送信先の管理テーブルが確保できませんでした。
	プロセスのローカルメモリが不足しています。
72000	先頭セグメントを受信する CBLDCMCF('RECEIVE△')を呼び出す前に，CBLDCMCF('SENDRECV')を呼び出しています。
72001	データ名 P に設定した論理端末名称が間違っています。
	データ名 P に設定した論理端末名称は，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。
72073	非同期型のメッセージの送信処理が仕掛り中です。

ステータスコード	意味
73001	データ名 M5 に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
	出力キューからのメッセージの読み込み時に障害が発生しました。
	相手システムから否定応答（-RSP）を受信しました。
	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。
73005	データ名 M5 に設定した時間が経過しましたが，論理端末からの応答がありません。
	応答ダミーデータ（空白メッセージ）を受信しました。
73008	論理端末が閉塞中，または MCF が終了処理中に，CBLDCMCF('SENDRECV')を呼び出しました。
73010	入力または出力メッセージ編集 UOC で障害が発生しました。
	メッセージの読み込み時に障害が発生しました。
	メッセージの編集エラーが発生しました。
73015	出力先の論理端末は，ほかの UAP で仕掛り中です。
73018	データ名 M5 に設定した値が間違っています。
73020	出力先の論理端末は停止しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

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

形式

PROCEDURE DIVISION の指定

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

DATA DIVISION の指定

```
01 一意名1.
   02 データ名A   PIC X(8)   VALUE ’TACTCN  ’.
   02 データ名B   PIC X(5).
   02 FILLER      PIC X(3).
   02 データ名C   PIC X(4).
   02 データ名D1  PIC X(1)   VALUE SPACE.
   02 データ名D2  PIC X(1)   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 SLU - TypeP2 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 SLU - TypeP2 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。
71010	MCF 通信プロセスに論理端末の閉塞の解除を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。

ステータスコード	意味
71011	論理端末が削除されているため，CBLDCMCF('TACTLE△△')が受け付けられません。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D1，データ名 D2，またはデータ名 D3 に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。

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

形式

PROCEDURE DIVISION の指定

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

DATA DIVISION の指定

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

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

機能

コネクションを解放します。

なお，CBLDCMCF('TDCTCN△△')の正常終了は，コネクション解放要求を SLU - TypeP2 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。
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。

ステータスコード	意味
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 SLU - TypeP2 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。
71010	MCF 通信プロセスに論理端末の閉塞を要求しましたが，受け付けられませんでした。原因については，メッセージログファイルを参照してください。

ステータスコード	意味
71011	論理端末が削除されているため，CBLDCMCF('TDCTLE△△')が受け付けられません。
72028	データ名 A に設定した値が間違っています。
72052	データ名 K に 0 でない値が設定されています。
72053	データ名 L に 0 でない値が設定されています。
72058	データ名 C に空白でない値が設定されています。
72059	データ名 D2，またはデータ名 D3 に空白でない値が設定されています。
72061	データ名 E に 0 未満または 240 以上の値が設定されています。
72063	データ名 F1 に空白が設定されています。
72065	データ名 F2 に空白でない値が設定されています。
72066	データ名 G に空白でない値が設定されています。
72068	データ名 H に空白でない値が設定されています。
72070	データ名 I に空白でない値が設定されています。
72072	データ名 J に空白でない値が設定されています。
72074	データ名 F1 に設定された文字列中に不正な文字があります。
72075	データ名 D1 に空白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 'SL2△'

SLUTYPE-P プロトコル（2 次局）

●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。
71007	指定されたコネクション ID は登録されていません。

ステータスコード	意味
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。
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

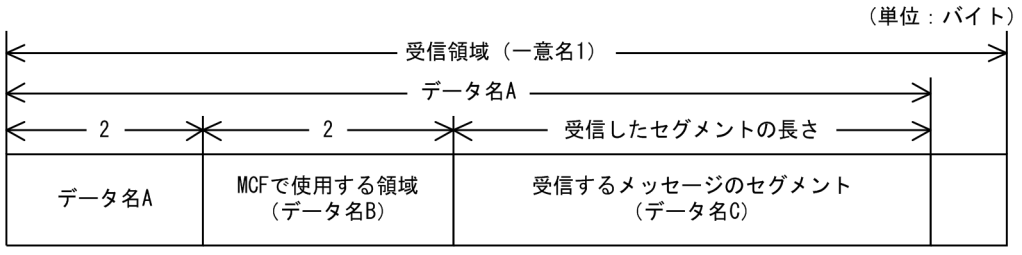
```
RECEIVE    通信記述名
    〔FIRST〕  SEGMENT
         INTO  一意名1.
```

機能

次に示す CALL インタフェースの機能を実現します。

- 一方送信メッセージの受信　CBLDCMCF('RECEIVE△')

セグメントを受信する領域（一意名 1 で示す領域）の形式を次に示します。



UAP で値を設定する項目

●FOR 句

次のどちらかの値を設定します。

INPUT または I-O

一方送信メッセージの受信

●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

非同期型のメッセージの受信を示す，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 回以上実行しています。中間セグメントまたは最終セグメントを受信する場合は，FIRST を設定しないで RECEIVE 文を実行してください。
71001	メッセージの最終セグメントを受信したあとで，次のセグメントを受信する RECEIVE 文を実行しています。直前に実行した RECEIVE 文でメッセージはすべて受信しました。 このステータスコードが返されたあとに，再び次のメッセージを受信する RECEIVE 文を実行した場合は，ステータスコード 72000 が返されます。
71002	メッセージキューからの入力処理中に障害が発生しました。
	メッセージキューが閉塞されています。
	メッセージキューが割り当てられていません。
	MCF が終了処理中のため，メッセージの受信を受け付けられません。
71108	メッセージの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 文を実行する前に，中間セグメントまたは最終セグメントを受信する RECEIVE 文を実行しています。先頭セグメントを受信する場合は，FIRST を設定して RECEIVE 文を実行してください。 • ステータスコード 71001 が返されたあとで，再び非同期型のメッセージを受信する RECEIVE 文を実行しています。
	＜ SPP の実行でリターンした場合＞

ステータスコード	意味
72000	SPP では非同期型のメッセージを受信する RECEIVE 文を実行できません。
72001	SYMBOLIC TERMINAL 句に設定した論理端末名称が間違っています。
	SYMBOLIC TERMINAL 句に設定した論理端末名称は，定義されていません。
	RECEIVE 文を実行できない論理端末を設定しています。
	SYNCHRONOUS MODE 句に SYNC が設定されているか，データ名 6 に'1'が設定されています。
72013	一意名 1 のサイズを超えるセグメントを受信しました。一意名 1 のサイズを超え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|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〕
    〔WAITING  TIME  IS  データ名11〕.
```

DATA DIVISION（データ記述項）の指定

```
01  一意名1.
    02  データ名A   PIC 9(4)  COMP.
    02  データ名B   PIC X(2).
    02  データ名C   PIC X(n).
01  一意名3.
    02  データ名D   PIC 9(4)  COMP.
    02  データ名E   PIC X(2).
    02  データ名F   PIC X(n).
```

PROCEDURE DIVISION（通信文）の指定

```
SEND  通信記述名  FROM  一意名1
     〔WITH  {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|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

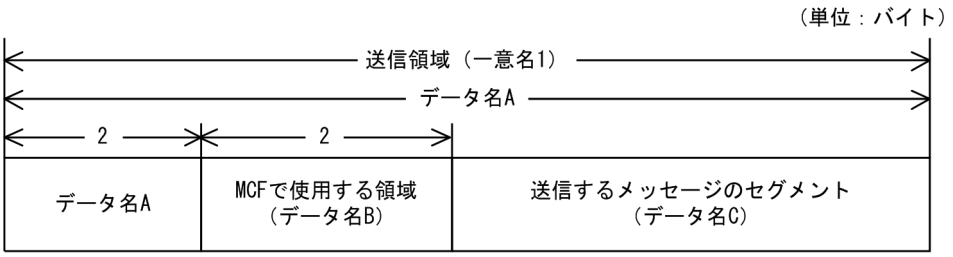
次に示す CALL インタフェースの機能を実現します。

- 一方送信メッセージの送信　CBLDCMCF('SEND△△△△')

- 同期型の問い合わせメッセージの送受信　CBLDCMCF('SENDRECV')

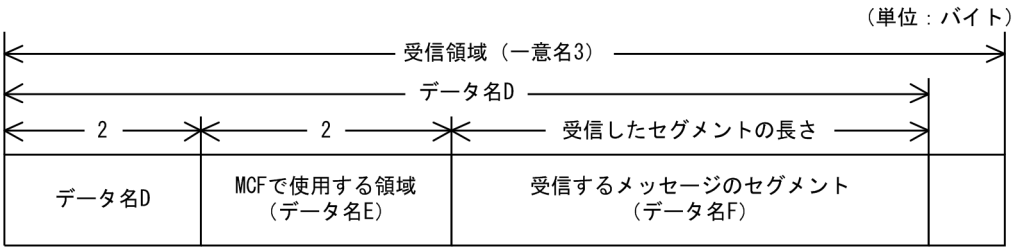
同期型の問い合わせメッセージの送受信では，単一セグメントの論理メッセージだけを扱えます。

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

次のどちらかの値を設定します。

OUTPUT

一方送信メッセージの送信

I-O

同期型のメッセージの送受信または一方送信メッセージの送信※

注※

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

●SYMBOLIC TERMINAL 句

論理端末名称を設定したデータ項目を指定します。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

同期型の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

非同期型のメッセージの送信

一方送信メッセージの送信のとき設定します。

データ名 6

次の値を設定したデータ項目

'0'または'△'：非同期型のメッセージの送信

'1'：同期型の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

一方送信メッセージの場合に，一般か優先かを設定します。

NORMAL

一般の一方送信メッセージ

PRIOR

優先の一方送信メッセージ

データ名 7

次の値を設定したデータ項目

'0'または'△'：一般の一方送信メッセージ

'1'：優先の一方送信メッセージ

省略した場合および FOR 句に I-O を設定した場合は，NORMAL（一般の一方送信メッセージ）が設定されます。

●DETAIL MODE 句

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

データ名 10

次の値を設定したデータ項目

'0'または'△'：出力通番を付けます。

'1'：出力通番を付けません。

省略した場合および FOR 句に I-O を設定した場合は，出力通番を付けません。

●WAITING TIME 句

SEND 文を実行してから終了するまでの，最大時間を設定したデータ項目を設定します。同期型のメッセージの送受信の場合に設定します。

データ名 11

監視時間の値を HHMMSS00（HH：時　MM：分　SS：秒　00 は固定）の形式で設定したデータ項目を設定します。

省略した場合，またはデータ名 11 に'00000000'を設定した場合，MCF マネジャ定義の UAP 共通定義で指定した同期型送受信監視時間（mcfmuap -t sndrcvtim）が設定されます。

注意事項

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

●データ名 A

送信するセグメントの長さ＋ 4 を設定します。

●データ名 B

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）

'2'：EMI（最終セグメントまたは単一セグメント）

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，同期型のメッセージの送受信（SENDRECV）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

同期型のメッセージの送受信の場合に，セグメントを受信するデータ項目（一意名 3）を設定します。一方送信メッセージの送信の場合，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が確保できませんでした。
	プロセスのローカルメモリが不足しています。

ステータスコード	意味
72000	＜MHP の実行でリターンした場合＞ • 先頭セグメントを受信する RECEIVE 文を実行する前に，SEND 文を実行しています。 • FOR 句に I-O を設定した一方送信メッセージを送信する SEND 文を実行していますが，UAP 共通定義（mcfmuap -c）の noansreply オペランドに no を指定しています。
	＜SPP の実行でリターンした場合＞ • トランザクションでない SPP の処理から，一方送信メッセージを送信する SEND 文を実行しています。 • SPP では，応答メッセージを送信する SEND 文を実行できません。
72001	SYMBOLIC TERMINAL 句に設定した論理端末名称が間違っています。
	SYMBOLIC TERMINAL 句に設定した論理端末名称は，定義されていません。
	SEND 文を実行できない論理端末を設定しています。
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。
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。
72037	BEFORE 句に設定した値が間違っています。
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。
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。
72073	非同期型のメッセージの送信処理で仕掛り中です。
73001	データ名 11 に 60 秒を加算した時間が経過しましたが，MCF 通信プロセスからの応答がありません。
	出力キューからのメッセージの読み込み時に障害が発生しました。
	相手システムから否定応答（-RSP）を受信しました。

ステータスコード	意味
73001	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。
73005	データ名 11 に設定した時間が経過しましたが，論理端末からの応答がありません。
73008	論理端末が閉塞中，または MCF が終了処理中に，SEND 文を実行しました。
73010	入力または出力メッセージ編集 UOC で障害が発生しました。
	メッセージの読み込み時に障害が発生しました。
	メッセージの編集エラーが発生しました。
73015	出力先の論理端末は，ほかの UAP で仕掛り中です。
73018	WAITING TIME 句に設定した値が間違っています。
73020	出力先の論理端末は停止しています。
上記以外	プログラムの破壊などによる，予期しないエラーが発生しました。

5 ユーザオウンコーディング，MCF イベントインタフェース

この章では，SLU - TypeP2 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**（以降，**UOC** と略します）といいます。

SLU - TypeP2 で使用する UOC を次に示します。

- 入力メッセージ編集 UOC
- 出力メッセージ編集 UOC
- 送信メッセージの通番編集 UOC

UOC を使用する場合は，あらかじめ MCF メイン関数または UAP のメイン関数に UOC 関数のアドレスを登録し，UOC 関数のオブジェクトファイルを MCF 通信プロセスまたは UAP の実行形式プログラムに結合（リンケージ）しておく必要があります。また，UOC は C 言語で作成します。

5.1.1　入力メッセージの編集とアプリケーション名の決定

入力メッセージ編集 UOC は，受信した論理メッセージをユーザ任意の形式に変換します。次に，受信した論理メッセージを基に，ユーザ任意のアプリケーション名を決定できます。

UOC は，UAP を起動するメッセージの最終セグメントを受信すると起動します。ただし，MCF イベント発生時と，UAP からのアプリケーションプログラム起動時は，UOC は起動しません。

ユーザは，MCF メイン関数で UOC 関数アドレスを設定します。また，必要に応じてコネクション定義（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)　入力メッセージの編集

受信したメッセージが格納されている受信バッファ，および定義で指定した編集バッファを引き渡します。UOC では，これらのバッファを使用して入力メッセージの編集ができます。

また，UAP に通知するメッセージのセグメントは，受信バッファ，または編集バッファのどちらかに格納されたものを使用できます。どちらのセグメントを使用するかは，UOC から返されるリターンコードによって選択できます。

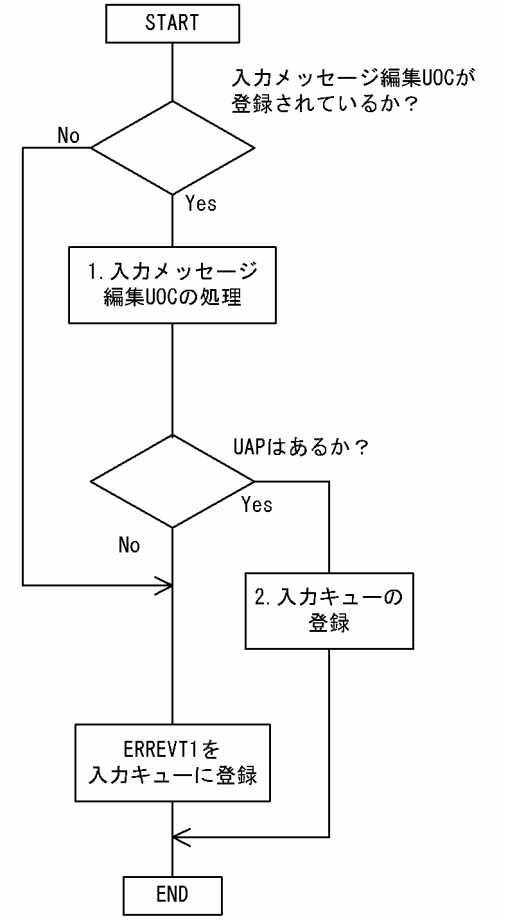
(2)　アプリケーション名の決定

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，論理メッセージの受信と同時にアプリケーション名を決定できます。

UOC でアプリケーション名を決定する場合，アプリケーション名の形式は，アプリケーション名格納領域の先頭から'¥0'の手前までの，1～8 バイトの英数字です。先頭から 9 バイト目までに'¥0'がないときは，アプリケーション名を不正とし，ERREVT1 を起動します。

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



1. 入力メッセージの中の情報からアプリケーション名を決定し，指定領域（aplname）にアプリケーション名を設定します。
2. 入力キューの登録後，正常処理へ続きます。

(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，SLU - TypeP2 は論理端末を閉塞し，障害通知イベント（CERREVT）を通知します。

UOC で障害を検出し，MCF イベント処理用 MHP を起動したい場合は，ユーザ任意の MCF イベント処理用 MHP のアプリケーション名を設定します。MCF には DCMCF_UOC_MSG_OK，または DCMCF_UOC_MSG_OK_RCV でリターンします。ただし，この場合は SLU - TypeP2 は正常なメッセージとして処理するため，UAP 側で障害処理をしてください。

(4)　UOC パラメタ不正の場合の処理

UOC で設定したパラメタに不正があった場合，SLU - TypeP2 は論理端末を閉塞し，障害通知イベント（CERREVT）を通知します。

(5) OpenTP1 への組み込み方法

スタート関数（dc_mcf_svstart）を発行する MCF メイン関数に，作成した UOC の関数アドレスを設定します。入力メッセージ編集 UOC の関数アドレスは任意に決められます。UOC 関数をコンパイルして生成した UOC オブジェクトファイルを，UOC 関数を登録した MCF メイン関数と結合して，SLU - TypeP2 の実行形式プログラムを生成します。MCF メイン関数の詳細については，「8.2 MCF メイン関数の作成」を参照してください。

5.1.2 入力メッセージ編集 UOC インタフェース

入力メッセージ編集 UOC は，次に示す形式で呼び出します。

(1) 形式

ANSI C，C++の場合

```
#include<dcmcfuoc.h>

DCLONG    uoc_func(dcmcf_uoc_min_n *parm)
```

K&R 版 C の場合

```
#include<dcmcfuoc.h>

DCLONG    uoc_func(parm)
dcmcf_uoc_min_n   *parm;
```

(2) 説明

uoc_func（入力メッセージ編集 UOC）を呼び出すとき，MCF は次に示す所定のパラメタを parm に設定します。

(3) パラメタの内容

(a) dcmcf_uoc_min_n の内容

```
typedef struct {
    DCLONG pro_kind;            … プロトコル種別
    char le_name[9];            … 論理端末名称
    char reserve1[7];           … 予備
    DCLONG rcv_prim;            … 受信サービスプリミティブ
    dcmcf_uocbuff_list_n *buflist_adr;
                                … 受信バッファリストアドレス
    dcmcf_uocbuff_list_n *ebuflist_adr;
                                … 編集バッファリストアドレス
    char aplname[9];            … アプリケーション名
    char reserve2[7];           … 予備
    char *pro_indv_ifa;         … MCF使用領域
```

```
    DCLONG rtn_detail;          … 詳細リターンコード
    char reserve3[8];           … 予備
}dcmcf_uoc_min_n;
```

(b) dcmcf_uocbuff_list_n（バッファリスト）の内容

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

(c) dcmcf_uocbufinf_n（バッファ情報）の内容

```
typedef struct {
    char *buf_adr;              … バッファアドレス
    DCULONG buf_size;           … バッファ最大長
    DCULONG seg_size;           … バッファ使用長
    char reserve1[4];           … 予備
    dcmcfuoc_w_type buff_id;    … MCF内部情報1
    DCMLONG buff_addr;          … MCF内部情報2
    char reserve2[4];           … 予備
}dcmcf_uocbufinf_n;
```

(4) MCF が値を設定する項目

(a) dcmcf_uoc_min_n

- pro_kind

  プロトコル種別として，次の値が設定されます。

  DCMCF_UOC_PRO_SLUP2

  SLUTYPE-P プロトコル（2 次局）

- 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受信

  DCMCF_UOC_RCV_REP_RE

  応答メッセージ受信

- buflist_adr

  受信用バッファリストのアドレスが設定されます。

- ebuflist_adr
  編集用バッファリストのアドレスが設定されます。
  メッセージ編集バッファが未定義の場合，つまり，コネクション定義（mcftalccn）の-e オプションを省略した場合，ebuflist_adr には NULL が設定されます。
- aplname
  論理端末定義（mcftalcle）の-v オプションで指定したアプリケーション名が設定されます。
- 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）

- 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，バッファの使用長が設定されます。
- buff_id，buff_addr
  MCF で使用するパラメタです。

(5) ユーザが値を設定する項目

(a) dcmcf_uoc_min_n

- aplname
  UOC で決定したアプリケーション名を設定します。
- rtn_detail
  詳細リターンコードを設定します。
  このコードは，UOC が DCMCF_UOC_MSG_NG をリターンしたときに，MCF に渡されます。MCF は，詳細リターンコードをメッセージログファイルに出力します。詳細リターンコードは，-19999～-19000 の範囲で設定してください。

(b) dcmcf_uocbuff_list_n（バッファリスト）

- used_buf_num
  使用したバッファ情報の数を設定します。

(c) dcmcf_uocbufinf_n（バッファ情報）

- seg_size
  バッファの使用長を設定します。

(6) リターン値

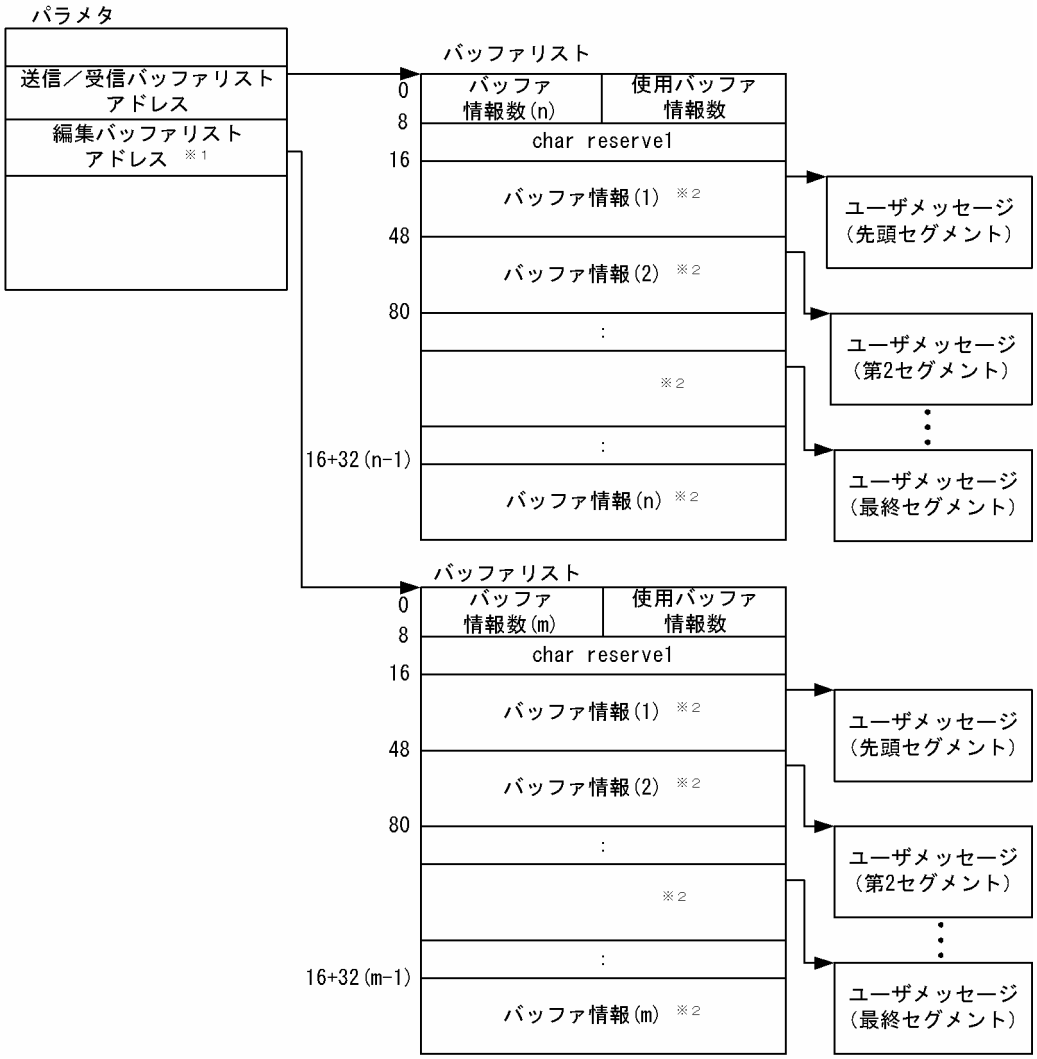
uoc_func()は次のコードでリターンしてください。

リターン値	意味
DCMCF_UOC_MSG_OK	正常リターン（編集バッファでスケジューリング）
DCMCF_UOC_MSG_OK_RCV	正常リターン（受信バッファでスケジューリング）
DCMCF_UOC_MSG_NG	メッセージ編集エラー

(7) パラメタとバッファの関係

UOC インタフェース用のパラメタとバッファの関係を次の図に示します。

図 5-2　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い場合，NULL となり，バッファリストとバッファは確保されません。

注※2

バッファ情報は次の形式をしています。

5.1.3　出力メッセージの編集

出力メッセージ編集 UOC は，応答メッセージまたは一方送信メッセージの編集をする 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実際に送信する前に処理するように位置させます。出力キューから全セグメントを読み出すと起動します。

ユーザは，MCF メイン関数で UOC 関数アドレスを設定します。また，必要に応じてコネクション定義（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引き渡します。UOC では，これらのバッファを使用して，出力メッセージの編集処理ができます。

また，UOC からのリターンコードによって UAP に通知するメッセージとして送信バッファに格納されたものを使用するか，編集バッファに格納されたものを使用するかを選択できます。

(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，SLU - TypeP2 は論理端末を閉塞します。なお，このとき MCF イベント（ERREVTn）は通知しません。

(3)　UOC パラメタ不正の場合の処理

UOC で設定した値に不正があった場合，MCF はメッセージログを出力し，障害通知イベント（CERREVT）を通知します。該当するメッセージは破棄します。

(4)　OpenTP1 への組み込み方法

入力メッセージ編集 UOC の組み込み方法と同じです。「5.1.1（5）　OpenTP1 への組み込み方法」を参照してください。

5.1.4　出力メッセージ編集 UOC インタフェース

出力メッセージ編集 UOC は，次に示す形式で呼び出します。

(1)　形式

ANSI C，C++の場合

