IBM Communications Server para Linux

Guía del programador para APPC

Versión 6.2.2

SC10-9854-01

IBM Communications Server para Linux

Guía del programador para APPC

Versión 6.2.2

SC10-9854-01

Nota:
Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información de carácter general que figura en el Apéndice E, "Avisos", en la página 305.

Tercera edición (julio de 2006)

Esta publicación es la traducción del original inglés *IBM Communications Server for Linux, APPC Programmer's Guide*, (SC31-6773-01).

Esta edición es aplicable a IBM Communications Server para Linux, Versión 6.2.2 y a todos los releases y modificaciones subsiguientes hasta que se indique lo contrario en nuevas ediciones o boletines técnicos.

Puede solicitar las publicaciones a través del representante local de IBM o de la sucursal de IBM que preste servicios en su localidad. No hay existencias de las publicaciones en la dirección indicada más abajo.

IBM agradece sus comentarios. Al final de esta publicación encontrará una hoja de comentarios del lector. En caso de que el formulario ya se haya utilizado o no esté presente, puede enviar los comentarios a la siguiente dirección:

IBM España, S.A.
National Language Solutions Center
Avda. Diagonal 571, Edif. "L'Illa"
08029 Barcelona
España

Si prefiere enviar sus comentarios de manera electrónica, utilice uno de los métodos siguientes:
- Internet: HOJACOM@es.ibm.com
- Fax: 34-93-321-6134

Al enviar información a IBM, se otorga a IBM un derecho no exclusivo para utilizar o distribuir esa información del modo que IBM considere oportuno, sin incurrir por ello en ninguna obligación con el remitente.

Contenido

Tablas

Figuras

Acerca de este manual

Esta publicación es una guía para desarrollar programas de aplicación de lenguaje C que utilizan APPC (comunicaciones avanzadas programa a programa) para intercambiar datos en un entorno SNA (arquitectura de red de sistemas).

IBM Communications Server para Linux es un producto de software de IBM que hace posible que un sistema que ejecute Linux intercambie información con otros nodos en una red SNA.

Existen dos variantes diferentes de la instalación de IBM Communications Server para Linux, dependiendo del hardware en el que funciona:

Communications Server para Linux
Communications Server para Linux, número de producto programa 5724–i33, funciona en los siguientes sistemas:

- Estaciones de trabajo Intel de 32 bits que ejecutan Linux (i686)
- Estaciones de trabajo AMD64/Intel EM64T de 64 bits que ejecutan Linux (x86_64)
- Sistemas IBM pSeries que ejecutan Linux (ppc64)

Communications Server para Linux sobre System z
Communications Server para Linux sobre System z, número de producto programa 5724–i34, funciona en sistemas principales System z que ejecutan Linux para System z (s390 o s390x).

En este manual, el nombre Communications Server para Linux se utiliza para indicar alguna de estas dos variantes, y el término "sistema Communications Server para Linux" se utiliza para indicar cualquier tipo de sistema que ejecuta Communications Server para Linux, excepto donde se describan las diferencias de manera explícita.

La implementación de APPC de Communications Server para Linux se basa en la implementación de APPC de IBM en sus productos para OS/2 (como por ejemplo **Communications Server para OS/2**), con modificaciones para el entorno Linux.

Los programas desarrollados para utilizar la implementación de APPC de Communications Server para Linux pueden intercambiar datos con los programas desarrollados para utilizar otras implementaciones de APPC compatibles con la arquitectura de unidad lógica (LU) 6.2 SNA.

Este manual se aplica a la versión 6.2.2 de Communications Server para Linux.

A quién va dirigido este manual

Esta publicación va dirigida a expertos programadores de lenguaje C que desarrollan programas de transacciones SNA (arquitectura de red de sistemas) para sistemas que utilizan Communications Server para Linux. Los programadores pueden tener o no experiencia previa con SNA o los recursos de comunicación de Communications Server para Linux.

Administradores del sistema
Los administradores del sistema se encargan de instalar Communications

Server para Linux, configurar el sistema para la conexión en red y llevar a cabo el mantenimiento del sistema. Deben estar familiarizados con el hardware en que funciona Communications Server para Linux y con el sistema operativo Linux. También deben conocer la red a la que el sistema está conectado y entender los conceptos de SNA en general.

Programadores de aplicaciones

Los programadores de aplicaciones diseñan y codifican programas de transacción y de aplicación que utilizan las interfaces de programación de Communications Server para Linux para enviar y recibir datos a través de una red SNA. Deben conocer bien la arquitectura SNA, el programa remoto con el que se comunica el programa de transacción o de aplicación, y los entornos operativos y de programación del sistema operativo AIX o Linux.

Puede ver información más detallada acerca de cómo desarrollar programas de aplicación en el manual de cada una de las API.

Cómo utilizar este manual

Este apartado describe cómo se organiza y presenta la información de este manual.

Organización de este manual

Este manual está organizado de la forma siguiente:

- El Capítulo 1, “Conceptos”, en la página 1 presenta los conceptos básicos de APPC. Está destinado a los programadores que no están familiarizados con APPC.
- El Capítulo 2, “Desarrollo de programas de transacciones”, en la página 27 contiene la información de carácter general que necesita un programador de APPC al desarrollar programas de transacciones (TP).
- El Capítulo 3, “Verbos de control APPC”, en la página 67 describe de forma detallada cada uno de los verbos de control APPC. Cada una de las descripciones incluye la siguiente información: finalidad, bloque de control de verbo (VCB), parámetros suministrados y devueltos, estados de conversación en que puede emitirse el verbo y cambios de estado de conversación que se producen una vez que se ha ejecutado el verbo. Cuando hay diferencias entre las implementaciones de APPC para distintos sistemas operativos, se indican; normalmente se trata de variaciones menores debidas a las diferencias entre sistemas operativos.
- El Capítulo 4, “Verbos de conversación APPC”, en la página 97 describe de forma detallada cada uno de los verbos de conversación APPC. Cada una de las descripciones incluye la siguiente información: finalidad, bloque de control de verbo (VCB), parámetros suministrados y devueltos, estados de conversación en que puede emitirse el verbo y cambios de estado de conversación que se producen una vez que se ha ejecutado el verbo. Cuando hay diferencias entre las implementaciones de APPC para distintos sistemas operativos, se indican; normalmente se trata de variaciones menores debidas a las diferencias entre sistemas operativos.
- El Capítulo 5, “Verbos de servidor de TP”, en la página 253 describe de forma detallada cada uno de los verbos de servidor de TP APPC. Cada una de las descripciones incluye la siguiente información: finalidad, bloque de control de verbo (VCB), parámetros suministrados y devueltos, estados de conversación en que puede emitirse el verbo y cambios de estado de conversación que se producen una vez que se ha ejecutado el verbo. Cuando hay diferencias entre las

implementaciones de APPC para distintos sistemas operativos, se indican; normalmente se trata de variaciones menores debidas a las diferencias entre sistemas operativos.

- El Capítulo 6, “Programas de transacciones de ejemplo”, en la página 271 describe los programas de transacciones APPC de ejemplo, que ilustran el uso de los verbos APPC. Este capítulo también facilita instrucciones para compilar, enlazar y ejecutar los TP en cada uno de los sistemas operativos soportados.
- Apéndice A, “Valores de código de retorno”, en la página 275, lists all the possible return codes in the APPC interface in numerical order and gives their meanings.
- El Apéndice B, “Códigos de retorno comunes”, en la página 283 ofrece información sobre algunos códigos de retorno primarios y secundarios comunes a varios verbos.
- El Apéndice C, “Cambios de estado APPC”, en la página 295 proporciona información sobre los estados de conversación de APPC: qué verbos pueden emitirse en cada uno de los estados y el estado al que cambia la conversación cuando vuelve cada uno de los verbos.
- El Apéndice D, “Soporte de LU 6.2 SNA”, en la página 301 proporciona información de consulta sobre la relación entre la implementación de APPC de Communications Server para Linux y la arquitectura LU 6.2 SNA, así como sobre los verbos de operador de control de LU 6.2 cuya función proporciona en Communications Server para Linux el programa de administración a través de la línea de mandatos **snaadmin** y los verbos NOF (Node Operator Facility).

Convenios tipográficos

La Tabla 1 muestra los estilos tipográficos utilizados en esta publicación.

Tabla 1. Convenios tipográficos

Elemento especial	Ejemplo de tipografía
Título de publicación	*Communications Server para Linux - Guía de administración*
Nombre de vía de acceso o de archivo	**sna_tps**
Programa o aplicación	**snaadmin**
Mandato o programa de utilidad AIX / Linux	**vi**; **define_mode**
Referencia genérica a todos los valores de un tipo determinado	`AP_SEC_BAD_*` (hace referencia a todos los valores de retorno que empiezan con `AP_SEC_BAD`)
Opción o indicador	**-I**
Parámetro o campo de Motif	*primary_rc*; *what_rcvd*
Valor literal o selección que el usuario puede entrar (incluidos los valores por omisión)	`0`; `32.767`
Constante o señal	`AP_DATA_COMPLETE_CONFIRM`
Valor de retorno	`AP_OK`; `AP_SYNC_LEVEL_NOT_SUPPORTED`; `TRUE`
Variable que representa un valor proporcionado	*lParam*; *ReturnedHandle*
Variable de entorno	`LD_RUN_PATH`
Verbo de programación	RECEIVE_ALLOCATE
Datos entrados por el usuario	**cc -I**
Función, llamada o punto de entrada	`APPC_Async`; `WinAsyncAPPC`
Estructura de datos	`WAPPCDATA`
Valor hexadecimal	`0x20`

Convenciones gráficas

Este símbolo sirve para indicar el comienzo de una sección de texto que corresponde únicamente al sistema operativo AIX o Linux. Se aplica a servidores Linux y al IBM Remote API Client que se ejecuta sobre AIX, Linux, Linux para pSeries o Linux para System z.

Este símbolo sirve para indicar el comienzo de una sección de texto que corresponde a IBM Remote API Client sobre Windows.

Este símbolo indica el final de una sección de texto específica del sistema operativo. La información que se encuentra después de este símbolo es válida independientemente de cuál sea el sistema operativo.

Publicaciones relacionadas

Si desea obtener información sobre la arquitectura de SNA o LU 6.2, consulte las publicaciones siguientes de IBM:

- *IBM Systems Network Architecture:*
  - *Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*, SC30-3269
  - *Formats*, GA27-3136
  - *Introduction to APPC*, GG24-1584
  - *Technical Overview*, GC30-3073
  - *Transaction Programmer's Reference Manual for LU Type 6.2*, GC30-3084

Capítulo 1. Conceptos

Este capítulo presenta los conceptos básicos de la comunicación avanzada programa a programa (APPC) en un entorno de proceso distribuido:

- ¿Qué es APPC?
- Conversación correlacionada sencilla
- Proceso de confirmación
- Envío y recepción de información de estado con datos
- Estados de conversación
- Cambio de los estados de conversación
- Llamadas APPC síncronas y asíncronas
- Recepción asíncrona de datos

AIX, LINUX

- Soporte de punto de sincronización

- APPC y CPI-C (interfaz común de programación para comunicaciones)
- API de servidor de TP

¿Qué es APPC?

APPC (comunicación avanzada programa a programa) es una interfaz de programación de aplicaciones (API) que permite establecer comunicaciones de igual a igual entre programas en un entorno SNA (arquitectura de red de sistemas).

A través de APPC, los programas de aplicación distribuidos a través de una red pueden trabajar juntos, comunicarse unos con otros e intercambiar datos para efectuar una sola tarea de proceso como las siguientes:

- Consultar una base de datos remota.
- Copiar un archivo remoto.
- Enviar o recibir correo electrónico.

Una secuencia de comunicaciones completa entre dos programas de aplicación, que pueden efectuar una o más tareas de proceso, se denomina conversación.

Dos aplicaciones APPC que se comunican pueden estar en la misma máquina o en dos distintas; no es necesario que una aplicación conozca la ubicación de su aplicación asociada. Una aplicación APPC puede ejecutarse en un sistema servidor o cliente.

Programas de transacciones

Una transacción es una tarea de proceso que efectúan los programas que utilizan APPC. Así pues, los programas que utilizan APPC se denominan programas de transacciones (TP). Estos programas se comunican como iguales y no según una estructura jerárquica. Los TP de aplicación efectúan tareas para usuarios finales. Los TP de servicio proporcionan servicios a otros programas.

Juntos, los TP distribuidos a través de una red de área local o amplia realizan un proceso de transacciones distribuido.

Comunicación entre TP

Para que dos TP puedan comunicarse son necesarios muchos elementos de hardware y software en el entorno SNA. La Figura 1 ilustra los elementos que atañen directamente a los programadores que desarrollan TP.

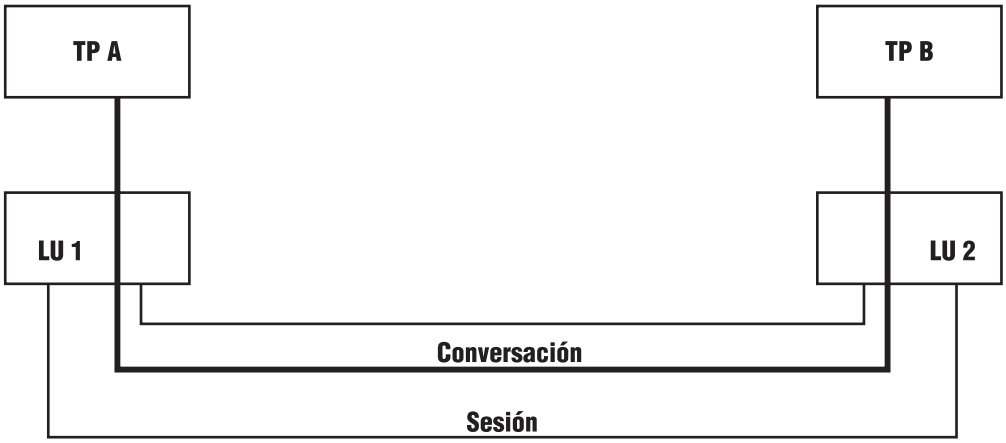

Figura 1. Elementos para desarrollar TP

Unidad lógica 6.2

Cada TP está asociado a una unidad lógica (LU) de un nodo local Communications Server para Linux concreto y accede a la red a través de esta LU. Varios TP pueden estar asociados a la misma LU. Cada tipo de LU utiliza un protocolo específico. APPC está soportado por LU 6.2.

Sesiones

Para que dos TP puedan comunicarse, las LU deben estar conectadas a través de una sesión LU-LU. Una sesión LU-LU es una conexión lógica entre las dos LU. La modalidad de la sesión (un conjunto de características de red) determina cómo se mueven los datos entre las dos LU.

Conversaciones

La comunicación entre los dos TP se produce como una conversación dentro de la sesión LU-LU. Un TP puede participar simultáneamente en varias conversaciones.

Verbos APPC

Un TP accede a APPC mediante verbos APPC. Los TP utilizan estos verbos para indicar a APPC que inicie una conversación, envíe o reciba datos y finalice una conversación. Cada verbo proporciona parámetros a APPC, que realiza la función deseada y devuelve los parámetros al TP. Algunos verbos finalizan rápidamente, después de algún proceso local (por ejemplo, enviar una pequeña cantidad de datos); otros verbos requieren algún tiempo para finalizar, según el TP asociado y la vía de acceso de comunicaciones (por ejemplo, esperar a recibir datos o una confirmación del TP asociado).

El TP que emite el verbo se denomina TP local y el otro TP, TP asociado.

Del mismo modo, la LU que da servicio al TP local es la LU local y la LU que da servicio al TP asociado es la LU asociada.

Los TP y las LU que residen en otros nodos de la red también se denominan TP remotos y LU remotas.

Proceso de la conversación

Una conversación se inicia cuando se producen las dos condiciones siguientes:

1. Un TP (el TP que invoca) indica a APPC que inicie otro TP (el TP invocado) y asigne una conversación entre los dos TP.
2. El TP invocado notifica a APPC que ya está listo para comunicarse con el TP que invoca.

Durante la conversación, los dos TP intercambian información de estado y datos de aplicación.

Una conversación finaliza cuando un TP indica a APPC que desasigne la conversación.

Tipos de conversación

Una conversación puede ser correlacionada o básica. En el momento de la asignación, el TP que invoca especifica si una conversación será básica o correlacionada. Algunos verbos APPC se utilizan sólo en conversaciones correlacionadas, y otros en conversaciones básicas. No se puede utilizar un verbo de conversación básica en una conversación correlacionada o viceversa.

En general, los TP de aplicación utilizan las conversaciones correlacionadas. Los TP de aplicación son programas que efectúan tareas para usuarios finales. Las conversaciones correlacionadas son menos complejas que las conversaciones básicas. En una conversación correlacionada, el TP emisor envía un registro cada vez y el TP receptor recibe un registro cada vez.

Los TP de servicio normalmente utilizan las conversaciones básicas. Los TP de servicio son programas que proporcionan servicios a otros programas locales. Las conversaciones básicas ofrecen un alto grado de control sobre la transmisión y el manejo de datos a un programador de LU 6.2 experimentado. Para ver más información, consulte “Conversaciones básicas” en la página 62.

Conversaciones múltiples

Un TP puede participar simultáneamente en varias conversaciones. Cada conversación requiere una sesión LU-LU.

Las conversaciones múltiples habitualmente se utilizan para que un TP invocado invoque otro TP que, a su vez, invoca otro TP, y así sucesivamente. La Figura 2 en la página 4 muestra cómo el TP A invoca el TP B, y el TP B después invoca el TP C. Los TP A y C no mantienen una conversación el uno con el otro, sino sólo con el TP B.

