IBM Communications Server for Linux or AIX

APPC プログラマーズ・ガイド

バージョン 6.4

SC88-5825-00
(英文原典：SC23-8592-00)

IBM Communications Server for Linux or AIX

APPC プログラマーズ・ガイド

バージョン 6.4

SC88-5825-00
(英文原典：SC23-8592-00)

ご注意

本書および本書で紹介する製品をご使用になる前に、 329 ページの『付録 E. 特記事項』に記載されている情報をお読みください。

本書は、IBM Communications Server for AIX バージョン 6.4 (プログラム番号 5765-E51) および新しい版またはテクニカル・ニュースレターで明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC23-8592-00
IBM Communications Server for Linux or AIX
APPC Programmer's Guide
V6.4

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2009.4

目次

表

図

本書について

本書は、拡張プログラム間通信機能 (APPC) を使用してシステム・ネットワーク体系 (SNA) 環境でデータを交換する C プログラミング言語アプリケーション・プログラムを開発するためのガイドです。

この資料は、IBM Communications Server に適用されます。IBM Communications Server は、AIX® を実行しているサーバー、または Linux を実行しているコンピューターが SNA ネットワーク上の別のノードと情報を交換できるようにする、IBM® のソフトウェア製品です。

IBM Communications Server には、作動するためのハードウェアに応じた 3 つの異なるインストール変種があります。

IBM Communications Server for AIX (CS/AIX)
IBM Communications Server for AIX は、AIX バージョン 5.2、5.3、または 6.1 基本オペレーティング・システムを実行するサーバーで作動します。

IBM Communications Server for Linux (Communications Server for Linux)
IBM Communications Server for Linux、プログラム製品番号 5724-i33 は、次のハードウェア上で作動します。

- Linux が稼働する 32 ビット Intel ワークステーション (i686)
- Linux が稼働する 64 ビット AMD64/Intel EM64T ワークステーション (x86_64)
- Linux が稼働する IBM pSeries コンピューター (ppc64)

IBM Communications Server for Linux on System z (Communications Server for Linux on System z)
IBM Communications Server for Linux on System z (プログラム製品番号 5724-i34) は、Linux for System z が稼働する System z メインフレーム (s390 または s390x) で作動します。

本書では、相違が明示的に記述されていない限り、Communications Server という名称はこれらの変種のいずれかを示すために使用され、「Communications Server コンピューター」という用語は、Communications Server が稼働しているコンピューターのタイプを示す場合に使用されます。

Communications Server にインプリメントされた APPC は、IBM が、OS/2® 製品 (**Communications Server for OS/2** など) にインプリメントされた APPC に AIX または Linux 環境用の修正を加えたものが基礎になっています。

Communications Server にインプリメントされた APPC を使用するように作成されたプログラムは、 SNA 論理装置 (LU) 6.2 アーキテクチャーに準拠するその他のインプリメンテーション形態の APPC を使用するように作成されたプログラムとデータを交換できます。

本書は Communications Server のバージョン 6.4 に適用されます。

本書の対象読者

本書は、Communications Server がインストールされているシステム用のシステム・ネットワーク体系 (SNA) トランザクション・プログラムを作成する、熟練した C プログラマーを対象としています。ただし、SNA や Communications Server の通信機能に関する実務経験は必ずしも必要ではありません。

システム管理者

システム管理者は、Communications Server のインストール、システムのネットワーク接続の構成、およびシステムの保守を行います。システム管理者は、Communications Server が作動するハードウェアと AIX / Linux オペレーティング・システムに習熟している必要があります。また、システムを接続するネットワークについて知識があり、SNA の概念全般を理解している必要もあります。

アプリケーション・プログラマー

アプリケーション・プログラマーは、Communications Server プログラミング・インターフェースを使用して、SNA ネットワークを介してデータを送受信するトランザクションおよびアプリケーション・プログラムの設計およびコーディングを行います。したがって、アプリケーション・プログラマーは、SNA、トランザクションまたはアプリケーション・プログラムが通信するリモート・プログラム、および AIX / Linux のオペレーティング・システムのプログラミング環境と運用環境について、十分に理解していることが必要です。

アプリケーション・プログラムの作成についての詳細は、個々の API の資料に説明があります。Communications Server の資料の詳細については、『参考文献』を参照してください。

本書の使用法

この節では、本書の構成と表記について説明します。

本書の構成

本書は、次の各章で編成されています。

- 1 ページの『第 1 章 概念』では、APPC の基本概念を紹介します。 この章は、APPC にあまり精通していないプログラマー向けに書かれています。
- 29 ページの『第 2 章 トランザクション・プログラムの作成』では、APPC のプログラマーがトランザクション・プログラム (TP) を作成するときに必要となる一般的な情報を記載します。
- 73 ページの『第 3 章 APPC 制御 verb』では、APPC の制御 verb を 1 つずつ詳しく説明します。 それぞれの verb について、目的、verb 制御ブロック (VCB)、指定パラメーターと戻りパラメーター、verb を発行できる会話の状態、verb を実行したあとの会話の状態変更を説明します。 オペレーティング・システムの違いによる APPC のインプリメンテーションの差異については、発生するところで説明します。一般にこれらは、オペレーティング・システムの違いによる、小規模な差異となります。
- 105 ページの『第 4 章 APPC 会話 verb』では、APPC の会話 verb を 1 つずつ詳しく説明しています。 それぞれの verb について、目的、verb 制御ブロック

(VCB)、指定パラメーターと戻りパラメーター、verb を発行できる会話の状態、verb を実行したあとの会話の状態変更を説明します。 オペレーティング・システムの違いによる APPC のインプリメンテーションの差異については、発生するところで説明します。一般にこれらは、オペレーティング・システムの違いによる、小規模な差異となります。

- 277 ページの『第 5 章 TP サーバー verb』では、APPC の TP サーバー verb を 1 つずつ詳しく説明しています。 それぞれの verb について、目的、verb 制御ブロック (VCB)、指定パラメーターと戻りパラメーター、verb を発行できる会話の状態、 verb を実行したあとの会話の状態変更を説明します。 オペレーティング・システムの違いによる APPC のインプリメンテーションの差異については、発生するところで説明します。一般にこれらは、オペレーティング・システムの違いによる、小規模な差異となります。
- 295 ページの『第 6 章 トランザクション・プログラムの例』では、APPC verb の使用法を示す APPC トランザクション・プログラムのサンプルが説明されています。この章には、サポートされるオペレーティング・システムごとの TP のコンパイル、リンク、実行の説明が含まれています。
- 299 ページの『付録 A. 戻りコードの値』 は、APPC インターフェースの発生しうるすべての戻りコードを番号順にリストし、意味を説明します。
- 307 ページの『付録 B. 共通戻りコード』では、いくつかの verb に共通した特定の 1 次および 2 次戻りコードについて説明します。
- 319 ページの『付録 C. APPC の状態の変更』では、APPC の会話の状態、つまり、それぞれの状態でどの verb が許可されているか、また、各 verb からの戻りで会話がどのような状態に変更されるかについて説明します。
- 325 ページの『付録 D. SNA LU6.2 サポート』には、Communications Server としてインプリメンテーションされた APPC が SNA LU6.2 アーキテクチャーにどのように関連付けられているかについての参照情報と、Communications Server ではコマンド行管理プログラム **snaadmin** および NOF (ノード・オペレーター機能) verb によって機能が提供される LU6.2 制御オペレーター verb に関する情報が記載されています。

表記上の規則

本書では、表 1 に示すような表記上の規則を使用しています。

表 1. 表記上の規則

内容	表記例
資料名	*IBM Communications Server for Linux or AIX APPC アプリケーション・スイート*
ファイル名またはパス名	**sna_tps**
プログラムまたはアプリケーション	**snaadmin**
コマンドまたは AIX / Linux ユーティリティー	**vi**; **define_mode**
特定のタイプのすべての値への一般的な参照	`AP_SEC_BAD_*` (`AP_SEC_BAD` で始まるすべての戻り値を示す)
オプションまたはフラグ	**-I**
パラメーター・フィールドまたは Motif フィールド	*primary_rc*; *what_rcvd*

表 1. 表記上の規則 (続き)

内容	表記例
ユーザーが入力できるリテラル値または選択項目 (デフォルト値を含む)	`0`; `32,767`
定数またはシグナル	`AP_DATA_COMPLETE_CONFIRM`
戻り値	`AP_OK`; `AP_SYNC_LEVEL_NOT_SUPPORTED`; `TRUE`
指定値を表す変数	*lParam*; *ReturnedHandle*
環境変数	`LD_RUN_PATH`
プログラミング verb	RECEIVE_ALLOCATE
ユーザーの入力	**cc -I**
関数、コール、またはエントリー・ポイント	`APPC_Async`; `WinAsyncAPPC`
データ構造	`WAPPCDATA`
16 進値	`0x20`

シンボルの意味

AIX, LINUX

このシンボルは、AIX または Linux オペレーティング・システムだけに該当する説明のセクション開始を表します。これは AIX / Linux サーバーと、AIX、 Linux、Linux for pSeries、または Linux for System z 上で稼働する IBM Remote API Client に適用されます。

WINDOWS

このシンボルは、Windows 上の IBM Remote API Client に該当する説明のセクション開始を表します。

このシンボルは、オペレーティング・システム固有の説明のセクションの終了を表します。このシンボルに続く情報は、どのオペレーティング・システムにも適用されます。

詳細について

Communications Server ライブラリーのその他の資料、および SNA ワークステーションと AIX / Linux ワークステーション関連事項についての追加情報は、参考文献を参照してください。

第 1 章 概念

この章では、分散処理環境における拡張プログラム間通信 (APPC) の次のような基本概念を紹介します。

- APPC とは
- 単純なマップ式会話
- 確認処理
- データと共に状況を送受信
- 会話状態
- 会話状態の変更
- 同期および非同期の APPC コール
- データの非同期受信

AIX, LINUX

- 同期点サポート

- APPC と CPI-C (共通プログラミング・インターフェース・コミュニケーション)
- TP サーバー API

APPC とは

APPC とは拡張プログラム間通信 (Advanced Program-to-Program Communication) のことで、これはシステム・ネットワーク体系 (SNA) 環境内のプログラム間で対等通信を可能にするアプリケーション・プログラム・インターフェース (API) です。

APPC を介して、ネットワーク内に分散されたアプリケーション・プログラムは互いに通信し、データを交換しながら共同して、次のような単一の処理タスクを達成します。

- リモート・データベースの照会
- リモート・ファイルのコピー
- 電子メールの送受信

1 つ以上の処理タスクを達成するために 2 つのアプリケーション・プログラム間で行われる一連の完全な通信を、会話と呼びます。

交信する 2 つの APPC アプリケーションは、同じコンピューター上にあっても、2 つの別々のコンピューター上にあっても構いません。各アプリケーションは、パートナー・アプリケーションの位置を認識している必要はありません。 APPC アプリケーションは、サーバーまたはクライアント・コンピューターのいずれかで稼働します。

トランザクション・プログラム

トランザクションとは、APPC を使用しているプログラムによって実行される処理タスクのことです。したがって、APPC を使用するプログラムはトランザクション・プログラム (TP) と呼ばれます。それらのプログラムに階層関係はなく、対等なピアとして通信します。アプリケーション TP は、エンド・ユーザーのためのタスクを実行します。サービス TP は、他のプログラムへのサービスを提供します。

ローカル・ネットワークまたは広域ネットワーク内に分散された TP は、共同して分散トランザクション処理を行います。

TP 間の通信

2 つの TP が互いに通信するためには、SNA 環境で多数のハードウェアとソフトウェアのエレメントが必要です。 図 1 に、TP を作成するプログラマーに直接関係するエレメントを示します。

図 1. TP を作成するためのエレメント

論理装置 6.2

1 つの TP は特定の Communications Server ローカル・ノード上にある 1 つの論理装置 (LU) へ関連付けられ、その LU を通じてネットワークにアクセスします。複数の TP を同じ LU へ関連付けることもできます。個々の LU タイプは、特定のプロトコルを使用します。 APPC は、LU6.2 によってサポートされています。

セッション

2 つの TP が通信するためには、前もってそれらの LU が LU 間セッションを通じて接続されていなければなりません。 LU 間セッションとは、2 つの LU の間の論理接続のことです。このセッションのモード (ネットワーク機能の属性セット) は、2 つの LU 間をデータが移動する方法を決定します。

会話

2 つの TP 間の通信は、LU 間セッション内の会話として発生します。 1 つの TP が同時にいくつかの会話に関係することもできます。

APPC verb

TP は、APPC verb を介して APPC にアクセスします。 TP はそれらの verb を使用して、会話の開始、データの送受信、会話の終了を APPC に指示します。各 verb は APPC にパラメーターを提供し、APPC は指定された機能を実行して TP にパラメーターを戻します。 verb は何らかのローカル処理 (たとえば、少量のデータの送信など) のあと迅速に完了するものもあれば、パートナー TP と通信パスに応じて完了までに多少時間がかかるものもあります (たとえば、パートナー TP からのデータや確認を待つため)。

verb を発行する TP はローカル TP と呼ばれ、もう一方の TP はパートナー TP と呼ばれます。

同様に、ローカル TP にサービスする LU はローカル LU であり、パートナー TP にサービスする LU はパートナー LU です。

他のネットワーク・ノード上にある TP および LU も、リモート TP およびリモート LU と呼ばれます。

会話プロセス

会話は、次の両方が発生した時点で開始されます。

1. 一方の TP (呼び出し元 TP) が APPC に他方の TP (呼び出し先 TP) を開始して 2 つの TP 間に会話を割り振るよう指示します。
2. 呼び出し先 TP は、呼び出し元 TP と通信する準備ができていることを APPC に通知します。

会話が行われている間、2 つの TP は状況情報とアプリケーション・データを交換します。

会話は、いずれかの TP が APPC に会話の割り振りを解除するよう指示した時点で終了します。

会話タイプ

会話は、マップ式会話と基本会話のどちらかです。 呼び出し元 TP は、割り振り時に会話を基本会話とマップ式会話のどちらにするかを指定します。特定の APPC verb はマップ式会話でのみ使用され、それ以外は基本会話で使用されます。基本会話 verb をマップ式会話に使用することはできず、その逆も同様です。

一般に、マップ式会話はアプリケーション TP によって使用されます。アプリケーション TP は、エンド・ユーザーのためのタスクを実行するプログラムです。マップ式会話は、基本会話より単純です。マップ式会話では、送信側 TP は一度に 1 つのレコードを送信し、受信側 TP は一度に 1 つのレコードを受信します。

通常、基本会話はサービス TP によって使用されます。サービス TP は、他のローカル・プログラムにサービスを提供するプログラムです。熟練した LU6.2 プログラマーならば、基本会話を使用してかなりの程度、データの伝送と処理を制御できます。 詳細については、67 ページの『基本会話』を参照してください。

複数の会話

1 つの TP が同時にいくつかの会話に関係することもできます。 1 つの会話には 1 つの LU 間セッションが必要です。

複数の会話が一般的に使用されるのは、呼び出し先 TP が別の TP を呼び出し、その TP がさらに別の TP を呼び出すというような場合です。 図 2 は、TP A が TP B を呼び出し、TP B が TP C を呼び出す様子を示しています。TP A および TP C は互いに会話せず、TP B とのみ会話します。

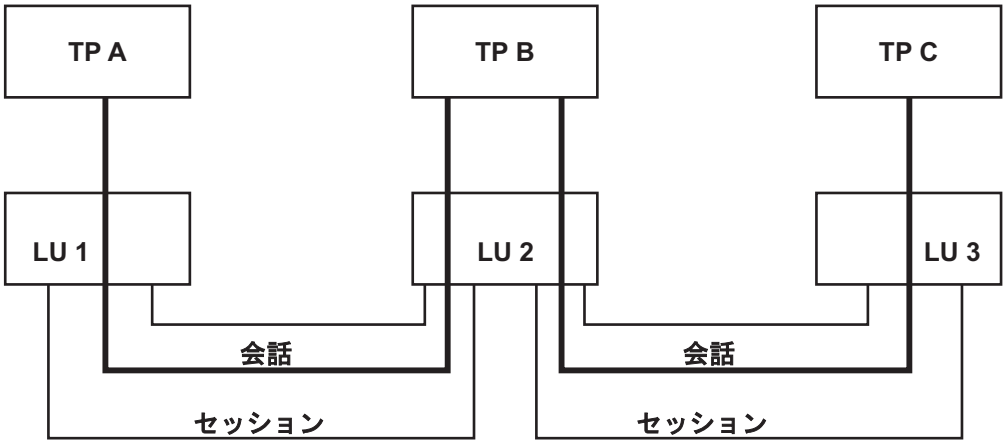

図 2. 複数の会話を使用している TP の呼び出し

半二重および全二重会話

会話内の 2 つの TP がどのように相互に作用する必要があるかによって、会話は次の 2 つの方法で作動します。

半二重 (Half-duplex)

半二重会話では、2 つの TP 間で制御が受け渡しされるため、常に 1 つの TP が会話を制御することになります。制御中の TP は、データを送信するか、または他の TP へ制御を渡すことができます。他の TP は、データを受信することはできますが、送信はできません。ただし、エラー通知を制御中の TP に送るか、または会話の制御を要求することはできます。

半二重会話は、通常は次の状態で使用されます。

- 会話の進行が、伝送されるデータにより異なる場合。例えば、最初の TP が 2 番目の TP が処理しなければならない要求を送信し、この両方の TP の次の操作が、要求が承認またはリジェクトされるかにより異なる場合です。
- 最初の TP は 2 番目の TP にデータを送るだけで、応答を受け取る必要がない場合。

制御は 2 つの TP 間で受け渡しされるため、半二重会話では処理を続ける前に、TO が会話の各段階を確認できます。最初の TP は、一定量のデータを送信した後、処理を続ける前にデータが受信され、正しく処理されたかどうか確認を要求することができます。この時点で、2 番目の TP は、データを確認して、最初の TP がさらにデータを送信できるようにできます。ある

いは、エラー応答を戻してから、2 番目の TP 自身が会話を制御する (エラーの詳細を提供する、またはデータの修正済みバージョンを戻すなど) こともできます。

全二重 (Full-duplex)
全二重会話では、両方の TP がデータをいつでも送信することができます。どちらの TP も制御されているとはみなされません。

全二重会話は、通常 2 方向に送信されるデータが独立している場合に使用されます。そのため、会話の進行は、伝送されるデータに依存しません。どちらの TP も会話を制御しているとみなさないため、確認処理はサポートされません。TP はいつでもエラー応答を送信することができますが、これにより他の TP がデータの送信を続行できなくなることはありません。

会話を割り振る TP は、会話が全二重または半二重のどちらで動作するかどうかを指定します。この選択は会話が続いている間ずっと適用されます。会話中に全二重操作と半二重操作を切り替えることはできません。全二重会話で verb を発行するには、 105 ページの『第 4 章 APPC 会話 verb』で説明しているように、TP が verb の *opext* パラメーターに追加オプションを設定する必要があります。

すべての APPC インプリメンテーションが全二重操作をサポートしているわけではありません。リモート TP が全二重操作をサポートしない APPC インプリメンテーションを使用していると、 TP は半二重モードでのみ作動します。

以降のセクションでは、半二重会話の操作について説明します。全二重会話には、TP 間での確認処理と制御の受け渡しに使用する verb が含まれていないため、以降のセクションには、全二重会話に適用されない情報があります。全二重会話の操作の要約については、 15 ページの『全二重会話』 を参照してください。

単純なマップ式会話 (半二重)

表 2 に、会話の開始、データの交換、会話の終了に使用する APPC verb を示した単純なマップ式会話の例を示します。矢印は、データのフローを示します。すべての *primary_rc* 値は、特に断りがない限り AP_OK です。

表 2. 単純なマップ式会話

呼び出し元 TP	フロー	呼び出し先 TP
TP_STARTED MC_ALLOCATE MC_SEND_DATA MC_DEALLOCATE		
	—>	
TP_ENDED		RECEIVE_ALLOCATE MC_RECEIVE_AND_WAIT (*what_rcvd* = AP_DATA_COMPLETE) MC_RECEIVE_AND_WAIT (*primary_rc*=AP_DEALLOC_NORMAL) TP_ENDED

多くの verb の先頭に付く MC_ という文字は、マップ式会話を意味しています。APPC verb のパラメーターと結果は、括弧でくくります。

会話の開始

会話を開始するため、呼び出し元 TP は次の verb を発行します。

- TP_STARTED。これは APPC に、アプリケーションを呼び出し元 TP として示します。
- MC_ALLOCATE。これは APPC に、呼び出し元 TP と呼び出し先 TP の間に会話を確立するよう要求します。

呼び出し先 TP は、RECEIVE_ALLOCATE verb を発行し、この verb は、呼び出し先 TP が会話を開始する準備ができていることを APPC に通知します。 Communications Server は、2 つの TP 間に会話を確立するため、RECEIVE_ALLOCATE verb を呼び出し元 TP が発行した MC_ALLOCATE verb へ関連付けます。

データの送信

MC_SEND_DATA verb は、パートナー TP へ伝送するアプリケーション・データを提供します。 このデータはローカル LU の送信バッファーに保持され、次のいずれかのイベントが発生するまでパートナー TP へ伝送されません。

- 送信バッファーがいっぱいになった。
- TP が APPC に強制的にバッファーをフラッシュさせる (パートナー TP へデータを送信させる) verb を発行する

送信バッファーには、データのほかに MC_ALLOCATE 要求 (これはデータに先行します) が入っています。この例では、MC_DEALLOCATE verb はバッファーをフラッシュし、 MC_ALLOCATE 要求とデータをパートナー TP へ伝送します。バッファーをフラッシュするその他の verb としては、 MC_CONFIRM と MC_FLUSH があります。

データの受信

MC_RECEIVE_AND_WAIT verb は、パートナー TP からデータを受信します。現在使用可能なデータがない場合、ローカル TP はデータが到着するまで待ちます。

この verb は、データを受信するのみでなく、パートナー TP から状況インディケーター (会話の終了を示すインディケーターやデータの受信を確認する要求など) を受信する場合もあります。 TP がどのようにこれらのインディケーターを処理するかについての詳細は、7 ページの『確認処理 (半二重)』と 11 ページの『会話状態 (半二重)』を参照してください。

この例では、呼び出し先 TP は、データを受信するために最初の MC_RECEIVE_AND_WAIT verb を発行します。 呼び出し先 TP は、完全なデータ・レコード (*what_rcvd* = AP_DATA_COMPLETE) の受信を終了すると、再度 MC_RECEIVE_AND_WAIT verb を発行して戻りコードを受信します。 戻りコード AP_DEALLOC_NORMAL は、会話の割り振りが解除されたことを示します。

MC_RECEIVE_IMMEDIATE verb は MC_RECEIVE_AND_WAIT verb と同じ機能を実行しますが、データがパートナー TP から現在入手可能でない場合は待機しません。 その代わりに、呼び出し元 TP にデータ使用不可の応答を戻します。

会話の終了

会話を終了するには、どちらか一方の TP が MC_DEALLOCATE verb を発行し、それによって APPC が 2 つの TP 間の会話の割り振りを解除します。

この会話の割り振りが解除されたあと、それぞれの TP は別の [MC_]ALLOCATE verb か RECEIVE_ALLOCATE verb を発行して (このパートナー TP または別のパートナー TP と) 別の会話を開始するか、TP_ENDED verb を発行します。

1 つの TP が同時に複数の会話に参加することもできます。この場合、TP はすべての会話の割り振りが解除されたときに TP_ENDED verb を発行します。

確認処理 (半二重)

TP は、確認処理を使用してデータと共に確認要求を送信します。受信側 TP はデータの受信を確認するか、エラーが発生したことを知らせます。 2 つの TP は、確認要求と応答を交換するたびに同期を取ります。

表 3 の確認処理の例は、データの転送を確認する 2 つの方法、つまりデータを送信したあとの確認の要求 (CONFIRM verb を使用) と、トランザクション終了時の確認の要求 (DEALLOCATE verb で確認を要求) を示しています。 確認は、PREPARE_TO_RECEIVE verb でも要求できます。この verb は、データの受信を確認してデータ自体の送信を開始するようパートナー TP に依頼します。詳細については、12 ページの『状態の変更』を参照してください。 TP のペアは、これらのメカニズムの 1 つのみを使用することを選択しても構いません。次の例では、呼び出し先 TP は確認を要求せずに PREPARE_TO_RECEIVE verb を使用しています。この verb は、単にデータを送信するようパートナー TP に指示します。

表 3. 確認処理

呼び出し元 TP	フロー	呼び出し先 TP
TP_STARTED MC_ALLOCATE (*sync_level* = AP_CONFIRM_SYNC_LEVEL) MC_SEND_DATA MC_CONFIRM		
	—>	
		RECEIVE_ALLOCATE MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_DATA_COMPLETE) MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_CONFIRM) MC_SEND_ERROR
	<—	
(MC_CONFIRM が戻る。*primary_rc* = AP_PROG_ERROR_PURGING)		
		MC_PREPARE_TO_RECEIVE (*ptr_type* = AP_FLUSH)
	<—	

表 3. 確認処理 (続き)

呼び出し元 TP	フロー	呼び出し先 TP
MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_SEND) MC_SEND_DATA MC_CONFIRM		
	—>	
		MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_DATA_COMPLETE) MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_CONFIRM) MC_CONFIRMED
	<—	
MC_DEALLOCATE (*dealloc_type* = AP_SYNC_LEVEL)		
	—>	
		MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* =AP_CONFIRM_DEALLOCATE) MC_CONFIRMED
	<—	
TP_ENDED		TP_ENDED

同期レベルの確立

MC_ALLOCATE verb の *sync_level* パラメーターは、会話の同期レベルを決定します。 同期レベルに指定できる値は、次のとおりです。

- AP_CONFIRM_SYNC_LEVEL。この場合、TP はデータの受信の確認を要求でき、データの受信を確認するような要求に応答することができます。
- AP_NONE。この場合、確認処理は発生しません。

AIX, LINUX

3 番目のレベルとして AP_SYNCPT (同期点) も使用できますが、これには追加ソフトウェアが必要です。 詳細については、25 ページの『同期点サポート』を参照してください。

確認要求の送信

MC_CONFIRM verb には次の 2 つの効果があります。

- ローカル LU の送信バッファーをフラッシュする。これによって、バッファーに入っているデータがパートナー TP へ送信されます。
- 確認要求を送信する。これは、パートナー TP が受信 verb の *what_rcvd* パラメーターを介して受信します。

MC_CONFIRM verb は、確認 (またはエラーが検出されたことを示す通知) をパートナー TP から受信するまで完了しません。

データと確認要求の受信

MC_RECEIVE_AND_WAIT verb の *what_rcvd* パラメーターは、次のことを示します。

- 受信したデータの状況が完全であるか不完全である
- ローカル TP に要求される将来の処理 (たとえば、確認要求またはデータの送信を開始せよという指示)

呼び出し先 TP は、完全なデータ・レコード (*what_rcvd* = AP_DATA_COMPLETE) の受信を終了すると、再度 MC_RECEIVE_AND_WAIT verb を発行して確認要求 (*what_rcvd* = AP_CONFIRM) を受信します。

確認要求への応答

通常、呼び出し先 TP は MC_CONFIRMED verb を発行してデータの受信を確認します。これは、処理を再開するために呼び出し元 TP を解放します。

呼び出し先 TP は、受信したデータの中でエラーを検出すると、MC_SEND_ERROR verb を発行してそのエラー条件を示すことができます。

会話の割り振り解除

MC_DEALLOCATE verb は、次の両方の条件が該当する場合、データと共に確認要求を送信します。

- 会話の同期レベル (MC_ALLOCATE verb の *sync_level* パラメーターによって確立されたもの) が AP_CONFIRM_SYNC_LEVEL である
- MC_DEALLOCATE verb の *dealloc_type* パラメーターが AP_SYNC_LEVEL に設定されている

呼び出し先 TP から発行された最終の MC_RECEIVE_AND_WAIT verb の *what_rcvd* パラメーターには、APPC が会話を割り振り解除する前にデータ受信の確認が必要であることを示す AP_CONFIRM_DEALLOCATE が入っています。 呼び出し元 TP は、呼び出し先 TP が MC_CONFIRMED verb を発行してデータが正常に受信されたことを示すまで (またはデータが正常に受信されなかったことを示す MC_SEND_ERROR verb を発行するまで)、この確認を待ちます。

データと共に状況を送信および受信する (半二重)

7 ページの表 3 では、呼び出し元 TP は MC_SEND_DATA verb を使用してデータを送信したあと、MC_CONFIRM verb を使用して呼び出し先 TP からの確認を要求しました。 このように異なる 2 つの verb を発行する代わりに、[MC_]SEND_DATA verb にパラメーターを使用すると、データの送信後に [MC_]CONFIRM verb (または [MC_]DEALLOCATE、[MC_]FLUSH、[MC_]PREPARE_TO_RECEIVE のいずれか) の機能を実行するよう APPC に指示することができます。

同様に、 7 ページの表 3 の呼び出し先 TP は、MC_RECEIVE_AND_WAIT verb を 2 回発行しました。1 回目はデータを受信するため、2 回目は呼び出し元 TP が確認を要求しているという状況情報を受信するためでした。この場合も、異なる 2 つの受信 verb を発行しなくても、いずれかの [MC_]RECEIVE verb にパラメーターを使用すれば、データと同じ受信 verb についての状況情報を戻すよう APPC に指示することができます。

表 4 は、MC_SEND_DATA に「送信タイプ」パラメーターを使用して MC_CONFIRM verb の機能を実行し、MC_RECEIVE_AND_WAIT に「データと共に状況を戻す」パラメーターを使用してデータと共に状況情報を受信する方法を示しています。 すべての *primary_rc* 値は、特に断りがない限り、 AP_OK と見なすことができます。

表 4. データと共に状況情報を受信

呼び出し元 TP	フロー	呼び出し先 TP
TP_STARTED MC_ALLOCATE MC_SEND_DATA (*type* = AP_SEND_DATA_CONFIRM)		
	—>	
		RECEIVE_ALLOCATE MC_RECEIVE_AND_WAIT (*rtn_status* = NO) (*what_rcvd*= AP_DATA_COMPLETE) MC_RECEIVE_AND_WAIT (*rtn_status* = NO) (*what_rcvd* = AP_CONFIRM_WHAT_RECEIVED) MC_CONFIRMED
	<—	
MC_SEND_DATA (*type* = AP_NONE) MC_CONFIRM		
	—>	
		MC_RECEIVE_AND_WAIT (*rtn_status* = YES) (*what_rcvd* = AP_DATA_COMPLETE_CONFIRM) MC_CONFIRMED
	<—	
MC_DEALLOCATE TP_ENDED		
	—>	
		MC_RECEIVE_AND_WAIT (*primary_rc*=AP_DEALLOC_NORMAL) TP_ENDED

データと共に状況情報を送信

表 4 では、呼び出し元 TP が最初に発行した MC_SEND_DATA verb の送信タイプは AP_SEND_DATA_CONFIRM です。これは、データの送信後に MC_CONFIRM verb の機能を実行するよう APPC に指示します。 [MC_]SEND_DATA verb は、CONFIRM の代わりに [MC_]FLUSH、 [MC_]DEALLOCATE、または

[MC_]PREPARE_TO_RECEIVE の機能を実行したり、 (この例の 2 番目の MC_SEND_DATA verb の場合のように) 追加機能を実行しないことも指定できます。

データと共に状況情報を受信

10 ページの表 4 で、呼び出し先 TP が発行した 3 番目の MC_RECEIVE_AND_WAIT verb には、AP_YES の *rtn_status* パラメーターがあります。 これは、状況情報を従えたデータがあれば、APPC はデータのほかにこの verb に関する状況情報を戻せることを示しています。 この verb は、AP_DATA_COMPLETE_CONFIRM の *what_rcvd* パラメーターを伴って戻り、これは、呼び出し元 TP がデータを送信し、そのあとに確認を要求したことを示しています。 最初の 2 つの MC_RECEIVE_AND_WAIT verb では、*rtn_status* パラメーターが AP_NO に設定されているので、APPC はデータと共に状況情報を戻しません。最初の verb はデータを受信し、2 番目の verb は状況情報を受信します。

会話状態 (半二重)

半二重会話では、APPC は半二重処理として作動します。これは、データの送信を許可されるのは、常に 2 つの TP のいずれか一方のみであることを意味します。一般に、一方の TP が特定の量のデータを送信したあと、次のどちらかを実行します。

- 他の TP に、データの受信を確認するよう要求する
- 他の TP に送信を許可する

TP は、任意の時点の会話でも特定の会話状態にあると見なします。会話状態は、TP が特定の時点でどの APPC verb を発行できるかを制御します。たとえば、会話状態が Send と Send-Pending のどちらでもない場合、TP は MC_SEND_DATA verb を発行できません。それぞれの状態で発行できる APPC verb についての詳細は、 319 ページの『付録 C. APPC の状態の変更』を参照してください。

可能な会話状態のリストを次に示します。

Confirm (確認)
TP はデータ受信の確認要求を受信しました。TP は肯定的な応答をするか、パートナー TP にエラー情報を送信しなければなりません。

Confirm-Deallocate (確認 - 割り振り解除)
TP は確認の要求を受信したため、肯定的な応答をするかエラー情報を送信しなければなりません。 TP が肯定的な応答をした場合、パートナー TP は、会話を割り振り解除します。

Confirm-Send (確認 - 送信)
TP は確認の要求を受信したため、肯定的な応答をするかエラー情報を送信しなければなりません。応答のあと、TP はデータの送信を開始できます。

Pending-Post (保留 - 通知)
TP はデータを非同期に受信中です。 TP は、その会話に関連していない別の処理を実行できます。 TP がデータの受信を終了すると、通常の場合、状態は Receive になります。

Receive (受信)
TP は、パートナー TP からアプリケーション・データと状況情報を受信できます。会話が Receive 状態の場合、TP はエラー情報を送信してデータを送信する許可を要求することもできます。

Reset (リセット)
会話は、まだ開始されていないか、すでに終了しました。

Send (送信)
TP はパートナー TP にデータを送信して確認を要求できます。会話が Send 状態の場合、TP はデータの受信を開始することもでき、それによって状態が Receive に変化します。

Send-Pending (送信 - 保留)
TP は、パートナー TP からデータと SEND 指示を受信しました。この状態は Send 状態によく似ていますが、TP は [MC_]SEND_ERROR verb に追加のパラメーターを使用して、検出したエラーのソースを示すことができます。

TP から見た会話

特定の状態に置かれるのは、TP ではなくむしろ会話です。 TP は複数の会話を行うことができ、それぞれの会話が異なる状態に置かれる場合があります。会話が Send 状態になっている場合、これは常に、ローカル TP から見て、会話が Send 状態にあることを意味します。パートナー TP から見た場合、その会話は別の状態 (Receive など) になります。

状態の変更

会話状態の変更は、APPC verb の完了時に発生します。状態の変更は、次のいずれかの結果として発生します。

- ローカル TP から発行された verb (たとえば、Send 状態で RECEIVE_AND_WAIT を発行すると、会話状態は Receive に変更されます。)
- パートナー TP から発行された verb (たとえば、パートナー TP が CONFIRM verb を発行した場合、ローカル TP は、そのことを示す指示をいずれかの受信 verb で受信し、その時点で、会話は Receive 状態から Confirm 状態に変更されます。)
- エラー条件

一般に、会話の新しい状態は、完了した APPC verb の 1 次戻りコードによって異なります。詳細については、73 ページの『第 3 章 APPC 制御 verb』および 105 ページの『第 4 章 APPC 会話 verb』の verb の説明を参照するか、319 ページの『付録 C. APPC の状態の変更』を参照してください。

状態検査

状態検査 (状態エラー) は、TP が APPC verb を発行したとき、会話が適切な状態にない場合に発生します。 たとえば、会話が Receive 状態のときに TP が MC_SEND_DATA verb を発行した場合、状態検査が発生します。状態検査が発生すると、APPC は verb を実行せず、 1 次および 2 次戻りコードを通じて状態検査情

報を戻します。 verb が戻す状態検査エラー・コードについての詳細は、 73 ページの『第 3 章 APPC 制御 verb』および 105 ページの『第 4 章 APPC 会話 verb』の verb の説明を参照してください。

会話状態の変更 (半二重)

表 5 では、会話状態を左右の端に示します。この表に、APPC verb によって会話の状態が Send から Receive に、また Receive から Send に変更される様子を示します。

表 5. APPC verb を使用した会話状態の変更

状態	呼び出し元 TP	フロー	呼び出し先 TP	状態
	TP_STARTED			
Reset (リセット)				
	MC_ALLOCATE(*sync_level* = AP_CONFIRM_SYNC_LEVEL)			
Send (送信)				
	MC_SEND_DATA MC_PREPARE_TO_RECEIVE (*ptr_type* = AP_SYNC_LEVEL)			
		—>		
				Reset (リセット)
			RECEIVE_ALLOCATE	
				Receive (受信)
			MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_DATA_COMPLETE) MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_CONFIRM_SEND)	
				Confirm-Send (確認 - 送信)
			MC_CONFIRMED	
				Send (送信)
		<—		
	(MC_PREPARE_TO_RECEIVE が戻る。*primary_rc*= AP_OK)			
Receive (受信)				
			MC_SEND_DATA MC_CONFIRM	
		<—		
	MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_DATA_COMPLETE) MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_CONFIRM)			
Confirm (確認)				

表 5. APPC verb を使用した会話状態の変更 (続き)

状態	呼び出し元 TP	フロー	呼び出し先 TP	状態
	MC_REQUEST_TO_SEND MC_CONFIRMED			
Receive (受信)				
		—>		
			(MC_CONFIRM が戻る。*primary_rc* = AP_OK, *rts_rcvd* = AP_YES)	
				Send (送信)
			MC_PREPARE_TO_RECEIVE (*ptr_type* = AP_SYNC_LEVEL)	
		<—		
	MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_CONFIRM_SEND)			
Confirm-Send (確認 - 送信)				
	MC_CONFIRMED			
		—>		
			(MC_PREPARE_TO_RECEIVE が戻る。*primary_rc*= AP_OK)	
				Receive (受信)
Send (送信)				
	MC_SEND_DATA MC_DEALLOCATE (*dealloc_type* = AP_SYNC_LEVEL)			
		—>		
			MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* = AP_DATA_COMPLETE) MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd* =AP_CONFIRM_DEALLOCATE)	
				Confirm-Deallocate (確認 - 割り振り解除)
			MC_CONFIRMED	
		<—		
			(MC_DEALLOCATE が戻る。*primary_rc* = AP_OK)	
Reset (リセット)				Reset (リセット)
	TP_ENDED		TP_ENDED	

初期状態

会話が割り振られる前に、両方の TP の状態が Reset になります。会話が割り振られたあと、初期状態は、呼び出し元 TP では Send、呼び出し先 TP では Receive になります。

Receive 状態への変更

MC_PREPARE_TO_RECEIVE verb を使用すると、TP は会話を Send 状態から Receive 状態に変更できます。この verb は、次の動作を行います。

- ローカル LU の送信バッファーをフラッシュします。
- 受信 verb の *what_rcvd* パラメーターを通じて、パートナー TP に AP_CONFIRM_SEND インディケーターを送信します。 このインディケーターは、パートナー TP がデータの送信を開始する前に MC_CONFIRMED 応答が予期されていることをパートナー TP に知らせます。
- 次の各条件が真であるので、確認処理を実行します。
  - 会話の同期レベルが AP_CONFIRM_SYNC_LEVEL に設定されている
  - パラメーター *ptr_type* が AP_SYNC_LEVEL に設定されている

会話が Send 状態のときに MC_RECEIVE_AND_WAIT verb を発行すると、LU の送信バッファーのフラッシュも行われ、会話状態が Receive に変更されます。この方法で会話状態を変更する場合、確認処理はサポートされません。

Send 状態への変更

MC_REQUEST_TO_SEND verb は、パートナー TP (会話は Send 状態) に、ローカル TP (会話は Receive 状態) がデータの送信を要求していることを通知します。

この要求は、MC_CONFIRM verb の *rts_rcvd* パラメーターを通じてパートナー TP へ通信されます (*rts_rcvd* パラメーターは、MC_SEND_DATA およびその他の verb にも戻されます)。

パートナー TP が MC_PREPARE_TO_RECEIVE verb を発行した場合、会話状態はパートナー TP では Receive に変更され、ローカル TP はデータを送信できるようになります。

MC_REQUEST_TO_SEND verb を発行しても、会話の状態は変更されません。 送信要求を受信した場合、パートナー TP は会話状態を変更する必要はありません。パートナー TP はその要求を無視できます。

全二重会話

表 6 に、会話の開始、データの交換、会話の終了に使用する APPC verb を示した全二重会話話の例を示します。矢印は、データのフローを示します。すべての *primary_rc* 値は、特に断りがない限り AP_OK です。

表 6. 全二重会話

呼び出し元 TP	フロー	呼び出し先 TP
TP_STARTED MC_ALLOCATE		

表 6. 全二重会話 (続き)

呼び出し元 TP	フロー	呼び出し先 TP
MC_SEND_DATA		
		RECEIVE_ALLOCATE
MC_SEND_DATA	—>	
	<—	MC_SEND_DATA
MC_RECEIVE_AND_WAIT (*what_rcvd* = AP_DATA_COMPLETE)		
MC_DEALLOCATE		
		MC_RECEIVE_AND_WAIT (*what_rcvd* = AP_DATA_COMPLETE)
		MC_RECEIVE_AND_WAIT (*primary_rc*=AP_DEALLOC_NORMAL)
	<—	MC_SEND_DATA
		MC_DEALLOCATE
		TP_ENDED
MC_RECEIVE_AND_WAIT (*what_rcvd* = AP_DATA_COMPLETE)		
MC_RECEIVE_AND_WAIT (*what_rcvd* = AP_DEALLOC_NORMAL)		
TP_ENDED		

多くの verb の先頭に付く MC_ という文字は、マップ式会話を意味しています。APPC verb のパラメーターと結果は、括弧でくくります。

会話の開始

会話を開始するため、呼び出し元 TP は次の verb を発行します。

- TP_STARTED。これは APPC に、アプリケーションを呼び出し元 TP として示します。
- MC_ALLOCATE。これは APPC に、呼び出し元 TP と呼び出し先 TP の間に会話を確立するよう要求します。MC_ALLOCATE verb のパラメーターは、会話が全二重となることを指定します。

呼び出し先 TP は、RECEIVE_ALLOCATE verb を発行し、この verb は、呼び出し先 TP が会話を開始する準備ができていることを APPC に通知します。Communications Server は、2 つの TP 間に会話を確立するため、RECEIVE_ALLOCATE verb を呼び出し元 TP が発行した MC_ALLOCATE verb へ関連付けます。RECEIVE_ALLOCATE verb の戻りパラメーターは、会話が全二重となることを指定します。

注: 全二重会話の開始後、TP が全二重モードで動作するためには、この会話で発行されたすべての verb の *opext* パラメーターに追加オプションを設定する必要があります。詳細については、 105 ページの『第 4 章 APPC 会話 verb』の各 verb の説明を参照してください。

データの送信

MC_SEND_DATA verb は、パートナー TP へ伝送するアプリケーション・データを提供します。 このデータはローカル LU の送信バッファーに保持され、次のいずれかのイベントが発生するまでパートナー TP へ伝送されません。

- 送信バッファーがいっぱいになった。
- TP が APPC に強制的にバッファーをフラッシュさせる (パートナー TP へデータを送信させる) verb を発行する

送信バッファーには、データのほかに MC_ALLOCATE 要求 (これはデータに先行します) が入っています。この例では、呼び出し元 TP の 2 番目の MC_SEND_DATA verb がバッファーを埋め、MC_ALLOCATE 要求とデータをパートナー TP へ強制的に伝送します。バッファーをフラッシュするその他の verb としては、 MC_DEALLOCATE と MC_FLUSH があります。

これは全二重会話であるため、両方の TP がデータを同時に送信することができます。この例では、呼び出し先 TP は、呼び出し元 TP によって送信されるデータを受け取る前に、データを送信します。

データの受信

MC_RECEIVE_AND_WAIT verb は、パートナー TP からデータを受信します。現在使用可能なデータがない場合、ローカル TP はデータが到着するまで待ちます。

この verb は、データを受信するのみでなく、パートナー TP から状況インディケーター (会話の終了を示すインディケーターなど) を受信する場合もあります。 TP がどのようにこれらのインディケーターを処理するかについての詳細は、『会話の終了』を参照してください。

この例では、呼び出し先 TP は、データを受信するために最初の MC_RECEIVE_AND_WAIT verb を発行し、完全なデータ・レコード (*what_rcvd*= AP_DATA_COMPLETE) を受け取ります。

MC_RECEIVE_IMMEDIATE verb は MC_RECEIVE_AND_WAIT verb と同じ機能を実行しますが、データがパートナー TP から現在入手可能でない場合は待機しません。 その代わりに、呼び出し元 TP にデータ使用不可の応答を戻します。

会話の終了

会話を終了するには、どちらか一方の TP が MC_DEALLOCATE verb を発行し、これ以上送信するデータがないことを示します。もう一方の TP は、その後の受信 verb で、戻りコード AP_DEALLOC_NORMAL を受け取ります。これは会話が割り振り解除されたことを示します。

半二重HDでは、MC_DEALLOCATE verb により APPC が 2 つの TP 間での会話を割り振り解除します。そのため、もう一方の TP は、AP_DEALLOC_NORMAL 戻りコードを受信後には会話を続行できません。ただし、全二重会話では、もう一方の TP にまだ送信するデータがあるか、または既にデータを送信しているが、それを最初の TP がまだ受信していない可能性があります。このため、この時点では会話は終了しません。代わりに最初の TP は受信専用モードで作動し、引き続き受信 verb を発行します (これ以上データを送信することはできません)。

2 番目の TP は、AP_DEALLOC_NORMAL 戻りコードを受信すると、送信専用モードで動作します。それ以上受信 verb を発行することはできません (受信するデータがないため) が、引き続きデータを送信することはできます。この例では、呼び出し先 TP は、会話を割り振り解除する前にもう 1 回 MC_SEND_DATA verb を発行します。

両方の TP がこの会話の割り振りを解除したあと、それぞれの TP は別の [MC_]ALLOCATE verb か RECEIVE_ALLOCATE verb を発行して (このパートナー TP または別のパートナー TP と) 別の会話を開始するか、TP_ENDED verb を発行します。

1 つの TP が同時に複数の会話に参加することもできます。この場合、TP はすべての会話の割り振りが解除されたときに TP_ENDED verb を発行します。

会話状態

TP は、半二重会話と同じ方法で、全二重会話を特定の会話状態としてみなします。ただし、全二重会話で考えられる会話状態は、以下のように半二重会話の状態とは異なります。それぞれの状態で発行できる APPC verb についての詳細は、 319 ページの『付録 C. APPC の状態の変更』を参照してください。

可能な会話状態のリストを次に示します。

Send-Receive (送信 - 受信)
TP は、データまたはエラー情報を送信し、パートナー TP からアプリケーション・データと状況情報を受信できます。

Receive-Only (受信専用)
ローカル TP は会話を割り振り解除します。パートナー TP からのデータと状況情報の受信は継続しますが、さらにデータを送信することはできません。

Send-Only (送信専用)
リモート TP は会話を割り振り解除します。ローカル TP はパートナー TP へのデータの送信を継続できますが、これ以上のデータを受信することはなく、またこれ以上の受信 verb を発行することができません。

Reset (リセット)
会話は、まだ開始されていないか、すでに終了しました。

半二重 Verb は全二重会話で使用できない

次の verb は、半二重会話にのみ適用され、全二重会話では不要です。これらの verb が全二重会話で発行されると、いずれもエラー戻りコードを受け取ります。

- [MC_]CONFIRM
- [MC_]CONFIRMED
- [MC_]PREPARE_TO_RECEIVE
- [MC_]RECEIVE_AND_POST
- [MC_]REQUEST_TO_SEND
- [MC_]TEST_RTS
- [MC_]TEST_RTS_AND_POST

優先データを送信および受信する

[MC_]SEND_DATA を使用して送信され、受信 verb を使用して受信される通常のデータの他に、 APPC TP は、SNA 優先データの送受信も行います。このデータは、SNA ネットワークにより別に処理され、通常のデータの前に宛先に到着する可能性があります。 TP は、[MC_]SEND_EXPEDITED_DATA verb を使用して優先データを送信し、 [MC_]RECEIVE_EXPEDITED_DATA verb を使用してこのデータを受信します。優先データが、標準受信 verb で戻ることはありません。

すべてのインプリメンテーション形態の APPC が優先データをサポートしているわけではありません。リモート TP が優先データをサポートしない APPC インプリメンテーションを使用していると、ローカル TP は優先データの送受信ができません。

優先データは通常データとは別の流れで処理されるため、TP は [MC_]SEND_EXPEDITED_DATA または [MC_]RECEIVE_EXPEDITED_DATA を Reset 状態以外のどの会話でも発行できます。これらの verb は、状態の変更を起こしません。

同期および非同期の APPC コール

AIX, LINUX

AIX / Linux システムの場合、Communications Server では APPC ライブラリーへのエントリー・ポイントとして次の 2 つの選択肢があります。

- 同期エントリー・ポイント、APPC。アプリケーションがこのエントリー・ポイントを使用した場合、Communications Server は、 verb 処理が終了するまでアプリケーションに制御を戻しません。
- 非同期エントリー・ポイント、APPC_Async。アプリケーションがこのエントリー・ポイントを使用した場合、Communications Server は、即時にアプリケーションに制御を戻します。 verb 処理が終了した場合、Communications Server はアプリケーションが提供するコールバック・ルーチンを使用して verb 処理の結果をアプリケーションに戻します。

WINDOWS

Windows システムの場合、Remote API では APPC ライブラリーへのエントリー・ポイントとして次の 3 つの選択肢があります。

- 同期エントリー・ポイント、APPC。アプリケーションがこのエントリー・ポイントを使用した場合、Remote API は、 verb 処理が終了するまでアプリケーションに制御を戻しません。
- 非同期エントリー・ポイント、WinAsyncAPPC。アプリケーションがこのエントリー・ポイントを使用した場合、Remote API は、即時にアプリケーションに制御を戻します。 verb 処理が完了すると、Remote API はアプリケーションのウィンドウ・プロシージャーへメッセージを通知することで完了を示します。

- 非同期エントリー・ポイント、WinAsyncAPPCEx。verb 処理が完了すると、Remote API はアプリケーションで提供するイベント・ハンドルをシグナル通知することで完了を示します。

これらのエントリー・ポイントについての詳細は、29 ページの『第 2 章 トランザクション・プログラムの作成』を参照してください。

非同期エントリー・ポイントを使用すると、アプリケーションは verb の完了を待つ間、他の処理を続行できます。 アプリケーションは、他の APPC の会話に verb を発行するか、新しい会話を開始するために verb を発行するか、APPC に関連していないその他の処理を行うことができます。 ただし、同じ会話で発行された他の verb はキューに入れられ、未解決の verb が完了するまで処理されない可能性があります。詳細については、下記の 23 ページの『非ブロッキング操作』を参照してください。

ただ 1 つの例外は、[MC_]RECEIVE_AND_POST verb が未解決の場合です。 この場合、アプリケーションは同じ会話に対して限られた範囲の verb を発行できます。詳細については、次の項を参照してください。

データの非同期受信

APPC verb の MC_RECEIVE_AND_POST および RECEIVE_AND_POST を使用すると、TP はプログラム内で発生する他のイベントを考慮せずに、データを非同期に受信することができます。 したがって、プログラムはデータを受信しながら別のタスクを実行できます。

AIX, LINUX

AIX / Linux システムでは、[MC_]RECEIVE_AND_POST のパラメーターにコールバック・ルーチンのアドレスが含まれます。このアドレスは、データの受信を TP に通知するために APPC が使用するものです。このコールバック・ルーチンは、非同期 APPC エントリー・ポイントで指定されるコールバック・ルーチンとは別のものです。 [MC_]RECEIVE_AND_POST は、同期と非同期のどちらのエントリー・ポイントを使用しても発行できます。APPC は verb のパラメーターで指定されたコールバック・ルーチンを使用して、受信したデータを TP に戻し、非同期エントリー・ポイントへ提供されたコールバック・ルーチンを使用するのは、 VCB でヌル・アドレスが設定されている場合のみです。

WINDOWS

Windows システムでは、verb はアプリケーションで提供されたイベント・ハンドルをシグナル通知して完了します。*sema* パラメーターには、イベント・ハンドル

(Windows の CreateEvent または OpenEvent 関数のいずれかを呼び出して取得します) が含まれます。これは、データの受信を TP に通知するために APPC が使用するものです。

[MC_]RECEIVE_AND_POST の動作は、非同期エントリー・ポイントを使用して発行された [MC_]RECEIVE_AND_WAIT の動作によく似ており、制御は即時にアプリケーションに戻され、要求されたデータは、その後にコールバック・ルーチンで戻されます。 主な相違点は、[MC_]RECEIVE_AND_POST を発行すると会話が定義済みの状態である Pending-Post 状態に置かれ、TP はコールバック・ルーチンが呼び出されるのを待つ間、その会話に対して限られた範囲の APPC verb を発行できることです。 Pending-Post 状態で発行できる verb は、次のとおりです。

- GET_TYPE
- CANCEL_CONVERSATION
- 割り振り解除タイプが AP_ABEND、AP_ABEND_PROG、AP_ABEND_SVC、または AP_ABEND_TIMER である [MC_]DEALLOCATE
- [MC_]GET_ATTRIBUTES
- [MC_]RECEIVE_EXPEDITED_DATA
- [MC_]REQUEST_TO_SEND
- [MC_]SEND_ERROR
- [MC_]SEND_EXPEDITED_DATA
- [MC_]TEST_RTS
- TP_ENDED

アプリケーションで非同期エントリー・ポイントを使用して [MC_]RECEIVE_AND_WAIT (またはその他の APPC verb) を発行した場合、コールバック・ルーチンが呼び出されるまで、その会話に対して他の APPC verb を発行してはなりません。唯一の例外として、CANCEL_CONVERSATION を発行し、未解決の受信操作を取り消して会話を終了させることができます。

表 7 では、呼び出し先 TP はデータを非同期に受信します。

表 7. データの非同期受信

状態	呼び出し元 TP	フロー	呼び出し先 TP	状態
	TP_STARTED			
Reset (リセット)				
	MC_ALLOCATE			
Send (送信)				
	MC_FLUSH			
		—>		
				Reset (リセット)
			RECEIVE_ALLOCATE	

表 7. データの非同期受信 (続き)

状態	呼び出し元 TP	フロー	呼び出し先 TP	状態
				Receive (受信)
			MC_RECEIVE_AND_POST (*primary_rc*=AP_OK)	
				Pending-Post (保留 - 通知)
			TP は他のタスクを実行するか、Request to Send (送信要求) または Get Attributes (属性の取得) などの verb を発行します。 その他のほとんどの APPC verb は、この会話状態では使用できません (許可される verb については、 319 ページの『付録 C. APPC の状態の変更』を参照)。	
	MC_SEND_DATA MC_DEALLOCATE			
Reset (リセット)				
	TP_ENDED			
		—>		
			データが受信されます。 APPC は、提供されたコールバック・ルーチンを呼び出します。 MC_RECEIVE_AND_POST は *primary_rc*=AP_OK、*what_rcvd*=AP_DATA_COMPLETE を戻します	
				Receive (受信)
			(TP は、コールバック・ルーチンが呼び出されたかどうかを検査します。) MC_RECEIVE_AND_WAIT (*primary_rc*=AP_DEALLOC_NORMAL)	
				Reset (リセット)
			TP_ENDED	

21 ページの表 7 で、呼び出し先 TP は次の手順を実行してデータを非同期に受信します。

1. MC_RECEIVE_AND_POST verb を発行します。パラメーターのうちのいずれかが、(AIX / Linux で) APPC が呼び出すコールバック・ルーチンのアドレスであるか、または (Windows で) APPC がデータの受信をシグナル通知するイベント・ハンドルです。
2. *primary_rc* (1 次戻りコード) が、データの非同期受信を TP が開始したことを示す AP_OK であるかどうかを確認します。

3. データを非同期に受信している間、この会話に関連しないタスクを実行します。ほとんどの APPC verb は、この会話状態では無効です。
4. コールバック・ルーチン (AIX / Linux で) が呼び出されるか、イベント・ハンドル (Windows で) が信号で知らせるのを待機します。これは、TP がデータの非同期受信を完了したことを示します。
5. MC_RECEIVE_AND_POST verb の *primary_rc* を再度検査します。 2 番目の *primary_rc* は、データがエラーなしに受信されたかどうかを示します。
6. MC_RECEIVE_AND_POST verb の *what_rcvd* パラメーターが AP_DATA_COMPLETE であるかどうかを検証します。
7. 割り振り解除インディケーターを受け取るために MC_RECEIVE_AND_WAIT verb を発行します。

注: [MC_]RECEIVE_AND_POST verb は *primary_rc* パラメーターと *secondary_rc* パラメーターを 2 回戻します。 1 回目は、この verb を発行した後に verb が正常にデータ待ちを開始したかどうかを示すため、2 回目はデータを受信した後に戻ります。

呼び出し先 TP が MC_RECEIVE_AND_POST verb を発行して AP_OK の *primary_rc* を取得した後、会話は Pending-Post 状態に変更されます。

TP がデータの非同期受信を完了し、APPC が提供されたコールバック・ルーチン (AIX / Linux で) を呼び出すか、または提供されたイベント・ハンドル (Windows で) に信号を送ると、*what_rcvd* パラメーターに AP_DATA_COMPLETE が入るため、会話が Receive 状態に変わります。

非ブロッキング操作

Communications Server は、APPC 会話 verb に対し、キュー・レベルの非ブロッキング操作をサポートします。そのため、それぞれの verb が完了するのを待機しなくても、TP は同じ会話で複数の verb を発行できます。これは、通常 APPC エントリー・ポイントと併用され、前の verb の処理がまだ完了していなくても、TP は操作を続行することができます。ただし、これは複数のスレッドから APPC verb を発行するマルチスレッド TP の同期エントリー・ポイントでも使用されます。 非ブロッキング・モードで verb を発行するには、 105 ページの『第 4 章 APPC 会話 verb』で説明しているように、TP が verb の *opext* パラメーターにオプションを設定します。

TP および会話では、APPC は、処理を待機している間に、verb が保留できるキューの数を提供します。キュー・ハンドルはそれぞれ有効な APPC verb の別のサブセットを処理し、verb はそれぞれ別のキューに関連付けられます。

- TP が verb を発行し、該当するキューですでに処理されている verb がない場合、その verb がすぐに処理されます。
- TP が非ブロッキング verb を発行し、他の verb がすでに該当するキューで処理されている場合、その verb はキューに (すでに該当するキューで待機している他の verb の後に) 入れられます。これは、すでにキューに入っている verb が完了してから処理されます (以下で説明する「キューの割り当て」は除きます)。
- TP がブロッキング verb を発行し、他の verb が該当するキューですでに処理されていると、その verb はエラー戻りコードで拒否されます。

TP で有効なキューと、各キューが処理する verb は以下のとおりです。

キューの割り当て

アクティブな TP ごとに、次の verb を処理するキューが 1 個用意されています。

- [MC_]ALLOCATE
- [MC_]SEND_CONVERSATION

このキューにある複数の verb は同時に処理されます。これらの verb が発行された順に完了する保証はありません。

送信 - 受信キュー (半二重会話専用)

アクティブな半二重会話ごとに、次の verb を処理するキューが 1 個用意されています。

- [MC_]CONFIRM
- [MC_]CONFIRMED
- [MC_]DEALLOCATE
- [MC_]FLUSH
- [MC_]PREPARE_TO_RECEIVE
- [MC_]RECEIVE_AND_WAIT
- [MC_]RECEIVE_IMMEDIATE
- [MC_]SEND_DATA
- [MC_]SEND_ERROR

[MC_]RECEIVE_AND_POST verb は、このキューでは保留となりません。受信 verb のいずれかが、会話ですでに処理されている場合、この verb はブロッキングまたは非ブロッキング・モードでは発行されません。

送信キュー (全二重会話専用)

アクティブな全二重会話ごとに、次の verb を処理するキューが 1 個用意されています。

- [MC_]DEALLOCATE
- [MC_]FLUSH
- [MC_]SEND_DATA
- [MC_]SEND_ERROR

受信キュー (全二重会話専用)

アクティブな全二重会話ごとに、次の verb を処理するキューが 1 個用意されています。

- [MC_]RECEIVE_AND_WAIT
- [MC_]RECEIVE_IMMEDIATE

送信優先キュー

アクティブな会話 (全二重または半二重) ごとに、次の verb を処理するキューが 1 個用意されています。

- [MC_]REQUEST_TO_SEND (半二重会話専用)
- [MC_]SEND_EXPEDITED_DATA

受信優先キュー

アクティブな会話 (全二重または半二重) ごとに、次の verb を処理するキューが 1 個用意されています。

- [MC_]RECEIVE_EXPEDITED_DATA

CANCEL_CONVERSATION verb は、どのキューでも保留となりません。この verb を発行すると、この会話のためにすでにキューに入れられている他の verb が取り消されます。

次の会話 verb はどのキューにも関連しないため、すでにキューに入っている verb に関係なくいつでも発行することができます。 verb の *opext* パラメーターにある非ブロッキング・モードのオプションは無視されます。

- GET_TYPE
- [MC_]GET_ATTRIBUTES
- [MC_]TEST_RTS
- [MC_]TEST_RTS_AND_POST

APPC 制御 verbs は常にブロッキング・モードで発行されます。

同期点サポート

AIX, LINUX

Communications Server APPC API は、LU6.2 同期点プロトコルを使用するセッションと会話をサポートします。 したがって、この API は、同期点 Level 2 のサポートを必要とするトランザクション・ モニターと共に使用できます。この API 自体は同期点サポートを完全に実現するために必要なコンポーネントを提供するわけではありませんが、ベンダーが提供する同期点機能を支える基本サポートを実現します。ベンダーは、次のコンポーネントを提供する必要があります。

- 同期点管理プログラム (Syncpoint Manager: SPM)
- Conversation-Protected Resource Manager (C-PRM)
- 再同期 TP

本書の目的は同期点の機能を解説することではなく、それらの機能をインプリメントするために Communications Server が提供するサポートについてのみを説明することです。Communications Server APPC と共に使用するために同期点管理プログラムを開発する場合は、同期点の概念をよく理解しておかなければなりません。必要であれば、詳細について IBM LU6.2 の資料を参照してください。

本書で示すパラメーター値と戻りコード値の一部には、「同期点処理をサポートする TP のみが使用」または「会話の *sync_level* が AP_SYNCPT の場合にのみ使用」というただし書きが付いています。 同期点機能を使用しない APPC アプリケーションを作成する場合は、これらのパラメーターを使用しないでください。ほとんどの場合、パラメーターの説明にあるように、これらの値と適切な同期点機能の間の変換は、同期点管理プログラムが行います。

Communications Server のサプライヤーまたは別のベンダーが提供する同期点製品を使用するアプリケーションを作成する場合、その製品を使用するために必要な追加情報はベンダーから提供されます。

APPC と CPI-C

もう 1 つの Communications Server API である共通プログラミング・インターフェース・コミュニケーション (CPI-C) アプリケーション・プログラミング・インターフェースは、多くの APPC 機能を提供しますが、インターフェースの形態は異なっています。

APPC アプリケーションで verb 制御ブロック内にパラメーターを設定し、そのブロックのアドレスを指定して APPC への単一のエントリー・ポイントを呼び出すと、CPI-C プログラムは verb ごとに異なるエントリー・ポイントを呼び出し、必要な情報をコールのパラメーターとして渡します。

APPC と CPI-C のプログラミング・インターフェースは異なりますが、実際にプログラム間で伝送されるデータは同じです。 したがって、CPI-C アプリケーションは、2 つの APPC TP または 2 つの CPI-C アプリケーションが相互に通信できるのとまったく同様に、APPC TP と通信できます。 APPC TP では、APPC TP のパートナーが APPC TP アプリケーションか CPI-C アプリケーションかを認識している必要はありません。

CPI-C アプリケーションと通信する APPC TP では、 CPI-C が PIP データの受信をサポートしないため、会話の割り振り時にプログラム初期化パラメーター (PIP データ) を送信してはならないという制限があるだけです。 PIP データについての詳細は、 105 ページの『第 4 章 APPC 会話 verb』で [MC]ALLOCATE verb の説明を参照してください。

TP サーバー API

AIX, LINUX

TP サーバーの verb は、アプリケーションが割り振り要求 (Attach) への応答として TP の開始に参加できるよう、APPC API を拡張したものです。

Communications Server は、TP を自動的に開始するためのデフォルトの機能を提供します。自動的に開始される TP は、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」で説明されているように、**sna_tps** ファイルに構成されます。

トランザクション・モニターなど、一部のアプリケーションでは、TP の開始について、このデフォルトの機能で提供される以上の制御 (割り振り要求へのアクセスなど) が必要です。 TP サーバーの拡張機能により、そのようなアプリケーションに必要なレベルがサポートされています。 TP サーバーについての詳細は、 69 ページの『TP サーバーの作成』を参照してください。

第 2 章 トランザクション・プログラムの作成

この章では、次のトピックに関する情報が記載されており、トランザクション・プログラム (TP) の作成に役立ちます。

- APPC verb のカテゴリー
- APPC verb の要約
- APPC エントリー・ポイント

AIX, LINUX

- AIX / Linux の考慮事項

WINDOWS

- Windows の考慮事項

- 構成情報
- 会話セキュリティー
- TP の開始
- LU 間セッション
- 基本会話

AIX, LINUX

- TP サーバーの作成

- 移植可能な TP の作成

APPC verb のカテゴリー

APPC verb は、次のカテゴリーに分類できます。

- 制御 verb。これについては 73 ページの『第 3 章 APPC 制御 verb』で説明します。
- 会話 verb。これについては、 105 ページの『第 4 章 APPC 会話 verb』で説明します。

制御 verb

制御 verb は、TP の開始と終了、TP の特性に関する情報の取得を行います。

- TP_STARTED
- TP_ENDED
- RECEIVE_ALLOCATE
- GET_TP_PROPERTIES

AIX, LINUX

- SET_TP_PROPERTIES
- GET_LU_STATUS

会話 verb

会話 verb を使用すると、TP から会話の割り振り、データの送受信、会話状態の変更、会話の割り振り解除を行うことができます。

次の verb は、基本会話またはマップ式会話のいずれかで発行されます。

- GET_TYPE
- CANCEL_CONVERSATION

ほとんどの会話 verb は、次の 2 つのグループに分類できます。

- マップ式会話 verb。これはマップ式会話でのみ TP から発行できます。
- 基本会話 verb。これは基本会話でのみ TP から発行できます。

会話 verb は、表 8 に示すように、マップ式会話か基本会話かのタイプによって分類されます。

表 8. マップ式会話および基本会話の verb

マップ式会話 verb	基本会話 verb
MC_ALLOCATE	ALLOCATE
MC_CONFIRM	CONFIRM
MC_CONFIRMED	CONFIRMED
MC_DEALLOCATE	DEALLOCATE
MC_FLUSH	FLUSH
MC_GET_ATTRIBUTES	GET_ATTRIBUTES
MC_PREPARE_TO_RECEIVE	PREPARE_TO_RECEIVE
MC_RECEIVE_AND_POST	RECEIVE_AND_POST
MC_RECEIVE_AND_WAIT	RECEIVE_AND_WAIT
MC_RECEIVE_IMMEDIATE	RECEIVE_IMMEDIATE
MC_RECEIVE_EXPEDITED_DATA	RECEIVE_EXPEDITED_DATA
MC_REQUEST_TO_SEND	REQUEST_TO_SEND
MC_SEND_CONVERSATION	SEND_CONVERSATION
MC_SEND_DATA	SEND_DATA
MC_SEND_ERROR	SEND_ERROR

表 8. マップ式会話および基本会話の verb (続き)

マップ式会話 verb	基本会話 verb
MC_SEND_EXPEDITED_DATA	SEND_EXPEDITED_DATA
MC_TEST_RTS	TEST_RTS
MC_TEST_RTS_AND_POST	TEST_RTS_AND_POST

マップ式 verb および基本 verb は、それぞれの会話タイプで同じ機能を備えていますが、パラメーターと戻りコードが多少異なる場合があります。

TP サーバー verb

AIX, LINUX

TP サーバー verb を使用すると、アプリケーションで Communications Server からの次のような要求に対する応答として、TP を開始することができます。