```
#include<dcmcfuoc.h>

DCLONG    uoc_func(dcmcf_uoc_mout_n *parm)
```

K&R 版 C の場合

```
#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使用領域
    DCLONG rtn_detail;        … 詳細リターンコード
    char reserve3[20];        … 予備
}dcmcf_uoc_mout_n;
```

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

「5.1.2　入力メッセージ編集 UOC インタフェース」を参照してください。

(4) MCF が値を設定する項目

(a) dcmcf_uoc_mout_n

- pro_kind

  プロトコル種別として，次の値が設定されます。

  DCMCF_UOC_PRO_SLUP2

  　SLUTYPE-P プロトコル（2 次局）

- 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）の-e オプションを省略した場合，ebuflist_adr には NULL が設定されます。
- output_no
  メッセージ出力通番が設定されます。ただし，outputno_flag が DCMCF_UOC_OUTPUTNO_OK のときだけ有効です。
- msg_type
  メッセージ種別として，次のどちらかの値が設定されます。

  'n'
  一般の一方送信メッセージ

  'p'
  優先の一方送信メッセージ

  's'
  同期問い合わせメッセージ
- outputno_flag
  メッセージ出力通番の有効フラグとして，次のどちらかの値が設定されます。

  DCMCF_UOC_OUTPUTNO_OK
  メッセージ出力通番を有効にします。

  DCMCF_UOC_OUTPUTNO_NG
  メッセージ出力通番を無効にします。
- resend_flag
  再送メッセージフラグとして，次のどちらかの値が設定されます。

  'r'
  再送メッセージです。

  'n'
  再送メッセージではありません。
- pro_indv_ifa
  MCF で使用するパラメタです。

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

「5.1.2 入力メッセージ編集 UOC インタフェース」を参照してください。

(5)　ユーザが値を設定する項目

(a)　dcmcf_uoc_mout_n

- rtn_detail

  詳細リターンコードを設定します。

  このコードは，UOC が DCMCF_UOC_MSG_NG をリターンしたときに，MCF に渡されます。MCF は，詳細リターンコードをメッセージログファイルに出力します。

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

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

「5.1.2　入力メッセージ編集 UOC インタフェース」を参照してください。

(6)　リターン値

uoc_func()は，次のコードでリターンしてください。

リターン値	意味
DCMCF_UOC_MSG_OK	正常リターン（編集バッファでスケジューリング）
DCMCF_UOC_MSG_OK_SND	正常リターン（送信バッファでスケジューリング）
DCMCF_UOC_MSG_NG	メッセージ編集エラー

(7)　パラメタとバッファの関係

UOC インタフェース用のパラメタとバッファの関係は，入力メッセージの編集の場合と同じです。
「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 では，受け取った出力通番を基に，ユーザ独自の処理ができます。例えば，出力通番を使用して送信メッセージを編集できます。

送信メッセージの通番編集 UOC を起動する場合，メッセージ送信の関数で出力通番を付けるように設定してください。UOC は，UAP が先頭セグメントを送信する send 関数または resend 関数を呼び出したときに，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。

(1)　OpenTP1 への組み込み方法

UAP のメイン関数の中に，送信メッセージの通番編集 UOC の関数アドレスを登録しておきます。UAP のメイン関数に登録する dc_mcf_regster 関数の形式を次に示します。

(a) 形式

ANSI C，C++の場合

```
#include<dcmcf.h>

int dc_mcf_regster(DCLONG flags, DCLONG (*uoc_addr)(DCLONG flags,
                    char   *termname, DCLONG sendno,
                    DCLONG sendid,DCLONG dataleng,
                    char   *senddata))
```

K&R 版 C の場合

```
#include<dcmcf.h>

int dc_mcf_regster(flags, uoc_addr)
DCLONG flags;
DCLONG (*uoc_addr)();
```

(b) ユーザが値を設定する引数

- flags

  DCMCF_SEND_UOC を設定します。

- 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ました。

(d) メイン関数への登録例

- MHP の場合

```
main()
{
extern DCLONG send_uoc();

dc_rpc_open();
dc_mcf_open();
dc_mcf_regster(DCMCF_SEND_UOC, send_uoc);
dc_mcf_mainloop();
dc_mcf_close();
dc_rpc_close();
}
```

- SPP の場合

```
main()
{
extern DCLONG send_uoc();

dc_rpc_open();
dc_mcf_open();
dc_mcf_regster(DCMCF_SEND_UOC,send_uoc);
dc_rpc_mainloop();
dc_mcf_close();
dc_rpc_close();
}
```

5.1.6　送信メッセージの通番編集 UOC インタフェース

送信メッセージの通番編集 UOC は，次に示す形式で，send_uoc 関数として作成します。

なお，UOC の関数名称はユーザの任意です。

(1)　形式

ANSI C，C++の場合

```
#include<dcmcf.h>

DCLONG send_uoc(DCLONG flags, char *termname, DCLONG sendno,
                DCLONG sendid,DCLONG dataleng, char *senddata)
```

K&R 版 C の場合

```
#include<dcmcf.h>

DCLONG send_uoc(flags, termname, sendno, sendid, dataleng,
                senddata)
DCLONG        flags;
char          *termname;
DCLONG        sendno;
DCLONG        sendid;
DCLONG        dataleng;
char          *senddata;
```

(2)　MCF が値を設定する引数

- flags

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

  DCMCF_SEND_DML

  　メッセージを送信する関数または命令文が呼び出されたとき

  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設定されます。
- sendno
  送信メッセージの出力通番が設定されます。
- sendid
  送信するメッセージ種別が設定されます。次のどちらかが設定されます。

  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設定されます。
- senddata
  先頭セグメントの内容が設定されます。

(3) リターン値

リターン値	意味
DC_OK	正常に終了しました。

5.1.7 UOC 作成上の注意事項

UOC 作成上の注意事項を次に示します。

(1) UOC の構造

UOC で使用するローカル変数のサイズの合計は，各 UOC で 1024 バイト以内になるよう作成してください。また，UOC の中で関数の再帰呼び出しはしないでください。

(2) UOC で使用できる関数

UOC を作成する場合，UOC では次に示す関数だけが使用できます。ほかの関数を使用した場合，正常に動作しないことがあるためご注意ください。

- メモリ操作をする関数
  - データ領域管理（例：malloc，free）
  - 共有メモリ管理関数（例：shmctl，shmget，shmop）
  - メモリ操作（例：memcpy）

- 文字列操作（例：strcpy）

- 時間取得関数

(3) UOC の異常処理

SLU - TypeP2 の UOC で異常を検知した場合，MCF の所定のリターンコードを使用して，MCF に異常の発生を通知してください。UOC でプロセス終了となるシグナルまたは abort()を発行すると，MCF が異常終了します。

(4) UOC の実行タイミング

MCF が起動する UOC の実行タイミングは，OpenTP1 システムおよび UAP の開始，終了シーケンスと同期しない場合があります。したがって，UAP より先に UOC が実行されたり，UAP がすべて終了してから UOC が呼び出されたりしてもよいように作成してください。

(5) ユーザセグメントの操作方法

UOC でユーザセグメントを参照または設定する場合，ユーザセグメントの先頭アドレスがバウンダリ調整されていないことを確認してください。ユーザセグメントの参照・設定方法によっては，バウンダリアクセス例外が発生する場合があります。必要に応じて，メモリ操作関数（memcpy，memset など）を使用してください。

5.2 MCF イベントインタフェース

SLU - TypeP2 でメッセージ送受信をすると，OpenTP1 の各種システム情報が MHP に通知されます。これを **MCF イベント**といいます。メッセージ送受信処理でエラーや障害が発生した場合，システム内で何が起こっているのかが MCF イベントの内容でわかります。MCF イベントに対応する MHP を **MCF イベント処理用 MHP** といいます。

MCF イベントは入力キューに渡されて，MCF イベント処理用 MHP で回復処理をします。なお，MCF イベントの発生時は入力メッセージ編集 UOC を呼び出しません。詳細については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

5.2.1 MCF イベントの種類

SLU - TypeP2 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 SLU - TypeP2 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。 問い合わせメッセージの場合は，応答メッセージを出力できます。
メッセージ廃棄通知イベント	ERREVT2	次の理由で受信メッセージを破棄しました。 • 入力キューに障害が発生しました。 • 入力メッセージ最大格納数を超過しました。 • 動的共用メモリが不足しました。 • キューファイルが満杯になりました。 • MHP のサービス，サービスグループ，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。	メッセージを廃棄したことを報告します。 問い合わせメッセージの場合は，応答メッセージを出力できます。

MCF イベント名	MCF イベントコード	発生した原因	MCF イベント処理用 MHP での処理の例
メッセージ廃棄通知イベント	ERREVT2	• 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。 問い合わせメッセージの場合は，応答メッセージを出力できます。
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。 問い合わせメッセージの場合は，応答メッセージを出力できます。
タイマ起動メッセージ廃棄通知イベント	ERREVT4	アプリケーションのタイマ起動時に障害が発生しました。	メッセージを廃棄したことを報告します。
未処理送信メッセージ廃棄通知イベント	ERREVTA	次の理由で未処理送信メッセージを破棄しました。 • MCF の正常終了処理時に，未処理送信メッセージの滞留時間監視の時間切れ（タイムアウト）が発生しました。 • 運用コマンド（mcftdlqle）の入力，または API（dc_mcf_tdlqle 関数もしくは CBLDCMCF('TDLQLE△△')）の発行によって，出力キューが削除されました。 • 閉塞されている論理端末の出力キューに未処理送信メッセージが残った状態で，dcstop コマンドが実行されました。	未処理送信メッセージを廃棄したことを報告します。
障害通知イベント	CERREVT	通信管理プログラムのコネクション障害，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。

注※

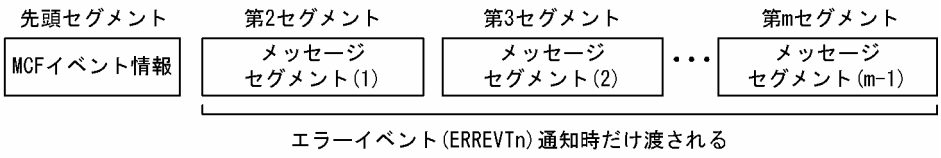
MCF アプリケーション定義(mcfaalcap -g)の recvmsg オペランドに r を指定した場合，または dc_mcf_rollback の action に DCMCFRTRY または DCMCFRRTN を指定した場合は除きます。

5.2.2 MCF イベント通知時のセグメント構成

MCF イベントを MHP に通知する場合，先頭セグメントに MCF イベント情報を設定します。エラーイベント（ERREVTn）の場合は，第 2 セグメント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 5-3　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 に通知されるデータの形式が異なります。

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 ERREVT1，ERREVT2，ERREVT3，および ERREVTA のバッファ形式別に位置（バイト）を分けて説明しています。

なお，ERREVT4 についてはマニュアル「OpenTP1 プログラム作成リファレンス」の該当する言語編を参照してください。

5.2.3　MCF イベント情報の形式（C 言語）

C 言語の場合は，イベント別の情報が，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。なお，dc_mcf_evtheader は<dcmcf.h>で定義されています。<dcmslm.h>の前に<dcmcf.h>を取り込んでおいてください。

(1)　MCF イベントの共通ヘッダ

(a)　形式

```
struct dc_mcf_evtheader{
    char mcfevt_name[9];        … MCFイベントコード
    char le_name[16];           … 入力元論理端末名称
                                  (ERREVT1, ERREVT2, ERREVT3,
                                   ERREVT4の場合)
                                … 出力先論理端末名称
                                  (ERREVTAの場合)
    char cn_name[9];            … コネクション名
    unsigned char format_kind; … MCF使用領域
    char reserve01;             … 予備
    DCLONG time;                … メッセージ入力時刻
};
```

(b) MCF イベントとして設定される項目

- le_name

  ERREVT1，ERREVT2，または ERREVT3 では，メッセージを入力した論理端末名称が設定されます。

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

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

  ERREVTA では，メッセージを出力する論理端末名称が設定されます。

  CERREVT，COPNEVT，または CCLSEVT の場合は無効です。

- cn_name

  コネクション名が設定されます。

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

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

- time

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

(2) ERREVT1

(a) 形式

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

(b) MCF イベントとして設定される項目

- ap_name

  次に示すどちらかが設定されます。

  - 形式不正の場合…不正となったアプリケーション名
  - 定義されていない場合…定義されていないアプリケーション名

アプリケーション名は，MHP から送信されたメッセージの場合に設定されます。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 H　理由コードおよびセンスコード一覧」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

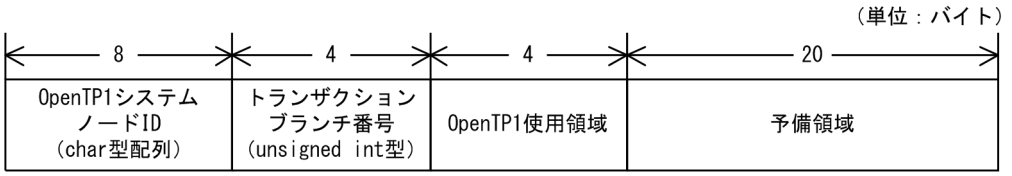
  異常が発生した MHP のアプリケーション名に対応するサービス名が設定されます。

- 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。



(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

(a) 形式

```
typedef struct{
    struct dc_mcf_evtheader  header;
                                ... MCFイベント共通ヘッダ
    DCLONG err_fact;            ... 障害要因コード（4バイト）
    DCLONG err_reason1;         ... 理由コード1（4バイト）
    DCLONG err_reason2;         ... 理由コード2（4バイト）
    DCLONG err_rcv_action;      ... 回復動作情報
    char reserve1[42];          ... 予備
}dcmslm_cerrevt;
```

(b) MCF イベントとして設定される項目

- err_fact

  CERREVT の障害要因コードが次に示す値で設定されます。

  $(00000030)_{16}$

  コネクション障害発生

  $(00000031)_{16}$

  論理端末障害発生

- err_reason1，err_reason2

  CERREVT の理由コードが設定されます。「付録 H　理由コードおよびセンスコード一覧」を参照してください。

- err_rcv_action

  CERREVT 通知時に，回復動作情報として次の値が設定されます。

  DCMSLM_RSV_MANUAL

  コマンド入力による手動回復

  DCMSLM_RSV_AUTO

  システムによる自動回復

  回復動作情報は，該当するコネクションが端末起動の場合は手動回復になり，ホスト起動の場合は自動回復になります。

(7) COPNEVT，CCLSEVT

(a) 形式

```
typedef struct{
    struct dc_mcf_evtheader header;
                                ... MCFイベント共通ヘッダ
    DCLONG cls_rcv_action;      ... 回復動作情報
                                    (CCLSEVTの場合だけ有効)
```

```
    char reserve1[54];          … 予備
}dcmslm_statevt;
```

(b) MCF イベントとして設定される項目

- cls_rcv_action

  CCLSEVT 通知時に，回復動作情報として次の値が設定されます。

  DCMSLM_RSV_MANUAL

  　コマンド入力による手動回復

  DCMSLM_RSV_AUTO

  　システムによる自動回復

  回復動作情報は，該当するコネクションが端末起動の場合は手動回復になり，ホスト起動の場合は自動回復になります。

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，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 UAP に通知される MCF イベント情報の内容（ERREVT1）

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備（形式 1 のときだけ）	0	−	2	−	−
予備（形式 1 のときだけ）	2	−	2	−	−
エラーイベントコード	4	0	3	英数字	'ERR'が設定されます。
	7	3	3	−	−
	10	6	2	英数字	ERREVT1 を示す'1△'が設定されます。
入力元論理端末名称	12	8	8	英数字	メッセージを入力した論理端末名称です。
予備	20	16	20	−	−
アプリケーション名	40	36	8	英数字	次に示すどちらかが設定されます。 • 形式不正となったアプリケーション名 • 定義されていないアプリケーション名
予備	48	44	8	−	−
予備	56	52	8	−	−
予備	64	60	8	−	−

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
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 UAP に通知される MCF イベント情報の内容（ERREVT2）

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備（形式 1 のときだけ）	0	－	2	－	－
予備（形式 1 のときだけ）	2	－	2	－	－
エラーイベントコード	4	0	3	英数字	'ERR'が設定されます。
	7	3	3	－	－
	10	6	2	英数字	ERREVT2 を示す'2△'が設定されます。
入力元論理端末名称	12	8	8	英数字	メッセージを入力した論理端末名称です。次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	20	16	20	－	－
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。

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
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 H　理由コードおよびセンスコード一覧」を参照してください。

表 5-4　COBOL 言語の UAP に通知される 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	−	−

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
エラーイベントコード	4	0	3	英数字	'ERR'が設定されます。
	7	3	3	−	−
	10	6	2	英数字	ERREVT3 を示す'3△'が設定されます。
入力元論理端末名称	12	8	8	英数字	メッセージを入力した論理端末名称です。次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	20	16	20	−	−
予備	40	36	8	−	−
マップ名	48	44	8	−	MCF が使用します。
アプリケーション名	56	52	8	英数字	異常が発生したメッセージのアプリケーション名です。
予備	64	60	8	−	−
コネクション名	72	68	8	英数字	コネクション名です。次に示す場合は，'*'が設定されます。 • SPP からアプリケーション起動機能で起動した MHP で，障害が発生した場合 • 上記の障害が発生したあとに，MCF イベントとして起動した MHP からさらにアプリケーション起動機能で起動した MHP で，障害が発生した場合
予備	80	76	16	−	−
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。 YYYYMMDD の形式です。 YYYY：西暦の年 MM：月 DD：日
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。 HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
予備	112	108	16	−	−

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
サービス名	128	124	31	英数字	異常が発生した UAP のアプリケーション名に対応するサービス名です。
予備	159	155	1	−	−
サービスグループ名	160	156	31	英数字	異常が発生した UAP のサービスグループ名です。
予備	191	187	1	−	−
トランザクションブランチ ID（BID）	192	188	36	英数字	異常が発生したトランザクションの BID です。 トランザクションブランチ ID の形式については，表 5-5 を参照してください。
予備	228	224	28	−	−

（凡例）

−：該当しません。または，使用されません。

表 5-5　トランザクションブランチ ID の形式

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

（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 UAP に通知される 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	−	−

項目	位置（バイト）		長さ（バイト）	属性	内容
	形式 1	形式 2			
予備	40	36	8	−	−
マップ名	48	44	8	−	MCF が使用します。
アプリケーション名	56	52	8	英数字	正常終了したメッセージのアプリケーション名です。 MHP から送信されたメッセージの場合設定されます。MHP 以外から送信された場合は空白が設定されます。
予備	64	60	8	−	−
コネクション名	72	68	8	英数字	コネクション名です。
予備	80	76	16	−	−
メッセージが入力された日付	96	92	8	外部 10 進数字	端末入力メッセージを入力した日付です。 YYYYMMDD の形式です。 YYYY：西暦の年 MM：月 DD：日
メッセージが入力された時刻	104	100	8	外部 10 進数字	端末入力メッセージを入力した時刻です。 HHMMSS00 の形式です。 HH：時 MM：分 SS：秒 00 は固定です。
予備	112	108	16	−	−
予備	128	124	31	−	−
予備	159	155	1	−	−
予備	160	156	31	−	−
予備	191	187	1	−	−
予備	192	188	36	−	−
予備	228	224	28	−	−

（凡例）

−：該当しません。または，使用されません。

表 5-7　COBOL 言語の UAP に通知される MCF イベント情報の内容（CERREVT）

項目	位置（バイト）	長さ（バイト）	属性	内容
イベントコード	0	8	英数字	イベントコード「CERREVT」が設定されます。
入力元論理端末名称	8	8	英数字	障害の発生した論理端末名称が設定されます（コネクション障害時は無効です）。
予備	16	8	－	－
入力元コネクション名	24	8	英数字	コネクション名が設定されます。
メッセージ入力日付	32	8	外部 10 進数字	CERREVT を入力した日付です。
メッセージ入力時刻	40	8	外部 10 進数字	CERREVT を入力した時刻です。
障害要因コード	48	4	2 進数字	障害要因コードが設定されます。 $(00000030)_{16}$：コネクション障害 $(00000031)_{16}$：論理端末障害
理由コード 1※	52	4	2 進数字	理由コード 1 が設定されます。
理由コード 2※	56	4	2 進数字	理由コード 2 が設定されます。
回復動作情報	60	4	2 進数字	障害時，システムを回復する方法を示す値が設定されます。 $(00000000)_{16}$：システムによる自動回復 $(ffffffff)_{16}$：コマンド入力による手動回復
予備	64	44	－	－

（凡例）

－：該当しません。または，使用されません。

注※

理由コード 1 および理由コード 2 の内容については，「付録 H　理由コードおよびセンスコード一覧」を参照してください。

表 5-8　COBOL 言語の UAP に通知される 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	無効です。
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。

項目	位置 (バイト)	長さ (バイト)	属性	内容
メッセージ入力日付	32	8	外部 10 進数字	COPNEVT，CCLSEVT を入力した日付です。
メッセージ入力時刻	40	8	外部 10 進数字	COPNEVT，CCLSEVT を入力した時刻です。
回復動作情報	48	4	2 進数字	障害時，システムを回復する方法を示す値が設定されます。 $(00000000)_{16}$：システムによる自動回復 $(ffffffff)_{16}$：コマンド入力による手動回復 回復動作情報は，該当するコネクションが端末起動の場合は手動回復になり，ホスト起動の場合は自動回復になります。
予備	52	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

この章では，SLUTYPE-P 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 SLU - TypeP2 固有のシステム定義について説明します。また，自システム内にある通信管理プログラムと関連づける内容，および定義例について説明します。

SLU - TypeP2 の定義の概要

SLU - TypeP2 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。また，自システムの通信管理プログラムと，システム定義内容を関連づける必要があります。

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，SLU - TypeP2 に固有の定義について説明します。

使用する定義ファイル

MCF および SLU - TypeP2 を起動するには，定義ファイルに環境情報を指定する必要があります。MCF で使用する定義ファイルを次の表に示します。

表 6-1　MCF で使用する定義ファイル

定義の種類	定義のソースファイル	定義の内容
MCF マネジャ定義	MCF マネジャ定義ソースファイル	MCF 全体の実行環境
MCF 通信構成定義	共通定義ソースファイル	プロトコルごとの実行環境
	プロトコル固有定義ソースファイル	
MCF アプリケーション定義	MCF アプリケーション定義ソースファイル	アプリケーションの属性

定義のソースファイルは，定義コマンド，オプション，およびオペランドを指定して作成します。それらの中には，プロトコルで共通のものと，プロトコルに固有のものがあります。表 6-1 の定義の中で，SLU - TypeP2 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

この章では，SLU - TypeP2 に固有の定義コマンド，オプション，およびオペランドについて説明します。プロトコルで共通の定義については，マニュアル「OpenTP1 システム定義」を参照してください。ただし，mcftbuf（バッファグループ定義）の length，count オペランドの指定値については，mcftalccn の注意事項に記載してあります。

SLU - TypeP2 の組み込み時に必要なファイル

次に示すファイルは，SLU - TypeP2 を OpenTP1 システムに組み込むときに必要なファイルです。

- システムサービス情報定義ファイル
- システムサービス共通情報定義ファイル
- MCF 定義オブジェクトファイル

この章では，システムサービス情報定義ファイルとシステムサービス共通情報定義ファイルの記述内容，およびMCF定義オブジェクトファイルを生成するユティリティの起動コマンドについて説明します。SLU - TypeP2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

SLUTYPE-P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，SLU - TypeP2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と関連づける必要があります。

この章では，自システムの通信管理プログラム（XNF/AS）と関連づける内容を示します。

SLU - TypeP2 固有のシステム定義の種類

OpenTP1 のネットワークコミュニケーション定義のうち，SLU - TypeP2 に固有の定義の一覧を次の表に示します。

表 6-2　SLU - TypeP2 固有の定義の一覧

定義名		コマンド	オプション・オペランド		定義内容	指定値((値範囲))《省略時解釈値》
MCF マネジャ定義		mcfmuap※	-t	sndrcvtim	同期型送受信監視時間	符号なし整数((0～65535))《0》（単位：秒）
MCF 通信構成定義	共通定義	プロトコル共通のコマンドだけで指定できます。共通のコマンドについては，マニュアル「OpenTP1 システム定義」を参照してください。				
	プロトコル固有定義	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	slup2
			-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
			-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）
			-k	－	起動種別	host | ws | auto
			-n	ownnode	自局ノード名称	符号なし整数((0～253))
			-q	hostnode	ホストノード名称	1～8 けたの 16 進数字
			-r	sndrusiz	送信最大 RU 長	符号なし整数((0～32767))
				rcvrusiz	受信最大 RU 長	符号なし整数((8～32767))
			-o	－	LOGON モード名称	1～8 文字の識別子 | space

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	-j	pluname	PLU 名称	1～8 文字の識別子
			-d	－	問い合わせメッセージの応答識別	《rqd》｜rqe
			-w	cmderrstp	状態変更不可時の運用コマンドを正常リターンするかどうかを指定	yes｜《no》
				firststsn	OpenTP1 の開始後，初めての STSN コマンドに対する応答の種類	positive｜《negative》
			-f	－	FMH を使用するかどうかを指定	《yes》|no
			-t	termself	TERM-SELF の終了方式	《orderly》|forced
				setplu	TERM-SELF に PLU 名称を設定するかどうかを指定	yes|《no》
				dactlu	TERM-SELF の DACTLU 送信フィールドに設定する送信種別	on|《off》
		mcftalcle（論理端末定義） 指定数：1～512	-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	send｜receive｜request
			-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
			-v	－	アプリケーション名	1～8 文字の識別子
		mcftalced（コネクション定義の終了）	－	－	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	指定数：mcftalccn と同数	－	－	コネクション定義の終了	－

（凡例）

－：該当しません。

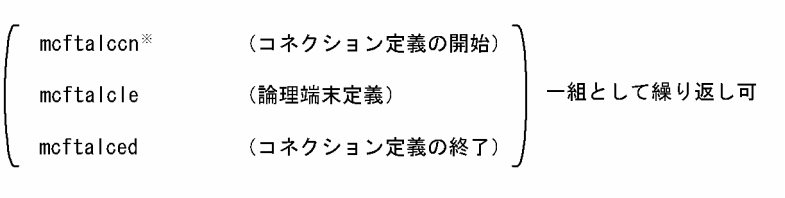
注※

SLU - TypeP2 に固有の定義だけ記載してあります。このほかにも，プロトコルで共通の定義コマンド，オプション，オペランドがあります。それらについては，マニュアル「OpenTP1 システム定義」を参照してください。

定義の指定順序

SLU - TypeP2 のプロトコル固有定義コマンドの指定順序を次の図に示します。MCF 通信構成定義コマンドを指定するときは，必ずこの順序に従ってください。

図 6-1　SLU - TypeP2 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 1 になるようにしてください。

mcfmuap（UAP 共通定義）

形式