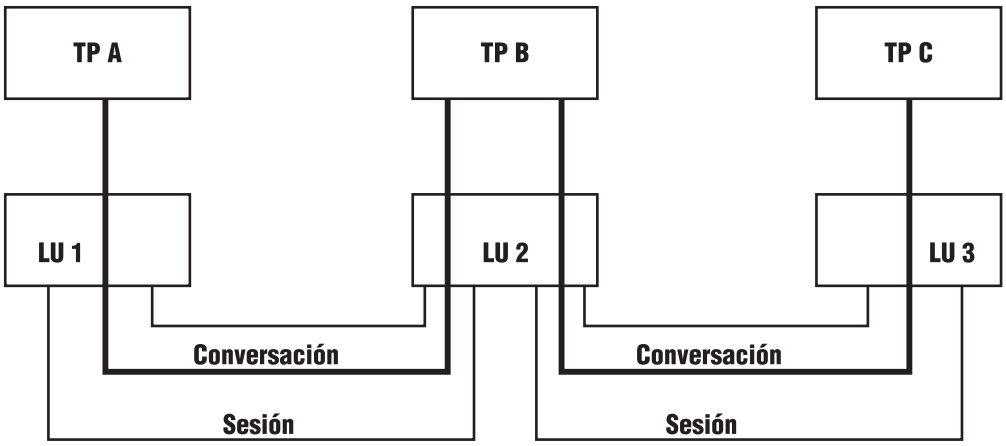

Figura 2. Invocación de TP utilizando conversaciones múltiples

Conversaciones semidúplex y dúplex

Según el modo en que necesitan interactuar los dos TP de una conversación, la conversación puede funcionar de dos modos.

Semidúplex

En una conversación semidúplex, se pasa el control entre los dos TP de modo que un TP siempre tiene el control de la conversación en cualquier momento. El TP que tiene el control puede enviar datos o puede pasar el control al otro TP. El otro TP puede recibir datos pero no puede enviarlos; no obstante, puede enviar una indicación de error al TP que tiene el control o puede solicitar el control de la conversación.

Las conversaciones semidúplex se utilizan normalmente en las siguientes situaciones:

- Cuando el progreso de la conversación depende de los datos que se están transmitiendo; por ejemplo, si el primer TP envía una petición que el segundo TP debe procesar y la siguiente operación de los dos TP depende de si la petición se acepta o se rechaza.
- Cuando el primer TP simplemente envía datos al segundo TP y no necesita recibir ninguna respuesta.

Puesto que se pasa el control entre los dos TP, las conversaciones semidúplex permiten que los TP confirmen cada fase de la conversación antes de continuar. Después de enviar una determinada cantidad de datos, el primer TP puede solicitar una confirmación de que los datos se han recibido y procesado satisfactoriamente antes de continuar. Llegados a este punto, el segundo TP puede confirmar los datos y permitir que el primer TP envíe más o puede devolver una respuesta de error y, a continuación, tomar el control de la conversación (para proporcionar más detalles del error o para devolver una versión corregida de los datos, por ejemplo).

Dúplex

En una conversación dúplex, los dos TP pueden enviar datos en cualquier momento; no se considera que ningún TP tenga el control.

Las conversaciones dúplex se utilizan normalmente cuando los datos que se envían en las dos direcciones son independientes, de modo que el progreso de la conversación no depende de los datos que se están transmitiendo. Puesto que no se considera que ninguno de los dos TP

tenga el control de la conversación el proceso de confirmación no recibe soporte; un TP puede enviar una respuesta de error en cualquier momento, pero esto no impide que el otro TP siga enviando datos.

El TP que asigna la conversación especifica si la conversación funcionará en dúplex o en semidúplex. Esta elección se aplicará durante toda la conversación; no es posible pasar de dúplex a semidúplex y viceversa durante la conversación. Para emitir un verbo en una conversación dúplex, el TP debe establecer una opción adicional en el parámetro *opext* del verbo, tal como se describe en el Capítulo 4, "Verbos de conversación APPC", en la página 97.

No todas las implementaciones de APPC permiten el funcionamiento en dúplex. Si el TP remoto utiliza una implementación de APPC que no permite el funcionamiento dúplex, los TP sólo podrán funcionar en modalidad semidúplex.

Los apartados siguientes describen el funcionamiento de las conversaciones semidúplex. Puesto que las conversaciones dúplex no incluyen los verbos utilizados para el proceso de confirmación y para pasar el control entre los TP, hay partes de la información de estos apartados que no son válidas para las conversaciones dúplex. Si desea ver un resumen del funcionamiento de una conversación dúplex, consulte el apartado "Conversaciones dúplex" en la página 15.

Conversación correlacionada sencilla (semidúplex)

La Tabla 2 ofrece un ejemplo de una conversación correlacionada sencilla en que se muestran los verbos APPC utilizados para iniciar una conversación, intercambiar datos y finalizar la conversación. La flecha indica el flujo de datos. Todos los valores de *primary_rc* son `AP_OK` a menos que se muestre lo contrario.

Tabla 2. Una conversación correlacionada sencilla

TP que invoca	Flujo	TP invocado
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

Los caracteres MC_ al principio de muchos de los verbos son las iniciales de Mapped Conversation (conversación correlacionada). Los parámetros y los resultados de los verbos APPC figuran entre paréntesis.

Inicio de una conversación

Para iniciar una conversación el TP que invoca emite los verbos siguientes:

- TP_STARTED, que identifica la aplicación para APPC como un TP que invoca.
- MC_ALLOCATE, que solicita que APPC establezca una conversación entre el TP que invoca y el TP invocado.

El TP invocado emite el verbo RECEIVE_ALLOCATE, que informa a APPC de que el TP invocado ya está listo para empezar una conversación. Communications Server para Linux asocia el verbo RECEIVE_ALLOCATE con el verbo MC_ALLOCATE emitido por el TP que invoca para establecer la conversación entre los dos TP.

Envío de datos

El verbo MC_SEND_DATA proporciona datos de aplicación que deben transmitirse al TP asociado. Estos datos se retienen en el almacenamiento intermedio de envío de la LU local; no se transmiten al TP asociado hasta que se produce alguno de los sucesos siguientes:

- El almacenamiento intermedio de envío se llena.
- El TP emite un verbo que fuerza a APPC a vaciar el almacenamiento intermedio y enviar los datos al TP asociado.

Además de los datos, el almacenamiento intermedio de envío también contiene la petición MC_ALLOCATE (que precede a los datos). En este ejemplo, el verbo MC_DEALLOCATE vacía el almacenamiento intermedio, con lo que transmite la petición MC_ALLOCATE y los datos al TP asociado. Otros verbos que vacían el almacenamiento intermedio son MC_CONFIRM y MC_FLUSH.

Recepción de datos

El verbo MC_RECEIVE_AND_WAIT recibe datos del TP asociado. Si actualmente no hay datos disponibles, el TP local espera a que lleguen.

Además de recibir datos, el verbo puede recibir un indicador de estado del TP asociado (como por ejemplo una indicación de que la conversación está finalizando, una petición de confirmación de la recepción de datos, etc.). Para obtener más información sobre cómo el TP maneja estos indicadores, consulte los apartados “Proceso de confirmación (semidúplex)” en la página 7 y “Estados de conversación (semidúplex)” en la página 11.

En el ejemplo, el TP invocado emite el primer verbo MC_RECEIVE_AND_WAIT para recibir datos. Cuando ha acabado de recibir el registro de datos completo (*what_rcvd*=AP_DATA_COMPLETE), vuelve a emitir el verbo MC_RECEIVE_AND_WAIT para recibir un código de retorno. El código de retorno AP_DEALLOC_NORMAL indica que se ha desasignado la conversación.

El verbo MC_RECEIVE_IMMEDIATE realiza la misma función que el verbo MC_RECEIVE_AND_WAIT, excepto que no espera si no hay datos disponibles actualmente del TP asociado. En su lugar, devuelve una respuesta que informa de que no hay datos disponibles al TP que llama.

Finalización de una conversación

Para finalizar una conversación, uno de los TP emite el verbo MC_DEALLOCATE, que hace que APPC desasigne la conversación entre los dos TP.

Una vez desasignada la conversación, cada TP emite otro verbo [MC_]ALLOCATE o RECEIVE_ALLOCATE para iniciar otra conversación (con este u otro TP asociado), o bien emite el verbo TP_ENDED.

Un TP puede participar en varias conversaciones simultáneamente. En este caso, el TP emite el verbo TP_ENDED cuando se han desasignado todas las conversaciones.

Proceso de confirmación (semidúplex)

Mediante el proceso de confirmación, un TP envía una petición de confirmación con los datos y el TP receptor confirma la recepción de los datos o indica que ha ocurrido un error. Cada vez que los dos TP intercambian una petición de confirmación y una respuesta, se sincronizan.

El proceso de confirmación de ejemplo de la Tabla 3 muestra dos modos de confirmar la transferencia de datos: solicitar la confirmación después de enviar los datos (utilizando el verbo CONFIRM) y solicitar la confirmación al final de una transacción (solicitando la confirmación en el verbo DEALLOCATE). También puede solicitarse la confirmación en el verbo PREPARE_TO_RECEIVE; éste pide al TP asociado que confirme la recepción de datos y después empiece a enviar datos. Para ver más información, consulte “Cambios de estado” en la página 12. Un par de TP puede elegir utilizar sólo uno de estos mecanismos; en el ejemplo siguiente, el TP invocado utiliza el verbo PREPARE_TO_RECEIVE sin solicitar confirmación; simplemente indica al TP asociado que envíe datos.

Tabla 3. Proceso de confirmación

TP que invoca	Flujo	TP invocado
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 vuelve, *primary_rc*=AP_PROG_ERROR_PURGING)		
		MC_PREPARE_TO_RECEIVE (*ptr_type*=AP_FLUSH)
	<—	
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)		

Tabla 3. Proceso de confirmación (continuación)

TP que invoca	Flujo	TP invocado
	—>	
		MC_RECEIVE_AND_WAIT (*primary_rc*=AP_OK) (*what_rcvd*=AP_CONFIRM_DEALLOCATE)
		MC_CONFIRMED
	<—	
TP_ENDED		TP_ENDED

Establecimiento del nivel de sincronización

El parámetro *sync_level* del verbo MC_ALLOCATE determina el nivel de sincronización de la conversación. Los valores posibles de los niveles de sincronización son:

- AP_CONFIRM_SYNC_LEVEL, en el que los TP pueden solicitar la confirmación de la recepción de datos y responder a estas peticiones.
- AP_NONE, en el que no se produce el proceso de confirmación.

AIX, LINUX

También puede utilizarse un tercer nivel, AP_SYNCPT (punto de sincronización), pero se necesita software adicional. Para ver más información, consulte "Soporte de punto de sincronización" en la página 23.

Envío de una petición de confirmación

El verbo MC_CONFIRM tiene dos efectos:

- Vaciar el almacenamiento intermedio de envío de la LU local, con lo que se envían todos los datos contenidos en el almacenamiento intermedio al TP asociado.
- Enviar una petición de confirmación, que el TP asociado recibe mediante el parámetro *what_rcvd* de un verbo de recepción.

El verbo MC_CONFIRM no finaliza hasta que se recibe la confirmación (o una indicación de que se ha detectado un error) del TP asociado.

Recepción de datos y una petición de confirmación

El parámetro *what_rcvd* del verbo MC_RECEIVE_AND_WAIT indica lo siguiente:

- Estado de los datos recibidos (completo o incompleto).
- Proceso futuro esperado del TP local (por ejemplo, una petición de confirmación o una indicación de que debe empezar a enviar datos).

Cuando el TP invocado termina de recibir el registro de datos completo (*what_rcvd*=AP_DATA_COMPLETE), vuelve a emitir el verbo MC_RECEIVE_AND_WAIT y recibe una petición de confirmación (*what_rcvd*=AP_CONFIRM).

Respuesta a la petición de confirmación

El TP invocado normalmente emite el verbo MC_CONFIRMED para confirmar la recepción de datos; esto libera el TP que invoca para reanudar el proceso.

En cambio, si el TP invocado ha detectado un error en los datos recibidos, puede emitir el verbo MC_SEND_ERROR para indicar esta condición de error.

Desasignación de la conversación

El verbo MC_DEALLOCATE envía una petición de confirmación con los datos cuando se cumplen las dos condiciones siguientes:

- El nivel de sincronización de la conversación (establecido por el parámetro *sync_level* del verbo MC_ALLOCATE) está definido en `AP_CONFIRM_SYNC_LEVEL`.
- El parámetro *dealloc_type* del verbo MC_DEALLOCATE está definido en `AP_SYNC_LEVEL`.

El parámetro *what_rcvd* del último verbo MC_RECEIVE_AND_WAIT emitido por el TP invocado tiene el valor `AP_CONFIRM_DEALLOCATE`, que indica que se requiere una confirmación de recepción de datos para que APPC desasigne la conversación. El TP que invoca espera esta confirmación hasta que el TP invocado emite el verbo MC_CONFIRMED para indicar que los datos se han recibido de forma correcta (o en su lugar podría emitir el verbo MC_SEND_ERROR para indicar que los datos no se han recibido correctamente).

Envío y recepción de información de estado con datos (semidúplex)

En la Tabla 3 en la página 7, el TP que invoca ha utilizado el verbo MC_SEND_DATA para enviar los datos y después el verbo MC_CONFIRM para solicitar la confirmación del TP invocado. Se puede utilizar un parámetro en el verbo [MC_]SEND_DATA para indicar que APPC también debe realizar la función del verbo [MC_]CONFIRM (o [MC_]DEALLOCATE, [MC_]FLUSH o [MC_]PREPARE_TO_RECEIVE) después de enviar los datos, en lugar de tener que emitir dos verbos diferentes.

Del mismo modo, el TP invocado de la Tabla 3 en la página 7 ha emitido dos veces el verbo MC_RECEIVE_AND_WAIT, primero para recibir los datos y después para recibir la información de estado de que el TP que invoca ha solicitado confirmación. Se puede utilizar un parámetro en cualquiera de los verbos [MC_]RECEIVE para indicar que APPC debe devolver la información de estado sobre el mismo verbo de recepción que los datos, en lugar de tener que emitir dos verbos de recepción diferentes.

La Tabla 4 muestra el uso del parámetro de tipo de envío en MC_SEND_DATA para realizar la función del verbo MC_CONFIRM y del parámetro de retorno de estado con datos en MC_RECEIVE_AND_WAIT para recibir la información de estado con datos. Puede suponerse que todos los valores de *primary_rc* son `AP_OK` salvo que se muestre lo contrario.

Tabla 4. Recepción de información de estado con datos

TP que invoca	Flujo	TP invocado
TP_STARTED MC_ALLOCATE MC_SEND_DATA (*type*=`AP_SEND_DATA_CONFIRM`)		

Tabla 4. Recepción de información de estado con datos (continuación)

TP que invoca	Flujo	TP invocado
	—>	
		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

Envío de información de estado con datos

En la Tabla 4 en la página 9, el primer verbo MC_SEND_DATA emitido por el TP que invoca tiene el tipo de envío AP_SEND_DATA_CONFIRM. Esto indica que APPC debe realizar la función del verbo MC_CONFIRM después de enviar los datos. En lugar de CONFIRM, el verbo [MC_]SEND_DATA también puede realizar la función de [MC_]FLUSH, [MC_]DEALLOCATE o [MC_]PREPARE_TO_RECEIVE, o puede especificar (como en el caso del segundo verbo MC_SEND_DATA del ejemplo) que no se debe realizar ninguna función adicional.

Recepción de información de estado con datos

En la Tabla 4 en la página 9, el tercer verbo MC_RECEIVE_AND_WAIT emitido por el TP invocado tiene el parámetro *rtn_status* con el valor AP_YES. Esto indica que si hay disponibles datos seguidos de información de estado, APPC puede devolver la información de estado sobre este verbo además de los datos. El verbo vuelve con un parámetro *what_rcvd* definido en AP_DATA_COMPLETE_CONFIRM, que indica que el TP que invoca ha enviado datos y después ha solicitado confirmación. Los dos primeros verbos MC_RECEIVE_AND_WAIT tienen el parámetro *rtn_status* con el valor AP_NO, por lo que APPC no devuelve información de estado con los datos; el primer verbo recibe los datos y el segundo recibe la información de estado.

Estados de conversación (semidúplex)

En una conversación semidúplex, APPC funciona como un proceso semidúplex, que significa que sólo uno de los dos TP puede enviar datos en cualquier momento. En general, un TP envía cierta cantidad de datos y después lleva a cabo una de las acciones siguientes:

- Solicita al otro TP que confirme la recepción de los datos
- Permite que el otro TP envíe datos

En todo momento, para el TP la conversación se encuentra en un estado de conversación determinado; el estado de conversación rige qué verbos APPC puede emitir un TP en un momento dado. Por ejemplo, un TP no puede emitir el verbo MC_SEND_DATA si la conversación no está en estado Enviar o Enviar-Pendiente. Para ver más información sobre los verbos APPC que pueden emitirse en cada estado, consulte el Apéndice C, “Cambios de estado APPC”, en la página 295.

A continuación se presenta una lista de los posibles estados de conversación:

Confirmar
El TP ha recibido una petición de confirmación de recepción de datos; debe responder positivamente o enviar información de error al TP asociado.

Confirmar-Desasignar
El TP ha recibido una petición de confirmación y debe responder positivamente o enviar información de error. Si el TP responde positivamente, el TP asociado desasigna la conversación.

Confirmar-Enviar
El TP ha recibido una petición de confirmación; debe responder positivamente o enviar información de error. Después de responder, el TP puede empezar a enviar datos.

Pendiente-Envío
El TP recibe los datos de forma asíncrona. El TP puede realizar otro proceso no relacionado con esta conversación. Cuando el TP termina de recibir datos, el estado normalmente es Recibir.