REGISTER_TP_SERVER
UNREGISTER_TP_SERVER
REGISTER_TP
UNREGISTER_TP
QUERY_ATTACH
ACCEPT_ATTACH
REJECT_ATTACH
ABORT_ATTACH

注: TP サーバー verb は、277 ページの『第 5 章 TP サーバー verb』で説明されているように、同期エントリー・ポイント `APPC` ではなく、非同期エントリー・ポイント `APPC_Async` を使用して発行される必要があります。

APPC verb の要約

この項では、個々の APPC verb について簡単に説明します。 APPC verb は、機能別に分類されています。 (verb の全体的な機能は、この要約に示すより広範な場合があります。各 verb についての詳細は、73 ページの『第 3 章 APPC 制御 verb』、または 105 ページの『第 4 章 APPC 会話 verb』を参照してください。)

会話の開始

次の各 verb は、2 つの TP 間で会話を開始するために使用します。詳細については、63 ページの『TP の開始』を参照してください。

TP_STARTED

この verb は、呼び出し元 TP が発行します。この verb は、呼び出し元 TP が開始されようとしていることを APPC に通知します。正常に実行されると、この verb は呼び出し元 TP の TP ID (*tp_id*) を戻します。

AIX, LINUX

SET_TP_PROPERTIES
この verb は、TP がローカル TP に関連した特性を設定するために使用し、その後、それらの特性は新規の会話を割り振るときに使用されます。この verb を使用すると、TP から次のものを指定できます。

- 会話へ関連付ける作業論理単位 (特定のタスクを実行するための APPC TP 間のトランザクション)
- 新規の会話を割り振り、会話セキュリティーがすでに検証済みであることを示すときに使用されるユーザー ID

MC_ALLOCATE または ALLOCATE
この verb は、呼び出し元 TP が発行します。この verb はローカル LU とリモート LU 間のセッションを割り振り、呼び出し元 TP と呼び出し先 TP 間の会話を確立します。

ALLOCATE verb は、基本会話とマップ式会話のどちらでも確立できます。MC_ALLOCATE verb は、マップ式会話のみを開始できます。 いずれの verb も、全二重または半二重会話を開始できます。

会話が割り振られたあと、APPC はこの verb を通じて会話 ID (*conv_id*) を戻します。

MC_SEND_CONVERSATION または SEND_CONVERSATION
この verb は、呼び出し元 TP が発行します。この verb はローカル LU とリモート LU 間のセッションを割り振り、呼び出し元 TP と呼び出し先 TP 間の会話を確立します。また、その会話でデータ・レコードを 1 つだけ送信し、会話の割り振りを解除します。

RECEIVE_ALLOCATE
この verb は、呼び出し先 TP が発行します。この verb は、呼び出し先 TP が、[MC_]ALLOCATE verb を発行した呼び出し元 TP との会話を開始できる状態であることを確認します。 正常に実行されると、RECEIVE_ALLOCATE は呼び出し先 TP の TP ID (*tp_id*) と会話 ID (*conv_id*) を戻します。

データの送信

次の各 verb は、パートナー TP へデータを送信します。

MC_SEND_DATA または SEND_DATA
この verb は、パートナー TP へ伝送するデータをローカル LU の送信バッファーに入れます。

ローカル LU の送信バッファー内に収集されたデータは、次のいずれかが発生した時点でパートナー LU (およびパートナー TP) へ伝送されます。

- 送信バッファーがいっぱいになった。

- ローカル TP が、[MC_]FLUSH や [MC_]CONFIRM などの LU の送信バッファーをフラッシュする verb を発行する。([MC_]CONFIRM は半二重会話にのみ適用されます。)

[MC_]SEND_DATA verb は、[MC_]CONFIRM、[MC_]DEALLOCATE、[MC_]FLUSH、[MC_]PREPARE_TO_RECEIVE のいずれかの verb の機能も実行できます。 ([MC_]CONFIRM と [MC_]PREPARE_TO_RECEIVE は、半二重会話にのみ適用されます。)

MC_SEND_EXPEDITED_DATA または SEND_EXPEDITED_DATA
この verb は、パートナー TP へ伝送するデータをローカル LU の優先送信バッファーに入れます。

ローカル LU の送信バッファー内に収集されたデータは、[MC_]SEND_DATA verb と同じ方法で、パートナー LU (およびパートナー TP) へ伝送されます。ただし、データは優先データとしてネットワークをまたいで送信されるため、 [MC_]SEND_DATA を使用して先に送信されたデータの前に到着する可能性があります。

MC_FLUSH または FLUSH
この verb はローカル LU の送信バッファーをフラッシュし、バッファーの内容をパートナー LU (および TP) へ送信します。 送信バッファーが空の場合、アクションは発生しません。

MC_CONFIRM または CONFIRM (半二重会話専用)
この verb は、ローカル LU の送信バッファーの内容と確認要求の両方をパートナー TP へ送信します。

MC_PREPARE_TO_RECEIVE または PREPARE_TO_RECEIVE (半二重会話専用)
この verb は、会話の状態を Send から Receive に変更します。会話状態を変更する前に、この verb は [MC_]FLUSH verb または [MC_]CONFIRM verb に相当する動作を実行します。この verb が正常に実行された後、TP はデータを受信できます。

MC_REQUEST_TO_SEND または REQUEST_TO_SEND (半二重会話専用)
この verb は、ローカル TP からデータを送信することをパートナー TP に通知します。パートナー TP が [MC_]PREPARE_TO_RECEIVE verb または [MC_]RECEIVE_AND_WAIT verb を発行すると、パートナー TP の会話状態が Receive に変更され、ローカル TP はデータを送信できるようになります。

データの受信

次の verb は、TP がパートナー TP からデータを受信できるようにします。

MC_RECEIVE_AND_WAIT または RECEIVE_AND_WAIT
会話が Receive 状態のときにこの verb を発行すると、ローカル TP はパートナー TP から現在入手可能なデータを受信します。入手可能なデータがない場合、ローカル TP はデータが到着するまで待ちます。

会話が Send 状態にあるときにこの verb を発行すると、 LU の送信バッファーがフラッシュされ、会話状態が Receive に変更されます。 その後、ローカル TP はデータの受信を開始します。

MC_RECEIVE_AND_POST または RECEIVE_AND_POST
会話が Receive 状態のときにこの verb を発行すると、会話状態は Pending_Post に変更され、ローカル TP はデータを非同期に受信するようになります。これにより、ローカル TP はローカル LU がまだデータを受信中に処理を進めることができます。

会話が Send 状態にあるときにこの verb を発行すると、 LU の送信バッファーがフラッシュされ、会話状態が Pending_Post に変更されます。 その後、ローカル TP はデータの非同期受信を開始します。

MC_RECEIVE_IMMEDIATE または RECEIVE_IMMEDIATE
この verb は、パートナー TP から現在入手可能なデータを受信します。入手可能なデータがない場合、ローカル TP は待ちません。他の受信 verb とは異なり、この verb は Receive 状態でのみ発行されます。Send 状態では発行されません。

MC_RECEIVE_EXPEDITED_DATA または RECEIVE_EXPEDITED_DATA
この verb は、パートナー TP から現在入手可能な優先データを受信します。入手可能なデータがない場合、verb は即時に戻るか、またはデータが到着するまで待機することができます。

データの受信の確認またはエラーの報告

次の verb は、データの受信を確認するか、エラーを報告します。

MC_CONFIRMED または CONFIRMED
この verb は、パートナー TP からの確認要求に応答します。この verb は、ローカル TP がエラーなしにデータを受信し、処理したことをパートナー TP に通知します。

MC_SEND_ERROR または SEND_ERROR
この verb は、ローカル TP がアプリケーション・レベルのエラーを検出したことをパートナー TP に通知します。

情報の取得

次の verb は、TP に情報を提供します。

MC_GET_ATTRIBUTES または GET_ATTRIBUTES
この verb は、TP で会話の属性を取得するために使用します。

GET_TYPE
この verb は TP によって使用され、特定の会話の会話タイプ (基本またはマップ式) と、会話が全二重または半二重モードで作動するかどうかを判別します。この情報により、TP がこの会話で発行する正しい verb を判別することができます。

AIX, LINUX

GET_LU_STATUS
この verb は、ローカル LU と指定したパートナー LU 間のセッションの数についての情報や、前回この verb を発行してからその時点までにセッションの数が 0 (ゼロ) に落ちたかどうかについて情報を取得するために TP

が使用します。 これにより、TP でパートナー TP との接続を失ったかどうか (失った場合は再同期が必要になることがあります) を検査できます。

GET_TP_PROPERTIES
この verb は、ローカル TP の属性と、その TP が参加している作業論理単位 (特定のタスクを実行するための APPC TP 間のトランザクション) の属性に関する情報を取得するために TP が使用します。

MC_TEST_RTS または TEST_RTS
この verb は、パートナー TP から REQUEST_TO_SEND 通知を受信したかどうかを判別します。

MC_TEST_RTS_AND_POST または TEST_RTS_AND_POST
この verb は、パートナー TP から REQUEST_TO_SEND 通知を受信したことをアプリケーションに非同期に通知します。

会話の終了

会話は、呼び出し先 TP と呼び出し元 TP のどちらからでも終了できます。次の各 verb は、会話を終了します。

MC_DEALLOCATE または DEALLOCATE
この verb は、2 つの TP 間の会話の割り振りを解除します。会話の割り振りを解除する前に、この verb は [MC_]FLUSH verb または [MC_]CONFIRM verb に相当する動作を実行します。

CANCEL_CONVERSATION
この verb は、2 つの TP 間の会話の割り振りを解除します。これは、同じ会話上の他の verb が未解決になっている間に発行できることを除き、MC_DEALLOCATE または DEALLOCATE の *dealloc_type* を AP_ABEND または任意の AP_ABEND_* 値に設定することと同等です。(これらの未解決の verb の結果は未定義となり、アプリケーションには返されません。たとえば、CANCEL_CONVERSATION を、[MC_]SEND_DATA が未解決の間に発行する場合は、パートナー TP にデータが送信されているかどうかを判別できません。)

TP_ENDED
この verb は、呼び出し元 TP と呼び出し先 TP の両方が発行します。この verb は、TP が終了しようとしていることを APPC に通知します。 この verb を発行すると、活動状態になっている可能性があるその他の会話も終了します。

トランザクション・プログラム (TP) の開始

AIX, LINUX

次の各 verb は、アプリケーションを Communications Server の TP ロード・プロセスに参加できるようにするために使用します。

REGISTER_TP_SERVER
この verb は、アプリケーションがトランザクション・プログラム (TP) を自動的に開始できることを Communications Server に通知するために TP が使用します。

REGISTER_TP
この verb は、アプリケーションで処理する TP 開始要求 (Attach) の発行元である TP の名前を Communications Server に登録するために使用します。

QUERY_ATTACH
この verb は、TP を開始するかどうかを判別できるよう、アプリケーションで TP 開始要求のパラメーターを判別するために使用します。

ACCEPT_ATTACH
この verb は、その Attach に対応する TP をアプリケーションが開始しようとしていることを Communications Server に通知するために使用します。

REJECT_ATTACH
この verb は、その Attach に対応する TP をアプリケーションが開始しようとしていないことを Communications Server に通知するために使用します。

ABORT_ATTACH
この verb は、ACCEPT_ATTACH verb を使用して Attach を受け入れたあと、TP サーバーまたは TP がその後の処理中にエラーを検出したために、この TP サーバーによる Attach の処理を終了するために使用します。

UNREGISTER_TP
この verb は、前に登録した TP 用の Attach をアプリケーションがそれ以上処理しないことを Communications Server に通知するために使用します。

UNREGISTER_TP_SERVER
この verb は、指定した TP 用の Attach 通知の受信をアプリケーションが要求していないことを Communications Server に通知するために使用します。

APPC エントリー・ポイント: AIX または Linux システム

AIX, LINUX

アプリケーションは、次のエントリー・ポイントを使用して APPC にアクセスします。

APPC APPC verb を同期的に発行します。 Communications Server は、verb 処理が終了するまでアプリケーションに制御を戻しません。

APPC_Async
APPC verb を非同期的に発行します。 Communications Server は即時にアプリケーションに制御を戻し、verb 処理がまだ進行中であるか、すでに完

了したかを戻り値で示します。ほとんどの場合、制御がアプリケーションに戻った時点では、まだ verb 処理が進行中です。その後、Communications Server はアプリケーションが提供するコールバック・ルーチンを使用して verb 処理の結果を戻します。場合によっては、Communications Server がアプリケーションに制御を戻した時点で verb 処理が完了しており、Communications Server がアプリケーションのコールバック・ルーチンを使用しないこともあります。

注: TP サーバー verb は、277 ページの『第 5 章 TP サーバー verb』で説明されているように、同期エントリー・ポイント APPC ではなく、非同期エントリー・ポイント APPC_Async を使用して発行される必要があります。

エントリー・ポイント APPC と APPC_Async は、 APPC ヘッダー・ファイル **/usr/include/sna/appc_c.h** (AIX の場合) または **/opt/ibm/sna/include/appc_c.h** (Linux の場合) に定義されています。

1 つのタスクを実行し、Communications Server またはリモート・システムからの情報を待つ間に中断できるアプリケーションは、 APPC (同期) エントリー・ポイントのみを使用する必要があります。

アプリケーションが複数のタスクを実行するとき (たとえば、一度に複数のリモート・プログラムと通信したり、APPC verb のほかに別の処理を実行する場合など) は、情報を待つ間、中断を行わないようにしなければならない場合があります。その場合、アプリケーションは APPC_Async (非同期) エントリー・ポイントを使用し、Communications Server が入手可能な情報を戻すときに使用できるコールバック・ルーチンを提供する必要があります。

次の各項では、それらのエントリー・ポイントについて説明し、アプリケーションからエントリー・ポイントに提供する必要がある追加アプリケーション定義関数の一部についても説明します。

APPC エントリー・ポイント

アプリケーションは、APPC を使用して APPC verb を同期的に発行します。Communications Server は、verb 処理が終了するまでアプリケーションに制御を戻しません。

関数コール

```
void APPC       (
                 void * vcb
                );
```

以前にインプリメントされた APPC との互換性を保つため、Communications Server はエントリー・ポイント APPC_C および APPC_P も提供します。これらのエントリー・ポイントは、APPC と同様に使用できます。

指定パラメーター

アプリケーションで APPC エントリー・ポイントを使用して verb を発行する場合は、次のパラメーターをアプリケーションから提供します。

vcb 発行する verb のパラメーターが入っている verb 制御ブロック (VCB) を指すポインターです。個々の verb の VCB 構造体については、

73 ページの『第 3 章 APPC 制御 verb』と 105 ページの『第 4 章 APPC 会話 verb』で説明しています。 これらの構造体は、APPC ヘッダー・ファイル **/usr/include/sna/appc_c.h** (AIX の場合) または **/opt/ibm/sna/include/appc_c.h** (Linux の場合) に定義されています。

注: APPC VCB には、「予約済み」とマークされたパラメーターが多数含まれています。予約済みパラメーターには、Communications Server ソフトウェアで内部的に使用されているものや、このバージョンでは使用されていなくても将来のバージョンで使用される可能性があるものがあります。アプリケーションでは、予約済みパラメーターにアクセスしてはなりません。その代わりに、verb で使用する他のパラメーターを設定する前に、VCB の全内容をゼロに設定し、これらのパラメーターすべてをゼロにしてください。このようにすると、内部的に使用されるパラメーターが Communications Server で誤って解釈されることがなく、将来のバージョンでこれらのパラメーターが新しい関数を提供するために使用されても、アプリケーションは引き続き動作します。

VCB の内容をゼロに設定するには、`memset` を使用します。

memset(vcb, 0, sizeof(vcb));

戻り値

この関数は値を戻しません。コールが戻った時点で、アプリケーションは VCB 内のパラメーターを調べて、verb が正常に終了したかどうかを判別できます。

APPC_Async エントリー・ポイント

アプリケーションは、`APPC_Async` を使用して APPC verb を非同期的に発行します。 Communications Server は即時にアプリケーションに制御を戻し、verb 処理がまだ進行中であるか、すでに完了したかを戻り値で示します。ほとんどの場合、制御がアプリケーションに戻った時点では、まだ verb 処理が進行中です。 その後、Communications Server はアプリケーションが提供するコールバック・ルーチンを使用して verb 処理の結果を戻します。 場合によっては、Communications Server がアプリケーションに制御を戻した時点で verb 処理が完了しており、Communications Server がアプリケーションのコールバック・ルーチンを使用しないこともあります。

関数コール

```
unsigned short APPC_Async  (
                            void *        vcb,
                            AP_CALLBACK   (*comp_proc),
                            AP_CORR       corr
                           );

typedef void (*AP_CALLBACK) (
                             void *          vcb,
                             unsigned char   tp_id[8],
                             AP_UINT32       conv_id,
                             AP_CORR corr
                            );
```

```
typedef union ap_corr  {
                        void *          corr_p;
                        AP_UINT32       corr_l;
                        AP_INT32        corr_i;
                       } AP_CORR;
```

これらのエントリー・ポイントと APPC VCB で使用される AP_UINT32 などのパラメーター・タイプは、共通ヘッダー・ファイル **/usr/include/sna/values_c.h** (AIX の場合) または **/opt/ibm/sna/include/values_c.h** (Linux の場合) に定義されています。これらは APPC ヘッダー・ファイル **/usr/include/sna/appc_c.h** (AIX の場合) または **/opt/ibm/sna/include/appc_c.h** (Linux の場合) によってインクルードされます。

指定パラメーター

アプリケーションで APPC_Async エントリー・ポイントを使用して verb を発行する場合は、次のパラメーターをアプリケーションから提供します。

vcb
発行する verb のパラメーターが入っている verb 制御ブロック (VCB) を指すポインターです。個々の verb の VCB 構造体については、 73 ページの『第 3 章 APPC 制御 verb』と 105 ページの『第 4 章 APPC 会話 verb』で説明しています。 これらの構造体は、APPC ヘッダー・ファイル **appc_c.h** の中で定義されています。

注: APPC VCB には、「予約済み」とマークされたパラメーターが多数含まれています。予約済みパラメーターには、Communications Server ソフトウェアで内部的に使用されているものや、このバージョンでは使用されていなくても将来のバージョンで使用される可能性があるものがあります。アプリケーションでは、予約済みパラメーターにアクセスしてはなりません。その代わりに、verb で使用する他のパラメーターを設定する前に、VCB の全内容をゼロに設定し、これらのパラメーターすべてをゼロにしてください。このようにすると、内部的に使用されるパラメーターが Communications Server で誤って解釈されることがなく、将来のバージョンでこれらのパラメーターが新しい関数を提供するために使用されても、アプリケーションは引き続き動作します。

VCB の内容をゼロに設定するには、memset を使用します。

memset(vcb, 0, sizeof(vcb));

comp_proc
verb の完了時に Communications Server が呼び出すコールバック・ルーチン。コールバック・ルーチンの要件についての詳細は、 40 ページの『非同期 verb の完了のためのコールバック・ルーチン』を参照してください。

corr
アプリケーションで使用するオプションの相関係数。このパラメーターは、アプリケーションで 3 つの異なるパラメーター・タイプ (ポインター、符号なし long、または整数) のいずれかを指定できるよう、C の共用体として定義されます。

Communications Server はこの値を使用しませんが、verb の完了時にこの値をパラメーターとしてコールバック・ルーチンに渡します。この値をアプリケーションで使用すると、戻された情報をアプリケーションの別の処理へ関連付けることができます。

戻り値

この非同期エントリー・ポイントは、次のいずれかの値を戻します。

AP_COMPLETED

verb は、すでに完了しました。アプリケーションから VCB 内のパラメーターを検査して、verb が正常に完了したかどうかを判別できます。Communications Server は、この verb 用に提供されたコールバック・ルーチンを呼び出しません。

AP_IN_PROGRESS

verb は、まだ完了していません。アプリケーションは他の処理を続行でき、現行の verb の完了に依存しない別の APPC verb を発行することもできます。ただし、この verb へ提供された VCB 内のパラメーターの検査または変更はできません。

Communications Server は、verb 処理がいつ完了するかを示すため、提供されたコールバック・ルーチンを呼び出します。その後、アプリケーションは VCB パラメーターを検査できます。

非同期エントリー・ポイントの使用

非同期エントリー・ポイントを使用する場合は、次のことに注意してください。

- アプリケーションで *comp_proc* パラメーターにヌル・ポインターを指定した場合、verb は同期的に完了します (アプリケーションで同期エントリー・ポイントを使用して verb を発行した場合と同じ)。
- APPC_Async への呼び出しをアプリケーション・コールバックの中から行う場合、*comp_proc* パラメーターにヌル・ポインターを指定することはできません。 そのような場合、Communications Server は 1 次戻りコード値 AP_PARAMETER_CHECK および 2 次戻りコード値 AP_SYNC_NOT_ALLOWED で verb をリジェクトします。
- コールバック・ルーチンが呼び出されるまで、アプリケーションで VCB 内のパラメーターの使用や変更を行ってはなりません。
- 複数の verb は、必ずしも発行した順序で完了しません。 特に、アプリケーションから非同期 verb を発行した後に同期 verb を発行した場合、同期 verb の完了は非同期 verb がすでに完了していることを保証するものではありません。
- [MC_]RECEIVE_AND_POST verb は、VCB パラメーターの 1 つとしてコールバック・ルーチンへのポインターを含んでいます。 この verb は、同期エントリー・ポイントと非同期エントリー・ポイントのどちらを使用しても発行できます。 Communications Server は、VCB 内で指定されたコールバック・ルーチンを使用してこの verb の結果を戻します。非同期エントリー・ポイントで指定したコールバック・ルーチンは、アプリケーションが VCB 内でコールバック・ルーチンにヌル・ポインターを指定した場合にのみ使用されます。

非同期 verb の完了のためのコールバック・ルーチン

非同期エントリー・ポイントを使用する場合、アプリケーションはコールバック・ルーチンへのポインターを提供する必要があります。 この項では、Communications Server がそのルーチンをどのように使用するか、および、そのルーチンが実行する必要がある機能について説明します。

関数コール

```
AP_CALLBACK (*comp_proc);
typedef void (*AP_CALLBACK)  (
                               void *          vcb,
                               unsigned char   tp_id[8],
                               AP_UINT32       conv_id,
                               AP_CORR         corr
                             );
typedef union ap_corr  {
                          void *               corr_p;
                          AP_UINT32            corr_l;
                          AP_INT32             corr_i;
                        } AP_CORR;
```

指定パラメーター

Communications Server は、次のパラメーターを使用してコールバック・ルーチンを呼び出します。

vcb アプリケーションが提供した VCB へのポインター。 VCB には、Communications Server が設定した戻りパラメーターが含まれます。

tp_id verb を発行した TP の 8 バイトからなる TP ID。

conv_id
verb を発行した会話の会話 ID。

corr アプリケーションが提供した相関係数の値。この値をアプリケーションで使用すると、戻された情報をアプリケーションの別の処理へ関連付けることができます。

コールバック・ルーチンでは、これらのパラメーターをすべて使用する必要はありません。コールバック・ルーチンでは、戻された VCB に対して必要なすべての処理を実行できます。あるいは、verb が完了したことをメインプログラムに通知するために、単に変数を設定するだけでも構いません。

戻り値

この関数は値を戻しません。

非同期 verb の完了のためのコールバック・ルーチンの使用

非同期 verb の完了のためにコールバック・ルーチンを使用する場合、アプリケーションは、必要であればそのコールバック・ルーチンの中から追加の非同期 APPC verb を発行できます。 Communications Server はコールバック・ルーチンの中から発行されたすべての同期 verb を、AP_PARAMETER_CHECK と AP_SYNC_NOT_ALLOWED の 1 次および 2 次戻りコードでリジェクトします。

APPC エントリー・ポイント: Windows システム

WINDOWS

Windows アプリケーションは、次のエントリー・ポイントを使用して APPC にアクセスします。

WinAPPCStartup
アプリケーションを Windows APPC ユーザーとして登録し、 APPC ソフトウェアが、アプリケーションで必要な関数のレベルをサポートするかどうかを判別します。

WinAsyncAPPC
APPC verb を発行します。verb は、通常非同期に完了し、ブロックされません。APPC は、アプリケーション・ウィンドウにメッセージを通知することで、完了を示します。

WinAsyncAPPCEx
APPC verb を発行します。verb が非同期に完了すると、APPC は、イベント・ハンドルをシグナル通知することで完了を示します。verb のブロック化バージョンではなく、この関数を使用すると、同じスレッドで複数のセッションをハンドルすることができます。

WinAPPCCancelAsyncRequest
未解決の非同期 verb (WinAsyncAPPC エントリー・ポイントを使用して発行される) を取り消します。どの verb が未解決であるかにより、これは会話または TP を終了したり、会話で使用されているセッションの活動を停止する可能性があります。

WinAPPCCleanup
アプリケーションが APPC を使用して完了したときに、このアプリケーションの登録を解除します。

APPC APPC verb を発行します。verb がブロックされます。つまり、アプリケーションの処理は、APPC が verb の処理を完了し、結果を戻すまで中断されます。

WinAPPCCancelBlockingCall
未解決のブロッキング verb (APPC エントリー・ポイントを使用して発行される) を取り消します。どの verb が未解決であるかにより、これは会話または TP を終了したり、会話で使用されているセッションの活動を停止する可能性があります。このコールが必要である事情についての詳細は、 49 ページの『ブロッキング Verb』を参照してください。

WinAPPCIsBlocking
このアプリケーションに未解決のブロッキング verb があるかどうかを検査します。このコールが必要である事情についての詳細は、 49 ページの『ブロッキング Verb』を参照してください。

WinAPPCSetBlockingHook
ブロッキング verb の処理中に APPC が使用するブロッキング・プロシージャーを指定します。これは APPC のデフォルトのブロッキング・プロシージャーを置き換えます。ブロッキング・プロシージャーは、verb 処理が完了するまで繰り返して呼び出されます。詳細については、 49 ページの『ブロッキング Verb』を参照してください。

WinAPPCUnhookBlockingHook
前の WinAPPCSetBlockingHook コールで指定されたブロッキング・プロシージャーを登録解除します。そのため、APPC はデフォルトのブロッキング・プロシージャーの使用に戻ります。

GetAppcConfig
指定されたローカル LU およびモードでの使用に構成された リモート LU に関する情報を戻します。この関数は、5250 エミュレーション・プログラムによる使用に提供されます。戻り情報は、Communications Server 構成の 5250 ユーザー・レコードから持ってきます。

GetAppcReturnCode
APPC verb で取得した 1 次および 2 次戻りコードの印刷可能文字ストリングを生成します。

これらのエントリー・ポイントは、Windows APPC ヘッダー・ファイル **winappc.h** に定義されています。このファイルは、Windows Client ソフトウェアをインストールしたディレクトリーにあるサブディレクトリー **/sdk** にインストールされます。

アプリケーションは、APPC verb を発行しようとする前に、`WinAPPCStartup` を呼び出す必要があります。

続いて、次のエントリー・ポイントのいずれかを使用して、APPC verb を発行します。

- `WinAsyncAPPC` または `WinAsyncAPPCEx` (非同期)。 Windows の新規アプリケーションを開発中である場合、次のエントリー・ポイントのいずれかを使用します。
- `APPC` (ブロッキング)。このエントリー・ポイントは、 AIX / Linux APPC インプリメンテーションとの互換性をとるために提供されます。 49 ページの『ブロッキング Verb』には、 Windows 環境でのブロッキング verb がどのように作動するかについて、詳しい情報の説明があります。

5250 エミュレーションを提供するアプリケーションは、`GetAppcConfig` を使用して、指定されたローカル LU を使用してアクセスできるリモート APPC LU に関する情報を取得します。

verb が `AP_OK` 以外の戻りコードで戻る場合、アプリケーションは `GetAppcReturnCode` を使用して、標準エラー・メッセージを生成するために使用できる、これらの戻りコードのテキスト・ストリング表現を取得します。

アプリケーションが APPC verb の発行を完了すると、終了前に `WinAPPCCleanup` を呼び出す必要があります。`WinAPPCCleanup` の呼び出し後、アプリケーションが最初に `WinAPPCStartup` を呼び出して、再度初期化しない限り、アプリケーションはそれ以上の APPC verb の発行を行いません。

次のセクションでは、これらの Windows エントリー・ポイントについて説明しています。

WinAPPCStartup

アプリケーションは `WinAPPCStartup` を使用して、Windows APPC ユーザーとして登録し、APPC ソフトウェアが、このアプリケーションが必要とする Windows APPC バージョンをサポートするかどうかを判別します。

関数コール

```
int WINAPI WinAPPCStartup  (
                             WORD            wVersionRequired;
                             WAPPCDATA far * lpData;
                           )

typedef struct
 {
   WORD          wVersion;
   char          szDescription[128];
 } WAPPCDATA;
```

指定パラメーター

アプリケーションで WinAPPCStartup エントリー・ポイントを使用して verb を発行する場合は、次のパラメーターをアプリケーションから提供します。

wVersionRequired

アプリケーションが必要とする Windows APPC のバージョンです。下位のバイトは、主バージョンの番号を指定し、上位のバイトはマイナー・バージョン番号を指定します。次に例を示します。

バージョン	wVersionRequired
1.0	0x0001
1.1	0x0101
2.0	0x0002

アプリケーションで複数のバージョンが使用可能である場合、アプリケーションが使用できるバージョンで一番高いバージョンを指定します。

戻り値

WinAPPCStartup は次のいずれかの値を戻します。

0 (zero)

アプリケーションは正常に登録され、Windows APPC ソフトウェアは、アプリケーションにより指定されたバージョン番号以下のバージョンをサポートしています。アプリケーションは、WAPPCDATA 構造体にあるバージョン番号をチェックして、それがサポートされるレベルであるかどうか確認する必要があります。

WAPPCVERNOTSUPPORTED

アプリケーションにより指定されたバージョン番号が、 Windows APPC ソフトウェアがサポートする最も低いバージョンよりも低いものでした。アプリケーションは登録されませんでした。

WAPPCSYSNOTREADY

アプリケーションは登録されませんでした。これは、Windows ソフトウェア上の Remote API Client が開始されなかったか、またはローカル・ノードがアクティブでないためか、あるいはリソース不足などの別のシステム障害が発生したためであると考えられます。

WinAPPCStartup の戻り値が 0 (ゼロ) である場合、 WAPPCDATA 構造体には、Windows APPC ソフトウェアによるサポート情報が含まれます。戻り値がゼロ以外

の値である場合、この構造体の内容は未定義で、アプリケーションは構造体の内容をチェックする必要はありません。この構造体にあるパラメーターは次のとおりです。

wVersion

wVersionRequired パラメーターと同一形式で、ソフトウェアがサポートする Windows APPC のバージョン番号です。ソフトウェアが要求されるバージョン番号をサポートする場合、このパラメーターは *wVersionRequired* パラメーターと同一の値に設定されます。そうでない場合、ソフトウェアがサポートする最も高いバージョンに設定されますが、これは、アプリケーションによって指定されるバージョン番号より低いものです。アプリケーションは戻り値を検査し、以下のようなアクションを行います。

- 戻されたバージョン番号が要求されたバージョン番号と同一である場合、アプリケーションはこの Windows APPC インプリメンテーションを使用することができます。
- 戻されたバージョン番号が要求されたバージョン番号より低い場合、アプリケーションはこの Windows APPC インプリメンテーションを使用できますが、要求されたバージョン番号でサポートされない機能は使用しないでください。低いバージョンでは使用できない機能を要求したために実行できない場合、初期化に失敗し、APPC verb を発行しなくなります。

szDescription

Windows APPC ソフトウェアを記述するテキスト・ストリング。

WinAsyncAPPC

アプリケーションは、この関数を使用して APPC verb を発行します。 verb が非同期に完了すると、APPC は、アプリケーションの Windows ハンドルにメッセージを通知することで完了を示します。

最初に `WinAsyncAPPC` コールを使用する前に、アプリケーションが `RegisterWindowMessage` コールを使用して、非同期 verb が完了したことを知らせるメッセージに APPC が使用するメッセージ ID を取得する必要があります。詳細については、 46 ページの『使用法』を参照してください。

関数コール

```
HANDLE WINAPI WinAsyncAPPC  (
                              HWND  hWnd,
                              long  vcbptr
                            );
```

指定パラメーター

指定パラメーターは次のとおりです。

hWnd APPC が、非同期 verb が完了したことを知らせるメッセージの通知に使用するウィンドウ・ハンドルです。

vcbptr verb の VCB 構造体へのポインター。このパラメーターは、長整数として定義されているため、ポインターから長整数へキャストする必要があります。 VCB 構造体と、個々の verb の使用方法についての詳細は、 73 ページの『第 3 章 APPC 制御 verb』と 105 ページの『第 4 章 APPC 会話 verb』を参照してください。

注: APPC VCB には、「予約済み」とマークされたパラメーターが多数含まれています。予約済みパラメーターには、Communications Server ソフトウェアで内部的に使用されているものや、このバージョンでは使用されていなくても将来のバージョンで使用される可能性があるものがあります。アプリケーションでは、予約済みパラメーターにアクセスしてはなりません。その代わりに、verb で使用する他のパラメーターを設定する前に、VCB の全内容をゼロに設定し、これらのパラメーターすべてをゼロにしてください。このようにすると、内部的に使用されるパラメーターが Communications Server で誤って解釈されることがなく、将来のバージョンでこれらのパラメーターが新しい関数を提供するために使用されても、アプリケーションは引き続き動作します。

VCB の内容をゼロに設定するには、`memset` を使用します。

```
memset(vcb, 0, sizeof(vcb));
```

戻り値

関数からの戻り値は次のいずれかです。

ハンドル (`Handle`)
関数コールが成功しました (承認されました)。verb が後で完了すると、APPC はこのハンドルをアプリケーションのウィンドウ・プロシージャーに渡すメッセージの ID として使用します (詳細については、『使用法』を参照してください)。アプリケーションは、未解決の verb を取り消す必要がある場合、このハンドルを `WinAPPCCancelAsyncRequest` コールへのパラメーターとしても使用します。

`0` (zero)
関数コールが成功しませんでした (承認されませんでした)。

使用法

最初に `WinAsyncAPPC` を使用する前に、アプリケーションが `RegisterWindowMessage` コールを使用して、非同期 verb が完了したことを知らせるメッセージに APPC が使用するメッセージ ID を取得する必要があります。`RegisterWindowMessage` は、標準の Windows 関数コールで、APPC に固有ではありません。関数についての詳細は、Windows の資料を参照してください。(後続の APPC verb の前に再度このコールを発行する必要はありません。戻り値は、アプリケーションが発行したすべてのコールのものと同じです。)

アプリケーションは、ストリング 『WinAsyncAPPC』 を関数に渡します。戻り値はメッセージ ID です。

`WinAsyncAPPC` エントリー・ポイントを使用して発行された APPC verb が非同期に完了するたびに、APPC は `WinAsyncAPPC` コールで指定される Windows ハンドルにメッセージを通知します。メッセージのフォーマットは次のとおりです。

- メッセージ ID は、`RegisterWindowMessage` コールから戻された値です。
- *lParam* 引数には、元の `WinAsyncAPPC` コールに指定された VCB のアドレスが入ります。アプリケーションは、このアドレスを使用して、 VCB 構造体の戻り値にアクセスします。
- *wParam* 引数には、元の `WinAsyncAPPC` コールに戻ったハンドルが入ります。

WinAsyncAPPCEx

アプリケーションは、この関数を使用して APPC verb を発行します。verb が非同期に完了すると、APPC は、イベント・ハンドルをシグナル通知することで完了を示します。verb のブロック化バージョンではなく、この関数を使用すると、同じスレッドで複数のセッションをハンドルすることができます。

関数コール

```
HANDLE WINAPI WinAsyncAPPCEx  (
                                HANDLE  eventhandle,
                                long    vcbptr
                              );
```

指定パラメーター

指定パラメーターは次のとおりです。

eventhandle

APPC が非同期 verb の完了を示すためにシグナル通知するイベント・ハンドル。

vcbptr verb の VCB 構造体へのポインター。このパラメーターは、長整数として定義されているため、ポインターから長整数へキャストする必要があります。 VCB 構造体と、個々の verb の使用方法についての詳細は、 73 ページの『第 3 章 APPC 制御 verb』と 105 ページの『第 4 章 APPC 会話 verb』を参照してください。

注: APPC VCB には、「予約済み」とマークされたパラメーターが多数含まれています。予約済みパラメーターには、Communications Server ソフトウェアで内部的に使用されているものや、このバージョンでは使用されていなくても将来のバージョンで使用される可能性があるものがあります。アプリケーションでは、予約済みパラメーターにアクセスしてはなりません。その代わりに、verb で使用する他のパラメーターを設定する前に、VCB の全内容をゼロに設定し、これらのパラメーターすべてをゼロにしてください。このようにすると、内部的に使用されるパラメーターが Communications Server で誤って解釈されることがなく、将来のバージョンでこれらのパラメーターが新しい関数を提供するために使用されても、アプリケーションは引き続き動作します。

VCB の内容をゼロに設定するには、`memset` を使用します。

```
memset(vcb, 0, sizeof(vcb));
```

戻り値

関数からの戻り値は次のいずれかです。

ハンドル (Handle)

関数コールが成功 (承認) し、戻り値は非同期タスク・ハンドルです。verb が後で完了すると、 APPC はこのハンドルをアプリケーションへのイベント通知に使用します (詳細については、 48 ページの『使用法』を参照してください)。アプリケーションは、未解決の verb を取り消す必要がある場合、このハンドルを `WinAPPCCancelAsyncRequest` コールへのパラメーターとしても使用します。

0 (zero)
関数コールが成功しませんでした (承認されませんでした)。

使用法

この関数は、Windows API の WaitForSingleObject または WaitForMultipleObjects と併用されます。非同期操作が完了すると、アプリケーションはイベントのシグナルから通知を受けます。イベントのシグナル通知にエラー状態がないか、1 次戻りコードと 2 次戻りコードを調べてください。

WinAPPCCancelAsyncRequest

アプリケーションはこの関数を使用して、未解決の APPC verb (WinAsyncAPPC エントリー・ポイントを使用して発行) を取り消します。

関数コール

```
int WINAPI WinAPPCCancelAsyncRequest (HANDLE  Handle);
```

指定パラメーター

指定パラメーターは次のとおりです。

ハンドル (Handle)
verb の元の WinAsyncAPPC コールに戻ったハンドルです。

戻り値

関数からの戻り値は次のいずれかです。

0 (zero)
未解決の verb が正常に取り消されました。

WAPPCINVALID
指定パラメーターは、未解決の APPC verb のいずれのハンドルにも一致しませんでした。

WAPPCALREADY
指定ハンドルで示した APPC verb はすでに完了しています。 (アプリケーションが verb の完了から発生するメッセージをすでに処理しているか、メッセージがアプリケーションのメッセージ・キューでまだ待機している可能性があります。)

使用法

未解決 verb の取り消しの他に、さらに APPC は、verb が発行された会話または TP の終了や、セッションの終了 (またはこの両方の終了) もすることがあります。実行されるアクションは、取り消された verb により異なります。APPC は、アプリケーションに対して、取り消された verb の完了を示すメッセージを通知します。verb の 1 次戻りコードは AP_CANCELLED です。

WinAPPCCleanup

アプリケーションは、APPC verb の発行を完了後、WinAPPCCleanup 関数を使用して Windows APPC ユーザーとして登録を抹消します。

関数コール

```
BOOL WINAPI WinAPPCCleanup (void);
```

指定パラメーター

WinAPPCCleanup 関数ではパラメーターは指定されません。

戻り値

関数からの戻り値は次のいずれかです。

TRUE アプリケーションは正常に登録解除されました。

FALSE コールの処理中にエラーが発生しました。アプリケーションは登録抹消されませんでした。ログ・ファイルを確認して、エラーの原因を示すメッセージを参照してください。

ブロッキング Verb

このセクションでは、呼び出し側アプリケーションがシングル・スレッドの場合にブロッキング verb が Windows 環境でどのように動作するかと、ブロッキング verb を使用するためにアプリケーションを記述する際に留意しなければならない情報について説明します。(通常、 Windows アプリケーションは、マルチスレッドを使用して、ブロッキング verb のプログラムがアプリケーション全体をブロックする問題を回避します。)

APPC エントリー・ポイントに発行された verb は、 verb 処理が完了するまでアプリケーションを中断するように見えますが、APPC ライブラリーは、 Remote API Client が処理を完了するのを待つ間、他のプロセスが稼働できるように、システムの制御を放棄する必要があります。そのために、アプリケーションはブロッキング関数を使用し、ライブラリーが待機中にこの関数が繰り返し呼び出されます。この関数により、Windows メッセージを他のプロセスに送信することができます この関数についての詳細は、 50 ページの『デフォルトのブロッキング関数』を参照してください。

ブロッキング関数は、元のブロッキング verb を発行したアプリケーションにメッセージをディスパッチすることができます。この場合、アプリケーションは、未解決のブロッキング・コールがあっても再入できます。これらの環境では、アプリケーションは APPC verb の発行に関連しない他の処理を継続することができます。ただし、最初の verb が未解決であるうちは、他の verb を APPC エントリー・ポイント (または、他の Remote API Client API) に発行することはできません。 verb は 1 次戻りコード AP_THREAD_BLOCKING で拒否されます。

アプリケーションは、WinAPPCIsBlocking 関数を使用して、ブロッキング verb が未解決であるか (すなわち、verb が未解決である間に受信したメッセージの結果として再入力されたかどうか) を確認することができます (詳細については、 52 ページの『WinAPPCIsBlocking』を参照してください)。この関数がブロッキング・コールが未解決であることを示す場合、アプリケーションは、ブロッキング・エントリー・ポイントを使用して、さらに APPC verb を発行すべきではありません。ただし、アプリケーションは以下を行うことができます。

- 他の処理を継続する。
- 非同期エントリー・ポイントを使用して APPC verb を発行する。

- WinAPPCCancelBlockingCall を発行して、未解決のブロッキング verb を取り消す。

デフォルトのブロッキング関数

Windows APPC ライブラリーで使用される標準のブロッキング関数は、次の通りです。

```
BOOL far pascal DefaultBlockingHook (void)  {
       MSG msg;
       /* get the next message if any */
       if ( PeekMessage (&msg,NULL,0,0,PM_NOREMOVE)  )  {
             if ( msg.message = = WM_QUIT  )
                    return FALSE;   // let app process WM_QUIT
             PeekMessage (&msg,NULL,0,0,PM_REMOVE);
             TranslateMessage (&msg);
             DispatchMessage (&msg);
       }
       /* TRUE if no WM_QUIT received */
       return TRUE;
}
```

アプリケーションが、ブロッキング関数の一部として実行される他の処理を含む必要がある場合、固有のブロッキング関数を指定して、APPC が提供するデフォルトのものと置き換えることができます。これを行うには、WinAPPCSetBlockingHook コールを使用します (52 ページの『WinAPPCSetBlockingHook』を参照してください)。

ブロッキング関数は、WM_QUIT メッセージを受け取ると、FALSE を戻す必要があります。これは、Windows APPC はアプリケーションに制御を戻し、アプリケーションはメッセージを処理して終了できるということです。そうでない場合、関数は TRUE を戻す必要があります。

APPC

アプリケーションは、この関数を使用して APPC verb を発行します。これは、verb 処理が完了するまでブロックされます。以前にインプリメントされた APPC との互換性を保つため、Remote API Client はエントリー・ポイント APPC_C および APPC_P も提供します。これらのエントリー・ポイントは、APPC と同様に使用できます。

このエントリー・ポイントでは、Windows の同期 APPC verb に対するサポートを指定します。これは、他のオペレーティング・システム環境からのマイグレーションに有効です。

関数コール

```
void WINAPI APPC  (
                    long  vcbptr
                  )
```

指定パラメーター

指定パラメーターは次のとおりです。

vcbptr verb の VCB 構造体へのポインター。このパラメーターは、長整数として定義されているため、ポインターから長整数へキャストする必要がありま

す。 APPC verb ごとの VCB 構造体の定義については、 73 ページの『第 3 章 APPC 制御 verb』および 105 ページの『第 4 章 APPC 会話 verb』を参照してください。

注: APPC VCB には、「予約済み」とマークされたパラメーターが多数含まれています。予約済みパラメーターには、Communications Server ソフトウェアで内部的に使用されているものや、このバージョンでは使用されていなくても将来のバージョンで使用される可能性があるものがあります。アプリケーションでは、予約済みパラメーターにアクセスしてはなりません。その代わりに、verb で使用する他のパラメーターを設定する前に、VCB の全内容をゼロに設定し、これらのパラメーターすべてをゼロにしてください。このようにすると、内部的に使用されるパラメーターが Communications Server で誤って解釈されることがなく、将来のバージョンでこれらのパラメーターが新しい関数を提供するために使用されても、アプリケーションは引き続き動作します。

VCB の内容をゼロに設定するには、`memset` を使用します。

```
memset(vcb, 0, sizeof(vcb));
```

戻り値

この関数は値を戻しません。コールが戻った時点で、アプリケーションは VCB 構造体内の *primary_rc* と *secondary_rc* パラメーターをチェックして、verb が正常に終了したかどうかを判別できます。 VCB 構造体で戻るパラメーターについての詳細は、 73 ページの『第 3 章 APPC 制御 verb』と 105 ページの『第 4 章 APPC 会話 verb』の個々の verb の説明を参照してください。

WinAPPCCancelBlockingCall

アプリケーションは `WinAPPCCancelBlockingCall` 関数を使用して、未解決のブロッキング APPC verb (APPC エントリー・ポイントを使用して発行) を取り消します。

関数コール

```
BOOL WINAPI WinAPPCCancelBlockingCall (void);
```

指定パラメーター

このエントリー・ポイントではパラメーターは指定されません。(未解決のブロッキング verb はいつでも 1 個しか存在しないため、取り消し対象の verb を示す必要はありません。)

戻り値

関数からの戻り値は次のいずれかです。

TRUE 未解決の verb が正常に取り消されました。

FALSE 未解決のブロッキング APPC verb がなかったか、コールと verb の処理中に発生したエラーが取り消されませんでした。

使用法

未解決 verb の取り消しの他に、さらに APPC は、verb が発行された会話の終了とセッションの終了もすることがあります。verb が、RECEIVE_ALLOCATE や TP_STARTED などの会話にではなく、TP へ関連付ける場合、 APPC はその TP を終了します。

WinAPPCIsBlocking

アプリケーションは WinAPPCIsBlocking 関数を使用して、未解決のブロッキング APPC verb (APPC エントリー・ポイントを使用して発行される verb) があるかどうかを確認します。

関数コール

```
BOOL WINAPI WinAPPCIsBlocking (void);
```

指定パラメーター

この関数ではパラメーターは指定されません。

戻り値

関数からの戻り値は次のいずれかです。

TRUE ブロッキング APPC verb が未解決です。必要な場合、アプリケーションは WinAPPCCancelBlockingCall 関数を使用してこの verb を取り消します。

FALSE ブロッキング APPC verb は解決しています。

WinAPPCSetBlockingHook

アプリケーションは、このコールを使用して、APPC がデフォルトのブロッキング関数の代わりに使用する、固有のブロッキング関数を指定します。ブロッキング関数がどのように動作するかということと、実行する必要のある関数についての詳細は、 49 ページの『ブロッキング Verb』を参照してください。

関数コール

```
FARPROC WINAPI WinAPPCSetBlockingHook (FARPROC lpBlockFunc);
```

指定パラメーター

指定パラメーターは次のとおりです。

lpBlockFunc

アプリケーションのブロッキング関数のプロシージャー・インスタンス・アドレスです。アプリケーションは、 MakeProcInstance コールを使用して、このアドレスを取得する必要があります。詳細は、Windows の文書を参照してください。

戻り値

戻り値は、直前のブロッキング関数のプロシージャー・インスタンス・アドレスです。アプリケーションが複数のブロッキング関数を使用し、後で直前のブロッキング関数を復元する必要のある場合、このアドレスを保管する必要があります。保管した値を使用し、再度 WinAPPCSetBlockingHook を発行して、直前のブロッキング

関数を復元することができます。使用しているブロッキング関数が 1 つだけで、直前の値を復元する必要のない場合、このコールからの戻り値を無視することができます。

使用法

新規のブロッキング関数は、アプリケーションが以下のコールのいずれかを発行するまで有効です。

- WinAPPCSetBlockingHook (異なるプロシージャー・インスタンス・アドレスを持つ)。新規のブロッキング関数を指定するか、直前のブロッキング関数を復元する。
- WinAPPCUnhookBlockingHook (『WinAPPCUnhookBlockingHook』を参照してください)。現在のブロッキング関数の使用を停止し、デフォルトのブロッキング関数に戻す。

WinAPPCUnhookBlockingHook

アプリケーションは、このコールを使用して、これより前に WinAPPCSetBlockingHook を使用して指定した、固有のブロッキング関数を除去し、APPC のデフォルトのブロッキング関数を使用するように戻します。

関数コール

```
BOOL WINAPI WinAPPCUnhookBlockingHook (void);
```

指定パラメーター

この関数ではパラメーターは指定されません。

戻り値

関数からの戻り値は次のいずれかです。

TRUE ブロッキング機能は、正常に除去されました。今後、ブロッキング・コールは、デフォルトのブロッキング関数を使用します。

FALSE コールが正常に完了しませんでした。

GetAppcConfig

GetAppcConfig 関数は、5250 エミュレーション・プログラムによる使用に提供されます。この関数は、Communications Server 構成の 5250 エミュレーション・ユーザー・レコードで定義されているように、指定されたローカル LU がアクセスできるリモート LU に関する情報を戻します。

このコールで必要な情報を判断するため、CS/AIX では、Windows クライアントに構成されたユーザー名を、 Communications Server 構成で定義された 5250 ユーザー・レコードに対してチェックします。 (または、ユーザー名が定義されていない場合は、<DEFAULT> レコードをチェックします)。適切なユーザー・レコードでは、セッション定義に対して、このコールで指定されたローカル LU 別名とモード名を突き合わせを行い、一致するセッションごとにリモート LU を戻します。

アプリケーションは、verb が非同期に完了するときに APPC がメッセージを通知する Windows のハンドルを提供します。最初に GetAppcConfig を使用する前に、アプリケーションが RegisterWindowMessage を使用して、コールが非同期に完了した

ことを知らせるメッセージに APPC が使用するメッセージ ID を取得し、WinAPPCStartup を使用して Windows の APPC アプリケーションとして登録する必要があります。詳細については、 43 ページの『WinAPPCStartup』と 56 ページの『使用法』にある WinAPPCStartup の説明を参照してください。

コールが完了したことを示す別の方法として、整数値へのポインター (*AsyncRetCode* パラメーター) の提供があります。 APPC は、コールが失敗したか、処理中であるか、または完了したかを示す値を戻すことができます。Windows アプリケーションでは、Windows のハンドルを提供する最初の方法を使用することを推奨します。

関数コール

```
HANDLE WINAPI GetAppcConfig (
                              HWND       hWnd,
                              char far   *LocalLU,
                              char far   *Mode,
                              int far    *NumRemLU,
                              int        MaxRemLU,
                              char far   *RemLU,
                              int far    *AsyncRetCode
                            );
```

指定パラメーター

指定パラメーターは次のとおりです。

hWnd コールが非同期に完了したことを知らせるメッセージの通知に APPC が使用するウィンドウ・ハンドルです。このパラメーターが使用される場合は、*AsyncRetCode* パラメーターへのポインターを null ポインターにしてください。

LocalLU

構成情報が必要なローカル LU の別名へのポインターです。これは、8 文字以内で、ヌル文字 (2 進ゼロ) で終了する ASCII ストリングです。LU 別名が 8 文字より短いと、ヌル文字が入ります。スペースは入りません。

デフォルトのローカル LU を示すには、このパラメーターをヌル文字のあとに ASCII スペースを 8 個入れたストリングを指すよう設定します。

Mode 構成情報が必要なモード名 (ローカル LU が使用) へのポインターです。これは、8 文字以内で、ヌル文字 (2 進ゼロ) で終了する ASCII ストリングです。モード名が 8 文字より短いと、ヌル文字が入ります。スペースは入りません。 5250 エミュレーション・プログラムの場合、モード名は常に QPCSUPP です。

NumRemLU

APPC が構成済みのリモート LU 数を戻すために使用する整数へのポインターです。

MaxRemLU

指定されたデータ・バッファー (以下のパラメーターを参照してください) に適応できるリモート LU の別名の最大数です。1 つの LU 別名に 9 バイトが必要なため、指定されるバッファーの長さは、 *MaxRemLU* で指定された値の 9 倍は必要です。

RemLU

戻されるリモート LU 別名が入るバッファーです。

AsyncRetCode

アプリケーションが、完了の通知に推奨方法を使用していると、このパラメーターが予約されます。アプリケーションはヌル・ポインターを指定する必要があります。

アプリケーションが推奨方法以外を使用していると、このパラメーターは、関数からの非同期戻りコードに APPC が使用する整数へのポインターとなります。この場合、*hWnd* パラメーターをヌル・ハンドルにしてください。

戻り値

コールが戻った時点で、アプリケーションは式「*ReturnedHandle* & APPC_CFG_SUCCESS」の値を調べて、関数が正常に終了したかどうかを判別できます。

式 「*ReturnedHandle* & APPC_CFG_SUCCESS」の値が TRUE であり、アプリケーションが完了の通知に推奨方法を使用していると、戻り値はハンドルになります。関数が後で完了すると、 APPC はこのハンドルをアプリケーションのウィンドウ・プロシージャーに渡すメッセージの ID として使用します (詳細については、 56 ページの『使用法』を参照してください)。

式 「*ReturnedHandle* & APPC_CFG_SUCCESS」の値が TRUE であり、アプリケーションが完了の通知に推奨方法以外を使用していると、*AsyncRetCode* パラメーターが APPC_CFG_PENDING に設定されます。完了したかどうか確認するために、アプリケーションはこの値を定期的に調べる必要があります。関数が後で完了すると、APPC はこのパラメーターを 56 ページの『使用法』にリストされている非同期戻りコードのいずれかに設定します。

式の値が FALSE の場合、関数コールは承認されません。*ReturnedHandle* の値は以下のうちいずれかです。

APPC_CFG_ERROR_NO_APPC_INIT

アプリケーションが WinAPPCStartup コールを発行していません。このコールは GetAppcConfig が使用される前に発行してください。

APPC_CFG_ERROR_INVALID_HWND

アプリケーションが、無効な Windows ハンドルを指定しました。

APPC_CFG_ERROR_BAD_POINTER

アプリケーションが、完了の通知に代替方法を使用するため Windows ハンドルへのヌル・ポインターを指定しました。ただし、*AsyncRetCode* パラメーターに指定されたポインターが無効でした。

APPC_CFG_ERROR_UNCLEAR_COMPLETION_MODE

アプリケーションは、Windows ハンドル (*hWnd* パラメーターで指定) と、*AsyncRetCode* パラメーターへのヌル以外のポインターとの両方を指定しました。そのため APPC は非同期完了を通知する方法を判断できませんでした。

APPC_CFG_ERROR_TOO_MANY_REQUESTS

すでに未解決となっている GetAppcConfig 要求が多すぎます。アプリケーションを中断し、他のプロセスを実行できるようにして、このコールを後からやり直してください。

APPC_CFG_ERROR_GENERAL_FAILURE
システム・エラーが発生しました。

使用法

最初に GetAppcConfig を使用する前に、アプリケーションが RegisterWindowMessage コールを使用して、非同期完了を知らせるメッセージに APPC が使用するメッセージ ID を取得する必要があります。RegisterWindowMessage は、標準の Windows 関数コールで、APPC に固有ではありません。関数についての詳細は、Windows の資料を参照してください。(後続の GetAppcConfig コールの前にアプリケーションが再度このコールを発行する必要はありません。戻り値は、アプリケーションが発行したすべてのコールのものと同じです。)

アプリケーションは、値 WIN_APPC_CFG_COMPLETION_MSG を関数に渡します。以下に述べるように、戻り値はメッセージ ID です。

アプリケーションは、Windows ハンドルを使用するか、次のように別の方法を使用して、完了を通知することができます。

- アプリケーションが、完了の通知に Windows ハンドルを使用していると、APPC はコールが非同期に完了したときに、この Windows ハンドルへメッセージを通知します。メッセージのフォーマットは次のとおりです。
  - メッセージ ID は、RegisterWindowMessage コールから戻された値です。
  - *wParam* 引数には、元の GetAppcConfig コールに戻ったハンドルが入ります。
  - *lParam* 引数には、次の非同期戻りコードのいずれかが入ります。

    APPC_CFG_SUCCESS_NO_DEFAULT_REMOTE
    構成が正常に取得されました。指定されたローカル LU とモードに対して構成されたデフォルトのリモート LU はありません。

    APPC_CFG_SUCCESS_DEFAULT_REMOTE
    構成が正常に取得されました。指定されたローカル LU とモードに対してデフォルトのリモート LU が構成されます。(Communications Server には、デフォルトのリモート LU を構成する概念がないため、この値は戻されません。ただし、アプリケーションでは、この戻りコードが他の APPC インプリメンテーションと互換性を持たせるようにする必要があります。)

    APPC_CFG_ERROR_NO_DEFAULT_LOCAL_LU
    アプリケーションは、デフォルトのローカル LU を示すブランクのローカル LU 別名を指定しましたが、デフォルトのローカル LU が構成されていません。

    APPC_CFG_ERROR_BAD_LOCAL_LU
    指定されたローカル LU 別名が、5250 エミュレーションで使用されている構成済みのローカル LU に一致しませんでした。

    APPC_CFG_ERROR_GENERAL_FAILURE
    システム・エラーが発生しました。
- アプリケーションが、完了の通知に代替方法を使用していると、 APPC は、コールの完了時点で、非同期戻りコードを *lParam* 引数 (Windows メッセージ内) のリストにある戻りコードのいずれかに設定します。

アプリケーションは、「*RetCode* & APPC_CFG_SUCCESS」式または「*RetCode* & APPC_CFG_FAILURE」式をテストして、正常に終了しているか、失敗しているかを確認します。*RetCode* は Windows メッセージ内の *lParam* 引数であるか、アプリケーションに戻る *AsyncRetCode* パラメーターのいずれかです。「*RetCode* & APPC_CFG_SUCCESS」が TRUE であると、コールは正常に終了しています。「*RetCode* & APPC_CFG_FAILURE」が TRUE であると、コールは失敗しています。

コールが正常に終了していると、次にアプリケーションは *NumRemLU* と *RemLU* パラメーターの値をチェックします。

- *NumRemLU* には、構成済みのリモート LU の合計が入ります。この数が、指定された *MaxRemLU* パラメーターより大きいと、指定されたバッファーは、リモート LU 別名をすべて入れるには大きさが不足しています。アプリケーションは戻された別名を使用するか、あるいはすべての別名が入る大きさのバッファーを指定して GetAppcConfig を再発行することができます。
- *RemLU* には、リモート LU の別名が入ります。別名はそれぞれ、ヌル文字に続く 8 文字までのストリングとなり、バッファーの 9 バイトを使用します。LU 別名の戻り数は、指定パラメーター *MaxRemLU* と、戻りパラメーター *NumRemLU* の小さい方になります。

このコールで必要な情報を判断するため、CS/AIX では、Windows クライアントに構成されたユーザー名を、 Communications Server 構成で定義された 5250 ユーザー・レコードに対してチェックします。 (または、ユーザー名が定義されていない場合は、<DEFAULT> レコードをチェックします)。適切なユーザー・レコードでは、セッション定義に対して、このコールで指定されたローカル LU 別名とモード名を突き合わせを行い、一致するセッションごとにリモート LU を戻します。

GetAppcReturnCode

このコールは、指定された VCB からの戻りコードを解釈する印刷可能文字ストリングを戻します。このストリングは、AP_OK 以外の戻りコードに対してアプリケーションのエラー・メッセージを生成するために使用されます。

このコールでは、APPC アプリケーションのエンド・ユーザーに表示するストリングを提供します。構成上の問題やユーザー・エラー (必要なコンポーネントが構成されていなかったり、開始されていないなど) を示す戻りコードの場合、ストリングには、ユーザーが問題を修正できる十分な情報が入ります。アプリケーション・エラー (アプリケーションが無効な verb を発行したり、必須パラメーターを指定できないなど) を示す戻りコードの場合、一般的にはユーザーが問題を修正することはできません。この場合、ストリングはアプリケーション開発者にとってのみ重要となります。

関数コール

```
int WINAPI GetAppcReturnCode (
                                long           vcbptr,
                                unsigned int   buffer_length,
                                unsigned char far *  buffer_addr
                               );
```

指定パラメーター

指定パラメーターは次のとおりです。

vcbptr verb の VCB 構造体へのポインター。このパラメーターは、長整数として定義されているため、ポインターから長整数へキャストする必要があります。 VCB 構造体と、個々の verb の使用方法についての詳細は、 73 ページの『第 3 章 APPC 制御 verb』または 105 ページの『第 4 章 APPC 会話 verb』を参照してください。

buffer_length
戻りデータ・ストリングを保持するアプリケーションで指定されるバッファーの長さ (バイト単位)です。推奨長は 256 バイトです。

buffer_addr
戻りデータ・ストリングを保持するアプリケーションで指定されるバッファーのアドレスです。

戻り値

関数からの戻り値は次のいずれかです。

0 (zero)
関数の処理が正常に完了しました。戻り文字ストリングは、 *buffer_addr* パラメーターで示されるバッファーにあります。このストリングは、ヌル文字 (2 進ゼロ) で終了します。このあとに改行 (¥n) 文字はつきません。

0x20000001
APPC は、指定された VCB から読み取ることができなかったか、指定されたデータ・バッファーに書き込むことができませんでした。

0x20000002
指定されたデータ・バッファーが小さすぎるため、戻り文字ストリングが保持できません。

0x20000003
この関数に戻り文字ストリングを生成するダイナミック・リンク・ライブラリー **APPCST32.DLL** をロードできませんでした。

AIX または Linux の考慮事項

AIX, LINUX

このセクションでは、AIX または Linux 環境で使用する TP を開発する際に、考慮する必要のある情報を要約しています。

複数のプロセス

TP_STARTED または RECEIVE_ALLOCATE を開始したプロセスが fork して子プロセスを作成する場合、その子プロセスは、親プロセスに戻された *tp_id* を使用す

ることはできません。ただし、子プロセスは独自の TP_STARTED または RECEIVE_ALLOCATE を発行して独自の *tp_id* を取得できます。

同じ TP の 2 つ以上のインスタンスを異なるプロセスとして実行できますが、個々のインスタンスには独自の *tp_id* が割り当てられます。

1 つのプロセスが、個々に独自の *tp_id* を備えた多数の TP を含むようなアプリケーションを作成することもできます。 ただし、そのようなアプリケーションは、「デッドロック」状態にならないよう、慎重に設計する必要があります。デッドロック状態では、 APPC verb は同じプロセス内にある別の会話や TP の状態が原因となり、完了することができません。 そのような状態が起きる可能性があるのは、プログラムが 1 つの会話上で情報が送信されてくるのを待ってから何か別のデータを戻すことになっており、そのデータを同じプロセスから別の会話が待っていて、そのデータを受信しなければ最初の会話が必要としている情報を送信できないというような場合です。これは、TP ごとに別個のプロセスを使用することによって、ある程度回避できます。

APPC アプリケーションのコンパイルとリンク

AIX アプリケーション

32 ビット・アプリケーションのコンパイルとリンクを行うには、次のオプションを使用します。

-bimport:/usr/lib/sna/appc_r.exp -I /usr/include/sna

64 ビットのアプリケーションをコンパイルおよびリンクする場合は、以下のオプションを使用します。

-bimport:/usr/lib/sna/appc_r64_5.exp -I /usr/include/sna

Linux アプリケーション

APPC アプリケーションのコンパイルとリンクを行う前に、共用ライブラリーが保管されているディレクトリーを指定します。これにより、アプリケーションが実行時に共用ライブラリーを見つけることができます。64 ビット・アプリケーションをコンパイルする場合は、環境変数 `LD_RUN_PATH` を **/opt/ibm/sna/lib** または **/opt/ibm/sna/lib64** に変更してください。

32 ビット・アプリケーションのコンパイルとリンクを行うには、次のオプションを使用します。

-I /opt/ibm/sna/include -L /opt/ibm/sna/lib -lappc -lsna_r -lpthread -lpLiS

64 ビットのアプリケーションをコンパイルおよびリンクする場合は、以下のオプションを使用します。

-I /opt/ibm/sna/include -L /opt/ibm/sna/lib64 -lappc -lsna_r -lpthread -lpLiS

オプション -lpLiS は、Communications Server サーバー上でアプリケーションを実行する場合のみ必要です。アプリケーションを IBM Remote API Client 上でビルドして、それをクライアントでのみ実行する場合は、このオプションを使用する必要はありません。このオプションを使用する代わりに、アプリケーションをコンパイルおよびリンクする前に、環境変数の LD_PRELOAD を **/usr/lib/libpLiS.so** に設定することができます。

Windows の考慮事項

WINDOWS

このセクションでは、Windows の Remote API Client 上でアプリケーションを開発する際に、認識する必要のある考慮事項の処理を要約しています。Windows 処理の考慮事項は次のとおりです。

- APPC プログラムのコンパイルとリンク
- アプリケーションの終了

APPC プログラムのコンパイルとリンク

Windows で APPC プログラムのコンパイルとリンクを行う場合、次の処理上の考慮事項が重要です。

構造体パッキングのコンパイラー・オプション

APPC verb に対する VCB 構造体はパックされません。このパッキング・メソッドを変更するコンパイラー・オプションを使用しないでください。BYTE パラメーターは BYTE 境界に、WORD パラメーターは WORD 境界に、 DWORD パラメーターは DWORD 境界に関するものです。

ヘッダー・ファイル

Windows APPC アプリケーションにインクルードされるメインの APPC ヘッダー・ファイルは **winappc.h** という名前です。アプリケーションが GetAppcConfig コールを使用すると、**appccfg.h** ヘッダー・ファイルもインクルードする必要があります。これらのファイルは、Windows Client ソフトウェアをインストールしたディレクトリー内の、サブディレクトリー **\sdk** (32 ビット・アプリケーションの場合) または **\sdk64** (64 ビット・アプリケーションの場合) にインストールされています。

ロード時リンク

ロード時に TP を APPC にリンクするには、TP をライブラリー **\sdk\wappc32.lib** (32 ビット・アプリケーションの場合)、または **\sdk64\wappc32.lib** (64 ビット・アプリケーションの場合) にリンクします。

実行時リンク

実行時に TP を APPC にリンクするには、 TP に以下のコールを組み込みます。

- APPC ダイナミック・リンク・ライブラリー **wappc32.dll** をロードする LoadLibrary

- 必要な APPC エントリー・ポイント (WinAsyncAPPC、WinAPPCStartup、および WinAPPCCleanup など) ごとに APPC を指定する GetProcAddress
- ライブラリーが不要になった場合の FreeLibrary

アプリケーションの終了

APPC は、Windows でアプリケーションが終了するタイミングを識別できません。このため、アプリケーションをクローズする必要のある場合 (例えば、WM_CLOSE メッセージを受け取った場合)、アプリケーションは、WinAPPCCleanup コールを発行する必要があります。この関数を発行しないと、システムは不確定な状態になります。ただし、APPC がアプリケーションが終了したことを検出すると、可能な限りのクリーンアップが行われます。

構成情報

システム管理者によってセットアップと保守が行われる Communications Server 構成ファイルには、TP 同士が通信するために必要な情報が入っています。 構成に関する追加情報については、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。

呼び出し先 TP

呼び出し先 TP を作成する前に、システム管理者と共にローカル TP 名を調整する必要があります。 この名前には、最大 64 文字まで使用することができます。

AIX, LINUX

RECEIVE_ALLOCATE verb の拡張形 (着呼会話要求を受け入れるローカル LU をアプリケーションから指定できます) を使用する場合は、システム管理者とローカル LU 別名 (ローカル LU をローカル TP に認識させるための名前) についても調整する必要があります。この別名には、最大 8 文字まで使用することができます。