```
mcfmuap          :
       〔-t ”〔sndrcvtim=同期型送受信監視時間〕”〕
                 :
```

機能

UAP に共通する環境を定義します。

オプション

この定義コマンドには，ほかにもオプションおよび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。

●-t

（オペランド）

sndrcvtim=同期型送受信監視時間　　～<符号なし整数>((0～65535))《0》（単位：秒）

同期型のメッセージ送受信の仕掛り開始（sendrecv（EMI）発行）から仕掛り終了（sendrecv 終了）までの限界監視時間を指定します。このオペランドでは，相手システムからの応答時間を監視します。0 を指定した場合，送受信時間の監視はしません。

注意事項

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

mcftalccn（コネクション定義の開始）

形式

```
mcftalccn  -c コネクションID
           -p slup2
           -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”
          〔-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〕”〕
           -k host｜ws｜auto
           -n ”ownnode=自局ノード名称”
           -q ”hostnode=x’ホストノード名称’”
           -r ”sndrusiz=送信最大RU長
               rcvrusiz=受信最大RU長”
           -o e’LOGONモード名称’｜space
           -j ”pluname=e’PLU名称’”
          〔-d rqd｜rqe〕
          〔-w ”〔cmderrstp=yes｜no〕
                〔firststsn=positive｜negative〕”〕
          〔-f yes｜no〕
          〔-t ”〔termself=orderly｜forced〕
                〔setplu=yes｜no〕
                〔dactlu=on｜off〕”〕
```

機能

コネクションに関する環境を定義します。

オプション

●-c コネクション ID　～＜1～8 文字の識別子＞

OpenTP1 システム内で，一意となるコネクション ID を指定します。

●-p slup2

プロトコルの種別を指定します。

slup2

SLUTYPE-P プロトコル（2 次局）

●-g

（オペランド）

sndbuf=メッセージ送信用バッファグループ番号　～<符号なし整数>((1～512))

メッセージ送信用バッファグループ番号を指定します。

mcftbuf コマンドの-g オプションの groupno オペランドで指定するバッファグループ番号を指定してください。

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

メッセージ受信用バッファグループ番号を指定します。

mcftbuf コマンドの-g オプションの groupno オペランドで指定するバッファグループ番号を指定してください。

●-e

(オペランド)

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

入力，出力メッセージ編集 UOC 呼び出し時に，メッセージ編集用として使用するバッファグループ番号を指定します。このオペランドを省略した場合は，メッセージ編集用バッファは確保されません。

メッセージ編集用バッファグループ番号には，mcftbuf コマンドの-g オプションの groupno オペランドで指定するバッファグループ番号を指定してください。

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 mcftbuf コマンドの-g オプションの count，および extend オペランドで指定するバッファ数の中から，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 mcftbuf コマンドの-g オプションの count，および extend オペランドで指定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指定します。

xnfas

XNF/AS を使用します。

●-i auto | manual　～《manual》

オンライン開始時および再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

オンライン開始時および再開始時にコネクションを自動的に確立します。

manual

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

-k オプションで host または auto を指定した場合，-i オプションの指定は無効になります。

●-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，コネクションの確立再試行をするかどうかを指定します。

yes

コネクションの確立再試行をします。

no

コネクションの確立再試行をしません。

-k オプションで host を指定した場合，bretry オペランドの指定は無効になります。

-k オプションで auto を指定した場合，bretry オペランドの指定に関係なくコネクションの確立再試行をします。ここで，bretry オペランドに no を指定した場合，bretryint オペランドの指定に関係なく直ちにコネクションの確立再試行をします。

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 を指定した場合は，無限に確立再試行を繰り返します。

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。-k オプションで host または auto を指定した場合，bretrycnt オペランドの指定は無効になります。

通信管理から再試行不可能な障害が通知された場合，または相手システムから確立要求に対する拒否応答を受けた場合は，再試行を中止します。

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

コネクションの確立再試行をする場合の確立再試行間隔を指定します。0 を指定した場合，障害が発生するたび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。-k オプションで host 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。

●-k host | ws | auto

起動種別を指定します。

host

ホスト（1 次局）側から SLU - TypeP2 へコネクションを確立します。SLU - TypeP2 は，ホスト（1 次局）側からのコネクション確立要求(BIND)を待ち合わせます。

ws

SLU - TypeP2 からホスト（1 次局）側へコネクションを確立します。

auto

INI-SELF 送信制御機能でコネクションを確立します。

●-n

（オペランド）

ownnode=自局ノード名称　　～<符号なし整数>((0～253))

自局ノード名称を指定します。XNF/AS の構成定義で指定した LU 番号を指定してください。

詳細については，この章の「自システムの通信管理プログラムと関連づける内容」を参照してください。

●-q

（オペランド）

hostnode=x'ホストノード名称'　　～< 1～8 けたの 16 進数字>

ホストノード名称を指定します。x は，16 進数形式で指定することを意味します。

XNF/AS の構成定義で指定した PU 番号と同じ値を指定してください。

詳細については，この章の「自システムの通信管理プログラムと関連づける内容」を参照してください。

●-r

（オペランド）

sndrusiz=送信最大 RU 長　　～<符号なし整数>((0～32767))

送信最大 RU 長を指定します。

rcvrusiz=受信最大 RU 長　　～<符号なし整数>((8～32767))

受信最大 RU 長を指定します。

●-o

（オペランド）

e'LOGON モード名称'　～< 1～8 文字の識別子>

LOGON モード名称を指定します。e は，EBCDIC コードに変換されることを意味します。

指定した文字数が 8 文字に満たない場合，指定領域の右側はスペースで埋められます。LOGON モード名称は EBCDIC コードに変換されます。

space

LOGON モード名称に 8 文字のスペース（EBCDIC コードの$(4040404040404040)_{16}$）を設定します。

●-j

(オペランド)

pluname=e'PLU 名称'　～< 1～8 文字の識別子>

PLU 名称を指定します。e は，EBCDIC コードに変換されることを意味します。

●-d rqd | rqe　～《rqd》

ホスト（1 次局）への問い合わせ応答時の，応答識別を指定します。複数セグメントの問い合わせメッセージを送信する場合，このオプションの指定値は，最終セグメントだけ有効です。先頭セグメントおよび中間セグメントの応答識別は，このオプションの指定内容に関係なく，RQE になります。

rqd

問い合わせメッセージの応答識別に RQD を指定します。RQD を指定すると，問い合わせメッセージに対するホストからの応答（+RSP または-RSP）が送信されます。

rqe

問い合わせメッセージの応答識別に RQE を指定します。RQE を指定すると，エラーが発生した場合だけホストからの応答（-RSP）が送信されます。

●-w

(オペランド)

cmderrstp=yes | no　～《no》

SLU - TypeP2 が運用コマンド（mcftactcn，mcftactle，mcftdctcn，および mcftdctle）による状態変更ができない場合，運用コマンドを正常に受け付けたことにするかどうかを指定します。状態変更ができない場合とは，コネクションが確立済みのときに mcftactcn を入力したなどが該当します。

ただし，コネクションおよび論理端末の状態と受け付けた運用コマンドの組み合わせが以下の場合，このオペランドの指定に関係なくエラーとします。

- コネクション確立処理障害時（mcftactcn）
- コネクション使用中（mcftdctcn）
- コネクション解放処理障害（正常解放時，および強制解放時）（mcftdctcn）
- 論理端末閉塞解除処理障害（mcftactle）
- 論理端末閉塞処理障害（mcftdctle）

- 論理端末使用中（mcftdctle）

yes

入力された運用コマンドを正常に受け付けたことにします。

no

入力された運用をエラーとします。運用コマンドはエラーメッセージを出力します。

firststsn=positive | negative　～《negative》

OpenTP1 の開始後初めてのコネクションの確立において，ホストから STSN コマンドのアクションコードが"SET&TEST 要求"の送達確認がされた場合に，ホストに送信する応答の種類を指定します。

positive

ホストに肯定応答（TEST POSITIVE）を送信します。

negative

ホストに否定応答（TEST NEGATIVE）を送信します。

●-f yes | no　～《yes》

ホストに送信するデータで，FMH を使用するかどうかを指定します。

yes

ホストへのデータ送信で FMH を使用します（RH 内の FI フィールドに ON を設定する）。

ただし，FMH データは UAP で送信メッセージ内に設定する必要があります。

no

ホストへのデータ送信で FMH を使用しません（RH 内の FI フィールドに OFF を設定する）。

ただし，このオペランドの指定値に関係なく，ホスト側から受信する BIND セションパラメタは，FMH を使用する設定（バイト 6，ビット 1=ON）になっている必要があります。FMH を使用しない設定の場合，BIND パラメタ不正として-RSP（センスコード：0x0821）が応答され，コネクションの確立が拒否されます。

●-t

（オペランド）

termself =orderly | forced　～《orderly》

TERM-SELF コマンドの終了方式フィールド（バイト 3，ビット 2）に設定する終了方式を指定します。

orderly

計画終了（'1'）を設定します。

forced

強制終了（'0'）を設定します。

setplu=yes | no　～《no》

TERM-SELF コマンドに PLU 名称を設定するかどうかを指定します。設定する PLU 名称は，-j オプションに指定した PLU 名称です。

yes

PLU 名称を設定します。

no

PLU 名称を設定しません。

なお，TERM-SELF コマンドの解放セションフィールド（バイト 3，ビット 5～6）への値の設定は，SLU - TypeP2 が setplu オペランドの指定値を基に決定します。

setplu オペランドの指定	解放セションフィールドへの設定値
yes の場合	'00'
no の場合	'10'
-t オプションの setplu は省略し，そのほかのオペランドを指定した場合	'10'
-t オプション全体を省略した場合	'00'：SLU - TypeP2 が TERM-SELF コマンドを送信する場合 '10'：通信管理が TERM-SELF コマンドを送信する場合

注 1

解放セションフィールドの設定値の意味は次のとおりです。

- '00'が設定されると，TERM-SELF コマンドの送信先が PLU に限定されます。
- '10'が設定されると，PLU，SLU の区別に関係なく TERM-SELF コマンドが送信されます。

注 2

解放セションフィールドの設定値は，-t オプション全体を省略する場合と-t オプションの各オペランドに省略時解釈値を明示指定する場合で異なります。解放セションフィールドの設定値をこのオプションが未サポートの SLU - TypeP2 旧バージョンを使用する場合と同じにするには，-t オプション全体を省略してください。

dactlu=on | off　～《off》

TERM-SELF コマンドの DACTLU 送信フィールド（バイト 3，ビット 3）に設定する送信種別を指定します。

on

送信可（'1'）を設定します。

off

送信不可（'0'）を設定します。

注意事項

-g オプション，および-e オプションで指定するバッファグループ番号は，バッファグループ定義の mcftbuf コマンドに対応しています。mcftbuf コマンドでは，1 コネクション単位に次の表に示す資源が必要です。バッファグループ定義については，マニュアル「OpenTP1 システム定義」を参照してください。

バッファ種別	バッファサイズ	バッファ面数
sndbuf	ユーザデータ長（送信最大 RU 長）	最大セグメント分割数＋ 1
rcvbuf	ユーザデータ長（受信最大 RU 長）	最大セグメント分割数＋ 2
msgbuf	次に示すどちらか大きい方の値 • ユーザデータ長（送信最大 RU 長） • ユーザデータ長（受信最大 RU 長）	メッセージ編集用バッファ数（mcftalccn -e count）×2

mcftalced（コネクション定義の終了）

形式

```
mcftalced
```

機能

コネクション定義の終了を示します。

オプション

ありません。

mcftalcle（論理端末定義）

形式

```
mcftalcle -l 論理端末名称
          -t send | receive | request
         〔-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〕”〕
         〔-v アプリケーション名〕
```

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 send | receive | request

この論理端末の端末タイプを指定します。

send
　送信型論理端末

receive
　受信型論理端末

request
　問い合わせ型論理端末

●-m

（オペランド）

mmsgcnt=メモリ出力メッセージ最大格納数　～＜符号なし整数＞((0～65535))《0》
　メモリキューで待ち合わせをする出力メッセージの最大格納数を指定します。
　出力メッセージの待ち合わせ数が指定した最大数になると，それ以後 UAP からの送信要求（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）になります。ただし，実際に待ち合わせをできる出力メッセージ数は動的共用メモリの容量に依存します。

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

ディスクキューで待ち合わせをする出力メッセージの最大格納数を指定します。

出力メッセージの待ち合わせ数が指定した最大数になると，それ以後 UAP からの送信要求（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）になります。ただし，実際に待ち合わせをできる出力メッセージ数はメッセージキューファイルの容量に依存します。

●-k

(オペランド)

quekind=memory | disk　～《memory》

出力メッセージの割り当て先（メモリキューまたはディスクキュー）を指定します。

memory

メモリキューだけに割り当てます。

disk

ディスクキューおよびメモリキューに割り当てます。disk を指定した場合，必ず quegrpid オペランドを指定してください。

quegrpid=キューグループ ID　～< 1～8 文字の識別子>

ディスクキューで待ち合わせをする出力メッセージに使用するキューグループ ID を指定します。MCF マネジャ定義の mcfmqgid コマンドで指定するキューグループ ID（キューグループ種別は 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 を指定した場合だけ指定してください。

●-o

(オペランド)

aj=yes | no　～《yes》

メッセージ送信が完了した場合に，メッセージ送信完了ジャーナル（AJ）を取得するかどうかを指定します。

yes

メッセージ送信完了ジャーナルを取得します。

no

メッセージ送信完了ジャーナルを取得しません。

●-v アプリケーション名　　～< 1～8 文字の識別子>

入力メッセージを受信した場合に起動するアプリケーション名（MHP）を指定します。MCF アプリケーション定義（mcfaalcap -n オプションの name オペランド）で指定した名称を指定してください。MCF アプリケーション定義については，マニュアル「OpenTP1 システム定義」を参照してください。

このオプションを省略した場合は，入力メッセージ編集 UOC で指定された値がアプリケーション名となります。

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=”SLU - TypeP2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="SLU - TypeP2 の実行形式プログラム名"　　～< 1～8 文字の識別子>

MCF 通信サービスを起動するための実行形式プログラム名を指定します。

MCF 実行形式プログラムには，MCF 通信プロセスのためのものとアプリケーション起動プロセスのためのものがあります。

MCF 実行形式プログラムは，MCF 通信プロセス同士，アプリケーション起動プロセス同士で共有できます。

SLU - TypeP2 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

SLU - TypeP2 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

SLU - TypeP2 の場合，MCF 通信構成定義に定義したコネクションの総数を 2 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 内の各 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，SLU - TypeP2 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。

SLU - TypeP2 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，SLU - TypeP2 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，SLU - TypeP2 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，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

```
mcfslup2 -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 SLU - TypeP2 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。

SLU - TypeP2 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。

自システムの通信管理プログラムと関連づける内容

SLU - TypeP2 の定義には，LU の定義について，通信管理と関連づける項目があります。

自局ノード名称とホストノード名称

自局ノード名称とホストノード名称は，通信管理の構成定義で指定した値と一致させてください。通信管理の構成定義については，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。

ownnode オペランド（mcftalccn -n）

ownnode オペランドで指定する自局ノード名称は，HNA2 用全体定義文（HNA2_configuration max_SLUS_LU）で指定した SLUS 用の LU 番号と一致させます。

hostnode オペランド（mcftalccn -q）

hostnode オペランドで指定するホストノード名称は，HNA2 用 PU 定義文（HNA2_PU PU_number）で指定した PU 番号と一致させます。

自システムと相手システムの LU の構成に応じた定義例

自システムの通信管理プログラムのローカルアドレス割り当て機能を使用しない場合

ローカルアドレス割り当て機能を使用しない場合は，通信管理プログラムが「LU 番号＋ 2」をローカルアドレスとして自動的に割り当てます。

SLUS 用 LU だけで構成する例と，SLUS 用 LU 以外の LU と混在する例をそれぞれ示します。

図 6-2　SLUS 用 LU だけの構成例（ローカルアドレス割り当て機能を使用しない場合）

定義する LU	LU 数	使用できる LU の番号	1 次局のローカルアドレス
SLUS 用 LU	10	0～9	2～11

自システム(HNA2次局)の通信管理プログラムの定義文

```
HNA2_configuration
    default_slot_no     1
    max_SLUS_LU         10
    max_SLU_count       10
;

HNA2_PU
    PU_number           0
    destination_name    mcf02_vc
;
```

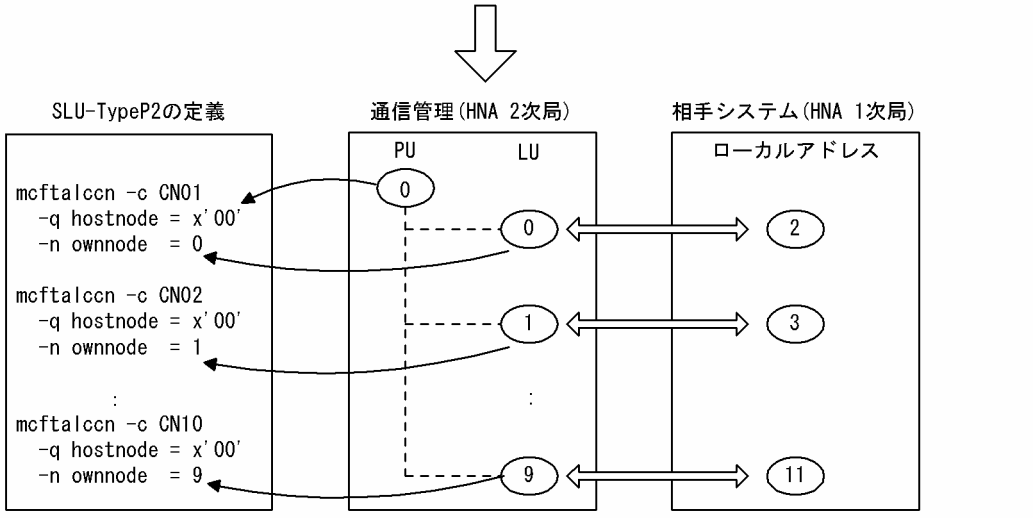
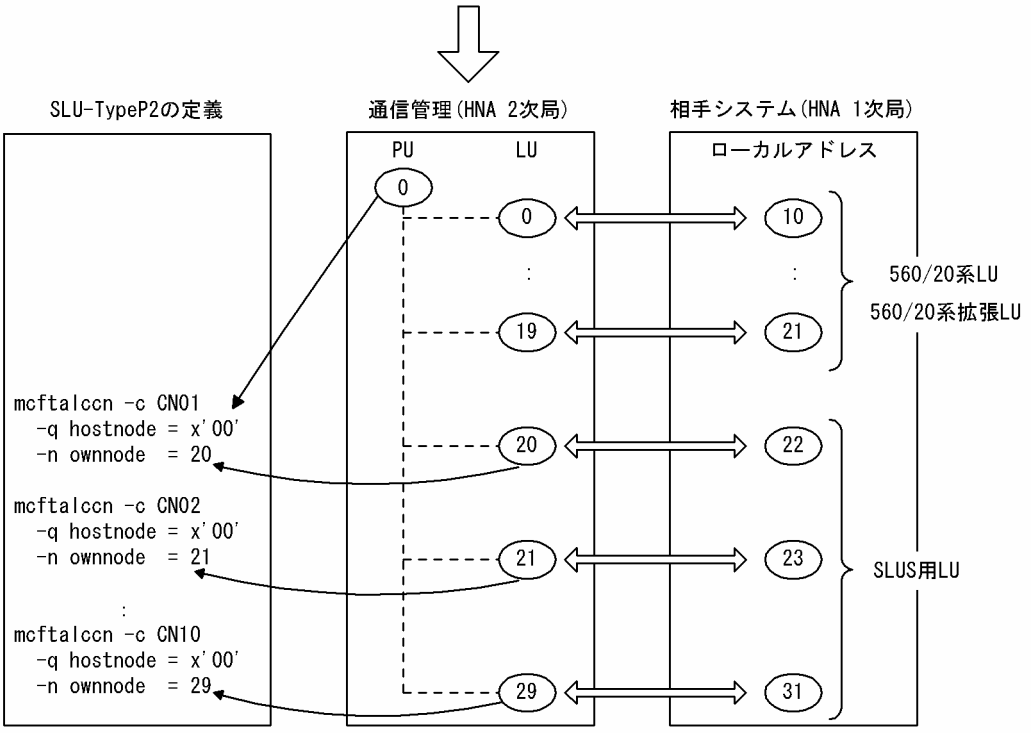


図 6-3　SLUS 用 LU 以外の LU と混在する構成例（ローカルアドレス割り当て機能を使用しない場合）

定義する LU	LU 数	使用できる LU の番号	1 次局のローカルアドレス
560/20 系 LU	10	0～9	2～11
560/20 系拡張 LU	10	10～19	12～21
SLUS 用 LU	10	20～29	22～31

自システム(HNA2次局)の通信管理プログラムの定義文

```
HNA2_configuration
   default_slot_no      1
   max_560_LU           10
   max_extend_LU        10
   max_SLUS_LU          10
   max_SLU_count        10
;

HNA2_PU
   PU_number            0
   destination_name     mcf02_vc
;
```



自システムの通信管理プログラムのローカルアドレス割り当て機能を使用する場合

ローカルアドレス割り当て機能を使用する場合に，SLUS 用 LU だけで構成する例と，SLUS 用 LU 以外の LU と混在する例をそれぞれ示します。

図 6-4　SLUS 用 LU だけの構成例（ローカルアドレス割り当て機能を使用する場合）

定義する LU	LU 数	使用できる LU の番号	1 次局のローカルアドレス
SLUS 用 LU	10	0～9	1～10

自システム(HNA2次局)の通信管理プログラムの定義文

```
HNA2_configuration
    default_slot_no     1
    max_SLUS_LU         10
    max_SLU_count       10
;
HNA2_PU
    PU_number           0
    destination_name
    mcf02_vc
    assign_LA           yes
    first_SLUS_LA       1
;
```

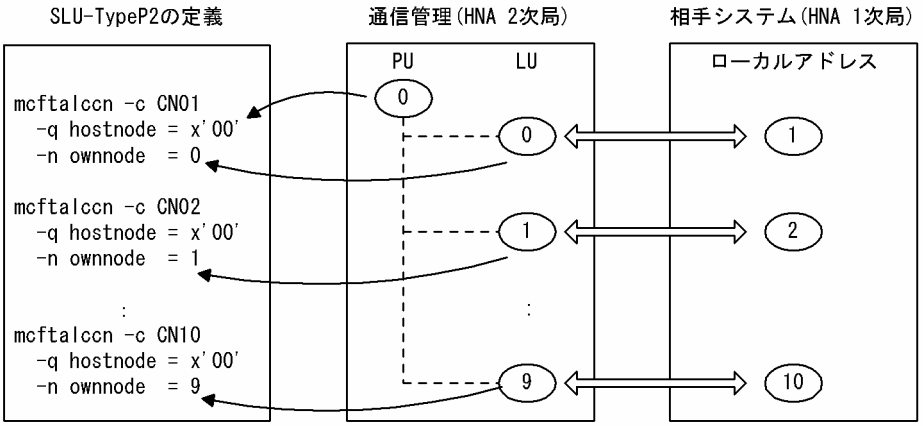
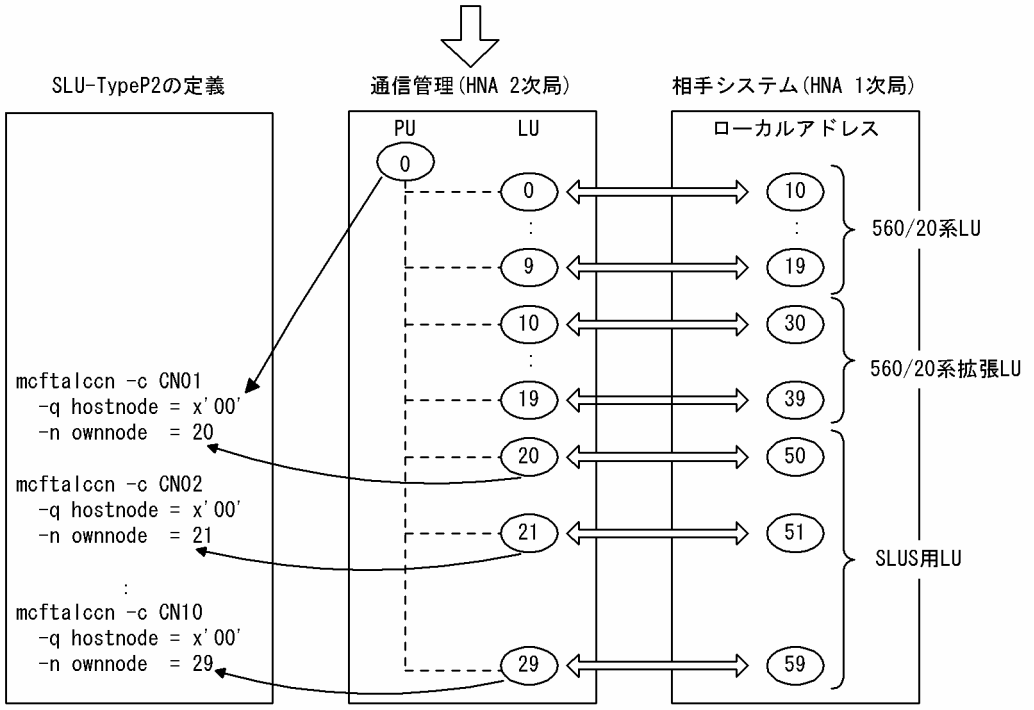