Recibir
El TP puede recibir datos de aplicación e información de estado del TP asociado. Cuando la conversación está en estado Recibir, el TP también puede enviar información de error y solicitar permiso para enviar datos.

Restablecer
La conversación no se ha iniciado o se ha finalizado.

Enviar El TP puede enviar datos al TP asociado y solicitar confirmación. Cuando la conversación está en estado Enviar, el TP también puede empezar a recibir datos, lo que hace que el estado cambie a Recibir.

Enviar-Pendiente
El TP ha recibido datos junto con una indicación SEND del TP asociado. Este estado es parecido al estado Enviar, con la diferencia de que el TP puede utilizar un parámetro adicional en el verbo [MC_]SEND_ERROR para indicar el origen de un error que se ha detectado.

Visión de la conversación por parte del TP

Es la conversación, y no el TP, la que se encuentra en un estado determinado. Un TP puede mantener varias conversaciones, cada una de ellas en un estado diferente. Si se dice que una conversación está en estado Enviar, siempre significa

que desde el punto de vista del TP local la conversación está en estado Enviar. Para el TP asociado, la conversación está en otro estado (como Recibir).

Cambios de estado

Al finalizar un verbo APPC se produce un cambio en el estado de conversación. El cambio de estado puede ser debido a cualquiera de las condiciones siguientes:

- Un verbo emitido por el TP local (por ejemplo, si emite RECEIVE_AND_WAIT en estado Enviar el estado de conversación cambia a Recibir).
- Un verbo emitido por el TP asociado (por ejemplo, si el TP asociado emite el verbo CONFIRM, el TP local recibe una indicación al respecto en uno de los verbos RECEIVE; en este momento la conversación cambia de estado Recibir a Confirmar).
- Una condición de error

El nuevo estado de la conversación depende generalmente del código de retorno primario del verbo APPC completado. Para ver más información, consulte las descripciones de los distintos verbos en el Capítulo 3, “Verbos de control APPC”, en la página 67 y el Capítulo 4, “Verbos de conversación APPC”, en la página 97 o consulte el Apéndice C, “Cambios de estado APPC”, en la página 295.

Comprobaciones de estado

Una comprobación de estado (error de estado) se produce cuando un TP emite un verbo APPC y la conversación no está en el estado apropiado. Por ejemplo, se produciría una comprobación de estado si un TP emitiera el verbo MC_SEND_DATA mientras la conversación estuviera en estado Recibir. Cuando se produce una comprobación de estado, APPC no ejecuta el verbo, sino que devuelve la información de comprobación de estado mediante códigos de retorno primarios y secundarios. Para ver más información sobre los códigos de error de la comprobación de estado que pueden devolverse para cada verbo, consulte las descripciones de los distintos verbos en el Capítulo 3, “Verbos de control APPC”, en la página 67 y el Capítulo 4, “Verbos de conversación APPC”, en la página 97.

Cambio de los estados de conversación (semidúplex)

En la Tabla 5, los estados de conversación aparecen en los márgenes derecho e izquierdo. Esta tabla muestra cómo los verbos APPC pueden cambiar el estado de la conversación de Enviar a Recibir y de Recibir a Enviar.

Tabla 5. Utilización de los verbos APPC para cambiar los estados de conversación

Estado	TP que invoca	Flujo	TP invocado	Estado
	TP_STARTED			
Restablecer				
	MC_ALLOCATE(*sync_level*=AP_CONFIRM_SYNC_LEVEL)			
Enviar				
	MC_SEND_DATA MC_PREPARE_TO_RECEIVE (*ptr_type*=AP_SYNC_LEVEL)			
		—>		
				Restablecer
			RECEIVE_ALLOCATE	
				Recibir

Tabla 5. Utilización de los verbos APPC para cambiar los estados de conversación (continuación)

Estado	TP que invoca	Flujo	TP invocado	Estado
			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)	
				Confirmar-Enviar
			MC_CONFIRMED	
				Enviar
		<—		
	(MC_PREPARE_TO_RECEIVE vuelve, *primary_rc*=AP_OK)			
Recibir				
			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)			
Confirmar				
	MC_REQUEST_TO_SEND MC_CONFIRMED			
Recibir				
		—>		
			(MC_CONFIRM vuelve, *primary_rc*=AP_OK, *rts_rcvd*=AP_YES)	
				Enviar
			MC_PREPARE_TO_RECEIVE (*ptr_type*=AP_SYNC_LEVEL)	
		<—		
	MC_RECEIVE_AND_WAIT (*primary_rc*=AP_OK) (*what_rcvd*=AP_CONFIRM_SEND)			
Confirmar-Enviar				
	MC_CONFIRMED			
		—>		
			(MC_PREPARE_TO_RECEIVE vuelve, *primary_rc*=AP_OK)	
				Recibir
Enviar				
	MC_SEND_DATA MC_DEALLOCATE (*dealloc_type*=AP_SYNC_LEVEL)			
		—>		
			MC_RECEIVE_AND_WAIT (*primary_rc* = AP_OK) (*what_rcvd*= AP_DATA_COMPLETE)	

Tabla 5. Utilización de los verbos APPC para cambiar los estados de conversación (continuación)

Estado	TP que invoca	Flujo	TP invocado	Estado
			MC_RECEIVE_AND_WAIT (*primary_rc*=AP_OK) (*what_rcvd*=AP_CONFIRM_DEALLOCATE)	
				Confirmar-Desasignar
			MC_CONFIRMED	
		<—		
			(MC_DEALLOCATE vuelve, *primary_rc*=AP_OK)	
Restablecer				Restablecer
	TP_ENDED		TP_ENDED	

Estados iniciales

Antes de asignarse la conversación, el estado para ambos TP es Restablecer. Después de asignarse la conversación, el estado inicial para el TP que invoca es Enviar y para el TP invocado es Recibir.

Cambio al estado Recibir

El verbo MC_PREPARE_TO_RECEIVE permite a un TP cambiar la conversación del estado Enviar al estado Recibir. Este verbo lleva a cabo lo siguiente:

- Vacía el almacenamiento intermedio de envío de LU local.
- Envía el indicador AP_CONFIRM_SEND al TP asociado mediante el parámetro *what_rcvd* de un verbo de recepción. Este indicador informa al TP asociado de que se espera una respuesta MC_CONFIRMED para que el TP asociado pueda empezar a enviar datos.
- Realiza el proceso de confirmación ya que se cumplen las condiciones siguientes:
  - El nivel de sincronización de la conversación está definido en AP_CONFIRM_SYNC_LEVEL.
  - El parámetro *ptr_type* está definido en AP_SYNC_LEVEL.

Si se emite el verbo MC_RECEIVE_AND_WAIT mientras la conversación está en estado Enviar, también se vacía el almacenamiento intermedio de envío de la LU y se cambia el estado de conversación a Recibir. Esta forma de cambiar el estado de conversación no permite utilizar el proceso de confirmación.

Cambio al estado Enviar

El verbo MC_REQUEST_TO_SEND informa al TP asociado (para el que la conversación está en estado Enviar) de que el TP local (para el que la conversación está en estado Recibir) quiere enviar datos.

Esta petición se comunica al TP asociado mediante el parámetro *rts_rcvd* del verbo MC_CONFIRM. (El parámetro *rts_rcvd* también se devuelve a MC_SEND_DATA y otros verbos.)

Cuando un TP asociado emite el verbo MC_PREPARE_TO_RECEIVE, el estado de conversación cambia a Recibir para el TP asociado, lo que permite que el TP local envíe datos.

La emisión del verbo MC_REQUEST_TO_SEND no cambia el estado de la conversación. Tras recibir una petición de envío, el TP asociado no está obligado a cambiar el estado de conversación, sino que puede hacer caso omiso de la petición.

Conversaciones dúplex

La Tabla 6 ofrece un ejemplo de una conversación dúplex en que se muestran los verbos APPC utilizados para iniciar una conversación, intercambiar datos y finalizar la conversación. La flecha indica el flujo de datos. Todos los valores de *primary_rc* son AP_OK a menos que se muestre lo contrario.

Tabla 6. Conversación dúplex

TP que invoca	Flujo	TP invocado
TP_STARTED MC_ALLOCATE 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		

Los caracteres MC_ al principio de muchos de los verbos son las iniciales de Mapped Conversation (conversación correlacionada). Los parámetros y los resultados de los verbos APPC figuran entre paréntesis.

Inicio de una conversación

Para iniciar una conversación el TP que invoca emite los verbos siguientes:

- TP_STARTED, que identifica la aplicación para APPC como un TP que invoca.
- MC_ALLOCATE, que solicita que APPC establezca una conversación entre el TP que invoca y el TP invocado. Los parámetros del verbo MC_ALLOCATE especifican que la conversación será dúplex.

El TP invocado emite el verbo RECEIVE_ALLOCATE, que informa a APPC de que el TP invocado ya está listo para empezar una conversación. Communications Server para Linux asocia el verbo RECEIVE_ALLOCATE con el verbo MC_ALLOCATE emitido por el TP que invoca para establecer la conversación entre los dos TP. Los parámetros devueltos del verbo RECEIVE_ALLOCATE especifican que la conversación será dúplex.

Nota: Una vez que se haya iniciado una conversación dúplex, el TP deberá enviar una opción adicional en el parámetro *opext* de todos los verbos emitidos en esta conversación para funcionar en modalidad dúplex. Si desea obtener información detallada, consulte las descripciones de los verbos en el Capítulo 4, “Verbos de conversación APPC”, en la página 97.

Envío de datos

El verbo MC_SEND_DATA proporciona datos de aplicación que deben transmitirse al TP asociado. Estos datos se retienen en el almacenamiento intermedio de envío de la LU local; no se transmiten al TP asociado hasta que se produce alguno de los sucesos siguientes:

- El almacenamiento intermedio de envío se llena.
- El TP emite un verbo que fuerza a APPC a vaciar el almacenamiento intermedio y enviar los datos al TP asociado.

Además de los datos, el almacenamiento intermedio de envío también contiene la petición MC_ALLOCATE (que precede a los datos). En este ejemplo, el segundo verbo MC_SEND_DATA del TP que invoca llena el almacenamiento intermedio, lo que fuerza la petición MC_ALLOCATE y que se transmitan datos al TP asociado. Otros verbos que vacían el almacenamiento intermedio son MC_DEALLOCATE y MC_FLUSH.

Puesto que esta conversación es dúplex, los dos TP pueden enviar datos a la vez. En el ejemplo, el TP invocado envía datos antes de recibir datos enviados por el TP que invoca.

Recepción de datos

El verbo MC_RECEIVE_AND_WAIT recibe datos del TP asociado. Si actualmente no hay datos disponibles, el TP local espera a que lleguen.

Además de recibir datos, el verbo puede recibir un indicador de estado del TP asociado (como, por ejemplo, una indicación de que la conversación está finalizando). Para obtener más información sobre cómo el TP maneja estos indicadores, consulte el apartado “Finalización de una conversación”.

En el ejemplo, el TP invocado emite el primer verbo MC_RECEIVE_AND_WAIT para recibir datos y recibe el registro de datos completo (*what_rcvd*= `AP_DATA_COMPLETE`).

El verbo MC_RECEIVE_IMMEDIATE realiza la misma función que el verbo MC_RECEIVE_AND_WAIT, excepto que no espera si no hay datos disponibles actualmente del TP asociado. En su lugar, devuelve una respuesta que informa de que no hay datos disponibles al TP que llama.

Finalización de una conversación

Para finalizar una conversación,uno de los TP emite el verbo MC_DEALLOCATE, que indica que no tiene más datos que enviar. El otro TP recibe el código de retorno `AP_DEALLOC_NORMAL` en un verbo de recepción posterior, lo que indica que la conversación se ha desasignado.

En una conversación semidúplex, el verbo MC_DEALLOCATE hace que APPC desasigne la conversación entre los dos TP, de modo que el otro TP no podrá seguir con la conversación una vez que haya recibido el código de retorno `AP_DEALLOC_NORMAL`. Sin embargo, en una conversación dúplex es posible que el

otro TP aún tenga datos que enviar o que ya haya enviado datos que el primer TP aún no haya recibido. Por este motivo, la conversación no finaliza en este momento, sino que el primer TP funciona en modalidad de sólo recepción, de modo que sigue emitiendo verbos de recepción (pero no puede enviar más datos).

Cuando el segundo TP recibe el código de retorno AP_DEALLOC_NORMAL, pasa a funcionar en modalidad de sólo envío. No puede seguir emitiendo verbos de recepción (porque no habrá más datos que recibir) pero puede seguir enviando datos. En el ejemplo, el TP invocado emite el otro verbo MC_SEND_DATA antes de desasignar la conversación.

Una vez que ambos TP han desasignado la conversación, cada TP emite otro verbo [MC_]ALLOCATE o RECEIVE_ALLOCATE para iniciar otra conversación (con este u otro TP asociado) o bien emite el verbo TP_ENDED.

Un TP puede participar en varias conversaciones simultáneamente. En este caso, el TP emite el verbo TP_ENDED cuando se han desasignado todas las conversaciones.

Estados de conversación

Un TP considera que una conversación dúplex se encuentra en un estado de conversación determinado, del mismo modo que lo hace en el caso de una conversación semidúplex. No obstante, los estados de conversación posibles de una conversación dúplex son distintos a los de una conversación semidúplex, tal como se indica a continuación. Para ver más información sobre los verbos APPC que pueden emitirse en cada estado, consulte el Apéndice C, “Cambios de estado APPC”, en la página 295.

A continuación se presenta una lista de los posibles estados de conversación:

Envío-Recepción
El TP puede enviar datos o información de error y puede recibir datos de aplicación e información de estado del TP asociado.

Sólo recepción
El TP local ha desasignado la conversación. Puede seguir recibiendo datos e información de estado del TP asociado, pero no puede seguir enviando datos.

Sólo envío
El TP remoto ha desasignado la conversación. El TP local puede seguir enviando datos al TP asociado, pero no recibirá más datos y, por lo tanto, no podrá emitir más verbos de recepción.

Restablecer
La conversación no se ha iniciado o se ha finalizado.

Verbos semidúplex que no están permitidos en las conversaciones dúplex

Los siguientes verbos sólo son válidos para las conversaciones semidúplex y no son necesarios para las conversaciones dúplex. Si cualquiera de estos verbos se emite en una conversación dúplex, recibirá un código de retorno de error.

- [MC_]CONFIRM
- [MC_]CONFIRMED
- [MC_]PREPARE_TO_RECEIVE
- [MC_]RECEIVE_AND_POST

- [MC_]REQUEST_TO_SEND
- [MC_]TEST_RTS
- [MC_]TEST_RTS_AND_POST

Envío y recepción de información de datos acelerados

Además de los datos normales enviados utilizando [MC_]SEND_DATA y recibidos utilizando verbos de recepción, es posible que los TP APPC también envíen y reciban datos acelerados SNA. La red SNA maneja estos datos por separado, y es posible que lleguen a su destino antes que los datos normales. Un TP envía datos acelerados utilizando el verbo [MC_]SEND_EXPEDITED_DATA y los recibe utilizando el verbo [MC_]RECEIVE_EXPEDITED_DATA; con los verbos de recepción estándar nunca se devolverán datos acelerados.

No todas las implementaciones de APPC permiten utilizar datos acelerados. Si el TP remoto utiliza una implementación de APPC que no permite utilizar datos acelerados, el TP local no podrá enviarlos ni recibirlos.

Puesto que los datos acelerados y los datos normales fluyen por separado, un TP puede emitir [MC_]SEND_EXPEDITED_DATA o [MC_]RECEIVE_EXPEDITED_DATA en cualquier estado de conversación excepto el estado Restablecer. Los siguientes verbos no provocan ningún cambio de estado.

Llamadas APPC síncronas y asíncronas

AIX, LINUX

En los sistemas AIX o Linux, Communications Server para Linux proporciona dos puntos de entrada posibles a la biblioteca de APPC:

- Un punto de entrada síncrono, `APPC`. Si la aplicación utiliza este punto de entrada, Communications Server para Linux no devolverá el control a la aplicación hasta que finalice el proceso de los verbos.
- Un punto de entrada asíncrono, `APPC_Async`. Si la aplicación utiliza este punto de entrada, Communications Server para Linux devolverá el control a la aplicación inmediatamente. Cuando finaliza el proceso de los verbos, Communications Server para Linux utiliza una rutina callback suministrada por la aplicación para devolver los resultados del proceso de los verbos a la aplicación.

WINDOWS

En los sistemas Windows, la API remota proporciona tres puntos de entrada posibles a la biblioteca de APPC:

- Un punto de entrada síncrono, `APPC`. Si la aplicación utiliza este punto de entrada, la API remota no devolverá el control a la aplicación hasta que finalice el proceso de los verbos.
- Un punto de entrada asíncrono, `WinAsyncAPPC`. Si la aplicación utiliza este punto de entrada, la API remota devolverá el control a la aplicación inmediatamente. Cuando finaliza el proceso de los verbos, la API remota lo indica enviando un mensaje al procedimiento de la ventana de la aplicación.

- Un punto de entrada asíncrono, `WinAsyncAPPCEx`. Cuando finaliza el proceso de los verbos, la API remota lo indica señalando un descriptor de contexto de sucesos proporcionado por la aplicación.

Para ver más información sobre estos puntos de entrada, consulte el Capítulo 2, “Desarrollo de programas de transacciones”, en la página 27.

El uso del punto de entrada asíncrono permite que una aplicación siga con otro proceso mientras espera a que finalice un verbo. La aplicación puede emitir verbos en otras conversaciones APPC, o emitir verbos para iniciar nuevas conversaciones, o puede efectuar otro proceso no relacionado con APPC. No obstante, es posible que los demás verbos emitidos durante la misma conversación se pongan en cola y no se procesen hasta que el verbo pendiente haya finalizado; si desea obtener más información, consulte el apartado “Operación de no bloqueo” en la página 22 más adelante.