詳細については、 73 ページの『第 3 章 APPC 制御 verb』で RECEIVE_ALLOCATE verb の項を参照してください

呼び出し元 TP

次のリストは、呼び出し元 TP を作成する前にシステム管理者から取得する (またはシステム管理者と調整する) 必要がある情報の要約です。

ローカル LU 別名

ローカル LU をローカル TP に認識させるための名前。この名前には、最

大 8 文字まで使用することができます。詳細については、 73 ページの『第 3 章 APPC 制御 verb』で TP_STARTED verb の説明を参照してください。

パートナー TP 名
この名前には、最大 64 文字まで使用することができます。詳細については、 105 ページの『第 4 章 APPC 会話 verb』で [MC_]ALLOCATE verb の説明を参照してください。

パートナー LU 別名
パートナー LU をローカル TP に認識させるための名前です。この名前には、最大 8 文字まで使用することができます。詳細については、 105 ページの『第 4 章 APPC 会話 verb』で [MC_]ALLOCATE verb の説明を参照してください。

モード名
LU 間セッションで使用する一連の特性です。この名前には、最大 8 文字まで使用することができます。詳細については、 105 ページの『第 4 章 APPC 会話 verb』で [MC_]ALLOCATE verb の説明を参照してください。

会話セキュリティー
会話セキュリティーを使用する場合は、呼び出し先 TP にアクセスするためにユーザー ID とパスワードの有効な組み合わせが必要です。ユーザー ID とパスワードには、最大 10 文字まで使用することができます。どちらのパラメーターも大文字小文字の区別があり、システムは大文字と小文字を区別します。セキュリティー情報は、セキュリティー・ファイルに格納されます。詳細については、『会話セキュリティーの概要』を参照してください。

会話セキュリティーの概要

会話セキュリティーを使用すると、呼び出し元 TP がユーザー ID とパスワードを提供しなければ、APPC が呼び出し先 TP との会話を割り振ることができないようになります。

呼び出し先 TP の構成時に、システム管理者は会話セキュリティーを使用するかどうかを指定できます。対話セキュリティーを使用した場合、呼び出し元 TP は [MC_]ALLOCATE verb のパラメーターとして、*user_id* と *password* の組み合わせを提供する必要があります。 これらのパラメーターは、構成時に指定された *user_id* パラメーターと *password* パラメーターの組み合わせの 1 つに一致しなければなりません。

呼び出し先 TP が別の TP を呼び出すという特殊な場合があります (1 ページの『第 1 章 概念』を参照)。 TP A が、セキュリティー情報を必要とする TP B を呼び出し、 TP B は、やはりセキュリティー情報を必要とする TP C を呼び出すと仮定します。 TP B は、[MC_]ALLOCATE verb を通じて、会話セキュリティーがすでに検証済みであることを指定できます。その場合、APPC は TP A が TP B に提供したユーザー ID を取り、そのユーザー ID を TP C に「検査済み」のインディケーターを付けて送信します。このため、TP C はパスワードを検査する必要がありません。

AIX, LINUX

TP が別の TP から呼び出されたわけではなく、適切なセキュリティー情報を別の手段 (たとえば、ユーザーがログオンの手順の中でユーザー ID とパスワードを入力するなど) によって入手し検証した場合には、TP から「検査済み」のセキュリティーを指示しなければならないこともあります。 Communications Server は、これを次のようにサポートします。

- 「検査済み」を指定している TP が、ユーザー ID とパスワードを指定した別の TP から呼び出された場合、APPC はそのユーザー ID を送信します。
- それ以外の場合は、APPC は、TP の実行時の AIX / Linux のユーザー名を受け取り、必要であればそれを 10 文字に切り捨て、会話セキュリティーのユーザー ID として使用します。その名前が、有効な AE ストリング文字で構成され、呼び出し先 TP にとって有効なユーザー名であるようにしてください。
- アプリケーションが別の方法を使用してセキュリティー情報を取得する場合 (例えば、アプリケーションがユーザーに対して、AIX / Linux のシステム・セキュリティー依存せずに、ユーザー ID とパスワードを明示的に指定するよう要求する場合)、アプリケーションは、SET_TP_PROPERTIES verb を使用して、この *user_id* を APPC に指定してから [MC_]ALLOCATE verb を発行することができます。

Communications Server は、LU 間セッション・セキュリティーもサポートしています。このセキュリティーでは、ローカルとリモートの APPC LU 間でセッションを開始するときに、セキュリティー検査を行います。LU 間セッション・セキュリティーは構成時に指定され、 APPC プログラムの中ではアクションを必要としません。 詳細については、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。

TP の開始

会話は、呼び出し元 TP と呼び出し先 TP の間で発生します。この項では、呼び出し元 TP と呼び出し先 TP の開始方法について説明します。

呼び出し元 TP

呼び出し元 TP は、ユーザーがコマンドを入力することによって開始されるか、シェル・スクリプトによって開始されるか、あるいはバッチ・ファイル・コマンドによって開始されます。

呼び出し先 TP

呼び出し先 TP は、ユーザーによって開始されるか、Communications Server によって自動的に開始されるか、TP サーバー・アプリケーションによって自動的に開始されます。 システム管理者は、個々の呼び出し先 TP を構成するときに、その TP が自動で開始されるか、ユーザーによって開始されるかを指定する必要があります。

呼び出し先 TP: ユーザー開始

呼び出し先 TP をユーザーが開始するように構成した場合、ユーザーは呼び出し元 TP の前と後のどちらでも呼び出し先 TP を開始できます。 この方法で開始された TP は、待機オペレーター開始 TP と呼ばれます。

- ユーザーが呼び出し元 TP を先に開始し、TP を開始するためのタイムアウト値 (66 ページの『呼び出し先 TP のタイムアウト値』を参照) より前に呼び出し先 TP を開始しなかった場合、着呼 Allocate は失敗します。
- ユーザーが、呼び出し元 TP から [MC_]ALLOCATE verb が発行される前に呼び出し先 TP を開始した場合、呼び出し先 TP は、呼び出し元 TP から Attach が到着するか RECEIVE_ALLOCATE タイムアウト値 (66 ページの『呼び出し先 TP のタイムアウト値』を参照) に到達するまで待ちます。

呼び出し先 TP: Communications Server 接続マネージャーによる自動開始

呼び出し先 TP は、次のいずれかの条件下で自動的に開始されるように構成できます。

- その呼び出し先 TP にサービスする LU が初めて Attach (割り振り要求が入っているリモート LU からの SNA メッセージ) を受信したとき。 この方法で開始された TP は、待機自動開始 TP と呼ばれます。

  呼び出し先 TP が実行されていない場合、最初の着呼 Allocate によって呼び出し先 TP が開始されます。着呼 Allocate への応答は、呼び出し先 TP 内で RECEIVE_ALLOCATE verb が実行されるまで (またはタイムアウトが発生するまで。 66 ページの『呼び出し先 TP のタイムアウト値』を参照) 保持されます。 RECEIVE_ALLOCATE verb が実行された時点で、APPC は会話 ID を割り当て、その会話 ID が両方の TP へ会話の ID として戻されます。

  呼び出し先 TP がすでに実行されている場合、Attach は呼び出し先 TP が別の RECEIVE_ALLOCATE verb を発行するまで、または呼び出し先 TP が実行を終了して再始動が可能になるまで (またはタイムアウトが発生するまで。 66 ページの『呼び出し先 TP のタイムアウト値』を参照) 、キューに入ります。
- その呼び出し先 TP にサービスする LU が Attach を受信するたび。 Attach が着呼するたびにプログラムの新規インスタンスがロードされ、開始されます。 この方法で開始された TP は、非待機自動開始 TP と呼ばれます。

  Attach は、RECEIVE_ALLOCATE verb が呼び出し先 TP の中で実行されるまで (またはタイムアウトが発生するまで。 66 ページの『呼び出し先 TP のタイムアウト値』を参照) キューの中に置かれます。 RECEIVE_ALLOCATE が実行された時点で、APPC は会話 ID を割り当て、その会話 ID が両方の TP へ会話の ID として戻されます。

  会話終了後、呼び出し先 TP は終了するか、または別の RECEIVE_ALLOCATE を発行します。 そのため、頻繁に使用されるプログラムでは、会話ごとにプログラムの新規インスタンスを開始することによるパフォーマンスのオーバーヘッドを回避できます。 Communications Server は、自動開始された非待機 TP の Attach を受信するごとに、この TP のインスタンスに未解決の RECEIVE_ALLOCATE があるかどうかを検査します。 未解決の

RECEIVE_ALLOCATE がある場合、この TP が着呼会話に対して使用されます。未解決の RECEIVE_ALLOCATE がない場合、Communications Server はプログラムの新規インスタンスを開始します。

呼び出し先 TP: TP サーバー・アプリケーションによる自動開始

AIX, LINUX

Attach が Communications Server ノードに到着すると、Communications Server は Attach の受信登録を済ませた TP サーバー・アプリケーションに Attach を配布します。Communications Server が Attach を適切な TP サーバーへ経路指定するために使用するプロセスは、次の手順からなっています。

1. 1 つ以上のアプリケーションが、LU 名および TP 名への Attach の受信登録をします。 TP サーバー・アプリケーションは、ワイルドカードを使用して、その TP サーバーが受信登録する Attach のスコープを指定することもできます。 TP サーバー・アプリケーションは、次のいずれにもワイルドカードを使用できます。
   - ローカル LU 別名
   - TP 名
   - 完全修飾パートナー LU 名です。これは、次のいずれにもワイルドカードを使用できます。
     - ネットワーク名の一部分
     - ネットワーク名全体
     - ネットワーク名の一部分に CP 名が続いたもの
     - 完全修飾パートナー LU 名

   ワイルドカードも含め、TP と LU の 1 つの組み合わせについて登録できる TP サーバー・アプリケーションは 1 つのみです。たとえば、1 つの TP サーバー・アプリケーションで TPNAME1 および * を登録でき、同じ TP サーバー・アプリケーションまたは別の TP サーバー・アプリケーションで TPNAME1 および LUNAME1 を登録します。このタイプの登録は有効ですが、2 つの TP サーバー・アプリケーションが両方とも TPNAME1 および LUNAME1 を登録することはできません。 2 番目の登録の試みは失敗します。
2. Attach が Communications Server に到着すると、Communications Server はその Attach で受信した TP 名、LU 別名、完全修飾 LU 名に登録が最もよく一致する TP サーバー・アプリケーションを検出しようとします。 最も近いものがどれか、次の順序で一致するかどうか検査されます。
   a. TP 名の一致
   b. LU 別名の一致
   c. 完全修飾パートナー LU 名の厳密な一致
   d. 完全修飾パートナー LU 名のワイルドカードでの一致

   Communications Server は、一致するものを検出した場合、Attach をその TP サーバー・アプリケーションに送達します。TP サーバーには、次のオプションがあります。

- Attach をリジェクトする。この場合、Communications Server は Attach を呼び出し元 TP に戻し、TP サーバー・アプリケーションが提供したエラー・コードを組み込みます。
- Attach を受け入れる。この場合、Communications Server は Attach が受け入れられたことを呼び出し元 TP に通知します。

3. すべての組み合わせについて上記の検索基準を試みた後、一致するものがなかった場合、Communications Server はその Attach をリジェクトし、該当するエラー・コードを付けて呼び出し元 TP に戻します。

呼び出し先 TP のタイムアウト値

Communications Server 構成は、2 つの TP 間に会話を確立するために APPC がどれくらい待機するかを定義する次の 2 つのタイムアウト値を指定します。

開始側 TP のタイムアウト
この値は、呼び出し先 TP が開始されて RECEIVE_ALLOCATE verb を発行するのを待つために Attach をキューに入れておく時間の長さを定義します。この時間内に RECEIVE_ALLOCATE が発行されなかった場合、呼び出し元 TP 内の [MC_]ALLOCATE verb は失敗します。このタイムアウトは、呼び出し元 TP が使用するローカル LU の構成の中で定義されます。

サービス側 TP のタイムアウト
この値は、呼び出し先 TP が発行した RECEIVE_ALLOCATE verb が、呼び出し元 TP からの Attach を待つ時間の長さを定義します。この時間内に Attach を受信しなかった場合、呼び出し先 TP 内の RECEIVE_ALLOCATE verb は失敗します。構成では、次のいずれかを指定できます。

無限タイムアウト
RECEIVE_ALLOCATE は無限に待機します。

ゼロ・タイムアウト
RECEIVE_ALLOCATE は、Attach がすでに受信されている場合以外、失敗します。

有限タイムアウト
特定のタイムアウト値が提供されます。

このタイムアウトは、呼び出し先 TP 用に Communications Server 呼び出し可能 TP データ・ファイルの中で定義されます。

起動される TP の構成についての詳細は、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。

LU 間セッション

LU 間セッションは、2 つの LU の間の論理接続です。 TP 間の会話は、セッションの中で発生します。 1 つの会話は一度に 1 つのセッションを使用できます。多数の会話が同じセッションを順次に再利用することもできます。

Communications Server では、LU タイプ 6.2 で複数セッション (異なるパートナー LU を使用した 2 つ以上の並行セッション) と並列セッション (同じパートナー LU を使用した 2 つ以上の並行セッション) を使用できます。

構成のとき、システム管理者は特定の LU がサポートするセッションの数と、その LU に並列セッションをサポートさせるかどうかを決定します。

コンテンション

両方の LU が同時に同じセッション上に会話を割り振ろうとした場合、一方が勝ち (コンテンション勝者)、一方が負ける (コンテンション敗者) 必要があります。 コンテンション勝者の LU とコンテンション敗者の LU は、セッションが確立されたときに決定されます。

あるセッションの中で、コンテンション敗者の LU は会話を割り振る前にコンテンション勝者の LU の許可を求める必要があります。コンテンション勝者は、許可を与えても与えなくても構いません。その一方、コンテンション勝者の LU は会話のみは必要なときに割り振ります。

構成のとき、システム管理者はモードを定義できます。モードとは、ネットワーク機能の一連の特性のことです。システム管理者がモード定義の中で指定できる特性の中に、そのモードを使用するローカル LU とパートナー LU のコンテンション勝者およびコンテンション敗者セッション数があります。 ([MC_]ALLOCATE verb を発行する TP は、モード、ローカル LU、パートナー LU を指定します。)

基本会話

一般に、基本会話はサービス TP で使用します。サービス TP は、他のローカル・プログラムにサービスを提供するプログラムです。基本会話はマップ式会話より複雑ですが、熟練した LU6.2 プログラマーであれば、基本会話を使用してデータの伝送と処理の制御を強化、充実することができます。この項では、マップ式会話と異なる基本会話の特性を要約して説明します。

論理レコード

基本会話では、データは論理レコードの形で送信されます。論理レコードとは、この項で説明する汎用データ・ストリーム (GDS) 構文を備えたレコードのことです。GDS 構文についての詳細は、IBM 資料の「*SNA フォーマット*」を参照してください。

送信側 TP はデータを論理レコードの形式に設定しなければならず、受信側 TP は論理レコードを使用可能なデータにデコードする必要があります。 TP は 1 つの SEND_DATA verb で複数の論理レコードを送信したり、単一の論理レコードを複数の部分 (セグメントと呼ばれる) に分けて複数の SEND_DATA verb で送信できます。 TP は 1 つの受信 verb (RECEIVE_AND_WAIT、RECEIVE_IMMEDIATE、RECEIVE_AND_POST のいずれか) で複数の論理レコードを受信したり、複数の部分に分かれた単一の論理レコードを複数の受信 verb で受信できます。

論理レコードは、単一レコードである場合には、次のフィールドからなっています。

- 2 バイトのレコード長 (LL) フィールド

- 2 バイトの GDS 識別子 (ID) フィールド (たとえば、 0x12FF はデータがアプリケーション・データであることを示します)
- 長さが 0 ～ 32,763 バイトの範囲のデータ・フィールド

最初の 4 バイトは、LLID と呼ばれます。

1 つの論理レコードが複数の部分からなる場合、最初の部分は単一レコードと同じ形式であり、それ以後のすべての部分は次のフィールドからなっています。

- 2 バイトのレコード長 (LL) フィールド
- 長さが 0 ～ 32,765 バイトの範囲のデータ・フィールド

LL フィールドの 16 進値には、LL フィールド自体の 2 バイトが (また、ID フィールドが存在する場合はその 2 バイトも) 含まれます。 たとえば、単一部分からなる GDS でデータがゼロ・バイトのものは、 LL フィールドの値が 0x0004 になります。 LL フィールドは、バイト・スワップ形式でなくハイ・ローの形式でなければなりません。 たとえば、230 バイトという長さは、0xE600 でなく 0x00E6 として表されます。

LL のバイト 0 のビット 0 (最上位ビット) は、長さの継続 (セグメント化) を示すために使用されます。次の例は、10 バイトのデータ (データの個々のバイトの値は DD) を 3 つの GDS セグメントに分割した様子を示しています。 最初と 2 番目のセグメントには、4 バイトずつデータが入っており、最後のセグメントには 2 バイトのデータが入っています。

```
8008 12FF DDDD DDDD
8006 DDDD DDDD
0004 DDDD
```

次の値は、LL フィールドには無効です (『論理レコードでの PS ヘッダーの送信』で説明するように、PS ヘッダーを送信する場合は除きます)。

- 0x0000
- 0x0001
- 0x8000
- 0x8001

マップ式会話では、送信側 TP は一度に 1 つのデータ・レコードを送信し、受信側 TP は一度に 1 つのデータ・レコードを受信します。 TP でレコードを変換する必要はありません。

論理レコードでの PS ヘッダーの送信

会話の同期レベルが AP_SYNCPT の場合、アプリケーションがデータを PS ヘッダーの形で送受信しなければならないときがあります。 同期点管理プログラム は、アプリケーションが必要とする同期点機能に基づいて、パートナー・アプリケーションへ送信する適切な PS ヘッダーをセットアップする責任を負うほか、パートナー・アプリケーションから受信した PS ヘッダーに基づいて、必要な同期点処理を実行します。

0x0001 という LL フィールドは、データが PS ヘッダーであることを示しています。送信側アプリケーションは、データ・フィールドの長さを指定する代わりに

0x0001 という LL フィールドを指定する必要があり、受信側アプリケーションは、0x0001 という LL フィールドを受信した場合、そのデータを PS ヘッダーとして解釈する必要があります。 会話の同期レベルが AP_SYNCPT でない場合、 0x0001 という値は有効な LL フィールドではないため、リジェクトされます。

エラーおよび異常終了の報告

基本会話では、TP はエラーまたは異常終了 (プログラムの異常終了) が起きた原因が、サービス TP であるのか、サービス TP を使用するプログラムであるのかを示すことができます。 これにより、通信中の 2 つのサービス TP で、それら自身が原因となって起きたエラーと、それらがサービスするプログラムが原因となって起きた可能性があるエラーを区別できます。

エラー・ログ

基本会話でエラーまたは異常終了が起きた場合、TP は汎用データ・ストリーム (GDS) エラー・ログ変数の形でローカル・ログとパートナー LU にエラー・メッセージを送信できます。

タイムアウトとクリティカル・エラー

基本会話では、TP は異常終了が起きた原因が、タイムアウトであるのか、クリティカル・エラーであるのかを示すことができます。

TP サーバーの作成

AIX, LINUX

TP サーバーを作成するには、次の操作ガイドラインを使用してください。

1. TP サーバー verb は、同期エントリー・ポイント APPC ではなく、非同期エントリー・ポイント APPC_Async を使用して発行される必要があります。
2. アプリケーションを TP サーバーとして登録するには、REGISTER_TP_SERVER を使用します。 REGISTER_TP_SERVER verb は、あとで Attach の通知に使用するコールバック関数のアドレスを提供します。
3. その TP サーバーで Attach を処理する TP 名とローカル LU およびリモート LU を登録するには、REGISTER_TP を使用します。 TP サーバーは、TP 名と LU 名の両方にワイルドカードを使用できるため、ある TP から特定の LU の対に対して Attach を処理することも、すべての LU に対してすべての TP 用に Attach を処理することも、それらの条件の任意の組み合わせについて処理することもできます。
4. Attach パラメーターに照会して Attach の処理を行うかどうかとその方法を判別するには、QUERY_ATTACH (と、登録済み TP または LU 用の Attach の受信後に通知コールバックを行って受信した固有の ID) を使用します。 TP サーバーは、REJECT_ATTACH を使用して Attach をリジェクトするか、ACCEPT_ATTACH を使用して Attach を受け入れ、Attach の処理に適したアプリケーションを開始できます。

5. Attach を取り出すには、標準 RECEIVE_ALLOCATE コールを発行します。以前予約されている *dload_id* パラメーターは、Attach の固有 ID を指定するために使用されます。
6. ACCEPT_ATTACH を発行した後にエラーを検出した場合、それ以上の処理を取り消すには、ABORT_ATTACH を使用します。
7. 前に登録した TP と LU について TP サーバーの登録を解除するには、UNREGISTER_TP を使用します。
8. アプリケーションの TP サーバーとしての登録を解除するには、UNREGISTER_TP_SERVER verb を使用します。

TP サーバーの役割

TP サーバーは、Attach を処理する場合、通常であれば Communications Server Attach Manager が負ういくつかの役割を継承します。 それらの役割は次のとおりです。

- TP が自動的に開始されるように構成されている場合、会話を処理するために TP を開始する
- QUERY_ATTACH で戻された Attach データの解析を含む会話セキュリティーの処理
- それ以後の会話を処理するために TP サーバーが開始されたという情報を Attach から TP へ伝達する

デフォルト TP サーバー

Communications Server は、すべてのシステムにインストールされるデフォルト TP サーバーである **snatpsrvd** を提供します。 この TP サーバーは、ロードできる TP を構成するソースとして、 **sna_tps** ファイルを使用します。 その他の TP サーバーは、DEFINE_TP_LOAD_INFO verb を使用することにより、このファイルを変更して使用できます。 DEFINE_TP_LOAD_INFO verb についての詳細は、「*IBM Communications Server for Linux NOF プログラマーズ・ガイド*」または「*IBM Communications Server for AIX NOF プログラマーズ・ガイド*」を参照してください。APPC は REGISTER_TP verb の *tp_file_updates* パラメーターを提供し、それによって TP サーバーは **sna_tps** ファイルに変更が加えられた時点で通知を受け取り、必要なアクションを実行できます。

移植可能な TP の作成

次のガイドラインは、他のオペレーティング・システム環境に移植できる TP を作成するためのものです。

- APPC ヘッダー・ファイルを、パス名の接頭部を付けずにインクルードします。ファイルを見つけるには、コンパイラーでインクルード・オプションを使用します (この章の前の方で述べている、ご使用のオペレーティング・システムの該当するセクションを参照してください)。これにより、その TP をファイル・システムが異なる環境で使用できるようになります。

- パラメーター値と戻りコードには、ヘッダー・ファイルに示された数値でなく、記号定数名を使用します。これにより、メモリー内での格納方法に関係なく、正しい値が使用されます。
- 現在ご使用のオペレーティング・システムに適用できるもの以外の戻りコードの検査を組み込み (例えば、switch ステートメントで「デフォルトの」ケースを使用する)、適切な診断を行います。
- 非同期のエントリー・ポイントを使用します。
- TP を完全に移植可能にする場合は、 [MC_]RECEIVE_AND_POST verb を使用できません。 この verb を使用した場合は、TP のセクションを他の環境で使用するために書き直す必要が生じます。 変更を容易にするため、この verb の使用を少数の特殊なルーチンのみに制限しても構いません。
- CANCEL_CONVERSATION verb、および RECEIVE_ALLOCATE verb の拡張形は、Communications Server にインプリメントされた APPC に特有のものであり、他のインプリメンテーション形態の APPC では提供されない場合があります。CANCEL_CONVERSATION または RECEIVE_ALLOCATE の拡張形を使用する場合は、TP のセクションを他の環境で使用するために書き直す必要があります。 変更を容易にするため、これらの機能の使用を少数の特殊なルーチンのみに制限しても構いません。

第 3 章 APPC 制御 verb

この章には、個々の APPC 制御 verb の説明が記載されています。それぞれの verb について、次の情報を提供します。

- 各 verb の定義。
- 各 verb が使用する verb 制御ブロック (VCB) の構造体定義。この構造体は、APPC ヘッダー・ファイル **/usr/include/sna/appc_c.h** (AIX)、**/opt/ibm/sna/include/appc_c.h** (Linux)、**sdk/winappc.h** (Windows) に定義されています。 *reserv* で始まるパラメーターは予約されています。
- APPC へ提供するか APPC から戻されるパラメーター (VCB フィールド)。各パラメーターについて、次の情報を提供します。
  - 説明
  - 指定可能な値
  - 追加情報
- 各 verb を発行できる会話状態。
- 各 verb から戻ったときの、変更された会話状態。状態の変更を引き起こさない条件については、特に言及しません。 たとえば、パラメーター検査と状態検査では、状態の変更は起こりません。
- 各 verb の使用法を説明した追加情報。

APPC へ提供するパラメーターと APPC から戻されるパラメーターのほとんどは、16 進値です。コーディングを単純にするため、これらの値は、ヘッダー・ファイル **values_c.h** の中で定義されている、意味のある記号定数によって表されます。このヘッダー・ファイルは、APPC ヘッダー・ファイル **appc_c.h** によってインクルードされます。たとえば、TP_STARTED verb の *opcode* パラメーターは、記号定数 `AP_TP_STARTED` によって表される 16 進値です。 ファイル **values_c.h** は、 APPC VCB で使用される `AP_UINT16` などのパラメーター・タイプの定義もインクルードします。

指定されたパラメーターに値を設定する場合、または戻されたパラメーターの値をテストする場合には、16 進値ではなく記号定数を使用することが重要です。これは、さまざまなオペレーティング・システムがさまざまな方法でそれらの値をメモリー内に格納するため、示されている値が使用しているシステムで認識される形式になっていない場合もあるためです。

WINDOWS

Windows の場合、指定パラメーターと戻りパラメーター値の定数は、 Windows APPC ヘッダー・ファイル **winappc.h** に定義されています。

「[MC_]verb」という表記は、APPC verb のマップ式対話と基本対話の両方の形を表しています。 たとえば、[MC_]SEND_DATA は、MC_SEND_DATA と SEND_DATA という両方の verb を表しています。

注: APPC VCB には、「予約済み」とマークされたパラメーターが多数含まれています。予約済みパラメーターには、Communications Server ソフトウェアで内部的に使用されているものや、このバージョンでは使用されていなくても将来のバージョンで使用される可能性があるものがあります。アプリケーションでは、予約済みパラメーターにアクセスしてはなりません。その代わりに、verb で使用する他のパラメーターを設定する前に、VCB の全内容をゼロに設定し、これらのパラメーターすべてをゼロにしてください。このようにすると、内部的に使用されるパラメーターが Communications Server で誤って解釈されることがなく、将来のバージョンでこれらのパラメーターが新しい関数を提供するために使用されても、アプリケーションは引き続き動作します。

VCB の内容をゼロに設定するには、`memset` を使用します。

```
memset(vcb, 0, sizeof(vcb));
```

ここでは、制御 verb を次の順序で説明します。

TP_STARTED
TP_ENDED
RECEIVE_ALLOCATE

AIX, LINUX

GET_LU_STATUS

GET_TP_PROPERTIES

AIX, LINUX

SET_TP_PROPERTIES

TP_STARTED

TP_STARTED verb は、呼び出し元 TP が発行します。この verb は TP が開始されようとしていることを APPC に通知し、その TP が使用するローカル LU を指定します。

TP は、複数の並行した会話に従属 LU を使用する場合、会話ごとに別々の TP_STARTED verb を (その後に [MC_]ALLOCATE を続けて) 発行し、会話ごとに異なる LU を取得する必要があります。これは、1 つの従属 LU は一度に 1 つの会話のみをサポートするためです。

この verb への応答で、APPC は呼び出し元 TP の ID を生成します。この ID は、それ以後に呼び出し元 TP から発行される APPC verb の必須パラメーターです。

VCB 構造体: TP_STARTED

AIX, LINUX

TP_STARTED verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct tp_started
{
  AP_UINT16      opcode;
  unsigned char  opext;                 /* Reserved             */
  unsigned char  format;                /* Reserved             */
  AP_UINT16      primary_rc;
  AP_UINT32      secondary_rc;
  unsigned char  lu_alias[8];
  unsigned char  tp_id[8];
  unsigned char  tp_name[64];
  unsigned char  delay_start;           /* Reserved             */
  unsigned char  enable_pool;           /* Reserved             */
  unsigned char  pip_dlen;              /* Reserved             */
} TP_STARTED;
```

VCB 構造体: TP_STARTED (Windows)

WINDOWS

TP_STARTED verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct tp_started
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     lu_alias[8];
  unsigned char     tp_id[8];
  unsigned char     tp_name[64];
} TP_STARTED;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_TP_STARTED

lu_alias

ローカル LU をローカル TP に認識させるための別名。この名前は、構成時に指定された LU 別名の 1 つに一致する必要があります。

このパラメーターは、8 バイトの ASCII 文字ストリングです。 これは、次の文字から構成できます。

- 英大文字
- 数字の 0 ～ 9
- ブランク
- 特殊文字の $、#、%、および @

このストリングの 1 文字目をブランクにすることはできません (ストリング全体がブランクで構成される場合は除きます)。

LU 別名が 8 文字より短い場合は、右側に ASCII のブランク (0x20) を埋め込んでください。

構成によっては、デフォルトのローカル LU (システム管理者と一緒に検査してください) をアプリケーションで使用することを指定しても構いません。これを行うには、*lu_alias* を 8 つの 2 進ゼロからなるストリングに設定します。 他の APPC インプリメンテーションとの互換性を保つため、Communications Server はデフォルト LU を示す 8 つの ASCII ブランクからなるストリングも受け入れます。ただし、新しいアプリケーションでは 2 進ゼロを使用してください。

tp_name

ローカル TP の名前。この名前の最初の 8 文字は ASCII に変換され、実行中の APPC TP のリスト内でその TP を識別するために Communications Server 管理プログラムによって使用されます。

このパラメーターは、大文字小文字の区別がある 64 バイトの EBCDIC 文字ストリングです。通常、*tp_name* パラメーターはタイプ AE の EBCDIC 文字セットの文字からなっています (サービス TP の名前である場合は除きます)。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 ～ 9
- 特殊文字の $、#、?、およびピリオド (.)

TP 名が 64 バイト未満の場合は、EBCDIC のブランク (0x40) を右側に埋め込みます。

サービス TP の命名に関する SNA 規則は、通常の *tp_name* パラメーターの例外です。この名前は最大 4 文字からなり、1 文字目は 0x00 ～ 0x3F の 16 進バイトです。 その他の文字は EBCDIC AE 文字セットからのものです。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

tp_id ローカル TP の ID。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_LU_ALIAS
lu_alias パラメーターの値が無効でした。

AIX, LINUX

AP_INVALID_FORMAT
予約済みパラメーター *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。 コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: この verb で状態検査エラーは発生しません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_COMM_SUBSYSTEM_NOT_LOADED
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP_STARTED は、呼び出し元 TP が最初に発行する APPC verb である必要があります。したがって、活動状態である会話は存在せず、会話状態も存在しません。

1 つの APPC プログラムから複数の TP_STARTED verb を発行することもできます。 1 つの verb につき 1 つの論理的に異なる APPC TP が作成されます。ただし、それらはすべて同じプロセスの中で実行されます。

状態の変更

適用外 (会話が開始されていないため、会話状態は存在しません)。

TP_ENDED

TP_ENDED verb は、呼び出し元 TP と呼び出し先 TP の両方が発行します。この verb は、TP が終了しようとしていることを APPC に通知します。 この verb への応答で、APPC は TP が使用したリソースを解放します。

APPC 会話がまだ進行中の場合、TP_ENDED は、[MC_]DEALLOCATE verb で *dealloc_type* を AP_ABEND (マップ式会話の場合) または AP_ABEND_PROG (基本会話の場合) に設定した機能を実行します。この verb が実行された後、TP ID と会話 ID は無効になり、 TP はその会話についてそれ以上 APPC verb を発行できません。

VCB 構造体: TP_ENDED

AIX, LINUX

TP_ENDED verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct tp_ended
{
  AP_UINT16         opcode;
  unsigned char     opext;                /* Reserved      */
  unsigned char     format;               /* Reserved      */
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  unsigned char     type;
} TP_ENDED;
```

VCB 構造体: TP_ENDED (Windows)

WINDOWS

TP_ENDED verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct tp_ended
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned char     type;
} TP_ENDED;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_TP_ENDED

tp_id ローカル TP の ID。

このパラメーターの値は、TP_STARTED verb が呼び出し元 TP について戻した値か、RECEIVE_ALLOCATE verb が呼び出し先 TP について戻した値です。

type TP の終了方法を指定します。次の値があります。

AP_SOFT
活動状態である APPC 会話が存在する場合、APPC は個々の会話について、パートナー TP に会話が終了したことを通知するため、[MC_]DEALLOCATE verb の機能を実行します。 TP_ENDED verb は、[MC_]DEALLOCATE が完了するまで戻りません。

AP_HARD
APPC は、その TP が使用したすべてのセッションをクローズし、TP_ENDED は即時に戻ります。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

この verb が正常に実行された場合、APPC は *secondary_rc* を戻しません。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_TP_ID
APPC は *tp_id* を、割り当てられた TP ID として認識しませんでした。

AP_BAD_TYPE
type パラメーターの値が無効でした。

AIX, LINUX

AP_INVALID_FORMAT
予約済みパラメーター *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: この verb で状態検査エラーは発生しません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_COMM_SUBSYSTEM_NOT_LOADED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話 (TP が複数の会話に関与している場合は複数の会話) はどのような状態にあっても構いません。

状態の変更

正常に実行 (*primary_rc* が AP_OK) された後、 APPC 状態は存在しません。

RECEIVE_ALLOCATE

RECEIVE_ALLOCATE verb は、呼び出し先 TP が発行します。この verb は、呼び出し先 TP が、[MC_]ALLOCATE verb を発行した呼び出し元 TP との会話を開始できる状態であることを確認します。

この verb への応答で、APPC は 2 つの TP 間に会話を確立し、呼び出し先 TP の TP ID を生成し、会話 ID を生成します。 これらの ID は、それ以後の APPC verb の必須パラメーターです。

AIX, LINUX

Communications Server にインプリメンテーションされた APPC は、他のインプリメンテーション形態の APPC が提供する標準形の RECEIVE_ALLOCATE verb と、アプリケーションが特定のローカル LU からの着呼 Attach を受信できるようにする拡張形の両方を提供します。 この項では、これら 2 つの形をまとめて説明し、必要な箇所では「標準形」および「拡張形」として区別します。

WINDOWS

RECEIVE_ALLOCATE の拡張形式は Windows には提供されていません。

VCB 構造体: RECEIVE_ALLOCATE

AIX, LINUX

RECEIVE_ALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct receive_allocate
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reserved      */
  unsigned char     format;
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_name[64];
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     sync_level;
  unsigned char     conv_type;
  unsigned char     user_id[10];
  unsigned char     lu_alias[8];
  unsigned char     plu_alias[8];
  unsigned char     mode_name[8];
  unsigned char     reserv3[2];
  AP_UINT32         conv_group_id;
  unsigned char     fqplu_name[17];
  unsigned char     pip_incoming;
  unsigned char     duplex_type;
  unsigned char     reserv4[3];
  unsigned char     password[10];
  unsigned char     reserv5[2];
  unsigned char     dload_id[8];
} RECEIVE_ALLOCATE;
```

VCB 構造体: RECEIVE_ALLOCATE (Windows)

WINDOWS

RECEIVE_ALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct receive_allocate
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_name[64];
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     sync_level;
  unsigned char     conv_type;
  unsigned char     user_id[10];
  unsigned char     lu_alias[8];
  unsigned char     plu_alias[8];
  unsigned char     mode_name[8];
  unsigned char     reserv3[2];
  unsigned long     conv_group_id;
  unsigned char     fqplu_name[17];
  unsigned char     reserv4[5];
} RECEIVE_ALLOCATE;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_RECEIVE_ALLOCATE (標準形)

AIX, LINUX

AP_RECEIVE_ALLOCATE_EX (拡張形)

tp_name

ローカル TP の名前。 APPC は、この名前を、着呼 Attach の中で指定された TP 名 (これは、呼び出し元 TP 内の [MC_]ALLOCATE verb によって生成されたものです) と突き合わせます。 この TP を Communications Server に自動的に開始させる場合、この TP 名は呼び出し可能 TP データ・ファイル内で指定されている TP 名に一致しなければなりません。

このパラメーターは、大文字小文字の区別がある 64 バイトの EBCDIC 文字ストリングです。通常、*tp_name* パラメーターはタイプ AE の EBCDIC 文字セットの文字からなっています。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 〜 9
- 特殊文字の $、#、?、およびピリオド (.)

TP 名が 64 バイト未満の場合は、EBCDIC のブランク (0x40) を右側に埋め込みます。

サービス TP の命名に関する SNA 規則は、上記の例外です。この名前は最大 4 文字からなり、1 文字目は 0x00 〜 0x3F の 16 進バイトです。その他の文字は EBCDIC AE 文字セットからのものです。

AIX, LINUX

TP ですべての TP 名について着呼 Attach を受け入れることを指定するには、このパラメーターを 64 個の EBCDIC のスペースに設定します。 Communications Server が着呼 Attach を TP に転送する方法についての詳細は、 88 ページの『着呼 Attach の経路指定』を参照してください。

lu_alias

標準形の RECEIVE_ALLOCATE の場合、このパラメーターは、予約済みなのでヌル・ストリングに設定してください。この TP が Communications Server によって自動的に開始される場合、呼び出し可能 TP データ・ファイル内のこの TP 用のエントリーで LU 別名を指定してはなりません。

拡張形の RECEIVE_ALLOCATE の場合は、TP が着呼 Attach を受け入れるローカル LU の別名を指定してください。 これは ASCII 文字ストリングです。 この TP が Communications Server によって自動的に開始される場合、この LU 別名は、呼び出し可能 LU データ・ファイル内でこの TP 用に指定した LU 別名に一致しなければなりません。

この TP がすべてのローカル LU からの着呼 Attach を受け入れることを示すには、このパラメーターを 8 個の ASCII のスペースに設定します。 こ

の TP が Communications Server によって自動的に開始される場合、呼び出し可能 TP データ・ファイル内のこの TP 用のエントリーで LU 別名を指定してはなりません。

Communications Server が着呼 Attach を TP に転送する方法についての詳細は、 88 ページの『着呼 Attach の経路指定』を参照してください。

dload_id

TP サーバー・アプリケーションが提供した Attach ID。 TP が TP サーバーと連携しない場合は、このフィールドをすべてゼロに設定してください。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

AIX, LINUX

tp_name

アプリケーションでスペースだけからなる TP 名を指定した場合、Communications Server は呼び出し元 TP が [MC_]ALLOCATE verb で提供した TP 名を戻します。

tp_id ローカル TP の ID。

conv_id

会話 ID。

この値は、APPC が 2 つのパートナー TP 間に確立した会話を識別します。

sync_level

会話の同期レベル。

このパラメーターは、TP がデータの受信の確認を要求できるかどうか、またデータの受信を確認できるかどうかを決定します。次の値があります。

AP_CONFIRM_SYNC_LEVEL

パートナー TP は、この会話で確認処理を使用できます。

AP_SYNCPT

各 TP は、この会話で LU6.2 同期点機能を使用できます。 詳細については、 25 ページの『同期点サポート』を参照してください。

AP_NONE
この会話では、確認処理は使用されません。

conv_type
パートナー TP が [MC_]ALLOCATE verb を使用して選択した会話のタイプ。 次の値があります。
AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

user_id パートナー TP が [MC_]ALLOCATE の *security* パラメーターを AP_PGM または AP_SAME に設定した場合、このパラメーターにはパートナー TP から送信されたユーザー ID が入っています。 このユーザー ID はタイプ AE の EBCDIC 文字ストリングで、必要な場合は、10 文字になるよう右側に EBCDIC のスペースが埋め込まれます。 パートナー TP が [MC_]ALLOCATE の security パラメーターを AP_NONE に設定した場合、このパラメーターは 10 個の EBCDIC のブランクに設定されます。

lu_alias
ローカル LU をローカル TP に認識させるための別名。これは ASCII 文字ストリングです。

plu_alias
パートナー LU (着呼 Allocate の受信元) をローカル TP に認識させるための別名。これは ASCII 文字ストリングです。

mode_name
パートナー TP 内で [MC_]ALLOCATE verb によって指定されたモード名。これは、構成のときに定義されたネットワーク機能の特性セットの名前です。この名前は、タイプ A の EBCDIC 文字ストリングです。

conv_group_id
新規の会話が使用するセッションの会話グループ ID。

fqplu_name
パートナー LU の完全修飾名。

このパラメーターには、ネットワーク名、EBCDIC のピリオド、およびパートナー LU 名が入っています。 2 つの名前は、それぞれ 8 バイトの EBCDIC 文字ストリングで、次のようなタイプ A の EBCDIC 文字セットの文字からなっています。

- 英大文字
- 数字の 0 ～ 9
- 特殊文字の $、#、および ?

AIX, LINUX

pip_incoming
パートナー TP が [MC_]ALLOCATE 要求でプログラム初期設定パラメーター (PIP) データを提供したかどうかを示します。 次の値があります。

AP_YES パートナー TP は PIP データを提供しました。ローカル TP は、いずれかの [MC_]RECEIVE verb を発行してデータを受信する必要があり、最初に受信されるデータ・レコードは PIP データです。

AP_NO パートナー TP は PIP データを提供しませんでした。

duplex_type
新規会話の全二重タイプ。次の値があります。
AP_HALF_DUPLEX
AP_FULL_DUPLEX

password
パートナー TP が [MC_]ALLOCATE の *security* パラメーターを AP_PGM に設定した場合、このパラメーターにはパートナー TP が [MC_]ALLOCATE verb で指定したパスワードが入っています。このパスワードはタイプ AE の EBCDIC 文字ストリングで、必要な場合は、10 文字になるよう右側に EBCDIC のスペースが埋め込まれます。 パートナー TP が [MC_]ALLOCATE の *security* パラメーターを AP_NONE または AP_SAME に設定した場合、このパラメーターは 10 個の EBCDIC のブランクに設定されます。

セキュリティー上の理由から、Communications Server はパスワードを RECEIVE_ALLOCATE verb で戻した後、そのパスワードを保管しません。このパラメーターをアプリケーションで検査する必要がある場合は、RECEIVE_ALLOCATE の戻り値を使用する必要があります。それ以後の verb では、パスワードは戻されません。ユーザー ID は、会話の実行中にアプリケーションから GET_TP_PROPERTIES verb を発行することにより、いつでも取り出すことができます。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AIX, LINUX

AP_BAD_DLOAD_ID
dload_id パラメーターに指定した値が認識されませんでした。

AP_INVALID_FORMAT
format パラメーターが無効な値に設定されていました。

AP_INVALID_LU_ALIAS
lu_alias パラメーターに無効な文字が入っていました (この値は、標準形でなく拡張形の RECEIVE_ALLOCATE の場合に戻ります)。

AIX, LINUX

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc

AP_ALLOCATE_NOT_PENDING
APPC は、RECEIVE_ALLOCATE verb が提供した TP 名か LU 別名、またはその両方を突き合わせるための (呼び出し元 TP からの) 着呼 Allocate を検出しませんでした。 RECEIVE_ALLOCATE verb は着呼 Allocate を待機し、最終的にタイムアウトになりました。詳細については、 88 ページの『待機の回避』と 88 ページの『着呼 Attach の経路指定』を参照してください。

この戻りコードは、呼び出し可能 TP データ・ファイルの中で非待機 TP として定義された TP を開始しようとした場合にも発生します。 非待機 TP は、着呼 Attach への応答として Communications Server によって自動的に開始されます。非待機 TP を手動で開始しようとすると、TP を待機している着呼 Attach が存在しないため、RECEIVE_ALLOCATE verb は失敗します。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_COMM_SUBSYSTEM_NOT_LOADED
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_STACK_TOO_SMALL

AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

これは、呼び出し先 TP が最初に発行する APPC verb である必要があります。初期状態は Reset です。

単一の呼び出し先 TP から複数の RECEIVE_ALLOCATE verb を発行でき、 1 つの RECEIVE_ALLOCATE verb につき 1 つの論理的に異なる APPC TP が開始されます。ただし、それらはすべて同じプロセスの中で実行されます。

状態の変更

verb が正常に実行される (*primary_rc* が AP_OK である) と、この状態は、半二重会話では Receive に、全二重会話では Send_Receive に変わります。

待機の回避

呼び出し先 TP から RECEIVE_ALLOCATE verb を発行し、それに対応する着呼 Allocate (呼び出し元 TP から発行された [MC_]ALLOCATE verb の結果として生成されるもの) が存在しない場合、呼び出し先 TP は着呼 Allocate が到着するか、この verb がタイムアウトになるまで待機します。 デフォルトでは、着呼 Allocate を無限に待機しますが、これは、 0 (ゼロ) のタイムアウト (RECEIVE_ALLOCATE は、着呼 Allocate がすでに待機していない場合、失敗します) を使用するか、有限の値 (RECEIVE_ALLOCATE は、指定した時間内に着呼 Allocate が到着しなかった場合、失敗します) を使用して TP を構成することによって変更できます。 詳細については、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。

着呼 Attach の経路指定

AIX, LINUX

アプリケーションが RECEIVE_ALLOCATE で *dload_id* を指定しないと、RECEIVE_ALLOCATE の *tp_name* と *lu_alias* パラメーターを使用して、アプリケーションが受け入れる着呼 Attach の範囲を指定できます。TP 名を指定すると、パートナー TP がその TP 名を [MC_]ALLOCATE verb で指定した場合にのみ、パートナー TP からの着呼 Attach が受け入れられます。拡張形の RECEIVE_ALLOCATE を使用して LU 別名を指定すると、着呼 Attach が特定の Communications Server ローカル LU に到着した場合にのみ、それらの着呼 Attach が受け入れられます。 いずれの場合も、TP でブランクの名前を指定することにより、その TP がすべての TP 名についての着呼 Attach を受け入れるか、すべてのローカル LU からの着呼 Attach を受け入れることを示すことができます。

Communications Server は、着呼 Attach を次の優先順位で適切な TP 内の RECEIVE_ALLOCATE verb と突き合わせます。

1. TP 名と LU 別名を指定しており、その両方が着呼 Attach に一致する TP。
2. 着呼 Attach に一致する TP 名を指定しており、LU 別名を指定していない TP。
3. 着呼 Attach を受信した LU に一致する LU 別名を指定しており、TP 名を指定していない TP。
4. TP 名も LU 別名も指定していない TP。1 台の Communications Server コンピューター上で 1 つの TP のみがこの機能を使用するようにしてください。2 つの TP が両方とも TP 名と LU 別名を指定せずに RECEIVE_ALLOCATE verb を発行した場合、どちらの TP が着呼 Attach を受信するかを判別できません。

ある範囲の着呼 Attach を受け入れるため、TP でブランクの TP 名か LU 別名、またはその両方を使用した場合、その TP で、この verb からの戻りパラメーターを検査して、着呼 Attach 上で指定された TP 名か着呼 Attach を受信した LU の LU 別名、またはその両方を判別できます。 これは、単一の TP ですべての着呼 Attach を処理し、複数の TP 名か LU、またはその両方のそれぞれについて、適切な処理を実行できることを意味しています。 この TP は、認識できないか権限のないパートナー TP からの着呼 Attach を受信した場合、必要であれば、[MC_]DEALLOCATE verb を適切な *dealloc_type* パラメーターと共に発行することにより、新規の会話をリジェクトできます。

着呼 Attach を受け入れる TP は、すでに RECEIVE_ALLOCATE を発行済みのオペレーター開始 TP でも、Communications Server 呼び出し可能 TP データ・ファイルの中に登録された自動開始 TP でも構いません。Communications Server は、このファイルの中で指定された TP 名か LU 別名、またはその両方を使用して、着呼 Attach を突き合わせるために TP を開始するかどうかを判別します。このファイル形式についての詳細は、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。RECEIVE_ALLOCATE verb で自動的に開始される TP によって指定される TP 名か LU 別名、またはその両方は、Communications Server が着呼 Attach を正しく経路指定できるよう、このファイル内で指定されたものと一致する必要があります。

GET_LU_STATUS

AIX, LINUX

この verb は、必要な場合に再同期が取れるよう、パートナー TP との通信が失われたかどうかを検査する必要がある同期点 TP 用に提供されています。

注: 2 つ以上の TP がローカル LU とパートナー LU の同じ組み合わせを使用している場合は、1 つの TP のみがこの verb を発行することが重要です。Communications Server は LU の個々の対ごとに、それらの LU を使用する TP とは別個にゼロ・セッション・インディケーターを管理し、この verb が発行されるたびに、そのインディケーターをリセットします。したがって、セッション・カウントがいったん 0 (ゼロ) に落ちてからセッションが再度活動化され、

その後、2 つの TP が GET_LU_STATUS を発行した場合は、最初の TP のみにゼロ・セッション・カウントが通知されます。 同じ LU を使用している複数の TP が、LU とセッション状況をチェックする必要がある場合、NOF verb を使用してチェックします。詳細については、「*IBM Communications Server for Linux NOF プログラマーズ・ガイド*」または「*IBM Communications Server for AIX NOF プログラマーズ・ガイド*」を参照してください。

VCB 構造体: GET_LU_STATUS

GET_LU_STATUS verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct get_lu_status
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reserved      */
  unsigned char     format;                 /* Reserved      */
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  unsigned char     plu_alias[8];
  AP_UINT16         active_sess;
  unsigned char     zero_sess;
  unsigned char     reserv3[7];
} GET_LU_STATUS;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_GET_LU_STATUS

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

plu_alias

パートナー LU をローカル TP に認識させるための別名。これは 8 バイトの ASCII 文字ストリングです。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

active_sess

現在ローカル LU と指定したパートナー LU との間で活動状態であるセッションの数を示します。

zero_sess

前回に GET_LU_STATUS verb を発行して以来、2 つの LU 間でアクティブ・セッションの数が一度でも 0 (ゼロ) に落ちたことがあるかどうかを示します。次の値があります。

AP_YES セッション・カウントは 0 (ゼロ) に落ちたことがあります。

AP_NO この verb が前回発行されて以来、最低でも 1 つのセッションが常に活動状態でした。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc

AP_PARAMETER_CHECK

secondary_rc

次の値があります。

AP_BAD_TP_ID

tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_INVALID_FORMAT

予約済みパラメーター *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED

アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: この verb で状態検査エラーは発生しません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の値があります。

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Reset 状態を除き、どのような状態にあっても構いません。

状態の変更

この verb では、会話状態は変更されません。

GET_TP_PROPERTIES

GET_TP_PROPERTIES verb は、ローカル TP の属性に関する情報と、その TP が参加している作業論理単位 (LUW) の属性に関する情報を戻します。作業論理単位とは、特定のタスクを実行するための APPC TP 間のトランザクションのことです。これには、通信中の 2 つの TP が関与するか、複数の TP 間での一連の会話が関与します。

VCB 構造体: GET_TP_PROPERTIES

AIX, LINUX

GET_TP_PROPERTIES verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct get_tp_properties
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reserved      */
  unsigned char     format;
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  unsigned char     tp_name[64];
  unsigned char     lu_alias[8];
  LUWID_OVERLAY     luw_id;
  unsigned char     fqlu_name[17];
  unsigned char     reserv3[9];
  unsigned char     verified;
  unsigned char     user_id[10];
  LUWID_OVERLAY     prot_luw_id;
} GET_TP_PROPERTIES;

typedef struct luwid_overlay
{
  unsigned char     fq_length;
  unsigned char     fq_luw_name[17];
  unsigned char     instance[6];
  unsigned char     sequence[2];
} LUWID_OVERLAY;
```

VCB 構造体: GET_TP_PROPERTIES (Windows)

WINDOWS

GET_TP_PROPERTIES verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct get_tp_properties
{
  unsigned short    opcode;
  unsigned char     reserv2[2];
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned char     tp_name[64];
  unsigned char     lu_alias[8];
  unsigned char     luw_id[26];
  unsigned char     fqlu_name[17];
  unsigned char     reserv3[10];
  unsigned char     user_id[10];
} GET_TP_PROPERTIES;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_GET_TP_PROPERTIES

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

tp_name

TP_STARTED verb または RECEIVE_ALLOCATE verb で指定されたローカル TP の TP 名です。これは、64 バイトの EBCDIC 文字ストリングです。

lu_alias

TP_STARTED verb または RECEIVE_ALLOCATE verb で指定された、ローカル TP に認識されているローカル LU の別名。これは 8 バイトの ASCII 文字ストリングです。

AIX, LINUX

luw_id その TP が参加しているトランザクションの無保護の作業論理単位 ID (LUWID)。LUWID はトランザクションを開始した TP を代表するものとして割り当てられ、LUWID を使用すると、そのトランザクションを形成する

さまざまな会話を相互に関連付けることができます。 無保護の LUWID は、無保護の会話に相互に関連付けるために使用されます (*sync_level* が AP_NONE または AP_CONFIRM_SYNC_LEVEL の会話); 同期点処理を使用する TP の場合、 *sync_level* が AP_SYNCPT の会話に対し、追加の保護付き LUWID があります (詳細については、*prot_luw_id* パラメーターを参照してください)。

LUWID は、次のパラメーターからなっています。

luw_id.fq_length
その作業論理単位へ関連付けられている完全修飾 LU 名の長さ (1 ～ 17 バイト) (LU 名自体は、*luw_id.fq_luw_name* パラメーターによって示されます)。

luw_id.fq_luw_name
作業論理単位に対応する LU の完全修飾名。この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。この名前は、1 ～ 8 文字の A ストリング文字からなるネットワーク ID、EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名で構成されます。

luw_id.instance
その作業論理単位へ関連付けられているインスタンス番号 (6 バイトの 2 進数)。

luw_id.sequence
その作業論理単位の現行セグメントの順序番号 (2 バイトの 2 進数)。

WINDOWS

luw_id その TP が参加しているトランザクションの作業論理単位 ID (LUWID)。LUWID はトランザクションを開始した TP を代表するものとして割り当てられ、LUWID を使用すると、そのトランザクションを形成するさまざまな会話を相互に関連付けることができます。

LUWID は 26 バイトのストリングで、 表 9で示すパラメーターから構成されます。

表 9. LUWID パラメーター

パラメーター	長さ	説明
fq_length	1	その作業論理単位へ関連付けられている完全修飾 LU 名の長さ (1 ～ 17 バイト) (LU 名自体は、*fq_luw_name* パラメーターによって示されます)。
fq_luw_name	1-17	作業論理単位に対応する LU の完全修飾名。この名前は、EBCDIC ストリングで、1 ～ 8 文字の A ストリング文字からなるネットワーク ID、 EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名で構成されます。この名前にはスペースを使用できません。 *fq_length* パラメーターは名前のバイト数を指定し、 *instance* パラメーターは、このバイト数のすぐあとに続きます。

表 9. LUWID パラメーター (続き)

パラメーター	長さ	説明
instance	6	その作業論理単位へ関連付けられているインスタンス番号 (6 バイトの 2 進数)。
sequence	2	その作業論理単位の現行セグメントの順序番号 (2 バイトの 2 進数) です。これは常に 1 に設定されます。

fq_length パラメーターが、LU 名が 17 バイトより短いことを示す場合、先行パラメーターの合計長は 26 バイトより短くなります。残りのバイトには EBCDIC スペースが入ります。

fqlu_name
その TP へ関連付けられているローカル LU の完全修飾 LU 名。この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。この名前は、1 ～ 8 文字の A ストリング文字からなるネットワーク ID、 EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名で構成されます。

AIX, LINUX

verified
この会話について、会話セキュリティーが検証されたかどうかを示します。次の値があります。

AP_YES 会話セキュリティーが検証されました。呼び出し元 TP はユーザー ID (この verb で *user_id* パラメーターとして戻されたもの) を提供し、有効なパスワードを提供したか、会話セキュリティーがすでに検証されていたことを示しました。

AP_NO 会話セキュリティーは検証されませんでした。呼び出し先 TP はユーザー ID とパスワードを必要としていません。

user_id その TP へ関連付けられているユーザー ID。これは 10 バイトからなるタイプ AE の EBCDIC ストリングで、この ID が 10 バイトに満たない場合は、右側に EBCDIC のスペースが埋め込まれます。 パスワードはこの verb では戻されず、RECEIVE_ALLOCATE verb で戻されます。

AIX, LINUX

prot_luw_id
その TP が参加しているトランザクションの保護付きの作業論理単位 ID (LUWID)。

保護付き LUWID は、保護会話 (*sync_level* が AP_SYNCPT のもの) を相互に関連付けるために使用されます。 これは、次のパラメーターからなります。

prot_luw_id.fq_length
その作業論理単位へ関連付けられている完全修飾 LU 名の長さ (1 ～ 17 バイト) (LU 名自体は、*prot_luw_id.fq_luw_name* パラメーターによって示されます)。

prot_luw_id.fq_luw_name
作業論理単位に対応する LU の完全修飾名。この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。この名前は、1 ～ 8 文字の A ストリング文字からなるネットワーク ID、EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名で構成されます。

prot_luw_id.instance
その作業論理単位へ関連付けられているインスタンス番号 (6 バイトの 2 進数)。

prot_luw_id.sequence
その作業論理単位の現行セグメントの順序番号 (2 バイトの 2 進数)。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT
format パラメーターが無効な値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: この verb で状態検査エラーは発生しません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の値があります。

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Reset 状態を除き、どのような状態にあっても構いません。

状態の変更

この verb では、会話状態は変更されません。

SET_TP_PROPERTIES

AIX, LINUX

SET_TP_PROPERTIES verb は、アプリケーションからローカル TP の属性を設定できるようにします。それらの属性は、その TP に新規の会話を割り振るときに使用されます。 この verb では、次の属性にアクセスできます。

- 「検査済み」セキュリティーを指定して新規の会話を割り振るときに使用するユーザー ID です。一般に、TP が「検査済み」セキュリティーを使用するのは、有効なユーザー ID とパスワードを指定した別の TP から呼び出され、同じトランザクションの一部として第 3 の TP を呼び出すときです。その場合、APPC はパスワードを必要とせず、最初の TP からのユーザー ID を送信します。 代わ

りに、TP が別の TP によって呼び出されなかった場合、APPC はアプリケーションが会話セキュリティーのユーザー ID として稼働している AIX または Linux ユーザー名を使用します。

しかし、TP が別の方法 (たとえば、会話を割り振る前にユーザーがユーザー ID とパスワードを明示的に入力する必要がある場合など) でユーザー ID とパスワードを取得し、検証した場合、TP は「検査済み」セキュリティーを使用して別の TP を呼び出す前に、 SET_TP_PROPERTIES を使用して APPC にユーザー ID を提供する必要があります。

- その TP が参加している作業論理単位の ID です。作業論理単位とは、特定のタスクを実行するための APPC TP 間のトランザクションのことです。これには、通信中の 2 つの TP が関与するか、複数の TP 間での一連の会話が関与します。 TP へ関連付けられる作業論理単位 ID (LUWID) は 2 つあります。 *sync_level* が AP_NONE か AP_CONFIRM_SYNC_LEVEL の会話に使用する無保護の LUWID と、 *sync_level* が AP_SYNCPT の会話に使用する保護付き LUWID です。

VCB 構造体: SET_TP_PROPERTIES

SET_TP_PROPERTIES verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct set_tp_properties
{
  AP_UINT16         opcode;
  unsigned char     opext;                /* Reserved      */
  unsigned char     format;               /* Reserved      */
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  unsigned char     set_prot_id;
  unsigned char     new_prot_id;
  LUWID_OVERLAY     prot_id;
  unsigned char     set_unprot_id;
  unsigned char     new_unprot_id;
  LUWID_OVERLAY     unprot_id;
  unsigned char     set_user_id;
  unsigned char     set_password;
  unsigned char     user_id[10];
  unsigned char     new_password[10];
} SET_TP_PROPERTIES;

typedef struct luwid_overlay
{
  unsigned char    fq_length;
  unsigned char    fq_luw_name[17];
  unsigned char    instance[6];
  unsigned char    sequence[2];
} LUWID_OVERLAY;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_SET_TP_PROPERTIES

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

set_prot_id

APPC が保護付き作業論理単位 ID を変更するかどうかを指定します。 次の値があります。

AP_YES この TP の保護付き LUWID を変更します。

AP_NO 保護付き LUWID は変更しません。

new_prot_id

APPC が新規の保護付き作業論理単位 ID を生成するか、この verb で指定するものを使用するかを指定します。 このパラメーターは、 *set_prod_id* が AP_NO に設定されている場合に予約されます。以下の値を設定できます。

AP_YES 新規の保護付き LUWID を生成します。

AP_NO TP の保護付き LUWID を、この verb で指定する LUWID に設定します。

prot_id *set_prot_id* を AP_YES に設定し、 *new_prot_id* を AP_NO に設定した場合、この構造体は TP 用の新しい保護付き LUWID を指定します。それ以外の場合、この構造体は予約済みです。 この構造体には、次のパラメーターが含まれます。

prot_id.fq_length

その作業論理単位へ関連付けられている完全修飾 LU 名の長さ (1 ～ 17 バイト) (LU 名自体は、次のパラメーターによって示されます)。

prot_id.fq_luw_name

作業論理単位に対応する LU の完全修飾名。この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。この名前は、1 ～ 8 文字の A ストリング文字からなるネットワーク ID、 EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名で構成されます。

prot_id.instance

その作業論理単位へ関連付けられているインスタンス番号 (6 バイトの 2 進数)。

prot_id.sequence

その作業論理単位の現行セグメントの順序番号 (2 バイトの 2 進数)。

set_unprot_id

APPC が無保護の作業論理単位 ID を変更するかどうかを指定します。 次の値があります。

AP_YES この TP の無保護 LUWID を変更します。

AP_NO 無保護 LUWID を変更しません。

new_unprot_id

APPC が新規の無保護作業論理単位 ID を生成するか、この verb で指定するものを使用するかを指定します。 このパラメーターは、 *set_unprot_id* が AP_NO に設定されている場合に予約されます。以下の値を設定できます。

AP_YES 新規の無保護 LUWID を生成します。

AP_NO TP の無保護 LUWID を、この verb で指定する LUWID に設定します。

unprot_id

set_unprot_id を AP_YES に設定し、 *new_unprot_id* を AP_NO に設定した場合、この構造体は TP 用の新しい無保護 LUWID を指定します。それ以外の場合、この構造体は予約済みです。 この構造体には、次のパラメーターが含まれます。

unprot_id.fq_length

その作業論理単位へ関連付けられている完全修飾 LU 名の長さ (1 ～ 17 バイト) (LU 名自体は、次のパラメーターによって示されます)。

unprot_id.fq_luw_name

作業論理単位に対応する LU の完全修飾名。この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。この名前は、1 ～ 8 文字の A ストリング文字からなるネットワーク ID、 EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名で構成されます。

unprot_id.instance

その作業論理単位へ関連付けられているインスタンス番号 (6 バイトの 2 進数)。

unprot_id.sequence

その作業論理単位の現行セグメントの順序番号 (2 バイトの 2 進数)。

set_user_id

APPC がユーザー ID を変更するかどうかを指定します。次の値があります。

AP_YES この TP のユーザー ID を変更します。

AP_NO ユーザー ID を変更せずに残します。

set_password

APPC が *new_password* パラメーターに関連したパスワードを変更するかどうかを指定します。 次の値があります。

AP_YES APPC はパスワードを変更します。

AP_NO APPC はパスワードを変更しません。

user_id *set_user_id* を AP_YES に設定した場合、このパラメーターは新規ユーザー ID を指定します。それ以外の場合、このパラメーターは予約済みです。

new_password

set_password を AP_YES に設定した場合、このパラメーターは新規パスワードを指定します。それ以外の場合、このパラメーターは予約済みです。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

prot_id *set_prot_id* と *new_prot_id* の両方を AP_YES に設定した場合、この構造体は APPC が TP 用に新規に生成した保護付き LUWID を示します。 この構造体には、次のパラメーターが含まれます。

prot_id.fq_length
その作業論理単位へ関連付けられている完全修飾 LU 名の長さ (1 ～ 17 バイト) (LU 名自体は、*prot_id.fq_luw_name* パラメーターによって示されます)。

prot_id.fq_luw_name
作業論理単位に対応する LU の完全修飾名。この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。この名前は、1 ～ 8 文字の A ストリング文字からなるネットワーク ID、EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名で構成されます。

prot_id.instance
その作業論理単位へ関連付けられているインスタンス番号 (6 バイトの 2 進数)。

prot_id.sequence
その作業論理単位の現行セグメントの順序番号 (2 バイトの 2 進数)。

unprot_id
set_unprot_id と *new_unprot_id* の両方を AP_YES に設定した場合、この構造体は APPC が TP 用に新規に生成した無保護 LUWID を示します。 この構造体には、次のパラメーターが含まれます。

unprot_id.fq_length
その作業論理単位へ関連付けられている完全修飾 LU 名の長さ (1 ～ 17 バイト) (LU 名自体は、*unprot_id.fq_luw_name* パラメーターによって示されます)。

unprot_id.fq_luw_name
作業論理単位に対応する LU の完全修飾名。この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。この名前は、1 ～ 8 文字の A ストリング文字からなるネットワーク ID、EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名で構成されます。

unprot_id.instance
その作業論理単位へ関連付けられているインスタンス番号 (6 バイトの 2 進数)。

unprot_id.sequence
その作業論理単位の現行セグメントの順序番号 (2 バイトの 2 進数)。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_INVALID_FORMAT
予約済みパラメーター *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: この verb で状態検査エラーは発生しません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の値があります。

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話はどのような状態にあっても構いません。

状態の変更

この verb では、会話状態は変更されません。

使用法と制約事項

同期点機能を使用する TP では、ローカル・アプリケーションが保護付き LUWID を変更した場合、同期点管理プログラムは、パートナー・アプリケーションに新規の保護付き LUWID を通知するため、パートナー・アプリケーションに適切な PS ヘッダーを送信する責任を負います。同様に、同期点管理プログラムは、新規の保護付き LUWID が入った PS ヘッダーを受信した場合、SET_TP_PROPERTIES を発行してローカル LU に新規 LUWID を通知する必要があります。

第 4 章 APPC 会話 verb

この章には、個々の APPC 会話 verb の説明が記載されています。それぞれの verb について、次の情報を提供します。

- 各 verb の定義。
- 各 verb が使用する verb 制御ブロック (VCB) の構造体定義。この構造体は、APPC ヘッダー・ファイル **/usr/include/sna/appc_c.h** (AIX)、**/opt/ibm/sna/include/appc_c.h** (Linux)、**sdk/winappc.h** (Windows) に定義されています。 *reserv* で始まるパラメーターは予約されています。
- APPC へ提供するか APPC から戻されるパラメーター (VCB フィールド)。各パラメーターについて、次の情報を提供します。
  - 説明
  - 指定可能な値
  - 追加情報
- 各 verb を発行できる会話状態。
- 各 verb から戻ったときの、変更された会話状態。状態の変更を引き起こさない条件については、特に言及しません。 たとえば、パラメーター検査と状態検査では、状態の変更は起こりません。
- 各 verb の使用法を説明した追加情報。

APPC へ提供するパラメーターと APPC から戻されるパラメーターのほとんどは、16 進値です。コーディングを単純にするため、これらの値は、ヘッダー・ファイル **values_c.h** の中で定義されている、意味のある記号定数によって表されます。このヘッダー・ファイルは、APPC ヘッダー・ファイル **appc_c.h** によってインクルードされます。たとえば、MC_SEND_DATA verb の *opcode* パラメーターは、記号定数 `AP_M_SEND_DATA` によって表される 16 進値です。

指定されたパラメーターに値を設定する場合、または戻されたパラメーターの値をテストする場合には、16 進値ではなく記号定数を使用することが重要です。これは、さまざまなオペレーティング・システムがさまざまな方法でそれらの値をメモリー内に格納するため、示されている値が使用しているシステムで認識される形式になっていない場合もあるためです。

WINDOWS

Windows の場合、指定パラメーターと戻りパラメーター値の定数は、 Windows APPC ヘッダー・ファイル **winappc.h** に定義されています。

「[MC_]verb」という表記は、APPC verb のマップ式対話と基本対話の両方の形を表しています。 たとえば、[MC_]SEND_DATA は、MC_SEND_DATA と SEND_DATA という両方の verb を表しています。