図 6-5　SLUS 用 LU 以外の LU と混在する構成例（ローカルアドレス割り当て機能を使用する場合）

定義する LU	LU 数	使用できる LU の番号	1 次局のローカルアドレス
560/20 系 LU	10	0～9	10～19
560/20 系拡張 LU	10	10～19	30～39
SLUS 用 LU	10	20～29	50～59

自システム(HNA2次局)の通信管理プログラムの定義文

```
HNA2_configuration
    default_slot_no     1
    max_560_LU          10
    max_extend_LU       10
    max_SLUS_LU         10
    max_SLU_count       10
;
HNA2_PU
    PU_number           0
    destination_name
    mcf02_vc
    assign_LA           yes
    first_560_LA        10
    first_extend_LA     30
    first_SLUS_LA       50
;
```



通信管理プログラムの自動ログオン機能

SLU - TypeP2 は，NOTIFY コマンドが未サポートのため，HNA2 全体定義文（HNA2_configuration），HNA2 用 PU 定義文（HNA2_PU）および HNA2 用 LU 定義文（HNA2_LU）のどれにも，自動ログオン定義（auto_logon）に yes を指定できません。自動ログオン定義を省略するか，no を指定する必要があります。

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。

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 6-3　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 -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 6-4　コネクション（論理端末）を追加する場合に見直しが必要な OpenTP1 ファイルの一覧

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

また，SLU - TypeP2 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。

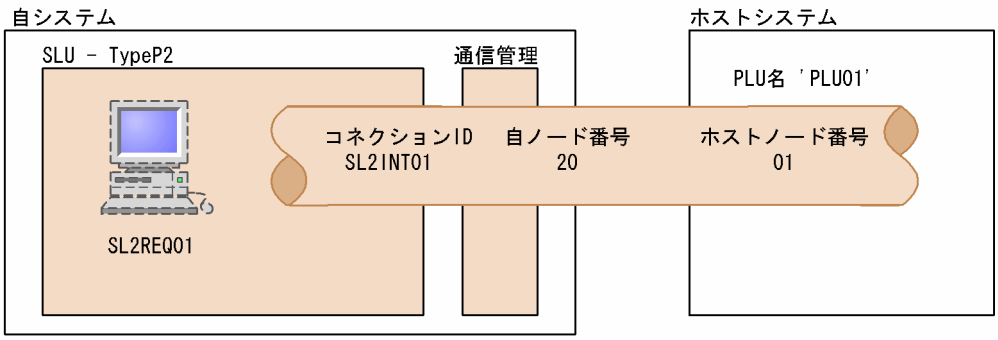
定義例

SLU - TypeP2 のシステム構成例を次の図に，その構成に沿った定義例をそのあとに示します。

SLU - TypeP2 では，この定義のコーディング例を次のファイルで提供しています。

- /BeTRAN/examples/mcf/SLUP2/conf/com_d

図 6-6　SLU - TypeP2 のシステム構成例




```
#################################################################
#          MCF COMMUNICATION CONFIGURATION DEFINITION           #
#                  for SLU - TypeP2 protocol                    #
#################################################################
#-------------Connection definition (mcftalccn)-----------#
mcftalccn   -c    SL2INT01                  ¥
            -p    slup2                     ¥
            -n   "ownnode    = 20"          ¥
            -q   "hostnode   = x'01'"       ¥
            -o  e'LOGONMOD'                 ¥
            -j   "pluname    = e'PLU01'"    ¥
            -g   "sndbuf     = 1            ¥
                  rcvbuf     = 2"           ¥
            -e   "msgbuf     = 3            ¥
                  count      = 10"          ¥
            -m   "mode       = xnfas"       ¥
            -i    auto                      ¥
            -b   "bretry     = yes          ¥
                  bretrycnt  = 10           ¥
                  bretryint  = 2"           ¥
            -k    ws                        ¥
            -d    rqe                       ¥
            -w   "firststsn = positive"     ¥
            -t   "termself = forced"        ¥
            -r   "sndrusiz   = 1024         ¥
                  rcvrusiz   = 1024"
#-------------------------------------------------------------#
########LE definition  (SL2REQ01)
mcftalcle   -l    SL2REQ01                  ¥
            -t    request                   ¥
            -m   "mmsgcnt    = 0            ¥
                  dmsgcnt    = 0"           ¥
```

```
            -k   ”quekind    = disk        ¥
                  quegrpid   = quegrp01”   ¥
            -o   ”aj         = yes”        ¥
            -v    slmhprv1
#-------------------------------------------------------------#
###Connection definition end  (SL2INT01)
mcftalced
```

7 運用コマンド

OpenTP1 システムは，運用コマンドを使用してシステムを運用できます。この章では，SLU - TypeP2 で使用する運用コマンドについて説明します。

SLU - TypeP2 の運用コマンド

SLU - TypeP2 は，オンライン中に運用コマンドを入力できます。SLU - TypeP2 で使用する運用コマンドを次の表に示し，それぞれについて説明します。

なお，ここでは，SLU - TypeP2 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，運用コマンドの入力方法，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。

表 7-1　SLU - TypeP2 で使用する運用コマンドの一覧

機能		コマンド名称	定義中組み込み	オフライン中に実行	オンライン中に実行	UAP から実行
コネクション管理	コネクションの確立	mcftactcn	×	×	○	○
	コネクションの解放	mcftdctcn	×	×	○	○
	コネクション状態表示	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。

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。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべてのコネクションを確立します。

先行文字列*：先行文字列で始まるすべてのコネクションを確立します。

〈複数指定の例〉 cnn1，cnn2，cnn3 を指定する場合

```
-c ”cnn1△cnn2△cnn3”
```

〈一括指定の例〉 cnn で始まるすべてのコネクションを指定する場合

```
-c ”cnn*”
```

出力メッセージ

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
KFCA15330-E	MCF 運用コマンド処理中に異常が発生しました。	標準エラー出力
KFCA15332-E	コネクションが確立済みのため運用コマンドは受け付けられません。	標準エラー出力
KFCA15333-E	コネクション確立処理中のため運用コマンドは受け付けられません。	標準エラー出力
KFCA15334-E	コネクション解放処理中のため運用コマンドは受け付けられません。	標準エラー出力
KFCA15342-E	ホスト起動方式のコネクションのため運用コマンドは受け付けられません。	標準エラー出力
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の名称を指定してください。

論理端末名称は一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数指定するときは引用符（"）で囲んで，論理端末名称と論理端末名称との間を空白で区切ります。同一論理端末名称は重複して指定できません。

また，論理端末名称は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外の論理端末名称を混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべての論理端末の閉塞を解除します。

先行文字列*：先行文字列で始まるすべての論理端末の閉塞を解除します。

〈複数指定の例〉 len1，len2，len3 を指定する場合

```
-l ”len1△len2△len3”
```

〈一括指定の例〉 len で始まるすべての論理端末を指定する場合

```
-l ”len*”
```

出力メッセージ

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	指定したコネクション ID と論理端末名称の対応が正しくありません。	標準エラー出力
KFCA10391-E	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
KFCA15330-E	MCF 運用コマンド処理中に異常が発生しました。	標準エラー出力

出力メッセージ ID	内容	出力先
KFCA15337-E	論理端末が活性化済みのため運用コマンドは受け付けられません。	標準エラー出力
KFCA15339-E	論理端末閉塞処理中のため運用コマンドは受け付けられません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- 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。

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。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべてのコネクションを解放します。

先行文字列*：先行文字列で始まるすべてのコネクションを解放します。

〈複数指定の例〉 cnn1，cnn2，cnn3 を指定する場合

```
-c ”cnn1△cnn2△cnn3”
```

〈一括指定の例〉 cnn で始まるすべてのコネクションを指定する場合

```
-c ”cnn*”
```

●-f

該当するコネクションを強制的に解放します。

このオプションを指定した場合，該当するコネクションが仕掛り中のとき，仕掛り中の処理を終了しないで強制的に解放します。

このオプションの指定を省略した場合，該当するコネクションが仕掛り中の場合，mcftdctcn コマンドはエラーとなります。

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
KFCA10501-I	ヘルプメッセージ	標準出力
KFCA15330-E	MCF 運用コマンド処理中に異常が発生しました。	標準エラー出力
KFCA15331-E	コネクションが未確立のため運用コマンドは受け付けられません。	標準エラー出力
KFCA15333-E	コネクション確立処理中のため運用コマンドは受け付けられません。	標準エラー出力
KFCA15334-E	コネクション解放処理中のため運用コマンドは受け付けられません。	標準エラー出力

出力メッセージ ID	内容	出力先
KFCA15341-E	コネクション使用中のため運用コマンドは受け付けられません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- mcftdctcn コマンドが正常に受け付けられたかどうかは，コマンドのリターン値で判断しないでください。コマンドが出力したメッセージの内容で判断してください。

mcftdctle（論理端末の閉塞）

形式

```
mcftdctle 〔-s MCF通信プロセス識別子〕〔-c コネクションID〕
          -l 論理端末名称
```

機能

論理端末を閉塞します。

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の名称を指定してください。

論理端末名称は一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数指定するときは引用符（"）で囲んで，論理端末名称と論理端末名称との間を空白で区切ります。同一論理端末名称は重複して指定できません。

また，論理端末名称は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外の論理端末名称を混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべての論理端末を閉塞します。

先行文字列*：先行文字列で始まるすべての論理端末を閉塞します。

〈複数指定の例〉 len1，len2，len3 を指定する場合

```
-l ”len1△len2△len3”
```

〈一括指定の例〉 len で始まるすべての論理端末を指定する場合

```
-l ”len*”
```

出力メッセージ

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	指定したコネクション ID と論理端末名称の対応が正しくありません。	標準エラー出力
KFCA10391-E	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
KFCA15330-E	MCF 運用コマンド処理中に異常が発生しました。	標準エラー出力

出力メッセージ ID	内容	出力先
KFCA15335-E	論理端末が閉塞済みのため運用コマンドは受け付けられません。	標準エラー出力
KFCA15338-E	論理端末使用中のため運用コマンドは受け付けられません。	標準エラー出力
KFCA15339-E	論理端末閉塞処理中のため運用コマンドは受け付けられません。	標準エラー出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

注意事項

- 受信仕掛り中に運用コマンド（mcftdctle）を入力した場合は，論理端末が閉塞状態でも，メッセージを受信します。論理端末閉塞によるメッセージ受信処理への影響はありません。
- 送信仕掛り中に運用コマンド（mcftdctle）を入力した場合は，運用コマンドがエラーリターンします。送信仕掛り中でない場合は，論理端末は閉塞されます。
- 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表示します。

オプション

●-s MCF 通信プロセス識別子　～<数字（0～9），a～f >((01～ef))

処理対象のコネクションを制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。

MCF 通信プロセス識別子は複数指定できません。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。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべてのコネクションの状態を表示します。

先行文字列*：先行文字列で始まるすべてのコネクションの状態を表示します。

〈複数指定の例〉 cnn1，cnn2，cnn3 を指定する場合

```
-c ”cnn1△cnn2△cnn3”
```

〈一括指定の例〉 cnn で始まるすべてのコネクションを指定する場合

```
-c ”cnn*”
```

●-d

コネクションの状態と該当するコネクションに対応する論理端末の情報を表示します。

このオプションの指定を省略すると，コネクションの状態だけを表示します。

出力形式

```
mmm cccccccc ppp sssss dddd
    llllllll ttt uuuu xxxx
```

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
  SL2…SLUTYPE-P 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
  SND…send 型
  RCV…receive 型
  REQ…request 型

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
KFCA10353-W	入力形式が誤っています。	標準エラー出力

出力メッセージ ID	内容	出力先
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表示します。

オプション

●-s MCF 通信プロセス識別子　～<数字（0～9），a～f >((01～ef))

処理対象の論理端末を制御する MCF 通信サービスの MCF 通信プロセス識別子を指定します。アプリケーション起動サービスのアプリケーション起動プロセス識別子は指定できません。

MCF 通信プロセス識別子は複数指定できません。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てください。

論理端末名称は一度に 8 個まで指定できます。多数入力する運用を行う場合は，次に示す複数指定または一括指定を使用して，一つの運用コマンドで行う並列処理数を増やし，運用コマンド入力数を減らすように運用設計を行ってください。

複数指定するときは引用符（"）で囲んで，論理端末名称と論理端末名称との間を空白で区切ります。同一論理端末名称は重複して指定できません。

また，論理端末名称は，*を使って一括指定ができます。一括指定は一つだけ指定できます。一括指定と一括指定以外の論理端末名称を混在して指定できません。一括指定をするときは，引用符（"）で囲んで指定します。

*：すべての論理端末の状態を表示します。

先行文字列*：先行文字列で始まるすべての論理端末の状態を表示します。

〈複数指定の例〉 len1，len2，len3 を指定する場合

```
-l ”len1△len2△len3”
```

〈一括指定の例〉 len で始まるすべての論理端末を指定する場合

```
-l ”len*”
```

●-q

指定した論理端末に対応する出力キューの保留状態を表示します。

このオプションを省略すると，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	指定したコネクション ID と論理端末名称の対応が正しくありません。	標準エラー出力
KFCA10391-E	mcftlsle コマンドはサポートされていません。	標準エラー出力
KFCA10395-E	指定したコネクションには未接続の論理端末名称が指定されています。	標準エラー出力

出力メッセージ ID	内容	出力先
KFCA10505-I	ヘルプメッセージ	標準出力
KFCA16402-E	RPC 障害が発生しました。	標準エラー出力

8 組み込み方法

この章では，SLU - TypeP2 を OpenTP1 システムに組み込む方法について説明します。

8.1　SLU - TypeP2 の組み込みの流れ

SLU - TypeP2 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

SLU - TypeP2 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

SLU - TypeP2 を実行するためには，MCF サービス名をシステムサービス構成定義で定義しておく必要があります。

MCF サービス名は MCF マネジャ定義オブジェクトファイル名と一致させてください。

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

SLU - TypeP2 は，OpenTP1 プロセスサービスによって起動されます。

SLU - TypeP2 を起動するためには，ユーザが MCF メイン関数をコーディングし，コンパイル，およびリンケージを行って SLU - TypeP2 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，mcfplslup2 コマンドを使用します。

MCF メイン関数からは，スタート関数（dc_mcf_svstart）を呼び出します。UOC を使用する場合は，MCF メイン関数で UOC の関数アドレスを指定してください。UOC は，MCF メイン関数と同じ言語（ANSI C，C++または K&R 版 C）で作成してください。

MCF メイン関数のコーディング概要を図 8-1 と図 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。

- /BeTRAN/examples/mcf/SLUP2/cmlib/ansi/com.c
- /BeTRAN/examples/mcf/SLUP2/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）

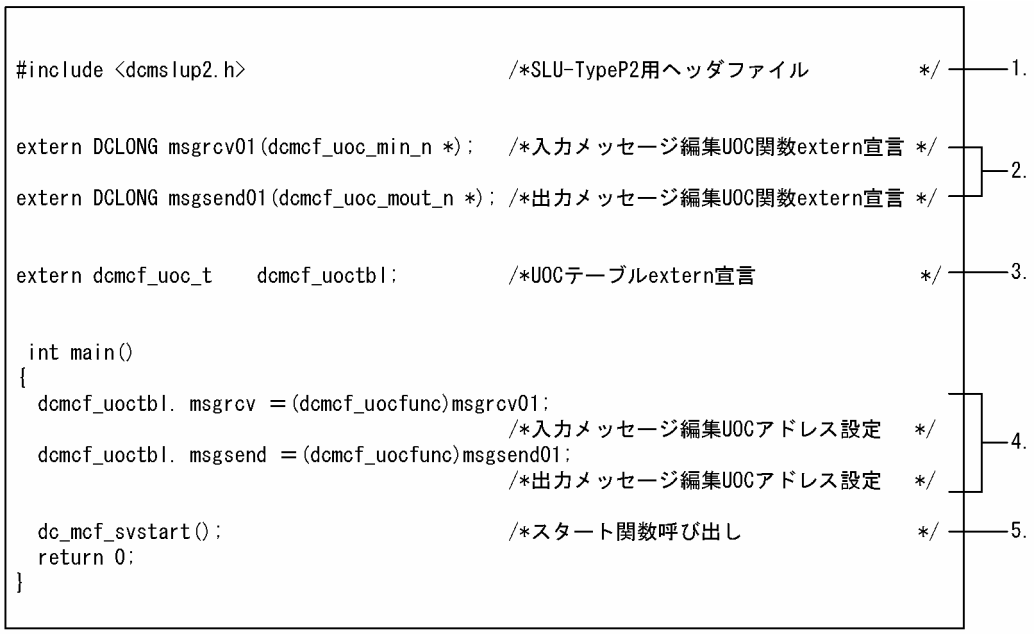
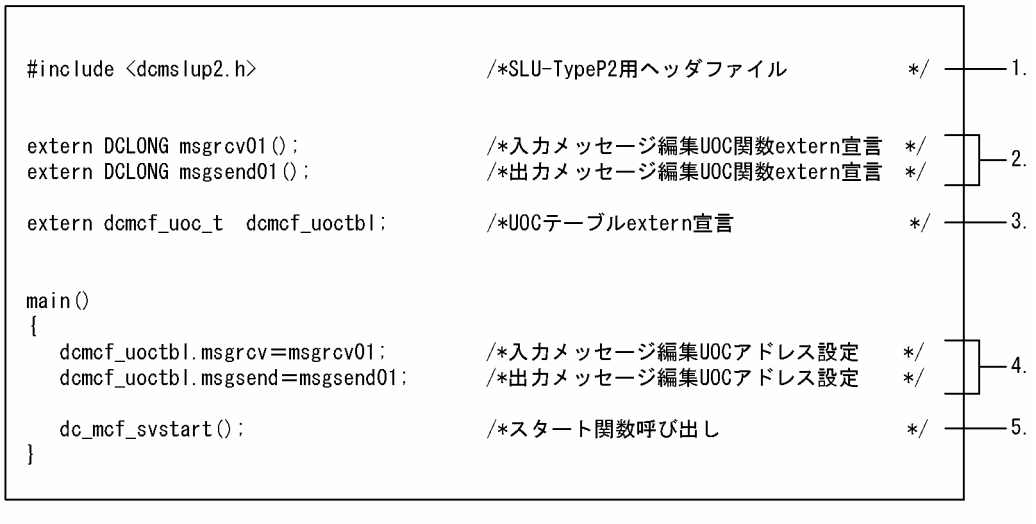


図 8-2　MCF メイン関数のコーディング概要（K&R 版 C の場合）



1. SLU - TypeP2 で提供するヘッダファイルを取り込みます。
2. 使用する UOC 関数を extern 宣言します。UOC のリターン値は DCLONG 型にしてください。

   UOC をまったく使用しない場合，このコーディングは必要ありません。
3. UOC テーブルを extern 宣言します。UOC を使用する場合，必ずこのとおりにコーディングしてください。

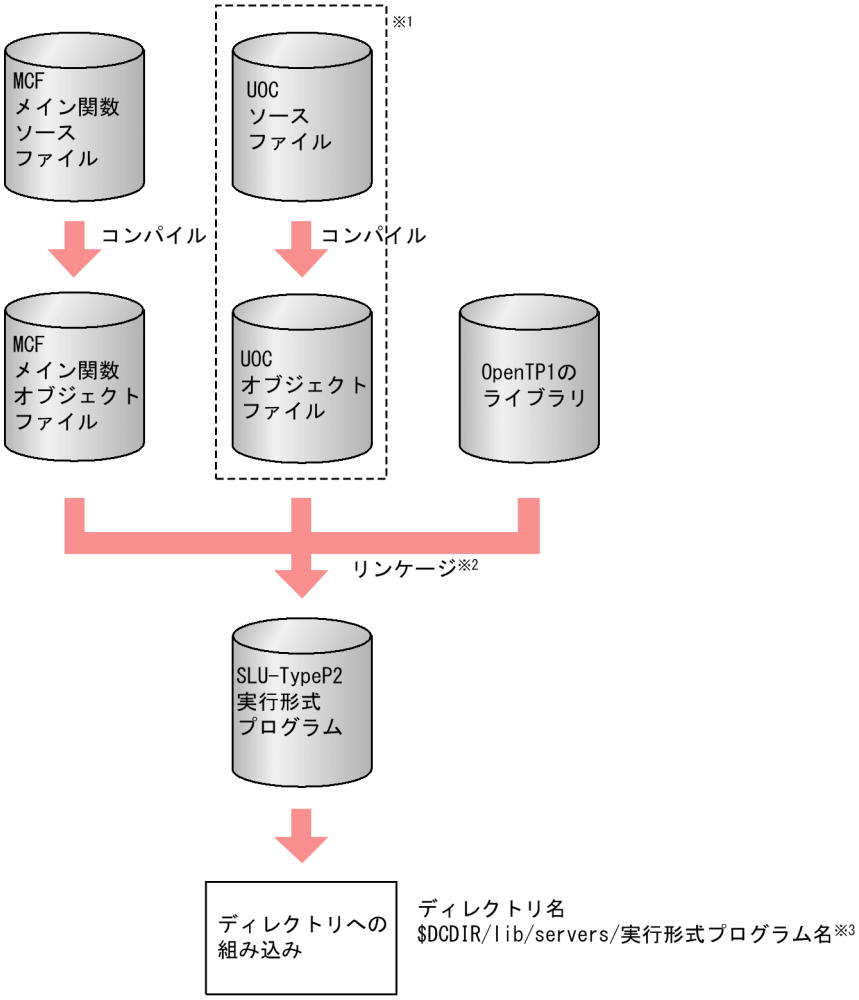
   UOC をまったく使用しない場合，このコーディングは必要ありません。
4. 各 UOC 関数のアドレスを，次に示すシステム提供変数に設定します。使用する UOC だけコーディングしてください。

   ```
   dcmcf_uoctbl.msgrcv　/*入力メッセージ編集UOCアドレス*/
   dcmcf_uoctbl.msgsend /*出力メッセージ編集UOCアドレス*/
   ```

   UOC をまったく使用しない場合，このコーディングは必要ありません。
5. スタート関数を呼び出します。MCF メイン関数には必ずコーディングしてください。

   スタート関数を呼び出したあとは，MCF メイン関数に制御が戻りません。そのため，スタート関数のあとにコーディングした処理は実行されませんので注意してください。

図 8-3　MCF メイン関数のディレクトリへの組み込み方法の概要



注※1

UOC を使用しない場合は，必要ありません。

注※2

mcfplslup2 コマンドでリンケージします。

mcfplslup2 コマンドの詳細については，SLU - TypeP2 の「リリースノート」を参照してください。

注※3

SLU - TypeP2 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

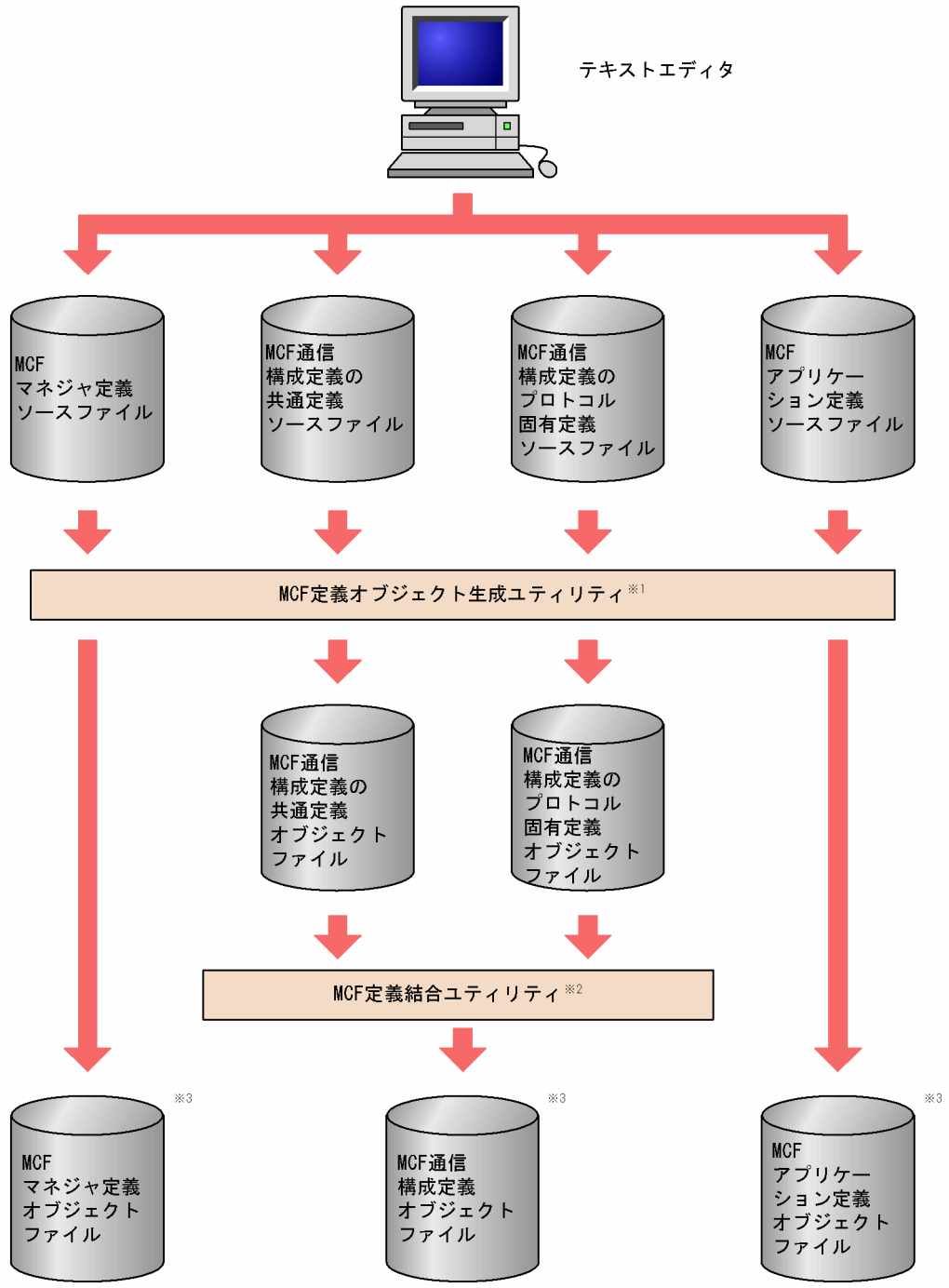
定義オブジェクトファイルを次の手順で生成します。

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てはいけません。変更した場合，再開始ができなくなることがあります。