La única excepción a esta norma se produce cuando el verbo [MC_]RECEIVE_AND_POST está pendiente. En estos casos, la aplicación puede emitir un conjunto limitado de verbos en la misma conversación. Para ver más información, consulte el apartado siguiente.

Recepción asíncrona de datos

Los verbos APPC MC_RECEIVE_AND_POST y RECEIVE_AND_POST permiten que un TP reciba datos de manera asíncrona, sin tener en cuenta otros sucesos que se producen en el programa. Por lo tanto, el programa puede realizar otras tareas mientras recibe datos.

AIX, LINUX

Para los sistemas AIX o Linux, los parámetros de [MC_]RECEIVE_AND_POST incluyen la dirección de una rutina callback, que APPC utiliza para informar al TP de que se han recibido datos. Esta rutina callback es independiente de la rutina callback suministrada en el punto de entrada asíncrono `APPC`. Se puede emitir [MC_]RECEIVE_AND_POST utilizando el punto de entrada síncrono o asíncrono; APPC utiliza la rutina callback suministrada en los parámetros del verbo para devolver los datos recibidos al TP y sólo utiliza la rutina callback suministrada en el punto de entrada asíncrono si se suministra una dirección nula en el VCB.

WINDOWS

En los sistemas Windows, la finalización del verbo se indica señalando un descriptor de contexto de sucesos proporcionado por la aplicación. El parámetro *sema* contiene un descriptor de contexto de sucesos (obtenido llamando a las funciones `CreateEvent` u `OpenEvent` de Windows), que APPC utiliza para informar al TP de que se han recibido datos.

Recepción asíncrona de datos

La operación de [MC_]RECEIVE_AND_POST es parecida a la operación del verbo [MC_]RECEIVE_AND_WAIT emitido utilizando el punto de entrada asíncrono; el control vuelve a la aplicación de inmediato y los datos solicitados se devuelven posteriormente en la rutina callback. La principal diferencia es que al emitir [MC_]RECEIVE_AND_POST la conversación se coloca en un estado definido, el estado Pendiente-Envío, en el que el TP puede emitir un conjunto limitado de verbos APPC en esta conversación mientras espera a que se llame a la rutina callback. Los verbos que pueden emitirse en estado Pendiente-Envío son:

- GET_TYPE
- [MC_]DEALLOCATE con el tipo de desasignación `AP_ABEND`, `AP_ABEND_PROG`, `AP_ABEND_SVC` o `AP_ABEND_TIMER`
- [MC_]GET_ATTRIBUTES
- [MC_]RECEIVE_EXPEDITED_DATA
- [MC_]REQUEST_TO_SEND
- [MC_]SEND_ERROR
- [MC_]SEND_EXPEDITED_DATA
- [MC_]TEST_RTS
- TP_ENDED

Si la aplicación emite [MC_]RECEIVE_AND_WAIT (o cualquier otro verbo APPC) utilizando el punto de entrada asíncrono, no debe emitir ningún otro verbo APPC en esta conversación hasta que se haya llamado a la rutina callback.

En la Tabla 7, el TP invocado recibe datos de manera asíncrona.

Tabla 7. Recepción asíncrona de datos

Estado	TP que invoca	Flujo	TP invocado	Estado
	TP_STARTED			
Restablecer				
	MC_ALLOCATE			
Enviar				
	MC_FLUSH			
		—>		
				Restablecer
			RECEIVE_ALLOCATE	
				Recibir
			MC_RECEIVE_AND_POST (*primary_rc*=`AP_OK`)	
				Pendiente-Envío
			(El TP lleva a cabo otras tareas o emite verbos como los de petición de envío u obtención de atributos. La mayoría de los otros verbos APPC no pueden utilizarse en este estado de conversación; consulte el Apéndice C, “Cambios de estado APPC”, en la página 295 para ver más información sobre los verbos permitidos).	
	MC_SEND_DATA			
	MC_DEALLOCATE			
Restablecer				
	TP_ENDED			

Tabla 7. Recepción asíncrona de datos (continuación)

Estado	TP que invoca	Flujo	TP invocado	Estado
		—>		
			Los datos se han recibido. APPC llama a la rutina callback suministrada. MC_RECEIVE_AND_POST devuelve *primary_rc*=AP_OK, *what_rcvd*=AP_DATA_COMPLETE	
				Recibir
			(El TP comprueba que la rutina callback ha sido llamada.) MC_RECEIVE_AND_WAIT (*primary_rc*=AP_DEALLOC_NORMAL)	
				Restablecer
			TP_ENDED	

En la Tabla 7 en la página 20, el TP invocado sigue estos pasos para recibir datos de manera asíncrona:

1. Emite el verbo MC_RECEIVE_AND_POST. Uno de los parámetros es la dirección de la rutina callback que APPC llama (en AIX o Linux) o el descriptor de contexto de sucesos que APPC señala (en Windows) cuando se reciben los datos.
2. Verifica que el parámetro *primary_rc* (código de retorno primario) sea AP_OK, lo que indica que el TP ha empezado a recibir datos de manera asíncrona.
3. Realiza tareas no relacionadas con esta conversación mientras recibe datos de manera asíncrona. La mayoría de los verbos APPC no son válidos en este estado de conversación.
4. Espera a que se llame a la rutina callback (en AIX o Linux), o a que se señale el descriptor de contexto de sucesos (en Windows), lo que indica que el TP ha terminado de recibir datos de manera asíncrona.
5. Vuelve a verificar el parámetro *primary_rc* del verbo MC_RECEIVE_AND_POST. El segundo *primary_rc* indica si los datos se han recibido sin error.
6. Verifica que el parámetro *what_rcvd* del verbo MC_RECEIVE_AND_POST sea AP_DATA_COMPLETE.
7. Emite el verbo MC_RECEIVE_AND_WAIT para recibir el indicador de desasignación.

Nota: El verbo [MC_]RECEIVE_AND_POST devuelve los parámetros *primary_rc* y *secondary_rc* dos veces; la primera después de emitir el verbo, para indicar si el verbo ha empezado correctamente a esperar los datos, y la segunda después de haber recibido los datos.

Una vez que el TP invocado emite el verbo MC_RECEIVE_AND_POST y obtiene el parámetro *primary_rc* con el valor AP_OK, la conversación cambia al estado Pendiente-Envío.

Una vez que el TP ha terminado de recibir los datos de manera asíncrona y APPC llama a la rutina callback suministrada (en AIX o Linux) o señala el descriptor de contexto de sucesos proporcionado (en Windows), la conversación cambia al estado Recibir porque el parámetro *what_rcvd* tiene el valor AP_DATA_COMPLETE.

Operación de no bloqueo

Communications Server para Linux admite la operación de no bloqueo a nivel de cola para los verbos de conversación APPC, de modo que un TP puede emitir varios verbos durante la misma conversación sin tener que esperar a que cada verbo finalice. Esto se utiliza normalmente junto con el punto de entrada APPC asíncrono, lo que permite que el TP continúe funcionando aunque el proceso de un verbo anterior no haya finalizado, pero también se puede utilizar con el punto de entrada síncrono en un TP de múltiples subprocesos que emita verbos APPC desde más de un subproceso. Para emitir un verbo en modalidad de no bloqueo, el TP establece una opción en el parámetro *opext* del verbo, tal como se describe en el Capítulo 4, "Verbos de conversación APPC", en la página 97.

Para cada TP y cada conversación, APPC proporciona una serie de colas en que se pueden retener verbos mientras se espera a que se procesen. Cada cola maneja un subconjunto distinto de los verbos APPC válidos, de modo que cada verbo está asociado a una cola distinta.

- Si el TP emite un verbo y ya no hay verbos que se estén procesando para la cola apropiada, el verbo se procesará inmediatamente.
- Si el TP emite un verbo de no bloqueo y ya se está procesando otro verbo para la cola apropiada, el verbo se añadirá a la cola (detrás de todos los demás verbos que ya estén a la espera en la cola apropiada). Se procesará después de que los verbos que ya están en cola hayan finalizado (a excepción de la cola Asignar, tal como se describe a continuación).
- Si el TP emite un verbo de bloqueo y ya se está procesando para la cola apropiada, el verbo se rechazará con un código de retorno de error.

Las colas disponibles para el TP y los verbos que cada cola maneja son los siguientes.

Cola Asignar

Para cada TP activo, hay una sola cola que maneja los siguientes verbos:

- [MC_]ALLOCATE
- [MC_]SEND_CONVERSATION

Se pueden procesar dos o más verbos de esta cola a la vez, por lo que no está garantizado que finalicen en el mismo orden en que se hayan emitido.

Cola Enviar-Recibir (sólo conversaciones semidúplex)

Para cada conversación semidúplex activa, hay una sola cola que maneja los siguientes verbos:

- [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

El verbo [MC_]RECEIVE_AND_POST no se retiene en esta cola. Este verbo no se puede emitir en la modalidad de bloqueo ni en la de no bloqueo si cualquiera de los verbos de recepción ya se está procesando para la conversación.

Cola Enviar (sólo conversaciones dúplex)
Para cada conversación dúplex activa, hay una sola cola que maneja los siguientes verbos:

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

Cola Recibir (sólo conversaciones dúplex)
Para cada conversación dúplex activa, hay una sola cola que maneja los siguientes verbos:

- [MC_]RECEIVE_AND_WAIT
- [MC_]RECEIVE_IMMEDIATE

Cola de envío acelerado
Para cada conversación activa (ya sea dúplex o semidúplex), hay una sola cola que maneja los siguientes verbos:

- [MC_]REQUEST_TO_SEND (sólo conversaciones semidúplex)
- [MC_]SEND_EXPEDITED_DATA

Cola de recepción acelerada
Para cada conversación activa (ya sea dúplex o semidúplex), hay una sola cola que maneja los siguientes verbos:

- [MC_]RECEIVE_EXPEDITED_DATA

Los siguientes verbos de conversación no están asociados a ninguna cola y, por lo tanto, se pueden emitir en cualquier momento independientemente de los verbos que ya estén en cola. La opción de la modalidad de no bloqueo del parámetro *opext* del verbo se pasa por alto.

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

Los verbos de control APPC siempre se emiten en modalidad de bloqueo.

Soporte de punto de sincronización

AIX, LINUX

La API de APPC de Communications Server para Linux proporciona soporte para las sesiones y conversaciones que utilizan protocolos de punto de sincronización de LU 6.2. Esto significa que puede utilizarse junto con los supervisores de transacción que requieren soporte de punto de sincronización de nivel 2. En sí no suministra los componentes necesarios para una completa implementación de punto de sincronización, pero suministra el soporte subyacente para una implementación suministrada por el proveedor. El proveedor debe proporcionar los componentes siguientes:

- SPM (gestor de puntos de sincronización).
- C-PRM (gestor de recursos protegidos por conversación).
- TP de resincronización.

Este manual no intenta explicar las funciones de punto de sincronización, sino que únicamente describe el soporte que proporciona Communications Server para Linux para permitir que se implementen. Si está desarrollando un gestor de puntos de sincronización para utilizarlo con APPC de Communications Server para Linux, ya debe estar familiarizado con los conceptos de punto de sincronización; si es necesario, consulte los manuales acerca de IBM LU 6.2 para obtener más información.

En algunos valores de parámetros y de códigos de retorno de este manual se indica que "sólo son utilizados por los TP que soportan el proceso de punto de sincronización" o que "sólo se utilizan si el parámetro *sync_level* de la conversación es AP_SYNCPT". Si desarrolla aplicaciones APPC que no utilizan funciones de punto de sincronización, no intente utilizar estos parámetros. En la mayoría de los casos, el gestor de puntos de sincronización se encarga de realizar la conversión entre estos valores y las funciones de punto de sincronización apropiadas, como se indica en las descripciones de los parámetros.

Si está desarrollando aplicaciones para trabajar con una implementación de punto de sincronización suministrada por su proveedor de Communications Server para Linux o por otro proveedor, éste debe facilitarle la información adicional necesaria para utilizar la implementación.

APPC y CPI-C

La interfaz de programas de aplicación CPI-C (interfaz común de programación para comunicaciones), otra API de Communications Server para Linux, proporciona muchas de las funciones de APPC pero con un estilo de interfaz diferente.

Mientras que una aplicación APPC establece parámetros en un bloque de control de verbos y después llama un único punto de entrada a APPC con la dirección del bloque, un programa CPI-C llama un punto de entrada diferente para cada verbo y pasa la información necesaria como parámetros en la llamada.

Aunque las interfaces de programación para APPC y CPI-C son diferentes, los datos reales transmitidos entre programas son los mismos. Esto significa que una aplicación CPI-C puede comunicarse con un TP APPC, igual que dos TP APPC o dos aplicaciones CPI-C pueden comunicarse entre sí. No es necesario que el TP APPC sepa si su elemento asociado es un TP APPC o una aplicación CPI-C.

La única restricción a que está sujeto un TP APPC para comunicarse con una aplicación CPI-C es que no debe enviar Parámetros de inicialización de programa (datos PIP) al asignar la conversación, puesto que CPI-C no da soporte a la recepción de datos PIP. Para ver más información sobre los datos PIP, consulte la descripción del verbo [MC]ALLOCATE en el Capítulo 4, "Verbos de conversación APPC", en la página 97.

API de servidor de TP

AIX, LINUX

Los verbos de servidor de TP son una ampliación de la API de APPC para permitir que las aplicaciones participen en el inicio de los TP como respuesta a las peticiones de asignación (peticiones Attach).

Communications Server para Linux proporciona un mecanismo por omisión para iniciar TP automáticamente. Los TP que pueden iniciarse automáticamente se configuran en el archivo **sna_tps**, como se describe en la publicación *Communications Server para Linux - Guía de administración*.

Algunas aplicaciones, como los supervisores de transacción, necesitan más control sobre el inicio de los TP que el que suministra este mecanismo por omisión (como el acceso a la petición de asignación). Las ampliaciones de servidor de TP proporcionan el nivel de soporte que necesitan tales aplicaciones. Para ver más información sobre los servidores de TP, consulte “Desarrollo de servidores de TP” en la página 64.

Capítulo 2. Desarrollo de programas de transacciones

Este capítulo contiene información sobre los siguientes temas y le ayudará a desarrollar programas de transacciones (TP):

- Categorías de verbos APPC
- Resumen de verbos APPC
- Puntos de entrada APPC

AIX, LINUX

- Consideraciones sobre AIX o Linux

WINDOWS

- Consideraciones sobre Windows

- Información de configuración
- Visión general de la seguridad de conversación
- Inicio de TP
- Sesiones LU-LU
- Conversaciones básicas

AIX, LINUX

- Desarrollo de servidores de TP

- Desarrollo de TP portables

Categorías de verbos APPC

Los verbos APPC se clasifican en las siguientes categorías:

- Verbos de control, descritos en el Capítulo 3, "Verbos de control APPC", en la página 67.
- Verbos de conversación, descritos en el Capítulo 4, "Verbos de conversación APPC", en la página 97.

Verbos de control

Los verbos de control inician y finalizan TP, y obtienen información sobre las propiedades de los TP:

- TP_STARTED
- TP_ENDED
- RECEIVE_ALLOCATE
- GET_TP_PROPERTIES

AIX, LINUX

- SET_TP_PROPERTIES
- GET_LU_STATUS

Verbos de conversación

Los verbos de conversación permiten a los TP asignar una conversación, enviar y recibir datos, cambiar estados de conversación y desasignar una conversación.

El siguiente verbo puede emitirse en una conversación básica o correlacionada:

- GET_TYPE

La mayoría de los verbos de conversación pueden clasificarse en uno de los dos grupos siguientes:

- Verbos de conversación correlacionada, que pueden ser emitidos por un TP sólo en una conversación correlacionada.
- Verbos de conversación básica, que pueden ser emitidos por un TP sólo en una conversación básica.

Los verbos de conversación se clasifican por tipo, correlacionado o básico, según se muestra en la Tabla 8:

Tabla 8. Verbos de conversación correlacionada y básica

Verbos de conversación correlacionada	Verbos de conversación básica
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
MC_SEND_EXPEDITED_DATA	SEND_EXPEDITED_DATA
MC_TEST_RTS	TEST_RTS
MC_TEST_RTS_AND_POST	TEST_RTS_AND_POST

Los verbos correlacionados y básicos tienen las mismas funciones en sus respectivos tipos de conversación, pero pueden tener parámetros y códigos de retorno algo distintos.

Verbos de servidor de TP

AIX, LINUX

Los verbos de servidor de TP permiten que las aplicaciones inicien TP como respuesta a las peticiones de Communications Server para Linux:

REGISTER_TP_SERVER
UNREGISTER_TP_SERVER
REGISTER_TP
UNREGISTER_TP
QUERY_ATTACH
ACCEPT_ATTACH
REJECT_ATTACH
ABORT_ATTACH

Nota: Los verbos de servidor de TP, tal como se describe en el Capítulo 5, "Verbos de servidor de TP", en la página 253, se deben emitir utilizando el punto de entrada asíncrono `APPC_Async` y no el punto de entrada síncrono `APPC`.

Resumen de verbos APPC

Este apartado describe brevemente cada uno de los verbos APPC. Los verbos APPC están agrupados por función. (La funcionalidad total de un verbo puede ser más amplia que la que indica este resumen. Para ver una descripción detallada de un verbo concreto, consulte el Capítulo 3, "Verbos de control APPC", en la página 67 o el Capítulo 4, "Verbos de conversación APPC", en la página 97.)

Inicio de una conversación

Los siguientes verbos se utilizan para iniciar una conversación entre dos TP. Para ver más información, consulte "Inicio de TP" en la página 58.