注: APPC VCB には、「予約済み」とマークされたパラメーターが多数含まれています。予約済みパラメーターには、Communications Server ソフトウェアで内部的に使用されているものや、このバージョンでは使用されていなくても将来のバージョンで使用される可能性があるものがあります。アプリケーションでは、予約済みパラメーターにアクセスしてはなりません。その代わりに、verb で使用する他のパラメーターを設定する前に、VCB の全内容をゼロに設定し、これらのパラメーターすべてをゼロにしてください。このようにすると、内部的に使用されるパラメーターが Communications Server で誤って解釈されることがなく、将来のバージョンでこれらのパラメーターが新しい関数を提供するために使用されても、アプリケーションは引き続き動作します。

VCB の内容をゼロに設定するには、`memset` を使用します。

```
memset(vcb, 0, sizeof(vcb));
```

ここでは、会話 verb を次の順序で説明します。

GET_TYPE

CANCEL_CONVERSATION

[MC_]ALLOCATE

[MC_]CONFIRM

[MC_]CONFIRMED

[MC_]DEALLOCATE

[MC_]FLUSH

[MC_]GET_ATTRIBUTES

[MC_]PREPARE_TO_RECEIVE

[MC_]RECEIVE_AND_POST

[MC_]RECEIVE_AND_WAIT

[MC_]RECEIVE_IMMEDIATE

[MC_]RECEIVE_EXPEDITED_DATA

[MC_]REQUEST_TO_SEND

[MC_]SEND_CONVERSATION

[MC_]SEND_DATA

[MC_]SEND_ERROR

[MC_]SEND_EXPEDITED_DATA

[MC_]TEST_RTS

[MC_]TEST_RTS_AND_POST

GET_TYPE

GET_TYPE verb は、特定の会話の会話タイプ (基本またはマップ式) を戻し、会話が全二重または半二重モードで作動するかどうかを判別します。

この情報により、TP がこの会話で発行する正しい verb を判別することができます。

VCB 構造体: GET_TYPE

AIX, LINUX

GET_TYPE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct get_type
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reserved     */
  unsigned char     format;
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     conv_type;
  unsigned char     duplex_type;
} GET_TYPE;
```

VCB 構造体: GET_TYPE (Windows)

WINDOWS

GET_TYPE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct get_type
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     conv_type;
} GET_TYPE;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_GET_TYPE

format 新規 APPC アプリケーションを作成しているか、または既存の APPC アプリケーションを現在の APPC ヘッダー・ファイルで再コンパイルしている場合、このパラメーターを 1 に設定する必要があります。(ヘッダー・ファイルの前のバージョンで作成された既存のアプリケーションには、このパラメーターが予約されているため、アプリケーションは変更されないまま作動するので、再作成する必要はありません。)

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id

この TP が問い合わせている会話の ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

conv_type

conv_id によって識別された会話の会話タイプ。

次の値があります。

AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

duplex_type

conv_id によって識別された会話の全二重タイプ。

次の値があります。

AP_HALF_DUPLEX
AP_FULL_DUPLEX

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc

AP_PARAMETER_CHECK

secondary_rc

次の値があります。

AP_BAD_CONV_ID

conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID

tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT

format パラメーターが無効な値に設定されていました。

AP_SYNC_NOT_ALLOWED

アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: この verb で状態検査エラーは発生しません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Reset 状態を除き、どのような状態にあっても構いません。

状態の変更

この verb では、会話状態は変更されません。

CANCEL_CONVERSATION

CANCEL_CONVERSATION verb は、 2 つの TP 間の会話の割り振りを解除します。 これは、次の相違点を除くと、MC_DEALLOCATE または DEALLOCATE verb の *dealloc_type* パラメーターを AP_ABEND、または AP_ABEND_* 値のいずれかに設定することと同等です。

- MC_DEALLOCATE または DEALLOCATE を、この会話上で別の verb の進行中に使用することはできません。CANCEL_CONVERSATION を使用して、未解決の verb を取り消すことができます。
- DEALLOCATE (MC_DEALLOCATE ではない) には、オプションで、ローカル・エラー・ログに書き込まれるログ・データが含まれます。
  CANCEL_CONVERSATION にはこの機能はありません。

会話上のすべての未解決の verb の結果は未定義となり、アプリケーションには返されません。たとえば、CANCEL_CONVERSATION を、[MC_]SEND_DATA が未解決の間に発行する場合は、パートナー TP にデータが送信されているかどうかを判別できません。

この verb が正常に実行された後、会話 ID は無効になります。

VCB 構造体: CANCEL_CONVERSATION

AIX, LINUX

CANCEL_CONVERSATION verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct cancel_conversation
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
} CANCEL_CONVERSATION;
```

VCB 構造体: CANCEL_CONVERSATION (Windows)

WINDOWS

CANCEL_CONVERSATION verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct cancel_conversation
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     format;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
} CANCEL_CONVERSATION;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_CANCEL_CONVERSATION

opext verb が全二重会話で発行されている場合、または非ブロッキング verb として発行されている場合、次の値のいずれかまたは両方を指定します (論理 OR を使用して結合)。それ以外の場合、このパラメーターは予約済みです。

AP_FULL_DUPLEX_CONVERSATION
この verb は全二重会話で発行されます。

AP_NON_BLOCKING
この verb は非ブロッキング verb として発行されます。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id

会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT
format パラメーターが無効な値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED

AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

AIX, LINUX

primary_rc

AP_BACKED_OUT

secondary_rc

AP_BO_NO_RESYNC
AP_BO_RESYNC

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_DEALLOC_ABEND
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_DUPLEX_TYPE_MIXED
AP_PROG_ERROR_PURGING
AP_INVALID_VERB
AP_SVC_ERROR_PURGING
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP から CANCEL_CONVERSATION verb を発行する場合、会話は Reset 状態を除き、どのような状態にあっても構いません。

状態の変更

CANCEL_CONVERSATION verb が完了すると、会話がリセット状態になります。

MC_ALLOCATE および ALLOCATE

MC_ALLOCATE verb または ALLOCATE verb は、呼び出し元 TP が発行します。この verb はローカル LU とパートナー LU の間のセッションを割り振り、(RECEIVE_ALLOCATE verb と連携して) 呼び出し元 TP と呼び出し先 TP 間の会話を確立します。

MC_ALLOCATE verb はマップ式会話を確立します。 ALLOCATE verb は、基本会話とマップ式会話のどちらでも確立できます。 ALLOCATE verb を使用してマップ式会話を確立すると、TP で基本会話 verb を使用してマップ式会話のパートナー TP と通信できます。

この verb の実行が正常に終了すると、APPC は会話 ID (*conv_id*) を生成します。この ID は、他のすべての APPC 会話 verb の必須パラメーターです。

通常の場合、[MC_]ALLOCATE 要求は即時にはパートナー LU へ送信されません。この要求は、バッファー全体の内容を送信できるようになるまでローカル LU のキュー内に置かれます。 したがって、会話の割り振りでエラーがあっても、それは [MC_]ALLOCATE verb では報告されず、それ以後の verb で報告されるのが普通です。

VCB 構造体: MC_ALLOCATE

AIX, LINUX

MC_ALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_allocate
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    reserv3;
  unsigned char    sync_level;
  unsigned char    reserv4[2];
  unsigned char    rtn_ctl;
  unsigned char    duplex_type;
  AP_UINT32        conv_group_id;
  AP_UINT32        sense_data;
  unsigned char    plu_alias[8];
  unsigned char    mode_name[8];
  unsigned char    tp_name[64];
  unsigned char    security;
```

```
  unsigned char    reserv6[11];
  unsigned char    pwd[10];
  unsigned char    user_id[10];
  AP_UINT16        pip_dlen;
  unsigned char    *pip_dptr;
  unsigned char    reserv6a;
  unsigned char    fqplu_name[17];
  unsigned char    reserv7[8];
} MC_ALLOCATE;
```

VCB 構造体: ALLOCATE

ALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct allocate
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    conv_type;
  unsigned char    sync_level;
  unsigned char    reserv3[2];
  unsigned char    rtn_ctl;
  unsigned char    duplex_type;
  AP_UINT32        conv_group_id;
  AP_UINT32        sense_data;
  unsigned char    plu_alias[8];
  unsigned char    mode_name[8];
  unsigned char    tp_name[64];
  unsigned char    security;
  unsigned char    reserv5[11];
  unsigned char    pwd[10];
  unsigned char    user_id[10];
  AP_UINT16        pip_dlen;
  unsigned char    *pip_dptr;
  unsigned char    reserv5a;
  unsigned char    fqplu_name[17];
  unsigned char    reserv6[8];
} ALLOCATE;
```

VCB 構造体: MC_ALLOCATE (Windows)

WINDOWS

MC_ALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_allocate
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     reserv3;
  unsigned char     sync_level;
  unsigned char     reserv4[2];
  unsigned char     rtn_ctl;
  unsigned char     reserv5;
```

```
  unsigned long     conv_group_id;
  unsigned long     sense_data;
  unsigned char     plu_alias[8];
  unsigned char     mode_name[8];
  unsigned char     tp_name[64];
  unsigned char     security;
  unsigned char     reserv6[11];
  unsigned char     pwd[10];
  unsigned char     user_id[10];
  unsigned short    pip_dlen;
  unsigned char far *pip_dptr;
  unsigned char     reserv7;
  unsigned char     fqplu_name[17];
  unsigned char     reserv8[8];
} MC_ALLOCATE;
```

VCB 構造体: ALLOCATE (Windows)

ALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct allocate
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     conv_type;
  unsigned char     sync_level;
  unsigned char     reserv3[2];
  unsigned char     rtn_ctl;
  unsigned char     reserv4;
  unsigned long     conv_group_id;
  unsigned long     sense_data;
  unsigned char     plu_alias[8];
  unsigned char     mode_name[8];
  unsigned char     tp_name[64];
  unsigned char     security;
  unsigned char     reserv5[11];
  unsigned char     pwd[10];
  unsigned char     user_id[10];
  unsigned short    pip_dlen;
  unsigned char far *pip_dptr;
  unsigned char     reserv7;
  unsigned char     fqplu_name[17];
  unsigned char     reserv8[8];
} ALLOCATE;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_ALLOCATE
MC_ALLOCATE verb の場合

AP_B_ALLOCATE
ALLOCATE verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_ALLOCATE verb の場合

AP_BASIC_CONVERSATION
ALLOCATE verb の場合

verb が非ブロッキング verb として発行されている場合、上記の値を値 AP_NON_BLOCKING と (論理 OR を使用して) 結合します。

tp_id ローカル TP の ID。

このパラメーターの値は、TP_STARTED verb が呼び出し元 TP について戻した値か、RECEIVE_ALLOCATE verb が呼び出し先 TP について戻した値です。

conv_type
割り振る会話のタイプ。このパラメーターは、ALLOCATE verb のみが使用します。

次の値があります。
AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

ALLOCATE verb がマップ式会話を確立した場合、ローカル TP は基本会話 verb を発行して独自のマッピング層を提供し、データ・レコードを論理レコードに変換し、論理レコードをデータ・レコードに変換する必要があります。 パートナー TP は基本会話 verb を発行してマッピング層を提供するか、(パートナー TP が使用している APPC のインプリメンテーションがマップ式会話 verb をサポートしている場合は) マップ式会話 verb を使用できます。 詳細については、IBM 資料の「*Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*」を参照してください。

sync_level
会話の同期レベル。

このパラメーターは、TP がデータの受信の確認を要求できるかどうか、またデータの受信を確認できるかどうかを決定します。次の値があります。

AP_NONE
この会話では、確認処理は使用されません。

AP_CONFIRM_SYNC_LEVEL
各 TP は、この会話で確認処理を使用できます。この値は、半二重会話でのみ使用できます。確認処理は全二重会話ではサポートされません。

AIX, LINUX

AP_SYNCPT
各 TP は、この会話で LU6.2 同期点機能を使用できます。 この値は、標準 Communications Server 製品のほかに同期点管理プログラム (SPM) と Conversation Protected Resource Manager (C-PRM) が

ある場合にのみ設定してください。 詳細については、 25 ページの『同期点サポート』を参照してください。

rtn_ctl ローカル TP からのセッション要求で動作するローカル LU がローカル TP への制御を戻すことを指定します。セッションについての詳細は、 66 ページの『LU 間セッション』を参照してください。このパラメーターの値が何であっても、LU はゼロ・セッション限度 (これは、セッションが決して割り振られないことを意味します) など、特定のエラーを検出した場合は、即時に TP に制御を戻します。

次の値があります。

AP_IMMEDIATE

- DEFINE_MODE verb または **define_mode** コマンドの *auto_act* パラメーターが 0 (ゼロ) に設定されている場合、Communications Server はセッションの活動化を試みません。 コンテンション勝者セッションが即時に使用可能である (活動状態であり、別の会話に使用されていない) 場合、LU はこの会話をそのセッションに割り振り、即時に TP へ制御を戻します。 コンテンション勝者セッションが即時に使用可能でない場合は、即時に TP へ制御が戻され、*primary_rc* が AP_UNSUCCESSFUL になります。
- DEFINE_MODE verb または **define_mode** コマンドの *auto_act* パラメーターが、ゼロ以外の値に設定されている場合、Communications Server はセッションを活動化しようと試みます。

詳細については、「*IBM Communications Server for Linux NOF プログラマーズ・ガイド*」または「*IBM Communications Server for AIX NOF プログラマーズ・ガイド*」で DEFINE_MODE verb の説明を参照するか、「*IBM Communications Server for Linux 管理コマンド解説書*」または「*IBM Communications Server for AIX 管理コマンド解説書*」で **define_mode** コマンドの項を参照してください。

AP_WHEN_SESSION_ALLOCATED

セッションが即時に使用可能である (活動状態であり、別の会話に使用されていない) 場合、LU はこの会話をそのセッションに割り振ります。即時に使用可能でなくても、活動状態にできるセッションがある場合、LU はそのセッションを活動状態にし、この会話をそのセッションに割り振ります。セッションを活動状態にできない場合、LU はセッションの 1 つが解放されるまで待機します。

AP_WHEN_SESSION_FREE

セッションが即時に使用可能である (活動状態であり、別の会話に使用されていない) 場合、LU はこの会話をそのセッションに割り振ります。即時に使用可能でなくても、活動状態にできるセッションがある場合、LU はそのセッションを活動状態にし、この会話をそのセッションに割り振ります。活動状態で解放されたセッションがなく、別のセッションを活動状態にできない場合は、1 次戻りコード AP_ALLOCATION_ERROR および 2 次戻りコード AP_ALLOCATION_FAILURE_RETRY と共に制御が TP へ戻されます。こ

れは、セッションが解放されるのを LU が待機しないことを除けば、 AP_WHEN_SESSION_ALLOCATED によく似ています。

AP_WHEN_CONWINNER_ALLOC
AP_WHEN_SESSION_ALLOCATED に関しては、LU が常に会話をコンテンション勝者セッションに割り振ることを除き、コンテンション敗者セッションを使用しません。

AP_WHEN_CONLOSER_ALLOC
AP_WHEN_SESSION_ALLOCATED に関しては、LU が常に会話をコンテンション敗者セッションに割り振ることを除き、コンテンション勝者セッションを使用しません。

AP_WHEN_CONV_GROUP_ALLOC
この値は、新規の会話に前の会話と同じセッションを使用させる場合に使用します。*conv_group_id* パラメーターに、[MC_]ALLOCATE verb または RECEIVE_ALLOCATE verb から戻された前の会話の会話グループ ID を設定してください。

conv_group_id パラメーターで識別されるセッションが即時に使用可能である (活動状態であり、別の会話に使用されていない) 場合、LU はこの会話をそのセッションに割り振り、即時に TP へ制御を戻します。そのセッションが別の会話に使用されている場合、LU はそのセッションが解放されるまで待機します。 そのセッションが活動状態でなくなっている場合は、1 次戻りコード AP_ALLOCATION_ERROR および 2 次戻りコード AP_ALLOCATION_FAILURE_NO_RETRY と共に制御が TP へ戻されます。

duplex_type
新規会話の全二重タイプ。全二重会話と半二重会話の違いについての詳細は、4 ページの『半二重および全二重会話』を参照してください。

次の値があります。

AP_HALF_DUPLEX
半二重会話。

AP_FULL_DUPLEX
全二重会話。

conv_group_id
会話用に要求したセッションの会話グループ ID。このパラメーターは、*rtn_ctl* を AP_WHEN_CONV_GROUP_ALLOC に設定した場合にのみ使用します。*rtn_ctl* をそれ以外の値に設定した場合は、このパラメーターを 2 進ゼロに設定してください。

plu_alias
パートナー LU をローカル TP に認識させるための別名。

このパラメーターは 8 バイトの ASCII 文字ストリングで、別名が 8 文字より短い場合は、右側に ASCII のブランク (0x20) が埋め込まれます。 これは、次の文字から構成できます。

- 英大文字
- 数字の 0 ～ 9

- ブランク
- 特殊文字の $、#、%、および @

このストリングの 1 文字目をブランクにすることはできません。

LU を LU 別名でなく LU 名で識別するには、このパラメーターを 8 つの 2 進ゼロに設定し、*fqplu_name* パラメーターに LU 名を指定します。

mode_name

構成のときに定義されたネットワーク機能の特性セットの名前。

このパラメーターは、8 バイトの EBCDIC 文字ストリングです。 このパラメーターは、タイプ A の EBCDIC 文字セットの文字から構成できます。それらの文字は、次のとおりです。

- 英大文字
- 数字の 0 ～ 9
- 特殊文字の $、#、および ?

ストリングの先頭文字は、大文字でなければなりませんが、SNA 定義のモードのいずれかに対しては、 `#INTER` のように、# でも構いません。SNA 定義モードについての詳細は、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。モード名の長さが 8 文字未満の場合は、EBCDIC のブランク (`0x40`) を右側に埋め込んでください。

モード名は、すべて EBCDIC のブランク (`0x40`) にすることもできます。

マップ式会話では、SNASVCMG (APPC の内部で使用される予約済みのモード名) という名前にすることはできません。 この名前を基本会話で使用することも、推奨できません。

指定したモード名が SNA 定義モードまたは Communications Server 構成に定義されたモードのいずれにも一致しない場合、Communications Server は、構成に指定されたデフォルト (または、デフォルト・モードが定義されていない場合は、ブランク・モード名を使用した SNA 定義モード) に基づいて、新規モードを作成します。

tp_name

呼び出し先 TP の名前。

呼び出し元 TP 内で [MC_]ALLOCATE verb が指定する *tp_name* の値は、呼び出し先 TP 内で RECEIVE_ALLOCATE verb によって指定される *tp_name* の値に一致する必要があります。

このパラメーターは 64 バイトの EBCDIC 文字ストリングで、大文字小文字の区別があります。通常、*tp_name* パラメーターはタイプ AE の EBCDIC 文字セットの文字からなっています (サービス TP に命名する場合は除きます)。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 ～ 9
- 特殊文字の $、#、?、およびピリオド (.)

TP 名が 64 バイト未満の場合は、EBCDIC のブランク (`0x40`) を右側に埋め込みます。

サービス TP の命名に関する SNA 規則は、上記の例外です。この名前は最大 4 文字からなり、1 文字目は 0x00 ～ 0x3F の 16 進バイトです。その他の文字は EBCDIC AE 文字セットからのものです。

セキュリティー

パートナー LU が呼び出し先 TP へのアクセスの妥当性を検査するために必要な情報を指定します。

構成のときに呼び出し先 TP 用に確立された会話セキュリティーに基づいて、次のいずれかの値を使用してください。

AP_NONE

呼び出し先 TP は、会話セキュリティーを使用しません (この値を使用する場合、呼び出し先 TP は、会話セキュリティーを使用しないように構成されている必要があります)。

AP_PGM 呼び出し先 TP は会話セキュリティーを使用します。したがってユーザー ID とパスワードを必要とします。 この情報を *user_id* パラメーターと *pwd* パラメーターによって提供してください。

AP_PGM_STRONG

呼び出し先 TP は会話セキュリティーを使用します。したがってユーザー ID とパスワードを必要とします。 さらに、AP_PGM_STRONG を設定すると、 Communications Server はネットワークを通じてパスワードを送信するときに、パスワードを暗号化します。 ユーザー ID とパスワードを *user_id* パラメーターと *pwd* パラメーターによって提供してください。

AP_SAME

この値は、TP が別の TP から有効なユーザー ID とパスワードを使用して呼び出され、次に会話セキュリティーを必要とする第 3 の TP を呼び出そうとするときに使用します (ある TP が 2 番目の TP を呼び出し、2 番目の TP が第 3 の TP を呼び出す状況については、 4 ページの『複数の会話』に例を示しています。) この値は、第 3 の TP (呼び出し元 TP) に最初の呼び出し元 TP の会話セキュリティーがすでに検証済みであることを通知します。

この値を使用する場合、この [MC_]ALLOCATE verb で提供する *tp_id* は、この TP が呼び出されたときに RECEIVE_ALLOCATE verb で戻された tp_id と同じものである必要があります。

AIX, LINUX

この値は、TP が別の TP から呼び出されたわけではなく、適切なセキュリティー情報を別の手段で (たとえば、ログオン時に提供された AIX または Linux ユーザー名とパスワードから) 取得し検証した場合にも使用できます。 その場合、APPC はアプリケーションの実行に使用されている AIX または Linux ユーザー名を、必要であれば 10 文字に切り捨てて会話セキュリティー用のユーザー ID として使用します。この名前は、有効な AE ストリング文字 (詳細は *user_id* パラメーターの説明を参照) で構成し、呼び出し先の TP に有効なユーザー名になるようにしてください。

TP は、別の方法 (たとえば、会話を割り振る前にユーザーに有効なユーザー ID とパスワードの入力を要求するなど) でセキュリティー情報を取得した場合、[MC_]ALLOCATE を発行する前に、SET_TP_PROPERTIES を使用してそのユーザー ID を APPC に対して指定しなければなりません。

pwd *user_id* へ関連付けられているパスワード。

このパラメーターは、*security* パラメーターを AP_PGM または AP_PGM_STRONG に設定した場合にのみ必要です。それ以外の場合は予約済みです。

pwd パラメーターと *user_id* パラメーターは、呼び出し先 TP が置かれているコンピューター上に構成されたユーザー ID とパスワードの対に一致する必要があります。

このパラメーターは 10 バイトの EBCDIC 文字ストリングで、大文字小文字の区別があります。 *pwd* パラメーターは、タイプ AE の EBCDIC 文字セットの文字から構成できます。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 ～ 9
- 特殊文字の $、#、?、およびピリオド (.)

パスワードが 10 バイト未満の場合は、EBCDIC のブランク (0x40) を右側に埋め込みます。

user_id パートナー TP にアクセスするために必要なユーザー ID。

このパラメーターは、*security* パラメーターを AP_PGM または AP_PGM_STRONG に設定した場合にのみ必要です。それ以外の場合は予約済みです。

pwd パラメーターと *user_id* パラメーターは、呼び出し先 TP が置かれているコンピューター上に構成されたユーザー ID とパスワードの対に一致する必要があります。

このパラメーターは 10 バイトの EBCDIC 文字ストリングで、大文字小文字の区別があります。 *user_id* パラメーターは、タイプ AE の EBCDIC 文字セットの文字から構成できます。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 ～ 9
- 特殊文字の $、#、?、およびピリオド (.)

ユーザー ID が 10 バイト未満の場合は、EBCDIC のブランク (0x40) を右側に埋め込みます。

pip_dlen

パートナー TP へ渡すプログラム初期設定パラメーター (PIP) の長さ。この値の範囲は、0 ～ 32,767 です。

すべてのインプリメンテーション形態の APPC が PIP データをサポートしているわけではありません。 PIP データをサポートしないインプリメンテ

ーション形態の APPC をパートナー TP が使用している場合、またはパートナーが CPI-C アプリケーションである場合は、 *pip_dlen* を 0 (ゼロ) に設定してください。

pip_dptr

PIP データが入っているバッファーのアドレス。

このパラメーターは、*pip_dlen* が 0 (ゼロ) より大きい場合にのみ使用してください。

PIP データは、パートナー TP またはリモート・オペレーティング・システムが必要とする初期設定パラメーターまたは環境セットアップ情報から構成できます。 PIP データは、汎用データ・ストリームの形式に従っている必要があります。 詳細については、IBM 資料の「*Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*」を参照してください。

WINDOWS

PIP データ・バッファーは、静的データ域か、グローバル割り振り域にあります。データ・バッファーは、この領域と完全に一致していなければなりません。

fqplu_name

パートナー LU の完全修飾 LU 名。このパラメーターは、*plu_alias* をゼロに設定した場合にのみ使用します。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれ、次のいずれかが含まれます。

- 1 ～ 8 文字の A ストリング文字からなるネットワーク ID、 EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名
- 1 ～ 8 文字の A ストリング文字からなる LU 名 (ネットワーク ID や EBCDIC のドットを伴わない)

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

conv_id

会話 ID。この値は、2 つの TP 間に確立された会話を識別します。

conv_group_id
会話が使用するセッションの会話グループ ID。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_TYPE
(基本会話の ALLOCATE でのみ戻ります) *conv_type* に指定した値が有効ではありませんでした。

AP_BAD_DUPLEX_TYPE
duplex_type に指定した値が有効ではありませんでした。

AP_BAD_PARTNER_LU_ALIAS
次のいずれかが発生しました。

- *plu_alias* パラメーターが、定義されているどのパートナー LU 別名にも一致しませんでした。
- *fqplu_name* に指定した値が有効ではありませんでした。

AP_BAD_RETURN_CONTROL
rtn_ctl に指定した値が有効ではありませんでした。

AP_BAD_SECURITY
セキュリティーに指定した値が有効ではありませんでした。

AP_BAD_SYNC_LEVEL
sync_level に指定した値が有効ではありませんでした。

AP_BAD_TP_ID
tp_id に指定した値が有効ではありませんでした。

WINDOWS

AP_INVALID_DATA_SEGMENT
PIP データが割り振りデータ・セグメントより長かったか、PIP データ・バッファーのアドレスが間違っていました。

AP_CONFIRM_INVALID_FOR_FDX
sync_level パラメーターが、全二重会話で AP_CONFIRM_SYNC_LEVEL に設定されました。この値は、半二重会話でのみ使用できます。

AP_NO_USE_OF_SNASVCMG
(MC_ALLOCATE についてのみ戻ります) SNASVCMG は、*mode_name* の値として有効ではありません。

AP_PIP_LEN_INCORRECT
pip_dlen の値が 32,767 を超えていました。

AP_UNKNOWN_PARTNER_MODE
mode_name に指定した値が有効ではありませんでした。

AIX, LINUX

AP_INVALID_FORMAT
format パラメーターが無効な値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: この verb で状態検査エラーは発生しません。

セッションが使用不可の場合: *rtn_ctl* に指定した値によっては、APPC が次のパラメーターを戻す場合があります。

primary_rc

AP_UNSUCCESSFUL
指定パラメーター *rtn_ctl* で TP に即時に制御を戻すこと (AP_IMMEDIATE) を指定しましたが、使用可能なコンテンション勝者セッションがローカル LU にありませんでした。

割り振りエラー: Communications Server が会話を割り振れない場合、APPC は次のパラメーターを戻します。

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
次の値があります。

AP_ALLOCATION_FAILURE_NO_RETRY
構成エラーまたはセッション・プロトコル・エラーなど、永続的な条件のために会話を割り振ることができません。エラーを判別するため、システム管理者はエラー・ログ・ファイルを調べる必要があります。エラーの訂正が済むまで、割り振りの再試行を試みてはなりません。

この値は、要求した会話グループ ID に対応するセッションがもはや活動状態ではない場合にも戻ります。

AP_ALLOCATION_FAILURE_RETRY
リンクの障害など、一時的な条件のために会話を割り振ることができませんでした。障害の原因は、システム・エラー・ログに記録されています。割り振りを再試行してください。ただし、条件がクリアされるよう、できるだけタイムアウトの後に再試行してください。

AP_FDX_NOT_SUPPORTED_BY_LU
duplex_type パラメーターが、AP_FULL_DUPLEX に設定されていましたが、この TP で使用している LU は、全二重操作をサポートしません。

次の 2 次戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

sense_data
割り振りが失敗した原因について詳しい情報を提供する SNA センス・データ。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP がこの verb を発行するとき、会話状態は Reset です。この verb は、既存の会話がどのような状態にあるときでも発行できます。これは、この verb は常に新規の会話 (これは Reset 状態にあります) の開始を意味しているためです。

状態の変更

この verb が正常に実行される (*primary_rc* が AP_OK である) と、新しい会話の状態は、半二重会話では Send に、全二重会話では Send_Receive になります。この verb が失敗した場合、状態の変更はありません。

EBCDIC-ASCII、ASCII-EBCDIC 変換

[MC_]ALLOCATE verb のいくつかのパラメーターは、 EBCDIC か ASCII のストリングです。 TP は、共通サービス verb の CONVERT を使用してストリングをある文字セットから別の文字セットに変換できます。詳細については、「*IBM Communications Server for Linux or AIX 共通サービス Verb プログラマーズ・ガイド*」を参照してください。

即時割り振り

パートナーとの会話が即時に開始されるようにするため、呼び出し元 TP は [MC_]ALLOCATE verb の直後に [MC_]FLUSH verb または [MC_]CONFIRM verb を発行できます。 ([MC_]CONFIRM は半二重会話にのみ適用されます。)それらの verb を発行しなかった場合、[MC_]ALLOCATE 要求はローカル LU の送信バッファーに他のデータと共に入り、バッファーがいっぱいになるまで処理されません。

割り振りの確認 (半二重会話専用)

呼び出し元 TP は、[MC_]ALLOCATE の後に [MC_]CONFIRM verb を発行することにより、割り振りが成功したかどうか (*sync_level* が AP_CONFIRM_SYNC_LEVEL に設定されたかどうか) を即時に判別できます。

MC_CONFIRM および CONFIRM

MC_CONFIRM および CONFIRM verb は、ローカル LU の送信バッファーの内容と確認要求をパートナー TP へ送信します。

注: この verb は、半二重会話でのみ使用できます。全二重会話では無効です。

通常、パートナー TP は [MC_]CONFIRM verb への応答として [MC_]CONFIRMED verb を発行し、データをエラーなしに受信したことを確認します (パートナー TP は、エラーを検出した場合、[MC_]SEND_ERROR verb を発行するか、異常処理として会話の割り振りを解除します)。

TP が [MC_]CONFIRM verb を発行できるのは、[MC_]ALLOCATE verb によって確立された会話の同期レベルが、AP_CONFIRM_SYNC_LEVEL である場合のみです。

VCB 構造体: MC_CONFIRM

AIX, LINUX

MC_CONFIRM verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_confirm
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     rts_rcvd;
  unsigned char     expd_rcvd;
} MC_CONFIRM;
```

VCB 構造体: CONFIRM

CONFIRM verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct confirm
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     rts_rcvd;
  unsigned char     expd_rcvd;
} CONFIRM;
```

VCB 構造体: MC_CONFIRM (Windows)

WINDOWS

MC_CONFIRM verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_confirm
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     rts_rcvd;
} MC_CONFIRM;
```

VCB 構造体: CONFIRM (Windows)

CONFIRM verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct confirm
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     rts_rcvd;
} CONFIRM;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_CONFIRM
MC_CONFIRM verb の場合

AP_B_CONFIRM
CONFIRM verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_CONFIRM verb の場合

AP_BASIC_CONVERSATION
CONFIRM verb の場合

verb が非ブロッキング verb として発行されている場合、上記の値を値 AP_NON_BLOCKING と (論理 OR を使用して) 結合します。

format 新規 APPC アプリケーションを作成しているか、または既存の APPC アプリケーションを現在の APPC ヘッダー・ファイルで再コンパイルしている場合、このパラメーターを 1 に設定する必要があります。(ヘッダー・ファイルの前のバージョンで作成された既存のアプリケーションには、このパラメーターが予約されているため、アプリケーションは変更されないまま作動するので、再作成する必要はありません。)

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id

会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

rts_rcvd

送信要求受信インディケーター。 このパラメーターは、半二重会話でのみ使用できます。全二重会話では使用されません。

次の値があります。

AP_YES パートナー TP は、会話を Receive 状態に変更するようローカル TP に要求する [MC_]REQUEST_TO_SEND verb を発行しました。ローカル TP は、Receive 状態に変更するため、[MC_]PREPARE_TO_RECEIVE、[MC_]RECEIVE_AND_WAIT、[MC_]RECEIVE_AND_POST のいずれかの verb を使用できます。

AP_NO パートナー TP は [MC_]REQUEST_TO_SEND verb を発行しませんでした。

expd_rcvd

優先データ・インディケーター。

次の値があります。

AP_YES パートナー TP が、ローカル TP がまだ受信していない優先データを送信しました。このデータを受信するには、ローカル TP は [MC_]RECEIVE_EXPEDITED_DATA verb を使用することができます。

このインディケーターを、APPC verb 数に対して設定できます。これは、ローカル TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行してデータを受信するまで、後続の verb に対しても設定できます。

AP_NO 受信を待機している優先データはありません。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc

AP_PARAMETER_CHECK

secondary_rc

次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_CONFIRM_INVALID_FOR_FDX
ローカル TP は、全二重会話の中で [MC_]CONFIRM verb を使用しようとしました。この verb は、半二重会話でのみ使用できます。

AP_CONFIRM_ON_SYNC_LEVEL_NONE
ローカル TP は、同期レベルが AP_NONE の会話の中で [MC_]CONFIRM verb を使用しようとしました。 [MC_]ALLOCATE verb によって確立される同期レベルは、 AP_CONFIRM_SYNC_LEVEL である必要があります。

AIX, LINUX

AP_INVALID_FORMAT
format パラメーターが無効な値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc
次の値があります。

AP_CONFIRM_BAD_STATE
会話が Send と Send_Pending のどちらの状態でもありませんでした。

AP_CONFIRM_NOT_LL_BDY
(基本会話の CONFIRM の場合にのみ戻ります) ローカル TP 用の会話が Send 状態にあり、ローカル TP が論理レコードの送信を終了しませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc

AP_ALLOCATION_ERROR

secondary_rc

AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

AIX, LINUX

primary_rc

AP_BACKED_OUT

secondary_rc

AP_BO_NO_RESYNC
AP_BO_RESYNC

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_CONV_FAILURE_NO_RETRY

AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_PROG_ERROR_PURGING
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、MC_CONFIRM verb が戻します。

primary_rc

AP_DEALLOC_ABEND

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、CONFIRM verb が戻します。

primary_rc

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Send 状態か Send_Pending 状態でなければなりません。

状態の変更

次の表に要約した状態変更は、*primary_rc* パラメーターの値に基づいています。

primary_rc	**新しい状態**
AP_OK	Send (送信)

primary_rc	新しい状態
AP_PARAMETER_CHECK	変更なし
AP_STATE_CHECK	
AP_CONVERSATION_TYPE_MIXED	
AP_INVALID_VERB	
AP_INVALID_VERB_SEGMENT	
AP_STACK_TOO_SMALL	
AP_TP_BUSY	
AP_UNEXPECTED_DOS_ERROR	
AP_PROG_ERROR-PURGING	Receive (受信)
AP_SVC_ERROR_PURGING	
AP_ALLOCATION_ERROR	Reset (リセット)
AP_COMM_SUBSYSTEM_ABENDED	
AP_COMM_SUBSYSTEM_NOT_LOADED	
AP_CONV_FAILURE_RETRY	Reset (リセット)
AP_CONV_FAILURE_NO_RETRY	
AP_DEALLOC_ABEND	Reset (リセット)
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	

パートナー TP との同期

[MC_]CONFIRM verb は、パートナー TP からの応答を待機します。応答は、パートナー TP 内にある次のいずれかの verb によって生成されます。

- [MC_]CONFIRMED
- [MC_]SEND_ERROR
- *dealloc_type* が AP_ABEND に設定された MC_DEALLOCATE
- *dealloc_type* が AP_ABEND_PROG、AP_ABEND_SVC、または AP_ABEND_TIMER に設定された DEALLOCATE
- TP_ENDED

MC_CONFIRMED および CONFIRMED

MC_CONFIRMED verb または CONFIRMED verb は、パートナー TP からの確認要求に応答します。 この verb は、ローカル TP が受信データの中でエラーを検出しなかったことをパートナー TP に通知します。

注: この verb は、半二重会話でのみ使用できます。全二重会話では無効です。

確認要求を発行した TP は確認を待機するため、 [MC_]CONFIRMED verb は 2 つの TP の処理を同期させます。

確認要求のソース

確認要求は、パートナー TP 内にある次のいずれかの verb によって発行されます。

- [MC_]CONFIRM

- [MC_]PREPARE_TO_RECEIVE。ただし、*ptr_type* が AP_SYNC_LEVEL に設定されており、会話の同期レベル ([MC_]ALLOCATE verb によって確立されたもの) が AP_CONFIRM_SYNC_LEVEL の場合。
- [MC_]DEALLOCATE。ただし、*dealloc_type* が AP_SYNC_LEVEL に設定されており、会話の同期レベル ([MC_]ALLOCATE verb によって確立されたもの) が AP_CONFIRM_SYNC_LEVEL の場合。
- [MC_]SEND_DATA。ただし、*type* が AP_SEND_DATA_CONFIRM に設定されており、会話の同期レベル ([MC_]ALLOCATE verb によって確立されたもの) が AP_CONFIRM_SYNC_LEVEL の場合。

確認要求の受信

確認要求は、次のいずれかの verb の *what_rcvd* パラメーターを通じてローカル TP に受信されます。

- [MC_]RECEIVE_IMMEDIATE
- [MC_]RECEIVE_AND_WAIT
- [MC_]RECEIVE_AND_POST

ローカル TP が MC_CONFIRMED verb または CONFIRMED verb を発行できるのは、*what_rcvd* フィールドに次のいずれかの値が入っている場合のみです。

- AP_CONFIRM_WHAT_RECEIVED、 AP_DATA_CONFIRM、または AP_DATA_COMPLETE_CONFIRM
- AP_CONFIRM_SEND、AP_DATA_CONFIRM_SEND、または AP_DATA_COMPLETE_CONFIRM_SEND
- AP_CONFIRM_DEALLOCATE、AP_DATA_CONFIRM_DEALLOCATE、または AP_DATA_COMPLETE_CONFIRM_DEALL

VCB 構造体: MC_CONFIRMED

AIX, LINUX

MC_CONFIRMED verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_confirmed
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;                 /* Reserved            */
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
} MC_CONFIRMED;
```

VCB 構造体: CONFIRMED

CONFIRMED verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct confirmed
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     reserv2;
```

```
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
} CONFIRMED;
```

VCB 構造体: MC_CONFIRMED (Windows)

WINDOWS

MC_CONFIRMED verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_confirmed
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
} MC_CONFIRMED;
```

VCB 構造体: CONFIRMED (Windows)

CONFIRMED verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct confirmed
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
} CONFIRMED;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_CONFIRMED
MC_CONFIRMED verb の場合

AP_B_CONFIRMED
CONFIRMED verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_CONFIRMED verb の場合

AP_BASIC_CONVERSATION
CONFIRMED verb の場合

verb が非ブロッキング verb として発行されている場合、上記の値を値 AP_NON_BLOCKING と (論理 OR を使用して) 結合します。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id

会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc

AP_PARAMETER_CHECK

secondary_rc

次の値があります。

AP_BAD_CONV_ID

conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID

tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT

予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

AP_CONFIRMED_INVALID_FOR_FDX
ローカル TP は、全二重会話の中で [MC_]CONFIRMED verb を使用しようとしました。この verb は、半二重会話でのみ使用できます。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc

AP_CONFIRMED_BAD_STATE
会話の状態が、Confirm、Confirm_Send、Confirm_Deallocate のいずれでもありませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_CONVERSATION_TYPE_MIXED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は次のいずれかの状態でなければなりません。

- Confirm (確認)
- Confirm_Send (確認 _ 送信)
- Confirm_Deallocate (確認 _ 割り振り解除)

状態の変更

新しい状態は、古い状態、つまりローカル TP から [MC_]CONFIRMED verb を発行したときの会話の状態によって決まります。 古い状態は、直前の受信 verb の *what_rcvd* パラメーターの値によって示されます。 考えられる状態の変更を次の表に要約します。

古い状態	新しい状態
Confirm (確認)	Receive (受信)
Confirm_Send (確認 _ 送信)	Send (送信)
Confirm_Deallocate (確認 _ 割り振り解除)	Reset (リセット)

MC_DEALLOCATE および DEALLOCATE

MC_DEALLOCATE verb または DEALLOCATE verb は、 2 つの TP 間の会話の割り振りを解除します。

会話の割り振りを解除する前に、この verb は次のいずれかに相当する動作を実行します。

- ローカル LU の送信バッファーの内容をパートナー LU (および TP) へ送信する [MC_]FLUSH verb
- ローカル LU の送信バッファーの内容と確認要求をパートナー TP へ送信する [MC_]CONFIRM verb ([MC_]CONFIRM は半二重会話にのみ適用されます。)

この verb が正常に実行された後、会話 ID は無効になります。

VCB 構造体: MC_DEALLOCATE

AIX, LINUX

MC_DEALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_deallocate
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     expd_rcvd;
  unsigned char     dealloc_type;
  unsigned char     reserv4[2];
```

```
  unsigned char     reserv5[4];
  void              (*callback)();
  void *            correlator;
  unsigned char     reserv6[4];
} MC_DEALLOCATE;
```

VCB 構造体: DEALLOCATE

DEALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct deallocate
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     expd_rcvd;
  unsigned char     dealloc_type;
  AP_UINT16         log_dlen;
  unsigned char     *log_dptr;
  void              (*callback)();
  void *            correlator;
  unsigned char     reserv6[4];
} DEALLOCATE;
```

VCB 構造体: MC_DEALLOCATE (Windows)

WINDOWS

MC_DEALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_deallocate
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     reserv3;
  unsigned char     dealloc_type;
  unsigned char     reserv4[2];
  unsigned char     reserv5[4];
} MC_DEALLOCATE;
```

VCB 構造体: DEALLOCATE (Windows)

DEALLOCATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct deallocate
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     reserv3;
```

```
  unsigned char     dealloc_type;
  unsigned short    log_dlen;
  unsigned char far *log_dptr;
} DEALLOCATE;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_DEALLOCATE
MC_DEALLOCATE verb の場合

AP_B_DEALLOCATE
DEALLOCATE verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_DEALLOCATE verb の場合

AP_BASIC_CONVERSATION
DEALLOCATE verb の場合

verb が全二重会話で発行されている場合、または非ブロッキング verb として発行されている場合、上記の値を次の値のいずれかまたは両方と (論理 OR を使用して) 結合します。

AP_FULL_DUPLEX_CONVERSATION
この verb は全二重会話で発行されます。

AP_NON_BLOCKING
この verb は非ブロッキング verb として発行されます。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

dealloc_type
割り振り解除の実行方法を指定します。指定できる値は、下のリストに示すとおりです。

AP_ABEND 値または、いずれかの AP_ABEND_* 値を使用すると、異常処理として、会話の割り振り解除が行われます。会話が Send 状態にあるときにローカル TP から [MC_]DEALLOCATE verb を発行すると、APPC はローカル LU の送信バッファーの内容をパートナー TP へ送信してから会話の割

り振りを解除します。 会話が Receive または Pending-Post 状態であると、APPC は、会話を割り振り解除する前に、着信データをすべてパージします。

AP_ABEND
この値は、MC_DEALLOCATE verb にのみ有効です。 TP は、トランザクションの正常な終了を妨げるエラーを検出した場合、AP_ABEND を指定します。

AP_ABEND_PROG
この値は、DEALLOCATE verb にのみ有効です。アプリケーションまたはサービス TP は、トランザクションの正常な終了を妨げるエラーを検出した場合、AP_ABEND_PROG を指定します。

AP_ABEND_SVC
サービス TP は、パートナーのサービス TP が原因となったエラー (たとえば、パートナーのサービス TP から送信された制御情報の形式エラー) を検出した場合、AP_ABEND_SVC を指定します。

AP_ABEND_TIMER
サービス TP は、即時の割り振り解除を必要とするエラー (たとえば、オペレーターがプログラムを終了するのが早すぎた場合など) を検出した場合、AP_ABEND_TIMER を指定します。

AP_FLUSH
会話の割り振りを解除する前に、ローカル LU の送信バッファーの内容をパートナー TP へ送信します。 この値は、会話が Send 状態か Send_Pending 状態のときにのみ使用できます。

AP_CONFIRM_TYPE
この値は、会話の同期レベルが AP_SYNCPT の場合にのみ使用します。 この値は、会話の割り振りを解除する前に、パートナー TP からの確認が必要である (ただし、同期点処理は必要ない) ことを示します。

APPC は、ローカル LU の送信バッファーの内容と確認要求をパートナー TP へ送信します。 パートナー TP からの確認を受信すると、APPC は会話を割り振り解除します。 ただし、パートナー TP がエラーを報告した場合、会話は割り振られたままになります。

AP_SYNC_LEVEL
会話の同期レベル ([MC_]ALLOCATE verb によって確立されたもの) を使用して、会話の割り振り解除の方法を決定します。 この値は、会話が Send 状態か Send_Pending 状態のときにのみ使用できます。

会話の同期レベルが AP_NONE の場合、APPC はローカル LU の送信バッファーの内容をパートナー TP へ送信してから会話の割り振りを解除します。

同期レベルが AP_CONFIRM_SYNC_LEVEL の場合は、ローカル LU の送信バッファーの内容と確認要求が、APPC からパートナー TP に送信されます。パートナー TP からの確認を受信すると、APPC は

会話を割り振り解除します。 ただし、パートナー TP がエラーを報告した場合、会話は割り振られたままになります。

AIX, LINUX

会話の同期レベルが AP_SYNCPT の場合、APPC はローカル LU の送信バッファーの内容をパートナー TP へ送信してから会話の割り振りを解除します。同期点管理プログラムは、次のことを行います。

- *dealloc_type* が AP_SYNC_LEVEL に指定されたときに、[MC_]DEALLOCATE verb を代行受信する
- 必要な同期点処理を実行する
- 同期点処理が完了した時点で、オリジナルの [MC_]DEALLOCATE verb を Communications Server へそのまま渡す

Communications Server は、*sync_level* が AP_SYNCPT の会話で *dealloc_type* が AP_SYNC_LEVEL の [MC_]DEALLOCATE verb を受信した場合、すでに同期点管理プログラムが必要なすべての同期点処理を済ませているものと見なし、この verb を *sync_level* が AP_NONE であるものとして処理します。

AP_TP_NOT_AVAIL_RETRY
この値は、(すべての TP 名について着呼会話を受け入れるために) ブランクの TP 名を指定して RECEIVE_ALLOCATE を発行した TP にのみ使用してください。 この値は、着呼 Attach で指定された TP 名によって識別される TP が、一時的な条件のために使用不可であることを示しています。 パートナー TP へは、AP_ALLOCATION_FAILURE と AP_TRANS_PGM_NOT_AVAIL_RETRY の戻りコードでエラーが報告されます。パートナー TP は、割り振り要求を再試行できます。

AP_TP_NOT_AVAIL_NO_RETRY
この値は、(すべての TP 名について着呼会話を受け入れるために) ブランクの TP 名を指定して RECEIVE_ALLOCATE を発行した TP にのみ使用してください。 この値は、着呼 Attach で指定された TP 名によって識別される TP が、訂正を必要とする条件 (構成の問題など) のために使用不可であることを示しています。 パートナー TP へは、AP_ALLOCATION_FAILURE と AP_TRANS_PGM_NOT_AVAIL_NO_RETRY の戻りコードでエラーが報告されます。パートナー TP は、この割り振り解除の原因となった条件が訂正されるまで、割り振り要求を再試行してはなりません。

AP_TPN_NOT_RECOGNIZED
この値は、(すべての TP 名について着呼会話を受け入れるために) ブランクの TP 名を指定して RECEIVE_ALLOCATE を発行した TP にのみ使用してください。 この値は、着呼 Attach で指定された TP 名が、有効な TP 名として認識されなかったことを示してい

ます。 パートナー TP へは、AP_ALLOCATION_FAILURE と AP_TP_NAME_NOT_RECOGNIZED の戻りコードでエラーが報告されます。

AP_PIP_DATA_NOT_ALLOWED
この値は、パートナー TP が [MC_]ALLOCATE verb で PIP データを提供したにもかかわらず、ローカル TP が PIP データの受信を予期していなかったためにローカル TP が会話の割り振りを解除しようとしていることを示します。 パートナー TP へは、AP_ALLOCATION_FAILURE と AP_PIP_NOT_ALLOWED の戻りコードでエラーが報告されます。

AP_PIP_DATA_INCORRECT
この値は、ローカル TP がパートナー TP からの PIP データの受信を予期していたにもかかわらず、パートナー TP が誤った PIP データを提供したか PIP データを提供しなかったために、ローカル TP が会話の割り振りを解除しようとしていることを示します。 パートナー TP へは、AP_ALLOCATION_FAILURE と AP_PIP_NOT_SPECIFIED_CORRECTLY の戻りコードでエラーが報告されます。

AP_RESOURCE_FAILURE_NO_RETRY
この値は、ローカル TP の動作に必要なリソースに障害が起きたために、ローカル TP が会話の割り振りを解除しようとしていることを示します。

AP_CONV_TYPE_MISMATCH
この値は、パートナー TP によって [MC_]ALLOCATE で指定された会話タイプ (マップ式または基本) をローカル TP がサポートしていないため、ローカル TP が会話の割り振りを解除しようとしていることを示します。 パートナー TP へは、AP_ALLOCATION_FAILURE と AP_CONVERSATION_TYPE_MISMATCH の戻りコードでエラーが報告されます。

AP_SYNC_LVL_NOT_SUPPORTED
この値は、パートナー TP によって [MC_]ALLOCATE で指定された同期レベルをローカル TP がサポートしていないため、ローカル TP が会話の割り振りを解除しようとしていることを示します。 パートナー TP へは、AP_ALLOCATION_FAILURE と AP_SYNC_LEVEL_NOT_SUPPORTED の戻りコードでエラーが報告されます。

AP_SECURITY_PARAMS_INVALID
この値は、パートナー TP によって [MC_]ALLOCATE で指定された *security* パラメーターをローカル TP が受け入れなかったため、ローカル TP が会話の割り振りを解除しようとしていることを示します。 パートナー TP へは、AP_ALLOCATION_FAILURE と AP_SECURITY_NOT_VALID の戻りコードでエラーが報告されます。

log_dlen

エラー・ログ・ファイルへ送信するデータのバイト数。この値の範囲は、0 ～ 32,767 です。

このパラメーターは、*dealloc_type* パラメーターを AP_ABEND_PGM、AP_ABEND_SVC、AP_ABEND_TIMER のいずれかに設定した DEALLOCATE verb にのみ使用します。 MC_DEALLOCATE の場合、または *dealloc_type* がその他の値である場合、 このパラメーターは 0 (ゼロ) である必要があります。

log_dptr

エラー情報が入っているデータ・バッファーのアドレス。このデータは、ローカル・エラー・ログとパートナー LU へ送られます。

このパラメーターは、DEALLOCATE verb で *log_dlen* が 0 (ゼロ) より大きい場合に使用します。それ以外の場合、このパラメーターは予約済みです。

TP は、エラー・データを汎用データ・ストリーム (GDS) エラー・ログ変数として形式設定する必要があります。詳細については、IBM 資料の「*Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*」を参照してください。

WINDOWS

ログ・データ・バッファーは、静的データ域か、グローバル割り振り域にあります。データ・バッファーは、この領域と完全に一致していなければなりません。

AIX, LINUX

次のパラメーターは、会話の同期レベルが AP_SYNCPT の場合に使用され、それ以外の場合は予約済みです。

callback

TP が「暗黙 FORGET」という暗黙の指示を必要とする場合、このパラメーターは、その通知を提供するために Communications Server が呼び出すコールバック・ルーチンへのポインターを指定します。 TP がその通知を必要としない場合、TP はこのパラメーターを使用しません。 詳細については、150 ページの『暗黙 FORGET 通知』を参照してください。

correlator

アプリケーションで使用するオプションの相関係数。このパラメーターは、*callback* パラメーターを指定した場合にのみ使用し、それ以外の場合は予約済みです。

Communications Server はこの値を使用しませんが、この値を「暗黙 FORGET」という暗黙の通知と共にパラメーターとしてコールバック・ルー

チンに渡します。 この値をアプリケーションで使用すると、戻された情報をアプリケーションの別の処理へ関連付けることができます。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

expd_rcvd
優先データ・インディケーター。 このパラメーターは、全二重会話でのみ使用されます。TP は、[MC_]DEALLOCATE を正常に発行してから、優先データの受信を継続します。

次の値があります。

AP_YES パートナー TP が、ローカル TP がまだ受信していない優先データを送信しました。このデータを受信するには、ローカル TP は [MC_]RECEIVE_EXPEDITED_DATA verb を使用することができます。

このインディケーターを、APPC verb 数に対して設定できます。これは、ローカル TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行してデータを受信するまで、後続の verb に対しても設定できます。

AP_NO 受信を待機している優先データはありません。

半二重会話では、このパラメーターは常に AP_NO に設定されます。これは、[MC_]DEALLOCATE verb が正常に終了すると、会話が終了するため、ローカル TP はこれ以上優先データを受信できないためです。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_DEALLOC_BAD_TYPE
dealloc_type パラメーターが有効な値に設定されていませんでした。

AP_DEALLOC_LOG_LL_WRONG
(基本会話の DEALLOCATE でのみ戻ります) GDS エラー・ログ変数の LL フィールドがログ・データの実際の長さに一致しなかったか、*log_dlen* パラメーターの値が誤っていました。

AIX, LINUX

AP_INVALID_FORMAT
format パラメーターが無効な値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

WINDOWS

AP_INVALID_DATA_SEGMENT
(基本会話の DEALLOCATE のみに戻ります) ログ・データが割り振りデータ・セグメントより長かったか、ログ・データ・バッファーのアドレスが間違っていました。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc
次の値があります。

AP_DEALLOC_CONFIRM_BAD_STATE
会話が Send と Send_Pending のどちらの状態でもなく、TP が送信バッファーをフラッシュして確認要求を送信しようとしました。この試みが発生した原因は、*dealloc_type* パラメーターの値が AP_SYNC_LEVEL で、会話の同期レベルが AP_CONFIRM_SYNC_LEVEL であったことです。

AP_DEALLOC_FLUSH_BAD_STATE
会話が Send と Send_Pending のどちらの状態でもなく、TP が送信バッファーをフラッシュしようとしました。この試みが発生した原因は、*dealloc_type* パラメーターの値が AP_FLUSH であったことか、 *dealloc_type* パラメーターの値が AP_SYNC_LEVEL で、会話の同期レベルが AP_NONE であったことです。

AP_DEALLOC_NOT_LL_BDY
(基本会話の DEALLOCATE の場合にのみ戻ります) 会話が Send 状態にあり、TP が論理レコードの送信を終了しませんでした。*dealloc_type* パラメーターが AP_SYNC_LEVEL または AP_FLUSH に設定されていました。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc

AP_ALLOCATION_ERROR

secondary_rc

AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

AIX, LINUX

primary_rc
AP_BACKED_OUT

secondary_rc
AP_BO_NO_RESYNC
AP_BO_RESYNC

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_PROG_ERROR_PURGING
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、MC_DEALLOCATE verb が戻します。

primary_rc
AP_DEALLOC_ABEND

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、DEALLOCATE verb が戻します。

primary_rc
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP から [MC_]DEALLOCATE verb を発行した場合、会話は *dealloc_type* パラメーターの値に応じて、次の表に示した状態のいずれかになる可能性があります。

dealloc_type	許容される状態
AP_FLUSH	Send_Receive (送信 - 受信) (全二重会話専用)、 Send (送信) または Send_Pending (送信 - 保留)
AP_SYNC_LEVEL	Send_Receive (送信 - 受信) (全二重会話専用)、 Send (送信) または Send_Pending (送信 - 保留)
AP_ABEND	Reset (リセット) 以外のすべて
AP_ABEND_PROG	
AP_ABEND_SVC	
AP_ABEND_TIMER	

状態の変更

次の表に要約した状態変更は、*primary_rc* パラメーターの値に基づいています。

primary_rc	新しい状態
AP_OK	Receive_Only (*dealloc_type* を AP_FLUSH または AP_SYNC_LEVEL に設定した全二重会話)、または Reset (リセット) (他のすべての場合)
AP_PARAMETER_CHECK	変更なし
AP_STATE_CHECK	
AP_CONVERSATION_TYPE_MIXED	
AP_INVALID_VERB	
AP_INVALID_VERB_SEGMENT	
AP_STACK_TOO_SMALL	
AP_TP_BUSY	
AP_UNEXPECTED_DOS_ERROR	
AP_ALLOCATION_ERROR	Reset (リセット)
AP_CONV_FAILURE_RETRY	
AP_CONV_FAILURE_NO_RETRY	
AP_DEALLOC_ABEND	Reset (リセット)
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	
AP_PROG_ERROR_PURGING	Receive (受信)
AP_SVC_ERROR_PURGING	

暗黙 FORGET 通知

AIX, LINUX

同期点プロトコルには、「暗黙 FORGET」という機能が含まれています。これは、FORGET PS ヘッダー (同期点交換での最後のメッセージ) が必ずしも必要でないことを意味しています。 プロトコルがセッション上で次に受信するメッセージとして FORGET を要求すると、そのセッション上で受信した次のデータ・フローは、FORGET を受信したことを暗黙に意味します。

しかし、FORGET の前のメッセージで会話の割り振りが解除されようとしていることが示された場合、アプリケーションはそれ以上セッションにアクセスできなくなり、したがって次のデータ・フローがいつ発生するかを示すことができません。Communications Server では、その情報を提供するため、アプリケーションが [MC_]DEALLOCATE verb でコールバック・ルーチンを指定でき、Communications Server はセッション上に次のデータ・フローが発生するかセッションが終了 (正常終了か異常終了かは問わない) した時点で、そのルーチンを呼び出します。

この機能を使用するアプリケーションでは、その TP についての TP_ENDED verb を発行する前に、コールバック・ルーチンが呼び出されるのを待つ必要があります。 Communications Server は、TP_ENDED が発行された後では、コールバック・ルーチンを呼び出しません。

コールバック・ルーチンは、次のように定義されています。

```
void (*AP_CALLBACK) (
                        void *           vcb,
                        unsigned char    tp_id[8],
                        AP_UINT32        conv_id,
                        AP_UINT16        type,
                        AP_CORR          corr
                      );

typedef union ap_corr {
                         void *          corr_p;
                         AP_UINT32       corr_l;
                         AP_INT32        corr_i;
                       } AP_CORR;
```

Communications Server は、次のパラメーターを使用してルーチンを呼び出します。

vcb アプリケーションが提供したオリジナルの [MC_]DEALLOCATE VCB へのポインター。アプリケーションのコールバック・ルーチンの中で VCB パラメーターを使用する必要がある場合、コールバック・ルーチンの呼び出しが済むまで、アプリケーションで VCB に関連したメモリーを解放したり再利用したりしないでください。

tp_id verb を発行した TP の 8 バイトからなる TP ID。

conv_id
verb を発行した会話の会話 ID。アプリケーションは、この会話 ID を使用してさらに verb を発行することはできません。これは、[MC_]DEALLOCATE verb が完了した後、この会話 ID は有効でなくなるためです。

type Communications Server が報告しようとしているメッセージ・フローのタイプ。次の値があります。

AP_DATA_FLOW
セッション上の通常のデータ・フローです。

AP_UNBIND
セッションは正常に終了しました。

AP_FAILURE
セッションは異常終了しました。同期点管理プログラムは、再同期を実行しなければならない場合があります。

corr アプリケーションが提供した相関係数の値。この値をアプリケーションで使用すると、戻された情報をアプリケーションの別の処理へ関連付けることができます。

コールバック・ルーチンでは、これらのパラメーターをすべて使用する必要はありません。コールバック・ルーチンでは、戻された VCB に対して必要なすべての処理を実行できます。あるいは、通知を受信したことをメインプログラムに通知するために、単に変数を設定するだけでも構いません。

アプリケーションでシグナルによるスケジューリングを使用している場合、コールバック・ルーチンはシグナル・キャッチャーのコンテキストの中で実行されます。これは、ルーチン内で使用できるオペレーティング・システムのコールに制限があることを意味します。詳細については、オペレーティング・システムの資料を参照してください。

アプリケーションは、必要であればそのコールバック・ルーチンの中からさらに APPC verb を発行することができます。 ただし、それらの verb は非同期 verb である必要があります。コールバック・ルーチンの中から発行した同期 verb は、AP_PARAMETER_CHECK と AP_SYNC_NOT_ALLOWED の戻りコードでリジェクトされます。

MC_FLUSH および FLUSH

MC_FLUSH verb または FLUSH verb は、ローカル LU の送信バッファーの内容をパートナー LU (および TP) へ送信します。 送信バッファーが空の場合、アクションは起こりません。

バッファー・データのソース

[MC_]SEND_DATA verb によって処理されたデータと [MC_]ALLOCATE verb によって生成された割り振り要求は、次のいずれかが発生するまでローカル LU の送信バッファーの中に蓄積されます。

- ローカル TP が [MC_]FLUSH verb (または LU の送信バッファーをフラッシュする別の verb) を発行する
- バッファーがいっぱいになる

VCB 構造体: MC_FLUSH

AIX, LINUX

MC_FLUSH verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_flush
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;                 /* Reserved            */
  AP_UINT16         primary_rc;
```

```
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
} MC_FLUSH;
```

VCB 構造体: FLUSH

FLUSH verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct flush
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;                 /* Reserved             */
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
} FLUSH;
```

VCB 構造体: MC_FLUSH (Windows)

WINDOWS

MC_FLUSH verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_flush
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
} MC_FLUSH;
```

VCB 構造体: FLUSH (Windows)

FLUSH verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct flush
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
} FLUSH;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_FLUSH
MC_FLUSH verb の場合

AP_B_FLUSH
FLUSH verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_FLUSH verb の場合

AP_BASIC_CONVERSATION
FLUSH verb の場合

verb が全二重会話で発行されている場合、または非ブロッキング verb として発行されている場合、上記の値を次の値のいずれかまたは両方と (論理 OR を使用して) 結合します。

AP_FULL_DUPLEX_CONVERSATION
この verb は全二重会話で発行されます。

AP_NON_BLOCKING
この verb は非ブロッキング verb として発行されます。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT
予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc

AP_FLUSH_NOT_SEND_STATE
会話が Send と Send_Pending のどちらの状態でもありませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

```
AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT
```

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Send_Receive (全二重会話専用)、 Send または Send_Pending 状態でなければなりません。

状態の変更

実行が成功した場合、状態の変更はありません。

MC_GET_ATTRIBUTES および GET_ATTRIBUTES

MC_GET_ATTRIBUTES verb または GET_ATTRIBUTES verb は、会話の属性を戻します。 これらの属性についての詳細は、この資料の 1 ページの『第 1 章 概念』か、または「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。

VCB 構造体: MC_GET_ATTRIBUTES

AIX, LINUX

MC_GET_ATTRIBUTES verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_get_attributes
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;                 /* Reserved             */
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     reserv3;
  unsigned char     sync_level;
  unsigned char     mode_name[8];
  unsigned char     net_name[8];
  unsigned char     lu_name[8];
  unsigned char     lu_alias[8];
  unsigned char     plu_alias[8];
  unsigned char     plu_un_name[8];
  unsigned char     reserv4[2];
  unsigned char     fqplu_name[17];
  unsigned char     reserv5;
  unsigned char     user_id[10];
  AP_UINT32         conv_group_id;
  unsigned char     conv_corr_len;
  unsigned char     conv_corr[8];
```

```
  unsigned char     reserv6[13];
  LUWID_OVERLAY     luw_id;
  unsigned char     sess_id[8];
} MC_GET_ATTRIBUTES;

typedef struct luwid_overlay
{
  unsigned char     fq_length;
  unsigned char     fq_luw_name[17];
  unsigned char     instance[6];
  unsigned char     sequence[2];
} LUWID_OVERLAY;
```

VCB 構造体: GET_ATTRIBUTES

GET_ATTRIBUTES verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct get_attributes
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;                 /* Reserved             */
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     reserv3;
  unsigned char     sync_level;
  unsigned char     mode_name[8];
  unsigned char     net_name[8];
  unsigned char     lu_name[8];
  unsigned char     lu_alias[8];
  unsigned char     plu_alias[8];
  unsigned char     plu_un_name[8];
  unsigned char     reserv4[2];
  unsigned char     fqplu_name[17];
  unsigned char     reserv5;
  unsigned char     user_id[10];
  AP_UINT32         conv_group_id;
  unsigned char     conv_corr_len;
  unsigned char     conv_corr[8];
  unsigned char     reserv6[13];
  LUWID_OVERLAY     luw_id;
  unsigned char     sess_id[8];
} GET_ATTRIBUTES;

typedef struct luwid_overlay
{
  unsigned char     fq_length;
  unsigned char     fq_luw_name[17];
  unsigned char     instance[6];
  unsigned char     sequence[2];
} LUWID_OVERLAY;
```

VCB 構造体: MC_GET_ATTRIBUTES (Windows)

WINDOWS

MC_GET_ATTRIBUTES verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_get_attributes
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
```

```
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     reserv3;
  unsigned char     sync_level;
  unsigned char     mode_name[8];
  unsigned char     net_name[8];
  unsigned char     lu_name[8];
  unsigned char     lu_alias[8];
  unsigned char     plu_alias[8];
  unsigned char     plu_un_name[8];
  unsigned char     reserv4[2];
  unsigned char     fqplu_name[17];
  unsigned char     reserv5;
  unsigned char     user_id[10];
  unsigned long     conv_group_id;
  unsigned char     conv_corr_len;
  unsigned char     conv_corr[8];
  unsigned char     reserv6[13];
} MC_GET_ATTRIBUTES;
```

VCB 構造体: GET_ATTRIBUTES (Windows)

GET_ATTRIBUTES verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct get_attributes
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     reserv3;
  unsigned char     sync_level;
  unsigned char     mode_name[8];
  unsigned char     net_name[8];
  unsigned char     lu_name[8];
  unsigned char     lu_alias[8];
  unsigned char     plu_alias[8];
  unsigned char     plu_un_name[8];
  unsigned char     reserv4[2];
  unsigned char     fqplu_name[17];
  unsigned char     reserv5;
  unsigned char     user_id[10];
  unsigned long     conv_group_id;
  unsigned char     conv_corr_len;
  unsigned char     conv_corr[8];
  unsigned char     reserv6[13];
} GET_ATTRIBUTES;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_GET_ATTRIBUTES
MC_GET_ATTRIBUTES verb の場合

AP_B_GET_ATTRIBUTES
GET_ATTRIBUTES verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_GET_ATTRIBUTES verb の場合

AP_BASIC_CONVERSATION
GET_ATTRIBUTES verb の場合

verb が全二重会話で発行されている場合、上記の値と AP_FULL_DUPLEX_CONVERSATION を (論理 OR を使用して) 結合します。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。これらのパラメーターの意味と使用法についての詳細は、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。

primary_rc
AP_OK

sync_level
会話の同期レベル。このパラメーターは、TP がデータの受信の確認を要求できるかどうか、またデータの受信を確認できるかどうかを決定します。

次の値があります。

AP_CONFIRM_SYNC_LEVEL
各 TP は、この会話で確認処理を使用できます。

AP_SYNCPT
各 TP は、この会話で LU6.2 同期点機能を使用できます。 詳細については、 25 ページの『同期点サポート』を参照してください。

AP_NONE
この会話では、確認処理は使用されません。

mode_name
ネットワーク機能の特性セットの名前。

このパラメーターは、8 バイトの EBCDIC 文字ストリングです。 このパラメーターは、タイプ A の EBCDIC 文字セットの文字から構成できます。それらの文字は、次のとおりです。

- 英大文字
- 数字の 0 ～ 9
- 特殊文字の $、#、および ?

net_name

ローカル LU が入っているネットワークの名前。

このパラメーターは、8 バイトの EBCDIC 文字ストリングです。 このパラメーターは、タイプ A の EBCDIC 文字セットの文字から構成できます。それらの文字は、次のとおりです。

- 英大文字
- 数字の 0 ～ 9
- 特殊文字の $、#、および ?

lu_name

ローカル LU の名前。

このパラメーターは、8 バイトの EBCDIC 文字ストリングです。 このパラメーターは、タイプ A の EBCDIC 文字セットの文字から構成できます。それらの文字は、次のとおりです。

- 英大文字
- 数字の 0 ～ 9
- 特殊文字の $、#、および ?

lu_alias

ローカル LU をローカル TP に認識させるための別名。これは 8 バイトの ASCII 文字ストリングです。

plu_alias

パートナー LU をローカル TP に認識させるための別名。これは 8 バイトの ASCII 文字ストリングです。

plu_un_name

パートナー LU の非解釈名 (システム・サービス制御点 (SSCP) で定義されたとおりのパートナー LU の名前)。これは、Communications Server 構成ファイル内のリモート LU の構成から取ったものです。 このパラメーターが構成内に必要になるのは、ローカル LU が従属 LU である場合のみです。したがって、独立 LU について戻される名前はブランクまたはヌルであっても構いません。

このパラメーターは、 8 バイトの EBCDIC 文字ストリングで、大文字小文字の区別があります。このパラメーターは、タイプ AE の EBCDIC 文字セットの文字から構成できます。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 ～ 9
- 特殊文字の $、#、?、およびピリオド (.)

fqplu_name
パートナー LU の完全修飾名。

このフィールドには、ネットワーク名、EBCDIC のピリオド、およびパートナー LU 名が入っています。 2 つの名前は、それぞれ 8 バイトの EBCDIC 文字ストリングで、タイプ A の EBCDIC 文字セットの文字から構成できます。 それらの文字は、次のとおりです。

- 英大文字
- 数字の 0 ～ 9
- 特殊文字の $、#、および ?

user_id 呼び出し元 TP が呼び出し先 TP にアクセスするために [MC_]ALLOCATE verb によって送信したユーザー ID (該当する場合)。

このパラメーターは 10 バイトの EBCDIC 文字ストリングで、大文字小文字の区別があります。このパラメーターは、タイプ AE の EBCDIC 文字セットの文字から構成できます。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 ～ 9
- 特殊文字の $、#、?、およびピリオド (.)