1. テキストエディタを使用して，MCF の定義ファイルから，次に示す定義ソースファイルを作成します。
   - MCF マネジャ定義ソースファイル
   - MCF 通信構成定義の共通定義ソースファイル
   - MCF 通信構成定義の SLU - TypeP2 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 SLU - TypeP2 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

- mcfslup2：MCF 通信構成定義のプロトコル（SLU - TypeP2）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 mcfslup2 コマンドについては「MCF 定義オブジェクトの生成」を，その他のコマンドについてはマニュアル「OpenTP1 システム定義」を参照してください。

注※2

次に示すコマンドで，MCF 通信構成定義の二つのオブジェクトファイルを結合します。

```
mcflink△-i△共通定義オブジェクトファイル名
       △SLU - TypeP2定義オブジェクトファイル名
       △-o△出力オブジェクトファイル名
```

注※3

定義オブジェクトファイルは，システム環境定義の DCCONFPATH で指定したディレクトリに格納してください。システム環境定義については，マニュアル「OpenTP1 システム定義」を参照してください。

9 障害対策

この章では，SLU - TypeP2 運用中に発生する障害と，SLU - TypeP2 の対応処理，およびメッセージの処理について説明します。

9.1　障害の種類と対応処理

運用中に障害が発生すると，SLU - TypeP2 はシステムを回復します。このとき，システム定義を指定すると，MCF イベント処理用 MHP を起動することもできます。

9.1.1　SLU - TypeP2 運用中の障害と対応処理

SLU - TypeP2 運用中の障害と対応処理について，障害の種類ごとに示します。

また，センスコードの詳細は，「付録 H.3　SLU - TypeP2 が使用するセンスコード」を参照してください。

(1)　コネクション障害

コネクション障害の発生個所に応じた SLU - TypeP2 の障害処理については，「9.2　コネクション障害」を参照してください。

表 9-1　コネクション障害発生時の処理

障害の内容	SLU - TypeP2 の処理	ユーザの処置
通信管理からの H2_ABORT 受信など	1. 該当する論理端末ごとの障害処理をします（「9.1.2　論理端末ごとの障害処理」参照）。 2. コネクション障害を通知するメッセージログ（KFCA15121-E）を出力します。	端末起動の場合は次のどちらかの方法で再び確立処理をします。 ・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 ホスト起動の場合はホスト側から確立要求をするようにしてください。
通信管理マクロエラーリターン	1. 該当する論理端末ごとの障害処理をします（「9.1.2　論理端末ごとの障害処理」参照）。 2. コネクション障害を通知するメッセージログ（KFCA15120-E）を出力します。	
セションパラメタ不正の BIND コマンドを受信	1. -RSP を送信します。センスコードを次に示します。 セションパラメタ不正：08210000 2. -RSP 送信したことを通知するメッセージログ（KFCA15204-E）を出力します。 3. CERREVT（コネクション障害，下位層障害）を起動します。 4. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 5. コネクション確立処理を中断します。	SLU-TypeP2 の定義誤りの場合，誤りを訂正し，定義オブジェクトを作り直して OpenTP1 を再開始します。

(2)　論理端末障害

論理端末の端末タイプごとに処理が異なります。

(a) request 型

表 9-2 request 型論理端末の障害発生時の処理

障害の内容	SLU - TypeP2 の処理	ユーザの処置
メッセージ送信完了エラー，拒否応答（-RSP）受信	1. UAP にエラーリターンします。 リターン値 DCMCFRTN_73001（同期）を返します。 2. CERREVT（論理端末障害，送信中断）を起動します。 3. 出力キューを削除します（同期）。 4. 再送準備を要求します（非同期）。 5. 論理端末を閉塞します。 6. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 7. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。	障害要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する
出力キュー読み込み障害，出力メッセージ編集 UOC エラーリターン，送信バッファサイズ不足	1. UAP にエラーリターンします。 リターン値 DCMCFRTN_73001（同期）を返します。 2. CERREVT（論理端末障害）を起動します。 3. 出力キューを削除します（同期）。 4. 再送準備を要求します（非同期）。 5. 論理端末を閉塞します。 6. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 7. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 8. 内部処理障害を通知するメッセージログ（KFCA15396-E）を出力します。	出力キュー読み込み障害 障害要因を取り除いてください。 出力メッセージ編集 UOC エラーリターン 出力メッセージ編集 UOC 処理を見直してください。 送信バッファサイズ不足 システム定義を修正してください。
送信メッセージ削除失敗	処理を続行します。	ありません。
タイムアウト	1. UAP にエラーリターンします。 リターン値 DCMCFRTN_73005（同期）を返します。 2. 出力キューを削除します（同期）。 3. 論理端末を閉塞します。 4. コネクションを切断します。 5. CERREVT（コネクション障害，タイムアウト）を起動します。 6. セション解放を通知するメッセージログ（KFCA15202-I）を出力します。 7. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 8. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。	端末起動の場合は次のどちらかの方法で再び確立処理をします。 ・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 ホスト起動の場合はホスト側から確立要求をするようにしてください。
不正応答メッセージ受信（ブラケット違反，チェイン違反，RU 長不正）	1. UAP にエラーリターンします。 リターン値 DCMCFRTN_73001（同期）を返します。 2. 出力キューを削除します（同期）。	相手システムへ不正応答メッセージを受信したことを連絡してください。

障害の内容	SLU - TypeP2 の処理	ユーザの処置
不正応答メッセージ受信（ブラケット違反，チェイン違反，RU 長不正）	3. -RSP を送信します。 センスコードを次に示します。 ・ブラケット違反：20030000 ・チェイン違反：20020000 ・RU 長不正：10020000 4. -RSP 送信したことを通知するメッセージログ（KFCA15204-E）を出力します。 5. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。	相手システムへ不正応答メッセージを受信したことを連絡してください。
入力キュー書き込み障害，アプリケーションスケジュール失敗，入力メッセージ編集 UOC エラーリターン	1. CERREVT（論理端末障害，入力キュー障害）を起動します。 2. 論理端末を閉塞します。 3. -RSP を送信します。 センスコードを次に示します。 ・要求不実行：08000000 4. -RSP 送信したことを通知するメッセージログ（KFCA15204-E）を出力します。 5. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 6. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 7. 内部処理障害を通知するメッセージログ（KFCA15396-E）を出力します。	障害要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する
送信仕掛り中の mcftdctle 入力	1. コマンドをエラーリターンします。 2. 論理端末の使用中を通知するメッセージログ（KFCA15338-E）を出力します。	ありません。
応答ダミーデータ受信	1. UAP にエラーリターンします。 リターン値 DCMCFRTN_73005 を返します。	ありません。
メッセージコンテンション （ブラケット開始拒否）	1. -RSP を送信します。 センスコードを次に示します。 ・ブラケット開始拒否：08130000，08140000 2. -RSP 送信したことを通知するメッセージログ（KFCA15204-E）を出力します。	ありません。

(b) send 型

表 9-3 send 型論理端末の障害発生時の処理

障害の内容	SLU - TypeP2 の処理	ユーザの処置
メッセージ送信完了エラー，拒否応答（-RSP）受信	1. -RSP 受信したことを通知するメッセージログ（KFCA15205-E）を出力します。 2. CERREVT（論理端末障害，送信中断）を起動します。 3. 再送準備を要求します（非同期）。	障害要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。

障害の内容	SLU - TypeP2 の処理	ユーザの処置
メッセージ送信完了エラー，拒否応答（-RSP）受信	4. 論理端末を閉塞します。 5. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 6. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。	・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
出力キュー読み込み障害，出力メッセージ編集 UOC エラーリターン，送信バッファサイズ不足	1. CERREVT（論理端末障害）を起動します。 2. 論理端末を閉塞します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 4. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 5. 内部処理障害を通知するメッセージログ（KFCA15396-E）を出力します。 6. 再送準備を要求します。	出力キュー読み込み障害 障害要因を取り除いてください。 出力メッセージ編集 UOC エラーリターン 出力メッセージ編集 UOC 処理を見直してください。 送信バッファサイズ不足 システム定義を修正してください。
送信メッセージ削除失敗	処理を続行します。	ありません。
送信仕掛り中の mcftdctle 入力	1. コマンドをエラーリターンします。 2. 論理端末の使用中を通知するメッセージログ（KFCA15338-E）を出力します。	ありません。
メッセージコンテンション（ブラケット開始拒否）	1. -RSP を送信します。 センスコードを次に示します。 ・ブラケット開始拒否：08130000，08140000 2. -RSP 送信したことを通知するメッセージログ（KFCA15204-E）を出力します。	ありません。

(c) receive 型

表 9–4 receive 型論理端末の障害発生時の処理

障害の内容	SLU - TypeP2 の処理	ユーザの処置
入力キュー書き込み障害，アプリケーションスケジュール失敗，入力メッセージ編集 UOC エラーリターン	1. CERREVT（論理端末障害）を起動します。 2. -RSP を送信します。 センスコードを次に示します。 ・要求不実行：08000000 3. -RSP 受信したことを通知するメッセージログ（KFCA15205-E）を出力します。 4. 論理端末を閉塞します。 5. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 6. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 7. 内部処理障害を通知するメッセージログ（KFCA15396-E）を出力します。 8. 再送準備を要求します。	障害要因を取り除いたあと，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する

(3) その他の障害

表 9-5 論理端末の型に共通する，その他の障害発生時の処理

障害の種類	SLU - TypeP2 の処理	ユーザの処置
受信メッセージと論理端末の端末タイプの不一致※1	1. 受信メッセージを破棄します。 2. -RSP を送信します。 センスコードを次に示します。 ・要求不実行：08000000 3. -RSP 受信したことを通知するメッセージログ（KFCA15205-E）を出力します。 4. CERREVT（論理端末障害）を起動します。 5. 論理端末を閉塞します。 6. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。	ホスト側との構成を見直してください。
送信メッセージと論理端末の端末タイプの不一致※2	UAP にエラーリターンします。	UAP で後処理をする必要があります。
送信メッセージ消去失敗	処理を続行します。	ありません。
送信バッファ面数不足	1. メモリダンプを出力します。 2. コネクションを切断します。 3. CERREVT（コネクション障害）を起動します。 4. コネクション強制解放を通知するメッセージログ（KFCA15398-E）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（KFCA15302-E）を出力します。	バッファ面数を増やして OpenTP1 を再開始します。
受信バッファ面数不足	1. メモリダンプを出力します。 2. コネクションを切断します。 3. CERREVT（コネクション障害）を起動します。 4. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。	バッファ面数を増やして OpenTP1 を再開始します。
同期送受信（sendrecv）中に UAP 終了	1. 論理端末障害を通知する通知するメッセージログ（KFCA15303-E）を出力します。 2. 論理端末を閉塞するメッセージログ（KFCA15396-E）または処理を続行するメッセージログ（KFCA15397-E）を出力します。 3. メモリダンプを取得します。 （論理端末を閉塞する場合） 4. CERREVT（論理端末障害）を起動します。 5. 論理端末を閉塞します。	UAP を見直し，障害要因を取り除いたあと，論理端末が閉塞されている場合は，次のどちらかの方法で論理端末を閉塞解除します。 ・運用コマンド（mcftactle）を入力する ・API（dc_mcf_tactle 関数または CBLDCMCF('TACTLE△△')）を発行する
内部障害	1. メモリダンプを出力します。 2. 内部処理実行中異常を通知するメッセージログ（KFCA15399-E）を出力します。	OpenTP1 を再開始します。

障害の種類	SLU - TypeP2 の処理	ユーザの処置
内部障害	3. プロセスを異常終了します。	OpenTP1 を再開始します。

注※1

例えば，send 型の論理端末にメッセージ受信した場合に発生します。

注※2

例えば，次の場合に発生します。

- receive 型の論理端末に対して送信要求（send，sendrecv）を実行した場合
- send 型の論理端末に対して同期送受信関数（sendrecv）を実行した場合

9.1.2　論理端末ごとの障害処理

SLU - TypeP2 運用中の障害と対応処理について，論理端末ごとに示します。

(1)　request 型

表 9-6　request 型論理端末の障害処理

障害の内容		SLU - TypeP2 の処理
下位切断	非同期送信仕掛り中	1. 再送準備を要求します。 2. CERREVT（論理端末障害，下位層障害）を起動します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 6. CERREVT（コネクション障害，下位層障害）を起動します。 7. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 8. コネクションを切断します。
	同期送受信仕掛り中	1. 同期送受信タイマを解除します。 2. UAP をエラーリターンします。 リターン値 DCMCFRTN_73001（同期）を返します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 6. CERREVT（コネクション障害，下位層障害）を起動します。 7. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。

障害の内容		SLU - TypeP2 の処理
下位切断	同期送受信仕掛り中	8. コネクションを切断します。
	上記以外の場合	1. 論理端末を閉塞します。 2. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 3. CERREVT（コネクション障害，下位層障害）を起動します。 4. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 5. コネクションを切断します。
・mcftdctcn -f 入力による強制解放 ・API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）発行による強制解放	非同期送信仕掛り中	1. 再送準備を要求します。 2. CERREVT（論理端末障害，強制解放）を起動します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 6. CERREVT（コネクション障害，強制解放）を起動します。 7. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 8. コネクションを切断します。
	同期送受信仕掛り中	1. 同期送受信タイマを解除します。 2. UAP をエラーリターンします。 リターン値 DCMCFRTN_73001（同期）を返します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 6. CERREVT（コネクション障害，強制解放）を起動します。 7. コネクション障害を通知するメッセージログ（KFCA15302 -E）を出力します。 8. コネクションを切断します。
	上記以外の場合	1. 論理端末を閉塞します。 2. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 3. CERREVT（コネクション障害，強制解放）を起動します。 4. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 5. コネクションを切断します。
LU-LU セション解放	非同期送信仕掛り中	1. 再送準備を要求します。 2. CERREVT（論理端末障害，送信中断）を起動します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。

障害の内容		SLU - TypeP2 の処理
LU-LU セション解放	非同期送信仕掛り中	4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 6. CERREVT（コネクション障害，送信中断）を起動します。 7. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 8. コネクションを切断します。
	同期送受信仕掛り中	1. 同期送受信タイマを解除します。 2. UAP をエラーリターンします。 リターン値 DCMCFRTN_73001（同期）を返します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 6. CERREVT（コネクション障害，送信中断）を起動します。 7. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 8. コネクションを切断します。

(2) send 型

表 9–7 send 型論理端末の障害処理

障害の内容		SLU - TypeP2 の処理
下位切断	非同期送信仕掛り中	1. 再送準備を要求します。 2. CERREVT（論理端末障害，下位層障害）を起動します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 6. CERREVT（コネクション障害，下位層障害）を起動します。 7. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 8. コネクションを切断します。
	上記以外の場合	1. 論理端末を閉塞します。 2. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 3. CERREVT（コネクション障害，下位層障害）を起動します。 4. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 5. コネクションを切断します。

障害の内容		SLU - TypeP2 の処理
・mcftdctcn -f 入力による強制解放 ・API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）発行による強制解放	非同期送信仕掛り中	1. 再送準備を要求します。 2. CERREVT（論理端末障害，強制解放）を起動します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 6. CERREVT（コネクション障害，強制解放）を起動します。 7. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 8. コネクションを切断します。
	上記以外の場合	1. 論理端末を閉塞します。 2. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 3. CERREVT（コネクション障害，強制解放）を起動します。 4. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 5. コネクションを切断します。
LU-LU セション解放	非同期送信仕掛り中	1. 再送準備を要求します。 2. CERREVT（論理端末障害，送信中断）を起動します。 3. 論理端末障害を通知するメッセージログ（KFCA15303-E）を出力します。 4. 論理端末を閉塞します。 5. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 6. CERREVT（コネクション障害，送信中断）を起動します。 7. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 8. コネクションを切断します。

(3) recieve 型

表 9–8 receive 型論理端末の障害処理

障害の内容	SLU - TypeP2 の処理
下位切断	1. 論理端末を閉塞します。 2. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 3. CERREVT（コネクション障害，下位層障害）を起動します。 4. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 5. コネクションを切断します。
・mcftdctcn -f 入力による強制解放	1. 論理端末を閉塞します。

障害の内容	SLU - TypeP2 の処理
・API（dc_mcf_tdctcn 関数または CBLDCMCF('TDCTCN△△')）発行による強制解放	2. 論理端末閉塞を通知するメッセージログ（KFCA15312-I）を出力します。 3. CERREVT（コネクション障害，強制解放）を起動します。 4. コネクション障害を通知するメッセージログ（KFCA15302-E）を出力します。 5. コネクションを切断します。

9.2　コネクション障害

コネクションが切断された場合，端末起動型のシステムでは，運用コマンド（mcftactcn）を入力するか，API（dc_mcf_tactcn 関数または CBLDCMCF('TACTCN△△')）を発行するかしてコネクションを再確立します。ホスト起動型のシステムでは，自動的に相手システムからのコネクション確立を待ちます。自システムで管理している通信機器に障害が発生した場合も，同様にコネクション障害として処理します。

メッセージ送受信時にコネクション障害が発生した場合，メッセージは破棄されます。

コネクションの処理中に，障害の発生する個所の区分を次の図に示します。また，障害発生個所による SLU - TypeP2 の障害処理について次の表に示します。

図 9-1　コネクション障害

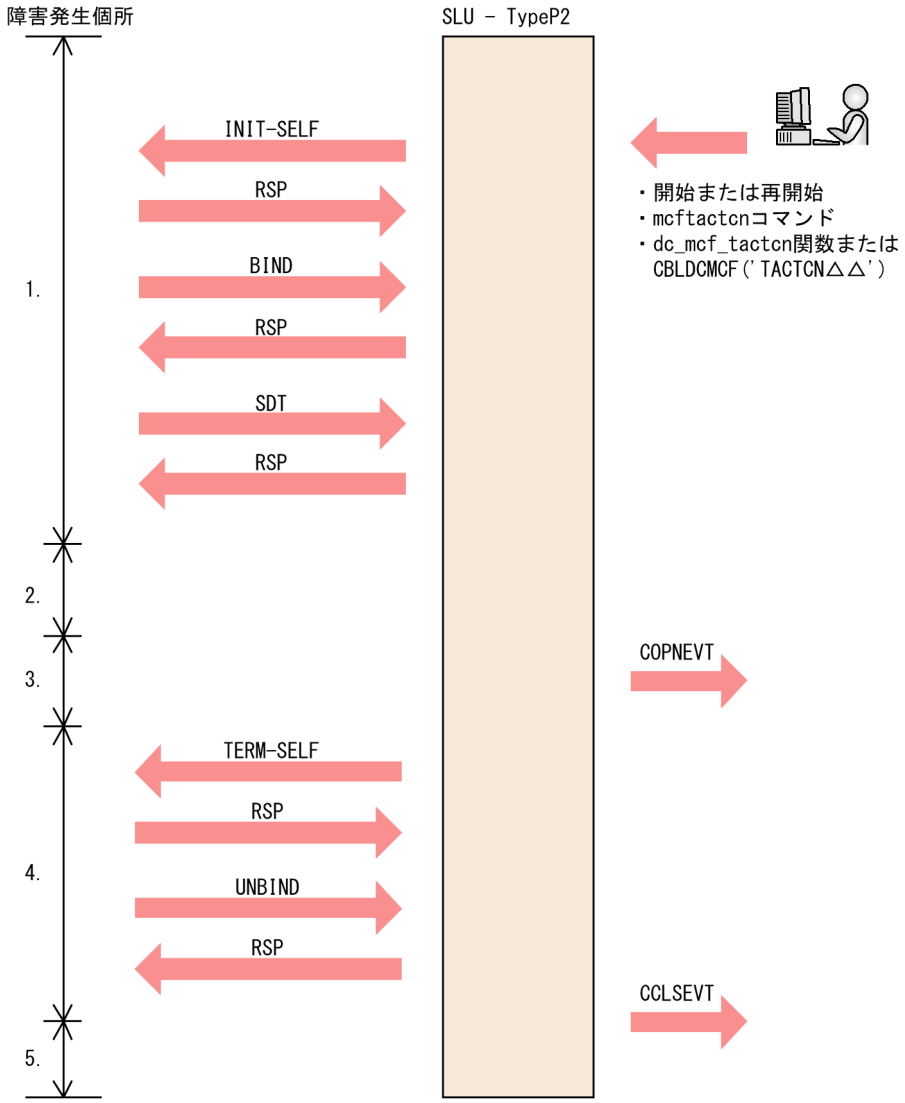


表 9-9　コネクション障害の処理

障害発生個所	内容	メッセージログの出力内容	SLU - TypeP2 の処理
1.	コネクション確立時のリトライオーバ（下位層の障害，またはタイムアウトによるリトライ）	コネクション障害	• CERREVT を起動します。 • コネクションを解放します。 解放後は運用コマンド（mcftactcn）または API（dc_mcf_tactcn 関数もしくは CBLDCMCF('TACTCN△△')）で再確立できます。
2.	入力キュー書き込みエラー（COPNEVT）	メッセージ入力障害	• メッセージを破棄します。
3.	コネクション確立後のコネクション障害	コネクション障害	• CERREVT を起動します。 • コネクションを解放します。 解放後は運用コマンド（mcftactcn）または API（dc_mcf_tactcn 関数もしくは CBLDCMCF('TACTCN△△')）で再確立できます。
4.	コネクション解放中のコネクション障害	コネクション障害	• CERREVT を起動します。 • コネクションを解放します。 解放後は運用コマンド（mcftactcn）または API（dc_mcf_tactcn 関数もしくは CBLDCMCF('TACTCN△△')）で再確立できます。
5.	入力キュー書き込みエラー（CCLSEVT）	メッセージ入力障害	• メッセージを破棄します。

9.3　論理端末障害

論理端末障害が発生したときの処理は，コネクション障害の場合の処理と同じです。処理の詳細については，「9.1　障害の種類と対応処理」および「9.2　コネクション障害」を参照してください。

付録

付録 A　バージョンアップ時の変更点

各バージョンでの変更点を次に示す分類ごとに示します。

- 関数，定義およびコマンドの追加・変更・削除
- 動作の変更
- 関数，定義およびコマンドのデフォルト値の変更

付録 A.1　07-50 での変更点

SLU - TypeP2 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　SLU - TypeP2 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	コネクション定義の開始（mcftalccn） • -k オプションの指定値に auto を追加
	コマンド	mcfslup2 コマンド • -r オプション
変更		なし
削除	関数	なし
	定義	なし
	コマンド	なし

SLU - TypeP2 07-50 での動作の変更点はありません。

SLU - TypeP2 07-50 でのデフォルト値の変更はありません。

付録 A.2　07-00 での変更点

SLU - TypeP2 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。

SLU - TypeP2 07-00 での動作の変更点はありません。

SLU - TypeP2 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_receive	3.　dc_mcf_receive － 一方送信メッセージの受信（C 言語）
	dc_mcf_recvsync	3.　dc_mcf_recvsync － 同期型の応答メッセージの受信（C 言語）
	dc_mcf_resend	3.　dc_mcf_resend － メッセージの再送（C 言語）
	dc_mcf_send	3.　dc_mcf_send － 一方送信メッセージの送信（C 言語）
	dc_mcf_sendrecv	3.　dc_mcf_sendrecv － 同期型の問い合わせ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_receive － 一方送信メッセージの受信

(a)　ANSI C，C++の形式

バージョン 6 以前

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

バージョン 7

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

(b) K&R 版 C の形式

バージョン 6 以前

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

バージョン 7

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

(2) dc_mcf_recvsync － 同期型の応答メッセージの受信

(a) ANSI C，C++の形式

バージョン 6 以前

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

バージョン 7

```
#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

```
#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;
```

バージョン 7

```
#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;
```

バージョン 6 以前	バージョン 7
long *time; long resv02;	DCLONG *time; DCLONG resv02;

(3) dc_mcf_resend － メッセージの再送

(a) ANSI C，C++の形式

バージョン 6 以前	バージョン 7
#include <dcmcf.h> int dc_mcf_resend(long action, long commform, char *rtermnam, char *resv01, long oseqid, long orgseq, char *otermnam, char *resv02, char *resv03, char *resv04, long opcd)	#include <dcmcf.h> int dc_mcf_resend(DCLONG action, DCLONG commform, char *rtermnam, char *resv01, DCLONG oseqid, DCLONG orgseq, char *otermnam, char *resv02, char *resv03, char *resv04, DCLONG opcd)

(b) K&R 版 C の形式

バージョン 6 以前	バージョン 7
#include <dcmcf.h> int dc_mcf_resend(action, commform, rtermnam, resv01, oseqid, orgseq, otermnam, resv02, resv03, resv04, opcd) long action; long commform; char *rtermnam; char *resv01; long oseqid; long orgseq; char *otermnam; char *resv02; char *resv03; char *resv04; long opcd;	#include <dcmcf.h> int dc_mcf_resend(action, commform, rtermnam, resv01, oseqid, orgseq, otermnam, resv02, resv03, resv04, opcd) DCLONG action; DCLONG commform; char *rtermnam; char *resv01; DCLONG oseqid; DCLONG orgseq; char *otermnam; char *resv02; char *resv03; char *resv04; DCLONG opcd;

(4) dc_mcf_send － 一方送信メッセージの送信

(a) ANSI C，C++の形式

バージョン６以前

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

バージョン７

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

(b) K&R 版 C の形式

バージョン６以前

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

バージョン７

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

(5) dc_mcf_sendrecv － 同期型の問い合わせ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 watchtime)
```