TP_STARTED
Este verbo es emitido por el TP que invoca. Notifica a APPC que el TP que invoca está iniciándose. Cuando se ejecuta correctamente, este verbo devuelve un identificador de TP (*tp_id*) para el TP que invoca.

AIX, LINUX

SET_TP_PROPERTIES
Este verbo es utilizado por un TP para establecer las propiedades relacionadas con el TP local, que después se utilizarán al asignar nuevas conversaciones. Esto permite que el TP especifique lo siguiente:

- Unidad lógica de trabajo (transacción entre los TP APPC para realizar una determinada tarea) con la que se asocia una conversación.
- Identificador de usuario que se utilizará cuando se asigne una nueva conversación y se indique que ya se ha verificado la seguridad de conversación.

MC_ALLOCATE o ALLOCATE
Este verbo es emitido por el TP que invoca. Asigna una sesión entre la LU local y una LU remota y establece una conversación entre el TP que invoca y el TP invocado.

El verbo ALLOCATE puede establecer una conversación básica o correlacionada. El verbo MC_ALLOCATE puede iniciar sólo una conversación correlacionada. Cada verbo puede iniciar una conversación dúplex o semidúplex.

Una vez asignada la conversación, APPC devuelve un identificador de conversación (*conv_id*) mediante este verbo.

MC_SEND_CONVERSATION o SEND_CONVERSATION
Este verbo es emitido por el TP que invoca. Asigna una sesión entre la LU local y una LU remota, establece una conversación entre el TP que invoca y el TP invocado, envía un solo registro de datos por esta conversación y desasigna la conversación.

RECEIVE_ALLOCATE
Este verbo es emitido por el TP invocado. Confirma que el TP invocado está preparado para empezar una conversación con el TP que invoca, que ha emitido el verbo [MC_]ALLOCATE. Una vez ejecutado correctamente, RECEIVE_ALLOCATE devuelve un identificador de TP (*tp_id*) para el TP invocado y el identificador de conversación (*conv_id*).

Envío de datos

Los siguientes verbos envían datos al TP asociado:

MC_SEND_DATA o SEND_DATA
Este verbo coloca datos en el almacenamiento intermedio de envío de la LU local para transmitirlos al TP asociado.

Los datos recopilados en el almacenamiento intermedio de envío de la LU local se transmiten a la LU asociada (y al TP asociado) cuando se produce una de las situaciones siguientes:

- El almacenamiento intermedio de envío se llena.
- El TP local emite un verbo que vacía el almacenamiento intermedio de envío de la LU local, como [MC_]FLUSH o [MC_]CONFIRM. ([MC_]CONFIRM sólo es válido para las conversaciones semidúplex).

El verbo [MC_]SEND_DATA también realiza la función de los verbos [MC_]CONFIRM, [MC_]DEALLOCATE, [MC_]FLUSH o [MC_]PREPARE_TO_RECEIVE. ([MC_]CONFIRM y [MC_]PREPARE_TO_RECEIVE sólo son válidos para las conversaciones semidúplex).

MC_SEND_EXPEDITED_DATA o SEND_EXPEDITED_DATA
Este verbo coloca datos en el almacenamiento intermedio de envío acelerado de la LU local para transmitirlos al TP asociado.

Los datos recopilados en el almacenamiento intermedio de envío de la LU local se transmiten a la LU asociada (y al TP asociado) del mismo modo que en el caso del verbo [MC_]SEND_DATA. No obstante, puesto que los datos se envían por la red como datos acelerados, es posible que lleguen antes que los datos que se han enviado antes utilizando [MC_]SEND_DATA.

MC_FLUSH o FLUSH
Este verbo vacía el almacenamiento intermedio de envío de la LU local y envía el contenido del almacenamiento intermedio a la LU asociada (y al TP). Si el almacenamiento intermedio de envío está vacío, no se efectúa ninguna acción.

MC_CONFIRM o CONFIRM (sólo conversaciones semidúplex)
Este verbo envía tanto el contenido del almacenamiento intermedio de envío de la LU local como una petición de confirmación al TP asociado.

MC_PREPARE_TO_RECEIVE o PREPARE_TO_RECEIVE (sólo conversaciones semidúplex)
Este verbo cambia el estado de la conversación de Enviar a Recibir. Antes de cambiar el estado de conversación, este verbo realiza el equivalente del verbo [MC_]FLUSH o [MC_]CONFIRM. Una vez que este verbo se ha ejecutado correctamente, el TP puede recibir datos.

MC_REQUEST_TO_SEND o REQUEST_TO_SEND (sólo conversaciones semidúplex)
Este verbo informa al TP asociado de que el TP local quiere enviar datos. Si el TP asociado emite el verbo [MC_]PREPARE_TO_RECEIVE o [MC_]RECEIVE_AND_WAIT, el estado de conversación cambia a Recibir para el TP asociado, lo que permite que el TP local empiece a enviar datos.

Recepción de datos

Los siguientes verbos permiten a un TP recibir datos de su TP asociado:

MC_RECEIVE_AND_WAIT o RECEIVE_AND_WAIT
Al emitir este verbo mientras la conversación está en estado Recibir, el TP local recibe del TP asociado los datos que están disponibles en este momento. Si no hay datos disponibles, el TP local espera a que lleguen.

Al emitir este verbo mientras la conversación está en estado Enviar, se vacía el almacenamiento intermedio de envío de la LU y se cambia el estado de conversación a Recibir. A continuación, el TP local empieza a recibir datos.

MC_RECEIVE_AND_POST o RECEIVE_AND_POST
Al emitir este verbo mientras la conversación está en estado Recibir, el estado de conversación cambia a Pendiente-Envío y el TP local recibe datos de manera asíncrona. De esta forma el TP local puede seguir procesando mientras llegan datos a la LU local.

Al emitir este verbo mientras la conversación está en estado Enviar, se vacía el almacenamiento intermedio de envío de la LU y se cambia el estado de conversación a Pendiente-Envío. A continuación, el TP local empieza a recibir datos de manera asíncrona.

MC_RECEIVE_IMMEDIATE o RECEIVE_IMMEDIATE
Este verbo recibe los datos que están disponibles actualmente del TP asociado. Si no hay datos disponibles, el TP local no espera. A diferencia de los otros verbos RECEIVE, este verbo se emite sólo en estado Recibir y no en estado Enviar.

MC_RECEIVE_EXPEDITED_DATA o RECEIVE_EXPEDITED_DATA
Este verbo recibe los datos acelerados que están disponibles actualmente del TP asociado. Si no hay datos disponibles, el verbo puede volver inmediatamente o esperar hasta que lleguen datos.

Confirmación de recepción de datos o información de errores

Los siguientes verbos confirman la recepción de datos o informan de un error:

MC_CONFIRMED o CONFIRMED
Este verbo responde a una petición de confirmación procedente del TP

asociado. Informa al TP asociado de que el TP local ha recibido y procesado los datos sin producirse ningún error.

MC_SEND_ERROR o SEND_ERROR
Este verbo notifica al TP asociado que el TP local ha encontrado un error a nivel de aplicación.

Obtención de información

Los siguientes verbos proporcionan información a los TP:

MC_GET_ATTRIBUTES o GET_ATTRIBUTES
Un TP utiliza este verbo para obtener los atributos de la conversación.

GET_TYPE
Un TP utiliza este verbo para determinar el tipo de una conversación determinada (básica o correlacionada) y si la conversación funciona en dúplex o semidúplex. Con esta información, el TP puede determinar los verbos correctos que se deben emitir en esta conversación.

AIX, LINUX

GET_LU_STATUS
Un TP utiliza este verbo para obtener información sobre el número de sesiones entre su LU local y una LU asociada especificada, y sobre si el número de sesiones ha disminuido a `0` (cero) en algún momento desde que se ha emitido por última vez el verbo. Esto permite que el TP verifique si ha perdido la conectividad con su TP asociado (en cuyo caso puede que tenga que volver a sincronizar).

GET_TP_PROPERTIES
Un TP utiliza este verbo para obtener información sobre los atributos del TP local y la unidad lógica de trabajo (una transacción entre TP APPC para efectuar una tarea determinada) en que el TP participa.

MC_TEST_RTS o TEST_RTS
Este verbo determina si se ha recibido una notificación REQUEST_TO_SEND del TP asociado.

MC_TEST_RTS_AND_POST o TEST_RTS_AND_POST
Este verbo notifica de manera asíncrona a la aplicación la recepción de una notificación REQUEST_TO_SEND del TP asociado.

Finalización de una conversación

Tanto el TP invocado como el que invoca pueden finalizar una conversación. Los siguientes verbos finalizan una conversación:

MC_DEALLOCATE o DEALLOCATE
Este verbo desasigna una conversación entre dos TP. Antes de desasignar la conversación, este verbo ejecuta el equivalente del verbo [MC_]FLUSH o [MC_]CONFIRM.

TP_ENDED
Este verbo es emitido por los TP que invocan y los TP invocados. Notifica a APPC que el TP está finalizando. Al emitir este verbo también se finalizan las demás conversaciones que están activas.

Inicio de un programa de transacciones (TP)

AIX, LINUX

Los siguientes verbos se utilizan para permitir que una aplicación participe en el proceso de carga del TP de Communications Server para Linux.

REGISTER_TP_SERVER
Un TP utiliza este verbo para notificar a Communications Server para Linux que la aplicación ya puede iniciar automáticamente programas de transacciones (TP).

REGISTER_TP
Este verbo se utiliza para registrar en Communications Server para Linux el nombre de un TP cuyas peticiones de inicio de TP (peticiones Attach) deben ser manejadas por la aplicación.

QUERY_ATTACH
La aplicación utiliza este verbo para determinar los parámetros de la petición para iniciar un TP, de modo que la aplicación pueda decidir si debe iniciar el TP.

ACCEPT_ATTACH
Este verbo se utiliza para notificar a Communications Server para Linux que la aplicación tiene intención de iniciar el TP que corresponde a esta petición Attach.

REJECT_ATTACH
Este verbo se utiliza para notificar a Communications Server para Linux que la aplicación no tiene intención de iniciar el TP que corresponde a esta petición Attach.

ABORT_ATTACH
Este verbo se utiliza para que este servidor de TP finalice el proceso de la petición Attach después de que se haya aceptado la petición Attach con un verbo ACCEPT_ATTACH porque el servidor de TP o el TP ha detectado un error durante el proceso posterior.

UNREGISTER_TP
Este verbo se utiliza para notificar a Communications Server para Linux que la aplicación ya no desea procesar peticiones Attach para este TP registrado previamente.

UNREGISTER_TP_SERVER
Este verbo se utiliza para notificar a Communications Server para Linux que la aplicación no quiere recibir notificaciones de petición Attach para el TP especificado.

Puntos de entrada APPC: Sistemas AIX o Linux

AIX, LINUX

Una aplicación accede a APPC mediante los siguientes puntos de entrada:

APPC Emite un verbo APPC de manera síncrona. Communications Server para Linux no devuelve el control a la aplicación hasta que finaliza el proceso del verbo.

APPC_Async
Emite un verbo APPC de manera asíncrona. Communications Server para Linux devuelve el control a la aplicación inmediatamente, con un valor devuelto que indica si el proceso del verbo todavía está ejecutándose o ha finalizado. En la mayoría de los casos, el proceso del verbo todavía está ejecutándose cuando el control vuelve a la aplicación. Más tarde, Communications Server para Linux utiliza una rutina callback suministrada por la aplicación para devolver los resultados del proceso del verbo. En algunos casos, el proceso del verbo ya ha finalizado cuando Communications Server para Linux devuelve el control a la aplicación; Communications Server para Linux no utiliza la rutina callback de la aplicación.

Nota: Los verbos de servidor de TP, tal como se describe en el Capítulo 5, "Verbos de servidor de TP", en la página 253, se deben emitir utilizando el punto de entrada asíncrono `APPC_Async` y no el punto de entrada síncrono `APPC`.

Los puntos de entrada `APPC` y `APPC_Async` están definidos en el archivo de cabecera APPC **/usr/include/sna/appc_c.h** (para AIX) o **/opt/ibm/sna/include/appc_c.h** (para Linux).

Una aplicación que ejecuta una sola tarea y que puede suspenderse mientras espera información de Communications Server para Linux o del sistema remoto sólo necesita utilizar el punto de entrada `APPC` (síncrono).

Si la aplicación ejecuta varias tareas (como comunicarse con más de un programa remoto a la vez o efectuar otro proceso además del de los verbos APPC), debe asegurarse de que no se suspende mientras espera información. En este caso, la aplicación debe utilizar el punto de entrada `APPC_Async` (asíncrono), suministrando una rutina callback que Communications Server para Linux puede utilizar para devolver información cuando esté disponible.

En los siguientes apartados se describen estos puntos de entrada y también algunas funciones adicionales definidas por las aplicaciones que la aplicación debe suministrar a dichos puntos.

Punto de entrada APPC

Una aplicación utiliza `APPC` para emitir un verbo APPC de manera síncrona. Communications Server para Linux no devuelve el control a la aplicación hasta que finaliza el proceso del verbo.

Llamada de función

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

Por motivos de compatibilidad con las implementaciones de APPC anteriores, Communications Server para Linux también proporciona los puntos de entrada `APPC_C` y `APPC_P`, que pueden utilizarse de la misma forma que `APPC`.

Parámetros suministrados

Cuando la aplicación utiliza el punto de entrada APPC para emitir un verbo, suministra el siguiente parámetro:

vcb Puntero a un bloque de control de verbo (VCB), que contiene los parámetros para el verbo que se emite. La estructura del VCB para cada verbo se describe en el Capítulo 3, "Verbos de control APPC", en la página 67 y en el Capítulo 4, "Verbos de conversación APPC", en la página 97. Estas estructuras están definidos en el archivo de cabecera APPC **/usr/include/sna/appc_c.h** (para AIX) o **/opt/ibm/sna/include/appc_c.h** (para Linux).

Nota: Los VCB APPC contienen muchos parámetros marcados como "reservados"; el software Communications Server para Linux utiliza internamente algunos de ellos y otros no se utilizan en esta versión, aunque pueden utilizarse en versiones futuras. Su aplicación no debe intentar acceder a ninguno de estos parámetros reservados, sino que debe establecer todo el contenido del VCB en cero para garantizar que todos estos parámetros tengan el valor cero, antes de que establezca otros parámetros utilizados por el verbo. Esto garantiza que Communications Server para Linux no interpretará de forma incorrecta ninguno de estos parámetros de uso interno y que su aplicación continuará funcionando con versiones posteriores de Communications Server para Linux en que estos parámetros pueden utilizarse para proporcionar nuevas funciones.

Para establecer el contenido del VCB en cero, utilice memset:

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

Valores devueltos

La función no devuelve ningún valor. Cuando la llamada vuelve, la aplicación puede examinar los parámetros del VCB para determinar si el verbo ha finalizado correctamente.

Punto de entrada APPC_Async

Una aplicación utiliza APPC_Async para emitir un verbo APPC de manera asíncrona. Communications Server para Linux devuelve el control a la aplicación inmediatamente, con un valor devuelto que indica si el proceso del verbo todavía está ejecutándose o ha finalizado. En la mayoría de los casos, el proceso del verbo todavía está ejecutándose cuando el control vuelve a la aplicación. Más tarde, Communications Server para Linux utiliza una rutina callback suministrada por la aplicación para devolver los resultados del proceso del verbo. En algunos casos, el proceso del verbo ya ha finalizado cuando Communications Server para Linux devuelve el control a la aplicación, de modo que Communications Server para Linux no utiliza la rutina callback de la aplicación.

Llamada de función

```
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;
```

Los tipos de parámetro como AP_UINT32, utilizados en estos puntos de entrada y en los VCB APPC, están definidos en el archivo de cabecera común **/usr/include/sna/values_c.h** (para AIX) o **/opt/ibm/sna/include/values_c.h** (para Linux), incluido en el archivo de cabecera APPC **/usr/include/sna/appc_c.h** (para AIX) o **/opt/ibm/sna/include/appc_c.h** (para Linux).

Parámetros suministrados

Cuando la aplicación utiliza el punto de entrada APPC_Async para emitir un verbo, suministra los siguientes parámetros:

vcb Puntero a un bloque de control de verbo (VCB), que contiene los parámetros para el verbo que se emite. La estructura del VCB para cada verbo se describe en el Capítulo 3, “Verbos de control APPC”, en la página 67 y en el Capítulo 4, “Verbos de conversación APPC”, en la página 97. Estas estructuras están definidas en el archivo de cabecera APPC **appc_c.h**.

Nota: Los VCB APPC contienen muchos parámetros marcados como “reservados”; el software Communications Server para Linux utiliza internamente algunos de ellos y otros no se utilizan en esta versión, aunque pueden utilizarse en versiones futuras. Su aplicación no debe intentar acceder a ninguno de estos parámetros reservados, sino que debe establecer todo el contenido del VCB en cero para garantizar que todos estos parámetros tengan el valor cero, antes de que establezca otros parámetros utilizados por el verbo. Esto garantiza que Communications Server para Linux no interpretará de forma incorrecta ninguno de estos parámetros de uso interno y que su aplicación continuará funcionando con versiones posteriores de Communications Server para Linux en que estos parámetros pueden utilizarse para proporcionar nuevas funciones.

Para establecer el contenido del VCB en cero, utilice memset:

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

comp_proc
Rutina callback que Communications Server para Linux en el archivo de cabecera APPC Linux llamará cuando el verbo finalice. Para ver más información sobre los requisitos para una rutina callback, consulte “Rutina callback para la finalización de verbos asíncronos” en la página 37.

corr Correlacionador opcional para ser usado por la aplicación. Este parámetro se define como una estructura de tipo union del lenguaje de programación C, de manera que la aplicación puede especificar cualquiera de los tres tipos diferentes de parámetros (puntero, largo sin signo o entero).