次の条件が該当する場合、このフィールドにはユーザー ID が入っています。

- 呼び出し先 TP が会話セキュリティーを必要としている。
- この verb が呼び出し先 TP から発行された。

これ以外の場合、このフィールドにはブランクが入っています。

conv_group_id
この会話が使用するセッションの会話グループ ID。

conv_corr_len
会話相関係数 (詳細については、*conv_corr* パラメーターの説明を参照) の長さ (0 ～ 8 バイト) です。

conv_corr
この会話が割り振られたときに、呼び出し元 TP のノードが割り当てた会話相関係数。

AIX, LINUX

同期点処理を使用する TP の場合、同期点管理プログラムは再同期のときに、このパラメーターを使用して会話を識別します。

luw_id この会話が参加しているトランザクションの作業論理単位 ID (LUWID)。LUWID はトランザクションを開始した TP を代表するものとして割り当てられ、LUWID を使用すると、そのトランザクションを形成するさまざまな会話を相互に関連付けることができます。

同期点処理を使用する TP へは 2 つの LUWID が関連付けられています。*sync_level* が AP_NONE か AP_CONFIRM_SYNC_LEVEL の会話に使用する無保護の LUWID と、*sync_level* が AP_SYNCPT の会話に使用する保護付き LUWID です。 同期点処理を使用しない TP には、無保護の LUWID のみ

があります。 この verb は、この会話へ関連付けられている LUWID を戻します。会話の *sync_level* が AP_SYNCPT の場合、これは保護付きの LUWID であり、それ以外の場合は無保護の LUWID です。 アプリケーションは、GET_TP_PROPERTIES verb を使用して TP の両方の LUWID を取得できます。

LUWID は、次のパラメーターからなっています。

luw_id.fq_length
その作業論理単位へ関連付けられている完全修飾 LU 名の長さ (1 ～ 17 バイト) (LU 名自体は、次のパラメーターによって示されます)。

luw_id.fq_luw_name
作業論理単位に対応する LU の完全修飾名。この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれます。この名前は、1 ～ 8 文字の A ストリング文字からなるネットワーク ID、EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名で構成されます。

luw_id.instance
その作業論理単位へ関連付けられているインスタンス番号 (6 バイトの 2 進数)。

luw_id.sequence
その作業論理単位の現行セグメントの順序番号 (2 バイトの 2 進数)。

sess_id この会話が使用するセッションのセッション ID。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT
format パラメーターが無効な値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: この verb で状態検査エラーは発生しません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Reset 状態を除き、どのような状態にあっても構いません。

状態の変更

この verb では、会話状態は変更されません。

MC_PREPARE_TO_RECEIVE および PREPARE_TO_RECEIVE

MC_PREPARE_TO_RECEIVE verb または PREPARE_TO_RECEIVE verb は、ローカル TP の会話の状態を Send または Send_Pending から Receive に変更します。

注: この verb は、半二重会話でのみ使用できます。全二重会話では無効です。

会話状態を変更する前に、この verb は *ptr_type* (受信準備タイプ) パラメーターに応じて、次のいずれかの verb に相当する動作を実行します。

- ローカル LU の送信バッファーの内容をパートナー LU (および TP) へ送信する [MC_]FLUSH verb
- ローカル LU の送信バッファーの内容と確認要求をパートナー TP へ送信する [MC_]CONFIRM verb

この verb が正常に実行された後、ローカル TP はデータを受信できます。

VCB 構造体: MC_PREPARE_TO_RECEIVE

AIX, LINUX

MC_PREPARE_TO_RECEIVE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_prepare_to_receive
{
  AP_UINT16       opcode;
  unsigned char   opext;
  unsigned char   format;                  /* Reserved            */
  AP_UINT16       primary_rc;
  AP_UINT32       secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32       conv_id;
  unsigned char   ptr_type;
  unsigned char   locks;
} MC_PREPARE_TO_RECEIVE;
```

VCB 構造体: PREPARE_TO_RECEIVE

PREPARE_TO_RECEIVE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct prepare_to_receive
{
  AP_UINT16       opcode;
  unsigned char   opext;
  unsigned char   format;                  /* Reserved            */
  AP_UINT16       primary_rc;
  AP_UINT32       secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32       conv_id;
  unsigned char   ptr_type;
  unsigned char   locks;
} PREPARE_TO_RECEIVE;
```

VCB 構造体: MC_PREPARE_TO_RECEIVE (Windows)

WINDOWS

MC_PREPARE_TO_RECEIVE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_prepare_to_receive
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     ptr_type;
  unsigned char     locks;
} MC_PREPARE_TO_RECEIVE;
```

VCB 構造体: PREPARE_TO_RECEIVE (Windows)

PREPARE_TO_RECEIVE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct prepare_to_receive
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     ptr_type;
  unsigned char     locks;
} PREPARE_TO_RECEIVE;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_PREPARE_TO_RECEIVE
MC_PREPARE_TO_RECEIVE verb の場合

AP_B_PREPARE_TO_RECEIVE
PREPARE_TO_RECEIVE verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_PREPARE_TO_RECEIVE verb の場合

AP_BASIC_CONVERSATION
PREPARE_TO_RECEIVE verb の場合

verb が非ブロッキング verb として発行されている場合、上記の値を値 AP_NON_BLOCKING と (論理 OR を使用して) 結合します。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id

会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

ptr_type

状態変更の実行方法を指定します。

次の値があります。

AP_FLUSH

ローカル LU の送信バッファーの内容をパートナー LU (および TP) へ送信してから、会話の状態を Receive に変更します。

AIX, LINUX

AP_CONFIRM_TYPE

この値は、会話の同期レベルが AP_SYNCPT の場合にのみ使用します。 この値は、会話の状態を Receive に変更する前に、パートナー TP からの確認が必要である (ただし、同期点処理は必要ない) ことを示します。

APPC は、ローカル LU の送信バッファーの内容と確認要求をパートナー TP へ送信します。 会話状態は、パートナー TP が要求された確認を送信する (またはエラーを報告する) まで Receive に変更されません。

AP_SYNC_LEVEL

会話の同期レベル ([MC_]ALLOCATE verb によって確立されたもの) を使用して、状態変更の実行方法を決定します。

会話の同期レベルが AP_NONE の場合、APPC はローカル LU の送信バッファーの内容をパートナー TP へ送信してから会話の状態を Receive に変更します。

同期レベルが AP_CONFIRM_SYNC_LEVEL の場合は、ローカル LU の送信バッファーの内容と確認要求が、APPC からパートナー TP に送信されます。パートナー TP からの確認を受信すると、APPC は会話の状態を Receive に変更します。 ただし、パートナー TP がエラーを報告した場合、状態は Receive または Reset に変更されます。 170 ページの『状態の変更』を参照してください。

AIX, LINUX

会話の同期レベルが AP_SYNCPT の場合、APPC はローカル LU の送信バッファーの内容をパートナー TP へ送信してから会話の状態を変更します。同期点管理プログラムは、次のことを行います。

- *ptr_type* が AP_SYNC_LEVEL に指定されたときに、[MC_]PREPARE_TO_RECEIVE verb を代行受信する

- 必要な同期点処理を実行する
- 同期点処理が完了した時点で、オリジナルの [MC_]PREPARE_TO_RECEIVE verb を Communications Server へそのまま渡す

Communications Server は、*sync_level* が AP_SYNCPT の会話で *ptr_type* が AP_SYNC_LEVEL の [MC_]PREPARE_TO_RECEIVE verb を受信した場合、すでに同期点管理プログラムが必要なすべての同期点処理を済ませているものとみなし、この verb を *sync_level* が AP_NONE であるものとして処理します。

locks APPC がローカル TP に制御を戻す時期を指定します。

このパラメーターは、*ptr_type* が AP_SYNC_LEVEL に設定されており、[MC_]ALLOCATE verb によって確立された会話の同期レベルが AP_CONFIRM_SYNC_LEVEL である場合にのみ使用してください (それ以外の場合、このパラメーターは無視されます)。

次の値があります。

AP_LONG
APPC は、パートナー TP からローカル LU に確認とそれに続くデータが到着した時点で制御をローカル TP へ戻します (この方式では、ネットワークの使用効率が高くなりますが、ローカル TP へ制御が戻るのに要する時間が長くなります)。

AP_SHORT
APPC は、パートナー TP からローカル LU に確認が到着した時点で制御をローカル TP へ戻します。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_P_TO_R_INVALID_FOR_FDX
ローカル TP は、全二重会話の中で [MC_]PREPARE_TO_RECEIVE verb を使用しようとしました。この verb は、半二重会話でのみ使用できます。

AP_P_TO_R_INVALID_TYPE
ptr_type パラメーターが有効な値に設定されていませんでした。

AIX, LINUX

AP_INVALID_FORMAT
予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc
次の値があります。

AP_P_TO_R_NOT_LL_BDY
(基本会話の PREPARE_TO_RECEIVE の場合にのみ戻ります) ローカル TP が論理レコードの送信を終了しませんでした。

AP_P_TO_R_NOT_SEND_STATE
会話が Send と Send_Pending のどちらの状態でもありませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc

AP_ALLOCATION_ERROR

secondary_rc

AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

AIX, LINUX

primary_rc

AP_BACKED_OUT

secondary_rc

AP_BO_NO_RESYNC
AP_BO_RESYNC

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_CONV_FAILURE_NO_RETRY

AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_PROG_ERROR_PURGING
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、MC_PREPARE_TO_RECEIVE verb が戻します。

primary_rc
AP_DEALLOC_ABEND

次の 1 次戻りコードは、PREPARE_TO_RECEIVE verb が戻します。

primary_rc
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Send 状態か Send_Pending 状態でなければなりません。

状態の変更

次の表に要約した状態変更は、*primary_rc* パラメーターの値に基づいています。

primary_rc	**新しい状態**
AP_OK	Receive (受信)
AP_PARAMETER_CHECK	変更なし
AP_STATE_CHECK	
AP_CONVERSATION_TYPE_MIXED	
AP_INVALID_VERB	
AP_INVALID_VERB_SEGMENT	
AP_STACK_TOO_SMALL	
AP_TP_BUSY	
AP_UNEXPECTED_DOS_ERROR	
AP_ALLOCATION_ERROR	Reset (リセット)

primary_rc	新しい状態
AP_CONV_FAILURE_RETRY	Reset (リセット)
AP_CONV_FAILURE_NO_RETRY	
AP_DEALLOC_ABEND_RESET	Reset (リセット)
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	
AP_PROG_ERROR_PURGING_RECEIVE	Receive (受信)
AP_SVC_ERROR_PURGING	

使用上の注意

パートナー TP の場合、会話は、パートナー TP が後続の受信 verb の *what_rcvd* パラメーターによって次のいずれかの値を受信するまで、Send または Send_Pending 状態に変更されません。

- AP_SEND、AP_DATA_SEND、AP_DATA_COMPLETE_SEND
- AP_CONFIRM_SEND、AP_DATA_CONFIRM_SEND、または AP_DATA_COMPLETE_CONFIRM_SEND (および [MC_]CONFIRMED を伴った応答)

受信 verb は、[MC_]RECEIVE_AND_WAIT、[MC_]RECEIVE_IMMEDIATE、および [MC_]RECEIVE_AND_POST です。

MC_RECEIVE verb および RECEIVE verb

APPC には、パートナー TP からデータを受信するために使用する 3 つの異なる verb があります。ほとんどのパラメーターと戻りコードは、3 つの verb すべてに共通していますが、それぞれの動作方法は異なり、提供する機能も異なります。この節では、3 つの verb のすべてに適用される共通情報をまとめて説明し、その後に個々の verb を詳しく解説します。

3 つの受信 verb は、次のとおりです。

- [MC_]RECEIVE_IMMEDIATE
- [MC_]RECEIVE_AND_WAIT
- [MC_]RECEIVE_AND_POST

注: [MC_]RECEIVE_EXPEDITED_DATA verb も、パートナー TP からデータを受信しますが、通常フローのデータではなく、急送フローとして送信されたデータを受信します。この verb は、他の受信 verb の後に別個に説明されています。

TP によるデータの受信方法

ローカル TP がデータを受信するプロセスは、次のとおりです。

1. ローカル TP は、1 つのデータ単位を完全に受信し終わるまで、受信 verb を発行します。 受信されるデータは、次のいずれかです。
   - マップ式会話で伝送された 1 つのデータ・レコード
   - 基本会話で伝送された 1 つの論理レコード

- 基本会話で論理レコード形式とは関係なく受信されたデータのバッファー

ローカル TP は、1 つのデータ単位を完全に受信するために複数の受信 verb を発行しなければならない場合もあります。 データ単位を完全に受信した後、ローカル TP はそのデータ単位を操作できます。

2. ローカル TP は、別の受信 verb を発行します。これには、次のいずれかの効果があります。
   - パートナー TP がデータの続きを送信した場合、ローカル TP は新規のデータ単位の受信を開始します。
   - パートナー TP がデータの送信を終了するか、確認を待機している場合、状況情報 (*what_rcvd* パラメーターから入手できます) はローカル TP が通常実行する次のアクションを示しています。詳細については、『what_rcvd パラメーター』を参照してください。

別の方法として、ローカル TP は受信 verb を発行するときに、パラメーター *rtn_status* を設定できます。このパラメーターは、入手可能な状況情報があれば、データと共に戻すよう指示します。 この場合、データの最後の部分を戻した受信 verb も状況情報を戻すため、ローカル TP は状況情報を得るために別個に受信 verb を発行する必要がありません。 詳細については、『what_rcvd パラメーター』を参照してください。

what_rcvd パラメーター

いずれかの [MC_]RECEIVE verb を発行した後、TP は通常、 *what_rcvd* パラメーターを使用して次のアクションを決定します。「ユーザー制御」のデータ・タイプを参照する値は、 AIX または Linux システムのマップ式会話に戻されます。「PS ヘッダー」のデータ・タイプを参照する値は、AP_SYNCPT の同期レベルで AIX または Linux システムのマップ式会話に戻されます。

次のリストは、考えられる *what_rcvd* パラメーターの値と、それぞれの値で TP が通常実行するアクションを説明したものです。

AP_DATA AP_DATA_COMPLETE AP_DATA_INCOMPLETE AP_USER_CONTROL_DATA_COMPLETE AP_USER_CONTROL_DATA_INCMP AP_PS_HEADER_COMPLETE AP_PS_HEADER_INCOMPLETE
ローカル TP は、パートナー TP からデータを受信しました。ローカル TP は通常、このリストにある別の *what_rcvd* パラメーターの 1 つを受信するまで、受信 verb を発行し続けます。

AP_SEND (半二重会話専用)
パートナー TP は、確認を要求せずに [MC_]PREPARE_TO_RECEIVE verb を発行したか、PREPARE_TO_RECEIVE の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。ローカル TP は現在 Send 状態にあるため、通常、データの送信を開始します。

AP_CONFIRM_DEALLOCATE (半二重会話専用)
パートナー TP は、確認が必要であることを示す *dealloc_type* パラメーター付きで [MC_]DEALLOCATE verb を発行したか、DEALLOCATE の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。 ローカル TP は現在 Confirm_Deallocate 状態にあるため、通常、 [MC_]CONFIRMED verb を発行して会話の割り振り解除を確認します。

AP_CONFIRM_SEND (半二重会話専用)
パートナー TP は、確認が必要であることを示す *ptr_type* パラメーターと *dealloc_type* パラメーター付きで [MC_]PREPARE_TO_RECEIVE verb を発行したか、 PREPARE_TO_RECEIVE_CONFIRM の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。 ローカル TP は現在 Confirm_Send 状態にあるため、通常は、[MC_]CONFIRMED verb を発行して状態の変更を確認してからデータの送信を開始します。

AP_CONFIRM_WHAT_RECEIVED (半二重会話専用)
パートナー TP は、[MC_]CONFIRM verb を発行したか、 CONFIRM の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。ローカル TP は現在 Confirm 状態にあるため、通常、 [MC_]CONFIRMED verb を発行します。

次の値は、ローカル TP が *rtn_status* (データと共に状況を戻す) パラメーターに AP_YES を指定した場合にのみ戻ります。

AP_DATA_SEND AP_DATA_COMPLETE_SEND
AP_UC_DATA_COMPLETE_SEND AP_PS_HDR_COMPLETE_SEND
パートナー TP は、データの送信後、確認を要求せずに [MC_]PREPARE_TO_RECEIVE verb を発行したか、PREPARE_TO_RECEIVE の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。ローカル TP は現在 Send_Pending 状態にあるため、通常、データの送信を開始します。

AP_DATA_CONFIRM_DEALLOCATE
AP_DATA_COMPLETE_CONFIRM_DEALL
AP_UC_DATA_COMPLETE_CNFM_DEALL
AP_PS_HDR_COMLETE_CNFM_DEALL
これらの値はすべて半二重会話にのみ適用されます。

パートナー TP は、データの送信後、確認が必要であることを示す *dealloc_type* パラメーター付きで [MC_]DEALLOCATE verb を発行したか、DEALLOCATE の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。ローカル TP は現在 Confirm_Deallocate 状態にあるため、通常、 [MC_]CONFIRMED verb を発行して会話の割り振り解除を確認します。

AP_DATA_CONFIRM_SEND, AP_DATA_COMPLETE_CONFIRM_SEND,
AP_UC_DATA_COMPLETE_CNFM_SEND,
AP_PS_HDR_COMPLETE_CNFM_SEND
これらの値はすべて半二重会話にのみ適用されます。

パートナー TP は、データの送信後、確認が必要であることを示す *ptr_type* パラメーターと *dealloc_type* パラメーター付きで [MC_]PREPARE_TO_RECEIVE verb を発行したか、PREPARE_TO_RECEIVE_CONFIRM の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。 ローカル TP は現在 Confirm_Send 状態にあるため、通常は、[MC_]CONFIRMED verb を発行して状態の変更を確認してからデータの送信を開始します。

AP_DATA_CONFIRM, AP_DATA_COMPLETE_CONFIRM, AP_UC_DATA_COMPLETE_CONFIRM, AP_PS_HDR_COMPLETE_CONFIRM
これらの値はすべて半二重会話にのみ適用されます。

パートナー TP は、データの送信後、[MC_]CONFIRM verb を発行したか、CONFIRM の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。 ローカル TP は現在 Confirm 状態にあるため、通常、[MC_]CONFIRMED verb を発行します。

上記のすべての CONFIRM の場合、TP は提供されたデータの中か処理中にエラーを検出したことを示すために [MC_]CONFIRMED verb ではなく [MC_]SEND_ERROR verb を発行することもあります。 TP は、(AP_DATA_SEND、AP_DATA_COMPLETE_SEND 、AP_UC_DATA_COMPLETE_SEND、AP_PS_HDR_COMPLETE_SEND のいずれかを受信した後に) Send_Pending 状態で [MC_]SEND_ERROR を発行する場合、提供されたデータの中でエラーを検出したのか、あるいは TP 自身のデータまたは処理の中でエラーを検出したのかを示すことができます。 詳細については、 249 ページの『MC_SEND_ERROR および SEND_ERROR』で [MC_]SEND_ERROR verb の説明を参照してください。

データの終わり

ローカル TP が基本会話の受信 verb の 1 つを発行し、 *fill* パラメーターを AP_BUFFER に設定した場合、データの受信は *max_len* またはデータの終わりに到達した時点で終了します。 データの終わりは、次のいずれかによって示されます。

- AP_OK 以外の値 (たとえば、AP_DEALLOC_NORMAL) の *primary_rc* パラメーター
- SEND、CONFIRM、CONFIRM_SEND、CONFIRM_DEALLOCATE のいずれかが入った *what_rcvd* パラメーター

データの終わりに到達したかどうかを判別するため、ローカル TP は次のいずれかの受信 verb を再発行します。 新しい *primary_rc* パラメーターに AP_OK が入っており、*what_rcvd* に AP_DATA または AP_DATA_INCOMPLETE が入っている場合は、まだデータの終わりに到達していません。 しかし、データの終わりに到達した場合は、*primary_rc* パラメーターまたは *what_rcvd* パラメーターがデータの終わりの原因を示します。

what_rcvd パラメーターのテスト

ローカル TP は、データを何も受け取らなくても、いずれかの [MC_]RECEIVE verb を使用して、パートナー TP が送信するデータを保持しているかどうか、確認を求めているかどうか、会話状態を変更したかどうかを判別できます。それを行うには、*max_len* パラメーターを 0 (ゼロ) に設定して [MC_]RECEIVE verb を発行し、その後 ([MC_]RECEIVE verb が AP_OK の *primary_rc* 付きで戻ったら) *what_rcvd* パラメーターをテストします。

MC_RECEIVE_AND_POST および RECEIVE_AND_POST

MC_RECEIVE_AND_POST verb または RECEIVE_AND_POST verb は、アプリケーション・データと状況情報を非同期的に受信します。 これにより、TP はローカル LU がまだデータを受信中に処理を進めることができます。

注: この verb は、半二重会話でのみ使用できます。全二重会話では無効です。

VCB 構造体: MC_RECEIVE_AND_POST

AIX, LINUX

MC_RECEIVE_AND_POST verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_receive_and_post
{
  AP_UINT16       opcode;
  unsigned char   opext;
  unsigned char   format;                  /* Reserved             */
  AP_UINT16       primary_rc;
  AP_UINT32       secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32       conv_id;
  AP_UINT16       what_rcvd;
  unsigned char   rtn_status;
  unsigned char   reserv4;
  unsigned char   rts_rcvd;
  unsigned char   expd_rcvd;
  AP_UINT16       max_len;
  AP_UINT16       dlen;
  unsigned char   *dptr;
  void            (*callback)();
  unsigned char   reserv6;
} MC_RECEIVE_AND_POST;
```

VCB 構造体: RECEIVE_AND_POST

RECEIVE_AND_POST verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct receive_and_post
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                  /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  AP_UINT16        what_rcvd;
  unsigned char    rtn_status;
  unsigned char    fill;
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
  AP_UINT16        max_len;
  AP_UINT16        dlen;
  unsigned char    *dptr;
  void             (*callback)();
  unsigned char    reserv5;
} RECEIVE_AND_POST;
```

VCB 構造体: MC_RECEIVE_AND_POST (Windows)

WINDOWS

MC_RECEIVE_AND_POST verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_receive_and_post
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned short    what_rcvd;
  unsigned char     rtn_status;
  unsigned char     reserv4;
  unsigned char     rts_rcvd;
  unsigned char     reserv5;
  unsigned short    max_len;
  unsigned short    dlen;
  unsigned char far *dptr;
  unsigned char far *sema;
  unsigned char     reserv6;
} MC_RECEIVE_AND_POST;
```

VCB 構造体: RECEIVE_AND_POST (Windows)

RECEIVE_AND_POST verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct receive_and_post
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned short    what_rcvd;
  unsigned char     rtn_status;
  unsigned char     fill;
  unsigned char     rts_rcvd;
  unsigned char     reserv4;
  unsigned short    max_len;
  unsigned short    dlen;
  unsigned char far *dptr;
  unsigned char far *sema;
  unsigned char     reserv5;
} RECEIVE_AND_POST;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_RECEIVE_AND_POST
MC_RECEIVE_AND_POST verb の場合

AP_B_RECEIVE_AND_POST
RECEIVE_AND_POST verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_RECEIVE_AND_POST verb の場合

AP_BASIC_CONVERSATION
RECEIVE_AND_POST verb の場合

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

rtn_status
状況情報とデータを同じ verb で戻せるかどうかを示します。

次の値があります。

AP_YES 状況情報は、入手できれば、データ・レコードの最後の部分と共に戻ります。

AP_NO 状況情報は、データと共には戻りません。ローカル TP は、データ・レコードの終わりを受信した後に、再度 [MC_]RECEIVE verb を発行して状況情報を取得する必要があります。

fill ローカル TP がデータを受信する方法を示します。

このパラメーターは、基本会話の RECEIVE_AND_POST verb のみが使用します。

次の値があります。

AP_BUFFER
ローカル TP は、*max_len* パラメーターで指定したバイト数に到達するまで、またはデータの終わりに到達するまで、データを受信します。 データは、論理レコード形式に関係なく受信されます。

AP_LL データは、論理レコード形式で受信されます。受信されるデータは、次のいずれかです。

- 1 つの完全な論理レコード
- ある論理レコードの *max_len* バイトの部分
- 論理レコードの終わり

max_len
ローカル TP が受信できるデータの最大バイト数。

この値の範囲は、0 ～ 65,535 です。

この値は、受信したデータを入れるバッファーの長さを超えてはなりません。

dptr ローカル LU が受信したデータを入れるバッファーのアドレス。

AIX, LINUX

callback
非同期受信動作が終了したときに APPC が呼び出すコールバック・ルーチンのアドレス。詳細については、187 ページの『使用上の注意』を参照してください。

WINDOWS

sema 2 つの Windows 関数`CreateEvent` または `OpenEvent` のいずれかを呼び出して得られる Windows のイベント・ハンドル。APPC は、このイベント・ハンドルに信号を送り、非同期受信動作の終了を TP に知らせます。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

この verb が発行されると、この verb が正常に発行されたかどうかを示す *primary_rc* を伴って即時に戻ります。 この段階で有効な戻りパラメーターは、*primary_rc*、 *secondary_rc* (*primary_rc* が `AP_OK` でない場合)、および *rts_rcvd* のみです。 考えられる *primary_rc* と *secondary_rc* の値については、この項で後述します。

この *primary_rc* が `AP_OK` の場合、この verb はデータの非同期受信を正常に開始しました。 APPC は、この verb が (正常にデータを受信するか、会話エラーによって終了したために) 完了した時点で、指定されたコールバック・ルーチンを呼び出します。 この時点で、戻りパラメーターは次に示すとおりです。 *primary_rc* パラメーターと *secondary_rc* パラメーターには、データを正常に受信したかどうかを示す新しい値が入っており、これらのパラメーターを再度検査する必要があります。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
`AP_OK`

what_rcvd
着信データと共に受信した状況情報。

TP が次に実行するアクションは、通常このパラメーターの値によって決まります。 詳細については、172 ページの『what_rcvd パラメーター』を参照してください。

次の値があります。

AP_CONFIRM_DEALLOCATE
パートナー TP は、*dealloc_type* を AP_SYNC_LEVEL に設定して [MC_]DEALLOCATE verb を発行し、 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_CONFIRM_SEND
パートナー TP は、*ptr_type* を AP_SYNC_LEVEL に設定して [MC_]PREPARE_TO_RECEIVE verb を発行し、 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_CONFIRM_WHAT_RECEIVED
パートナー TP は [MC_]CONFIRM verb を発行しました。

AP_DATA
この値は、基本会話の RECEIVE_AND_POST で *fill* パラメーターを AP_BUFFER に設定した場合に、戻ることがあります。この値は、MC_RECEIVE_AND_POST には適用されません。

ローカル TP は、*max_len* またはデータの終わりに到達するまでデータを受信しました。

AP_DATA_COMPLETE
MC_RECEIVE_AND_POST の場合、この値はローカル TP が完全なデータ・レコードを受信したか、データ・レコードの最後の部分を受信したことを示します。 *fill* パラメーターを AP_LL に設定した RECEIVE_AND_POST の場合、この値はローカル TP が完全な論理レコードを受信したか、論理レコードの終わりを受信したことを示します。

AP_DATA_INCOMPLETE
MC_RECEIVE_AND_POST の場合、この値はローカル TP が不完全なデータ・レコードを受信したことを示します。 *max_len* パラメーターで、データ・レコードの長さより小さい (または、これがレコードを読み取るための最初の受信 verb でない場合はデータ・レコードの残りの部分より小さい) 値を指定しました。

fill パラメーターを AP_LL に設定した RECEIVE_AND_POST の場合、この値はローカル TP が不完全な論理レコードを受信したことを示します。

AP_SEND
パートナー TP の場合、会話は Receive 状態に入りました。 ローカル TP の場合、会話は Send 状態に入りました。

次の値は、*rtn_status* が AP_YES に設定された場合にのみ戻ります。

AP_DATA_CONFIRM
これは、AP_DATA と AP_CONFIRM_WHAT_RECEIVED を組み合わせたものです。 パートナー TP は、データの送信後、[MC_]CONFIRM verb を発行したか、CONFIRM の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。

AP_DATA_COMPLETE_CONFIRM
これは、AP_DATA_COMPLETE と AP_CONFIRM_WHAT_RECEIVED を組み合わせたものです。 パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、[MC_]CONFIRM verb を発行したか、CONFIRM の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。

AP_DATA_CONFIRM_DEALLOCATE
これは、AP_DATA と AP_CONFIRM_DEALLOCATE を組み合わせたものです。 パートナー TP は、データの送信後、 *dealloc_type* を AP_SYNC_LEVEL に設定して [MC_]DEALLOCATE verb を発行したか、DEALLOC_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_COMPLETE_CONFIRM_DEALL
これは、AP_DATA_COMPLETE と AP_CONFIRM_DEALLOCATE を組み合わせたものです。 パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、*dealloc_type* を AP_SYNC_LEVEL に設定して [MC_]DEALLOCATE verb を発行したか、DEALLOC_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_CONFIRM_SEND
これは、AP_DATA と AP_CONFIRM_SEND を組み合わせたものです。 パートナー TP は、データの送信後、 *ptr_type* を AP_SYNC_LEVEL に設定して [MC_]PREPARE_TO_RECEIVE verb を発行したか、P_TO_R_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_COMPLETE_CONFIRM_SEND
これは、AP_DATA_COMPLETE と AP_CONFIRM_SEND を組み合わせたものです。 パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、*ptr_type* を AP_SYNC_LEVEL に設定して [MC_]PREPARE_TO_RECEIVE verb を発行したか、P_TO_R_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_SEND
パートナー TP は、データの送信後、Receive 状態に入りました。 ローカル TP の場合、会話は Send_Pending 状態に入りました。

AP_DATA_COMPLETE_SEND
パートナー TP は、完全なデータ・レコード (またはデータ・レコ

ードの終わり) を送信した後、Receive 状態に入りました。 ローカル TP の場合、会話は Send_Pending 状態に入りました。

次の値は、MC_RECEIVE_AND_POST verb で戻ります。

AP_USER_CONTROL_DATA_INCMP
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_INCOMPLETE と同じです。

AP_USER_CONTROL_DATA_COMPLETE
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE と同じです。

AP_UC_DATA_COMPLETE_SEND
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_SEND と同じです。

AP_UC_DATA_COMPLETE_CONFIRM
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM と同じです。

AP_UC_DATA_COMPLETE_CNFM_DEALL
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_DEALL と同じです。

AP_UC_DATA_COMPLETE_CNFM_SEND
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_SEND と同じです。

次の値は、*sync_level* を AP_SYNCPT に設定した MC_RECEIVE_AND_POST verb で戻ります。

AP_PS_HEADER_INCOMPLETE
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_INCOMPLETE と同じです。

AP_PS_HEADER_COMPLETE
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE と同じです。

AP_PS_HDR_COMPLETE_SEND
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_SEND と同じです。

AP_PS_HDR_COMPLETE_CONFIRM
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM と同じです。

AP_PS_HDR_COMPLETE_CNFM_DEALL
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_DEALL と同じです。

AP_PS_HDR_COMPLETE_CNFM_SEND
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_SEND と同じです。

rts_rcvd

送信要求受信インディケーター。このパラメーターは、半二重会話でのみ使用できます。全二重会話では使用されません。

次の値があります。

AP_YES パートナー TP は、会話を Receive 状態に変更するようローカル TP に要求する [MC_]REQUEST_TO_SEND verb を発行しました。

AP_NO パートナー TP は [MC_]REQUEST_TO_SEND verb を発行しませんでした。

このインディケーターを受信 verb で受信できる理由については、222 ページの『MC_REQUEST_TO_SEND および REQUEST_TO_SEND』を参照してください。

expd_rcvd

優先データ・インディケーター。

次の値があります。

AP_YES パートナー TP が、ローカル TP がまだ受信していない優先データを送信しました。このデータを受信するには、ローカル TP は [MC_]RECEIVE_EXPEDITED_DATA verb を使用することができます。

このインディケーターを、APPC verb 数に対して設定できます。これは、ローカル TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行してデータを受信するまで、後続の verb に対しても設定できます。

AP_NO 受信を待機している優先データはありません。

dlen 受信したデータのバイト数 (データは、*dptr* パラメーターで指定されたバッファーに格納されます)。 長さ 0 (ゼロ) は、データを受信しなかったことを示します。 このパラメーターは、データを受信したことを *what_rcvd* パラメーターが示した場合にのみ使用されます。

会話の割り振りが解除された場合: パートナー TP が確認を要求せずに会話の割り振りを解除した場合、 APPC は次のパラメーターを戻します。

primary_rc

AP_DEALLOC_NORMAL
パートナー TP は、*dealloc_type* を次のいずれかに設定して [MC_]DEALLOCATE verb を発行しました。

- AP_FLUSH
- 会話の同期レベルが AP_NONE として指定された AP_SYNC_LEVEL

dlen 受信したデータのバイト数 (データは、*dptr* パラメーターで指定されたバッファーに格納されます)。 長さ 0 (ゼロ) は、データを受信しなかったことを示します。 このパラメーターは、*rtn_status* を AP_YES に設定した場合にのみ使用されます。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_RETURN_STATUS_WITH_DATA
rtn_status パラメーターが無効な値に設定されていました。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_INVALID_FORMAT
予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

AP_INVALID_CALLBACK_HANDLE
callback パラメーターがヌル・ポインターに設定されており、同期エントリー・ポイントを使用して (またはコールバック・ルーチンへのヌル・ポインターと共に非同期エントリー・ポイントを使用して) verb が発行されました。 詳細については、 187 ページの『使用上の注意』を参照してください。

AP_RCV_AND_POST_BAD_FILL
この戻りコードは、基本会話の RECEIVE_AND_POST verb にのみ適用されます。 *fill* パラメーターが無効な値に設定されていました。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc
次の値があります。

AP_RCV_AND_POST_BAD_STATE
TP がこの verb を発行したとき、会話は Receive、Send、Send_Pending のいずれの状態でもありませんでした。

AP_RCV_AND_POST_NOT_LL_BDY
この戻りコードは、基本会話の RECEIVE_AND_POST verb にのみ適用されます。会話が Send 状態にありました。TP は開始されましたが、論理レコードの送信を終了しませんでした。

verb が取り消された場合: この戻りコードは、初期戻りコードとして戻ることはなく、初期戻りコードが AP_OK の場合に、それに続く戻りコードとしてのみ戻ります。

TP が発行した別の verb によってこの verb が取り消されたためにこの verb が実行されなかった場合、APPC は次のパラメーターを戻します。

primary_rc

AP_CANCELLED
ローカル TP は、Pending_Post 状態のときに、次のいずれかの verb を発行しました。

- *dealloc_type* が AP_ABEND_PROG、AP_ABEND_SVC、または AP_ABEND_TIMER に設定された DEALLOCATE
- *dealloc_type* が AP_ABEND に設定された MC_DEALLOCATE
- [MC_]SEND_ERROR
- TP_ENDED

Pending_Post 状態のときに、これらの verb の 1 つを発行すると、[MC_]RECEIVE_AND_POST verb は取り消されます。 コールバック・ルーチンは呼び出されません。ローカル TP は、もはやパートナー TP からデータを非同期的に受信していません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc

AP_ALLOCATION_ERROR

secondary_rc

AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY

AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

primary_rc

AP_BACKED_OUT

secondary_rc

AP_BO_NO_RESYNC
AP_BO_RESYNC

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_PROG_ERROR_NO_TRUNC
AP_PROG_ERROR_PURGING
AP_PROG_ERROR_TRUNC
AP_INVALID_VERB
AP_TP_BUSY

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、MC_RECEIVE_AND_POST が戻します。

primary_rc

AP_DEALLOC_ABEND

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、RECEIVE_AND_POST が戻します。

primary_rc

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_NO_TRUNC
AP_SVC_ERROR_PURGING
AP_SVC_ERROR_TRUNC

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP は、会話が Receive、Send、Send_Pending のいずれかの状態のときに [MC_]RECEIVE_AND_POST を発行できます。

Send 状態での verb の発行

会話が Send 状態のときに [MC_]RECEIVE_AND_POST verb を発行すると、次の効果があります。

- ローカル LU は、その送信バッファー内の情報と SEND インディケーターをパートナー TP へ送信します。
- 会話は Pending_Post 状態に変更され、ローカル TP はパートナー TP からの情報を非同期的に受信できる状態になります。

状態の変更

会話状態は 2 回変更されます。この verb の初期の戻りで、 *primary_rc* が AP_OK であれば、会話は Pending_Post 状態に変更されます。 APPC がコールバック・ルーチンを呼び出すか、verb の完了を示すためにセマフォーをクリアした場合、状態は、この項で説明するとおりに変更されます。

[MC_]RECEIVE_AND_POST の完了時の状態変更は、次の値によって異なります。

- *primary_rc* パラメーター
- *primary_rc* が AP_OK の場合は、 *what_rcvd* パラメーター

次の表に、*primary_rc* が AP_OK の場合に考えられる状態の変更を要約します。

what_rcvd パラメーター	新しい状態
AP_CONFIRM_WHAT_RECEIVED	Confirm (確認)
AP_DATA_CONFIRM	
AP_DATA_COMPLETE_CONFIRM	
AP_CONFIRM_DEALLOCATE	Confirm_Deallocate (確認 _ 割り振り解除)
AP_DATA_CONFIRM_DEALLOCATE	
AP_DATA_COMPLETE_CONFIRM_DEALL	
AP_CONFIRM_SEND	Confirm_Send (確認 _ 送信)
AP_DATA_CONFIRM_SEND	
AP_DATA_COMPLETE_CONFIRM_SEND	
AP_DATA	Receive (受信)
AP_DATA_COMPLETE	
AP_DATA_INCOMPLETE	
AP_SEND	Send (送信)
AP_DATA_SEND	Send_ Pending (送信 _ 保留)
AP_DATA_COMPLETE_SEND	

次の表に、*primary_rc* が AP_OK ではない場合に考えられる状態の変更を要約します。

primary_rc	新しい状態
AP_PARAMETER_CHECK AP_STATE_CHECK AP_CONVERSATION_TYPE_MIXED	変更なし (これらの戻りコードは、2 番目でなく最初の戻りコードとしてのみ発生します)
AP_INVALID_VERB AP_INVALID_VERB_SEGMENT	変更なし
AP_STACK_TOO_SMALL AP_TP_BUSY AP_UNEXPECTED_DOS_ERROR	変更なし
AP_CONV_FAILURE_RETRY AP_CONV_FAILURE_NO_RETRY AP_DEALLOC_ABEND AP_DEALLOC_ABEND_PROG AP_DEALLOC_ABEND_SVC AP_DEALLOC_ABEND_TIMER AP_DEALLOC_NORMAL	Reset (リセット)
AP_PROG_ERROR_PURGING AP_PROG_ERROR_NO_TRUNC AP_SVC_ERROR_PURGING AP_SVC_ERROR_NO_TRUNC AP_PROG_ERROR_TRUNC AP_SVC_ERROR_TRUNC	Receive (受信)
AP_CANCELLED	会話は、[MC_]RECEIVE_AND_POST verb が発行された状態 (Send または Receive) に戻ります。 AP_CANCELLED は、同じ TP から発行された別の verb によって発生した戻りコードであるため、会話状態は、後の方の verb が完了した時点で再度変更されます。

使用上の注意

この項では、次のトピックについて、使用上の追加情報を記載します。

- PS ヘッダー・データ
- コールバック・ルーチン
- verb が保留中の処理
- 他のインプリメンテーション形態の APPC との互換性
- TP での verb の使用法
- 無限待機の回避

PS ヘッダー・データ

同期レベルが AP_SYNCPT の会話では、受信データが PS ヘッダーの形式になっている場合があります。 マップ式会話では、これは *what_rcvd* パラメーターの値によって示されます。基本会話では、0x0001 という LL フィールドによって示されます (詳細については、 67 ページの『論理レコード』を参照)。 同期点管理プログラムは、データを適切な同期点コマンドに変換します。

コールバック・ルーチン

AIX, LINUX

アプリケーションは、VCB のパラメーターの 1 つとしてコールバック・ルーチンへのポインターを提供します。この項では、Communications Server がそのルーチンをどのように使用するか、および、そのルーチンが実行する必要がある機能について説明します。

コールバック・ルーチンは、次のように定義されています。

```
void (*callback) (
                  void *          vcb,
                  unsigned char   tp_id[8],
                  AP_UINT32       conv_id
                 );
```

Communications Server は、次のパラメーターを使用してルーチンを呼び出します。

vcb アプリケーションが提供した VCB へのポインター (Communications Server が設定した戻りパラメーターを含む)。

tp_id verb を発行した TP の 8 バイトからなる TP ID。

conv_id
verb を発行した会話の会話 ID。

コールバック・ルーチンでは、これらのパラメーターをすべて使用する必要はありません。コールバック・ルーチンは、戻された VCB に対して必要なすべての処理を実行するか、単に verb が完了したことをメインプログラムに通知するために変数を設定することもできます。

アプリケーションは、必要であればそのコールバック・ルーチンの中からさらに APPC verb を発行することができます。 ただし、それらの verb は非同期 verb である必要があります。コールバック・ルーチンの中から発行した同期 verb は、AP_PARAMETER_CHECK と AP_SYNC_NOT_ALLOWED の戻りコードでリジェクトされます。

アプリケーションが非同期 APPC エントリー・ポイントを使用して [MC_]RECEIVE_AND_POST verb を発行する場合は、2 つのコールバック・ルーチンが指定されます。1 つは VCB の中で、もう 1 つはエントリー・ポイントのパラメーターとして指定されます。一般に、APPC は VCB の中で指定されたコールバック・ルーチンを使用し、エントリー・ポイントのコールバック・ルーチンは無視します。しかし、アプリケーションが VCB 内のコールバック・ルーチンにヌル・ポインターを指定した場合、APPC はエントリー・ポイントのコールバック・ルーチンを使用します。

verb が保留中の他の処理の続行

[MC_]RECEIVE_AND_POST verb はデータの到着を待機せずに即時に戻るため、TP はこの verb の完了を待機する間、他の処理を続行できます。ただし、次の点に注意してください。

- [MC_]RECEIVE_AND_POST verb へ提供された VCB は、コールバック・ルーチンが戻るまで、引き続き使用されます。 その間、TP で VCB 内のフィールドに変更を加えてはなりません。 Pending_Post 状態の間に、TP から別の APPC verb を発行する場合は、その verb に別の VCB を使用する必要があります。
- 1 つの会話につき、一度に 1 つの [MC_]RECEIVE_AND_POST verb のみを活動状態にできます。

他のインプリメンテーション形態の APPC との互換性

AIX, LINUX

[MC_]RECEIVE_AND_POST verb の AIX または Linux インプリメンテーションは、 Windows の APPC インプリメンテーションと異なります。また、この verb は DOS にインプリメントされた APPC では使用できません。このため、[MC_]RECEIVE_AND_POST を使用する TP は、他のオペレーティング・システムに完全に移植できるとは限りません。この verb を TP で使用する場合、その TP を別のオペレーティング・システム上で実行したければ、この verb を使用する TP のセクションを書き直す必要があります。

TP での verb の使用法

[MC_]RECEIVE_AND_POST verb を使用するには、ローカル TP で次の手順を実行します。

1. [MC_]RECEIVE_AND_POST verb を発行します。
2. 次のように、1 次戻りコード *primary_rc* の値を検査します。

   1 次戻りコードが AP_OK である場合、受信バッファー (*dptr* パラメーターによって示されます) は、パートナー TP から非同期的にデータを受信しています。ローカル TP は、データを非同期的に受信している間、次のことを実行できます。

   - その会話に関連していないタスクを実行する
   - [MC_]REQUEST_TO_SEND verb を発行する
   - 次の verb を発行することにより、その会話に関する情報を収集する
     - GET_TYPE
     - [MC_]GET_ATTRIBUTES
     - [MC_]TEST_RTS
   - 次のいずれかの verb を発行することにより、 [MC_]RECEIVE_AND_POST verb を途中で取り消す

- *dealloc_type* が AP_ABEND_PROG、AP_ABEND_SVC、または AP_ABEND_TIMER に設定された DEALLOCATE
- *dealloc_type* が AP_ABEND に設定された MC_DEALLOCATE
- SEND_ERROR
- TP_ENDED

3. コールバック・ルーチン (この verb でパラメーターとして指定されたもの) が APPC によって呼び出されたかどうかを検査します。 TP がデータの非同期受信を終了すると、APPC は、このコールバック・ルーチンを呼び出します。
4. 1 次戻りコード *primary_rc* の新しい値を検査します。

   1 次戻りコードが AP_OK の場合、ローカル TP はその他の戻りパラメーターを検査でき、非同期に受信したデータを操作できます。

   1 次戻りコードが AP_OK でない場合は、*secondary_rc* パラメーターと *rts_rcvd* (送信要求受信) パラメーターのみに意味があります。

無限待機の回避

AIX, LINUX

ローカル TP が [MC_]RECEIVE_AND_POST verb を発行し、その後、コールバック・ルーチンが呼び出されるのを待つ場合、ローカル TP はパートナー TP からの情報を受信するまで中断状態になります。パートナー TP が、何の情報も送信しないか、パートナー LU に送信バッファーをフラッシュさせる verb を発行しなければ、ローカル TP は無限に待機する可能性があります。 TP を切れ目なく動作させる必要がある場合は、コールバック・ルーチンを待機しないようにするか、[MC_]RECEIVE_IMMEDIATE verb を使用します。

MC_RECEIVE_AND_WAIT および RECEIVE_AND_WAIT

MC_RECEIVE_AND_WAIT verb および RECEIVE_AND_WAIT verb は、パートナー TP から現在入手可能なデータを受信します。現在入手可能なデータがない場合、ローカル TP はデータが到着するまで待機します。

非同期 [MC_]RECEIVE_AND_WAIT が未解決の間、アプリケーションは同じ会話に対して次の verb を発行できます。

- GET_TYPE
- 割り振り解除タイプが AP_ABEND、AP_ABEND_PROG、AP_ABEND_SVC、または AP_ABEND_TIMER である [MC_]DEALLOCATE
- [MC_]GET_ATTRIBUTES
- 非ブロッキング・モードで発行されるという条件での追加 [MC_]RECEIVE verb
- [MC_]RECEIVE_EXPEDITED_DATA
- [MC_]REQUEST_TO_SEND

- [MC_]SEND_DATA (全二重会話専用)
- [MC_]SEND_EXPEDITED_DATA
- [MC_]SEND_ERROR
- [MC_]TEST_RTS
- TP_ENDED

VCB 構造体: MC_RECEIVE_AND_WAIT

AIX, LINUX

MC_RECEIVE_AND_WAIT verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_receive_and_wait
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                  /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  AP_UINT16        what_rcvd;
  unsigned char    rtn_status;
  unsigned char    reserv4;
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
  AP_UINT16        max_len;
  AP_UINT16        dlen;
  unsigned char    *dptr;
  unsigned char    reserv6[5];
} MC_RECEIVE_AND_WAIT;
```

VCB 構造体: RECEIVE_AND_WAIT

RECEIVE_AND_WAIT verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct receive_and_wait
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;               /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  AP_UINT16        what_rcvd;
  unsigned char    rtn_status;
  unsigned char    fill;
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
    AP_UINT16        max_len;
  AP_UINT16        dlen;
  unsigned char    *dptr;
  unsigned char    reserv5[5];
} RECEIVE_AND_WAIT;
```

VCB 構造体: MC_RECEIVE_AND_WAIT (Windows)

WINDOWS

MC_RECEIVE_AND_WAIT verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_receive_and_wait
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned short    what_rcvd;
  unsigned char     rtn_status;
  unsigned char     reserv4;
  unsigned char     rts_rcvd;
  unsigned char     reserv5;
  unsigned short    max_len;
  unsigned short    dlen;
  unsigned char far *dptr;
  unsigned char     reserv6[5];
} MC_RECEIVE_AND_WAIT;
```

VCB 構造体: RECEIVE_AND_WAIT (Windows)

RECEIVE_AND_WAIT verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct receive_and_wait
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned short    what_rcvd;
  unsigned char     rtn_status;
  unsigned char     fill;
  unsigned char     rts_rcvd;
  unsigned char     reserv4;
  unsigned short    max_len;
  unsigned short    dlen;
  unsigned char far *dptr;
  unsigned char     reserv5[5];
} RECEIVE_AND_WAIT;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_RECEIVE_AND_WAIT
MC_RECEIVE_AND_WAIT verb の場合

AP_B_RECEIVE_AND_WAIT
RECEIVE_AND_WAIT verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_RECEIVE_AND_WAIT verb の場合

AP_BASIC_CONVERSATION
RECEIVE_AND_WAIT verb の場合

verb が全二重会話で発行されている場合、または非ブロッキング verb として発行されている場合、上記の値を次の値のいずれかまたは両方と (論理 OR を使用して) 結合します。

AP_FULL_DUPLEX_CONVERSATION
この verb は全二重会話で発行されます。

AP_NON_BLOCKING
この verb は非ブロッキング verb として発行されます。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

rtn_status
状況情報とデータを同じ verb で戻せるかどうかを示します。 次の値があります。

AP_YES 状況情報は、入手できれば、データ・レコードの最後の部分と共に戻ります。

AP_NO 状況情報は、データと共には戻りません。ローカル TP は、データ・レコードの終わりを受信した後に、再度 [MC_]RECEIVE verb を発行して状況情報を取得する必要があります。

fill ローカル TP がデータを受信する方法を示します。

このパラメーターは、基本会話の RECEIVE_AND_WAIT verb のみが使用します。 次の値があります。

AP_BUFFER
ローカル TP は、*max_len* パラメーターで指定したバイト数に到達するまで、またはデータの終わりに到達するまで、データを受信します。 データは、論理レコード形式に関係なく受信されます。

AP_LL データは、論理レコード形式で受信されます。受信されるデータは、次のいずれかです。

- 1 つの完全な論理レコード
- ある論理レコードの *max_len* バイトの部分
- 論理レコードの終わり

max_len
ローカル TP が受信できるデータの最大バイト数。

この値の範囲は、0 ～ 65,535 です。

この値は、受信したデータを入れるバッファーの長さを超えてはなりません。

dptr ローカル LU が受信したデータを入れるバッファーのアドレス。

WINDOWS

データ・バッファーは、静的データ域か、グローバル割り振り域にあります。データ・バッファーは、この領域と完全に一致していなければなりません。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

what_rcvd
着信データと共に受信した状況情報。

TP が次に実行するアクションは、通常このパラメーターの値によって決まります。 詳細については、 172 ページの『what_rcvd パラメーター』を参照してください。

次の値があります。

AP_CONFIRM_DEALLOCATE
この値は、半二重会話でのみ戻ります。

パートナー TP は、*dealloc_type* を AP_SYNC_LEVEL に設定して [MC_]DEALLOCATE verb を発行し、 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_CONFIRM_SEND
この値は、半二重会話でのみ戻ります。

パートナー TP は、*ptr_type* を AP_SYNC_LEVEL に設定して [MC_]PREPARE_TO_RECEIVE verb を発行し、 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_CONFIRM_WHAT_RECEIVED
この値は、半二重会話でのみ戻ります。

パートナー TP は [MC_]CONFIRM verb を発行しました。

AP_DATA
この値は、基本会話の RECEIVE_AND_WAIT verb で *fill* パラメーターを AP_BUFFER に設定した場合に、戻ることがあります。この値は、MC_RECEIVE_AND_WAIT には適用されません。

ローカル TP は、*max_len* またはデータの終わりに到達するまでデータを受信しました。

AP_DATA_COMPLETE
マップ式会話の MC_RECEIVE_AND_WAIT verb の場合、この値はローカル TP が完全なデータ・レコードを受信したか、データ・レコードの最後の部分を受信したことを示します。

fill パラメーターを AP_LL に設定した基本会話の RECEIVE_AND_WAIT verb の場合、この値はローカル TP が完全な論理レコードを受信したか、論理レコードの終わりを受信したことを示します。

AP_DATA_INCOMPLETE
マップ式会話の MC_RECEIVE_AND_WAIT verb の場合、この値はローカル TP が不完全なデータ・レコードを受信したことを示します。 *max_len* パラメーターで、データ・レコードの長さより小さい (または、これがレコードを読み取るための最初の受信 verb でない場合はデータ・レコードの残りの部分より小さい) 値を指定しました。

fill パラメーターを AP_LL に設定した基本会話の RECEIVE_AND_WAIT verb の場合、この値はローカル TP が不完全な論理レコードを受信したことを示します。

AP_SEND
この値は、半二重会話でのみ戻ります。

パートナー TP の場合、会話は Receive 状態に入りました。 ローカル TP の場合、会話は Send 状態に入りました。

次の値は、半二重会話と、*rtn_status* が AP_YES に設定された場合にのみ戻ります。

AP_DATA_CONFIRM
これは、AP_DATA と AP_CONFIRM_WHAT_RECEIVED を組み合わせたものです。 パートナー TP は、データの送信後、[MC_]CONFIRM verb を発行したか、CONFIRM の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。

AP_DATA_COMPLETE_CONFIRM
これは、AP_DATA_COMPLETE と AP_CONFIRM_WHAT_RECEIVED を組み合わせたものです。 パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、[MC_]CONFIRM verb を発行したか、CONFIRM の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。

AP_DATA_CONFIRM_DEALLOCATE
これは、AP_DATA と AP_CONFIRM_DEALLOCATE を組み合わせたものです。 パートナー TP は、データの送信後、 *dealloc_type* を AP_SYNC_LEVEL に設定して [MC_]DEALLOCATE verb を発行したか、DEALLOC_SYNC_LEVEL の送信タイプを指定して

[MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_COMPLETE_CONFIRM_DEALL
これは、AP_DATA_COMPLETE と AP_CONFIRM_DEALLOCATE を組み合わせたものです。 パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、*dealloc_type* を AP_SYNC_LEVEL に設定して [MC_]DEALLOCATE verb を発行したか、DEALLOC_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_CONFIRM_SEND
これは、AP_DATA と AP_CONFIRM_SEND を組み合わせたものです。 パートナー TP は、データの送信後、 *ptr_type* を AP_SYNC_LEVEL に設定して [MC_]PREPARE_TO_RECEIVE verb を発行したか、P_TO_R_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_COMPLETE_CONFIRM_SEND
これは、AP_DATA_COMPLETE と AP_CONFIRM_SEND を組み合わせたものです。 パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、*ptr_type* を AP_SYNC_LEVEL に設定して [MC_]PREPARE_TO_RECEIVE verb を発行したか、P_TO_R_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_SEND
パートナー TP は、データの送信後、Receive 状態に入りました。 ローカル TP の場合、会話は Send_Pending 状態に入りました。

AP_DATA_COMPLETE_SEND
パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、Receive 状態に入りました。 ローカル TP の場合、会話は Send_Pending 状態に入りました。

AIX, LINUX

次の値は、MC_RECEIVE_AND_WAIT verb で戻ります。

AP_USER_CONTROL_DATA_INCMP
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_INCOMPLETE と同じです。

AP_USER_CONTROL_DATA_COMPLETE
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE と同じです。

AP_UC_DATA_COMPLETE_SEND
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_SEND と同じです。

AP_UC_DATA_COMPLETE_CONFIRM
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM と同じです。

AP_UC_DATA_COMPLETE_CNFM_DEALL
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_DEALL と同じです。

AP_UC_DATA_COMPLETE_CNFM_SEND
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_SEND と同じです。

次の値は、*sync_level* を AP_SYNCPT に設定した MC_RECEIVE_AND_WAIT verb で戻ります。

AP_PS_HEADER_INCOMPLETE
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_INCOMPLETE と同じです。

AP_PS_HEADER_COMPLETE
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE と同じです。

AP_PS_HDR_COMPLETE_SEND
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_SEND と同じです。

AP_PS_HDR_COMPLETE_CONFIRM
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM と同じです。

AP_PS_HDR_COMPLETE_CNFM_DEALL
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_DEALL と同じです。

AP_PS_HDR_COMPLETE_CNFM_SEND
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_SEND と同じです。

rts_rcvd
送信要求受信インディケーター。このパラメーターは、半二重会話でのみ使用できます。全二重会話では使用されません。

次の値があります。

AP_YES パートナー TP は、会話を Receive 状態に変更するようローカル TP に要求する [MC_]REQUEST_TO_SEND verb を発行しました。

AP_NO パートナー TP は [MC_]REQUEST_TO_SEND verb を発行しませんでした。

このインディケーターを受信 verb で受信できる理由については、 222 ページの『MC_REQUEST_TO_SEND および REQUEST_TO_SEND』を参照してください。

expd_rcvd

優先データ・インディケーター。

次の値があります。

AP_YES パートナー TP が、ローカル TP がまだ受信していない優先データを送信しました。このデータを受信するには、ローカル TP は [MC_]RECEIVE_EXPEDITED_DATA verb を使用することができます。

このインディケーターを、APPC verb 数に対して設定できます。これは、ローカル TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行してデータを受信するまで、後続の verb に対しても設定できます。

AP_NO 受信を待機している優先データはありません。

dlen このパラメーターは、データを受信したことを *what_rcvd* パラメーターが示した場合にのみ使用されます。

受信したデータのバイト数 (データは、*dptr* パラメーターで指定されたバッファーに格納されます)。 長さ 0 (ゼロ) は、データを受信しなかったことを示します。

会話の割り振りが解除された場合: パートナー TP が確認を要求せずに会話の割り振りを解除した場合、 APPC は次のパラメーターを戻します。

primary_rc

AP_DEALLOC_NORMAL
パートナー TP は、*dealloc_type* を次のいずれかに設定して [MC_]DEALLOCATE verb を発行しました。

- AP_FLUSH
- 会話の同期レベルが AP_NONE として指定された AP_SYNC_LEVEL

dlen 受信したデータのバイト数 (データは、*dptr* パラメーターで指定されたバッファーに格納されます)。 長さ 0 (ゼロ) は、データを受信しなかったことを示します。 このパラメーターは、*rtn_status* を AP_YES に設定した場合にのみ使用されます。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_RETURN_STATUS_WITH_DATA
rtn_status パラメーターが無効な値に設定されていました。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT
予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

WINDOWS

AP_INVALID_DATA_SEGMENT
データが割り振りデータ・セグメントより長かったか、データ・バッファーのアドレスが間違っていました。

AP_RCV_AND_WAIT_BAD_FILL
この戻りコードは、基本会話の RECEIVE_AND_WAIT verb にのみ適用されます。 *fill* パラメーターが無効な値に設定されていました。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc
次の値があります。

AP_RCV_AND_WAIT_BAD_STATE
TP がこの verb を発行したとき、会話は Receive、Send、Send_Pending のいずれの状態でもありませんでした。

AP_RCV_AND_WAIT_NOT_LL_BDY
この戻りコードは、基本会話の RECEIVE_AND_WAIT verb にのみ適用されます。会話が Send 状態にありました。TP は開始されましたが、論理レコードの送信を終了しませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

AIX, LINUX

primary_rc
AP_BACKED_OUT

secondary_rc
AP_BO_NO_RESYNC

AP_BO_RESYNC

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_PROG_ERROR_NO_TRUNC
AP_PROG_ERROR_PURGING
AP_PROG_ERROR_TRUNC
AP_INVALID_VERB
AP_TP_BUSY

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、MC_RECEIVE_AND_WAIT verb が戻します。

primary_rc

AP_DEALLOC_ABEND

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、RECEIVE_AND_WAIT verb が戻します。

primary_rc

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_NO_TRUNC
AP_SVC_ERROR_PURGING
AP_SVC_ERROR_TRUNC

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP は、会話が Receive、Send、Send_Pending のいずれかの状態のときに [MC_]RECEIVE_AND_WAIT verb を発行できます。

Send 状態での verb の発行 (半二重会話専用)

会話が Send 状態のときに [MC_]RECEIVE_AND_WAIT verb を発行すると、次の効果があります。

- ローカル LU は、その送信バッファー内の情報と SEND インディケーターをパートナー TP へ送信します。
- 会話は Receive 状態に変更され、ローカル TP はパートナー TP からの情報を受信するために待機します。

状態の変更

WINDOWS

verb が非同期エントリー・ポイントに対して発行されると、会話状態は 2 回変わります。この verb の初期の戻りで、 *primary_rc* が AP_OK であれば、会話は Pending_Post 状態に変更されます。 APPC が verb の完了を通知すると、状態は以下のように変わります。アプリケーションが Pending_Post 状態で実行できるアクションについての詳細は、 175 ページの『MC_RECEIVE_AND_POST および RECEIVE_AND_POST』を参照してください。

[MC_]RECEIVE_AND_WAIT verb の後の状態変更は、次の値によって異なります。

- *primary_rc* パラメーター
- *what_rcvd* パラメーター

考えられる状態の変更を次の表に要約します。

what_rcvd パラメーター	新しい状態
AP_CONFIRM_WHAT_RECEIVED AP_DATA_CONFIRM AP_DATA_COMPLETE_CONFIRM	Confirm (確認)
AP_CONFIRM_DEALLOCATE AP_DATA_CONFIRM_DEALLOCATE AP_DATA_COMPLETE_CONFIRM_DEALL	Confirm_Deallocate (確認 _ 割り振り解除)
AP_CONFIRM_SEND AP_DATA_CONFIRM_SEND AP_DATA_COMPLETE_CONFIRM_SEND	Confirm_Send (確認 _ 送信)
AP_DATA AP_DATA_COMPLETE AP_DATA_INCOMPLETE	Receive (受信) (半二重会話) または変更なし (全二重会話)
AP_SEND	Send (送信)
AP_DATA_SEND AP_DATA_COMPLETE_SEND	Send_Pending (送信 _ 保留)

primary_rc	新しい状態
AP_PARAMETER_CHECK	変更なし
AP_STATE_CHECK	
AP_CONVERSATION_TYPE_MIXED	
AP_INVALID_VERB	
AP_INVALID_VERB_SEGMENT	
AP_STACK_TOO_SMALL	
AP_TP_BUSY	
AP_UNEXPECTED_DOS_ERROR	
AP_CONV_FAILURE_RETRY	Reset (リセット)
AP_CONV_FAILURE_NO_RETRY	
AP_DEALLOC_ABEND	Reset (リセット)
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	
AP_DEALLOC_NORMAL	Reset (リセット)
AP_PROG_ERROR_PURGING	Receive (受信) (半二重会話) または変更なし (全二重会話)
AP_PROG_ERROR_NO_TRUNC	
AP_SVC_ERROR_PURGING	
AP_SVC_ERROR_NO_TRUNC	
AP_PROG_ERROR_TRUNC	
AP_SVC_ERROR_TRUNC	

使用上の注意

この項では、次のトピックについて、使用上の追加情報を記載します。

- PS ヘッダー・データ
- 無限待機の回避

PS ヘッダー・データ

AIX, LINUX

同期レベルが AP_SYNCPT の会話では、受信データが PS ヘッダーの形式になっている場合があります。 マップ式会話では、これは *what_rcvd* パラメーターの値によって示されます。基本会話では、0x0001 という LL フィールドによって示されます (詳細については、 67 ページの『論理レコード』を参照)。同期点管理プログラムは、データを適切な同期点コマンドに変換します。

無限待機の回避

ローカル TP は、[MC_]RECEIVE_AND_WAIT verb を発行した場合、パートナー TP から情報を受信するまで中断状態になります。パートナー TP が、何の情報も送信しないか、パートナー LU に送信バッファーをフラッシュさせる verb を発行しなければ、ローカル TP は無限に待機する可能性があります。 TP を切れ目なく

動作させる必要がある場合は、コールバック・ルーチンを待機しないようにして [MC_]RECEIVE_AND_POST verb を使用するか、[MC_]RECEIVE_IMMEDIATE verb を使用します。

MC_RECEIVE_IMMEDIATE および RECEIVE_IMMEDIATE

MC_RECEIVE_IMMEDIATE verb または RECEIVE_IMMEDIATE verb は、パートナー TP から現在入手可能なデータか状況情報、またはその両方を受信します。現在入手可能なものがない場合、ローカル TP は即時に戻り、待機しません。

WINDOWS

この verb は情報の受信を待機しませんが、 Windows APPC ライブラリーが他の処理の継続を放棄することはまだ可能です。この verb が放棄せずに戻ることは想定しないでください。

VCB 構造体: MC_RECEIVE_IMMEDIATE

AIX, LINUX

MC_RECEIVE_IMMEDIATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_receive_immediate
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  AP_UINT16        what_rcvd;
  unsigned char    rtn_status;
  unsigned char    reserv4;
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
  AP_UINT16        max_len;
  AP_UINT16        dlen;
  unsigned char    *dptr;
  unsigned char    reserv6[5];
} MC_RECEIVE_IMMEDIATE;
```

VCB 構造体: RECEIVE_IMMEDIATE

RECEIVE_IMMEDIATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct receive_immediate
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
```

```
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  AP_UINT16        what_rcvd;
  unsigned char    rtn_status;
  unsigned char    fill;
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
  AP_UINT16        max_len;
  AP_UINT16        dlen;
  unsigned char    *dptr;
  unsigned char    reserv5[5];
} RECEIVE_IMMEDIATE;
```

VCB 構造体: MC_RECEIVE_IMMEDIATE (Windows)

WINDOWS

MC_RECEIVE_IMMEDIATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_receive_immediate
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned short    what_rcvd;
  unsigned char     rtn_status;
  unsigned char     reserv4;
  unsigned char     rts_rcvd;
  unsigned char     reserv5;
  unsigned short    max_len;
  unsigned short    dlen;
  unsigned char far *dptr;
  unsigned char     reserv6[5];
} MC_RECEIVE_IMMEDIATE;
```

VCB 構造体: RECEIVE_IMMEDIATE (Windows)

RECEIVE_IMMEDIATE verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct receive_immediate
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned short    what_rcvd;
  unsigned char     rtn_status;
  unsigned char     fill;
  unsigned char     rts_rcvd;
  unsigned char     reserv4;
  unsigned short    max_len;
  unsigned short    dlen;
  unsigned char far *dptr;
  unsigned char     reserv5[5];
} RECEIVE_IMMEDIATE;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_RECEIVE_IMMEDIATE
MC_RECEIVE_IMMEDIATE verb の場合