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 watchtime)
```

(b) K&R 版 C の形式

バージョン 6 以前	バージョン 7

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

付録 C.2 ユーザオウンコーディング

(1) 入力メッセージ編集 UOC

(a) 形式

ANSI C，C++の形式

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

K&R 版 C の形式

バージョン 6 以前	バージョン 7

`#include <dcmcfuoc.h>`
`long uoc_func(parm)`
 	
`dcmcf_uoc_min_n   *parm ;`
	`#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	バージョン 7
typedef struct { long pro_kind; char le_name[9]; char reserve1[7]; long rcv_prim; dcmcf_uocbuff_list_n *buflist_adr; dcmcf_uocbuff_list_n *ebuflist_adr; char aplname[9]; char reserve2[7]; char *pro_indv_ifa; long rtn_detail; char reserve3[8]; } dcmcf_uoc_min_n;	typedef struct { DCLONG pro_kind; char le_name[9]; char reserve1[7]; DCLONG rcv_prim; dcmcf_uocbuff_list_n *buflist_adr; dcmcf_uocbuff_list_n *ebuflist_adr; char aplname[9]; char reserve2[7]; char *pro_indv_ifa; DCLONG rtn_detail; char reserve3[8]; } dcmcf_uoc_min_n;

dcmcf_uocbuff_list_n（バッファリスト）の内容

バージョン 6 以前	バージョン 7
typedef struct { long buf_num; long used_buf_num; char reserve1[8]; dcmcf_uocbufinf_n buf_array[DCMCF_UOC_BUFF_MAX]; } dcmcf_uocbuff_list_n;	typedef struct { DCLONG buf_num; DCLONG used_buf_num; char reserve1[8]; dcmcf_uocbufinf_n buf_array[DCMCF_UOC_BUFF_MAX]; } dcmcf_uocbuff_list_n;

dcmcf_uocbufinf_n（バッファ情報）の内容

バージョン 6 以前	バージョン 7
typedef struct { char *buf_adr; unsigned long buf_size; unsigned long seg_size; char reserve1[4]; dcmcfuoc_w_type buff_id; long buff_addr; char reserve2[4]; } dcmcf_uocbufinf_n;	typedef struct { char *buf_adr; DCULONG buf_size; DCULONG seg_size; char reserve1[4]; dcmcfuoc_w_type buff_id; DCMLONG buff_addr; char reserve2[4]; } dcmcf_uocbufinf_n;

(2) 出力メッセージ編集 UOC

(a) 形式

ANSI C，C++の形式

バージョン 6 以前	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	バージョン 7

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

K&R 版 C の形式

バージョン 6 以前	バージョン 7

`#include <dcmcf.h>`
`long send_uoc(flags,`
`termname,`
`sendno,`
`sendid,`
`dataleng,`
`senddata)`
`long flags;`
`char *termname;`
`long sendno;`
`long sendid;`
`long dataleng;`
`char *senddata;`
	`#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	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;`
`long time;`
`};`
	`struct dc_mcf_evtheader {`
	`char mcfevt_name[9];`
	`char le_name[16];`
	`char cn_name[9];`
	`unsigned char format_kind;`
	`char reserve01;`
	`DCLONG time;`
	`};`

(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;`	`struct dc_mcf_evta_type {` `struct dc_mcf_evtheader evtheader ;`

バージョン 6 以前	バージョン 7
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]; };	char reserve01[12]; char reserve02[10]; char reserve03[2]; char ap_name[10]; char reserve04[2]; char reserve05[32]; char reserve06[32]; DCLONG user_leng; char user_data[16]; char reserve07[16]; };

(6) CERREVT の形式

バージョン 6 以前	バージョン 7
typedef struct{ struct dc_mcf_evtheader header; long err_fact; long err_reason1; long err_reason2; long err_rcv_action; char reserve1[42]; }dcmslm_cerrevt;	typedef struct{ struct dc_mcf_evtheader header; DCLONG err_fact; DCLONG err_reason1; DCLONG err_reason2; DCLONG err_rcv_action; char reserve1[42]; }dcmslm_cerrevt;

(7) COPNEVT，CCLSEVT の形式

バージョン 6 以前	バージョン 7
typedef struct{ struct dc_mcf_evtheader header; long cls_rcv_action; char reserve1[54]; }dcmslm_statevt;	typedef struct{ struct dc_mcf_evtheader header; DCLONG cls_rcv_action; char reserve1[54]; }dcmslm_statevt;

付録 C.4 MCF メイン関数のコーディング概要

MCF メイン関数のコーディング概要の変更一覧を示します。変更個所は，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 <dcmslup2.h>                          /*SLU-TypeP2用ヘッダファイル          */

extern long  msgrcv01(dcmcf_uoc_min_n *);   /*入力メッセージ編集UOC関数extern宣言 */
extern long  msgsend01(dcmcf_uoc_mout_n *); /*出力メッセージ編集UOC関数extern宣言 */

extern dcmcf_uoc_t    dcmcf_uoctbl;          /*UOCテーブルextern宣言               */

 int main()
{
  dcmcf_uoctbl. msgrcv =(dcmcf_uocfunc)msgrcv01;
                                           /*入力メッセージ編集UOCアドレス設定   */
  dcmcf_uoctbl. msgsend =(dcmcf_uocfunc)msgsend01;
                                           /*出力メッセージ編集UOCアドレス設定   */
  dc_mcf_svstart();                        /*スタート関数呼び出し                */
  return 0;
}
```

(b) バージョン７

```
#include <dcmslup2.h>                          /*SLU-TypeP2用ヘッダファイル          */

extern DCLONG msgrcv01(dcmcf_uoc_min_n *);   /*入力メッセージ編集UOC関数extern宣言 */
extern DCLONG msgsend01(dcmcf_uoc_mout_n *); /*出力メッセージ編集UOC関数extern宣言 */

extern dcmcf_uoc_t    dcmcf_uoctbl;          /*UOCテーブルextern宣言               */

 int main()
{
  dcmcf_uoctbl. msgrcv =(dcmcf_uocfunc)msgrcv01;
                                           /*入力メッセージ編集UOCアドレス設定   */
  dcmcf_uoctbl. msgsend =(dcmcf_uocfunc)msgsend01;
                                           /*出力メッセージ編集UOCアドレス設定   */
  dc_mcf_svstart();                        /*スタート関数呼び出し                */
  return 0;
}
```

(2)　K&R 版 C の場合

(a)　バージョン 6 以前

```
#include <dcmslup2.h>                    /*SLU-TypeP2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                */
}
```