Communications Server para Linux no utiliza este valor, sino que lo pasa como parámetro a la rutina callback cuando el verbo finaliza. Este valor permite que la aplicación correlacione la información devuelta con su otro proceso.

Valores devueltos

El punto de entrada asíncrono devuelve uno de los siguientes valores:

AP_COMPLETED
El verbo ya ha finalizado. La aplicación puede examinar los parámetros del VCB para determinar si el verbo ha finalizado correctamente. Communications Server para Linux no llama a la rutina callback suministrada para este verbo.

AP_IN_PROGRESS
El verbo todavía no ha finalizado. La aplicación puede continuar con otro proceso, incluida la emisión de otros verbos APPC, siempre que éstos no dependan de la finalización del verbo actual. Sin embargo, la aplicación no debe intentar examinar o modificar los parámetros del VCB suministrados a este verbo.

Communications Server para Linux llama a la rutina callback suministrada para indicar cuándo finaliza el proceso del verbo. A continuación la aplicación puede examinar los parámetros del VCB.

Utilización del punto de entrada asíncrono

Al utilizar el punto de entrada asíncrono, tenga en cuenta lo siguiente:

- Si una aplicación especifica un puntero nulo en el parámetro *comp_proc*, el verbo finalizará de manera síncrona (como si la aplicación hubiera emitido el verbo utilizando el punto de entrada síncrono).
- Si la llamada a `APPC_Async` se efectúa desde una rutina callback de aplicación, no se permite especificar un puntero nulo en el parámetro *comp_proc*. En tales casos, Communications Server para Linux rechaza el verbo con el valor de código de retorno primario `AP_PARAMETER_CHECK` y el valor de código de retorno secundario `AP_SYNC_NOT_ALLOWED`.
- La aplicación no debe intentar utilizar o modificar ningún parámetro del VCB hasta que se haya llamado a la rutina callback.
- Hay varios verbos que no finalizan necesariamente en el orden en el que se emitieron. En concreto, si una aplicación emite un verbo asíncrono seguido de un verbo síncrono, la finalización del verbo síncrono no garantiza que el verbo asíncrono haya finalizado.
- El verbo [MC_]RECEIVE_AND_POST incluye un puntero a una rutina callback como uno de los parámetros del VCB. Este verbo puede emitirse utilizando tanto el punto de entrada síncrono como el asíncrono. Communications Server para Linux utiliza la rutina callback especificada en el VCB para devolver los resultados de este verbo. La rutina callback especificada en el punto de entrada asíncrono se utiliza sólo si la aplicación suministra un puntero nulo para la rutina callback en el VCB.

Rutina callback para la finalización de verbos asíncronos

Al utilizar el punto de entrada asíncrono, la aplicación debe suministrar un puntero a una rutina callback. En este apartado se describe cómo Communications Server para Linux utiliza esta rutina y las funciones que debe efectuar.

Llamada de función

```
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;
```

Parámetros suministrados

Communications Server para Linux llama a la rutina callback con los siguientes parámetros:

vcb Puntero al VCB suministrado por la aplicación. El VCB incluye ahora los parámetros devueltos establecidos por Communications Server para Linux.

tp_id Identificador de TP de 8 bytes del TP en que se ha emitido el verbo.

conv_id
Identificador de conversación de la conversación en que se ha emitido el verbo.

corr Valor de correlacionador suministrado por la aplicación. Este valor permite que la aplicación correlacione la información devuelta con su otro proceso.

La rutina callback no tiene que utilizar todos estos parámetros. Puede ejecutar todo el proceso necesario en el VCB devuelto o simplemente puede establecer una variable para informar al programa principal de que el verbo ha finalizado.

Valores devueltos

La función no devuelve ningún valor.

Utilización de la rutina callback para la finalización de verbos asíncronos

Al utilizar la rutina callback para la finalización de verbos asíncronos, la aplicación puede emitir verbos APPC asíncronos adicionales desde dentro de la rutina callback, si es necesario. Communications Server para Linux rechaza los verbos síncronos emitidos desde una rutina callback con los códigos de retorno primarios y secundarios `AP_PARAMETER_CHECK` y `AP_SYNC_NOT_ALLOWED`.

Puntos de entrada APPC: Sistemas Windows

WINDOWS

Una aplicación Windows accede a APPC mediante los siguientes puntos de entrada:

`WinAPPCStartup`
Registra la aplicación como un usuario de APPC de Windows y determina si el software de APPC admite el nivel de función que necesita la aplicación.

WinAsyncAPPC
Emite un verbo APPC. El verbo finaliza normalmente de manera asíncrona y no se bloquea; APPC indica la finalización enviando un mensaje a la ventana de la aplicación.

WinAsyncAPPCEx
Emite un verbo APPC. Si el verbo finaliza de manera asíncrona, APPC indica la finalización señalando un descriptor de contexto de sucesos. Utilice esta función en lugar de las versiones de bloqueo de los verbos para permitir el manejo de varias sesiones en el mismo subproceso.

WinAPPCCancelAsyncRequest
Cancela un verbo asíncrono pendiente (uno que se emite utilizando el punto de entrada `WinAsyncAPPC`). Según el verbo que esté pendiente, es posible que también finalice la conversación o el TP o que desactive la sesión que una conversación esté utilizando.

WinAPPCCleanup
Elimina el registro de la aplicación cuando ha acabado de utilizar APPC.

APPC Emite un verbo APPC. El verbo se bloquea; es decir, el proceso de la aplicación se suspende hasta que APPC termina de procesar el verbo y ha devuelto los resultados.

WinAPPCCancelBlockingCall
Cancela un verbo de bloqueo pendiente (uno que se emite utilizando el punto de entrada `APPC`). Según el verbo que esté pendiente, es posible que también finalice la conversación o el TP o que desactive la sesión que una conversación esté utilizando. Si desea obtener más información sobre las circunstancias en que se puede necesitar esta llamada, consulte el apartado "Verbos de bloqueo" en la página 45.

WinAPPCIsBlocking
Comprueba si hay un verbo de bloqueo pendiente para esta aplicación. Si desea obtener más información sobre las circunstancias en que se puede necesitar esta llamada, consulte el apartado "Verbos de bloqueo" en la página 45.

WinAPPCSetBlockingHook
Especifica el procedimiento de bloqueo que APPC utiliza mientras procesa verbos de bloqueo; esto sustituye el procedimiento de bloqueo por omisión de APPC. Se llama repetidamente al procedimiento de bloqueo hasta que el proceso del verbo ha finalizado. Para ver más información, consulte "Verbos de bloqueo" en la página 45.

WinAPPCUnhookBlockingHook
Deshace el registro del procedimiento de bloqueo especificado por una llamada anterior a `WinAPPCSetBlockingHook`, de modo que APPC vuelve a utilizar el procedimiento de bloqueo por omisión.

GetAppcConfig
Devuelve información sobre LU remotas configuradas para que las utilicen una LU local y una modalidad especificadas. Esta función se proporciona para que la utilicen programas de emulación 5250; la información devuelta se toma de los registros de usuario de 5250 de la configuración de Communications Server para Linux.

GetAppcReturnCode
Genera una serie de caracteres imprimible de los códigos de retorno principales y secundarios obtenidos en un verbo APPC.

Estos puntos de entrada están definidos en el archivo de cabecera APPC **winappc.h** de Windows. Este archivo está instalado en el subdirectorio **/sdk** del directorio en que se ha instalado el software del cliente Windows.

La aplicación debe llamar a WinAPPCStartup antes de intentar emitir cualquier verbo APPC.

A continuación, emite verbos APPC utilizando uno de los puntos de entrada siguientes:

- WinAsyncAPPC o WinAsyncAPPCEx (asíncrono). Si desarrolla aplicaciones nuevas para Windows, utilice uno de estos puntos de entrada.
- APPC (bloqueo). Este punto de entrada se proporciona para ofrecer compatibilidad con la implementación de APPC de AIX y Linux. En el apartado "Verbos de bloqueo" en la página 45 encontrará más información sobre cómo funcionan los verbos de bloqueo en el entorno Windows.

Una aplicación que proporciona emulación 5250 puede utilizar GetAppcConfig para obtener información sobre LU APPC remotas a las que se pueda acceder por medio de una determinada LU local.

Si un verbo devuelve códigos de retorno que no sean AP_OK, la aplicación puede utilizar GetAppcReturnCode para obtener una representación de estos códigos de retorno en forma de cadena de texto, que se puede utilizar para generar mensajes de error estándar.

Cuando la aplicación haya acabado de emitir verbos APPC, deberá llamar a WinAPPCCleanup antes de finalizar. Después de llamar a WinAPPCCleanup, la aplicación no debe intentar emitir más verbos APPC (a menos que llame primero a WinAPPCStartup para reinicializarse).

Los apartados siguientes describen los puntos de entrada de Windows.

WinAPPCStartup

La aplicación utiliza WinAPPCStartup para registrarse como un usuario de APPC para Windows y para determinar si el software de APPC admite la versión de APPC para Windows que la aplicación necesita.

Llamada de función

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

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

Parámetros suministrados

Cuando la aplicación utiliza el punto de entrada WinAPPCStartup para emitir un verbo, suministra los siguientes parámetros:

wVersionRequired
: Versión de APPC de Windows que la aplicación necesita. El byte de orden inferior especifica el número de versión principal, mientras que el byte de orden superior especifica el número de versión secundaria. Por ejemplo:

Versión	wVersionRequired
1.0	0x0001
1.1	0x0101
2.0	0x0002

Si la aplicación puede utilizar más de una versión, especifica la versión más alta que puede utilizar.

Valores devueltos

WinAPPCStartup devuelve uno de los siguientes valores:

0 (cero)
La aplicación se ha registrado satisfactoriamente y el software de APPC para Windows admite el número de versión especificado por la aplicación o bien una versión inferior. La aplicación debe comprobar el número de versión en la estructura WAPPCDATA para garantizar que sea suficientemente alto.

WAPPCVERNOTSUPPORTED
El número de versión especificado por la aplicación es inferior a la versión más baja que el software de APPC para Windows admite. La aplicación no se ha registrado.

WAPPCSYSNOTREADY
La aplicación no se ha registrado. Esto puede ser debido a que el software Remote API Client sobre Windows no se haya iniciado, a que el nodo local no esté activo o a otra anomalía del sistema como, por ejemplo, la escasez de recursos.

Si el valor de retorno de WinAPPCStartup es 0 (cero), la estructura WAPPCDATA contendrá información sobre el soporte que ofrece el software de APPC para Windows. Si el valor de retorno no es cero, el contenido de esta estructura será indefinido y la aplicación no deberá comprobarlo. Los parámetros de esta estructura son los siguientes:

wVersion
Número de versión de APPC para Windows que el software admite, en el mismo formato que el parámetro *wVersionRequired*. Si el software admite el número de versión solicitado, para este parámetro se establece el mismo valor que para el parámetro *wVersionRequired*; de lo contrario, se establece el número de versión más alto que el software admite, que es inferior al número de versión que proporciona la aplicación. La aplicación debe comprobar el valor devuelto y realizar una de las acciones siguientes:

- Si el número de versión devuelto es el mismo que el número de versión solicitado, la aplicación puede utilizar esta implementación de APPC para Windows.
- Si el número de versión devuelto es inferior al número de versión solicitado, la aplicación puede utilizar esta implementación de APPC para Windows pero no debe intentar utilizar características que el número de versión devuelto no admita. Si no puede hacerlo porque necesita características que no están disponibles en la versión más baja, deberá suspender la inicialización y no intentar emitir ningún verbo APPC.

szDescription
Cadena de texto que describe el software de APPC para Windows.

WinAsyncAPPC

La aplicación utiliza esta función para emitir un verbo APPC. Si el verbo finaliza de manera asíncrona, APPC indica la finalización enviando un mensaje al descriptor de contexto de Windows de la aplicación.

Antes de utilizar la llamada a `WinAsyncAPPC` por primera vez, la aplicación debe utilizar la llamada a `RegisterWindowMessage` para obtener el identificador de mensaje que APPC utilizará para los mensajes que indiquen la finalización de verbos asíncronos. Para ver más información, consulte "Uso" en la página 43.

Llamada de función

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

Parámetros suministrados

Los parámetros suministrados son los siguientes:

hWnd Descriptor de contexto de Windows que APPC utilizará para enviar un mensaje que indique la finalización de un verbo asíncrono.

vcbptr Puntero a la estructura del VCB del verbo. Este parámetro se define como un entero largo y, por lo tanto, se debe emitir desde un puntero a un entero largo. Si desea obtener más información sobre la estructura del VCB y sobre su uso para distintos verbos, consulte el Capítulo 3, "Verbos de control APPC", en la página 67 y el Capítulo 4, "Verbos de conversación APPC", en la página 97.

Nota: Los VCB APPC contienen muchos parámetros marcados como "reservados"; el software Communications Server para Linux utiliza internamente algunos de ellos y otros no se utilizan en esta versión, aunque pueden utilizarse en versiones futuras. Su aplicación no debe intentar acceder a ninguno de estos parámetros reservados, sino que debe establecer todo el contenido del VCB en cero para garantizar que todos estos parámetros tengan el valor cero, antes de que establezca otros parámetros utilizados por el verbo. Esto garantiza que Communications Server para Linux no interpretará de forma incorrecta ninguno de estos parámetros de uso interno y que su aplicación continuará funcionando con versiones posteriores de Communications Server para Linux en que estos parámetros pueden utilizarse para proporcionar nuevas funciones.

Para establecer el contenido del VCB en cero, utilice `memset`:

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

Valores devueltos

El valor de retorno de la función es uno de los siguientes:

`Handle` La llamada de función ha sido satisfactoria (se ha aceptado). Cuando el verbo finaliza más adelante, APPC utiliza este descriptor de contexto como identificador en el mensaje que se pasa al procedimiento de la ventana de la aplicación (si desea obtener más información, consulte el apartado "Uso" en la página 43). La aplicación también utiliza este descriptor de contexto como parámetro para la llamada a `WinAPPCCancelAsyncRequest` si necesita cancelar el verbo pendiente.

0 (cero)
La llamada de función no ha sido satisfactoria (no se ha aceptado).

Uso

Antes de utilizar WinAsyncAPPC por primera vez, la aplicación debe utilizar la llamada a RegisterWindowMessage para obtener el identificador de mensaje que APPC utilizará para los mensajes que indiquen la finalización de verbos asíncronos. RegisterWindowMessage es una llamada de función estándar de Windows, no específica de APPC; si desea obtener más información sobre la función, consulte la documentación de Windows. (No es necesario volver a emitir la llamada antes de los verbos APPC posteriores; el valor devuelto será el mismo para todas las llamadas que la aplicación emita).

La aplicación debe pasar la cadena "WinAsyncAPPC" a la función; el valor devuelto es un identificador de mensaje.

Cada vez que un verbo APPC que se ha emitido mediante el punto de entrada WinAsyncAPPC finaliza de manera asíncrona, APPC envía un mensaje al descriptor de contexto de Windows especificado en la llamada a WinAsyncAPPC. El formato del mensaje es el siguiente:

- El identificador del mensaje es el valor devuelto de la llamada a RegisterWindowMessage.
- El argumento *lParam* contiene la dirección del VCB que se ha proporcionado a la llamada original a WinAsyncAPPC; la aplicación puede utilizar esta dirección para acceder a los parámetros de retorno en la estructura del VCB.
- El argumento *wParam* contiene el descriptor de contexto que se ha devuelto a la llamada original a WinAsyncAPPC.

WinAsyncAPPCEx

La aplicación utiliza esta función para emitir un verbo APPC. Si el verbo finaliza de manera asíncrona, APPC indica la finalización señalando un descriptor de contexto de sucesos. Utilice esta función en lugar de las versiones de bloqueo de los verbos para permitir el manejo de varias sesiones en el mismo subproceso.

Llamada de función

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

Parámetros suministrados

Los parámetros suministrados son los siguientes:

eventhandle
Descriptor de contexto de sucesos que APPC señalará para indicar la finalización de verbo asíncrono.

vcbptr Puntero a la estructura del VCB del verbo. Este parámetro se define como un entero largo y, por lo tanto, se debe emitir desde un puntero a un entero largo. Si desea obtener más información sobre la estructura del VCB y sobre su uso para distintos verbos, consulte el Capítulo 3, "Verbos de control APPC", en la página 67 y el Capítulo 4, "Verbos de conversación APPC", en la página 97.

Nota: Los VCB APPC contienen muchos parámetros marcados como "reservados"; el software Communications Server para Linux utiliza

internamente algunos de ellos y otros no se utilizan en esta versión, aunque pueden utilizarse en versiones futuras. Su aplicación no debe intentar acceder a ninguno de estos parámetros reservados, sino que debe establecer todo el contenido del VCB en cero para garantizar que todos estos parámetros tengan el valor cero, antes de que establezca otros parámetros utilizados por el verbo. Esto garantiza que Communications Server para Linux no interpretará de forma incorrecta ninguno de estos parámetros de uso interno y que su aplicación continuará funcionando con versiones posteriores de Communications Server para Linux en que estos parámetros pueden utilizarse para proporcionar nuevas funciones.

Para establecer el contenido del VCB en cero, utilice `memset`:

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

Valores devueltos

El valor de retorno de la función es uno de los siguientes:

`Handle` La llamada de función ha sido satisfactoria (se ha aceptado) y el valor de retorno es un descriptor de contexto de tareas asíncronas. Cuando el verbo finaliza más adelante, APPC utiliza este descriptor de contexto para la notificación de sucesos a la aplicación (si desea obtener más información, consulte el apartado "Uso"). La aplicación también utiliza este descriptor de contexto como parámetro para la llamada a `WinAPPCCancelAsyncRequest` si necesita cancelar el verbo pendiente.