AP_B_RECEIVE_IMMEDIATE
RECEIVE_IMMEDIATE verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_RECEIVE_IMMEDIATE verb の場合

AP_BASIC_CONVERSATION
RECEIVE_IMMEDIATE verb の場合

verb が全二重会話で発行されている場合、または非ブロッキング verb として発行されている場合、上記の値を次の値のいずれかまたは両方と (論理 OR を使用して) 結合します。

AP_FULL_DUPLEX_CONVERSATION
この verb は全二重会話で発行されます。

AP_NON_BLOCKING
この verb は非ブロッキング verb として発行されます。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

rtn_status
状況情報とデータを同じ verb で戻せるかどうかを示します。 次の値があります。

AP_YES 状況情報は、入手できれば、データ・レコードの最後の部分と共に戻ります。

AP_NO 状況情報は、データと共には戻りません。ローカル TP は、データ・レコードの終わりを受信した後に、再度 [MC_]RECEIVE verb を発行して状況情報を取得する必要があります。

fill ローカル TP がデータを受信する方法を示します。

このパラメーターは、基本会話の RECEIVE_IMMEDIATE verb のみが使用します。 次の値があります。

AP_BUFFER
ローカル TP は、*max_len* パラメーターで指定したバイト数に到達するまで、またはデータの終わりに到達するまで、データを受信します。 データは、論理レコード形式に関係なく受信されます。

AP_LL データは、論理レコード形式で受信されます。受信されるデータは、次のいずれかです。

- 1 つの完全な論理レコード
- ある論理レコードの *max_len* バイトの部分
- 論理レコードの終わり

max_len
ローカル TP が受信できるデータの最大バイト数。

この値の範囲は、0 ～ 65,535 です。

この値は、受信したデータを入れるバッファーの長さを超えてはなりません。

dptr ローカル LU が受信したデータを入れるバッファーのアドレス。

WINDOWS

データ・バッファーは、静的データ域か、グローバル割り振り域にあります。データ・バッファーは、この領域と完全に一致していなければなりません。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

what_rcvd
着信データと共に受信した状況情報。

TP が次に実行するアクションは、通常このパラメーターの値によって決まります。 詳細については、 172 ページの『what_rcvd パラメーター』を参照してください。

次の値があります。

AP_CONFIRM_DEALLOCATE
パートナー TP は、*dealloc_type* を AP_SYNC_LEVEL に設定して [MC_]DEALLOCATE verb を発行し、 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_CONFIRM_SEND
パートナー TP は、*ptr_type* を AP_SYNC_LEVEL に設定して [MC_]PREPARE_TO_RECEIVE verb を発行し、 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_CONFIRM_WHAT_RECEIVED
パートナー TP は [MC_]CONFIRM verb を発行しました。

AP_DATA
この値は、基本会話の RECEIVE_IMMEDIATE verb で *fill* パラメーターを AP_BUFFER に設定した場合に、戻ることがあります。この値は、MC_RECEIVE_IMMEDIATE には適用されません。

ローカル TP は、*max_len* またはデータの終わりに到達するまでデータを受信しました。

AP_DATA_COMPLETE
マップ式会話の MC_RECEIVE_IMMEDIATE verb の場合、この値はローカル TP が完全なデータ・レコードを受信したか、データ・レコードの最後の部分を受信したことを示します。

fill パラメーターを AP_LL に設定した基本会話の RECEIVE_IMMEDIATE verb の場合、この値はローカル TP が完全な論理レコードを受信したか、論理レコードの終わりを受信したことを示します。

AP_DATA_INCOMPLETE
マップ式会話の MC_RECEIVE_IMMEDIATE verb の場合、この値はローカル TP が不完全なデータ・レコードを受信したことを示します。 *max_len* パラメーターで、データ・レコードの長さより小さい (または、これがレコードを読み取るための最初の受信 verb でない場合はデータ・レコードの残りの部分より小さい) 値を指定しました。

fill パラメーターを AP_LL に設定した基本会話の RECEIVE_IMMEDIATE verb の場合、この値はローカル TP が不完全な論理レコードを受信したことを示します。

AP_SEND
パートナー TP の場合、会話は Receive 状態に入りました。 ローカル TP の場合、会話は Send 状態に入りました。

次の値は、*rtn_status* が AP_YES に設定された場合にのみ戻ります。

AP_DATA_CONFIRM
これは、AP_DATA と AP_CONFIRM_WHAT_RECEIVED を組み合わせたものです。 パートナー TP は、データの送信後、[MC_]CONFIRM verb を発行したか、CONFIRM の送信タイプが指定された [MC_]SEND_DATA verb を発行しました。

AP_DATA_COMPLETE_CONFIRM
これは、AP_DATA_COMPLETE と AP_CONFIRM_WHAT_RECEIVED を組み合わせたものです。 パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、[MC_]CONFIRM

verb を発行したか、CONFIRM の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。

AP_DATA_CONFIRM_DEALLOCATE
これは、AP_DATA と AP_CONFIRM_DEALLOCATE を組み合わせたものです。 パートナー TP は、データの送信後、 *dealloc_type* を AP_SYNC_LEVEL に設定して [MC_]DEALLOCATE verb を発行したか、DEALLOC_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_COMPLETE_CONFIRM_DEALL
これは、AP_DATA_COMPLETE と AP_CONFIRM_DEALLOCATE を組み合わせたものです。 パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、*dealloc_type* を AP_SYNC_LEVEL に設定して [MC_]DEALLOCATE verb を発行したか、DEALLOC_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_CONFIRM_SEND
これは、AP_DATA と AP_CONFIRM_SEND を組み合わせたものです。 パートナー TP は、データの送信後、 *ptr_type* を AP_SYNC_LEVEL に設定して [MC_]PREPARE_TO_RECEIVE verb を発行したか、P_TO_R_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_COMPLETE_CONFIRM_SEND
これは、AP_DATA_COMPLETE と AP_CONFIRM_SEND を組み合わせたものです。 パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、*ptr_type* を AP_SYNC_LEVEL に設定して [MC_]PREPARE_TO_RECEIVE verb を発行したか、P_TO_R_SYNC_LEVEL の送信タイプを指定して [MC_]SEND_DATA verb を発行しました。 [MC_]ALLOCATE verb によって確立された会話の同期レベルは AP_CONFIRM_SYNC_LEVEL です。

AP_DATA_SEND
パートナー TP は、データの送信後、Receive 状態に入りました。 ローカル TP の場合、会話は Send_Pending 状態に入りました。

AP_DATA_COMPLETE_SEND
パートナー TP は、完全なデータ・レコード (またはデータ・レコードの終わり) を送信した後、Receive 状態に入りました。 ローカル TP の場合、会話は Send_Pending 状態に入りました。

AIX, LINUX

次の値は、MC_RECEIVE_IMMEDIATE verb で戻ります。

AP_USER_CONTROL_DATA_INCMP
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_INCOMPLETE と同じです。

AP_USER_CONTROL_DATA_COMPLETE
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE と同じです。

AP_UC_DATA_COMPLETE_SEND
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_SEND と同じです。

AP_UC_DATA_COMPLETE_CONFIRM
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM と同じです。

AP_UC_DATA_COMPLETE_CNFM_DEALL
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_DEALL と同じです。

AP_UC_DATA_COMPLETE_CNFM_SEND
受信したデータがユーザー制御データの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_SEND と同じです。

次の値は、*sync_level* を AP_SYNCPT に設定した MC_RECEIVE_IMMEDIATE verb で戻ります。

AP_PS_HEADER_INCOMPLETE
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_INCOMPLETE と同じです。

AP_PS_HEADER_COMPLETE
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE と同じです。

AP_PS_HDR_COMPLETE_SEND
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_SEND と同じです。

AP_PS_HDR_COMPLETE_CONFIRM
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM と同じです。

AP_PS_HDR_COMPLETE_CNFM_DEALL
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_DEALL と同じです。

AP_PS_HDR_COMPLETE_CNFM_SEND
受信したデータが PS ヘッダーの形式であることを除けば、AP_DATA_COMPLETE_CONFIRM_SEND と同じです。

rts_rcvd

送信要求受信インディケーター。このパラメーターは、半二重会話でのみ使用できます。全二重会話では使用されません。

次の値があります。

AP_YES パートナー TP は、会話を Receive 状態に変更するようローカル TP に要求する [MC_]REQUEST_TO_SEND verb を発行しました。

AP_NO パートナー TP は [MC_]REQUEST_TO_SEND verb を発行しませんでした。

このインディケーターを受信 verb で受信できる理由については、 222 ページの『MC_REQUEST_TO_SEND および REQUEST_TO_SEND』を参照してください。

expd_rcvd

優先データ・インディケーター。

次の値があります。

AP_YES パートナー TP が、ローカル TP がまだ受信していない優先データを送信しました。このデータを受信するには、ローカル TP は [MC_]RECEIVE_EXPEDITED_DATA verb を使用することができます。

このインディケーターを、APPC verb 数に対して設定できます。これは、ローカル TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行してデータを受信するまで、後続の verb に対しても設定できます。

AP_NO 受信を待機している優先データはありません。

dlen このパラメーターは、データを受信したことを *what_rcvd* パラメーターが示した場合にのみ使用されます。

受信したデータのバイト数 (データは、*dptr* パラメーターで指定されたバッファーに格納されます)。 長さ 0 (ゼロ) は、データを受信しなかったことを示します。

会話の割り振りが解除された場合: パートナー TP が確認を要求せずに会話の割り振りを解除した場合、 APPC は次のパラメーターを戻します。

primary_rc

AP_DEALLOC_NORMAL
パートナー TP は、*dealloc_type* を次のいずれかに設定して [MC_]DEALLOCATE verb を発行しました。

- AP_FLUSH
- 会話の同期レベルが AP_NONE として指定された AP_SYNC_LEVEL

dlen 受信したデータのバイト数 (データは、*dptr* パラメーターで指定されたバッファーに格納されます)。 長さ 0 (ゼロ) は、データを受信しなかったことを示します。 このパラメーターは、*rtn_status* を AP_YES に設定した場合にのみ使用されます。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc

AP_PARAMETER_CHECK

secondary_rc

次の値があります。

AP_BAD_CONV_ID

conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_RETURN_STATUS_WITH_DATA

rtn_status パラメーターが無効な値に設定されていました。

AP_BAD_TP_ID

tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT

予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED

アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

WINDOWS

AP_INVALID_DATA_SEGMENT

データが割り振りデータ・セグメントより長かったか、データ・バッファーのアドレスが間違っていました。

AP_RCV_IMMD_BAD_FILL

この戻りコードは、基本会話の RECEIVE_IMMEDIATE verb にのみ適用されます。 *fill* パラメーターが無効な値に設定されていました。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc
次の値があります。

AP_RCV_IMMD_BAD_STATE
TP がこの verb を発行したとき、会話は Receive 状態ではありませんでした。

データが入手可能でない場合: パートナー TP から即時に入手できるデータがない場合、APPC は次のパラメーターを戻します。

primary_rc
AP_UNSUCCESSFUL

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

AIX, LINUX

primary_rc
AP_BACKED_OUT

secondary_rc
AP_BO_NO_RESYNC
AP_BO_RESYNC

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_PROG_ERROR_NO_TRUNC
AP_PROG_ERROR_PURGING
AP_PROG_ERROR_TRUNC
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、MC_RECEIVE_IMMEDIATE verb が戻します。

primary_rc
AP_DEALLOC_ABEND

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、RECEIVE_IMMEDIATE verb が戻します。

primary_rc
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_NO_TRUNC
AP_SVC_ERROR_PURGING
AP_SVC_ERROR_TRUNC

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP は、会話が Send_Receive (全二重会話専用) または Receive 状態のときに [MC_]RECEIVE_IMMEDIATE verb のみを発行できます。

状態の変更

[MC_]RECEIVE_IMMEDIATE verb の後の状態変更は、次の値によって異なります。

- *primary_rc* パラメーター
- *primary_rc* が AP_OK の場合は、 *what_rcvd* パラメーター

考えられる状態の変更を次の表に要約します。

what_rcvd パラメーター	新しい状態
AP_CONFIRM_WHAT_RECEIVED	Confirm (確認)
AP_DATA_CONFIRM	
AP_DATA_COMPLETE_CONFIRM	
AP_CONFIRM_DEALLOCATE	Confirm_Deallocate (確認 _ 割り振り解除)
AP_DATA_CONFIRM_DEALLOCATE	
AP_DATA_COMPLETE_CONFIRM_DEALL	
AP_CONFIRM_SEND	Confirm_Send (確認 _ 送信)
AP_DATA_CONFIRM_SEND	
AP_DATA_COMPLETE_CONFIRM_SEND	
AP_DATA	Receive (受信) (半二重会話) または変更なし (全二重会話)
AP_DATA_COMPLETE	
AP_DATA_INCOMPLETE	
AP_SEND	Send (送信)
AP_DATA_SEND	Send_Pending (送信 _ 保留)
AP_DATA_COMPLETE_SEND	

primary_rc	新しい状態
AP_PARAMETER_CHECK	変更なし
AP_STATE_CHECK	
AP_CONVERSATION_TYPE_MIXED	
AP_INVALID_VERB	
AP_INVALID_VERB_SEGMENT	
AP_STACK_TOO_SMALL	
AP_TP_BUSY	
AP_UNEXPECTED_DOS_ERROR	
AP_CONV_FAILURE_RETRY	Reset (リセット)
AP_CONV_FAILURE_NO_RETRY	
AP_DEALLOC_ABEND	Reset (リセット) (半二重会話) または Send_Only (全二重会話)
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	
AP_DEALLOC_NORMAL	

primary_rc	新しい状態
AP_PROG_ERROR_PURGING AP_PROG_ERROR_NO_TRUNC AP_SVC_ERROR_PURGING AP_SVC_ERROR_NO_TRUNC AP_PROG_ERROR_TRUNC AP_SVC_ERROR_TRUNC	Receive (受信) (半二重会話) または変更なし (全二重会話)

PS ヘッダー・データ

AIX, LINUX

同期レベルが AP_SYNCPT の会話では、受信データが PS ヘッダーの形式になっている場合があります。 マップ式会話では、これは *what_rcvd* パラメーターの値によって示されます。基本会話では、0x0001 という LL フィールドによって示されます (詳細については、67 ページの『論理レコード』を参照)。 同期点管理プログラムは、データを適切な同期点コマンドに変換します。

MC_RECEIVE_EXPEDITED_DATA および RECEIVE_EXPEDITED_DATA

MC_RECEIVE_EXPEDITED_DATA または RECEIVE_EXPEDITED_DATA verb は、パートナー TP から現在入手可能な優先データを受信します。入手可能なデータがない場合、verb は即時に戻るか、またはデータの到着を待機することができます。

VCB 構造体: MC_RECEIVE_EXPEDITED_DATA

MC_RECEIVE_EXPEDITED_DATA verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_receive_expedited_data
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    rtn_ctl;
  unsigned char    reserv1[3];
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
  AP_UINT16        max_len;
  AP_UINT16        dlen;
  unsigned char    *dptr;
} MC_RECEIVE_EXPEDITED_DATA;
```

VCB 構造体: RECEIVE_EXPEDITED_DATA

RECEIVE_EXPEDITED_DATA verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct receive_expedited_data
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                  /* Reserved            */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    rtn_ctl;
  unsigned char    reserv1[3];
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
  AP_UINT16        max_len;
  AP_UINT16        dlen;
  unsigned char    *dptr;
} RECEIVE_EXPEDITED_DATA;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_RECEIVE_EXPEDITED_DATA
MC_RECEIVE_EXPEDITED_DATA verb の場合

AP_B_RECEIVE_EXPEDITED_DATA
RECEIVE_EXPEDITED_DATA verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_RECEIVE_EXPEDITED_DATA verb の場合

AP_BASIC_CONVERSATION
RECEIVE_EXPEDITED_DATA verb の場合

verb が全二重会話で発行されている場合、または非ブロッキング verb として発行されている場合、上記の値を次の値のいずれかまたは両方と (論理 OR を使用して) 結合します。

AP_FULL_DUPLEX_CONVERSATION
この verb は全二重会話で発行されます。

AP_NON_BLOCKING
この verb は非ブロッキング verb として発行されます。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

rtn_ctl verb が発行されたときに、入手可能な優先データがない場合、TP へ制御を戻す時期を示します。次の値があります。

AP_IMMEDIATE
入手可能な優先データがない場合、verb はこれを示す戻りコードで即時に戻ります。

AP_WHEN_EXPD_RCVD
入手可能な優先データがない場合、verb はデータが到着するまで待ちます。パートナー TP が優先データを発行しないと、 verb は無限に待機する可能性があります。

max_len
ローカル TP が受信できるデータの最大バイト数。

この値の範囲は、0 ～ 86 です。

この値は、受信したデータを入れるバッファーの長さを超えてはなりません。

dptr ローカル LU が受信したデータを入れるバッファーのアドレス。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

expd_rcvd
優先データ・インディケーター。

次の値があります。

AP_YES パートナー TP は、この verb に戻るデータの他に、ローカル TP がまだ受信していない優先データも送信しました。このデータを受信するには、ローカル TP は [MC_]RECEIVE_EXPEDITED_DATA verb を再発行することができます。

このインディケーターを、APPC verb 数に対して設定できます。これは、ローカル TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行してデータを受信するまで、後続の verb に対しても設定できます。

AP_NO 受信を待機している優先データはありません。

dlen 受信したデータのバイト数 (データは、*dptr* パラメーターで指定されたバッファーに格納されます)。 長さ 0 (ゼロ) は、データを受信しなかったことを示します。

受信したデータは不定形式で、2 バイト長のフィールド (LL) は入りません。

データが入手可能でない場合: *rtn_ctl* パラメーターが AP_IMMEDIATE に設定されていて、入手可能な優先データがない場合、APPC は次のパラメーターを戻します。

primary_rc

AP_UNSUCCESSFUL

会話の割り振りが解除された場合: パートナー TP が会話の割り振りを解除した場合、APPC は次の値のいずれかを戻します。

primary_rc

AP_DEALLOC_NORMAL
パートナー TP は、*dealloc_type* を次のいずれかに設定して [MC_]DEALLOCATE verb を発行しました。

- AP_FLUSH
- 会話の同期レベルが AP_NONE として指定された AP_SYNC_LEVEL

primary_rc

AP_CONVERSATION_ENDED
この verb は、非ブロッキング verb として発行され、先に発行された verb の後のキューに入ります。パートナー TP は、上記の AP_DEALLOC_NORMAL に [MC_]DEALLOCATE verb を発行し、 キューの最初の verb が *primary_rc* を AP_DEALLOC_NORMAL に設定して戻ります。これは、会話の終了を示します。次に、キューに入っている後続の verb が、*primary_rc* を AP_CONVERSATION_ENDED に設定して戻ります。これは、verb が処理される前に、会話がすでに終了していたことを示します。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

優先データがサポートされない: リモート LU が優先データをサポートしないために verb が実行されない場合には、 APPC は次のパラメーターを戻します。

primary_rc

AP_EXPD_NOT_SUPPORTED_BY_LU

データ・バッファーが小さすぎる: TP のデータ・バッファーが小さすぎて、入手可能な優先データのすべてが入らないために verb が実行されない場合には、 APPC は次のパラメーターを戻します。

primary_rc

AP_BUFFER_TOO_SMALL

dlen LU で入手可能な優先データのバイト数。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_INVALID_FORMAT
予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_EXPD_BAD_RETURN_CONTROL
rtn_ctl パラメーターが無効な値に設定されていました。

AP_RCV_EXPD_INVALID_LENGTH
max_len パラメーターが無効な値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc
次の値があります。

AP_EXPD_DATA_BAD_CONV_STATE
TP がこの verb を発行したとき、会話は Reset 状態でした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_CONVERSATION_TYPE_MISMATCH
AP_DUPLEX_TYPE_MIXED
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED

AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY

AIX, LINUX

primary_rc
AP_BACKED_OUT

secondary_rc
AP_BO_NO_RESYNC
AP_BO_RESYNC

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_UNEXPECTED_SYSTEM_ERROR
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_PROG_ERROR_NO_TRUNC
AP_PROG_ERROR_PURGING
AP_PROG_ERROR_TRUNC
AP_INVALID_VERB
AP_TP_BUSY

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、MC_RECEIVE_EXPEDITED_DATA verb が戻します。

primary_rc
AP_DEALLOC_ABEND

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、RECEIVE_EXPEDITED_DATA verb が戻します。

primary_rc
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_NO_TRUNC
AP_SVC_ERROR_PURGING
AP_SVC_ERROR_TRUNC

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP は、会話が Reset 以外の状態のときに [MC_]RECEIVE_EXPEDITED_DATA verb を発行できます。

状態の変更

[MC_]RECEIVE_EXPEDITED_DATA verb の後の状態変更は、 *primary_rc* パラメーターによって異なります。考えられる状態の変更を次の表に要約します。

primary_rc	**新しい状態**
AP_OK	変更なし
AP_PARAMETER_CHECK	変更なし
AP_STATE_CHECK	
AP_CONVERSATION_TYPE_MIXED	
AP_INVALID_VERB	
AP_INVALID_VERB_SEGMENT	
AP_STACK_TOO_SMALL	
AP_TP_BUSY	
AP_UNEXPECTED_DOS_ERROR	
AP_CONV_FAILURE_RETRY	Reset (リセット)
AP_CONV_FAILURE_NO_RETRY	
AP_CONVERSATION_ENDED	
AP_DEALLOC_ABEND	Reset (リセット)
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	
AP_DEALLOC_NORMAL	Reset (リセット)

MC_REQUEST_TO_SEND および REQUEST_TO_SEND

MC_REQUEST_TO_SEND verb または REQUEST_TO_SEND verb は、ローカル TP がデータの送信を必要としていることをパートナー TP に通知します。

注: この verb は、半二重会話でのみ使用できます。全二重会話では無効です。

パートナー TP のアクション

この要求に応答して、パートナー TP は会話を次のいずれかの状態に変更できます。

- [MC_]PREPARE_TO_RECEIVE verb または [MC_]RECEIVE_AND_WAIT verb を発行することにより、Receive 状態に変更する
- [MC_]RECEIVE_AND_POST verb を発行することにより、 Pending_Post 状態に変更する

パートナー TP は、送信要求を無視することもできます。

ローカル TP がデータを送信できる時点

ローカル TP の会話状態は、ローカル TP が後続の受信 verb の *what_rcvd* パラメーターを通じて次のいずれかの値を受信した時点で、Send に変更されます。

- AP_CONFIRM_SEND、AP_DATA_CONFIRM_SEND、または AP_DATA_COMPLETE_CONFIRM_SEND (および [MC_]CONFIRMED を伴った応答)
- AP_SEND

ローカル TP の会話状態は、ローカル TP が後続の受信 verb の *what_rcvd* パラメーターを通じて次のいずれかの値を受信した時点で、Send_Pending に変更されます。

- AP_DATA_SEND
- AP_DATA_COMPLETE_SEND

受信 verb は、[MC_]RECEIVE_AND_WAIT、[MC_]RECEIVE_IMMEDIATE、および [MC_]RECEIVE_AND_POST です。

VCB 構造体: MC_REQUEST_TO_SEND

AIX, LINUX

MC_REQUEST_TO_SEND verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_request_to_send
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved            */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
} MC_REQUEST_TO_SEND;
```

VCB 構造体: REQUEST_TO_SEND

REQUEST_TO_SEND verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct request_to_send
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved            */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
} REQUEST_TO_SEND;
```

VCB 構造体: MC_REQUEST_TO_SEND (Windows)

WINDOWS

MC_REQUEST_TO_SEND verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_request_to_send
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
```

```
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
} MC_REQUEST_TO_SEND;
```

VCB 構造体: REQUEST_TO_SEND (Windows)

REQUEST_TO_SEND verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct request_to_send
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
} REQUEST_TO_SEND;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_REQUEST_TO_SEND
MC_REQUEST_TO_SEND verb の場合

AP_B_REQUEST_TO_SEND
REQUEST_TO_SEND verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_REQUEST_TO_SEND verb の場合

AP_BASIC_CONVERSATION
REQUEST_TO_SEND verb の場合

verb が非ブロッキング verb として発行されている場合、上記の値を値 AP_NON_BLOCKING と (論理 OR を使用して) 結合します。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

会話の割り振りが解除された場合: パートナー TP が会話の割り振りを解除した場合、APPC は次の値を戻します。

primary_rc

AP_CONVERSATION_ENDED
この verb は、非ブロッキング verb として発行され、先に発行された verb の後のキューに入ります。パートナー TP は、上記の AP_DEALLOC_NORMAL に [MC_]DEALLOCATE verb を発行し、 キューの最初の verb が *primary_rc* を AP_DEALLOC_NORMAL に設定して戻ります。これは、会話の終了を示します。次に、キューに入っている後続の verb が、*primary_rc* を AP_CONVERSATION_ENDED に設定して戻ります。これは、verb が処理される前に、会話がすでに終了していたことを示します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT
予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

AP_R_TO_S_INVALID_FOR_FDX
ローカル TP は、全二重会話の中で [MC_]REQUEST_TO_SEND verb を使用しようとしました。この verb は、半二重会話でのみ使用できます。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc

AP_R_T_S_BAD_STATE
TP がこの verb を発行したとき、会話は許容される状態ではありませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_CONVERSATION_TYPE_MIXED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は次のいずれの状態にあっても構いません。

- Receive (受信)
- Confirm (確認)
- Pending_Post (保留 _ 送信)

状態の変更

この verb では、会話状態は変更されません。

送信要求通知の受信

送信要求通知は、次の verb の *rts_rcvd* パラメーターを通じて、パートナー・プログラムによって受信されます。

- [MC]CONFIRM
- [MC_]RECEIVE_AND_POST
- [MC_]RECEIVE_AND_WAIT
- [MC_]RECEIVE_IMMEDIATE
- [MC_]SEND_DATA
- [MC_]SEND_ERROR

これは、*primary_rc* が、[MC_]TEST_RTS verb で AP_OK であるか、[MC_]TEST_RTS_AND_POST verb のコールバックでも示されます。

送信要求通知は、即時にパートナー TP へ送信されます。APPC は、送信バッファーがいっぱいになるかフラッシュされるまで待機しません。 したがって、送信要求通知は、順序どおりに到着しない場合があります。たとえば、ローカル TP が Send 状態にあり、 [MC_]PREPARE_TO_RECEIVE verb に続いて [MC_]REQUEST_TO_SEND verb を発行した場合、Receive 状態にあるパートナー TP は送信通知を受信する前に送信要求通知を受信することもあります。このため、受信 verb で送信要求通知を TP へ報告することができます。

MC_SEND_CONVERSATION および SEND_CONVERSATION

MC_SEND_CONVERSATION verb または SEND_CONVERSATION verb はパートナー TP との会話を確立し、その会話でデータ・レコードを 1 つだけ送信し、会話の割り振りを解除します。これは、[MC_]ALLOCATE、 [MC_]SEND_DATA、[MC_]DEALLOCATE (FLUSH) という 3 つの verb を発行するのと同じことです。

VCB 構造体: MC_SEND_CONVERSATION

AIX, LINUX

MC_SEND_CONVERSATION verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_send_conversation
{
  AP_UINT16       opcode;
```

```
  unsigned char   opext;
  unsigned char   format;                  /* Reserved             */
  AP_UINT16       primary_rc;
  AP_UINT32       secondary_rc;
  unsigned char   tp_id[8];
  unsigned char   reserv3[8];
  unsigned char   rtn_ctl;
  unsigned char   reserv4;
  AP_UINT32       conv_group_id;
  AP_UINT32       sense_data;
  unsigned char   plu_alias[8];
  unsigned char   mode_name[8];
  unsigned char   tp_name[64];
  unsigned char   security;
  unsigned char   reserv6[11];
  unsigned char   pwd[10];
  unsigned char   user_id[10];
  AP_UINT16       pip_dlen;
  unsigned char   *pip_dptr;
  unsigned char   reserv6a;
  unsigned char   fqplu_name[17];
  unsigned char   reserv7[8];
  AP_UINT16       dlen;
  unsigned char   *dptr;
} MC_SEND_CONVERSATION;
```

VCB 構造体: SEND_CONVERSATION

SEND_CONVERSATION verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct send_conversation
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                  /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  unsigned char    reserv3[8];
  unsigned char    rtn_ctl;
  unsigned char    reserv4;
  AP_UINT32        conv_group_id;
  AP_UINT32        sense_data;
  unsigned char    plu_alias[8];
  unsigned char    mode_name[8];
  unsigned char    tp_name[64];
  unsigned char    security;
  unsigned char    reserv5[11];
  unsigned char    pwd[10];
  unsigned char    user_id[10];
  AP_UINT16        pip_dlen;
  unsigned char    *pip_dptr;
  unsigned char    reserv5a;
  unsigned char    fqplu_name[17];
  unsigned char    reserv6[8];
  AP_UINT16        dlen;
  unsigned char    *dptr;
} SEND_CONVERSATION;
```

VCB 構造体: MC_SEND_CONVERSATION (Windows)

WINDOWS

MC_SEND_CONVERSATION verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_send_conversation
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned char     reserv3[8];
  unsigned char     rtn_ctl;
  unsigned char     reserv4;
  unsigned long     conv_group_id;
  unsigned long     sense_data;
  unsigned char     plu_alias[8];
  unsigned char     mode_name[8];
  unsigned char     tp_name[64];
  unsigned char     security;
  unsigned char     reserv5[11];
  unsigned char     pwd[10];
  unsigned char     user_id[10];
  unsigned short    pip_dlen;
  unsigned char far *pip_dptr;
  unsigned char     reserv6;
  unsigned char     fqplu_name[17];
  unsigned char     reserv7[8];
  unsigned short    dlen;
  unsigned char far *dptr;
} MC_SEND_CONVERSATION;
```

VCB 構造体: SEND_CONVERSATION (Windows)

SEND_CONVERSATION verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct send_conversation
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned char     reserv3[8];
  unsigned char     rtn_ctl;
  unsigned char     reserv4;
  unsigned long     conv_group_id;
  unsigned long     sense_data;
  unsigned char     plu_alias[8];
  unsigned char     mode_name[8];
  unsigned char     tp_name[64];
  unsigned char     security;
  unsigned char     reserv5[11];
  unsigned char     pwd[10];
  unsigned char     user_id[10];
  unsigned short    pip_dlen;
  unsigned char far *pip_dptr;
  unsigned char     reserv6;
  unsigned char     fqplu_name[17];
  unsigned char     reserv7[8];
  unsigned short    dlen;
  unsigned char far *dptr;
} SEND_CONVERSATION;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_SEND_CONVERSATION
MC_SEND_CONVERSATION verb の場合

AP_B_SEND_CONVERSATION
SEND_CONVERSATION verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_SEND_CONVERSATION verb の場合

AP_BASIC_CONVERSATION
SEND_CONVERSATION verb の場合

verb が非ブロッキング verb として発行されている場合、上記の値を値 AP_NON_BLOCKING と (論理 OR を使用して) 結合します。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

rtn_ctl ローカル TP からのセッション要求で動作するローカル LU がローカル TP への制御を戻すことを指定します。セッションについての詳細は、66 ページの『LU 間セッション』を参照してください。このパラメーターの値が何であっても、LU はゼロ・セッション限度 (これは、セッションが決して割り振られないことを意味します) など、特定のエラーを検出した場合は、即時に TP に制御を戻します。

次の値があります。

AP_IMMEDIATE
コンテンション勝者セッションが即時に使用可能である (活動状態であり、別の会話に使用されていない) 場合、LU はこの会話をそのセッションに割り振り、即時に TP へ制御を戻します。 コンテンション勝者セッションが即時に使用可能でない場合は、即時に TP へ制御が戻され、*primary_rc* が AP_UNSUCCESSFUL になります。

AP_WHEN_SESSION_ALLOCATED
セッションが即時に使用可能である (活動状態であり、別の会話に使用されていない) 場合、LU はこの会話をそのセッションに割り振ります。即時に使用可能でなくても、活動状態にできるセッションがある場合、LU はそのセッションを活動状態にし、この会話をそのセッションに割り振ります。セッションを活動状態にできない場合、LU はセッションの 1 つが解放されるまで待機します。

AP_WHEN_SESSION_FREE
セッションが即時に使用可能である (活動状態であり、別の会話に使用されていない) 場合、LU はこの会話をそのセッションに割り振ります。即時に使用可能でなくても、活動状態にできるセッションがある場合、LU はそのセッションを活動状態にし、この会話をそのセッションに割り振ります。活動状態で解放されたセッション

がなく、別のセッションを活動状態にできない場合は、1 次戻りコード AP_ALLOCATION_ERROR および 2 次戻りコード AP_ALLOCATION_FAILURE_RETRY と共に制御が TP へ戻されます。これは、セッションが解放されるのを LU が待機しないことを除けば、 AP_WHEN_SESSION_ALLOCATED によく似ています。

AP_WHEN_CONWINNER_ALLOC
AP_WHEN_SESSION_ALLOCATED に関しては、LU が常に会話をコンテンション勝者セッションに割り振ることを除き、コンテンション敗者セッションを使用しません。

AP_WHEN_CONLOSER_ALLOC
AP_WHEN_SESSION_ALLOCATED に関しては、LU が常に会話をコンテンション敗者セッションに割り振ることを除き、コンテンション勝者セッションを使用しません。

AP_WHEN_CONV_GROUP_ALLOC
この値は、新規の会話に前の会話と同じセッションを使用させる場合に使用します。*conv_group_id* パラメーターに、[MC_]ALLOCATE verb または RECEIVE_ALLOCATE verb から戻された前の会話の会話グループ ID を設定してください。

conv_group_id パラメーターで識別されるセッションが即時に使用可能である (活動状態であり、別の会話に使用されていない) 場合、LU はこの会話をそのセッションに割り振り、即時に TP へ制御を戻します。そのセッションが別の会話に使用されている場合、LU はそのセッションが解放されるまで待機します。 そのセッションが活動状態でなくなっている場合は、1 次戻りコード AP_ALLOCATION_ERROR および 2 次戻りコード AP_ALLOCATION_FAILURE_NO_RETRY と共に制御が TP へ戻されます。

conv_group_id
会話用に要求したセッションの会話グループ ID。このパラメーターは、*rtn_ctl* を AP_WHEN_CONV_GROUP_ALLOC に設定した場合にのみ使用します。*rtn_ctl* をそれ以外の値に設定した場合は、このパラメーターを 2 進ゼロに設定してください。

plu_alias
パートナー LU をローカル TP に認識させるための別名。この名前は、構成のときに確立されたパートナー LU の名前に一致する必要があります。

このパラメーターは 8 バイトの ASCII 文字ストリングで、別名が 8 文字より短い場合は、右側に ASCII のブランク (0x20) が埋め込まれます。 これは、次の文字から構成できます。

- 英大文字
- 数字の 0 ～ 9
- ブランク
- 特殊文字の $、#、%、および @

このストリングの 1 文字目をブランクにすることはできません。

LU を LU 別名ではなく LU 名で識別するには、このパラメーターを 8 つの 2 進ゼロに設定し、*fqplu_name* パラメーターに LU 名を指定します。

mode_name
構成のときに定義されたネットワーク機能の特性セットの名前。

mode_name の値は、構成のときにパートナー LU へ関連付けられたモードの名前に一致する必要があります。

このパラメーターは、8 バイトの EBCDIC 文字ストリングです。 このパラメーターは、タイプ A の EBCDIC 文字セットの文字から構成できます。それらの文字は、次のとおりです。

- 英大文字
- 数字の 0 ～ 9
- 特殊文字の $、#、および ?

ストリングの 1 文字目は、英大文字か特殊文字でなければなりません。 モード名の長さが 8 文字未満の場合は、EBCDIC のブランク (0x40) を右側に埋め込んでください。

モード名は、すべて EBCDIC のブランク (0x40) にすることもできます。

マップ式会話では、SNASVCMG (APPC の内部で使用される予約済みのモード名) という名前にすることはできません。 この名前を基本会話で使用することも、推奨できません。

tp_name
呼び出し先 TP の名前。

呼び出し元 TP 内で [MC_]ALLOCATE verb が指定する *tp_name* の値は、呼び出し先 TP 内で RECEIVE_ALLOCATE verb によって指定される *tp_name* の値に一致する必要があります。

このパラメーターは 64 バイトの EBCDIC 文字ストリングで、大文字小文字の区別があります。通常、*tp_name* パラメーターはタイプ AE の EBCDIC 文字セットの文字からなっています (サービス TP に命名する場合は除きます)。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 ～ 9
- 特殊文字の $、#、?、およびピリオド (.)

TP 名が 64 バイト未満の場合は、EBCDIC のブランク (0x40) を右側に埋め込みます。

サービス TP の命名に関する SNA 規則は、上記の例外です。この名前は最大 4 文字からなり、1 文字目は 0x00 ～ 0x3F の 16 進バイトです。その他の文字は EBCDIC AE 文字セットからのものです。

セキュリティー
パートナー LU が呼び出し先 TP へのアクセスの妥当性を検査するために必要な情報を指定します。

構成のときに呼び出し先 TP 用に確立された会話セキュリティーに基づいて、次のいずれかの値を使用してください。

AP_NONE
呼び出し先 TP は、会話セキュリティーを使用しません (この値を使用する場合、呼び出し先 TP は、会話セキュリティーを使用しないように構成されている必要があります)。

AP_PGM 呼び出し先 TP は会話セキュリティーを使用します。したがってユーザー ID とパスワードを必要とします。 この情報を *user_id* パラメーターと *pwd* パラメーターによって提供してください。

AP_PGM_STRONG
呼び出し先 TP は会話セキュリティーを使用します。したがってユーザー ID とパスワードを必要とします。 さらに、AP_PGM_STRONG を設定すると、 Communications Server はネットワークを通じてパスワードを送信するときに、パスワードを暗号化します。 ユーザー ID とパスワードを *user_id* パラメーターと *pwd* パラメーターによって提供してください。

AP_SAME
この値は、TP が別の TP から有効なユーザー ID とパスワードを使用して呼び出され、次に会話セキュリティーを必要とする第 3 の TP を呼び出そうとするときに使用します。(ある TP が 2 番目の TP を呼び出し、2 番目の TP が第 3 の TP を呼び出す状況については、 4 ページの『複数の会話』に例を示しています。) この値は、第 3 の TP (呼び出し元 TP) に最初の呼び出し元 TP の会話セキュリティーがすでに検証済みであることを通知します。

この値を使用する場合、この [MC_]SEND_CONVERSATION verb で提供する *tp_id* は、この TP が呼び出されたときに RECEIVE_ALLOCATE verb で戻された tp_id と同じものである必要があります。

AIX, LINUX

この値は、TP が別の TP から呼び出されたわけではなく、適切なセキュリティー情報を別の手段で (たとえば、ログオン時に提供された AIX / Linux ユーザー名とパスワードから) 取得し検証した場合にも使用できます。 その場合、APPC はアプリケーションの実行に使用されている AIX / Linux ユーザー名を、必要であれば 10 文字に切り捨てて会話セキュリティー用のユーザー ID として使用します。この名前は、有効な AE ストリング文字 (*user_id* パラメーターの説明を参照) で構成し、呼び出し先の TP に有効なユーザー名になるようにしてください。

TP は、別の方法 (たとえば、会話を割り振る前にユーザーに有効なユーザー ID とパスワードの入力を要求するなど) でセキュリティー情報を取得した場合、[MC_]SEND_CONVERSATION を発行する前に、SET_TP_PROPERTIES を使用してそのユーザー ID を APPC に対して指定しなければなりません。

pwd *user_id* へ関連付けられているパスワード。

このパラメーターは、*security* パラメーターを AP_PGM または AP_PGM_STRONG に設定した場合にのみ必要です。それ以外の場合は予約済みです。

pwd パラメーターと *user_id* パラメーターは、呼び出し先 TP が置かれているコンピューター上に構成されたユーザー ID とパスワードの対に一致する必要があります。

このパラメーターは 10 バイトの EBCDIC 文字ストリングで、大文字小文字の区別があります。 *pwd* パラメーターは、タイプ AE の EBCDIC 文字セットの文字から構成できます。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 ～ 9
- 特殊文字の $、#、?、およびピリオド (.)

パスワードが 10 バイト未満の場合は、EBCDIC のブランク (0x40) を右側に埋め込みます。

user_id パートナー TP にアクセスするために必要なユーザー ID。

このパラメーターは、*security* パラメーターを AP_PGM または AP_PGM_STRONG に設定した場合にのみ必要です。それ以外の場合は予約済みです。

pwd パラメーターと *user_id* パラメーターは、呼び出し先 TP が置かれているコンピューター上に構成されたユーザー ID とパスワードの対に一致する必要があります。

このパラメーターは 10 バイトの EBCDIC 文字ストリングで、大文字小文字の区別があります。 *user_id* パラメーターは、タイプ AE の EBCDIC 文字セットの文字から構成できます。 それらの文字は、次のとおりです。

- 英大文字と英小文字
- 数字の 0 ～ 9
- 特殊文字の $、#、?、およびピリオド (.)

ユーザー ID が 10 バイト未満の場合は、EBCDIC のブランク (0x40) を右側に埋め込みます。

pip_dlen

パートナー TP へ渡すプログラム初期設定パラメーター (PIP) の長さ。

この値の範囲は、0 ～ 32,767 です。

すべてのインプリメンテーション形態の APPC が PIP データを受信できるわけではありません (ただし、PIP データを送信できる場合があります)。また、CPI-C は PIP データをサポートしていません。 PIP データをサポートしないインプリメンテーション形態の APPC をパートナー TP が使用している場合、またはパートナーが CPI-C アプリケーションである場合は、*pip_dlen* を 0 (ゼロ) に設定してください。

pip_dptr

PIP データが入っているバッファーのアドレス。

このパラメーターは、*pip_dlen* が 0 (ゼロ) より大きい場合にのみ使用してください。

PIP データは、パートナー TP またはリモート・オペレーティング・システムが必要とする初期設定パラメーターまたは環境セットアップ情報から構成できます。 PIP データは、汎用データ・ストリームの形式に従っている必要があります。 詳細については、IBM 資料の「*Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*」を参照してください。

fqplu_name

パートナー LU の完全修飾 LU 名。このパラメーターは、*plu_alias* をゼロに設定した場合にのみ使用します。 この名前は、構成のときに確立されたパートナー LU の名前に一致する必要があります。

この名前は 17 バイトの EBCDIC ストリングで、右側に EBCDIC のスペースが埋め込まれ、次のいずれかが含まれます。

- 1 ～ 8 文字の A ストリング文字からなるネットワーク ID、 EBCDIC のドット (ピリオド) 文字、および 1 ～ 8 文字の A ストリング文字からなる LU 名
- 1 ～ 8 文字の A ストリング文字からなる LU 名 (ネットワーク ID や EBCDIC のドットを伴わない)

dlen 送信するデータのバイト数。この値の範囲は、0 ～ 65,535 です。

dptr 送信するデータが入っているバッファーのアドレス。

WINDOWS

データ・バッファーは、静的データ域か、グローバル割り振り域にあります。データ・バッファーは、この領域と完全に一致していなければなりません。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

conv_group_id

この会話が使用するセッションの会話グループ ID です。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc

AP_PARAMETER_CHECK

secondary_rc

次の値があります。

AP_BAD_LL

この戻りコードは、SEND_CONVERSATION verb にのみ適用されます。無効な値が含まれる論理レコードの論理レコード長フィールドは、 0x0000、0x0001、0x8000、または 0x8001 です。詳細については、 67 ページの『論理レコード』を参照してください。

AP_BAD_RETURN_CONTROL

rtn_ctl に指定した値が有効ではありませんでした。

AP_BAD_SECURITY

セキュリティーに指定した値が有効ではありませんでした。

AP_BAD_TP_ID

tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_PIP_LEN_INCORRECT

pip_dlen の値が 32,767 を超えていました。

AP_UNKNOWN_PARTNER_MODE

plu_alias または *mode_name* に指定した値が有効ではありませんでした。

AIX, LINUX

AP_INVALID_FORMAT

予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED

アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: この verb で状態検査エラーは発生しません。

セッションが使用不可の場合: *rtn_ctl* に指定した値によっては、APPC が次のパラメーターを戻す場合があります。

primary_rc

AP_UNSUCCESSFUL
指定パラメーター *rtn_ctl* で TP に即時に制御を戻すこと (AP_IMMEDIATE) を指定しましたが、使用可能なコンテンション勝者セッションがローカル LU にありませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_REQUESTED_NOT_SUPPORTED
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_INVALID_VERB
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP がこの verb を発行するとき、会話状態は Reset です。この verb は、既存の会話がどのような状態にあるときでも発行できます。これは、この verb は常に新規の会話 (これは Reset 状態にあります) の開始を意味しているからです。

状態の変更

この verb では、会話状態は変更されません。

MC_SEND_DATA および SEND_DATA

MC_SEND_DATA verb または SEND_DATA verb は、パートナー TP へ伝送するデータをローカル LU の送信バッファーに入れます。

ローカル LU の送信バッファー内に収集されたデータは、次のいずれかが発生した時点でパートナー LU (およびパートナー TP) へ伝送されます。

- 送信バッファーがいっぱいになった。
- ローカル TP が、LU の送信バッファーをフラッシュする verb を発行する。これを行う verb は、[MC_]CONFIRM、 [MC_]DEALLOCATE、 [MC_]FLUSH、[MC_]PREPARE_TO_RECEIVE、 [MC_]RECEIVE_AND_WAIT、[MC_]RECEIVE_AND_POST、および [MC_]SEND_ERROR です。([MC_]CONFIRM、 [MC_]PREPARE_TO_RECEIVE、および [MC_]RECEIVE_AND_POST は、半二重会話にのみ適用されます。)

MC_SEND_DATA verb または SEND_DATA verb は、データを送信するほかに、この verb で [MC_]CONFIRM、[MC_]DEALLOCATE、[MC_]FLUSH、[MC_]PREPARE_TO_RECEIVE のいずれかの verb の機能を実行できるようにするオプションも含んでいます。 これは、[MC_]SEND_DATA に別の verb を続けて発行することに相当します。 ([MC_]CONFIRM と [MC_]PREPARE_TO_RECEIVE は、半二重会話にのみ適用されます。)

VCB 構造体: MC_SEND_DATA

AIX, LINUX

MC_SEND_DATA verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_send_data
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
  AP_UINT16        dlen;
  unsigned char    *dptr;
  unsigned char    type;
  unsigned char    data_type;
} MC_SEND_DATA;
```

VCB 構造体: SEND_DATA

SEND_DATA verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct send_data
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                  /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
  AP_UINT16        dlen;
  unsigned char    *dptr;
  unsigned char    type;
  unsigned char    reserv4;
} SEND_DATA;
```

VCB 構造体: MC_SEND_DATA (Windows)

WINDOWS

MC_SEND_DATA verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_send_data
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     rts_rcvd;
  unsigned char     reserv3;
  unsigned short    dlen;
  unsigned char far *dptr;
  unsigned char     type;
  unsigned char     reserv4;
} MC_SEND_DATA;
```

VCB 構造体: SEND_DATA (Windows)

SEND_DATA verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct send_data
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     rts_rcvd;
  unsigned char     reserv3;
  unsigned short    dlen;
  unsigned char far *dptr;
  unsigned char     type;
  unsigned char     reserv4;
} SEND_DATA;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_SEND_DATA
MC_SEND_DATA verb の場合

AP_B_SEND_DATA
SEND_DATA verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_SEND_DATA verb の場合

AP_BASIC_CONVERSATION
SEND_DATA verb の場合

verb が全二重会話で発行されている場合、または非ブロッキング verb として発行されている場合、上記の値を次の値のいずれかまたは両方と (論理 OR を使用して) 結合します。

AP_FULL_DUPLEX_CONVERSATION
この verb は全二重会話で発行されます。

AP_NON_BLOCKING
この verb は非ブロッキング verb として発行されます。

format このパラメーターは、マップ式会話の MC_SEND_DATA verb のみに適用されます。

新規 APPC アプリケーションを作成しているか、または既存の APPC アプリケーションを現在の Communications Server APPC ヘッダー・ファイルで再コンパイルしている場合、このパラメーターを 1 に設定する必要があります。(ヘッダー・ファイルの前のバージョンで作成された既存のアプリケーションには、このパラメーターが予約されているため、アプリケーションは Communications Server で変更されないまま作動するので、再作成する必要はありません。)

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id

会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

dlen ローカル LU の送信バッファーに入れるデータのバイト数。この値の範囲は、0 ～ 65,535 です。

dptr ローカル LU の送信バッファーに入れるデータが入っているバッファーのアドレス。

WINDOWS

データ・バッファーは、静的データ域か、グローバル割り振り域にあります。データ・バッファーは、この領域と完全に一致していなければなりません。

type [MC_]SEND_DATA に加えて、別の APPC verb の機能を実行するかどうかを指定します。 次の値があります。

AP_NONE
データの送信のみです。追加機能は実行しません。

AP_SEND_DATA_CONFIRM
このオプションは、半二重会話でのみ有効です。 *opext* パラメーターに AP_FULL_DUPLEX_CONVERSATION オプションが含まれている場合には、このオプションを使用しないでください。

[MC_]CONFIRM verb の機能を実行します。これは、[MC_]SEND_DATA に続けて [MC_]CONFIRM を発行することに相当します。 基本会話の SEND_DATA verb の場合、この verb で送信するデータは 1 つの完全な論理レコードであるか、論理レコードの終わりである必要があります。この値は、不完全な論理レコードを送信する場合には使用できません。

AP_SEND_DATA_FLUSH
[MC_]FLUSH verb の機能を実行します。これは、[MC_]SEND_DATA に続けて [MC_]FLUSH を発行することに相当します。 基本会話の SEND_DATA verb の場合、この verb で送信するデータは 1 つの完全な論理レコードであるか、論理レコードの終わりである必要があります。この値は、不完全な論理レコードを送信する場合には使用できません。

AP_SEND_DATA_P_TO_R_FLUSH
このオプションは、半二重会話でのみ有効です。 *opext* パラメータ

ーに AP_FULL_DUPLEX_CONVERSATION オプションが含まれている場合には、このオプションを使用しないでください。

ptr_type を AP_FLUSH に設定した [MC_]PREPARE_TO_RECEIVE verb の機能を実行します。これは、[MC_]SEND_DATA に続けて [MC_]PREPARE_TO_RECEIVE を発行することに相当します。基本会話の SEND_DATA verb の場合、この verb で送信するデータは 1 つの完全な論理レコードであるか、論理レコードの終わりである必要があります。この値は、不完全な論理レコードを送信する場合には使用できません。

AP_SEND_DATA_P_TO_R_CONFIRM

このオプションは、半二重会話でのみ有効です。 *opext* パラメーターに AP_FULL_DUPLEX_CONVERSATION オプションが含まれている場合には、このオプションを使用しないでください。

ptr_type を AP_CONFIRM_TYPE に設定した [MC_]PREPARE_TO_RECEIVE verb の機能を実行します。これは、[MC_]SEND_DATA に続けて [MC_]PREPARE_TO_RECEIVE を発行することに相当します。基本会話の SEND_DATA verb の場合、この verb で送信するデータは 1 つの完全な論理レコードであるか、論理レコードの終わりである必要があります。この値は、不完全な論理レコードを送信する場合には使用できません。

AP_SEND_DATA_P_TO_R_SYNC_LEVEL

このオプションは、半二重会話でのみ有効です。 *opext* パラメーターに AP_FULL_DUPLEX_CONVERSATION オプションが含まれている場合には、このオプションを使用しないでください。

ptr_type を AP_SYNC_LEVEL に設定した [MC_]PREPARE_TO_RECEIVE verb の機能を実行します。これは、[MC_]SEND_DATA に続けて [MC_]PREPARE_TO_RECEIVE を発行することに相当します。基本会話の SEND_DATA verb の場合、この verb で送信するデータは 1 つの完全な論理レコードであるか、論理レコードの終わりである必要があります。この値は、不完全な論理レコードを送信する場合には使用できません。

AP_SEND_DATA_DEALLOC_FLUSH

dealloc_type を AP_FLUSH に設定した [MC_]DEALLOCATE verb の機能を実行します。 これは、[MC_]SEND_DATA に続けて [MC_]DEALLOCATE を発行することに相当します。基本会話の SEND_DATA verb の場合、この verb で送信するデータは 1 つの完全な論理レコードであるか、論理レコードの終わりである必要があります。この値は、不完全な論理レコードを送信する場合には使用できません。

AP_SEND_DATA_DEALLOC_CONFIRM

このオプションは、半二重会話でのみ有効です。 *opext* パラメーターに AP_FULL_DUPLEX_CONVERSATION オプションが含まれている場合には、このオプションを使用しないでください。

dealloc_type を AP_CONFIRM_TYPE に設定した [MC_]DEALLOCATE verb の機能を実行します。これは、[MC_]SEND_DATA に続けて

[MC_]DEALLOCATE を発行することに相当します。基本会話の SEND_DATA verb の場合、この verb で送信するデータは 1 つの完全な論理レコードであるか、論理レコードの終わりである必要があります。この値は、不完全な論理レコードを送信する場合には使用できません。

AP_SEND_DATA_DEALLOC_SYNC_LEVEL

dealloc_type を AP_SYNC_LEVEL に設定した [MC_]DEALLOCATE verb の機能を実行します。これは、[MC_]SEND_DATA に続けて [MC_]DEALLOCATE を発行することに相当します。基本会話の SEND_DATA verb の場合、この verb で送信するデータは 1 つの完全な論理レコードであるか、論理レコードの終わりである必要があります。この値は、不完全な論理レコードを送信する場合には使用できません。

AP_SEND_DATA_DEALLOC_ABEND

dealloc_type を AP_ABEND に設定した MC_DEALLOCATE verb、または *dealloc_type* を AP_ABEND_PROG に設定した DEALLOCATE verb の機能を実行します。 これは、[MC_]SEND_DATA に続けて [MC_]DEALLOCATE を発行することに相当します。

次の場合には、[MC_]SEND_DATA を使用して [MC_]DEALLOCATE の機能を実行することはできません。

- 会話タイプが AP_BASIC_CONVERSATION で、 *dealloc_type* が AP_ABEND_SVC か AP_ABEND_TIMER でなければならない場合
- 会話の同期レベルが AP_SYNCPT で、TP が暗黙 FORGET 通知を要求する場合

このような場合には、[MC_]SEND_DATA と [MC_]DEALLOCATE を別々に発行する必要があります。詳細については、105 ページの『第 4 章 APPC 会話 verb』で [MC_]DEALLOCATE verb の説明を参照してください。

AIX, LINUX

data_type

送信するデータの形式を指定します。このパラメーターは、マップ式会話の MC_SEND_DATA verb のみが使用します。次の値があります。

AP_APPLICATION

標準 APPC アプリケーション・データ。 Communications Server は、データをアプリケーション・データ GDS 変数としてパートナー LU に送信します。

AP_USER_CONTROL_DATA

ユーザー制御データ。 Communications Server は、データをユーザー制御データ GDS 変数としてパートナー LU に送信します。このオプションは、パートナー LU がこの形式のデータを受け入れることができる場合以外、設定しないでください。

AP_PS_HEADER

PS ヘッダー・データ。このデータ形式は、同期点 TP のみが使用

します。この値は、会話の同期レベルが AP_SYNCPT の場合以外、設定しないでください。 同期点管理プログラムは、同期点コマンドを適切な PS ヘッダーに変換します。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

rts_rcvd

送信要求受信インディケーター。このパラメーターは、半二重会話でのみ使用できます。全二重会話では使用されません。

次の値があります。

AP_YES パートナー TP は、会話を Receive 状態に変更するようローカル TP に要求する [MC_]REQUEST_TO_SEND verb を発行しました。ローカル TP は、Receive 状態に変更するため、[MC_]PREPARE_TO_RECEIVE、[MC_]RECEIVE_AND_WAIT、[MC_]RECEIVE_AND_POST のいずれかの verb を使用できます。

AP_NO パートナー TP は [MC_]REQUEST_TO_SEND verb を発行しませんでした。

expd_rcvd

優先データ・インディケーター。

次の値があります。

AP_YES パートナー TP が、ローカル TP がまだ受信していない優先データを送信しました。このデータを受信するには、ローカル TP は [MC_]RECEIVE_EXPEDITED_DATA verb を使用することができます。

このインディケーターを、APPC verb 数に対して設定できます。これは、ローカル TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行してデータを受信するまで、後続の verb に対しても設定できます。

AP_NO 受信を待機している優先データはありません。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_LL
この戻りコードは、SEND_DATA verb にのみ適用されます。無効な値が含まれる論理レコードの論理レコード長フィールドは、0x0000、0x0001、0x8000、または 0x8001 です。詳細については、67 ページの『論理レコード』を参照してください。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

WINDOWS

AP_INVALID_DATA_SEGMENT
データが割り振りデータ・セグメントより長かったか、データ・バッファーのアドレスが間違っていました。

AP_SEND_DATA_INVALID_TYPE
type パラメーターが無効な値に設定されていました。

AIX, LINUX

AP_INVALID_FORMAT
format パラメーターが無効な値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

AP_SEND_TYPE_INVALID_FOR_FDX
アプリケーションがこの verb を全二重会話で発行しましたが、*type* パラメーターは、全二重会話で無効な送信タイプを指定しました。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc
次の値があります。

AP_SEND_DATA_NOT_SEND_STATE
ローカル TP は [MC_]SEND_DATA verb を発行しましたが、会話が Send と Send_Pending のどちらの状態でもありませんでした。

AP_SEND_DATA_CONFIRM_SYNC_NONE
ローカル TP は、*type* パラメーターを AP_SEND_DATA_CONFIRM に設定して [MC_]SEND_DATA verb を発行しましたが、会話の同期レベルが AP_NONE でした。 CONFIRM 機能が有効なのは、同期レベルが AP_CONFIRM_SYNC_LEVEL の場合のみです。

AP_SEND_DATA_NOT_LL_BDY
(基本会話の SEND_DATA verb でのみ戻ります) ローカル TP は、不完全な論理レコードを送信するために SEND_DATA verb を発行し、AP_NONE と AP_SEND_DATA_DEALLOC_ABEND のどちらでもない *type* パラメーターを使用しました。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING

AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

AIX, LINUX

primary_rc
AP_BACKED_OUT

secondary_rc
AP_BO_NO_RESYNC
AP_BO_RESYNC

[redacted]

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_PROG_ERROR_PURGING
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

[redacted]

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、MC_SEND_DATA verb が戻します。

primary_rc
AP_DEALLOC_ABEND

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、SEND_DATA verb が戻します。

primary_rc
AP_DEALLOC_ABEND_PROG

AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Send_Receive (全二重会話専用)、 Send または Send_Pending 状態でなければなりません。

状態の変更

次の表に要約した状態変更は、*primary_rc* パラメーターに基づいています。

primary_rc	新しい状態
AP_OK	Send (受信) (半二重会話) または変更なし (全二重会話)
AP_STATE_CHECK	変更なし
AP_PARAMETER_CHECK	
AP_CONVERSATION_TYPE_MIXED	
AP_INVALID_VERB	
AP_INVALID_VERB_SEGMENT	
AP_STACK_TOO_SMALL	
AP_TP_BUSY	
AP_UNEXPECTED_DOS_ERROR	
AP_ALLOCATION_ERROR	Reset (リセット)
AP_CONV_FAILURE_RETRY	Reset (リセット)
AP_CONV_FAILURE_NO_RETRY	
AP_DEALLOC_ABEND	Reset (リセット)
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	
AP_PROG_ERROR_PURGING	Receive (受信) (半二重会話)、
AP_SVC_ERROR_PURGING	Send_Receive (送信_受信) (全二重会話、 Send_Receive 状態で発行される verb)、または Reset (リセット) (全二重会話、 Send_Only 状態で発行される verb)

パートナー TP 待ち

[MC_]SEND_DATA verb は、パートナー TP が受信 verb を発行しないために無限に待機する場合があります。 その理由としては、送信バッファーがいっぱいになり、そのデータを受信するバッファーがパートナー LU にないために、APPC が送信バッファーの内容をパートナー LU へ伝送できないことが考えられます。

基本会話での論理レコード

基本会話の SEND_DATA verb を使用する場合、アプリケーションは論理レコードの形式で (各データ・レコードの冒頭に LLID フィールドを付けて) データを提供する必要があります。詳細については、 67 ページの『論理レコード』を参照してください。

AIX, LINUX

同期レベルが AP_SYNCPT の会話では、送信されるデータが PS ヘッダーの形式になっている場合があります。このことは、0x0001 の長さフィールドによって示されます。 同期点管理プログラムは、アプリケーションが必要とする同期点機能に基づいて、適切な PS ヘッダーをセットアップします。

MC_SEND_ERROR および SEND_ERROR

MC_SEND_ERROR verb または SEND_ERROR verb は、ローカル TP がアプリケーション・レベルのエラーを検出したことをパートナー TP に通知します。

ローカル TP は、エラー通知を即時にパートナー TP へ送信し、情報をローカル LU の送信バッファーに保持しません。

半二重会話では、この verb が正常に実行されると、会話は、ローカル TP にとっては Send 状態になり、パートナー TP 側では Receive 状態になります。全二重会話では、この verb が正常に実行された後に状態の変更はありません。

VCB 構造体: MC_SEND_ERROR

AIX, LINUX

MC_SEND_ERROR verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_send_error
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved            */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    rts_rcvd;
  unsigned char    err_type;
  unsigned char    err_dir;
  unsigned char    expd_rcvd;
  unsigned char    reserv5[2];
  unsigned char    reserv6[4];
} MC_SEND_ERROR;
```

VCB 構造体: SEND_ERROR

SEND_ERROR verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct send_error
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved            */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
```

```
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    rts_rcvd;
  unsigned char    err_type;
  unsigned char    err_dir;
  unsigned char    expd_rcvd;
  AP_UINT16        log_dlen;
  unsigned char    *log_dptr;
} SEND_ERROR;
```

VCB 構造体: MC_SEND_ERROR (Windows)

WINDOWS

MC_SEND_ERROR verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_send_error
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     rts_rcvd;
  unsigned char     reserv3;
  unsigned char     err_dir;
  unsigned char     reserv4;
  unsigned char     reserv5[2];
  unsigned char     reserv6[4];
} MC_SEND_ERROR;
```

VCB 構造体: SEND_ERROR (Windows)

SEND_ERROR verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct send_error
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     rts_rcvd;
  unsigned char     err_type;
  unsigned char     err_dir;
  unsigned char     reserv3;
  unsigned short    log_dlen;
  unsigned char far *log_dptr;
} SEND_ERROR;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_SEND_ERROR
MC_SEND_ERROR verb の場合

AP_B_SEND_ERROR
SEND_ERROR verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_SEND_ERROR verb の場合

AP_BASIC_CONVERSATION
SEND_ERROR verb の場合

verb が全二重会話で発行されている場合、または非ブロッキング verb として発行されている場合、上記の値を次の値のいずれかまたは両方と (論理 OR を使用して) 結合します。

AP_FULL_DUPLEX_CONVERSATION
この verb は全二重会話で発行されます。

AP_NON_BLOCKING
この verb は非ブロッキング verb として発行されます。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

err_type
報告するエラーのタイプを示します。これは、APPC がエラーを報告するためにパートナー TP へ送信する戻りコードを決定します。これらのすべての戻りコードは、 307 ページの『付録 B. 共通戻りコード』で説明しています。

WINDOWS

このパラメーターは、基本会話 の SEND_ERROR verb のみが使用します。

次の値があります。

AP_PROG
エラーは、同期点を使用していないアプリケーション・プログラムへ報告されます。 この値を指定すると、APPC はパートナー TP へ次のいずれかの戻りコードを送信します。

- AP_PROG_ERROR_TRUNC (論理レコードの一部を送信した後、SEND_ERROR verb が Send 状態で発行された場合)

- AP_PROG_ERROR_NO_TRUNC (MC_SEND_ERROR verb が Send 状態で発行された場合、または SEND_ERROR verb が Send 状態で発行され、不完全な論理レコードが送信されていない場合)
- AP_PROG_ERROR_PURGING (この verb が Send 以外の状態で発行された場合)

AP_SVC エラーは、サービス・プログラムへ報告されます。この値は、SEND_ERROR verb のみが使用します。 この値を指定すると、APPC はパートナー TP へ次のいずれかの戻りコードを送信します。

- AP_SVC_ERROR_TRUNC (論理レコードの一部を送信した後、SEND_ERROR verb が Send 状態で発行された場合)
- AP_SVC_ERROR_NO_TRUNC (SEND_ERROR verb が Send 状態で発行され、不完全な論理レコードが送信されていない場合)
- AP_SVC_ERROR_PURGING (SEND_ERROR verb が Send 以外の状態で発行された場合)

AIX, LINUX

AP_BACKOUT_NO_RESYNC
この値は、会話の同期レベルが AP_SYNCPT の場合にのみ使用できます。 ローカル TP (または、同じ作業論理単位に参加している別の TP) は BACKOUT 要求を発行し、ローカル TP はリソースのバックアウトを完了しました。 同期点管理プログラムは、BACKOUT 要求を受信したときに、この値を設定して [MC_]SEND_ERROR を発行します。 この値を指定すると、APPC は AP_BACKED_OUT および AP_BO_NO_RESYNC の 1 次および 2 次戻りコードをパートナー TP へ送信します。

AP_BACKOUT_RESYNC
この値は、会話の同期レベルが AP_SYNCPT の場合にのみ使用できます。 ローカル TP (または、同じ作業論理単位に参加している別の TP) はバックアウト要求を発行し、再同期がまだ進行中です。 同期点管理プログラムは、BACKOUT 要求を受信したときに、この値を設定して [MC_]SEND_ERROR を発行します。 この値を指定すると、APPC は AP_BACKED_OUT および AP_BO_RESYNC の 1 次および 2 次戻りコードをパートナー TP へ送信します。

err_dir 報告されるエラーが、パートナー TP から受信したデータの中にあるのか、ローカル TP が送信しようとしていたデータの中にあるのかを示します。

全二重会話では、このパラメーターは AP_SEND_DIR_ERROR に設定してください。半二重会話では、このパラメーターは、[MC_]SEND_ERROR verb が Send_Pending 状態で発行される場合にのみ使用されます。

次の値があります。

AP_RCV_DIR_ERROR
ローカル TP は、リモート TP から受信したデータの中でエラーを検出しました。

AP_SEND_DIR_ERROR
ローカル TP はそれ自体のデータの中でエラーを検出した (たとえば、ディスクからデータを読み取れなかったなど) か、それ自体の処理の中でエラーを検出しました。

log_dlen
エラー・ログ・ファイルへ送信するデータのバイト数。このパラメーターは、SEND_ERROR verb のみが使用します。

この値の範囲は、0 ～ 32,767 です。 長さ 0 (ゼロ) は、エラー・ログ・データがないことを示しています。

log_dptr
エラー情報が入っているデータ・バッファーのアドレス。このデータは、ローカル・エラー・ログとパートナー LU へ送られます。

このパラメーターは、*log_dlen* が 0 (ゼロ) より大きい場合に SEND_ERROR verb が使用します。

TP は、エラー・データを汎用データ・ストリーム (GDS) エラー・ログ変数として形式設定する必要があります。詳細については、IBM 資料の「*Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*」を参照してください。

WINDOWS

データ・バッファーは、静的データ域か、グローバル割り振り域にあります。データ・バッファーは、この領域と完全に一致していなければなりません。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

rts_rcvd
送信要求受信インディケーター。このパラメーターは、半二重会話でのみ使用できます。全二重会話では使用されません。次の値があります。

AP_YES パートナー TP は、会話を Receive 状態に変更するようローカル TP に要求する [MC_]REQUEST_TO_SEND verb を発行しました。

ローカル TP は、Receive 状態に変更するため、[MC_]PREPARE_TO_RECEIVE、[MC_]RECEIVE_AND_WAIT、[MC_]RECEIVE_AND_POST のいずれかの verb を使用できます。

AP_NO パートナー TP は [MC_]REQUEST_TO_SEND verb を発行しませんでした。

expd_rcvd

優先データ・インディケーター。

次の値があります。

AP_YES パートナー TP が、ローカル TP がまだ受信していない優先データを送信しました。このデータを受信するには、ローカル TP は [MC_]RECEIVE_EXPEDITED_DATA verb を使用することができます。

このインディケーターを、APPC verb 数に対して設定できます。これは、ローカル TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行してデータを受信するまで、後続の verb に対しても設定できます。

AP_NO 受信を待機している優先データはありません。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc

AP_PARAMETER_CHECK

secondary_rc

次の値があります。

AP_BAD_CONV_ID

conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_ERROR_DIRECTION

err_dir の値が有効ではありませんでした。

AP_BAD_TP_ID

tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_SEND_ERROR_BAD_TYPE

err_type の値が有効ではありませんでした。

AIX, LINUX

AP_INVALID_FORMAT
予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

WINDOWS

AP_INVALID_DATA_SEGMENT
ログ・データが割り振りデータ・セグメントより長かったか、ログ・データ・バッファーのアドレスが間違っていました。

次の *secondary_rc* 値は、SEND_ERROR verb のみが戻すことができます。

AP_SEND_ERROR_LOG_LL_WRONG
エラー・ログ GDS 変数の LL フィールドが、実際のデータの長さに一致しませんでした。

状態検査: この verb で状態検査エラーは発生しません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

許容されるいずれかの状態で verb を発行した場合: 次の戻りコードは、許容されるいずれかの状態で [MC_]SEND_ERROR verb を発行した場合に生成されることがあります。

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

Send 状態で verb を発行した場合: 次の戻りコードは、Send 状態で [MC_]SEND_ERROR verb を発行した場合にのみ生成されることがあります。

primary_rc

AP_ALLOCATION_ERROR

secondary_rc

AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

AIX, LINUX

primary_rc

AP_BACKED_OUT

secondary_rc

AP_BO_NO_RESYNC
AP_BO_RESYNC

primary_rc

AP_PROG_ERROR_PURGING

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の戻りコードは、Send 状態で MC_SEND_ERROR verb を発行した場合にのみ生成されることがあります。

primary_rc

AP_DEALLOC_ABEND

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の戻りコードは、Send 状態で SEND_ERROR verb を発行した場合にのみ生成されることがあります。

primary_rc

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

Receive 状態で verb を発行した場合: 次の戻りコードは、この verb を Receive 状態で発行した場合にのみ生成されることがあります。

primary_rc

AP_DEALLOC_NORMAL

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Reset 状態を除き、どのような状態にあっても構いません。

状態の変更

新しい状態は、1 次戻りコード *primary_rc* によって決定されます。 考えられる状態の変更を次の表に要約します。

primary_rc	**新しい状態**
AP_OK	Send (受信) (半二重会話) または変更なし (全二重会話)
AP_PARAMETER_CHECK	変更なし
AP_CONVERSATION_TYPE_MIXED	
AP_INVALID_VERB	
AP_INVALID_VERB_SEGMENT	
AP_STACK_TOO_SMALL	
AP_TP_BUSY	
AP_UNEXPECTED_DOS_ERROR	
AP_ALLOCATION_ERROR	Reset (リセット)
AP_CONV_FAILURE_RETRY	Reset (リセット)
AP_CONV_FAILURE_NO_RETRY	

primary_rc	新しい状態
AP_DEALLOC_ABEND AP_DEALLOC_ABEND_PROG AP_DEALLOC_ABEND_SVC AP_DEALLOC_ABEND_TIMER AP_DEALLOC_NORMAL	Reset (リセット)
AP_PROG_ERROR_PURGING AP_SVC_ERROR_PURGING	Receive (受信) (半二重会話)、 Send_Receive (送信_受信) (全二重会話、 Send_Receive 状態で発行される verb)、または Reset (リセット) (全二重会話、 Send_Only 状態で発行される verb)

データの除去

会話が Receive 状態にあるときに TP から [MC_]SEND_ERROR verb を発行すると、着信データは APPC によって除去されます。除去されるデータは、次のとおりです。

- [MC_]SEND_DATA verb によって送信されたデータ
- 戻りコード・インディケーター
- 確認要求
- 割り振り解除要求

APPC は、着呼 REQUEST_TO_SEND インディケーターを除去しません。

戻りコード・インディケーターの除去

次の 1 次戻りコードは、リモート TP または LU がエラーを検出したことを示し、通常、ローカル TP が次に発行した APPC verb で報告されます。 ただし、ローカル TP が [MC_]SEND_ERROR verb を発行した場合、これらの戻りコードは除去され、他の戻りコードに置き換えられます。

次の戻りコード・インディケーターは、除去された場合、 1 次戻りコード AP_OK に置き換えられます。