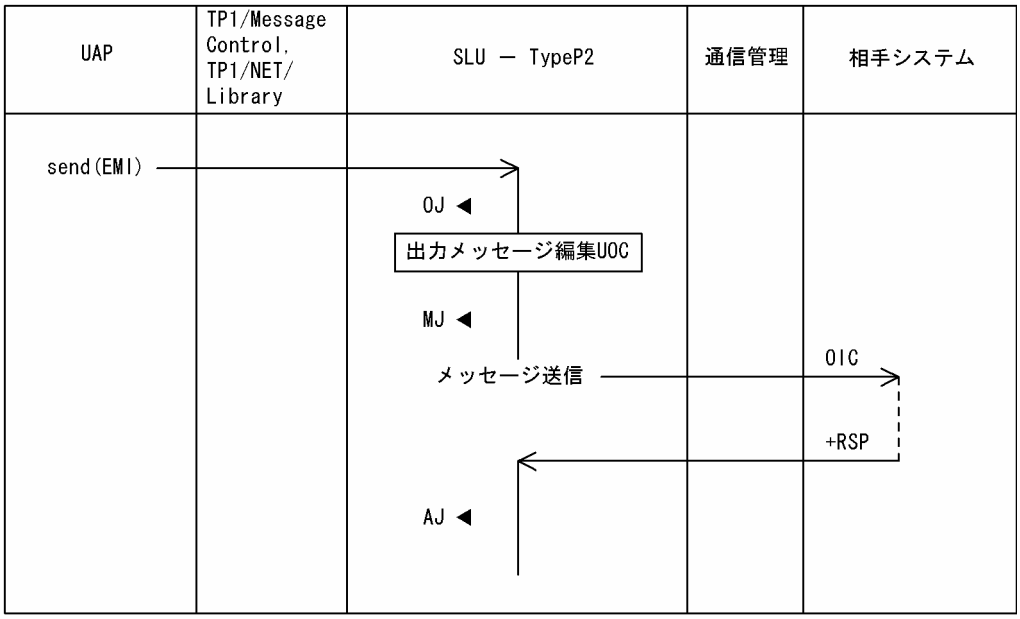
(b)　バージョン 7

```
#include <dcmslup2.h>                    /*SLU-TypeP2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

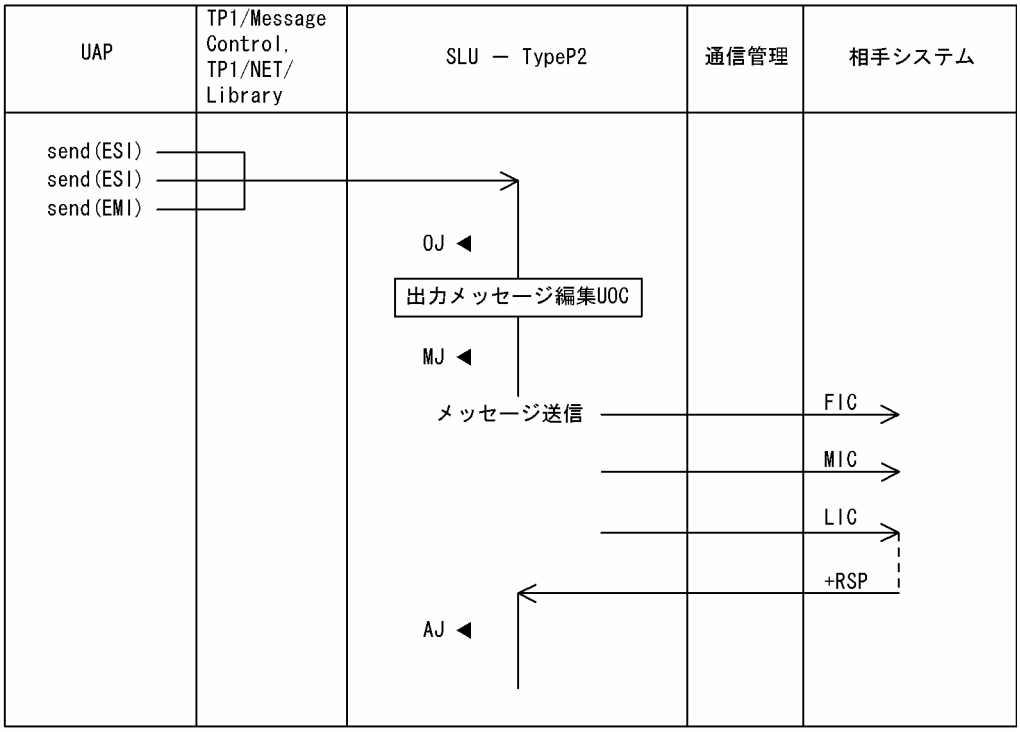
メッセージを送受信するときの処理の流れを図 D-1～図 D-6 に示します。

図 D-1　一方送信メッセージ（send 型／単独セグメントの場合）



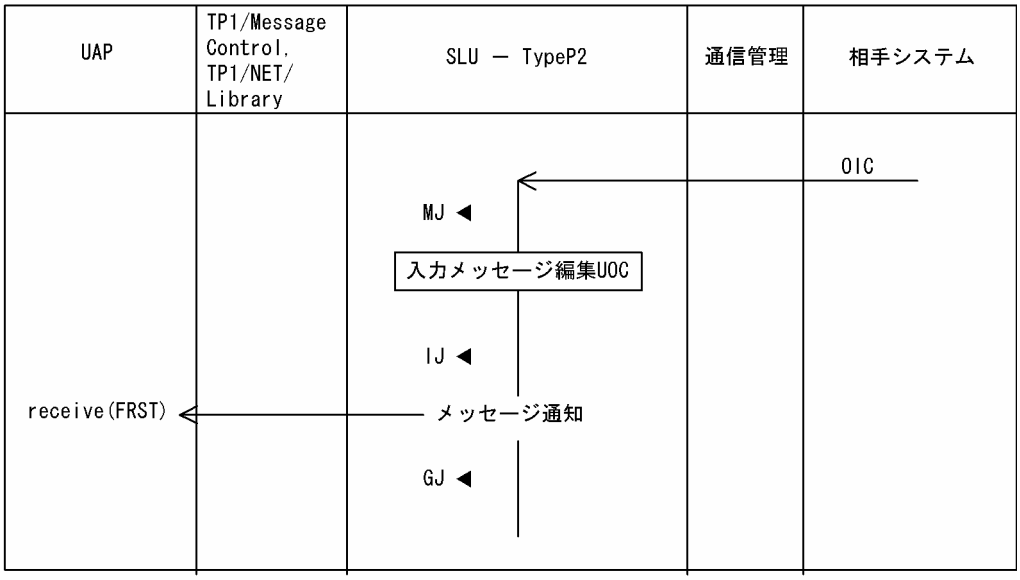
（凡例）
AJ ◀：メッセージ送信完了ジャーナル取得
MJ ◀：メッセージジャーナル取得
OJ ◀：メッセージ出力ジャーナル取得

図 D-2　一方送信メッセージ（send 型／複数セグメントの場合）



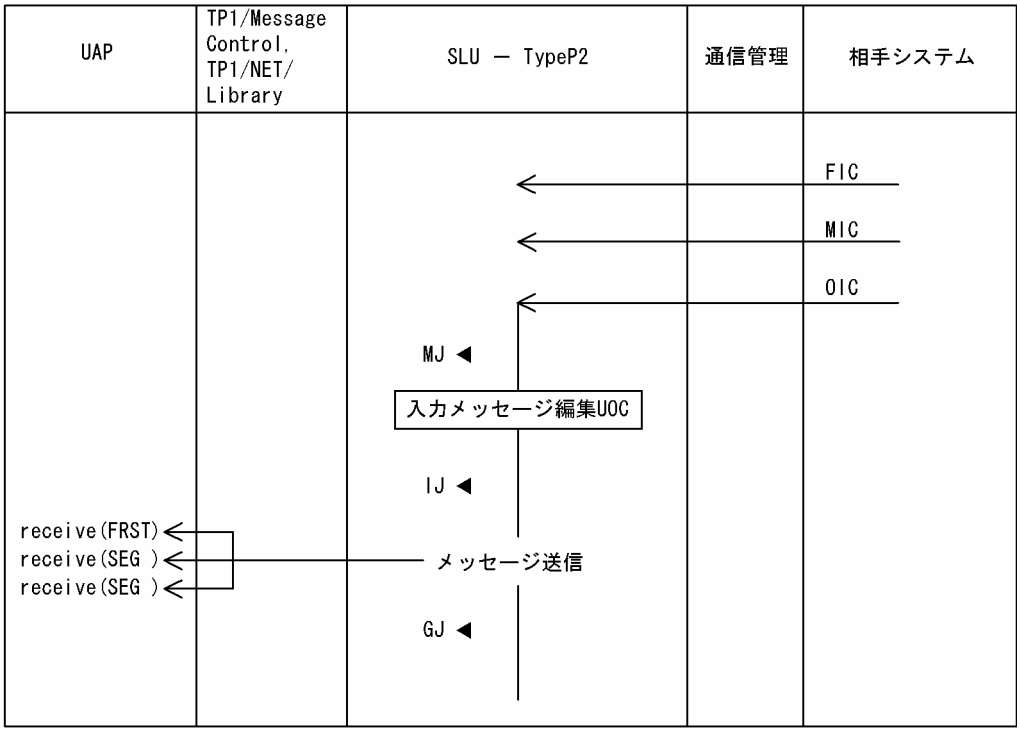
（凡例）
AJ ◀：メッセージ送信完了ジャーナル取得
MJ ◀：メッセージジャーナル取得
OJ ◀：メッセージ出力ジャーナル取得

図 D-3　一方送信メッセージ（receive 型／単独セグメントの場合）



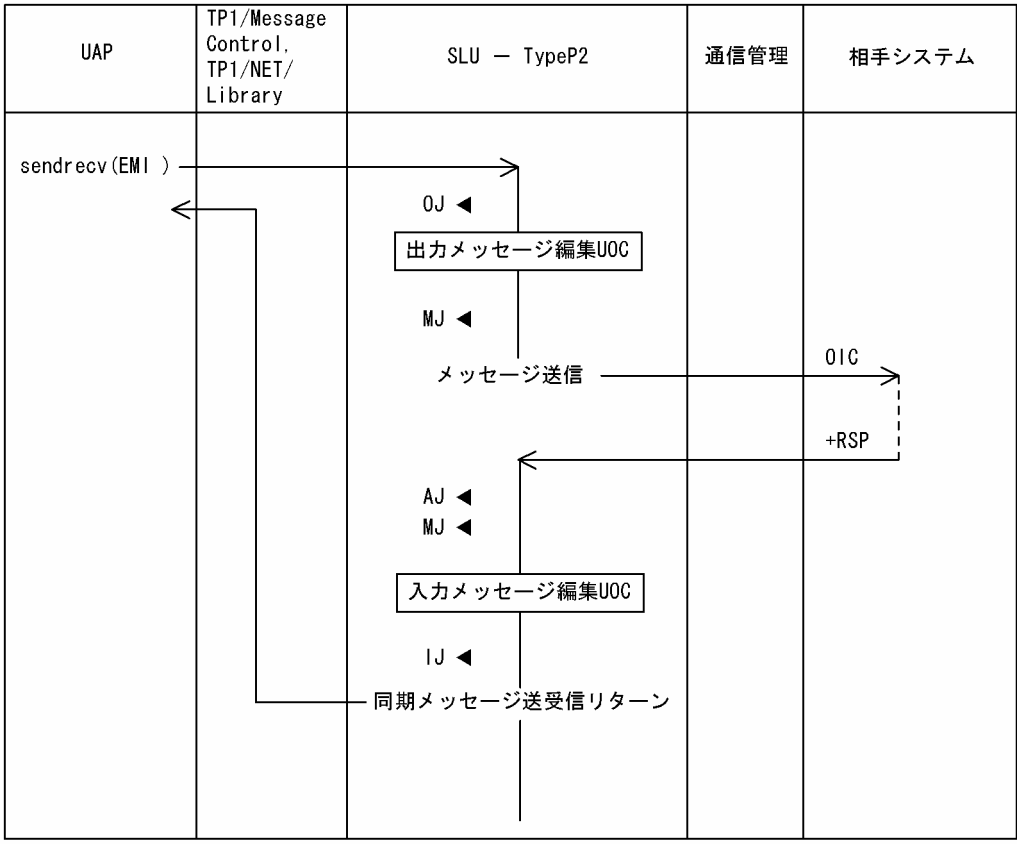
（凡例）
GJ ◀：メッセージ受信ジャーナル取得
IJ ◀：メッセージ入力ジャーナル取得
MJ ◀：メッセージジャーナル取得

図 D-4　一方送信メッセージ（receive 型／複数セグメントの場合）



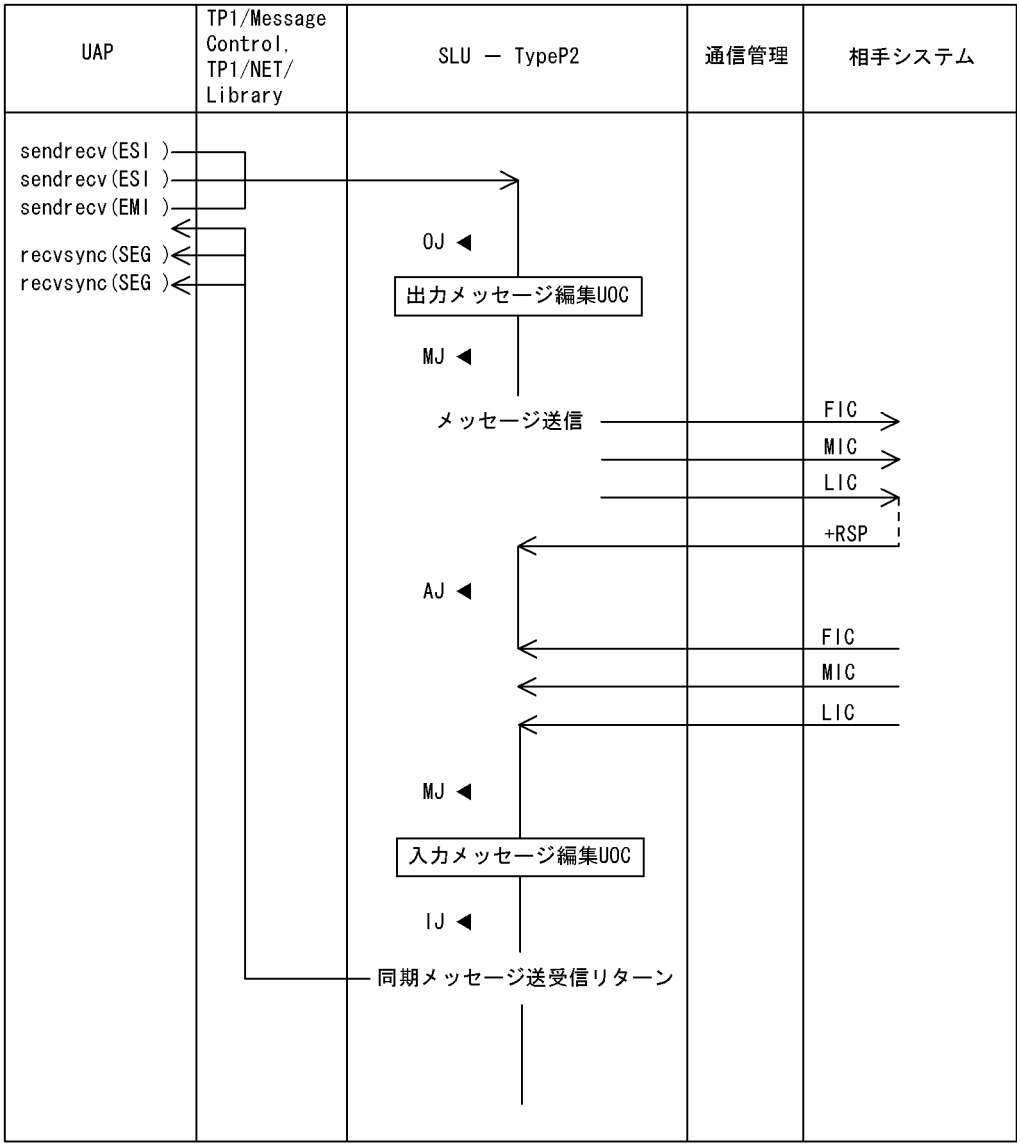
（凡例）
GJ ◀：メッセージ受信ジャーナル取得
IJ ◀：メッセージ入力ジャーナル取得
MJ ◀：メッセージジャーナル取得

図 D-5　問い合わせメッセージ（request 型／単独セグメントの場合）



（凡例）
AJ◀：メッセージ送信完了ジャーナル取得
IJ◀：メッセージ入力ジャーナル取得
MJ◀：メッセージジャーナル取得
OJ◀：メッセージ出力ジャーナル取得

図 D-6　問い合わせメッセージ（request 型／複数セグメントの場合）



（凡例）
AJ ◀：メッセージ送信完了ジャーナル取得
IJ ◀：メッセージ入力ジャーナル取得
MJ ◀：メッセージジャーナル取得
OJ ◀：メッセージ出力ジャーナル取得

付録 E　障害発生時の処理の流れ

障害発生時の処理の流れを図 E-1～図 E-5 に示します。

図 E-1　コネクション確立失敗

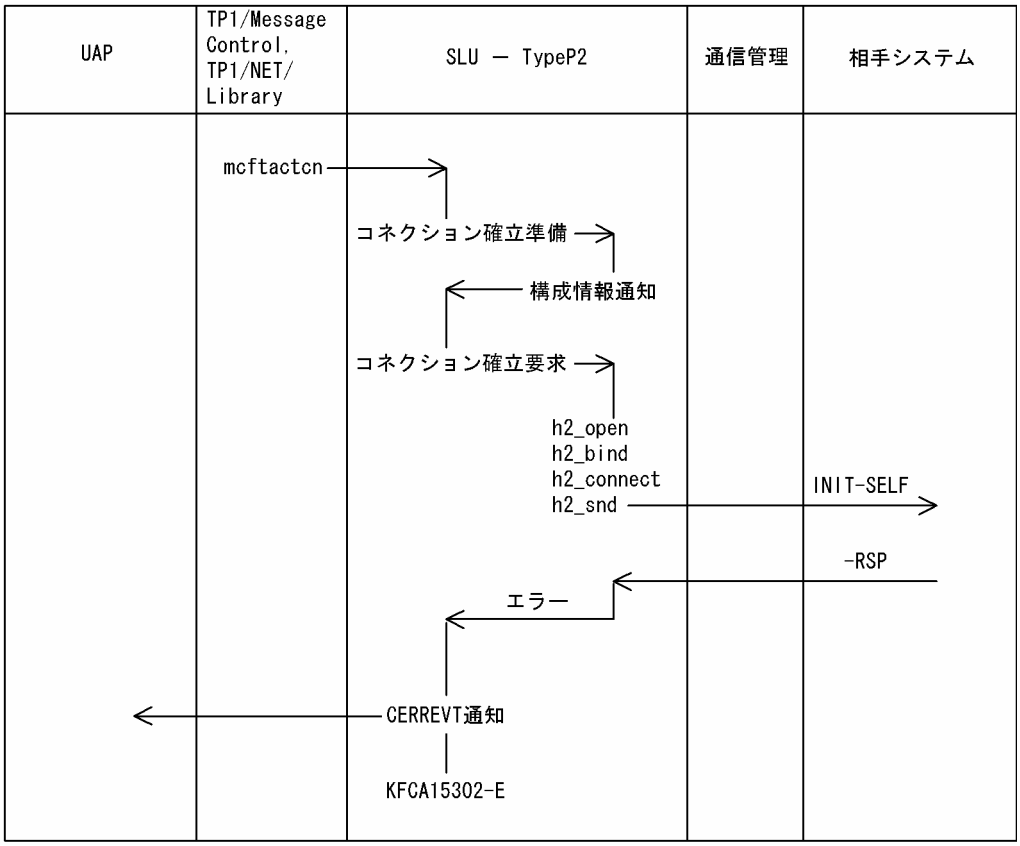


図 E-2　コネクション強制解放

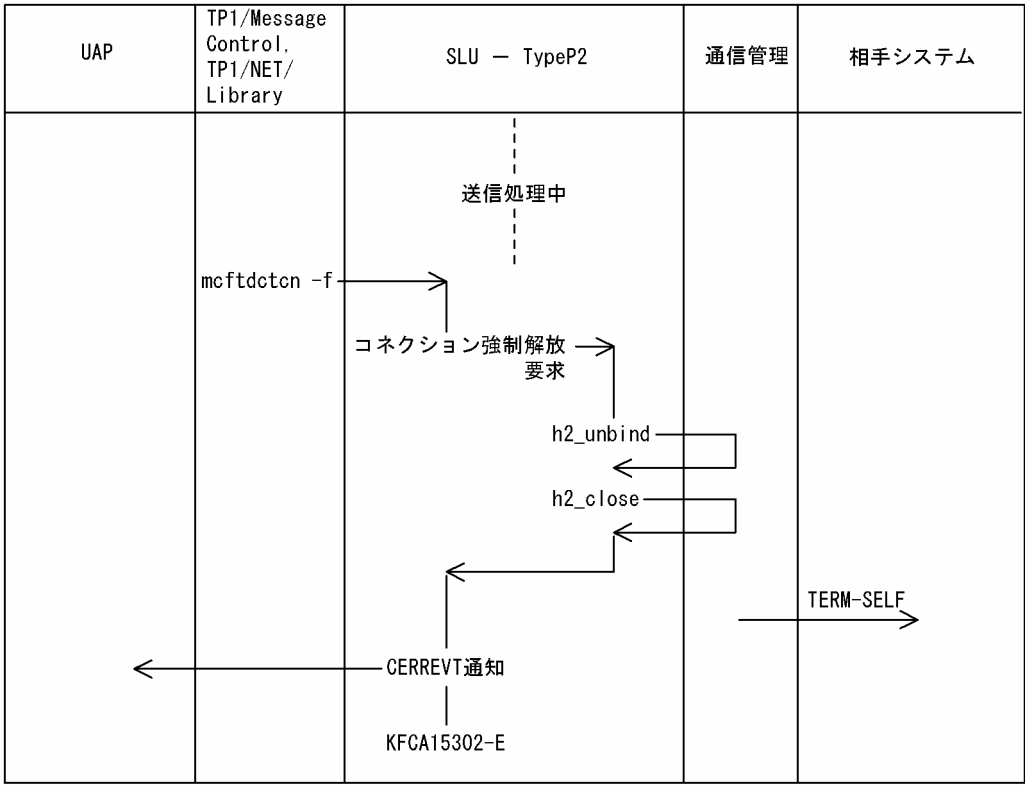
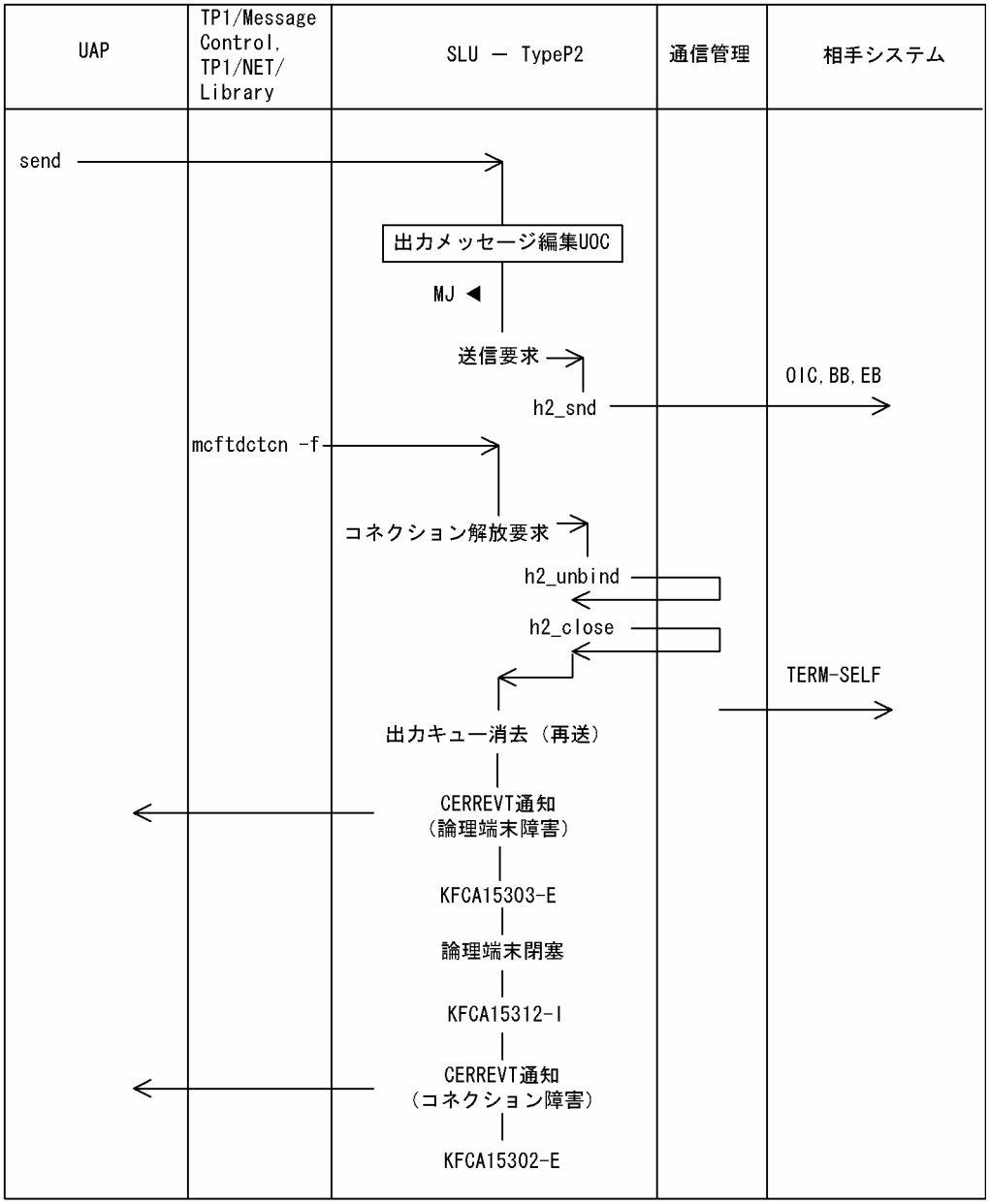
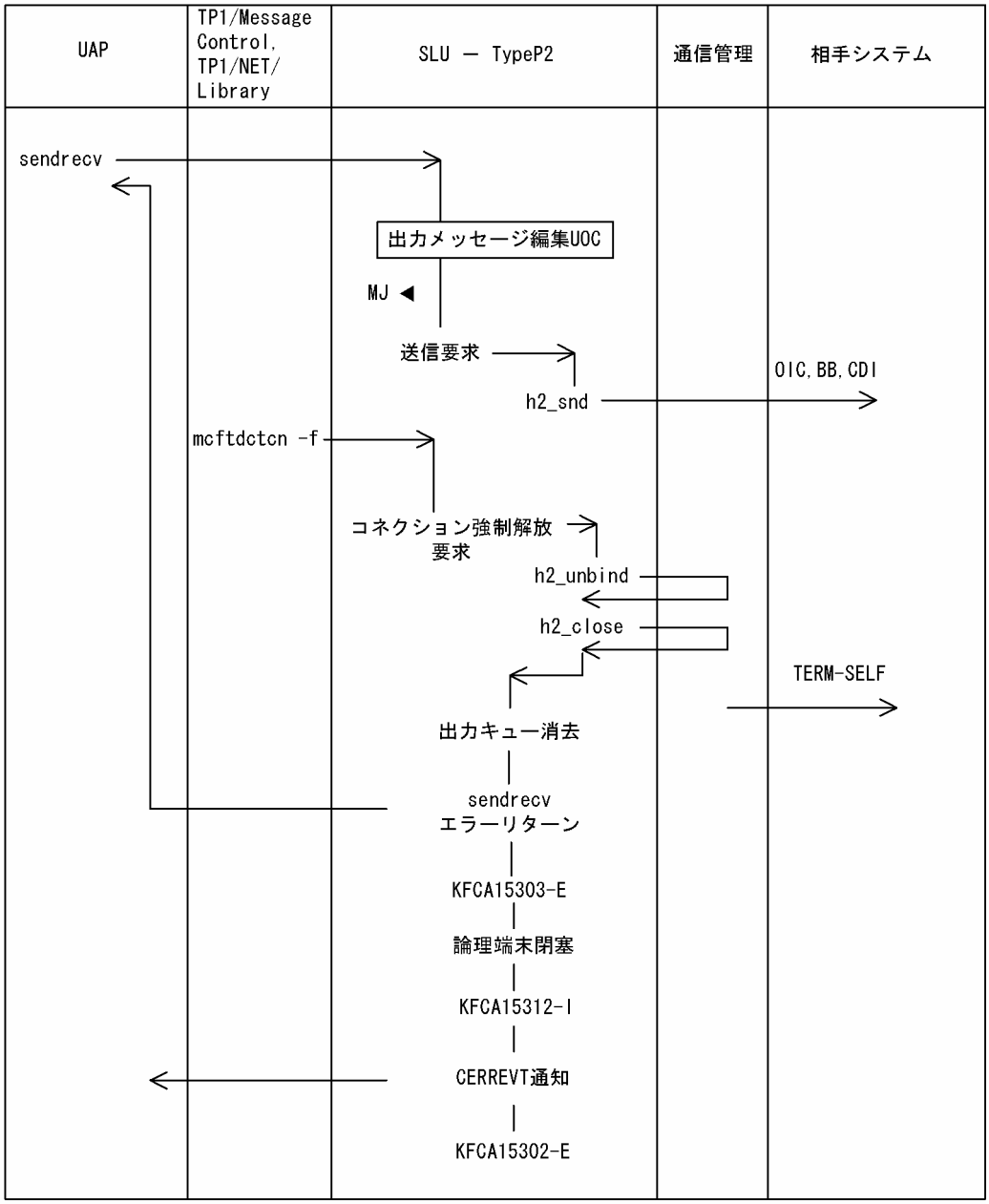


図 E-3　コネクション強制解放（send 実行中）



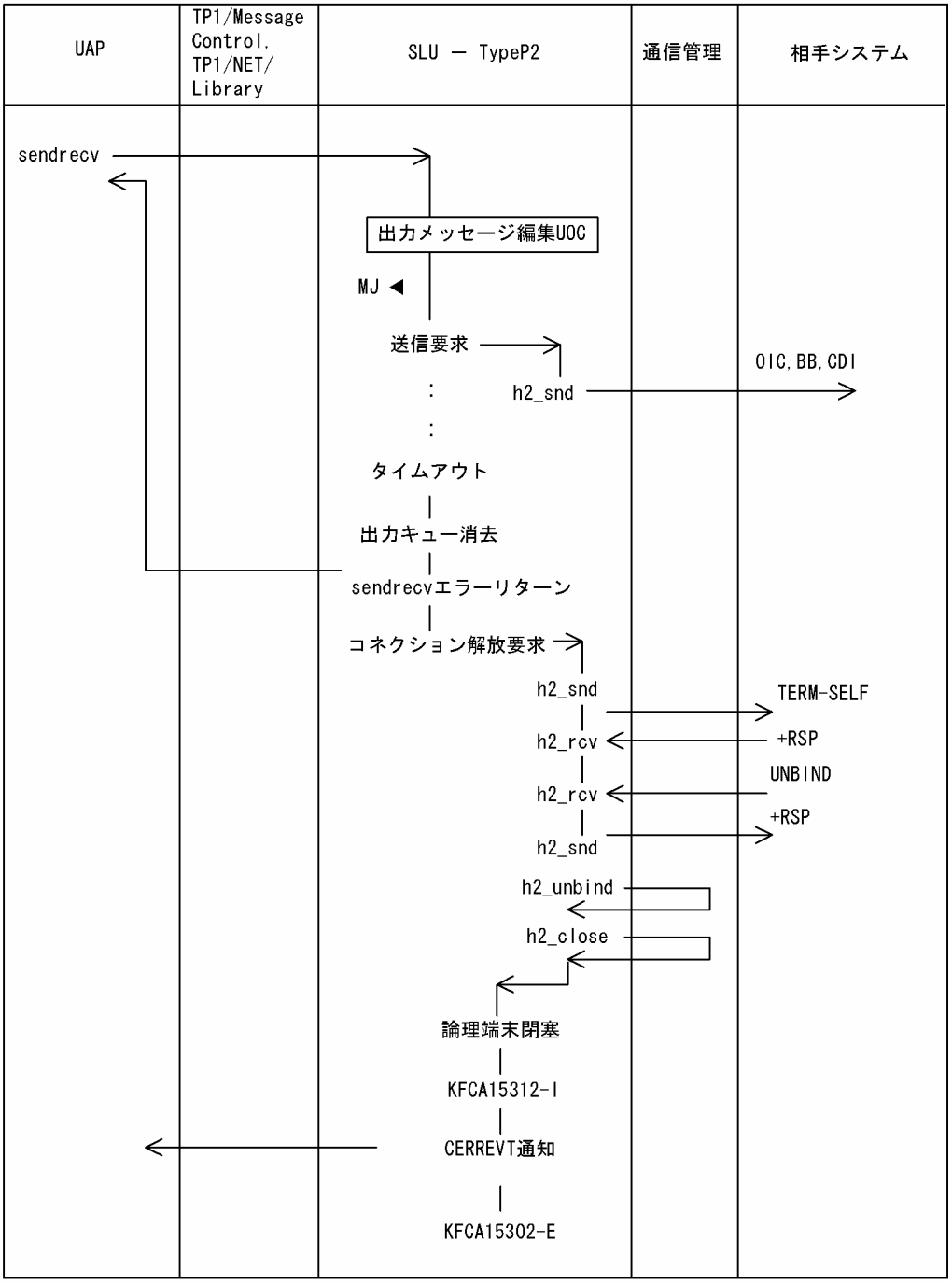
（凡例）
MJ◀：メッセージジャーナル取得

図 E-4　コネクション強制解放（sendrecv 実行中）



（凡例）
MJ ◀：メッセージジャーナル取得

図 E-5　タイムアウト



（凡例）
MJ◀：メッセージジャーナル取得

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)　UOC 呼び出し時

SLU - TypeP2 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-1　UOC 呼び出し時の MCF 固有情報のダンプ出力情報

イベント ID	オフセット				
	0x0000～0x0003	0x0004～0x0007	0x0008～0x000f	0x0010～0x0017	0x0018～0x001f
0xa070	MCF 識別子	スレッド ID	－	－	UOC 名称
0xa071			－	－	

（凡例）

－：情報を取得しません。

表 F-2　UOC 名称の出力情報

UOC の種類	UOC 名称出力情報
入力メッセージ編集 UOC	"MSGRCV"
出力メッセージ編集 UOC	"MSGSEND"
送信メッセージ通番編集 UOC	"SEND_UOC"

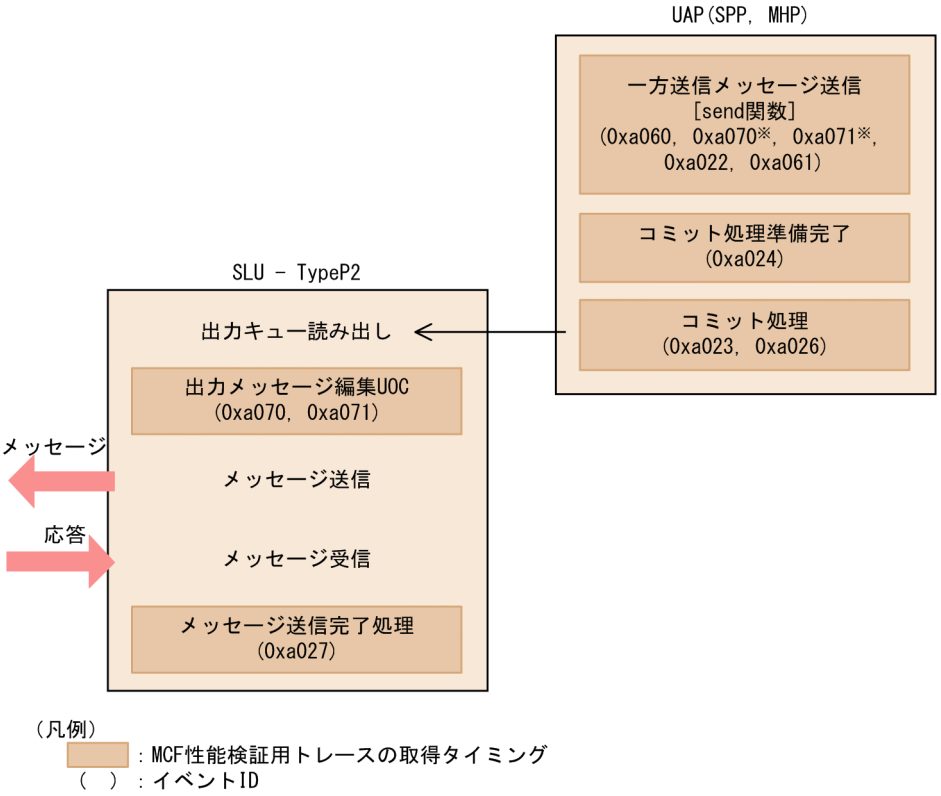
付録 F.2　MCF 性能検証用トレースの取得タイミング

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



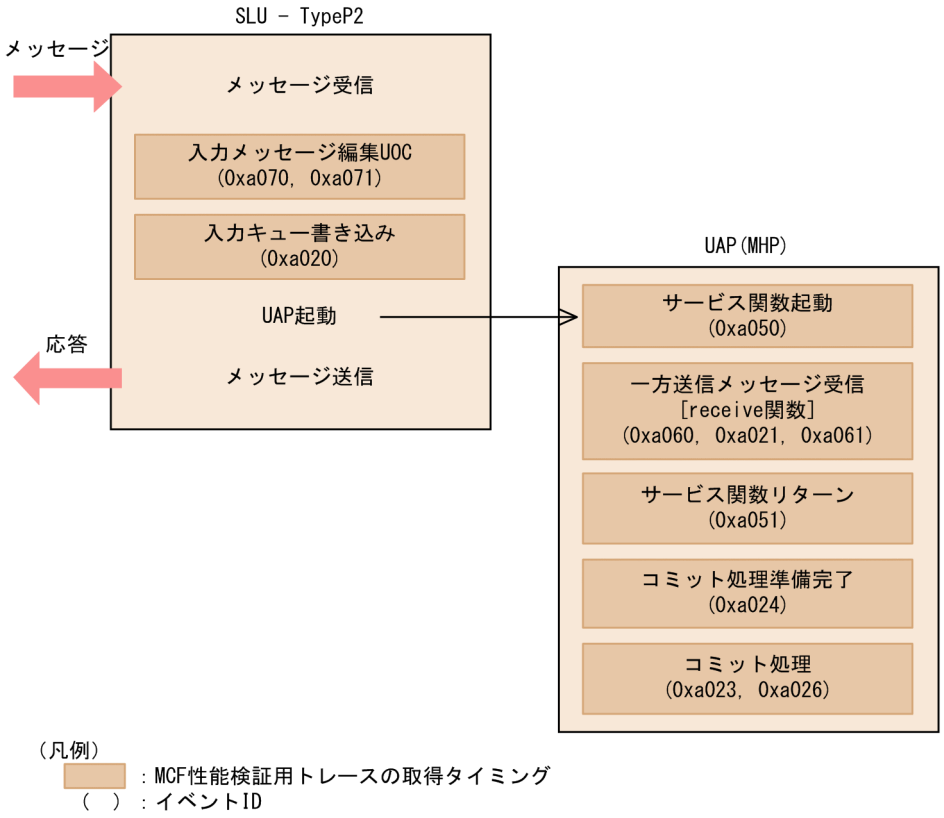
注※

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。

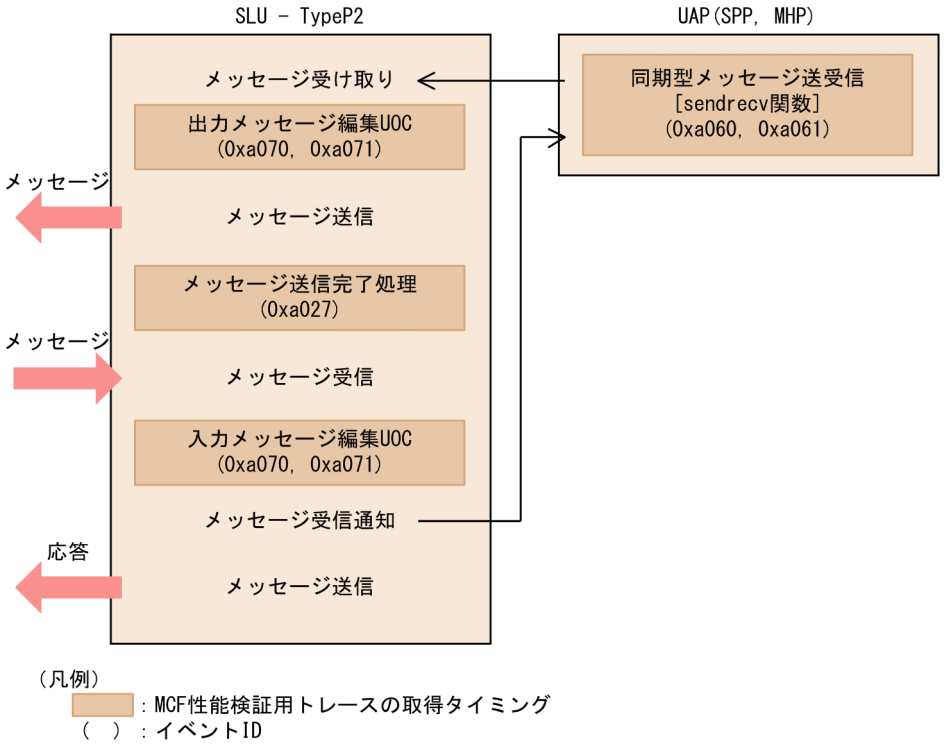
図 F-2　一方送信メッセージ受信時の MCF 性能検証用トレースの取得タイミング



(3)　同期型メッセージ送受信時

同期型メッセージ送受信時の MCF 性能検証用トレースの取得タイミングについて，次の図に示します。

図 F-3　同期型メッセージ送受信時の MCF 性能検証用トレースの取得タイミング



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。

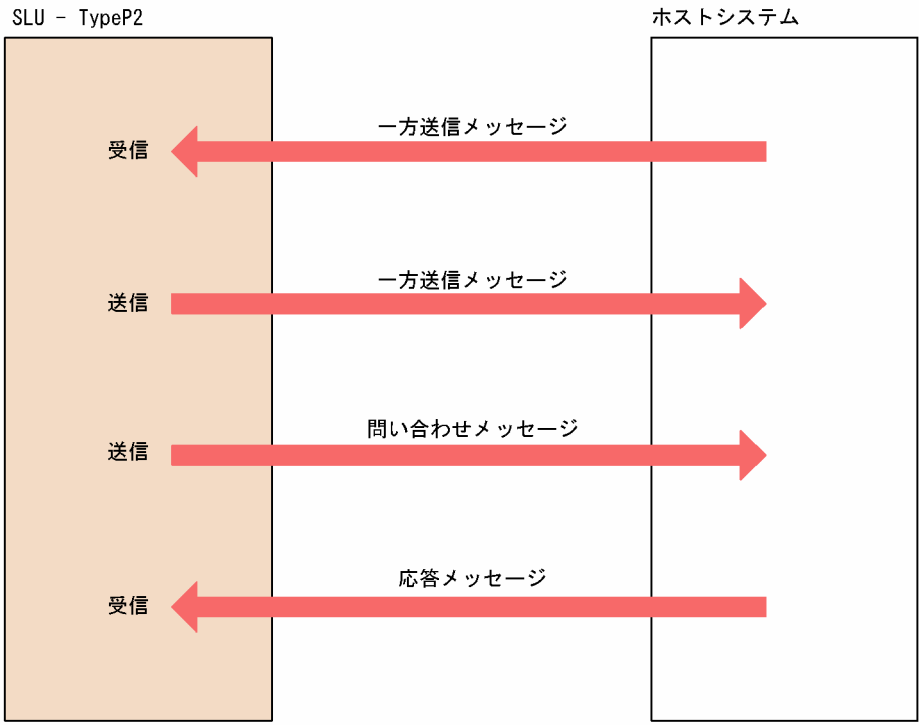
表 F-3　MCF 性能検証用トレースの取得量

メッセージの送受信形態	トレース取得量 (単位：キロバイト)
一方送信メッセージの送信	1.6
一方送信メッセージの受信	2.0
同期型のメッセージの送受信	0.9

付録 G　ユーザアプリケーションプログラムの作成例

UAP の作成例として，処理の流れを次の図に示し，その流れに沿ったコーディング例を C 言語（K&R 版）と COBOL 言語で示します。

図 G-1　処理の流れの例



付録 G.1　コーディング例

ここでは，ユーザアプリケーションプログラムのコーディング例を示します。

(1)　C 言語