`0` (cero)
La llamada de función no ha sido satisfactoria (no se ha aceptado).

Uso

Esta función está pensada para utilizarla con `WaitForSingleObject` o `WaitForMultipleObjects` en la API de Windows. Cuando la operación asíncrona finaliza, se envía una notificación a la aplicación señalando el suceso. Al señalar el suceso, examine el código de retorno primario y el código de retorno secundario para comprobar si hay alguna condición de error.

WinAPPCCancelAsyncRequest

La aplicación utiliza esta función para cancelar un verbo APPC pendiente (emitido mediante el punto de entrada `WinAsyncAPPC`).

Llamada de función

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

Parámetros suministrados

El parámetro suministrado es el siguiente:

Handle Descriptor de contexto que se ha devuelto a la llamada original a `WinAsyncAPPC` para el verbo.

Valores devueltos

El valor de retorno de la función es uno de los siguientes:

`0` (cero)
El verbo pendiente se ha cancelado satisfactoriamente.

WAPPCINVALID
El parámetro proporcionado no coincide con el descriptor de contexto de ningún verbo APPC pendiente.

WAPPCALREADY
El verbo APPC identificado por el descriptor de contexto proporcionado ya ha finalizado. (Es posible que la aplicación ya haya procesado el mensaje resultante de la finalización del verbo o que el mensaje aún esté a la espera en la cola de mensajes de la aplicación).

Uso
Además de cancelar el verbo pendiente, es posible que APPC también finalice la conversación o el TP en que se ha emitido el verbo, que desactive la sesión, o las dos cosas. La acción que se llevará a cabo depende del verbo que se haya cancelado. APPC también envía un mensaje a la aplicación indicando la finalización del verbo cancelado; el código de retorno primario del verbo es `AP_CANCELLED`.

WinAPPCCleanup
La aplicación utiliza la función `WinAPPCCleanup` para deshacer el registro como usuario de APPC para Windows una vez que ha terminado de emitir verbos APPC.

Llamada de función
```
BOOL WINAPI WinAPPCCleanup (void);
```

Parámetros suministrados
Con la función `WinAPPCCleanup` no se proporcionan parámetros.

Valores devueltos
El valor de retorno de la función es uno de los siguientes:

TRUE El registro de la aplicación se ha deshecho satisfactoriamente.

FALSE Se ha producido un error durante el proceso de la llamada y no se ha deshecho el registro de la aplicación. Compruebe los archivos de anotaciones para ver si hay mensajes que indiquen la causa de la anomalía.

Verbos de bloqueo
Este apartado describe el funcionamiento de los verbos de bloqueo en el entorno Windows si la aplicación que realiza la llamada es de un solo subproceso y proporciona información que se debe tener en cuenta el escribir aplicaciones para utilizar verbos de bloqueo. (Normalmente, una aplicación Windows utilizaría varios subprocesos para evitar el problema que supondría que un verbo de bloqueo bloquease toda la aplicación).

Aunque parece que un verbo emitido al punto de entrada `APPC` suspenda la aplicación hasta que el proceso del verbo finaliza, la biblioteca de APPC debe ceder el control del sistema mientras espera a que Remote API Client finalice el proceso para permitir que se ejecuten otros procesos. Para hacerlo, la aplicación utiliza una función de bloqueo a la que se llama repetidamente mientras la biblioteca está a la espera; la función permite que se envíen mensajes de Windows a otros procesos. Si desea obtener más información sobre esta función, consulte el apartado “Función de bloqueo por omisión” en la página 46.

Es posible que la función de bloqueo envíe un mensaje a la aplicación que ha emitido el verbo de bloqueo original; en este caso, se puede volver a acceder a la

aplicación aunque tenga una llamada de bloqueo pendiente. En estas circunstancias, la aplicación puede continuar con otros procesos que no estén relacionados con la emisión de verbos APPC. No obstante, no puede emitir otro verbo al punto de entrada APPC (ni a ninguna otra API de Remote API Client) mientras el primer verbo esté pendiente; el verbo se rechazará con el código de retorno primario AP_THREAD_BLOCKING.

La aplicación puede comprobar si hay un verbo de bloqueo pendiente (es decir, si se ha vuelto a acceder a ella como resultado de un mensaje recibido mientras el verbo estaba pendiente) utilizando la función WinAPPCIsBlocking (si desea obtener más información, consulte el apartado "WinAPPCIsBlocking" en la página 48). Si esta función indica que hay una llamada de bloqueo pendiente, la aplicación no debe intentar emitir más verbos APPC utilizando el punto de entrada de bloqueo. No obstante, la aplicación puede realizar las siguientes acciones:

- Continuar otros procesos.
- Emitir verbos APPC utilizando el punto de entrada asíncrono.
- Emitir WinAPPCCancelBlockingCall para cancelar el verbo de bloqueo pendiente.

Función de bloqueo por omisión

La función de bloqueo estándar que utiliza la biblioteca de APPC para Windows es la siguiente:

```
BOOL far pascal DefaultBlockingHook (void)  {
       MSG msg;
       /* obtener el siguiente mensaje si lo hay */
       if ( PeekMessage (&msg,NULL,0,0,PM_NOREMOVE)  )  {
             if ( msg.message = = WM_QUIT  )
                    return FALSE;   // permitir que la aplicación procese WM_QUIT
             PeekMessage (&msg,NULL,0,0,PM_REMOVE);
             TranslateMessage (&msg);
             DispatchMessage (&msg);
       }
       /* TRUE si no se ha recibido ningún mensaje WM_QUIT */
       return TRUE;
}
```

Si la aplicación necesita que se realicen otros procesos como parte de la función de bloqueo, puede especificar su propia función de bloqueo para que sustituya a la función por omisión que APPC proporciona. Para hacerlo, utiliza la llamada a WinAPPCSetBlockingHook (consulte "WinAPPCSetBlockingHook" en la página 48).

Una función de bloqueo debe devolver el valor FALSE si recibe un mensaje WM_QUIT; esto significa que APPC para Windows devuelve el control a la aplicación, que a continuación puede procesar el mensaje y finalizar. De lo contrario, la función debe devolver el valor TRUE.

APPC

La aplicación utiliza esta función para emitir un verbo APPC, que se bloquea hasta que el proceso del verbo finaliza. Por motivos de compatibilidad con las implementaciones de APPC anteriores, Remote API Client también proporciona los puntos de entrada APPC_C y APPC_P, que pueden utilizarse de la misma forma que APPC.

Este punto de entrada proporciona soporte para los verbos APPC síncronos en Windows, lo que puede resultar útil al hacer una migración de otros entornos de sistema operativo.

Llamada de función

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

Parámetros suministrados

El parámetro suministrado es el siguiente:

vcbptr Puntero a la estructura del VCB del verbo. Este parámetro se define como un entero largo y, por lo tanto, se debe emitir desde un puntero a un entero largo. Para conocer la definición de la estructura del VCB para cada verbo APPC, consulte el Capítulo 3, "Verbos de control APPC", en la página 67 y el Capítulo 4, "Verbos de conversación APPC", en la página 97.

Nota: Los VCB APPC contienen muchos parámetros marcados como "reservados"; el software Communications Server para Linux utiliza internamente algunos de ellos y otros no se utilizan en esta versión, aunque pueden utilizarse en versiones futuras. Su aplicación no debe intentar acceder a ninguno de estos parámetros reservados, sino que debe establecer todo el contenido del VCB en cero para garantizar que todos estos parámetros tengan el valor cero, antes de que establezca otros parámetros utilizados por el verbo. Esto garantiza que Communications Server para Linux no interpretará de forma incorrecta ninguno de estos parámetros de uso interno y que su aplicación continuará funcionando con versiones posteriores de Communications Server para Linux en que estos parámetros pueden utilizarse para proporcionar nuevas funciones.

Para establecer el contenido del VCB en cero, utilice `memset`:

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

Valores devueltos

La función no devuelve ningún valor. Cuando la llamada vuelve, la aplicación debe examinar los parámetros *primary_rc* y *secondary_rc* de la estructura del VCB para determinar si el verbo ha finalizado correctamente. Si desea obtener información sobre los parámetros devueltos en la estructura del VCB, consulte las descripciones de los distintos verbos en el Capítulo 3, "Verbos de control APPC", en la página 67 y el Capítulo 4, "Verbos de conversación APPC", en la página 97.

WinAPPCCancelBlockingCall

La aplicación utiliza la función `WinAPPCCancelBlockingCall` para cancelar un verbo de bloqueo APPC pendiente (emitido mediante el punto de entrada `APPC`).

Llamada de función

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

Parámetros suministrados

No se proporcionan parámetros para este punto de entrada. (Sólo puede haber un verbo de bloqueo pendiente en un momento dado, de modo que no es necesario identificar el verbo en concreto que se debe cancelar).

Valores devueltos

El valor de retorno de la función es uno de los siguientes:

TRUE El verbo pendiente se ha cancelado satisfactoriamente.

FALSE No había ningún verbo APPC de bloqueo pendiente o se ha producido un error durante el proceso de la llamada y el verbo no se ha cancelado.

Uso

Además de cancelar el verbo pendiente, APPC también finaliza la conversación en que se ha emitido el verbo y desactiva la sesión. Si el verbo está relacionado con un TP en lugar de una conversación (como RECEIVE_ALLOCATE o TP_STARTED), APPC finalizará el TP.

WinAPPCIsBlocking

La aplicación utiliza la función `WinAPPCIsBlocking` para comprobar si hay un verbo de bloqueo APPC pendiente (un verbo emitido mediante el punto de entrada `APPC`).

Llamada de función

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

Parámetros suministrados

No se proporcionan parámetros con esta función.

Valores devueltos

El valor de retorno de la función es uno de los siguientes:

TRUE Hay un verbo APPC de bloqueo pendiente. Si es necesario, la aplicación puede utilizar la función `WinAPPCCancelBlockingCall` para cancelarlo.

FALSE No hay ningún verbo APPC de bloqueo pendiente.

WinAPPCSetBlockingHook

La aplicación utiliza esta llamada para especificar su propia función de bloqueo, que APPC utilizará en lugar de la función de bloqueo por omisión. Si desea obtener más información sobre el funcionamiento de la función de bloqueo y sobre las funciones que debe realizar consulte el apartado “Verbos de bloqueo” en la página 45.

Llamada de función

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

Parámetros suministrados

El parámetro suministrado es el siguiente:

lpBlockFunc
Dirección de la instancia de procedimiento de la función de bloqueo de la aplicación. La aplicación debe utilizar la llamada a `MakeProcInstance` para obtener esta dirección; si desea obtener más información, consulte la documentación de Windows.

Valores devueltos

El valor de retorno es la dirección de la instancia de procedimiento de la función de bloqueo anterior. Si la aplicación está utilizando más de una función de bloqueo y necesitará restaurar la función de bloqueo anterior más adelante, deberá guardar esta dirección; a continuación puede volver a emitir `WinAPPCSetBlockingHook` utilizando el valor guardado para restaurar la función de bloqueo anterior. Si está utilizando sólo una función de bloqueo o no será necesario restaurar el valor anterior, puede pasar por alto el valor de retorno de esta llamada.

Uso

La nueva función de bloqueo permanecerá en vigor hasta que la aplicación emita una de las siguientes llamadas:

- `WinAPPCSetBlockingHook` (con una dirección de instancia de procedimiento distinta) para especificar una nueva función de bloqueo o para restaurar una función anterior
- `WinAPPCUnhookBlockingHook` (véase "WinAPPCUnhookBlockingHook"), para no seguir utilizando la función de bloqueo actual y volver a la función de bloqueo por omisión

WinAPPCUnhookBlockingHook

La aplicación utiliza esta llamada para eliminar su propia función de bloqueo, que se ha especificado anteriormente mediante `WinAPPCSetBlockingHook`, y volver a utilizar la función de bloqueo por omisión de APPC.

Llamada de función

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

Parámetros suministrados

No se proporcionan parámetros para esta función.

Valores devueltos

El valor de retorno de la función es uno de los siguientes:

TRUE La función de bloqueo se ha eliminado satisfactoriamente; las llamadas de bloqueo siguientes utilizarán la función de bloqueo por omisión.

FALSE La llamada no ha finalizado satisfactoriamente.

GetAppcConfig

La función `GetAppcConfig` se proporciona para que la utilicen programas de emulación 5250. La función devuelve información sobre las LU remotas a las que puede acceder una LU local especificada, tal como está definido en los registros de usuario de emulación 5250 en la configuración de Communications Server para Linux.

Para determinar la información que esta llamada necesita, Communications Server para Linux compara el nombre de usuario configurado para el cliente Windows con los registros de usuario de 5250 definidos en la el registro de configuración (o, si el nombre de usuario no está definido explícitamente, comprueba si hay un registro <DEFAULT>). En el registro de usuario apropiado, compara el alias de LU local y el nombre de modalidad proporcionado en esta llamada con las definiciones de la sesión y devuelve el alias de LU remota para cada sesión coincidente.

La aplicación proporciona un descriptor de contexto de Windows al que APPC puede enviar un mensaje cuando el verbo finaliza de manera asíncrona. Antes de utilizar `GetAppcConfig` por primera vez, la aplicación debe utilizar `RegisterWindowMessage` para obtener el identificador de mensaje que APPC utilizará para el mensaje que indique la finalización asíncrona de la llamada y `WinAPPCStartup` para registrarse como una aplicación APPC para Windows. Si desea obtener más información, consulte la descripción de `WinAPPCStartup` en los apartados "WinAPPCStartup" en la página 40 y "Uso" en la página 51.

Un método alternativo para indicar la finalización de la llamada es proporcionar un puntero a un valor entero (el parámetro *AsyncRetCode*) en que APPC puede

devolver valores para indicar que la llamada has resultado anómala, está en progreso o ha finalizado. Se recomienda que las aplicaciones para Windows utilicen el primer método, es decir, que proporcionen un descriptor de contexto de Windows.

Llamada de función

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

Parámetros suministrados

Los parámetros suministrados son los siguientes:

hWnd Descriptor de contexto de Windows que APPC utilizará para enviar un mensaje que indique la finalización asíncrona de esta llamada. Si se utiliza este parámetro, el puntero al parámetro *AsyncRetCode* debe ser un puntero nulo.

LocalLU
Puntero al alias de la LU local para la que se necesita información de configuración. Es una cadena ASCII de un máximo de ocho caracteres que termina con un carácter nulo (cero binario); si el alias de LU tiene menos de ocho caracteres, deberá ir inmediatamente seguido por el carácter nulo y no se deberá rellenar con espacios.

Para indicar la LU local por omisión, defina este parámetro de modo que apunte a una cadena que conste de ocho espacios ASCII seguidos de un carácter nulo.

Mode Puntero al nombre de la modalidad (utilizada por la LU local) para la que se necesita información de configuración. Es una cadena ASCII de un máximo de ocho caracteres que termina con un carácter nulo (cero binario); si el nombre de la modalidad tiene menos de ocho caracteres, deberá ir inmediatamente seguido por el carácter nulo y no se deberá rellenar con espacios. Para los programas de emulación 5250, el nombre de modalidad es normalmente `QPCSUPP`.

NumRemLU
Puntero a un entero que APPC puede utilizar para devolver el número de LU remotas configuradas.

MaxRemLU
Número máximo de alias de LU remota que pueden caber en el almacenamiento intermedio de datos proporcionado (vea el parámetro siguiente). Cada alias de LU necesita 9 bytes, de modo que la longitud del almacenamiento intermedio proporcionado debe ser, como mínimo, de nueve veces el valor proporcionado de *MaxRemLU*.

RemLU
Almacenamiento intermedio para almacenar los alias de LU remota.

AsyncRetCode
Si la aplicación utiliza el método recomendado para indicar la finalización, este parámetro está reservado; la aplicación debe proporcionar un puntero nulo.

Si la aplicación utiliza el método alternativo, este parámetro es un puntero al entero que APPC utiliza para el código de retorno asíncrono de la función. En este caso, el parámetro *hWnd* debe ser un descriptor de contexto nulo.

Valores devueltos

Cuando la llamada vuelve, la aplicación puede probar el valor de la expresión "*ReturnedHandle* & APPC_CFG_SUCCESS " para determinar si la función ha resultado satisfactoria.

Si el valor de la expresión "*ReturnedHandle* & APPC_CFG_SUCCESS" es `TRUE` y la aplicación utiliza el método recomendado para indicar la finalización, el valor de retorno será un descriptor de contexto. Cuando la función finaliza más adelante, APPC utiliza este descriptor de contexto como identificador en el mensaje que se pasa al procedimiento de la ventana de la aplicación (si desea obtener más información, consulte el apartado "Uso").

Si el valor de la expresión "*ReturnedHandle* & APPC_CFG_SUCCESS" es `TRUE` y la aplicación utiliza el método alternativo para indicar la finalización, se establecerá `APPC_CFG_PENDING` como valor del parámetro *AsyncRetCode*. La aplicación debe probar este valor periódicamente para comprobar si la función ha finalizado. Cuando la función finaliza más adelante, APPC establece uno de los códigos de retorno asíncronos listados en el apartado "Uso" como valor de este parámetro.

Si el valor de la expresión es `FALSE`, significa que la llamada de función no se ha aceptado. El valor de *ReturnedHandle* es uno de los siguientes:

APPC_CFG_ERROR_NO_APPC_INIT
La aplicación no ha emitido la llamada a `WinAPPCStartup`. Esta llamada se debe emitir antes de que se utilice `GetAppcConfig`.

APPC_CFG_ERROR_INVALID_HWND
La aplicación ha proporcionado un descriptor de contexto de Windows que no es válido.

APPC_CFG_ERROR_BAD_POINTER
La aplicación ha proporcionado un puntero nulo al descriptor de contexto de Windows para utilizar el método alternativo para indicar la finalización, pero ha proporcionado un puntero para el parámetro *AsyncRetCode* que no es válido.