```
AP_PROG_ERROR_NO_TRUNC
AP_PROG_ERROR_PURGING
AP_PROG_ERROR_TRUNC
AP_SVC_ERROR_NO_TRUNC
AP_SVC_ERROR_PURGING
AP_SVC_ERROR_TRUNC
```

次の戻りコード・インディケーターは、除去された場合、 1 次戻りコード AP_DEALLOC_NORMAL に置き換えられます。

primary_rc

```
AP_ALLOCATION_ERROR
```

secondary_rc

```
AP_ALLOCATION_FAILURE_NO_RETRY
AP_ALLOCATION_FAILURE_RETRY
AP_CONVERSATION_TYPE_MISMATCH
```

AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY
AP_SEC_BAD_PROTOCOL_VIOLATION
AP_SEC_BAD_PASSWORD_EXPIRED
AP_SEC_BAD_PASSWORD_INVALID
AP_SEC_BAD_USERID_REVOKED
AP_SEC_BAD_USERID_INVALID
AP_SEC_BAD_USERID_MISSING
AP_SEC_BAD_PASSWORD_MISSING
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
AP_SEC_BAD_UNAUTHRZD_AT_RLU
AP_SEC_BAD_UNAUTHRZD_FROM_LLU
AP_SEC_BAD_UNAUTHRZD_TO_TP
AP_SEC_BAD_INSTALL_EXIT_FAILED
AP_SEC_BAD_PROCESSING_FAILURE

primary_rc

AP_DEALLOC_ABEND
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER

MC_SEND_EXPEDITED_DATA および SEND_EXPEDITED_DATA

MC_SEND_EXPEDITED_DATA または SEND_EXPEDITED_DATA verb は、パートナー TP へ伝送するため、データをローカル LU の優先送信バッファーに入れます。

ローカル LU の送信バッファー内に収集されたデータは、 [MC_]SEND_DATA verb と同じ方法で、パートナー LU (およびパートナー TP) へ伝送されます。ただし、データは優先データとしてネットワークをまたいで送信されるため、[MC_]SEND_DATA を使用して先に送信されたデータの前に到着する可能性があります。

VCB 構造体: MC_SEND_EXPEDITED_DATA

MC_SEND_EXPEDITED_DATA verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_send_expedited_data
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved            */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    rts_rcvd;
```

```
  unsigned char    expd_rcvd;
  AP_UINT16        dlen;
  unsigned char    *dptr;
  unsigned char    reserv4[2];
} MC_SEND_EXPEDITED_DATA;
```

VCB 構造体: SEND_EXPEDITED_DATA

SEND_EXPEDITED_DATA verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct send_expedited_data
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                  /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    rts_rcvd;
  unsigned char    expd_rcvd;
  AP_UINT16        dlen;
  unsigned char    *dptr;
  unsigned char    reserv4[2];
} SEND_EXPEDITED_DATA;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_SEND_EXPEDITED_DATA
MC_SEND_EXPEDITED_DATA verb の場合

AP_B_SEND_EXPEDITED_DATA
SEND_EXPEDITED_DATA verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_SEND_EXPEDITED_DATA verb の場合

AP_BASIC_CONVERSATION
SEND_EXPEDITED_DATA verb の場合

verb が全二重会話で発行されている場合、または非ブロッキング verb として発行されている場合、上記の値を次の値のいずれかまたは両方と (論理 OR を使用して) 結合します。

AP_FULL_DUPLEX_CONVERSATION
この verb は全二重会話で発行されます。

AP_NON_BLOCKING
この verb は非ブロッキング verb として発行されます。

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

dlen ローカル LU の送信バッファーに入れるデータのバイト数。この値の範囲は、0 ～ 86 です。

dptr ローカル LU の送信バッファーに入れるデータが入っているバッファーのアドレス。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

rts_rcvd

送信要求受信インディケーター。このパラメーターは、半二重会話でのみ使用できます。全二重会話では使用されません。

次の値があります。

AP_YES パートナー TP は、会話を Receive 状態に変更するようローカル TP に要求する [MC_]REQUEST_TO_SEND verb を発行しました。ローカル TP は、Receive 状態に変更するため、[MC_]PREPARE_TO_RECEIVE、[MC_]RECEIVE_AND_WAIT、[MC_]RECEIVE_AND_POST のいずれかの verb を使用できます。

AP_NO パートナー TP は [MC_]REQUEST_TO_SEND verb を発行しませんでした。

expd_rcvd

優先データ・インディケーター。

次の値があります。

AP_YES パートナー TP が、ローカル TP がまだ受信していない優先データを送信しました。このデータを受信するには、ローカル TP は [MC_]RECEIVE_EXPEDITED_DATA verb を使用することができます。

このインディケーターを、APPC verb 数に対して設定できます。これは、ローカル TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行してデータを受信するまで、後続の verb に対しても設定できます。

AP_NO 受信を待機している優先データはありません。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

優先データがサポートされない: リモート LU が優先データをサポートしないために verb が実行されない場合には、 APPC は次のパラメーターを戻します。

primary_rc

AP_EXPD_NOT_SUPPORTED_BY_LU

会話の割り振りが解除された場合: パートナー TP が会話の割り振りを解除した場合、APPC は次の値を戻します。

primary_rc

AP_CONVERSATION_ENDED
この verb は、非ブロッキング verb として発行され、先に発行された verb の後のキューに入ります。パートナー TP は、上記の AP_DEALLOC_NORMAL に [MC_]DEALLOCATE verb を発行し、 キューの最初の verb が *primary_rc* を AP_DEALLOC_NORMAL に設定して戻ります。これは、会話の終了を示します。次に、キューに入っている後続の verb が、*primary_rc* を AP_CONVERSATION_ENDED に設定して戻ります。これは、verb が処理される前に、会話がすでに終了していたことを示します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_SEND_EXPD_INVALID_LENGTH
dlen パラメーターが無効な値に設定されていました。

AP_INVALID_FORMAT
予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

状態検査: TP がこの verb を発行したとき、会話が誤った状態にあった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_STATE_CHECK

secondary_rc
次の値があります。

AP_EXPD_DATA_BAD_CONV_STATE
ローカル TP は [MC_]SEND_DATA verb を発行しましたが、会話が Reset 状態ではありませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
AP_CONVERSATION_TYPE_MISMATCH
AP_PIP_NOT_ALLOWED
AP_PIP_NOT_SPECIFIED_CORRECTLY
AP_SECURITY_NOT_VALID
AP_SYNC_LEVEL_NOT_SUPPORTED
AP_TP_NAME_NOT_RECOGNIZED
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
AP_TRANS_PGM_NOT_AVAIL_RETRY

primary_rc
AP_BACKED_OUT

secondary_rc
AP_BO_NO_RESYNC
AP_BO_RESYNC

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_CONV_FAILURE_NO_RETRY
AP_CONV_FAILURE_RETRY
AP_CONVERSATION_TYPE_MIXED
AP_DUPLEX_TYPE_MIXED
AP_PROG_ERROR_PURGING
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、MC_SEND_EXPEDITED_DATA verb が戻します。

primary_rc
AP_DEALLOC_ABEND

APPC は、この 1 次戻りコードでは 2 次戻りコードを戻しません。

次の 1 次戻りコードは、SEND_EXPEDITED_DATA verb が戻します。

primary_rc
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Reset 状態を除き、どのような状態にもなります。

状態の変更

次の表に要約した状態変更は、*primary_rc* パラメーターに基づいています。

primary_rc	新しい状態
AP_OK	変更なし
AP_STATE_CHECK	変更なし
AP_PARAMETER_CHECK	
AP_CONVERSATION_TYPE_MIXED	
AP_INVALID_VERB	
AP_INVALID_VERB_SEGMENT	
AP_STACK_TOO_SMALL	
AP_TP_BUSY	
AP_UNEXPECTED_DOS_ERROR	
AP_ALLOCATION_ERROR	Reset (リセット)
AP_CONV_FAILURE_RETRY	Reset (リセット)
AP_CONV_FAILURE_NO_RETRY	
AP_CONVERSATION_ENDED	
AP_DEALLOC_ABEND	Reset (リセット)
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	

パートナー TP 待ち

[MC_]SEND_DATA と同様に、パートナー TP が [MC_]RECEIVE_EXPEDITED_DATA verb を発行しないため、[MC_]SEND_EXPEDITED_DATA verb は無限に待機する可能性があります。その理由としては、送信優先バッファーがいっぱいになり、そのデータを受信するバッファーがパートナー LU にないために、APPC が送信バッファーの内容をパートナー LU へ伝送できないことが考えられます。

MC_TEST_RTS および TEST_RTS

MC_TEST_RTS verb または TEST_RTS verb は、パートナー TP から REQUEST_TO_SEND 通知を受信したかどうかを判別します。

注: この verb は、半二重会話でのみ使用できます。全二重会話では無効です。

通常、パートナー TP が [MC_]REQUEST_TO_SEND verb を発行した場合、ローカル TP はそのことを後続の verb の *rts_rcvd* パラメーター (これは多数の verb で受信されるパラメーターです) によって通知されます。これは、このパラメーターを戻すことができる後続の最初の verb でのみ報告され、それ以後の verb では報告されません。 [MC_]TEST_RTS verb を使用すると、ローカル TP が最後に Receive 状態にあった後、送信要求通知を受信したかどうかをローカル TP で検査できます。

アプリケーションで [MC_]TEST_RTS を繰り返し発行する代わりに、 269 ページの『MC_TEST_RTS_AND_POST および TEST_RTS_AND_POST』で説明する [MC_]TEST_RTS_AND_POST を使用できます。この verb は、パートナー TP から REQUEST_TO_SEND 通知を受信した時点で非同期的に戻ります。 [MC_]TEST_RTS_AND_POST は [MC_]RECEIVE_AND_POST と同様に非同期に動作するため、この verb が未解決の間、アプリケーションから別の APPC verb を発行できます。

VCB 構造体: MC_TEST_RTS

AIX, LINUX

MC_TEST_RTS verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_test_rts
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved            */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    reserv3;
} MC_TEST_RTS;
```

VCB 構造体: TEST_RTS

TEST_RTS verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct test_rts
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                 /* Reserved            */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  unsigned char    reserv3;
} TEST_RTS;
```

VCB 構造体: MC_TEST_RTS (Windows)

WINDOWS

MC_TEST_RTS verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_test_rts
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     reserv3;
} MC_TEST_RTS;
```

VCB 構造体: TEST_RTS (Windows)

TEST_RTS verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct test_rts
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     reserv3;
} TEST_RTS;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_TEST_RTS
MC_TEST_RTS verb の場合

AP_B_TEST_RTS
TEST_RTS verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_TEST_RTS verb の場合

AP_BASIC_CONVERSATION
TEST_RTS verb の場合

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id
会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
パートナー TP から REQUEST_TO_SEND 通知を受信したかどうかを示します。次の値があります。

AP_OK REQUEST_TO_SEND 通知を受信しました。

AP_UNSUCCESSFUL
REQUEST_TO_SEND 通知を受信しませんでした。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_CONV_ID
conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID
tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AIX, LINUX

AP_INVALID_FORMAT
予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED
アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コール

バック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

AP_TEST_INVALID_FOR_FDX
ローカル TP は、全二重会話の中で [MC_]TEST_RTS verb を使用しようとしました。この verb は、半二重会話でのみ使用できます。

状態検査: この verb で状態検査エラーは発生しません。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
AP_CONVERSATION_TYPE_MIXED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR

WINDOWS

AP_COMM_SUBSYSTEM_NOT_LOADED
AP_STACK_TOO_SMALL
AP_INVALID_VERB_SEGMENT

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP からこの verb を発行する場合、会話は Reset 状態を除き、どのような状態にあっても構いません。

状態の変更

この verb では、会話状態は変更されません。

MC_TEST_RTS_AND_POST および TEST_RTS_AND_POST

MC_TEST_RTS_AND_POST verb または TEST_RTS_AND_POST verb は、パートナー TP から REQUEST_TO_SEND 通知を受信したことをアプリケーションに通知します。

注: この verb は、半二重会話でのみ使用できます。全二重会話では無効です。

通常、パートナー TP が [MC_]REQUEST_TO_SEND verb を発行した場合、ローカル TP はそのことを後続の verb の *rts_rcvd* パラメーター (これは多数の verb で受信されるパラメーターです) によって通知されるか、[MC_]TEST_RTS verb の正常な戻りコードによって通知されます。 [MC_]TEST_RTS_AND_POST verb を使用すると、ローカル TP は、通知を取得するために verb を繰り返し発行しない場合でも、 REQUEST_TO_SEND 通知が到着したときに、それを非同期に受信できます。

VCB 構造体: MC_TEST_RTS_AND_POST

AIX, LINUX

MC_TEST_RTS_AND_POST verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_test_rts_and_post
{
  AP_UINT16        opcode;
  unsigned char    opext;
  unsigned char    format;                    /* Reserved             */
  AP_UINT16        primary_rc;
  AP_UINT32        secondary_rc;
  unsigned char    tp_id[8];
  AP_UINT32        conv_id;
  void             (*callback)();
  unsigned char    reserv3;
} MC_TEST_RTS_AND_POST;
```

VCB 構造体: TEST_RTS_AND_POST

TEST_RTS_AND_POST verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct test_rts_and_post
{
  AP_UINT16         opcode;
  unsigned char     opext;
  unsigned char     format;                 /* Reserved             */
  AP_UINT16         primary_rc;
  AP_UINT32         secondary_rc;
  unsigned char     tp_id[8];
  AP_UINT32         conv_id;
  void              (*callback)();
  unsigned char     reserv3;
} TEST_RTS_AND_POST;
```

VCB 構造体: MC_TEST_RTS_AND_POST (Windows)

WINDOWS

MC_TEST_RTS_AND_POST verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct mc_test_rts_and_post
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     reserv3;
  unsigned long     sema;
} MC_TEST_RTS_AND_POST;
```

VCB 構造体: TEST_RTS_AND_POST (Windows)

TEST_RTS_AND_POST verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct test_rts_and_post
{
  unsigned short    opcode;
  unsigned char     opext;
  unsigned char     reserv2;
  unsigned short    primary_rc;
  unsigned long     secondary_rc;
  unsigned char     tp_id[8];
  unsigned long     conv_id;
  unsigned char     reserv3;
  unsigned long     sema;
} TEST_RTS_AND_POST;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode 次の値があります。

AP_M_TEST_RTS_AND_POST
MC_TEST_RTS_AND_POST verb の場合

AP_B_TEST_RTS_AND_POST
TEST_RTS_AND_POST verb の場合

opext 次の値があります。

AP_MAPPED_CONVERSATION
MC_TEST_RTS_AND_POST verb の場合

AP_BASIC_CONVERSATION
TEST_RTS_AND_POST verb の場合

tp_id ローカル TP の ID。

このパラメーターの値は、呼び出し元 TP 内の TP_STARTED verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

conv_id

会話 ID。

このパラメーターの値は、呼び出し元 TP 内の [MC_]ALLOCATE verb が戻した値か、呼び出し先 TP 内の RECEIVE_ALLOCATE が戻した値です。

AIX, LINUX

callback

REQUEST_TO_SEND 通知を受信したときに、APPC が呼び出すコールバック・ルーチンのアドレス。詳細については、273 ページの『使用上の注意』を参照してください。

WINDOWS

sema 2 つの Windows 関数`CreateEvent` または `OpenEvent` のいずれかを呼び出して得られる Windows のイベント・ハンドル。APPC は、このイベント・ハンドルに信号を送り、REQUEST_TO_SEND 通知の受信時期を TP に知らせます。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

注: この verb が発行されると、この verb が正常に発行されたかどうかを示す *primary_rc* を伴って即時に戻ります。 この段階で有効な戻りパラメーターは、*primary_rc* と *secondary_rc* のみです (*primary_rc* が `AP_OK` でない場合)。 考えられる *primary_rc* と *secondary_rc* の値については、この項で後述します。

この *primary_rc* が `AP_OK` の場合、この verb は REQUEST_TO_SEND 通知の待機を正常に開始しました。 APPC は、この verb が (通知を受信したか、会話の終わりまたはエラーによって終了したために) 完了した時点で、指定されたコールバック・ルーチンを呼び出します。 この時点で、戻りパラメーターは次に示すとおりです。 *primary_rc* パラメーターと *secondary_rc* パラメーターには、REQUEST_TO_SEND 通知を受信したかどうかを示す新しい値が入っており、これらのパラメーターを再度検査する必要があります。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

`AP_OK` REQUEST_TO_SEND 通知を受信しました。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc

AP_PARAMETER_CHECK

secondary_rc

次の値があります。

AP_BAD_CONV_ID

conv_id の値が、APPC によって割り当てられた会話 ID に一致しませんでした。

AP_BAD_TP_ID

tp_id の値が、APPC によって割り当てられた TP ID に一致しませんでした。

AP_INVALID_FORMAT

予約済みフィールド *format* がゼロ以外の値に設定されていました。

AP_SYNC_NOT_ALLOWED

アプリケーションが同期 APPC エントリー・ポイントを使用して、この verb をコールバック・ルーチンの中で発行しました。コールバック・ルーチンから発行する verb は、必ず非同期エントリー・ポイントを使用する必要があります。

AP_INVALID_CALLBACK_HANDLE

callback パラメーターがヌル・ポインターに設定されており、同期エントリー・ポイントを使用して (またはコールバック・ルーチンへのヌル・ポインターと共に非同期エントリー・ポイントを使用して) verb が発行されました。 詳細については、273 ページの『使用上の注意』を参照してください。

AP_TEST_INVALID_FOR_FDX

ローカル TP は、全二重会話の中で [MC_]TEST_RTS_AND_POST verb を使用しようとしました。この verb は、半二重会話でのみ使用できます。

状態検査: この verb で状態検査エラーは発生しません。

verb が取り消された場合: この戻りコードは、初期戻りコードとして戻ることはなく、初期戻りコードが AP_OK の場合に、それに続く戻りコードとしてのみ戻ります。 TP が発行した別の verb によってこの verb が取り消されたためにこの verb が実行されなかった場合、APPC は次のパラメーターを戻します。

primary_rc

AP_CANCELLED

ローカル TP は、[MC_]TEST_RTS_AND_POST が未解決のときに、次のいずれかの verb を発行しました。

- *dealloc_type* が AP_ABEND_PROG、AP_ABEND_SVC、または AP_ABEND_TIMER に設定された DEALLOCATE
- *dealloc_type* が AP_ABEND に設定された MC_DEALLOCATE
- [MC_]SEND_ERROR
- TP_ENDED

これらの verb の 1 つを発行すると、
[MC_]TEST_RTS_AND_POST verb は取り消されます。 コールバック・ルーチンは呼び出されません。

会話の終了: 会話が終了したためにこの verb が戻った場合、APPC は次のパラメーターを戻します。

primary_rc
AP_UNSUCCESSFUL

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は 1 次戻りコードを (該当する場合は 2 次戻りコードも) 戻します。それらの戻りコードについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

次の戻りコードがあります。

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR
AP_CONVERSATION_TYPE_MIXED
AP_INVALID_VERB
AP_TP_BUSY

APPC は、これらの 1 次戻りコードでは 2 次戻りコードを戻しません。

発行時の状態

TP は、会話が Reset 以外の状態のときに [MC_]TEST_RTS_AND_POST を発行できます。

状態の変更

この verb では、会話状態は変更されません。

使用上の注意

この項では、次のトピックについて、使用上の追加情報を記載します。

- コールバック・ルーチン
- verb が保留中の処理
- TP での verb の使用法
- 無限待機の回避

コールバック・ルーチン

AIX, LINUX

アプリケーションは、VCB のパラメーターの 1 つとしてコールバック・ルーチンへのポインターを提供します。この項では、Communications Server がそのルーチンをどのように使用するか、および、そのルーチンが実行する必要がある機能について説明します。

コールバック・ルーチンは、次のように定義されています。

```
void (*callback) (
                    void *          vcb,
                    unsigned char   tp_id[8],
                    AP_UINT32       conv_id
                  );
```

Communications Server は、次のパラメーターを使用してルーチンを呼び出します。

vcb アプリケーションが提供した VCB へのポインター (Communications Server が設定した戻りパラメーターを含む)。

tp_id verb を発行した TP の 8 バイトからなる TP ID。

conv_id
verb を発行した会話の会話 ID。

コールバック・ルーチンでは、これらのパラメーターをすべて使用する必要はありません。コールバック・ルーチンは、戻された VCB に対して必要なすべての処理を実行するか、単に verb が完了したことをメインプログラムに通知するために変数を設定することもできます。

アプリケーションは、必要であればそのコールバック・ルーチンの中からさらに APPC verb を発行することができます。 ただし、それらの verb は非同期 verb である必要があります。コールバック・ルーチンの中から発行した同期 verb は、AP_PARAMETER_CHECK と AP_SYNC_NOT_ALLOWED の戻りコードでリジェクトされます。

注: アプリケーションが非同期 APPC エントリー・ポイントを使用して [MC_]TEST_RTS_AND_POST verb を発行する場合は、2 つのコールバック・ルーチンが指定されます。1 つは VCB の中で、もう 1 つはエントリー・ポイントのパラメーターとして指定されます。 一般に、APPC は VCB の中で指定されたコールバック・ルーチンを使用し、エントリー・ポイントのコールバック・ルーチンは無視します。しかし、アプリケーションが VCB 内のコールバック・ルーチンにヌル・ポインターを指定した場合、APPC はエントリー・ポイントのコールバック・ルーチンを使用します。

verb が保留中の他の処理の続行

[MC_]TEST_RTS_AND_POST verb はデータの到着を待たずに即時に戻るため、TP はこの verb の完了を待つ間、他の処理を続行できます。 ただし、次の点に注意してください。

- [MC_]TEST_RTS_AND_POST verb へ提供された VCB は、コールバック・ルーチンが戻るまで、引き続き使用されます。 その間、TP で VCB 内のフィールドに変更を加えてはなりません。 [MC_]TEST_RTS_AND_POST が未解決の間に、TP から別の APPC verb を発行する場合は、新しい verb に別の VCB を使用する必要があります。
- 1 つの会話につき、一度に 1 つの [MC_]TEST_RTS_AND_POST verb のみを活動状態にできます。

TP での verb の使用法

[MC_]TEST_RTS_AND_POST verb を使用するには、ローカル TP で次の手順を実行します。

[MC_]TEST_RTS_AND_POST を使用するには

1. [MC_]TEST_RTS_AND_POST verb を発行します。
2. 次のように、1 次戻りコード *primary_rc* の値を検査します。
   - 1 次戻りコードが AP_OK である場合、この verb はパートナー TP からの REQUEST_TO_SEND 通知を待機します。ローカル TP は、データを非同期的に受信している間、次のことを実行できます。
     – その会話に関連していないタスクを実行する
     – その会話に対して別の APPC verb を発行する
     – 次のいずれかの verb を発行することにより、[MC_]TEST_RTS_AND_POST verb を途中で取り消す
       - *dealloc_type* が AP_ABEND_PROG、AP_ABEND_SVC、または AP_ABEND_TIMER に設定された DEALLOCATE
       - *dealloc_type* が AP_ABEND に設定された MC_DEALLOCATE
       - SEND_ERROR
       - TP_ENDED
   - ただし、1 次戻りコードが AP_OK でない場合、 [MC_]TEST_RTS_AND_POST verb は失敗しました。 その場合、ローカル TP はステップ 3 と 4 を実行しません。
3. コールバック・ルーチン (この verb でパラメーターとして指定されたもの) が APPC によって呼び出されたかどうかを検査します。 APPC は、パートナー TP から REQUEST_TO_SEND 通知を受信した時点で、このルーチンを呼び出します。
4. 1 次戻りコード *primary_rc* の新しい値を検査します。
   - 1 次戻りコードが AP_OK である場合、パートナー TP は [MC_]REQUEST_TO_SEND を発行しました。
   - 1 次戻りコードが AP_OK でない場合は、アプリケーションで *primary_rc* パラメーターと *secondary_rc* パラメーターを検査し、次に実行するアクションを判別する必要があります。

無限待機の回避

ローカル TP が [MC_]TEST_RTS_AND_POST verb を発行し、その後、コールバック・ルーチンが呼び出されるのを待つ場合、ローカル TP はパートナー TP から REQUEST_TO_SEND 通知を受信するまで中断状態になります。パートナー TP が [MC_]REQUEST_TO_SEND を発行しなければ、ローカル TP は無限に待機する可能性があります。 TP を切れ目なく動作させる必要がある場合は、コールバック・ルーチンを待機しないようにするか、 [MC_]TEST_RTS verb を使用します。

第 5 章 TP サーバー verb

AIX, LINUX

この章には、個々の APPC TP サーバー verb の説明が記載されています。それぞれの verb について、次の情報を提供します。

- 各 verb の定義。
- 各 verb が使用する verb 制御ブロック (VCB) の構造体定義。この構造体は、TP ヘッダー・ファイル **/usr/include/sna/tpsrv_c.h** (AIX) または **/opt/ibm/sna/include/tpsrv_c.h** (Linux) に定義されています。 (*rsrvd* で始まるパラメーターは予約されています。)
- APPC へ提供するか APPC から戻されるパラメーター (VCB フィールド)。各パラメーターについて、次の情報を提供します。
  - 説明
  - 指定可能な値
  - 追加情報
- 各 verb の使用法を説明した追加情報。

注:

1. TP サーバー verb は、同期エントリー・ポイント `APPC` ではなく、非同期エントリー・ポイント `APPC_Async` を使用して発行される必要があります。これらのエントリー・ポイントについての詳細は、29 ページの『第 2 章 トランザクション・プログラムの作成』を参照してください。
2. TP サーバー verb は、APPC の会話または状態に影響を及ぼしません。

TP サーバー verb で、APPC へ提供するパラメーターと APPC から戻されるパラメーターのほとんどは、16 進値です。コーディングを単純にするため、これらの値は、ヘッダー・ファイル **values_c.h** の中で定義されている、意味のある記号定数によって表されます。このヘッダー・ファイルは、TP サーバー・ヘッダー・ファイル **tpsrv_c.h** によって組み込まれます。たとえば、REGISTER_TP_SERVER verb の *opcode* パラメーターは、記号定数 `AP_REGISTER_TP_SERVER` によって表される 16 進値です。

指定されたパラメーターに値を設定する場合、または戻されたパラメーターの値をテストする場合には、16 進値ではなく記号定数を使用することが重要です。これは、さまざまなオペレーティング・システムがさまざまな方法でそれらの値をメモリー内に格納するため、示されている値が使用しているシステムで認識される形式になっていない場合もあるためです。

ここでは、TP サーバー verb を次の順序で説明します。

REGISTER_TP_SERVER
UNREGISTER_TP_SERVER
REGISTER_TP

UNREGISTER_TP
QUERY_ATTACH
ACCEPT_ATTACH
REJECT_ATTACH
ABORT_ATTACH

REGISTER_TP_SERVER

REGISTER_TP_SERVER verb は、トランザクション・プログラム (TP) をアプリケーションから自動的に開始できることを Communications Server に通知するために使用します。

VCB 構造体: REGISTER_TP_SERVER

REGISTER_TP_SERVER verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct register_tp_server
{
   AP_UINT16        opcode;
   unsigned char    rsrvd1;                  /* Reserved            */
   unsigned char    rsrvd2;                  /* Reserved            */
   AP_UINT16        primary_rc;
   AP_UINT32        secondary_rc;
   AP_UINT32        tps_id;
   unsigned char    tp_file_updates;
   AP_NOTIFY_CB     notify_cb;
} REGISTER_TP_SERVER;

  typedef void (*AP_NOTIFY_CB) (
                                unsigned char   reason,
                                unsigned char   attach_id[8],
                                AP_CORR         app_corr
                               );

 typedef union ap_corr {
                          void *          corr_p;
                          AP_UINT32       corr_l;
                          AP_INT32        corr_i;
                        } AP_CORR;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_REGISTER_TP_SERVER

tp_file_updates

sna_tps TP 構成ファイルが更新されたとき、それをアプリケーションに通知するかどうかを要求します。 次の値があります。

AP_YES アプリケーションは、**sna_tps** ファイルが変更されたことを通知するコールバックを要求します。

AP_NO アプリケーションは、**sna_tps** ファイルの変更について、通知を必要としません。

notify_cb

通知コールバック関数のアドレス。この関数は、次のいずれかが発生したことを TP サーバーに通知するため、REGISTER_TP verb で指定された *app_corr* パラメーターの値と共に APPC によって使用されます。

- 適切な Attach が使用可能である
- **sna_tps** TP 構成ファイルが変更された (アプリケーションが *tp_file_updates* パラメーターを AP_YES に設定することにより、この通知を要求した場合)

通知コールバック関数の使用方法についての詳細は、280 ページの『コールバック・ルーチン』を参照してください。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

tps_id この TP サーバーの固有 ID アプリケーションがそれ自体を TP サーバーとして登録した後、*tps_id* パラメーターの値はそのプロセスにのみ有効になります。*tps_id* パラメーターの値は、プロセスが変わると有効性を失います。 別のアプリケーションで、別の verb にこの *tps_id* パラメーターの値を使用すると、その verb は *primary_rc* 値が AP_PARAMETER_CHECK、*secondary_rc* 値が AP_BAD_TPS_ID となってリジェクトされます。

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_INVALID_CALLBACK
コールバック関数アドレスが有効ではありませんでした。

登録の失敗: アプリケーションを TP サーバーとして登録できなかった場合、APPC は次のパラメーターを戻します。

primary_rc
AP_REGISTER_FAIL

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は次の 1 次戻りコードを戻します。すべての verb に共通な戻りコードのリストについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

使用上の注意

この項では、コールバック・ルーチンの使用に関する追加情報を記載します。

コールバック・ルーチン

アプリケーションは、VCB のパラメーターの 1 つとしてコールバック・ルーチンへのポインターを提供します。この項では、Communications Server がそのルーチンをどのように使用するか、および、そのルーチンが実行する必要がある機能について説明します。

コールバック・ルーチンは、次のように定義されています。

```
typedef void (*AP_NOTIFY_CB) (
                               unsigned char   reason,
                               unsigned char   attach_id[8],
                               AP_CORR         app_corr
                              );
typedef union ap_corr {
                        void *          corr_p;
                        AP_UINT32       corr_l;
                        AP_INT32        corr_i;
                      } AP_CORR;
```

Communications Server は、次のパラメーターを使用してルーチンを呼び出します。

reason 通知のタイプ。次の値があります。

AP_ATTACH

この TP サーバーによって登録された TP 用に Attach が到着しました。 この場合は、*attach_id* パラメーターが通知コールバックに渡されます。これは、Attach の到着を使用して、次のことが 1 つ以上行われるためです。

- オプションとして、TP の自動開始に関する詳しい情報を照会する
- 必要であれば、Attach をリジェクトする
- どの TP を RECEIVE_ALLOCATE 処理用に自動開始するかを示す

AP_TP_FILE_CHANGE

sna_tps TP 構成ファイルに変更が加えられました。

attach_id

Attach 通知コールバックが戻した Attach の ID。

app_corr

アプリケーションが提供した相関係数の値。この値をアプリケーションで使用すると、戻された情報をアプリケーションの別の処理へ関連付けることができます。通知コールバックへ渡される相関係数の意味は、次のように、*reason* フラグの値が示す通知タイプによって異なります。

- *reason* が AP_ATTACH に設定されている場合、相関係数は、アプリケーションによって REGISTER_TP verb で指定された相関係数です。 そのため、アプリケーションは Attach を正しい登録済み TP に関連付けることができます。
- *reason* が AP_TP_FILE_CHANGE に設定されている場合、相関係数は REGISTER_TP_SERVER verb で指定された *tps_id* パラメーターの値です。

コールバック・ルーチンでは、これらのパラメーターをすべて使用する必要はありません。コールバック・ルーチンでは、戻された VCB に対して必要なすべての処理を実行できます。あるいは、単に verb が完了したことをメインプログラムに通知するために、変数を設定するだけでも構いません。

UNREGISTER_TP_SERVER

UNREGISTER_TP_SERVER verb は、アプリケーションで Attach 通知をそれ以上受信する必要がなくなった場合に使用します。

VCB 構造体: UNREGISTER_TP_SERVER

UNREGISTER_TP_SERVER verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct unregister_tp_server
{
   AP_UINT16        opcode;
   unsigned char    rsrvd1;                 /* Reserved            */
   unsigned char    rsrvd2;                 /* Reserved            */
   AP_UINT16        primary_rc;
   AP_UINT32        secondary_rc;
   AP_UINT32        tps_id;
} UNREGISTER_TP_SERVER;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_UNREGISTER_TP_SERVER

tps_id 前の REGISTER_TP_SERVER verb が戻した、登録解除する TP サーバーの ID。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_TPS_ID
tps_id パラメーターに指定した値が認識されませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は次の 1 次戻りコードを戻します。すべての verb に共通な戻りコードのリストについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

REGISTER_TP

REGISTER_TP verb は、TP サーバーに Attach を処理させる TP の名前をサービス管理プログラムに通知するために使用します。 また、すでに登録済みである TP について TP タイプまたは受信割り振りタイムアウトを変更するためにも使用します。

VCB 構造体: REGISTER_TP

REGISTER_TP verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct register_tp
{
   AP_UINT16        opcode;
   unsigned char    rsrvd1;                 /* Reserved            */
   unsigned char    rsrvd2;                 /* Reserved            */
   AP_UINT16        primary_rc;
   AP_UINT32        secondary_rc;
   AP_UINT32        tps_id;
   AP_UINT32        res_id;
   unsigned char    tp_name[64];
   char             lu_alias[8];
   unsigned char    fqplu_name[17];
   unsigned char    tp_type;
   AP_INT32         rcv_alloc_timeout;
   AP_UINT16        modify_existing;
   AP_CORR          app_corr;
} REGISTER_TP;

 typedef union ap_corr {
                         void *          corr_p;
                         AP_UINT32       corr_l;
                         AP_INT32        corr_i;
                       } AP_CORR;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_REGISTER_TP

tps_id 前の REGISTER_TP_SERVER verb が戻した TP サーバーの ID。

res_id REGISTER_TP を使用して既存の TP 登録を変更する (*modify_existing* パラメーターが AP_YES に設定されている) 場合、このパラメーターでは、元の REGISTER_TP verb で戻されたこのリソースの固有 ID を指定します。 それ以外の場合、このパラメーターは予約済みです。

tp_name

登録しようとしている TP の名前。この名前は EBCDIC で指定し、必要であれば、64 文字の長さになるよう EBCDIC のスペースを埋め込んでください。すべての Attach を処理する TP には、64 個の EBCDIC スペース (0x40) の値を指定してください。

lu_alias

ローカル LU 別名。この名前は ASCII で指定し、必要であれば、8 文字の長さになるよう ASCII のスペースを埋め込んでください。すべての Attach を処理する LU には、8 個の ASCII スペース (0x20) の値を指定してください。

fqplu_name

パートナー LU の完全修飾名。次のいずれかの EBCDIC ストリングを指定し、必要であれば、17 文字の長さになるよう EBCDIC のスペースを埋め込んでください。

- EBCDIC で表した完全修飾名。同じ完全修飾名を持つ Attach とだけ一致することを示します。
- すべての文字が EBCDIC のスペース (0x40)。任意のパートナー LU 名と一致することを示します。
- 部分名の後に EBCDIC の * (0x5C) を 1 つだけ続けたもの。これは、ワイルドカード LU 名を示します。

tp_type 登録しようとしている TP のタイプ。次の値があります。

AP_TP_TYPE_QUEUED

着呼 Attach は、新規の TP を開始したり、適切な TP を待つキューへ入る前に、実行中の TP のコピーへのキューに入ります。

AP_TP_TYPE_QUEUED_BROADCAST

TP の存在が Communications Server ドメインの周辺で広く知られている以外は AP_TP_TYPE_QUEUED と同じです。この TP の存在が広まると、多くのローカル LU に接続ルーティング・データを構成する必要がなくなり、単一マシンの同じ TP により、すべて処理されます。

AP_TP_TYPE_NON_QUEUED

Attach を受信するたびに、実行中のインスタンスに未解決の RECEIVE_ALLOCATE verb がある場合を除き、TP の新規インスタンスが開始されます。

rcv_alloc_timeout
TP の RECEIVE_ALLOCATE verb が自動開始 TP を待つ時間を秒単位で指定したもの。 次の値があります。

0 (zero)
待ちません。通常、この値を指定するのは、TP サーバーが着信 Attach への応答として TP を開始し、TP の RECEIVE_ALLOCATE 用に使用可能な Attach が常に存在する場合です。これに対する唯一の例外は、TP サーバーが TP を開始しようとしているときに Attach がタイムアウトになった場合です。

-1 無限に待ちます。

x (ただし、x は 0 より大きい)
x で示された秒数だけ待ちます。

modify_existing
この verb を使用して既存の登録を変更するか、または新規 TP を登録するかを指定します。次の値があります。

AP_YES この verb を使用して、既存の登録に対して *rcv_alloc_timeout* パラメーター、 *type* パラメーター、またはその両方のパラメーターを変更します。 次の制限があります。

- verb は、元の REGISTER_TP verb を発行したのと同じ TP サーバー・プログラムによって発行されなければなりません。
- *res_id* パラメーターは必ず指定する必要があり、元の REGISTER_TP verb で戻された値に一致しなければなりません。
- 元の REGISTER_TP verb の *rcv_alloc_timeout* パラメーター、*type* パラメーター、またはその両方のパラメーターを変更することはできますが、その他すべての指定パラメーターは、元の REGISTER_TP verb で使用されている値に一致しなければなりません。

AP_NO この verb を使用して新規 TP を登録します。

app_corr
Attach 通知コールバックに渡される、アプリケーションが提供する相関係数。詳細については、 280 ページの『使用上の注意』を参照してください。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

res_id このリソースの固有 ID

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_INVALID_TP_NAME
tp_name パラメーターに指定した値が有効ではありませんでした。

AP_INVALID_LU_ALIAS
lu_alias パラメーターに指定した値が有効ではありませんでした。

AP_INVALID_FQ_LU_NAME
fqplu_name パラメーターに指定した値が有効ではありませんでした。

AP_INVALID_TIMEOUT
rcv_alloc_timeout パラメーターに指定した値が有効ではありませんでした。

AP_BAD_TPS_ID
tps_id パラメーターに指定した値が認識されませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は次の 1 次戻りコードを戻します。すべての verb に共通な戻りコードのリストについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

UNREGISTER_TP

UNREGISTER_TP verb は、指定した TP 用の Attach 通知の受信をアプリケーションが必要としていないことをサービス管理プログラムに通知するために使用します。

VCB 構造体: UNREGISTER_TP

UNREGISTER_TP verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct unregister_tp
{
   AP_UINT16        opcode;
   unsigned char    rsrvd1;                 /* Reserved            */
   unsigned char    rsrvd2;                 /* Reserved            */
   AP_UINT16        primary_rc;
```

```
   AP_UINT32        secondary_rc;
   AP_UINT32        tps_id;
   AP_UINT32        res_id;
} UNREGISTER_TP;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_UNREGISTER_TP

tps_id 前の REGISTER_TP_SERVER verb が戻した TP サーバーの ID。

res_id 前の REGISTER_TP verb が戻した、このリソースの固有 ID。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_TPS_ID
tps_id パラメーターに指定した値が認識されませんでした。

AP_BAD_RES_ID
res_id パラメーターに指定した値が認識されませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は次の 1 次戻りコードを戻します。すべての verb に共通な戻りコードのリストについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_ATTACH

QUERY_ATTACH verb は、Communications Server からアプリケーションに通知された Attach に関する情報を取り出すために使用します。この verb はオプションです。Attach コールバックへ渡された TP サーバー相関係数によって表されるデータが、TP サーバーの使用に十分である場合、TP サーバーからこの verb を発行する必要はありません。

セキュリティー上の理由から、Attach 内のユーザー ID とパスワード情報は、事実上のユーザー ID が root である TP サーバーのみが使用できるようになります。root として実行されなかったアプリケーションでは、戻された Attach からアクセス・セキュリティー・サブフィールドが削除されています。

この verb は、必要に応じて TP サーバーから何回でも発行できます。ただし、PIP データは 1 回のみ抽出できます。 PIP データを取り出さずに Attach 情報を取り出すには、 *max_pip_len* を 0 (ゼロ) に設定して QUERY_ATTACH を発行します。

VCB 構造体: QUERY_ATTACH

QUERY_ATTACH verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct query_attach
{
   AP_UINT16         opcode;
   unsigned char     rsrvd1;                 /* Reserved            */
   unsigned char     rsrvd2;                 /* Reserved            */
   AP_UINT16         primary_rc;
   AP_UINT32         secondary_rc;
   AP_UINT32         tps_id;
   unsigned char     attach_id[8];
   unsigned char     tp_name[64];
   char              lu_alias[8];
   unsigned char     fq_plu_name[17]
   unsigned char     mode_name[8];
   AP_UINT16         max_pip_len;
   AP_UINT16         pip_dlen;
   unsigned char     *pip_dptr;
   AP_UINT16         max_fmh5_len;
   AP_UINT16         fmh5_dlen;
   unsigned char     *fmh5_dptr;
} QUERY_ATTACH;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_QUERY_ATTACH

tps_id 前の REGISTER_TP_SERVER verb が戻した TP サーバーの ID。

attach_id
Attach 通知コールバックが戻した Attach の ID。

max_pip_len
PIP データに使用可能な最大バッファー・スペース。

pip_dptr
戻される Attach PIP データ・バッファー用にコール元が割り振ったバッファーへのポインター。

max_fmh5_len
FM ヘッダー 5 (FMH5) データに使用可能な最大バッファー・スペース。

fmh5_dptr
戻される Attach FMH5 データ・バッファー用にコール元が割り振ったバッファーへのポインター。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

tp_name
Attach TP 名

lu_alias
Attach ローカル LU 別名

fq_plu_name
Attach 完全修飾パートナー LU 名

mode_name
Attach モード名

pip_dlen
戻された PIP データの実際のバイト数

fmh5_dlen
戻された FMH5 データの実際のバイト数

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_ATTACH_ID
attach_id パラメーターに指定した値が認識されませんでした。

AP_BAD_TPS_ID
tps_id パラメーターに指定した値が認識されませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は次の 1 次戻りコードを戻します。すべての verb に共通な戻りコードのリストについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR

ACCEPT_ATTACH

ACCEPT_ATTACH verb は、この TP サーバーによる Attach の処理を続行するために使用します。

VCB 構造体: ACCEPT_ATTACH

ACCEPT_ATTACH verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct accept_attach
{
   AP_UINT16        opcode;
   unsigned char    rsrvd1;                 /* Reserved            */
   unsigned char    rsrvd2;                 /* Reserved            */
   AP_UINT16        primary_rc;
   AP_UINT32        secondary_rc;
   AP_UINT32        tps_id;
   unsigned char    attach_id[8];
} ACCEPT_ATTACH;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_ACCEPT_ATTACH

tps_id 前の REGISTER_TP_SERVER verb が戻した TP サーバーの ID。

attach_id

Attach 通知コールバックが戻した Attach の ID。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc

AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_ATTACH_ID
attach_id パラメーターに指定した値が認識されませんでした。

AP_BAD_TPS_ID
tps_id パラメーターに指定した値が認識されませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は次の 1 次戻りコードを戻します。すべての verb に共通な戻りコードのリストについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

REJECT_ATTACH

REJECT_ATTACH verb は、この TP サーバーによる Attach の処理を終了するために使用します。

VCB 構造体: REJECT_ATTACH

REJECT_ATTACH verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct reject_attach
{
   AP_UINT16        opcode;
   unsigned char    rsrvd1;                 /* Reserved            */
   unsigned char    rsrvd2;                 /* Reserved            */
   AP_UINT16        primary_rc;
   AP_UINT32        secondary_rc;
   AP_UINT32        tps_id;
   unsigned char    attach_id[8];
   AP_UINT32        reason;
} REJECT_ATTACH;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_REJECT_ATTACH

tps_id 前の REGISTER_TP_SERVER verb が戻した TP サーバーの ID。

attach_id
Attach 通知コールバックが戻した Attach の ID。

reason 自動開始をリジェクトする理由。この値は、 291 ページの『SNA センス・コード』に示すような SNA センス・コードです。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_ATTACH_ID
attach_id パラメーターに指定した値が認識されませんでした。

AP_BAD_TPS_ID
tps_id パラメーターに指定した値が認識されませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は次の 1 次戻りコードを戻します。すべての verb に共通な戻りコードのリストについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

SNA センス・コード: 表 10 に、Attach をリジェクトするために使用する共通の SNA センス・コードを示します。

表 10. 共通の SNA センス・コード

記号	値	意味
AP_SECURITY_INVALID	080F6051	セキュリティーが有効でない
AP_SEC_BAD_PASSWORD_EXPIRED	080FFF00	パスワードの期限が切れている
AP_SEC_BAD_PASSWORD_INVALID	080FFF01	パスワードが有効でない
AP_SEC_BAD_USERID_REVOKED	080FFF02	ユーザー ID が取り消された

表 10. 共通の SNA センス・コード (続き)

記号	値	意味
AP_SEC_BAD_USERID_INVALID	080FFF03	ユーザー ID が有効でない
AP_SEC_BAD_USERID_MISSING	080FFF04	ユーザー ID が欠落している
AP_SEC_BAD_PASSWORD_MISSING	080FFF05	パスワードが欠落している
AP_SEC_BAD_GROUP_INVALID	080FFF06	グループが有効でない
AP_SEC_BAD_UID_REVOKED_IN_GRP	080FFF07	指定したグループ内でユーザー ID が取り消された
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP	080FFF08	指定したグループ内でユーザー ID が定義されていない
AP_SEC_BAD_UNAUTHRZD_AT_RLU	080FFF09	ユーザー ID がリモート LU を使用するよう定義されていない
AP_SEC_BAD_UNAUTHRZD_FROM_LLU	080FFF0A	ユーザー ID がローカル LU からリモート LU を使用するよう定義されていない
AP_SEC_BAD_UNAUTHRZD_TO_TP	080FFF0B	ユーザー ID がリモート LU で TP を使用するよう定義されていない
AP_SEC_BAD_INSTALL_EXIT_FAILED	080FFF0C	リモート LU でのインストール・システム出口処理が失敗した
AP_SEC_BAD_PROCESSING_FAILURE	080FFF0D	ローカル LU とリモート LU の間で処理が失敗したが、この条件は一時的である
AP_SEC_BAD_PROTOCOL_VIOLATION	080F6058	プロトコル違反の結果としてセキュリティーの妥当性検査が失敗した
AP_TRANS_PGM_NOT_AVAIL_RETRY	084B6031	TP が使用可能でない - 再試行可能
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY	084C0000	TP が使用可能でない - 再試行不能
AP_PIP_INVALID	1008201D	PIP データが有効でない
AP_ATTACH_LEN_INVALID	10086000	Attach の長さが有効でない

表 10. 共通の SNA センス・コード (続き)

記号	値	意味
AP_SECURITY_LEN_INVALID	10086005	セキュリティーの長さが有効でない
AP_PARM_LEN_INVALID	10086009	パラメーターの長さが有効でない
AP_LUWID_LEN_INVALID	10086011	LUWID の長さが有効でない
AP_TP_NAME_NOT_RECOGNIZED	10086021	TP 名が認識されなかった
AP_PIP_NOT_ALLOWED	10086031	PIP データは許容されない
AP_PIP_FIELDS_REQUIRED	10086032	PIP データが必須である
AP_CONVERSATION_TYPE_MISMATCH	10086034	会話タイプが不一致である
AP_LU_CAPABILITY_CONFLICT	10086040	Attach LU 機能がバインドと競合する
AP_SYNC_LEVEL_NOT_SUPPORTED	10086041	同期レベルが TP でサポートされていない

注: リモート LU が拡張セキュリティー情報を必要としていない場合、Communications Server は、080FFF00 ～ 080FFFFF の範囲のセンス・コードの代わりに、汎用 AP_SECURITY_INVALID センス・コード (080F651) を使用することがあります。

ABORT_ATTACH

ABORT_ATTACH verb は、ACCEPT_ATTACH verb を使用して Attach を受け入れた後、TP サーバーまたは TP がその後の処理中にエラーを検出したために、この TP サーバーによる Attach の処理を終了するときに使用します。たとえば、TP サーバーがその TP へ fork できなかった場合などです。 ABORT_ATTACH verb は、TP サーバーと TP プロセスのどちらからも発行できます。

VCB 構造体: ABORT_ATTACH

ABORT_ATTACH verb の VCB 構造体の定義は、次のとおりです。

```
typedef struct abort_attach
{
   AP_UINT16        opcode;
   unsigned char    rsrvd1;                 /* Reserved            */
   unsigned char    rsrvd2;                 /* Reserved            */
   AP_UINT16        primary_rc;
   AP_UINT32        secondary_rc;
   AP_UINT32        tps_id;
   unsigned char    attach_id[8];
   AP_UINT32        reason;
} ABORT_ATTACH;
```

指定パラメーター

TP は、次のパラメーターを APPC に指定します。

opcode AP_ABORT_ATTACH

tps_id 前の REGISTER_TP_SERVER verb が戻した TP サーバーの ID。

attach_id
Attach 通知コールバックが戻した、打ち切る Attach の ID。

reason 自動開始を打ち切る理由。この値は、 291 ページの『SNA センス・コード』に示すような SNA センス・コードです。

戻りパラメーター

この verb が実行された後、APPC は正常に実行されたかどうかを示すため、また、正常に実行されなかった場合は実行が失敗した原因を示すため、パラメーターを戻します。

正常に実行された場合

この verb が正常に実行された場合、APPC は次のパラメーターを戻します。

primary_rc
AP_OK

正常に実行されなかった場合

この verb が正常に実行されなかった場合、APPC はエラーのタイプを示す 1 次戻りコード・パラメーターと、実行が失敗した理由の詳細を示す 2 次戻りコード・パラメーターを戻します。

パラメーター検査: パラメーター・エラーのために verb が実行されない場合には、APPC は次のパラメーターを戻します。

primary_rc
AP_PARAMETER_CHECK

secondary_rc
次の値があります。

AP_BAD_ATTACH_ID
attach_id パラメーターに指定した値が認識されませんでした。

AP_BAD_TPS_ID
tps_id パラメーターに指定した値が認識されませんでした。

その他の条件: その他の条件が存在したためにこの verb が実行されなかった場合、 APPC は次の 1 次戻りコードを戻します。すべての verb に共通な戻りコードのリストについては、 307 ページの『付録 B. 共通戻りコード』を参照してください。

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

第 6 章 トランザクション・プログラムの例

これらの Communications Server APPC トランザクション・プログラム (TP) の例は、マップ式会話での APPC verb の使用法を示しています。

プログラムは、Communications Server では **asample1.c** および **asample2.c** として、ディレクトリー **/usr/lib/sna/samples** (AIX) または **/opt/ibm/sna/samples** (Linux) にあります。

この章には、次の情報を記載します。

- サンプル TP の処理の概要
- 各 TP の疑似コード
- 各 TP のコンパイル、リンク、実行の手順

処理の概要

この章で示す TP を使用すると、別のシステム上にあるファイルをブラウズできます。 ユーザーは、一度に 1 つのデータ・ブロックを 16 進数と文字の形式で表示できます。1 つのブロックを表示した後、ユーザーは次のブロックを要求するか、前のブロックを要求するか、または終了できます。

asample1 (呼び出し元 TP) は、**asample2** (呼び出し先 TP) へファイル名を送信します。 **asample2** は、そのファイルを見つけた場合、**asample1** へ最初のデータ・ブロックを戻します。ファイルが見つからなかった場合は、会話の割り振りを解除し、終了します。

asample1 は、ブロックを受信した場合、そのブロックを画面に表示し、次ブロックを表す **F**、前ブロックを表す **B**、終了を表す **Q** のいずれかが入力されるのを待ちます。 ユーザーが次ブロックか前ブロックを選択した場合、**asample1** はその要求を **asample2** へ送信し、**asample2** は該当するブロックを送信します。 この処理は、ユーザーが終了のオプションを選択するまで続行され、終了のオプションが選択された時点で、**asample1** は会話の割り振りを解除し、両方のプログラムが終了します。

ユーザーが次のブロックを要求し、**asample2** がすでに最後のブロックを送信し終わっている場合、**asample2** はファイルの先頭へ折り返します。 同様に、ユーザーが前のブロックを要求し、先頭ブロックが表示されている場、**asample2** は折り返し、最後のブロックを送信します。

どちらのプログラムも、エラーからの回復を試みません。APPC から正しくない戻りコードが戻った場合、プログラムはそれを説明するメッセージを表示して終了します。

疑似コード

この項には、**asample1** と **asample2** の各 TP の疑似コードが記載されています。

asample1 (呼び出し元 TP)

asample1 (呼び出し元 TP) の疑似コードは、次のとおりです。

```
TP_started
mc_allocate (sync_level none)
mc_send_data (data = filename), send type prepare_to_receive_flush
do while no error and prompt not Q
    mc_receive_and_wait
    if data block received
       display data block
    else if permission to send received
        get user prompt (F, B, or Q)
        if prompt = F or B     /* Not Q */
           mc_send_data (data = prompt), send type p_to_r_flush
        endif
    endif
 end do
 mc_deallocate
 TP_ended
```

asample2 (呼び出し先 TP)

asample2 (呼び出し先 TP) の疑似コードは、次のとおりです。

```
receive_allocate
do while conversing
   mc_receive_and_wait (return status with data)
   if data received and send indication received
      if first time (data = filename)
         open file
         if file not found
             mc_deallocate
             set conversing false
         endif
       else (data = prompt)
           read and store prompt
       endif
       if (conversing)
           read file block
           mc_send_data (file block)
       endif
    else if deallocate received
       set conversing false
    endif
    end while conversing
   close file
   tp_ended
```

TP のテスト

2 つのプログラムのソース・コードを調べてから、それらのプログラムをテストすることもできます。

通常の場合、APPC はローカル・コンピューターとリモート・コンピューターの間の通信に使用しますが、テストのため、両方の TP を同じ Communications Server コンピューター上で実行すると便利です。

これらの TP をコンパイルおよびリンクするには、次の手順を実行してください。

1. **asample1.c** と **asample2.c** の 2 つのファイルを、 **/usr/lib/sna/samples** (AIX) または **/opt/ibm/sna/samples** (Linux) からプライベート・ディレクトリーにコピーします。

2. AIX でプログラムをコンパイルしてリンクするには、次のコマンドを使用します。

   cc -o asample1 -I /usr/include/sna -bimport:/usr/lib/sna/appc_r.exp -bimport:/usr/lib/sna/csv_r.exp asample1.c

   cc -o asample2 -I /usr/include/sna -bimport:/usr/lib/sna/appc_r.exp -bimport:/usr/lib/sna/csv_r.exp asample2.c

   Linux でプログラムをコンパイルしてリンクするには、次のコマンドを使用します。

   gcc -o asample1 -I /opt/ibm/sna/include -L /opt/ibm/sna/lib -lappc -lsna_r -lcsv -lpLiS -lpthread asample1.c

   gcc -o asample2 -I /opt/ibm/sna/include -L /opt/ibm/sna/lib -lappc -lsna_r -lcsv -lpLiS -lpthread asample2.c

TP を実行するには、次の手順を実行します。これらの手順の一部では、Communications Server 構成を更新します。通常、この更新はシステム管理者が行います。

これらの TP は、同じコンピューター上で実行しても、別々のコンピューター上で実行しても構いません。次のステップでは、「ソース・コンピューター」は呼び出し側 TP「**asample1**」を実行するコンピューターを表し、「ターゲット・コンピューター」は呼び出し先 TP「**asample2**」を実行するコンピューターを表します。

1. 別々のコンピューターで TP を実行する場合、ソース・コンピューターとターゲット・コンピューターの間の CP-CP セッションをサポートするように、通信リンクを構成します。詳細については、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。
2. モードを構成します。モード名として LOCMODE を指定してください。その他のパラメーターは、デフォルト値のままにしておきます。
3. **asample1** (呼び出し元 TP) 用に、ソース・コンピューター上に論理装置 (LU) を構成します。 LU 名と LU 名の別名の両方を **TPLU1** (**asample1** プログラムで指定された LU 別名) に設定します。 その他のパラメーターは、デフォルト値のままにしておきます。
4. 別々のコンピューターで TP を実行する場合、ソース・コンピューター上のパートナー LU 別名を構成して、ターゲット LU を識別します。パートナー LU 名を *netname*.**TPLU2** に設定します。ここで、*netname* はターゲット・コンピューターの SNA ネットワーク名です。
5. 呼び出し先 TP 用のターゲット・コンピューター上に LU を構成します。 LU 名と LU 別名の両方を **TPLU2** (**asample2** にサービスする LU を **asample1** プログラムから参照する場合に使用する別名) に設定します。 その他のパラメーターは、デフォルト値のままにしておきます。
6. 呼び出し先 TP を、ターゲット・コンピューター上にある Communications Server 呼び出し可能 TP データ・ファイルの中に構成します。詳細については、「*IBM Communications Server for Linux 管理ガイド*」または「*IBM Communications Server for AIX 管理ガイド*」を参照してください。

- *TP name* パラメーターに **TPNAME2** (呼び出し元 TP によって指定された名前) を指定します。
- *Full path to TP executable* に、実行可能ファイル **asample2** の絶対パス名を入力します。
- *User ID* パラメーターに、ターゲット・コンピューターでの自分の AIX / Linux ユーザー ID を指定します。
- その他のパラメーターは、デフォルト値のままにしておきます。

7. 呼び出し先 TP が `root` の *user_id* で実行する場合は、それが可能になるように、実行可能ファイルに対する許可を変更します。 次のコマンドを使用します。

   chmod +s asample2

8. 呼び出し元プログラム **asample1** を開始します。このプログラムには、1 つのパラメーター、つまり表示するファイルの (ターゲット・コンピューターの) 絶対パス名が必要です。次に例を示します。

   asample1 /usr/john/myfile.text

9. **F** または **B** を入力して、要求したファイルの各ブロックを表示します。
10. **Q** を使用して、プログラム 1 を終了してください。プログラム 2 も終了します。

付録 A. 戻りコードの値

この付録は、APPC インターフェースで発生し得るすべての戻りコードを数値順にリストしています。値は、ヘッダー・ファイル **values_c.h** (AIX / Linux の場合) または **winappc.h** (Windows の場合) に定義されています。

この付録をリファレンスとして使用して、アプリケーションが受け取った戻りコードの意味を確認できます。

1 次戻りコード

次の 1 次戻りコードは、APPC アプリケーションで使用されます。

AP_OK	0x0000
AP_PARAMETER_CHECK	0x0100
AP_STATE_CHECK	0x0200
AP_INDICATION	0x0210
AP_TP_BUSY	0x02F0
AP_ALLOCATION_ERROR	0x0300
AP_ACTIVATION_FAIL_RETRY	0x0310
AP_COMM_SUBSYSTEM_ABENDED	0x03F0
AP_ACTIVATION_FAIL_NO_RETRY	0x0410
AP_COMM_SUBSYSTEM_NOT_LOADED	0x04F0
AP_DEALLOC_ABEND	0x0500
AP_LU_SESS_LIMIT_EXCEEDED	0x0510
AP_DEALLOC_ABEND_PROG	0x0600
AP_FUNCTION_NOT_SUPPORTED	0x0610
AP_THREAD_BLOCKING	0x06F0
AP_DEALLOC_ABEND_SVC	0x0700
AP_DEALLOC_ABEND_TIMER	0x0800
AP_DATA_POSTING_BLOCKED	0x0810
AP_INVALID_VERB_SEGMENT	0x08F0
AP_DEALLOC_NORMAL	0x0900
AP_PATH_SWITCH_NOT_ALLOWED	0x0910
AP_CP_CP_SESS_ACT_FAILURE	0x0A10
AP_PROG_ERROR_NO_TRUNC	0x0C00
AP_PROG_ERROR_TRUNC	0x0D00
AP_PROG_ERROR_PURGING	0x0E00
AP_CONV_FAILURE_RETRY	0x0F00
AP_CONV_FAILURE_NO_RETRY	0x1000
AP_SVC_ERROR_NO_TRUNC	0x1100
AP_UNEXPECTED_DOS_ERROR	0x11F0
AP_SVC_ERROR_TRUNC	0x1200
AP_SVC_ERROR_PURGING	0x1300
AP_UNSUCCESSFUL	0x1400
AP_STACK_TOO_SMALL	0x15F0
AP_MIXED_API_USED	0x16F0
AP_IN_PROGRESS	0x17F0
AP_CNOS_PARTNER_LU_REJECT	0x1800
AP_COMPLETED	0x18F0
AP_CONVERSATION_TYPE_MIXED	0x1900
AP_NODE_STOPPING	0x1A00
AP_NODE_NOT_STARTED	0x1B00
AP_CANCELLED	0x2100
AP_BACKED_OUT	0x2200
AP_DUPLEX_TYPE_MIXED	0x2300
AP_LS_FAILURE	0x2300
AP_OPERATION_INCOMPLETE	0x4000
AP_OPERATION_NOT_ACCEPTED	0x4100

AP_CONVERSATION_ENDED	0x4200
AP_ERROR_INDICATION	0x4300
AP_EXPD_NOT_SUPPORTED_BY_LU	0x4400
AP_BUFFER_TOO_SMALL	0x4500
AP_MEMORY_ALLOCATION_FAILURE	0x4600
AP_INVALID_VERB	0xFFFF

2 次戻りコード

次の 2 次戻りコードは、APPC アプリケーションで使用されます。

AP_AS_SPECIFIED	0x00000000
AP_ALLOCATION_ERROR_PENDING	0x00000300
AP_DEALLOC_ABEND_PROG_PENDING	0x00000600
AP_DEALLOC_ABEND_SVC_PENDING	0x00000700
AP_DEALLOC_ABEND_TIMER_PENDING	0x00000800
AP_UNKNOWN_ERROR_TYPE_PENDING	0x00001100
AP_BO_NO_RESYNC	0x00002408
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY	0x00004C08
AP_INVALID_SET_PROT	0x00070000
AP_INVALID_DLUS_NAME	0x00900000
AP_SEC_BAD_PASSWORD_EXPIRED	0x00FF0F08
AP_BAD_TP_ID	0x01000000
AP_BO_RESYNC	0x01002408
AP_INVALID_NEW_PROT	0x01070000
AP_DLC_ACTIVE	0x01100000
AP_NO_DEFAULT_DLUS_DEFINED	0x01900000
AP_BAD_TPSID	0x01FF0000
AP_SEC_BAD_PASSWORD_INVALID	0x01FF0F08
AP_BAD_CONV_ID	0x02000000
AP_SEND_ERROR_LOG_LL_WRONG	0x02010000
AP_INVALID_SET_UNPROT	0x02070000
AP_INVALID_NUMBER_OF_NODE_ROWS	0x02080000
AP_DUPLICATE_CP_NAME	0x02100000
AP_INVALID_PU_ID	0x02900000
AP_NOT_OWNER	0x02FF0000
AP_SEC_BAD_USERID_REVOKED	0x02FF0F08
AP_BAD_LU_ALIAS	0x03000000
AP_BAD_DLOAD_ID	0x03000001
AP_BAD_REMOTE_LU_ALIAS	0x03000002
AP_SEND_ERROR_BAD_TYPE	0x03010000
AP_INVALID_NEW_UNPROT	0x03070000
AP_DUPLICATE_DEST_ADDR	0x03100000
AP_PU_ALREADY_ACTIVATING	0x03900000
AP_INSUFFICIENT_PRIVILEGES	0x03FF0000
AP_SEC_BAD_USERID_INVALID	0x03FF0F08
AP_ALLOCATION_FAILURE_NO_RETRY	0x04000000
AP_SEND_ERROR_BAD_STATE	0x04010000
AP_INVALID_SET_USER	0x04070000
AP_NODE_ROW_WGT_LESS_THAN_LAST	0x04080000
AP_CANT_MODIFY_PORT_NAME	0x04100000
AP_PU_ALREADY_DEACTIVATING	0x04900000
AP_INVALID_CALLBACK	0x04FF0000
AP_SEC_BAD_USERID_MISSING	0x04FF0F08
AP_ALLOCATION_FAILURE_RETRY	0x05000000
AP_BAD_ERROR_DIRECTION	0x05010000
AP_INVALID_DATA_TYPE	0x05070000
AP_TG_ROW_WGT_LESS_THAN_LAST	0x05080000
AP_DUPLICATE_PORT_NUMBER	0x05100000
AP_PU_ALREADY_ACTIVE	0x05900000
AP_BAD_TP_TYPE	0x05FF0000
AP_SEC_BAD_PASSWORD_MISSING	0x05FF0F08
AP_INVALID_STATS_TYPE	0x06070000
AP_DUPLICATE_PORT_NAME	0x06100000
AP_PU_NOT_ACTIVE	0x06900000
AP_ALREADY_REGISTERED	0x06FF0000