```
#include<dcmcf.h>

#define rcvmax 5120
void slmhprcv1()
{
   char            termnam[16];
   char            mapname[16];
   char            sdatarea[2048];
   char            rdatarea[rcvmax];
   DCLONG          sdataleng;
   DCLONG          rdataleng;
   DCLONG          time;

   int             rtn;
```

```
   memset(mapname, 0, 16);
   memset(sdatarea, 0, 2048);
   memset(rdatarea, 0, rcvmax);

   rtn =dc_mcf_receive(   DCMCFFRST,
                          DCNOFLAGS,
                          termnam,
                          mapname,
                          rdatarea,
                          &rdataleng,
                          rcvmax,
                          &time   );

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

   if(rtn !=DCMCFRTN_00000)
   {
      goto ERROR;
   }
ERROR:;
   return;
}


#include<dcmcf.h>

#define sndmax   448
#define rcvmax   5120
#define segmax   (sndmax -9 -3 -4)          /*432 bytes */

/*448 bytes */
typedef   struct senddata_s{
   char    mcf_use[8];
   char    ap_name[9];
   char    snd_dmy[3];
   char    snd_lng[4];
   char    snd_seg[segmax];
}sdata_def;

void   slsppsnd1()
{
   static char   *lename =”SL2REQ01”;
   char            termnam[9];
   char            resv01[9];
   char            resv02[9];
   char            resv03[9];
   DCLONG          rdataleng;
   DCLONG          sdataleng;
   DCLONG          inbufleng;
   sdata_def       senddata;
   int     rtn_code;

   memset(resv01, 0, 9);
   memset(resv02, 0, 9);
```

```
   memset(resv03, 0, 9);

   memset(&senddata, 0x33, sizeof(senddata));

   memcpy(senddata.ap_name, "slup2snd ", 9);
   memcpy(senddata.snd_lng, "0448", 4);
   memcpy(senddata.snd_seg, "send:aaaaaaaaaa1aaaaaaaaaaaaaaa2
aaaaaaaaaaaaaaa3aaaaaaaaaaaaaaa4aaaaaaaaaaaaaaa5
aaaaaaaaaaaaaaa6aaaaaaaaaaaaaaa7aaaaaaaaaaaaaaa8
aaaaaaaaaaaaaaa9aaaaaaaaaaaaaaaAaaaaaaaaaaaaaaaB
aaaaaaaaaaaaaaaCaaaaaaaaaaaaaaaDaaaaaaaaaaaaaaaE
aaaaaaaaaaaaaaaFaaaaaaaaaaaaaaa0", 256);
   memcpy(senddata.snd_seg + segmax -3, "end", 3);

   rtn_code =dc_mcf_send(   DCMCFEMI,
                            DCMCFOUT,
                            lename,
                            resv01,
                            &senddata,
                            sndmax,
                            resv02,
                            DCNOFLAGS   );
   if(rtn_code !=DCMCFRTN_00000)
   {
      goto ERROR;
   }
ERROR:;
   return;
}


#include <dcmcf.h>

#define sndmax   448
#define rcvmax   5120
#define segmax   (sndmax -9 -3 -4)/*432 bytes */

/*448 bytes */
typedef   struct senddata_s{
   char    mcf_use[8];
   char    ap_name[9];
   char    snd_dmy[3];
   char    snd_lng[4];
   char    snd_seg[segmax];
}sdata_def;

void   slsppreq1()
{
   static char   rcv_buf[rcvmax];
   static char   *lename ="SL2REQ01";
   char            termnam[9];
   char            resv01[9];
   char            resv02[9];
   char            resv03[9];
   DCLONG          rdataleng;
   DCLONG          sdataleng;
   DCLONG          inbufleng;
   sdata_def       senddata;
```

```
   DCLONG          rcvtime;
   int             rtn_code;

   memset(resv01, 0, 9);
   memset(resv02, 0, 9);
   memset(resv03, 0, 9);

   memset(&senddata, 0x33, sizeof(senddata));

   memcpy(senddata.ap_name, "slup2req ", 9);
   memcpy(senddata.snd_lng, "0448", 4);
   memcpy(senddata.snd_seg, "*FMH*aaaaaaaaaa1aaaaaaaaaaaaaaa2
aaaaaaaaaaaaaaa3aaaaaaaaaaaaaaa4aaaaaaaaaaaaaaa5
aaaaaaaaaaaaaaa6aaaaaaaaaaaaaaa7aaaaaaaaaaaaaaa8
aaaaaaaaaaaaaaa9aaaaaaaaaaaaaaaAaaaaaaaaaaaaaaaB
aaaaaaaaaaaaaaaCaaaaaaaaaaaaaaaDaaaaaaaaaaaaaaaE
aaaaaaaaaaaaaaaFaaaaaaaaaaaaaaa0", 256);
   memcpy(senddata.snd_seg + segmax -3, "end", 3);

   rtn_code =dc_mcf_sendrecv (DCMCFEMI,
                              DCMCFIO,
                              lename,
                              resv01,
                              &senddata,
                              sndmax,
                              rcv_buf,
                              &rdataleng,
                              rcvmax,
                              &rcvtime,
                              0);

   if(rtn_code != DCMCFRTN_00000)
   {
      goto ERROR;
   }
ERROR:;
   return;
}
```

(2) COBOL 言語

```
 IDENTIFICATION    DIVISION.
 PROGRAM-ID.       UAPCBL.

 ENVIRONMENT       DIVISION.

 DATA              DIVISION.
 WORKING-STORAGE   SECTION.
 01   RCV.
      02  MSG-REQ          PIC X(8)   VALUE   'RECEIVE '.
      02  RTN              PIC X(5).
      02  FILLER           PIC X(3).
      02  RSV1             PIC X(4)   VALUE   'FRST'.
      02  RSV2             PIC X(4)   VALUE   SPACE.
      02  MCFUSING1        PIC 9(8).
      02  MCFUSING2        PIC 9(8).
```

```
      02  RSV3             PIC 9(9)   COMP    VALUE   2048.
      02  SEGKIND          PIC X(4)   VALUE   SPACE.
      02  RSV4             PIC X(4)   VALUE   SPACE.
      02  MSGKIND          PIC X(4)   VALUE   SPACE.
      02  OUTPUTNO         PIC X(4)   VALUE   SPACE.
      02  RSV5             PIC X(8)   VALUE   SPACE.
      02  RSV6             PIC X(4)   VALUE   SPACE.
      02  RSV7             PIC X(8)   VALUE   SPACE.
      02  RSV8             PIC X(4)   VALUE   SPACE.
      02  RSV9             PIC 9(9)   COMP    VALUE   ZERO.
      02  RSV10            PIC 9(9)   COMP    VALUE   ZERO.
      02  RSV11            PIC X(1)   VALUE   SPACE.
      02  RSV12            PIC X(1)   VALUE   SPACE.
      02  RSV13            PIC X(14)  VALUE   LOW-VALUE.
 01   CD1.
      02  SEG-CODE1        PIC X(4)   VALUE   SPACE.
      02  TERM-CODE        PIC X(8)   VALUE   SPACE.
      02  MCFUSE14         PIC X(8)   VALUE   SPACE.
      02  MCFUSE15         PIC X(8)   VALUE   SPACE.
      02  MCFUSE16         PIC X(28)  VALUE   LOW-VALUE.
 01   DATA1.
      02  MSGSEG-LENG1     PIC 9(9)   COMP.
      02  MCFUSE17         PIC X(4).
      02  MCFUSE18         PIC X(2).
      02  MCFUSE19         PIC X(1).
      02  MCFUSE20         PIC X(1).
      02  REC-MSGSEG1      PIC X(2048).

 PROCEDURE DIVISION.
*************************************************************
 CALL 'CBLDCMCF'  USING   RCV    CD1    DATA1.
*
*************************************************************
*                                                           *
*  User Service Part                                        *
*                                                           *
*************************************************************
*
 EXIT PROGRAM.

 IDENTIFICATION    DIVISION.
 PROGRAM-ID.       SPPSND.

 ENVIRONMENT       DIVISION.
 CONFIGURATION     SECTION.

 DATA              DIVISION.
 WORKING-STORAGE   SECTION.

 01   SND-AREA1.
      02  DATA-A           PIC X(8)   VALUE   'SEND    '.
      02  DATA-B           PIC X(5).
      02  FILLER           PIC X(3).
      02  DATA-C           PIC X(4)   VALUE   SPACE.
      02  DATA-D           PIC X(4)   VALUE   SPACE.
      02  DATA-E           PIC 9(8).
      02  DATA-F           PIC 9(8).
      02  DATA-G           PIC 9(9)   COMP    VALUE   ZERO.
```

```
     02  DATA-H          PIC X(4)   VALUE   'EMI '.
     02  DATA-I          PIC X(4)   VALUE   SPACE.
     02  DATA-J          PIC X(4)   VALUE   'NORM'.
     02  DATA-K          PIC X(4)   VALUE   'NSEQ'.
     02  DATA-L          PIC X(8)   VALUE   SPACE.
     02  DATA-M1         PIC X(4)   VALUE   SPACE.
     02  DATA-M2         PIC X(8)   VALUE   SPACE.
     02  DATA-M3         PIC X(4)   VALUE   SPACE.
     02  DATA-M4         PIC 9(9)   COMP    VALUE   ZERO.
     02  DATA-M5         PIC 9(9)   COMP    VALUE   ZERO.
     02  DATA-M6         PIC X(1)   VALUE   SPACE.
     02  DATA-M7         PIC X(1)   VALUE   '1'.
     02  DATA-N          PIC X(14)  VALUE   LOW-VALUE.
01   SND-AREA2.
     02  DATA-O          PIC X(4)   VALUE   'OUT '.
     02  DATA-P          PIC X(8)   VALUE   'SL2REQ01'.
     02  DATA-Q          PIC X(8)   VALUE   SPACE.
     02  DATA-R          PIC X(8)   VALUE   SPACE.
     02  DATA-T          PIC X(28)  VALUE   LOW-VALUE.
01   SND-AREA3.
     02  DATA-U          PIC 9(9)   COMP   VALUE   248.
     02  DATA-V          PIC X(8).
     02  DATA-W          PIC X(248).

PROCEDURE         DIVISION.
CALL 'CBLDCMCF'  USING   SND-AREA1    SND-AREA2
     SND-AREA3.
EXIT PROGRAM.

IDENTIFICATION    DIVISION.
PROGRAM-ID.       SPPSR.

ENVIRONMENT       DIVISION.
CONFIGURATION     SECTION.

DATA              DIVISION.
WORKING-STORAGE   SECTION.

01   SR-AREA1.
     02  DATA-A          PIC X(8)   VALUE   'SENDRECV'.
     02  DATA-B          PIC X(5).
     02  FILLER          PIC X(3).
     02  DATA-C          PIC X(4)   VALUE   SPACE.
     02  DATA-D          PIC X(4)   VALUE   SPACE.
     02  DATA-E          PIC 9(8).
     02  DATA-F          PIC 9(8).
     02  DATA-G          PIC 9(9)   COMP    VALUE   248.
     02  DATA-H          PIC X(4)   VALUE   'EMI '.
     02  DATA-I          PIC X(4)   VALUE   SPACE.
     02  DATA-J          PIC X(4)   VALUE   SPACE.
     02  DATA-K          PIC X(4)   VALUE   SPACE.
     02  DATA-L          PIC X(8)   VALUE   SPACE.
     02  DATA-M1         PIC X(4)   VALUE   SPACE.
     02  DATA-M2         PIC X(8)   VALUE   SPACE.
     02  DATA-M3         PIC X(4)   VALUE   SPACE.
     02  DATA-M4         PIC 9(9)   COMP    VALUE   ZERO.
     02  DATA-M5         PIC 9(9)   COMP    VALUE   ZERO.
     02  DATA-M6         PIC X(1)   VALUE   SPACE.
```

```
      02  DATA-M7          PIC X(1)   VALUE   SPACE.
      02  DATA-N           PIC X(14)  VALUE   LOW-VALUE.
 01   SR-AREA2.
      02  DATA-O           PIC X(4)   VALUE   'IO  '.
      02  DATA-P           PIC X(8)   VALUE   'SL2REQ01'.
      02  DATA-Q           PIC X(8)   VALUE   SPACE.
      02  DATA-R           PIC X(8)   VALUE   SPACE.
      02  DATA-T           PIC X(28)  VALUE   LOW-VALUE.
 01   SR-AREA3.
      02  DATA-U           PIC 9(9)   COMP    VALUE   248.
      02  DATA-V           PIC X(8).
      02  DATA-W           PIC X(248).
 01   SR-AREA4.
      02  DATA-X           PIC 9(9)   COMP.
      02  DATA-Y1          PIC X(7)   VALUE   SPACE.
      02  DATA-Y2          PIC X(1).
      02  DATA-Z           PIC X(248).

 PROCEDURE         DIVISION.
 CALL 'CBLDCMCF'  USING   SR-AREA1  SR-AREA2
      SR-AREA3  SR-AREA4.

 EXIT PROGRAM.
```

(3) データ操作言語

```
 IDENTIFICATION    DIVISION.
 PROGRAM-ID.       UAPDML.

 ENVIRONMENT       DIVISION.
 CONFIGURATION     SECTION.
*
*************************************************************
*
 DATA              DIVISION.
 WORKING-STORAGE   SECTION.
*
*************************************************************
*
 01   RECV-AREA1.
      02  RE-DATALENG1     PIC 9(4)   COMP   VALUE   1028.
      02  RE-RSV1          PIC X(2).
      02  RE-DATA1         PIC X(1024).
*
*************************************************************
*
 COMMUNICATION SECTION.
*
*************************************************************
*


     CD  RECV-IN1
     FOR  INPUT
     STATUS KEY IS                  RE-STATUS1
     SYMBOLIC TERMINAL IS           RE-TERMNAM
```

```
     MESSAGE DATE IS                 RE-DATE1
     MESSAGE TIME IS                 RE-TIME1.
*
************************************************************
*
 PROCEDURE DIVISION.
*
************************************************************
*
     RECEIVE  RECV-IN1
         FIRST  SEGMENT
         INTO  RECV-AREA1.
*
 EXIT PROGRAM.


 IDENTIFICATION    DIVISION.
 PROGRAM-ID.       SNDDML.

 ENVIRONMENT       DIVISION.
 CONFIGURATION     SECTION.
*
************************************************************
*
 DATA              DIVISION.
 WORKING-STORAGE   SECTION.
*
************************************************************
*
 01   SEND-AREA1.
      02  SE-DATALENG1   PIC 9(4)   COMP   VALUE   1028.
      02  SE-RSV1        PIC X(2).
      02  SE-DATA1       PIC X(1024).
*
************************************************************
*
 COMMUNICATION SECTION.
*
************************************************************
*

    CD  SEND-IN1
        FOR  OUTPUT
        STATUS KEY IS                SE-STATUS1
        SYMBOLIC TERMINAL IS         SE-TERMNAM
        SYNCHRONOUS MODE IS          ASYNC
        SWITCHING MODE IS            NORMAL.
*
************************************************************
*
 PROCEDURE DIVISION.
*
************************************************************
*
    MOVE 'SL2REQ01'TO SE-TERMNAM.
    SEND SEND-IN1
        FROM SEND-AREA1
                WITH EMI.
```

```
*
 EXIT PROGRAM.
```

付録 G.2　提供するサンプルコーディング

ここでは，SLU - TypeP2 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

/BeTRAN/examples/mcf/SLUP2/aplib/c/ap1.c

/BeTRAN/examples/mcf/SLUP2/aplib/c/sv1.c

/BeTRAN/examples/mcf/SLUP2/aplib/c/sv2.c

(2)　COBOL 言語

/BeTRAN/examples/mcf/SLUP2/aplib/cobol/ap1.cbl

/BeTRAN/examples/mcf/SLUP2/aplib/cobol/sv1.cbl

/BeTRAN/examples/mcf/SLUP2/aplib/cobol/sv2.cbl

(3)　データ操作言語

/BeTRAN/examples/mcf/SLUP2/aplib/dml/ap1.cbl

/BeTRAN/examples/mcf/SLUP2/aplib/dml/sv1.cbl

付録 H　理由コードおよびセンスコード一覧

付録 H.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 H-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 H.2　CERREVT の理由コード

CERREVT 通知時の理由コードを次の表に示します。

表 H-2　CERREVT の理由コード

理由コード 1 (10 進数字)	理由コード 2 (10 進数字)	発生条件	障害レベル
DCMSLM_RSN1_MCF (MCF 障害) (1)	DCMSLM_RSN2_ITQ (0)	メッセージ入力障害	コネクション
	DCMSLM_RSN2_APL (1)	アプリケーション名取得障害	コネクション
	DCMSLM_RSN2_OTGET (2)	メッセージ出力障害	コネクション
	DCMSLM_RSN2_SLCMP (3)	メッセージ送信処理障害	コネクションまたは論理端末 ※
	DCMSLM_RSN2_UAPAB (5)	UAP 異常による強制解放	コネクション
	DCMSLM_RSN2_SYCER (6)	UAP への同期リターン失敗	コネクション
	DCMSLM_RSN2_ENDER (7)	終了処理中のメッセージ拒否	コネクション
	DCMSLM_RSN2_DCTLE (8)	運用コマンドまたは API による論理端末閉塞	論理端末
	DCMSLM_RSN2_LETYPE (9)	論理端末の型不正	論理端末
	DCMSLM_RSN2_ERRIND (11)	エラーデータ受信	論理端末
	DCMSLM_RSN2_DCTCN_F (536870912)	運用コマンドまたは API による強制解放	コネクション
	DCMSLM_RSN2_ENDTO (536870913)	強制解放（応答メッセージ受信監視時間超過）	コネクション
	DCMSLM_RSN2_NOBUF (536870914)	バッファ取得失敗による強制解放	コネクション
	DCMSLM_RSN2_ERROR (-1)	MCF 内部障害による強制解放	コネクション
	その他	上記以外の障害（理由コード 2 は保守情報）	コネクション

理由コード 1 (10 進数字)	理由コード 2 (10 進数字)	発生条件	障害レベル
DCMSLM_RSN1_ABORT (プロトコル障害) (6)	DCMSLM_RSN2_XNF (16)	下位層障害	コネクション
	DCMSLM_RSN2_SLM (17)	メッセージ制御プロトコル障害	コネクション
DCMSLM_RSN1_UOC (UOC 検出障害) (3)	UOC からの詳細リターンコード	ユーザ（UOC）検出障害	コネクション
DCMSLM_RSN1_ACTER (コネクション障害) (5)	不定	コネクションの確立失敗	コネクション
その他	不定	上記以外の障害（理由コード 1，2 は保守情報）	不定

注※

コネクションに発生するのは，送信完了エラーおよび-RSP 受信時の場合です。そのほかの発生条件の場合は，論理端末に発生します。

付録 H.3　SLU - TypeP2 が使用するセンスコード

SLU - TypeP2 が使用するセンスコード（KFCA15204-E（-RSP 送信時ログ）出力時のセンスコード）を次の表に示します。

表 H-3　SLU - TypeP2 が使用するセンスコード

センスコード	意味	発生条件
0x08000000	要求不実行	入力キュー書き込み障害の発生
		メッセージ受信に対するアプリケーションスケジュール失敗
		入力メッセージ編集 UOC エラーリターン
0x08090000	モード不一致	RQD 指定のメッセージ受信後，レスポンス送信前にデータ受信
		レスポンス受信待ちで応答（EB）メッセージ受信
		チェイン受信途中ではないのに CANCEL 受信
0x08130000	ブラケット開始拒否（RTR 送信なし）	RTR のレスポンス受信前に一方（BB，EB）メッセージ受信
		ホストに送信権委譲後に BID 受信

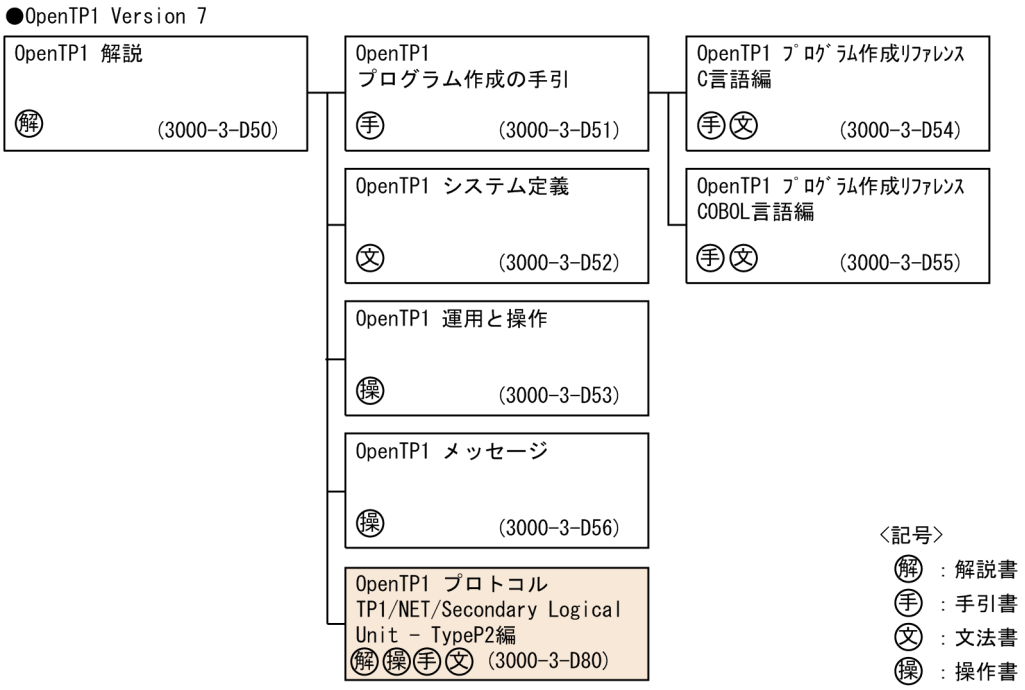
センスコード	意味	発生条件
0x08140000	ブラケット開始拒否（RTR 送信あり）	メッセージ送信に対するレスポンス待ちのときに，一方（BB，EB）メッセージまたは BID 受信
0x08210000	セションパラメタ不正	BIND 受信時にセションパラメタ不正検出
0x10020000	RU 長不正	受信したメッセージのレングスが，セションパラメタで指定された最大 RU 長を超過
0x10030000	機能未サポート	次に示す未サポートコマンドを受信 • LUSTAT • CHASE • QEC • RELQ • SIGNAL • SBI • BIS
0x20010000	シーケンス不正	NSPE を BIND 受信待ち以外で受信
		BIND を BIND 受信待ち以外で受信
		SHUTD の二重受信
0x20020000	チェイン状態違反	FIC の二重受信
		FIC，OIC と受信
		FIC 受信に対して，-RSP（2002 または 2003）送信後，MIC または LIC 受信
0x20030000	ブラケット違反	一方（BB，EB）MIC，LIC 受信待ちのときに応答（EB）メッセージ受信
		応答（EB）メッセージ受信待ちのときに一方（BB，EB）メッセージ受信
		ホストから BB 有り EB 無しのメッセージ受信
		ホストから BB，EB 無しのメッセージ受信

付録 I　このマニュアルの参考情報

付録 I.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）

付録 I.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

表記		製品名
SLU - TypeP2		uCosminexus TP1/NET/Secondary Logical Unit - TypeP2
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 I.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
AP	Application Program
FM	Function Management
HNA	Hitachi Network Architecture
LAN	Local Area Network
LU	Logical Unit
MCF	Message Control Facility
MHP	Message Handling Program
OS	Operating System
PLU	Primary Logical Unit
RU	Request Unit または Response Unit
SPP	Service Providing Program
TS	Transmission Services

英略語	英字での表記
UAP	User Application Program
UOC	User Own Cording

付録 I.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 J　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，一方送信メッセージの最終セグメントを送信したあと，相手システムから応答（+RSP）を受信した直後です。

GJ（メッセージ受信ジャーナル）

MCF が取得する統計用の履歴情報の一つです。入力メッセージサイズ，入力論理端末名などで構成されます。

GJ の取得タイミングは，非同期受信関数を発行して，MHP が受信メッセージを入力キューから取り出した直後です。

IJ（メッセージ入力ジャーナル）

MCF が取得する統計用の履歴情報の一つです。メッセージ入力通番，入力論理端末名，入力メッセージ種別，および入力メッセージなどで構成されます。

IJ の取得タイミングは，相手システムから受信したメッセージを入力キューに格納する直前です。

MJ（メッセージジャーナル）

MCF が取得する統計用の履歴情報の一つです。入力論理端末名または，出力論理端末名，メッセージジャーナル種別，入力または出力メッセージ（入力メッセージ編集前のデータ，または出力メッセージ編集後のデータ）などで構成されます。

MJ の取得タイミングは，入力メッセージの場合，入力メッセージ編集 UOC を呼び出す直前です。また，出力メッセージの場合，出力メッセージ編集 UOC を呼び出した直後です。

OJ（メッセージ出力ジャーナル）

MCF が取得する統計用の履歴情報の一つです。メッセージ出力通番，出力論理端末名，出力メッセージ種別，出力メッセージなどで構成されます。

OJ の取得タイミングは，非同期送信関数を発行して，UAP が送信メッセージを出力キューに格納した直後です。

SLUTYPE-P プロトコル

ホストシステムとインテリジェント端末（OpenTP1 システムの論理端末など）とを接続するためのプロトコルです。米国 IBM 社が開発したネットワークアーキテクチャである SNA に準拠しています。

SNA

米国 IBM 社が開発したネットワークアーキテクチャの名称です。SLU - TypeP2 は，このアーキテクチャに従ったプロトコルを実装しています。

（カ行）

コネクション

SLU - TypeP2 が AP 間通信をするときに，自システムと相手システムの間に確立する論理的通信路です。

（サ行）

セション

SLUTYPE-P プロトコルでの論理的通信路です。SLU - TypeP2 ではコネクションと呼びます。

（ラ行）

論理端末

SLU - TypeP2 と UAP との通信接点です。SLU - TypeP2 と UAP は，論理端末単位にメッセージを送受信します。

索引

う

お

か

き

く

こ

し

せ

そ

ゆ

よ

り

ろ