AP_SEC_BAD_GROUP_INVALID	0x06FF0F08
AP_AS_NEGOTIATED	0x07000000
AP_INVALID_TABLE_TYPE	0x07070000
AP_INVALID_DLC_NAME	0x07100000
AP_DLUS_REJECTED	0x07900000
AP_SEC_BAD_UID_REVOKED_IN_GRP	0x07FF0F08
AP_PORT_DEACTIVATED	0x08070000
AP_INVALID_DLC_TYPE	0x08100000
AP_DLUS_CAPS_MISMATCH	0x08900000
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP	0x08FF0F08
AP_ALLOCATE_NOT_PENDING	0x09050000
AP_INVALID_SET_PASSWORD	0x09070000
AP_INVALID_NUMBER_OF_TG_ROWS	0x09080000
AP_INVALID_LINK_ACTIVE_LIMIT	0x09100000
AP_PU_FAILED_ACTPU	0x09900000
AP_SEC_BAD_UNAUTHRZD_AT_RLU	0x09FF0F08
AP_SNA_DEFD_COS_CANT_BE_CHANGE	0x0A080000
AP_SNA_DEFD_COS_CANT_BE_CHANGED	0x0A080000
AP_PU_NOT_RESET	0x0A900000
AP_SEC_BAD_UNAUTHRZD_FROM_LLU	0x0AFF0F08
AP_INVALID_NUM_PORTS_SPECIFIED	0x0B100000
AP_PU_OWNS_LUS	0x0B900000
AP_SEC_BAD_UNAUTHRZD_TO_TP	0x0BFF0F08
AP_INVALID_PORT_NAME	0x0C100000
AP_INVALID_FILTER_OPTION	0x0C900000
AP_SEC_BAD_INSTALL_EXIT_FAILED	0x0CFF0F08
AP_INVALID_PORT_TYPE	0x0D100000
AP_INVALID_STOP_TYPE	0x0D900000
AP_SEC_BAD_PROCESSING_FAILURE	0x0DFF0F08
AP_UNRECOGNIZED_DEACT_TYPE	0x0E050000
AP_PORT_ACTIVE	0x0E100000
AP_PU_ALREADY_DEFINED	0x0E900000
AP_NO_PORTS_DEFINED_ON_DLC	0x0F100000
AP_DEPENDENT_LU_NOT_SUPPORTED	0x0F900000
AP_INVALID_DLC	0x10050000
AP_COS_NAME_NOT_DEFD	0x10080000
AP_DUPLICATE_PORT	0x10100000
AP_INVALID_DSPU_SERVICES	0x10900000
AP_BAD_CONV_TYPE	0x11000000
AP_SNA_DEFD_COS_CANT_BE_DELETE	0x11080000
AP_SNA_DEFD_COS_CANT_BE_DELETED	0x11080000
AP_STOP_PORT_PENDING	0x11100000
AP_DSPU_SERVICES_NOT_SUPPORTED	0x11900000
AP_BAD_SYNC_LEVEL	0x12000000
AP_LU_NAU_ADDR_ALREADY_DEFD	0x12020000
AP_INVALID_SESSION_ID	0x12050000
AP_LINK_DEACT_IN_PROGRESS	0x12100000
AP_INVALID_DSPU_NAME	0x12900000
AP_BAD_SECURITY	0x13000000
AP_INVALID_NN_SESSION_TYPE	0x13050000
AP_LINK_DEACTIVATED	0x13100000
AP_PARTNER_NOT_FOUND	0x13200000
AP_PARTNER_NOT_RESPONDING	0x13300000
AP_ERROR	0x13400000
AP_DSPU_ALREADY_DEFINED	0x13900000
AP_BAD_RETURN_CONTROL	0x14000000
AP_INVALID_MAX_NEGOT_SESS_LIM	0x14020000
AP_INVALID_SET_COLLECT_STATS	0x14050000
AP_LINK_ACT_BY_REMOTE	0x14100000
AP_INVALID_SOLICIT_SSCP_SESS	0x14900000
AP_INVALID_BACK_LEVEL_SUPPORT	0x15000000
AP_INVALID_MODE_NAME	0x15020000
AP_INVALID_SET_COLLECT_NAMES	0x15050000
AP_LINK_ACT_BY_LOCAL	0x15100000
AP_INVALID_TG_NUMBER	0x15500000
AP_MISSING_CP_NAME	0x15510000
AP_MISSING_CP_TYPE	0x15520000

AP_INVALID_CP_TYPE	0x15520000
AP_DUPLICATE_TG_NUMBER	0x15530000
AP_TG_NUMBER_IN_USE	0x15540000
AP_MISSING_TG_NUMBER	0x15550000
AP_PARALLEL_TGS_NOT_ALLOWED	0x15570000
AP_INVALID_BKUP_DLUS_NAME	0x15900000
AP_PIP_LEN_INCORRECT	0x16000000
AP_INVALID_RECV_PACING_WINDOW	0x16020000
AP_INVALID_SET_COLLECT_RSCVS	0x16050000
AP_SEC_REQUESTED_NOT_SUPPORTED	0x16900000
AP_NO_USE_OF_SNASVCMG	0x17000000
AP_INVALID_CNOS_SLIM	0x17020000
AP_LINK_NOT_DEFD	0x17100000
AP_INVALID_DUPLEX_SUPPORT	0x17900000
AP_UNKNOWN_PARTNER_MODE	0x18000000
AP_INVALID_TARGET_PACING_CNT	0x18020000
AP_PS_CREATION_FAILURE	0x18100000
AP_QUEUE_PROHIBITED	0x18900000
AP_INVALID_MAX_RU_SIZE_UPPER	0x19020000
AP_TP_ACTIVE	0x19100000
AP_INVALID_TEMPLATE_NAME	0x19900000
AP_INVALID_SNASVCMG_MODE_LIMIT	0x1A020000
AP_MODE_ACTIVE	0x1A100000
AP_CLASHING_NAU_RANGE	0x1A900000
AP_PLU_ACTIVE	0x1B100000
AP_INVALID_NAU_RANGE	0x1B900000
AP_INVALID_COS_SNASVCMG_MODE	0x1C020000
AP_INVALID_PLU_NAME	0x1C100000
AP_INVALID_NUM_DSLU_TEMPLATES	0x1C900000
AP_INVALID_DEFAULT_RU_SIZE	0x1D020000
AP_INVALID_SET_NEGOTIABLE	0x1D100000
AP_GLOBAL_TIMEOUT_NOT_DEFINED	0x1D900000
AP_INVALID_MIN_CONWINNERS	0x1E020000
AP_INVALID_MODE_NAME_SELECT	0x1E100000
AP_INVALID_RESOURCE_NAME	0x1E900000
AP_INVALID_RESPONSIBLE	0x1F100000
AP_INVALID_DLUS_RETRY_TIMEOUT	0x1F900000
AP_MODE_SESS_LIM_EXCEEDS_NEG	0x20020000
AP_INVALID_DRAIN_SOURCE	0x20100000
AP_INVALID_DLUS_RETRY_LIMIT	0x20900000
AP_CPSVCMG_ALREADY_DEFD	0x21020000
AP_INVALID_CN_NAME	0x21080000
AP_INVALID_DRAIN_TARGET	0x21100000
AP_TP_NAME_NOT_RECOGNIZED	0x21600810
AP_INVALID_MIN_CONLOSERS	0x21900000
AP_BAD_DUPLEX_TYPE	0x22000000
AP_INVALID_BYPASS_SECURITY	0x22020000
AP_DEF_LINK_INVALID_SECURITY	0x22080000
AP_INVALID_FORCE	0x22100000
AP_SYSTEM_TP_CANT_BE_CHANGED	0x22600810
AP_INVALID_MAX_RU_SIZE_LOW	0x22900000
AP_FDX_NOT_SUPPORTED_BY_LU	0x23000000
AP_TEST_INVALID_FOR_FDX	0x23010000
AP_INVALID_IMPLICIT_PLU_FORBID	0x23020000
AP_INVALID_PROPAGATION_DELAY	0x23080000
AP_SYSTEM_TP_CANT_BE_DELETED	0x23600810
AP_INVALID_MAX_RECV_PACING_WIN	0x23900000
AP_SEND_EXPD_INVALID_LENGTH	0x24010000
AP_INVALID_SPECIFIC_SECURITY	0x24020000
AP_INVALID_EFFECTIVE_CAPACITY	0x24080000
AP_INVALID_CLEANUP_TYPE	0x24100000
AP_INVALID_DYNAMIC_LOAD	0x24600810
AP_RU_SIZE_LOW_UPPER_MISMATCH	0x24900000
AP_RCV_EXPD_INVALID_LENGTH	0x25010000
AP_INVALID_DELAYED_LOGON	0x25020000
AP_INVALID_COS_NAME	0x25100000
AP_INVALID_ENABLED	0x25600810

AP_LU_ALREADY_ACTIVATING	0x25900000
AP_EXPD_BAD_RETURN_CONTROL	0x26010000
AP_INVALID_CNOS_PERMITTED	0x26020000
AP_PW_SUB_NOT_SUPP_ON_SESS	0x26050000
AP_INVALID_SESSION_LIMIT	0x26100000
AP_INVALID_PIP_ALLOWED	0x26600810
AP_LU_DEACTIVATING	0x26900000
AP_EXPD_DATA_BAD_CONV_STATE	0x27010000
AP_INVALID_DRAIN	0x27100000
AP_LU_ALREADY_ACTIVE	0x27900000
AP_INVALID_PRLL_SESS_SUPP	0x28100000
AP_INVALID_MIN_CONTENTION_SUM	0x28900000
AP_INVALID_LU_NAME	0x29100000
AP_COMPRESSION_NOT_SUPPORTED	0x29900000
AP_MODE_NOT_RESET	0x2A100000
AP_INVALID_MAX_COMPRESS_LVL	0x2A900000
AP_MODE_RESET	0x2B100000
AP_INVALID_COMPRESSION	0x2B900000
AP_CNOS_REJECT	0x2C100000
AP_INVALID_EXCEPTION_INDEX	0x2C900000
AP_INVALID_OP_CODE	0x2D100000
AP_INVALID_MAX_LS_EXCEPTION	0x2D900000
AP_INVALID_DISABLE	0x2E900000
AP_INVALID_MODIFY_TEMPLATE	0x2F900000
AP_INVALID_ALLOW_TIMEOUT	0x30900000
AP_CONFIRM_ON_SYNC_LEVEL_NONE	0x31000000
AP_PIP_NOT_ALLOWED	0x31600810
AP_TRANS_PGM_NOT_AVAIL_RETRY	0x31604B08
AP_POST_ON_RECEIPT_BAD_FILL	0x31900000
AP_CONFIRM_BAD_STATE	0x32000000
AP_UNKNOWN_USER	0x32100000
AP_POST_ON_RECEIPT_BAD_STATE	0x32900000
AP_CONFIRM_NOT_LL_BDY	0x33000000
AP_NO_PROFILES	0x33100000
AP_INVALID_HPR_SUPPORT	0x33900000
AP_CONFIRM_INVALID_FOR_FDX	0x34000000
AP_CONVERSATION_TYPE_MISMATCH	0x34600810
AP_INVALID_LU_MODEL	0x34900000
AP_INVALID_MODEL_NAME	0x35900000
AP_TOO_MANY_PROFILES	0x36100000
AP_INVALID_CRYPTOGRAPHY	0x36900000
AP_INVALID_UPDATE_TYPE	0x37100000
AP_INVALID_CLU_CRYPTOGRAPHY	0x37900000
AP_DIR_ENTRY_PARENT	0x38100000
AP_INVALID_RESOURCE_TYPES	0x38900000
AP_NODE_ALREADY_STARTED	0x39100000
AP_CHECKSUM_FAILED	0x39900000
AP_NODE_FAILED_TO_START	0x3A100000
AP_DATA_CORRUPT	0x3A900000
AP_LU_ALREADY_DEFINED	0x3B100000
AP_INVALID_RETRY_FLAGS	0x3B900000
AP_IMPLICIT_LU_DEFINED	0x3C100000
AP_DELAYED_VERB_PENDING	0x3C900000
AP_PORT_INACTIVE	0x3D100000
AP_DSLU_ACTIVE	0x3D900000
AP_ACTIVATION_LIMITS_REACHED	0x3E100000
AP_ACTIVATION_LIMITS_REACHED	0x3E100000
AP_INVALID_BRANCH_LINK_TYPE	0x3E900000
AP_PARALLEL_TGS_NOT_SUPPORTED	0x3F100000
AP_INVALID_BRNN_SUPPORT	0x3F900000
AP_DLC_INACTIVE	0x40100000
AP_BRNN_SUPPORT_MISSING	0x40900000
AP_CONFIRMED_BAD_STATE	0x41000000
AP_NO_LINKS_DEFINED	0x41100000
AP_SYNC_LEVEL_NOT_SUPPORTED	0x41600810
AP_INVALID_UPLINK	0x41900000
AP_CONFIRMED_INVALID_FOR_FDX	0x42000000

AP_STOP_DLC_PENDING	0x42100000
AP_INVALID_DOWNLINK	0x42900000
AP_INVALID_LS_ROLE	0x43100000
AP_INVALID_IMPLICIT_UPLINK	0x43900000
AP_INVALID_BTU_SIZE	0x44100000
AP_INVALID_ROCP_NAME	0x44900000
AP_LAST_LINK_ON_ACTIVE_PORT	0x45100000
AP_INVALID_REG_WITH_NN	0x45900000
AP_DYNAMIC_LOAD_ALREADY_REGD	0x46100000
AP_LS_PENDING_RETRY	0x46900000
AP_INVALID_LIST_OPTION	0x47100000
AP_INVALID_COS_TABLE_VERSION	0x47900000
AP_INVALID_RES_NAME	0x48100000
AP_CFRTP_REQUIRED_FOR_MLTG	0x48900000
AP_INVALID_RES_TYPE	0x49100000
AP_INVALID_MLTG_PAC_ALGORITHM	0x49900000
AP_INVALID_ADJ_NNCP_NAME	0x4A100000
AP_LIM_RESRCE_INVALID_FOR_MLTG	0x4A900000
AP_INVALID_NODE	0x4B100000
AP_AUTO_ACT_INVALID_FOR_MLTG	0x4B900000
AP_INVALID_ORIGIN_NODE	0x4C100000
AP_MLTG_LS_VISIBILITY_MISMATCH	0x4C900000
AP_INVALID_TG	0x4D100000
AP_SLTG_LINK_ACTIVE	0x4D900000
AP_INVALID_FQPCID	0x4E100000
AP_MLTG_LINK_PROPERTIES_DIFFER	0x4E900000
AP_INVALID_POOL_NAME	0x4F100000
AP_INVALID_ADJ_CP_NAME	0x4F900000
AP_BAD_TYPE	0x50020000
AP_INVALID_NAU_ADDRESS	0x50100000
AP_INVALID_ENABLE_POOL	0x50300000
AP_INVALID_SEND_TERM_SELF	0x50900000
AP_DEALLOC_BAD_TYPE	0x51000000
AP_LU_NAME_POOL_NAME_CLASH	0x51100000
AP_SECURITY_NOT_VALID	0x51600F08
AP_INVALID_TERM_METHOD	0x51900000
AP_DEALLOC_FLUSH_BAD_STATE	0x52000000
AP_INVALID_PRIORITY	0x52100000
AP_INVALID_DISABLE_BRANCH_AWRN	0x52900000
AP_DEALLOC_CONFIRM_BAD_STATE	0x53000000
AP_INVALID_DNST_LU_NAME	0x53100000
AP_INVALID_SHARING_PROHIBITED	0x53900000
AP_INVALID_HOST_LU_NAME	0x54100000
AP_INVALID_LINK_SPEC_FORMAT	0x54900000
AP_DEALLOC_NOT_LL_BDY	0x55000000
AP_PU_NOT_DEFINED	0x55100000
AP_INVALID_CN_TYPE	0x55900000
AP_INVALID_PU_NAME	0x56100000
AP_INVALID_PU_TYPE	0x56600000
AP_INCONSISTENT_BEST_EFFORT	0x56900000
AP_DEALLOC_LOG_LL_WRONG	0x57000000
AP_CNOS_MODE_NAME_REJECT	0x57010000
AP_INVALID_MAX_IFRM_RCVD	0x57100000
AP_INVALID_CN_TG	0x57900000
AP_INVALID_SYM_DEST_NAME	0x58100000
AP_SEC_BAD_PROTOCOL_VIOLATION	0x58600F08
AP_INVALID_LINK_SPEC_DATA	0x58900000
AP_INVALID_LENGTH	0x59100000
AP_DLC_UI_ONLY	0x59900000
AP_INVALID_ISR_THRESHOLDS	0x5A100000
AP_ADJ_CP_WRONG_TYPE	0x5A900000
AP_BAD_PARTNER_LU_ALIAS	0x5B010000
AP_INVALID_NUM_LUS	0x5B100000
AP_CP_CP_SESS_ALREADY_ACTIVE	0x5B900000
AP_EXCEEDS_MAX_ALLOWED	0x5C010000
AP_CANT_DELETE_ADJ_ENDNODE	0x5C100000
AP_NO_ACTIVE_CP_CP_LINK	0x5C900000

AP_LU_MODE_SESSION_LIMIT_ZERO	0x5D010000
AP_INVALID_RESOURCE_TYPE	0x5D100000
AP_PU_CONC_NOT_SUPPORTED	0x5E100000
AP_INVALID_IMPL_APPN_LINKS_LEN	0x5E900000
AP_CNOS_COMMAND_RACE_REJECT	0x5F010000
AP_DLUR_NOT_SUPPORTED	0x5F100000
AP_INVALID_LIMIT_ENABLE	0x5F900000
AP_INVALID_SVCMG_LIMITS	0x60010000
AP_INVALID_RTP_CONNECTION	0x60100000
AP_INVALID_LS_ATTRIBUTE	0x60900000
AP_FLUSH_NOT_SEND_STATE	0x61000000
AP_PATH_SWITCH_IN_PROGRESS	0x61100000
AP_HPR_NOT_SUPPORTED	0x62100000
AP_SOME_ENABLED	0x62900000
AP_RTP_NOT_SUPPORTED	0x63100000
AP_NONE_ENABLED	0x63900000
AP_COS_TABLE_FULL	0x64100000
AP_INCONSISTENT_IMPLICIT	0x64900000
AP_INVALID_DAYS_LEFT	0x65100000
AP_INVALID_PREFER_ACTIVE_DLUS	0x65900000
AP_ANYNET_NOT_SUPPORTED	0x66100000
AP_INVALID_PERSIST_PIPE_SUPP	0x66900000
AP_INVALID_DISCOVERY_SUPPORT	0x67100000
AP_ACTIVATION_PROHIBITED	0x67900000
AP_SESSION_FAIL_ALREADY_REGD	0x68100000
AP_INVALID_NULL_ADDR_MEANING	0x68900000
AP_CANT_MODIFY_VISIBILITY	0x69100000
AP_INVALID_CPLU_SYNCPT_SUPPORT	0x69900000
AP_CANT_MODIFY_WHEN_ACTIVE	0x6A100000
AP_INVALID_CPLU_ATTRIBUTES	0x6A900000
AP_INVALID_BASE_NUMBER	0x6B100000
AP_INVALID_REG_LEN_SUPPORT	0x6B900000
AP_DEACT_CG_INVALID_CGID	0x6C020000
AP_INVALID_NAME_ATTRIBUTES	0x6C100000
AP_LUNAME_CGID_MISMATCH	0x6C900000
AP_NAU_ADDRESS_MISMATCH	0x6D100000
AP_INVALID_DDDLU_OFFLINE	0x6D900000
AP_POSTED_DATA	0x6E100000
AP_POSTED_NO_DATA	0x6F100000
AP_DEF_PLU_INVALID_FQ_NAME	0x74020000
AP_DLC_DEACTIVATING	0x86020000
AP_INVALID_WILDCARD_NAME	0x8C020000
AP_DUPLICATE	0x8D020000
AP_LU_NAME_WILDCARD_NAME_CLASH	0x8E020000
AP_INVALID_USERID	0x90020000
AP_INVALID_PASSWORD	0x91020000
AP_INVALID_PROFILE	0x93020000
AP_INVALID_TP_NAME	0xA0020000
AP_P_TO_R_INVALID_TYPE	0xA1000000
AP_INVALID_CONV_TYPE	0xA1020000
AP_P_TO_R_NOT_LL_BDY	0xA2000000
AP_P_TO_R_NOT_SEND_STATE	0xA3000000
AP_INVALID_SYNC_LEVEL	0xA3020000
AP_P_TO_R_INVALID_FOR_FDX	0xA5000000
AP_INVALID_LINK_NAME_SPECIFIED	0xB0020000
AP_RCV_AND_WAIT_BAD_STATE	0xB1000000
AP_INVALID_LU_ALIAS	0xB1020000
AP_RCV_AND_WAIT_NOT_LL_BDY	0xB2000000
AP_INVALID_NUM_LS_SPECIFIED	0xB2020000
AP_PLU_ALIAS_CANT_BE_CHANGED	0xB3020000
AP_PLU_ALIAS_ALREADY_USED	0xB4020000
AP_RCV_AND_WAIT_BAD_FILL	0xB5000000
AP_INVALID_AUTO_ACT_SUPP	0xB5020000
AP_CANT_DELETE_IMPLICIT_LU	0xB6020000
AP_FORCED	0xB7020000
AP_INVALID_LS_NAME	0xB7030000
AP_INVALID_LFSID_SPECIFIED	0xB7040000

AP_INVALID_FILTER_TYPE	0xB7050000
AP_INVALID_MESSAGE_TYPE	0xB7060000
AP_CANT_DELETE_CP_LU	0xB7070000
AP_ALL_RESOURCES_NOT_DEFINED	0xB7090000
AP_INVALID_LIST_TYPE	0xB70A0000
AP_RESOURCE_NAME_NOT_ALLOWED	0xB70B0000
AP_LU_ALIAS_CANT_BE_CHANGED	0xB8020000
AP_LU_ALIAS_ALREADY_USED	0xB9020000
AP_INVALID_LINK_ENABLE	0xBA020000
AP_INVALID_CLU_COMPRESSION	0xBB020000
AP_INVALID_DLUR_SUPPORT	0xBC020000
AP_ALREADY_STARTING	0xC0010000
AP_RCV_IMMD_BAD_STATE	0xC1000000
AP_INVALID_LINK_NAME	0xC1010000
AP_INVALID_USER_DEF_1	0xC3010000
AP_RCV_IMMD_BAD_FILL	0xC4000000
AP_INVALID_USER_DEF_2	0xC4010000
AP_INVALID_NODE_TYPE	0xC4020000
AP_INVALID_USER_DEF_3	0xC5010000
AP_INVALID_NAME_LEN	0xC5020000
AP_INVALID_NETID_LEN	0xC6020000
AP_INVALID_NODE_TYPE_FOR_HPR	0xC8020000
AP_INVALID_MAX_DECOMPRESS_LVL	0xC9020000
AP_INVALID_CP_NAME	0xCA010000
AP_INVALID_COMP_IN_SERIES	0xCA020000
AP_INVALID_LIMITED_RESOURCE	0xCE010000
AP_RCV_AND_POST_BAD_STATE	0xD1000000
AP_INVALID_BYTE_COST	0xD1010000
AP_RCV_AND_POST_NOT_LL_BDY	0xD2000000
AP_RCV_AND_POST_BAD_FILL	0xD5000000
AP_INVALID_TIME_COST	0xD6010000
AP_BAD_RETURN_STATUS_WITH_DATA	0xD7000000
AP_LOCAL_CP_NAME	0xD7010000
AP_LS_ACTIVE	0xDA010000
AP_INVALID_FQ_OWNING_CP_NAME	0xDB020000
AP_R_T_S_BAD_STATE	0xE1000000
AP_R_T_S_INVALID_FOR_FDX	0xE2000000
AP_BAD_LL	0xF1000000
AP_SEND_DATA_NOT_SEND_STATE	0xF2000000
AP_CP_OR_SNA_SVCMG_UNDELETABLE	0xF3010000
AP_SEND_DATA_INVALID_TYPE	0xF4000000
AP_DEL_MODE_DEFAULT_SPCD	0xF4010000
AP_SEND_DATA_CONFIRM_SYNC_NONE	0xF5000000
AP_MODE_NAME_NOT_DEFD	0xF5010000
AP_SEND_DATA_NOT_LL_BDY	0xF6000000
AP_MODE_UNDELETABLE	0xF6010000
AP_SEND_TYPE_INVALID_FOR_FDX	0xF7000000
AP_INVALID_FQ_LU_NAME	0xFD010000
AP_INVALID_PARTNER_LU	0xFE010000
AP_INVALID_LOCAL_LU	0xFF010000

付録 B. 共通戻りコード

この付録では、複数の APPC verb に共通した 1 次戻りコードについて (該当する場合は 2 次戻りコードについても) 説明します。

verb 固有の戻りコードについては、個々の verb に関する資料で説明しています。

次の各項では、共通戻りコードについて説明します。

AP_ALLOCATION_ERROR

1 次および 2 次戻りコードは、次のとおりです。

primary_rc

AP_ALLOCATION_ERROR
APPC は会話の割り振りに失敗しました。会話状態は、RESET に設定されました。このコードは、[MC_]ALLOCATE の後に発行した verb によって戻される場合があります。

secondary_rc
次の値があります。

AP_ALLOCATION_FAILURE_NO_RETRY
構成エラーまたはセッション・プロトコル・エラーなど、永続的な条件のために会話を割り振ることができません。エラーを判別するため、システム管理者はエラー・ログ・ファイルを調べる必要があります。エラーの訂正が済むまで、割り振りの再試行を試みてはなりません。

AP_ALLOCATION_FAILURE_RETRY
リンクの障害など、一時的な条件のために会話を割り振ることができませんでした。障害の原因は、システム・エラー・ログに記録されています。割り振りを再試行してください。ただし、条件がクリアされるよう、できるだけタイムアウトの後に再試行してください。

AP_CONVERSATION_TYPE_MISMATCH
パートナー LU または TP が、割り振り要求で指定された会話タイプ (基本会話またはマップ式会話) をサポートしていません。

AP_PIP_NOT_ALLOWED
割り振り要求で PIP データが指定されましたが、パートナー TP は PIP データを受け入れませんでした。この原因としては、パートナー TP が、PIP データの受信をサポートしていない APPC インプリメンテーションを使用しているため、またはパートナーが CPI-C アプリケーションである (CPI-C は PIP データをサポートしていません) ために、このデータを必要としていないことが考えられます。

AP_PIP_NOT_SPECIFIED_CORRECTLY
パートナー TP は PIP データを必要としていますが、割り振り要求で PIP データが指定されなかったか、指定したパラメーターの数が正しくありませんでした。

AP_SECURITY_NOT_VALID
割り振り要求の中で指定したユーザー ID またはパスワードが、パートナー LU によって受け入れられませんでした。

AP_SYNC_LEVEL_NOT_SUPPORTED
パートナー TP が、割り振り要求の中で指定された *sync_level* (AP_NONE、AP_CONFIRM_SYNC_LEVEL、AP_SYNCPT のいずれか) をサポートしていないか、*sync_level* が認識されませんでした。

AP_TP_NAME_NOT_RECOGNIZED
パートナー LU が、割り振り要求の中で指定された TP 名を認識しませんでした。

AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
リモート LU は、要求されたパートナー TP を開始できなかったため、割り振り要求をリジェクトしました。この条件は永続的なものです。エラーの理由がリモート・ノードのログに記録されていることがあります。エラーの原因が訂正されるまで、割り振りを再試行してはなりません。

AP_TRANS_PGM_NOT_AVAIL_RETRY
リモート LU は、要求されたパートナー TP を開始できなかったため、割り振り要求をリジェクトしました。この条件は、タイムアウトなど、一時的なものです。エラーの理由がリモート・ノードのログに記録されていることがあります。割り振りを再試行してください。ただし、条件がクリアされるよう、できるだけタイムアウトの後に再試行してください。

AP_SEC_BAD_PROTOCOL_VIOLATION
リモート LU は、プロトコル違反のために割り振り要求をリジェクトしました。

AP_SEC_BAD_PASSWORD_EXPIRED
リモート LU は、指定されたパスワードが有効ではなくなったため、割り振り要求をリジェクトしました。

AP_SEC_BAD_PASSWORD_INVALID
リモート LU は、パスワードが有効でなかったため、割り振り要求をリジェクトしました。

AP_SEC_BAD_USERID_REVOKED
リモート LU は、ユーザー ID が有効ではなくなったため、割り振り要求をリジェクトしました。

AP_SEC_BAD_USERID_INVALID
リモート LU は、ユーザー ID が有効でなかったため、割り振り要求をリジェクトしました。

AP_SEC_BAD_USERID_MISSING
リモート LU は、ユーザー ID が必須であるのに指定されなかったため、割り振り要求をリジェクトしました。

AP_SEC_BAD_PASSWORD_MISSING
リモート LU は、パスワードが必須であるのに指定されなかったため、割り振り要求をリジェクトしました。

AP_SEC_BAD_GROUP_INVALID
リモート LU は、グループが有効でなかったため、割り振り要求をリジェクトしました。

AP_SEC_BAD_UID_REVOKED_IN_GRP
リモート LU は、ユーザー ID がグループ内になくなったため、割り振り要求をリジェクトしました。

AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
リモート LU は、ユーザー ID がグループ内になかったため、割り振り要求をリジェクトしました。

AP_SEC_BAD_UNAUTHRZD_AT_RLU
リモート LU は、この TP をリモート LU 上で開始する権限がそのユーザー ID にないため、割り振り要求をリジェクトしました。

AP_SEC_BAD_UNAUTHRZD_FROM_LLU
リモート LU は、この TP をローカル LU から開始する権限がそのユーザー ID にないため、割り振り要求をリジェクトしました。

AP_SEC_BAD_UNAUTHRZD_TO_TP
リモート LU は、この TP を開始する権限がそのユーザー ID にないため、割り振り要求をリジェクトしました。

AP_SEC_BAD_INSTALL_EXIT_FAILED
リモート LU は、必須の出口をインストールすることに失敗したため、割り振り要求をリジェクトしました。

AP_SEC_BAD_PROCESSING_FAILURE
リモート LU は、リモート LU での処理の失敗のため、割り振り要求をリジェクトしました。

セキュリティーの障害に関する詳細な情報を提供することでセキュリティー上の危険を招く可能性を考慮し、これら AP_SEC_BAD_* 戻りコードはサポートしないことも可能です。 その場合、これらのすべてのエラーは、アプリケーションには AP_SECURITY_NOT_VALID として報告されます。詳細については、「*IBM Communications Server for Linux 管理コマンド解説書*」または「*IBM Communications Server for AIX 管理コマンド解説書*」の **define_defaults** コマンドの項と、「*IBM Communications Server for Linux NOF プログラマーズ・ガイド*」または「*IBM Communications Server for AIX NOF プログラマーズ・ガイド*」の DEFINE_DEFAULTS NOF verb の項を参照してください。

AP_BACKED_OUT

AIX, LINUX

1 次および 2 次戻りコードは、次のとおりです。

primary_rc

AP_BACKED_OUT
パートナー TP (または、同じ作業論理単位に参加している別の TP) は、バックアウト要求を発行しました。同期点管理プログラムは、次のいずれかとなる 2 次戻りコードに基づいて適切な同期点処理を実行します。

secondary_rc
次の値があります。

AP_BO_NO_RESYNC
パートナー TP は、リソースのバックアウトを完了しました。

AP_BO_RESYNC
パートナー TP がリソースのバックアウトを試みているときに障害が発生しました。再同期は、まだ進行中です。

AP_CANCELLED

WINDOWS

1 次戻りコードは、次のとおりです。

primary_rc

AP_CANCELLED
verb は `WinAsyncAPPC` エントリー・ポイントを使用して発行され、次に `WinAPPCCancel` エントリー・ポイントを使用して取り消されました。これらのエントリー・ポイントについての詳細は、41 ページの『APPC エントリー・ポイント: Windows システム』を参照してください。

2 次戻りコードは戻りません。

AP_COMM_SUBSYSTEM_ABENDED

1 次戻りコードは、次のとおりです。

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
この戻りコードは、Communications Server ソフトウェアが異常終了

したか、 LAN に問題があることを示しています。異常終了の原因を判別するため、システム管理者はエラー・ログを調べる必要があります。

2 次戻りコードは戻りません。

AP_COMM_SUBSYSTEM_NOT_LOADED

1 次戻りコードは、次のとおりです。

primary_rc

AP_COMM_SUBSYSTEM_NOT_LOADED

この戻りコードは、次のいずれかの条件が原因で、 TP_STARTED verb または RECEIVE_ALLOCATE verb を使用して TP を開始する試みが受け入れられなかったことを示しています。

AIX, LINUX

- Communications Server ソフトウェアがロードされなかったか、この TP が使用する LU を所有しているローカル・ノードが開始されていませんでした。 訂正アクションについて、システム管理者に問い合わせてください。
- Communications Server ライセンスによって許可されている最大ユーザー数が、すでに Communications Server を使用しています。ユーザー制限を超えるため、現時点でこの TP を開始することはできません。後でシステム上のユーザー数が少なくなった時点で、TP を開始できるようになる可能性があります。

WINDOWS

- Communications Server ソフトウェアがロードされていないか、この APPC LU が使用する通信コンポーネントがアクティブになっていません。訂正アクションについて、システム管理者に問い合わせてください。
- TP_STARTED verb に指定された LU 別名が認識されません。指定された LU 別名が、構成ファイル内の APPC ローカル LU と一致しているか確認してください。
- Communications Server ライセンスで許可されている Communications Server ユーザーの最大数が、使用中の APPC ローカル LU を所有するローカル・ノードをすでに使用しています。ユーザー制限を超えるため、現時点でこの TP を開始することはできません。後でシステム上のユーザー数が少なくなった時点で、TP を開始できるようになる可能性があります。

2 次戻りコードは戻りません。

AP_CONV_FAILURE_NO_RETRY

1 次戻りコードは、次のとおりです。

primary_rc

AP_CONV_FAILURE_NO_RETRY
セッション・プロトコル・エラーなど、永続的な条件のために会話が終了しました。エラーの原因を判別するため、システム管理者はシステム・エラー・ログを調べる必要があります。エラーが訂正されるまで、会話を再試行してはなりません。

2 次戻りコードは戻りません。

AP_CONV_FAILURE_RETRY

1 次戻りコードは、次のとおりです。

primary_rc

AP_CONV_FAILURE_RETRY
一時エラーのために会話が終了しました。 TP を再始動して、問題が再度発生するかどうかを確認してください。問題が再度発生する場合は、システム管理者がエラー・ログを調べてエラーの原因を判別する必要があります。

2 次戻りコードは戻りません。

AP_CONVERSATION_TYPE_MIXED

1 次戻りコードは、次のとおりです。

primary_rc

AP_CONVERSATION_TYPE_MIXED
TP で基本とマップ式の両方の verb を発行しました。1 つの会話では、1 つのタイプのみを発行できます。

2 次戻りコードは戻りません。

AP_DEALLOC_ABEND

1 次戻りコードは、次のとおりです。

primary_rc

AP_DEALLOC_ABEND
次のいずれかの理由で会話の割り振りが解除されました。

- パートナー TP が、*dealloc_type* を AP_ABEND に設定して MC_DEALLOCATE verb を発行しました。
- パートナー TP が異常終了したため、パートナー LU が MC_DEALLOCATE 要求を送信しました。

2 次戻りコードは戻りません。

AP_DEALLOC_ABEND_PROG

1 次戻りコードは、次のとおりです。

primary_rc

AP_DEALLOC_ABEND_PROG
次のいずれかの理由で会話の割り振りが解除されました。

- パートナー TP が、*dealloc_type* を AP_ABEND_PROG に設定して DEALLOCATE verb を発行しました。
- パートナー TP が異常終了したため、パートナー LU が DEALLOCATE 要求を送信しました。

2 次戻りコードは戻りません。

AP_DEALLOC_ABEND_SVC

1 次戻りコードは、次のとおりです。

primary_rc

AP_DEALLOC_ABEND_SVC
パートナー TP が、*dealloc_type* を AP_ABEND_SVC に設定して DEALLOCATE verb を発行したため、会話の割り振りが解除されました。

2 次戻りコードは戻りません。

AP_DEALLOC_ABEND_TIMER

1 次戻りコードは、次のとおりです。

primary_rc

AP_DEALLOC_ABEND_TIMER
パートナー TP が、*dealloc_type* を AP_ABEND_TIMER に設定して DEALLOCATE verb を発行したため、会話の割り振りが解除されました。

2 次戻りコードは戻りません。

AP_DEALLOC_NORMAL

1 次戻りコードは、次のとおりです。

primary_rc

AP_DEALLOC_NORMAL
この戻りコードは、エラーを示すものではありません。

パートナー TP は、*dealloc_type* を次のいずれかに設定して [MC_]DEALLOCATE verb を発行しました。

- AP_FLUSH
- 会話の同期レベルが AP_NONE として指定された AP_SYNC_LEVEL

2 次戻りコードは戻りません。

AP_DUPLEX_TYPE_MIXED

1 次戻りコードは、次のとおりです。

primary_rc

AP_DUPLEX_TYPE_MIXED
TP が、会話に一致していない全二重タイプを指定して会話 verb を発行しました。会話が全二重 ([MC_]ALLOCATE または RECEIVE_ALLOCATE の *duplex_type* パラメーターで指定) であると、 TP は、この会話内の他のすべての verb の *opext* パラメーターに、AP_FULL_DUPLEX_CONVERSATION オプションを設定する必要があります。会話が半二重である場合は、このオプションを設定しないでください。

2 次戻りコードは戻りません。

AP_INVALID_VERB

1 次戻りコードは、次のとおりです。

primary_rc

AP_INVALID_VERB
verb に指定された命令コードが無効です。verb は実行されませんでした。

この戻りコードは、全二重会話で [MC_]RECEIVE_AND_POST verb を発行しようとした場合にも戻ります。[MC_]RECEIVE_AND_POST は、半二重会話でのみ使用できます。

2 次戻りコードは戻りません。

AP_INVALID_VERB_SEGMENT

WINDOWS

1 次戻りコードは、次のとおりです。

primary_rc

AP_INVALID_VERB_SEGMENT
verb 制御ブロックが、データ・セグメントの終わりを超過しています。verb は実行されませんでした。

2 次戻りコードは戻りません。

AP_PROG_ERROR_NO_TRUNC

1 次戻りコードは、次のとおりです。

primary_rc

AP_PROG_ERROR_NO_TRUNC
会話が SEND 状態のときに、パートナー TP が次のいずれかの verb を発行しました。

- *err_type* を AP_PROG に設定した SEND_ERROR
- MC_SEND_ERROR

データは切り捨てられませんでした。

2 次戻りコードは戻りません。

AP_PROG_ERROR_PURGING

1 次戻りコードは、次のとおりです。

primary_rc

AP_PROG_ERROR_PURGING
パートナー TP が次のいずれかの verb を発行しました。

- *err_type* を AP_PROG に設定した SEND_ERROR
- MC_SEND_ERROR

Receive、Pending_Post、Confirm、Confirm_Send、または Confirm_Deallocate 状態のとき。 送信され、まだ受信されていないデータは除去されます。

2 次戻りコードは戻りません。

AP_PROG_ERROR_TRUNC

1 次戻りコードは、次のとおりです。

primary_rc

AP_PROG_ERROR_TRUNC
パートナー TP は SEND 状態のときに、不完全な論理レコードを送信した後、 *err_type* を AP_PROG に設定した SEND_ERROR verb を発行しました。 ローカル TP は、受信 verb を通じてその論理レコードの最初の部分を受信した可能性があります。

2 次戻りコードは戻りません。

AP_SVC_ERROR_NO_TRUNC

1 次戻りコードは、次のとおりです。

primary_rc

AP_SVC_ERROR_NO_TRUNC
パートナー TP (またはパートナー LU) は SEND 状態のときに、*err_type* を AP_SVC に設定した SEND_ERROR verb を発行しました。 データは切り捨てられませんでした。

2 次戻りコードは戻りません。

AP_SVC_ERROR_PURGING

1 次戻りコードは、次のとおりです。

primary_rc

AP_SVC_ERROR_PURGING
パートナー TP (またはパートナー LU) は Receive、Pending_Post、Confirm、Confirm_Send、Confirm_Deallocate のいずれかの状態のときに、*err_type* を AP_SVC に設定した SEND_ERROR verb を発行しました。 パートナー TP へ送信したデータは、除去された可能性があります。

2 次戻りコードは戻りません。

AP_SVC_ERROR_TRUNC

1 次戻りコードは、次のとおりです。

primary_rc

AP_SVC_ERROR_TRUNC
パートナー TP (またはパートナー LU) は Send 状態のときに、不完全な論理レコードを送信した後、SEND_ERROR verb を発行しました。ローカル TP は、その論理レコードの最初の部分を受信した可能性があります。

2 次戻りコードは戻りません。

AP_THREAD_BLOCKING

WINDOWS

1 次戻りコードは、次のとおりです。

primary_rc

AP_THREAD_BLOCKING
verb は APPC (ブロッキング) エントリー・ポイントを使用して発行されましたが、他のブロッキング APPC verb がすでに未解決となっています。これらのエントリー・ポイントについての詳細は、41 ページの『APPC エントリー・ポイント: Windows システム』を参照してください。

2 次戻りコードは戻りません。

AP_TP_BUSY

1 次戻りコードは、次のとおりです。

primary_rc

AP_TP_BUSY
ローカル TP は、APPC が同じ TP からの別のコールを処理しているときに、APPC へのコールを発行しました。この戻りコードが発生する可能性があるのは、ローカル TP が複数のプロセスを開始し、2 つ以上のプロセスが同じ *tp_id* を使用して APPC コールを発行している場合です。 ただし、個々のプロセスで独自の TP_STARTED verb か RECEIVE_ALLOCATE verb を発行し、独自の *tp_id* を取得するようにしてください。複数のプロセスが同じ *tp_id* を使用した場合の結果は予測できません。

WINDOWS

この戻りコードは、verb を発行しているアプリケーションが、`PostMessage` ではなく Windows 関数 `SendMessage` を使用して呼び出されたことを示すこともあります。アプリケーションは、この状態ではどの verb を発行できません。詳細については、 60 ページの『Windows の考慮事項』を参照してください。

2 次戻りコードは戻りません。

AP_UNEXPECTED_SYSTEM_ERROR

1 次戻りコードは、次のとおりです。

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR
オペレーティング・システムが、ローカル TP からの APPC コールを処理中にエラーを検出しました。オペレーティング・システム戻りコードは、*secondary_rc* によって戻されます。問題が解決されない場合は、システム管理者に連絡してください。

AIX, LINUX

オペレーティング・システム戻りコードの意味については、エラーが発生したコンピューター上の **/usr/include/errno.h** ファイルを参照してください。

WINDOWS

オペレーティング・システムの戻りコードの意味については、オペレーティング・システムの資料を参照してください。

2 次戻りコードは戻りません。

付録 C. APPC の状態の変更

以下の表は、各 APPC verb を発行できる会話状態、および verb の完了時に起こる状態の変化を示しています。場合によっては、状態の変更が、verb へ戻された *primary_rc* パラメーターによって異なることもあります。そのような場合は、該当する *primary_rc* 値が verb と同じ列に示してあります。 *primary_rc* 値が示されていない場所は、各テーブルにある注で示されることを除き、状態変更はすべての戻りコードと同じです。

考えられる会話の状態は、列ヘッダーとして示しています。verb と *primary_rc* 値の組み合わせごとに、次の省略形と記号がそれぞれの状態の下に示されています。これは、その状態で verb を発行した結果を示しています。

X この状態では verb を発行できません。

T, S, R, ...
verb 完了後の会話の状態を示します。

半二重会話の場合:

T	Reset (リセット)
S	Send (送信)
SP	Send_Pending (送信 _ 保留)
R	Receive (受信)
C	Confirm (確認)
CS	Confirm_Send (確認 _ 送信)
CD	Confirm_Deallocate (確認 _ 割り振り解除)
P	Pending_Post (保留 _ 送信)

全二重会話の場合:

T	Reset (リセット)
SR	Send_Receive (送信_受信)
S	Send_Only (送信専用)
R	Receive_Only (受信専用)

- verb を発行した後、会話状態は存在しません。

/ verb は Reset 状態から新規会話を開始するため、前の状態の考慮は適用されません。既存の会話には影響がありません。

(ブランク)
この状態では、示されている戻りコードは戻されません。

半二重会話

verb と *primary_rc* 値	発行時の状態							
	Reset (リセット) **(T)**	**Send (送信)** **(S)**	**Send (送信)** **Pend** **(SP)**	**Recv** **(R)**	**Confm** **(C)**	**Confm** **Send (送信)** **(CS)**	**Confm** **Deall** **(CD)**	**Pend** **Post** **(PP)**
TP_STARTED								
AP_OK	T	/	/	/	/	/	/	/
その他の *primary_rc* 値	-							
TP_ENDED								
AP_OK	-	-	-	-	-	-	-	-
その他の *primary_rc* 値	T	S	SP	R	C	CS	CD	P
RECEIVE_ALLOCATE								
AP_OK	R	/	/	/	/	/	/	/
その他の *primary_rc* 値	-							
GET_LU_STATUS	X	S	SP	R	C	CS	CD	P
GET_TP_PROPERTIES	T	S	SP	R	C	CS	CD	P
SET_TP_PROPERTIES	T	S	SP	R	C	CS	CD	P
GET_TYPE	X	S	SP	R	C	CS	CD	P
CANCEL_CONVERSATION	X	T	T	T	T	T	T	T
[MC_]ALLOCATE								
AP_OK	S	/	/	/	/	/	/	/
その他の *primary_rc* 値	T							
[MC_]CONFIRM								
AP_OK	X	S	S	X	X	X	X	X
AP_ERROR		R	R					
[MC_]CONFIRMED	X	X	X	X	R	S	T	X
[MC_]DEALLOCATE								
AP_ABEND_* *dealloc_type* 値	X	T	T	T	T	T	T	T
その他の *dealloc_type* 値								
AP_ERROR	X	R	R	X	X	X	X	X
その他の *primary_rc* 値	T	T						
[MC_]FLUSH	X	S	S	X	X	X	X	X
[MC_]GET_ ATTRIBUTES	X	S	SP	R	C	CS	CD	P
[MC_]PREPARE_TO_RECEIVE	X	R	R	X	X	X	X	X
[MC_]RECEIVE_AND_POST (注 4 を参照)	X	P	P	P	X	X	X	X
[MC_]RECEIVE_AND_WAIT	X	注 5 を参照	注 5 を参照	注 5 を参照	X	X	X	X
[MC_]RECEIVE_IMMEDIATE	X	X	X	注 5 を参照	X	X	X	X
[MC_]RECEIVE_EXPEDITED_DATA	X	X	X	R	C	X	X	P
[MC_]REQUEST_TO_SEND	X	X	X	R	C	X	X	P
[MC_]SEND_CONVERSATION	T	/	/	/	/	/	/	/
[MC_]SEND_DATA								
AP_OK	X	S	S	X	X	X	X	X
AP_ERROR		R						
[MC_]SEND_ERROR								
AP_OK	X	S	S	S	S	S	S	S
AP_ERROR		R						
[MC_]SEND_EXPEDITED_DATA	X	X	X	R	C	X	X	P
[MC_]TEST_RTS	X	S	S	R	C	CS	CD	P
[MC_]TEST_RTS_AND_POST	X	S	S	R	C	CS	CD	P

注:

1. この表の戻りコードの列で、AP_ERROR という省略形は、次の戻りコードに使用されます。
   AP_BACKED_OUT
   AP_PROG_ERROR_TRUNC
   AP_PROG_ERROR_NO_TRUNC
   AP_PROG_ERROR_PURGING
   AP_SVC_ERROR_TRUNC
   AP_SVC_ERROR_NO_TRUNC
   AP_SVC_ERROR_PURGING
2. 以下の戻りコードのいずれかを受け取った場合、会話は常に Reset 状態となります。
   AP_ALLOCATION_ERROR
   AP_COMM_SUBSYSTEM_ABENDED
   AP_COMM_SUBSYSTEM_NOT_LOADED
   AP_CONV_FAILURE_RETRY
   AP_CONV_FAILURE_NO_RETRY
   AP_DEALLOC_ABEND
   AP_DEALLOC_ABEND_PROG
   AP_DEALLOC_ABEND_SVC
   AP_DEALLOC_ABEND_TIMER
   AP_DEALLOC_NORMAL
3. 次に示す OK 以外の戻りコードは、状態の変更を起こしません。会話は常に、verb が発行された状態に留まります。
   AP_CONVERSATION_TYPE_MIXED
   AP_INVALID_VERB
   AP_PARAMETER_CHECK
   AP_STATE_CHECK
   AP_TP_BUSY
   AP_UNEXPECTED_SYSTEM_ERROR
   AP_UNSUCCESSFUL
4. [MC_]RECEIVE_AND_POST が発行され、 AP_OK の最初の *primary_rc* を受け取ると、会話は Pending_Post 状態に変わります。指定されたコールバック・ルーチンがt呼び出され、verb の完了を示すと、新規会話状態は、注 5 の *primary_rc* と *what_rcvd* パラメーターによって異なります。
5. いずれかの受信 verb を発行した後の状態の変更は、 *primary_rc* と *what_rcvd* の両方のパラメーターによって異なります。

   primary_rc パラメーターが AP_PROG_ERROR か AP_SVC_ERROR、または ([MC_]RECEIVE_IMMEDIATE のみ) AP_UNSUCCESSFUL の場合、新しい状態は Receive です。

   primary_rc パラメーターが AP_DEALLOC の場合、新しい状態は Reset です。

   primary_rc パラメーターが AP_OK の場合、新しい状態は、次のように *what_rcvd* パラメーターの値によって異なります。

AP_DATA, AP_DATA_COMPLETE, AP_DATA_INCOMPLETE
Receive 状態

AP_SEND
Send 状態

AP_DATA_SEND, AP_DATA_COMPLETE_SEND
Send Pending 状態

AP_CONFIRM_WHAT_RCVD, AP_DATA_CONFIRM, AP_DATA_COMPLETE_CONFIRM
Confirm 状態

AP_CONFIRM_SEND, AP_DATA_CONFIRM_SEND, AP_DATA_COMPLETE_CONFIRM_SEND
Confirm Send 状態

AP_CONFIRM_DEALLOCATE, AP_DATA_CONFIRM_DEALLOCATE, AP_DATA_COMPLETE_CONFIRM_DEALL
Confirm Deallocate 状態

全二重会話

verb と *primary_rc* **値**	**発行時の状態**			
	Reset (リセット) **(T)**	**Send Receive (送信 受信)** **(SR)**	**Send Only (送信専用)** **(S)**	**Receive Only (受信専用)** **(R)**
TP_STARTED				
AP_OK	T	/	/	/
その他の *primary_rc* 値	-			
TP_ENDED				
AP_OK	-	-	-	-
その他の *primary_rc* 値	T	SR	S	R
RECEIVE_ALLOCATE				
AP_OK	SR	/	/	/
その他の *primary_rc* 値	-			
GET_LU_STATUS	X	SR	S	R
GET_TP_PROPERTIES	T	SR	S	R
SET_TP_PROPERTIES	T	SR	S	R
GET_TYPE	X	SR	S	R
CANCEL_CONVERSATION	X	T	T	T
[MC_]ALLOCATE				
AP_OK	SR	/	/	/
その他の *primary_rc* 値	T			
[MC_]DEALLOCATE				
AP_ABEND_* *dealloc_type* 値	X	T	T	T
その他の *dealloc_type* 値	X	R	T	X

verb と *primary_rc* 値	**発行時の状態**			
	Reset (リセット) **(T)**	**Send Receive (送信 受信)** **(SR)**	**Send Only (送信専用)** **(S)**	**Receive Only (受信専用)** **(R)**
[MC_]FLUSH	X	SR	S	X
[MC_]GET_ ATTRIBUTES	X	SR	S	R
[MC_]RECEIVE_AND_WAIT				
AP_OK	X	SR	X	R
AP_ERROR	X	SR	X	R
AP_DEALLOC_NORMAL	X	S	X	T
[MC_]RECEIVE_IMMEDIATE				
AP_OK	X	SR	X	R
AP_ERROR	X	SR	X	R
AP_DEALLOC_NORMAL	X	S	X	T
[MC_]RECEIVE_ EXPEDITED_DATA	X	SR	S	R
[MC_]SEND_DATA				
AP_OK	X	SR	S	X
AP_ERROR	X	SR	T	X
[MC_]SEND_ERROR				
AP_OK	X	SR	S	X
AP_ERROR	X	SR	T	X
[MC_]SEND_ EXPEDITED_DATA	X	SR	S	R

注:

1. この表の戻りコードの列で、AP_ERROR という省略形は、次の戻りコードに使用されます。
   AP_BACKED_OUT
   AP_PROG_ERROR_TRUNC
   AP_PROG_ERROR_NO_TRUNC
   AP_SVC_ERROR_TRUNC
   AP_SVC_ERROR_NO_TRUNC
2. 以下の戻りコードのいずれかを受け取った場合、会話は常に Reset 状態となります。
   AP_ALLOCATION_ERROR
   AP_COMM_SUBSYSTEM_ABENDED
   AP_COMM_SUBSYSTEM_NOT_LOADED
   AP_CONV_FAILURE_RETRY
   AP_CONV_FAILURE_NO_RETRY
   AP_DEALLOC_ABEND
   AP_DEALLOC_ABEND_PROG
   AP_DEALLOC_ABEND_SVC
   AP_DEALLOC_ABEND_TIMER

3. 次に示す OK 以外の戻りコードは、状態の変更を起こしません。会話は常に、verb が発行された状態に留まります。
   AP_CONVERSATION_TYPE_MIXED
   AP_INVALID_VERB
   AP_PARAMETER_CHECK
   AP_STATE_CHECK
   AP_TP_BUSY
   AP_UNEXPECTED_SYSTEM_ERROR
   AP_UNSUCCESSFUL

付録 D. SNA LU6.2 サポート

この付録では、Communications Server にインプリメンテーションされた APPC と LU 6.2 アーキテクチャーの関係について詳しく説明します。ここでは、次の情報を記載します。

- Communications Server がサポートする LU6.2 オプション・セットの要約
- Communications Server にインプリメンテーションされた APPC に組み込まれている制御オペレーター verb のリスト
- Communications Server で管理ツールまたは NOF API によって機能が実行される制御オペレーター verb のリスト

LU6.2 オプション・セット・サポート

Communications Server にインプリメントされた APPC は、LU6.2 機能の基本セットと、いくつかのオプション・セットをサポートします。これらのオプション・セットの一部は APPC verb によってサポートされ、それ以外は管理ツールまたは NOF API によってサポートされます。

以下の表は、Communications Server でサポートされるオプション・セットと、IBM の「*Transaction Programmer's Reference Manual for LU Type 6.2.*」で指定されているオプション・セットの参照番号の一覧です (この IBM 資料の前のバージョンは、別の参照番号を使用しています)。

APPC verb がサポートする LU6.2 オプション・セット

参照番号	オプション・セット
101	LU の送信バッファーのフラッシュ
102	GET_ATTRIBUTES
103	POST のためのテスト付き POST_ON_RECEIPT *
104	待ちが付いた POST_ON_RECEIPT *
105	PREPARE_TO_RECEIVE
106	RECEIVE_IMMEDIATE
109	TP 名とインスタンス ID の取得
110	GET_CONVERSATION_TYPE
112	全二重会話および優先データ
113	非ブロッキング・サポート
201	コンテンション勝者セッションの待機割り振り
203	セッションの即時割り振り
204	同じ LU に位置するプログラム間の会話
205	セッション解放時の待機割り振り
211	セッション・レベル LU-LU 検査
212	ユーザー ID 検証
213	プログラム提供のユーザー ID とパスワード
214	ユーザー ID 許可
241	PIP データの送信
242	PIP データの受信
243	アカウンティング

参照番号	オプション・セット
244	長いロック
245	REQUEST_TO_SEND を受信したかどうかのテスト
247	ユーザー制御データ
290	システム・ログへのデータのログ記録
291	マップ式会話 LU サービス・コンポーネント
401	高信頼片方向ブラケット
616	CPSVCMG モード名サポート

* オプション 103 と 104 は、[MC_]RECEIVE_AND_POST verb がサポートします。

管理ツールと NOF API によってサポートされる LU6.2 オプション・セット

参照番号	オプション・セット
501	CHANGE_SESSION_LIMIT
502	ACTIVATE_SESSION
504	DEACTIVATE_SESSION
505	LU 定義 verb
601	*min_conwinners_target* パラメーター
602	*responsible* (TARGET) パラメーター
603	*drain_target* (NO) パラメーター
604	*force* パラメーター
605	LU 間セッション限度
606	ローカルで認識される LU 名
607	解釈されない LU 名
610	最大 RU サイズ限界
611	セッション・レベルの強制暗号化
612	コンテンション勝者の自動活動化の限界
613	ローカル最大 (LU、モード) セッション限度

制御オペレーター verb サポート

次に示す制御オペレーター verb の機能は、Communications Server にインプリメントされた APPC の一部として提供されます。

RECEIVE_ALLOCATE
TP_STARTED
TP_ENDED

次に示す制御オペレーター verb の機能は、Communications Server 管理プログラムおよび NOF API によって提供されます。

INITIALIZE_SESSION_LIMITS
CHANGE_SESSION_LIMITS
RESET_SESSION_LIMITS
DISPLAY_LU
DISPLAY_REMOTE_LU
DISPLAY_MODE

DISPLAY_TP
ACTIVATE_SESSION
DEACTIVATE_SESSION
DEFINE_LOCAL_LU
DEFINE_REMOTE_LU
DEFINE_MODE
DELETE

付録 E. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
P.O. Box 12195
3039 Cornwallis Road
Research Triangle Park, NC 27709-2195
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物にも、次のように、著作権表示を入れていただく必要があります。 ® (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 ® Copyright IBM Corp. 2000, 2005, 2006, 2007, 2008, 2009. All rights reserved.

商標

IBM、IBM ロゴ、および ibm.com は、International Business Machines Corp. の商標または登録商標であって、世界中の多くの国または地域にて登録されています。その他の製品名およびサービス名は、IBM またはその他の企業の商標である可能性があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml の「Copyright and trademark information」をご覧ください。

Adobe は、Adobe Systems Incorporated の米国およびその他の国における登録商標です。

Intel および Pentium は、Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは Sun Microsystems, Inc.の米国およびその他の国における商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

Microsoft、Windows、および Windows NT は、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

参考文献

以下の IBM 資料では、本書で説明しているトピックについての情報を記載しています。資料は、次のトピック別に大きく分けてあります。

- IBM Communications Server for AIX
- IBM Communications Server for Linux
- システム・ネットワーク体系 (SNA)
- 拡張プログラム間通信機能 (APPC)
- プログラミング

IBM Communications Server for AIX および IBM Communications Server for Linux の資料については、その要旨が説明されています。他の資料については、ここではタイトルおよび資料番号のみが示されています。

IBM Communications Server for AIX 関連資料

IBM Communications Server for AIX ライブラリーは、以下の資料により構成されています。なお、これらの資料のソフトコピー版が CD-ROM で提供されています。CD-ROM 上のソフトコピー・ファイルにアクセスする方法については、「*IBM Communications Server for AIX 入門*」を参照してください。これらのソフトコピー・ブックをシステムにインストールするには、9 〜 15 MB のハードウェア・ディスク・スペースが必要になります (このスペースは、どの各国語バージョンをインストールするかによって異なります)。

- *IBM Communications Server for AIX 移行ガイド* (SC88-6949)

  この資料は、Communications Server for AIX バージョン 4.2 以前のバージョンから IBM Communications Server for AIX バージョン 6 への移行方法を説明しています。

- *IBM Communications Server for AIX 入門* (GC88-6947)

  この資料は IBM Communications Server for AIX の概要を示すもので、サポートされているネットワークの特性、インストール、構成、および操作について説明しています。

- *IBM Communications Server for AIX 管理ガイド* (SC88-6950)

  この資料は、SNA および IBM Communications Server for AIX の概要、および IBM Communications Server for AIX の構成と操作に関する解説です。

- *IBM Communications Server for AIX 管理コマンド解説書* (SD88-6675)

  この資料には、SNA および IBM Communications Server for AIX のコマンドに関する情報が記載されています。

- *IBM Communications Server for Linux or AIX CPI-C プログラマーズ・ガイド* (SC88-5826)

この資料では、「C」または Java™ の熟練したプログラマーを対象として、IBM Communications Server CPI 通信 API を使用する SNA トランザクション・プログラムの作成に関する情報を提供しています。

- *IBM Communications Server for Linux or AIX APPC プログラマーズ・ガイド* (SC88-5825)

  この資料では、拡張プログラム間通信機能 (APPC) を使用するアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for Linux or AIX LUA プログラマーズ・ガイド* (SC88-5827)

  この資料では、従来型 LU アプリケーション・プログラミング・インターフェース (LUA) を使用してアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for Linux or AIX 共通サービス Verb プログラマーズ・ガイド* (SC88-5824)

  この資料では、Common Service Verbs (CSV) アプリケーション・プログラミング・インターフェース (API) を使用してアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for Linux or AIX 管理サービス プログラマーズ・ガイド* (SC88-5829)

  この資料では、Management Services (MS) API を使用してアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for AIX NOF プログラマーズ・ガイド* (SC88-6958)

  この資料では、Node Operator Facility (NOF) API を使用してアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for AIX 診断ガイド* (SC88-6951)

  この資料では、SNA ネットワークの問題解決について説明しています。

- *IBM Communications Server for Linux or AIX APPC アプリケーション・スイート* (SC88-5828)

  この資料には、IBM Communications Server for AIX で使用される APPC アプリケーションに関する情報が記載されています。

- *IBM Communications Server for AIX 用語集* (SC88-6952)

  この資料は、IBM Communications Server for AIX 関連資料で頻繁に使用される用語とその定義を包括的に収録しています。

IBM Communications Server for Linux 関連資料

IBM Communications Server for Linux ライブラリーは、以下の資料により構成されています。なお、これらの資料のソフトコピー版が CD-ROM で提供されています。 CD-ROM 上のソフトコピー・ファイルにアクセスする方法については、「*IBM Communications Server for Linux 入門*」を参照してください。これらのソフトコピ

ー・ブックをシステムにインストールするには、9 〜 15 MB のハードウェア・ディスク・スペースが必要になります (このスペースは、どの各国語バージョンをインストールするかによって異なります)。

- *IBM Communications Server for Linux 入門* (GC88-9996 および GC88-9997)

  この資料は IBM Communications Server for Linux の概要を示すもので、サポートされているネットワークの特性、インストール、構成、および操作について説明しています。この資料には、次の 2 つのバージョンがあります。
  GC88-9996 は i686、x86_64、および ppc64 プラットフォーム上の IBM Communications Server for Linux 用です。
  GC88-9997 は、IBM Communications Server for Linux on System z 用です。

- *IBM Communications Server for Linux 管理ガイド* (SC88-9999)

  この資料は、SNA および IBM Communications Server for Linux の概要、および IBM Communications Server for Linux の構成と操作に関する解説です。

- *IBM Communications Server for Linux 管理コマンド解説書* (SC88-9998)

  この資料には、SNA および IBM Communications Server for Linux のコマンドに関する情報が記載されています。

- *IBM Communications Server for Linux or AIX CPI-C プログラマーズ・ガイド* (SC88-5826)

  この資料では、「C」または Java の熟練したプログラマーを対象として、IBM Communications Server CPI 通信 API を使用する SNA トランザクション・プログラムの作成に関する情報を提供しています。

- *IBM Communications Server for Linux or AIX APPC プログラマーズ・ガイド* (SC88-5825)

  この資料では、拡張プログラム間通信機能 (APPC) を使用するアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for Linux or AIX LUA プログラマーズ・ガイド* (SC88-5827)

  この資料では、従来型 LU アプリケーション・プログラミング・インターフェース (LUA) を使用してアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for Linux or AIX 共通サービス Verb プログラマーズ・ガイド* (SC88-5824)

  この資料では、Common Service Verbs (CSV) アプリケーション・プログラミング・インターフェース (API) を使用してアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for Linux or AIX 管理サービス プログラマーズ・ガイド* (SC88-5829)

  この資料では、Management Services (MS) API を使用してアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for Linux NOF プログラマーズ・ガイド* (SC88-8591)

この資料では、Node Operator Facility (NOF) API を使用してアプリケーション・プログラムを作成するために必要な情報を記載しています。

- *IBM Communications Server for Linux 診断ガイド* (GC88-8601)

  この資料では、SNA ネットワークの問題解決について説明しています。

- *IBM Communications Server for Linux or AIX APPC アプリケーション・スイート* (SC88-5828)

  この資料には、IBM Communications Server for Linux で使用される APPC アプリケーションに関する情報が記載されています。

- *IBM Communications Server for Linux 用語集* (GC88-8602)

  この資料は、IBM Communications Server for Linux 関連資料で頻繁に使用される用語とその定義を包括的に収録しています。

システム・ネットワーク体系 (SNA) 関連資料

次の資料では、SNA ネットワークについての情報を記載しています。

- *Systems Network Architecture: Format and Protocol Reference Manual—Architecture Logic for LU Type 6.2* (英文番号 SC30-3269)
- *Systems Network Architecture: Formats* (英文番号 GA27-3136)
- *Systems Network Architecture: Guide to SNA Publications* (英文番号 GC30-3438)
- *Systems Network Architecture: Network Product Formats* (英文番号 LY43-0081)
- *Systems Network Architecture: Technical Overview* (英文番号 GC30-3073)
- *Systems Network Architecture: APPN Architecture Reference* (英文番号 SC30-3422)
- *Systems Network Architecture: Sessions between Logical Units* (英文番号 GC20-1868)
- *Systems Network Architecture: LU6.2 Reference—Peer Protocols* (英文番号 SC31-6808)
- *Systems Network Architecture: Transaction Programmer's Reference Manual for LU Type 6.2* (英文番号 GC30-3084)
- *IBM 3270 情報表示システム データストリーム プログラマー用 解説書* (邦文番号 N:GA23-0059: 英文番号 GA23-0059)
- *Networking Blueprint Executive Overview* (英文番号 GC31-7057)
- *Systems Network Architecture: Management Services Reference* (英文番号 SC30-3346)

APPC 関連資料

次の資料では、拡張プログラム間通信機能 (APPC) についての情報を記載しています。

- *APPC Application Suite V1 User's Guide* (英文番号 SC31-6532)
- *APPC Application Suite V1 Administration* (英文番号 SC31-6533)
- *APPC Application Suite V1 Programming* (英文番号 SC31-6534)
- *APPC Application Suite V1 Online Product Library* (英文番号 SK2T-2680)

- *APPC Application Suite Licensed Program Specifications* (英文番号 GC31-6535)
- *z/OS V1R2.0 Communications Server: APPC Application Suite ユーザーズ・ガイド* (英文番号 SC31-8809)

プログラミング関連資料

次の資料では、プログラミングについての情報を記載しています。

- *共通プログラミング・インターフェース コミュニケーション・インターフェース CPI-C 解説書* (邦文番号 SC88-7217: 英文番号 SC26-4399)
- *Communications Server for OS/2 Warp 日本語版 32ビット アプリケーション・プログラミングの手引き* (邦文番号SC88-5585: 英文番号SC31-8152)

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

[数字]

A

C

D

F

G

L

M

P

Q

R

S

T

U

V

W

［特殊文字］

プログラム番号: 5765-E51

Printed in Japan

SC88-5825-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12