APPC_CFG_ERROR_UNCLEAR_COMPLETION_MODE
La aplicación ha proporcionado un descriptor de contexto de Windows (en el parámetro *hWnd*) y un puntero que no es nulo al parámetro *AsyncRetCode*, por lo que APPC no ha podido determinar cómo debe indicar la finalización asíncrona.

APPC_CFG_ERROR_TOO_MANY_REQUESTS
Hay demasiadas peticiones `GetAppcConfig` pendientes. La aplicación debería ceder, para permitir que se ejecuten otros procesos, y volver a intentar la llamada más adelante.

APPC_CFG_ERROR_GENERAL_FAILURE
Se ha producido un error del sistema.

Uso

Antes de utilizar `GetAppcConfig` por primera vez, la aplicación debe utilizar la llamada a `RegisterWindowMessage` para obtener el identificador de mensaje que APPC utilizará para los mensajes que indiquen la finalización asíncrona.

`RegisterWindowMessage` es una llamada de función estándar de Windows, no específica de APPC; si desea obtener más información sobre la función, consulte la documentación de Windows. (No es necesario que la aplicación vuelva a emitir la llamada antes de posteriores llamadas a `GetAppcConfig`; el valor devuelto será el mismo para todas las llamadas que la aplicación emita).

La aplicación debe pasar el valor `WIN_APPC_CFG_COMPLETION_MSG` a la función; el valor devuelto es un identificador de mensaje.

Una aplicación puede indicar la finalización utilizando un descriptor de contexto de Windows o utilizando un método alternativo, tal como se indica a continuación:

- Si la aplicación utiliza un descriptor de contexto de Windows para indicar la finalización, APPC enviará un mensaje a este descriptor de contexto de Windows cuando la llamada finalice de manera asíncrona. El formato del mensaje es el siguiente:
  - El identificador del mensaje es el valor devuelto de la llamada a `RegisterWindowMessage`.
  - El argumento *wParam* contiene el descriptor de contexto que se ha devuelto a la llamada original a `GetAppcConfig`.
  - El argumento *lParam* contiene uno de los siguientes códigos de retorno asíncronos:

    APPC_CFG_SUCCESS_NO_DEFAULT_REMOTE
    La configuración se ha recuperado satisfactoriamente. No hay ninguna LU remota por omisión configurada para la LU local y la modalidad especificadas.

    APPC_CFG_SUCCESS_DEFAULT_REMOTE
    La configuración se ha recuperado satisfactoriamente. Hay una LU remota por omisión configurada para la LU local y la modalidad especificadas. (Communications Server para Linux no devuelve este valor porque no tiene un concepto de configurar LU remotas por omisión; no obstante, la aplicación debería permitir este código de retorno para garantizar la compatibilidad con otras implementaciones de APPC).

    APPC_CFG_ERROR_NO_DEFAULT_LOCAL_LU
    La aplicación ha proporcionado un alias de LU local en blanco que indica la LU local por omisión, pero no hay ninguna LU local por omisión configurada.

    APPC_CFG_ERROR_BAD_LOCAL_LU
    El alias de LU local proporcionado no coincide con ningún alias de LU local configurada utilizada para la emulación 5250.

    APPC_CFG_ERROR_GENERAL_FAILURE
    Se ha producido un error del sistema.
- Si la aplicación utiliza el método alternativo para indicar la finalización, APPC establecerá como código de retorno asíncrono uno de los códigos de retorno de la lista correspondiente al argumento *lParam* (en el mensaje de Windows) cuando la llamada finalice.

La aplicación puede comprobar si la llamada ha sido satisfactoria o anómala probando las expresiones "*RetCode* & APPC_CFG_SUCCESS" o "*RetCode* & APPC_CFG_FAILURE", donde *RetCode* es el argumento *lParam* en el mensaje de Windows o el parámetro *AsyncRetCode* devuelto a la aplicación. Si "*RetCode* &

APPC_CFG_SUCCESS" es TRUE, significa que la llamada ha resultado satisfactoria; si "*RetCode* & APPC_CFG_FAILURE" es TRUE, significa que la llamada ha sido anómala.

Si la llamada ha resultado satisfactoria, la aplicación puede comprobar los valores de los parámetros *NumRemLU* y *RemLU*:

- *NumRemLU* contiene el número total de LU remotas configuradas. Si este número es superior al parámetro *MaxRemLU* proporcionado, significa que el almacenamiento intermedio proporcionado no era suficientemente grande para contener todos los alias de LU remotas. La aplicación puede utilizar los alias devueltos o puede volver a emitir GetAppcConfig con un almacenamiento intermedio suficientemente grande para contener todos los alias.
- *RemLU* contiene los alias de las LU remotas. Cada alias es un cadena de hasta ocho caracteres seguido de un carácter nulo y ocupa 9 bytes del almacenamiento intermedio. El número de alias de LU devuelto es el más pequeño del parámetro devuelto *MaxRemLU* y el parámetro devuelto *NumRemLU*.

Para determinar la información que esta llamada necesita, Communications Server para Linux compara el nombre de usuario configurado para el cliente Windows con los registros de usuario de 5250 definidos en la el registro de configuración (o, si el nombre de usuario no está definido explícitamente, comprueba si hay un registro <DEFAULT>). En el registro de usuario apropiado, compara el alias de LU local y el nombre de modalidad proporcionado en esta llamada con las definiciones de la sesión y devuelve el alias de LU remota para cada sesión coincidente.

GetAppcReturnCode

Esta llamada devuelve una cadena de caracteres imprimible que interpreta los códigos de retorno de un VCB proporcionado. La cadena se puede utilizar para generar mensajes de error de la aplicación para códigos de retorno que no sean AP_OK.

Esta llamada proporciona cadenas para mostrarlas al usuario final de una aplicación APPC. Para los códigos de retorno que indican errores de configuración o errores del usuario (por ejemplo, si un componente necesario no está configurado o no se ha iniciado), la cadena debe proporcionar suficiente información para ayudar al usuario a corregir el problema. En el caso de los códigos de retorno que indiquen errores de la aplicación (por ejemplo, si la aplicación ha emitido un verbo que no es válido o no ha podido proporcionar un parámetro necesario), normalmente el usuario no puede corregir el problema; en estos casos, la cadena sólo resulta útil para un desarrollador de aplicaciones.

Llamada de función

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

Parámetros suministrados

Los parámetros suministrados son los siguientes:

vcbptr Puntero a la estructura del VCB del verbo. Este parámetro se define como un entero largo y, por lo tanto, se debe emitir desde un puntero a un entero largo. Si desea obtener más información sobre la estructura del VCB

y sobre su uso para distintos verbos, consulte el Capítulo 3, “Verbos de control APPC”, en la página 67 o el Capítulo 4, “Verbos de conversación APPC”, en la página 97.

buffer_length
Longitud (en bytes) del almacenamiento intermedio proporcionado por la aplicación para contener la cadena de datos devuelta. La longitud recomendada es de 256 bytes.

buffer_addr
Dirección del almacenamiento intermedio proporcionado por la aplicación para contener la cadena de datos devuelta.

Valores devueltos

El valor de retorno de la función es uno de los siguientes:

0 (cero)
La función ha finalizado satisfactoriamente. La cadena de caracteres devuelta está en el almacenamiento intermedio identificado por el parámetro *buffer_addr*. Esta cadena termina con un carácter nulo (cero binario), pero no incluye un carácter final de nueva línea (\n).

0x20000001
APPC no ha podido leer del VCB proporcionado o no ha podido grabar en el almacenamiento intermedio de datos proporcionado.

0x20000002
El almacenamiento intermedio de datos proporcionado es demasiado pequeño para contener la cadena de caracteres devuelta.

0x20000003
La biblioteca de enlaces dinámicos **APPCST32.DLL**, que genera las cadenas de caracteres devueltas para esta función, no se ha podido cargar.

Consideraciones sobre AIX o Linux

AIX, LINUX

En este apartado se resume la información que se debe tener en cuenta para desarrollar TP para utilizarlos en el entorno AIX o Linux.

Procesos múltiples

Si el proceso que ha emitido TP_STARTED o RECEIVE_ALLOCATE crea un subproceso, el subproceso no puede utilizar el parámetro *tp_id* devuelto al proceso superior. Sin embargo, puede emitir su propio TP_STARTED o RECEIVE_ALLOCATE para obtener su propio *tp_id*.

Dos o más instancias del mismo TP pueden ejecutarse como procesos diferentes, pero cada instancia tiene asignado su propio *tp_id*.

Puede desarrollar una aplicación en la que un proceso contenga muchos TP, cada uno con su propio *tp_id*. Sin embargo, debe diseñar la aplicación con cuidado para evitar situaciones de punto muerto, en las que un verbo APPC no puede finalizar a causa del estado de otras conversaciones y otros TP del mismo proceso. Esto puede

ocurrir si el programa está esperando a que se le envíe información en una conversación para poder devolver otros datos y otra conversación del mismo proceso está esperando estos datos para poder enviar la información requerida en un principio por la primera conversación. Hasta cierto punto, esto puede evitarse utilizando un proceso aparte para cada TP.

Compilación y enlace de la aplicación APPC

Aplicaciones AIX

Para compilar y enlazar aplicaciones de 32 bits, utilice las opciones siguientes:

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

Para compilar y enlazar aplicaciones de 64 bits, utilice las opciones siguientes:

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

Aplicaciones Linux

Antes de compilar y enlazar una aplicación APPC, especifique el directorio en que se encuentran las bibliotecas compartidas, de modo que la aplicación pueda encontrarlas en tiempo de ejecución. Para hacerlo, establezca la variable de entorno `LD_RUN_PATH` en **/opt/ibm/sna/lib** o en **/opt/ibm/sna/lib64** si está compilando una aplicación de 64 bits.

Para compilar y enlazar aplicaciones de 32 bits, utilice las opciones siguientes:

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

Para compilar y enlazar aplicaciones de 64 bits, utilice las opciones siguientes:

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

Consideraciones sobre Windows

WINDOWS

Este apartado resume las consideraciones sobre el proceso que se deben tener en cuenta al desarrollar aplicaciones en un Remote API Client para Windows. Las consideraciones sobre el proceso para Windows son las siguientes:

- Compilación y enlace de programas APPC
- Finalización de aplicaciones

Compilación y enlace de programas APPC

Las siguientes consideraciones sobre el proceso son importantes al compilar y enlazar programas APPC en Windows:

Opciones del compilador para empaquetar estructuras
Las estructuras del VCB para verbos APPC no están empaquetadas. No utilice opciones del compilador que cambien este método de empaquetado. Los parámetros BYTE se encuentran dentro de los límites de BYTE, los

parámetros WORD se encuentran dentro de los límites de WORD y los parámetros DWORD se encuentran dentro de los límites de DWORD

Archivos de cabecera
El archivo de cabecera APPC principal que se debe incluir en las aplicaciones APPC para Windows se denomina **winappc.h**. Si la aplicación utiliza la llamada a `GetAppcConfig`, también se deberá incluir el archivo de cabecera **appccfg.h**. Estos archivos están instalados en el subdirectorio **\sdk** para aplicaciones de 32 bits o en **\sdk64** para aplicaciones de 64 bits, dentro del directorio en el que ha instalado el software del Cliente Windows.

Enlace durante la carga
Para enlazar el TP a APPC durante la carga, enlace el TP a la biblioteca **\sdk\wappc32.lib** para aplicaciones de 32 bits o a **\sdk64\wappc32.lib** para aplicaciones de 64 bits.

Enlace en tiempo de ejecución
Para enlazar el TP a APPC en tiempo de ejecución, incluya las siguientes llamadas en el TP:

- `LoadLibrary` para cargar la biblioteca de enlaces dinámicos de APPC **wappc32.dll**.
- `GetProcAddress` para especificar APPC en cada uno de los puntos de entrada APPC necesarios (como `WinAsyncAPPC`, `WinAPPCStartup` y `WinAPPCCleanup`)
- `FreeLibrary` cuando la biblioteca ya no es necesaria

Finalización de aplicaciones

APPC no puede determinar cuándo finaliza una aplicación en Windows. Por lo tanto, si una aplicación se debe cerrar (por ejemplo, si recibe un mensaje WM_CLOSE), la aplicación deberá emitir la llamada a `WinAPPCCleanup`. Si no se puede emitir la llamada, el sistema se quedará en un estado indeterminado; no obstante, cuando APPC detecta posteriormente que la aplicación ha finalizado se realiza tanta limpieza como es posible.

Información de configuración

El archivo de configuración de Communications Server para Linux, definido y mantenido por el administrador del sistema, contiene información necesaria para que los TP puedan comunicarse. Para ver más información sobre la configuración, consulte la publicación *Communications Server para Linux - Guía de administración*.

TP invocado

Antes de desarrollar un TP invocado, debe coordinar el nombre del TP local con el administrador del sistema. El nombre puede contener hasta 64 caracteres.

AIX, LINUX

Si intenta utilizar el formato ampliado del verbo RECEIVE_ALLOCATE, en el que la aplicación puede especificar una LU local de la que se aceptarán las peticiones de conversación entrantes, también deberá coordinar el alias de LU local (nombre

por el que la LU local se conoce en el TP local) con el administrador del sistema. Este alias puede contener hasta 8 caracteres.

Para ver más información, consulte el verbo RECEIVE_ALLOCATE en el Capítulo 3, “Verbos de control APPC”, en la página 67.

TP que invoca

La siguiente lista resume la información que necesita obtener de su administrador del sistema (o coordinar con él) antes de desarrollar un TP que invoca:

Alias de LU local
Nombre por el que la LU local se conoce en el TP local. Este nombre puede contener hasta ocho caracteres. Para ver más información, consulte la descripción del verbo TP_STARTED en el Capítulo 3, “Verbos de control APPC”, en la página 67.

Nombre de TP asociado
Este nombre puede contener hasta 64 caracteres. Para ver más información, consulte la descripción del verbo [MC_]ALLOCATE en el Capítulo 4, “Verbos de conversación APPC”, en la página 97.

Alias de LU asociada
Nombre por el que la LU asociada se conoce en el TP local. Este nombre puede contener hasta ocho caracteres. Para ver más información, consulte la descripción del verbo [MC_]ALLOCATE en el Capítulo 4, “Verbos de conversación APPC”, en la página 97.

Nombre de modalidad
Conjunto de características que se utilizarán en una sesión LU-LU. Este nombre puede contener hasta ocho caracteres. Para ver más información, consulte la descripción del verbo [MC_]ALLOCATE en el Capítulo 4, “Verbos de conversación APPC”, en la página 97.

Seguridad de conversación
Para utilizar la seguridad de conversación, se necesita una combinación válida de identificador de usuario y contraseña para acceder al TP invocado. El identificador de usuario y la contraseña pueden contener hasta 10 caracteres. Ambos parámetros son sensibles a las mayúsculas y minúsculas; el sistema distingue entre las letras en mayúsculas y en minúsculas. La información de seguridad se guarda en un archivo de seguridad. Para ver más información, consulte “Visión general de la seguridad de conversación”.

Visión general de la seguridad de conversación

La seguridad de conversación puede utilizarse para hacer que el TP que invoca proporcione un identificador de usuario y una contraseña para que APPC asigne una conversación con el TP invocado.

Al configurar el TP invocado, el administrador del sistema indica si se debe usar la seguridad de conversación. Si es así, el TP que invoca debe suministrar una combinación de *user_id* y *password* como parámetros del verbo [MC_]ALLOCATE. Estos parámetros deben coincidir con una de las combinaciones de parámetros *user_id* y *password* establecidas durante la configuración.

Un TP invocado que a su vez invoca otro TP es un caso especial (consulte el Capítulo 1, “Conceptos”, en la página 1). Consideremos que un TP A invoca un TP B, que requiere información de seguridad, y que el TP B, a su vez, invoca el TP C, que también requiere información de seguridad. Mediante el verbo [MC_]ALLOCATE, el TP B puede indicar que ya se ha verificado la seguridad de conversación. En este caso, APPC obtiene el identificador de usuario suministrado por el TP A al TP B y lo envía al TP C con una indicación de seguridad ya verificada; el TP C no necesita comprobar la contraseña.

AIX, LINUX

En algunos casos, el TP puede necesitar indicar la seguridad ya verificada cuando no ha sido invocado por otro TP pero ha obtenido y verificado por otros medios la información de seguridad pertinente (por ejemplo, un usuario que entra un identificador de usuario y una contraseña durante una secuencia de inicio de sesión). Communications Server para Linux da soporte a esta función de la siguiente forma:

- Si el TP que especifica la seguridad verificada ha sido invocado por otro TP que ha especificado un identificador de usuario y una contraseña, APPC envía este identificador de usuario.
- De lo contrario, APPC obtiene el nombre de usuario de AIX o Linux con el que se ejecuta el TP, truncado en 10 caracteres, si es necesario, y lo utiliza como identificador de usuario de seguridad de conversación. Asegúrese de que este nombre contiene caracteres de cadena de tipo AE válidos y que es un nombre de usuario válido para el TP que se invoca.
- Si la aplicación utiliza un método diferente para obtener información de seguridad (por ejemplo, si solicita al usuario que indique explícitamente un identificador de usuario y una contraseña, en lugar de confiar en la seguridad del sistema AIX o Linux), puede utilizar el verbo SET_TP_PROPERTIES para especificar este parámetro *user_id* para APPC antes de emitir el verbo [MC_]ALLOCATE.

Communications Server para Linux también da soporte a la seguridad de sesiones LU-LU, que proporciona la comprobación de la seguridad al iniciar la sesión entre las LU APPC locales y remotas. La seguridad de sesiones LU-LU se especifica durante la configuración y no requiere ninguna acción en los programas APPC. Para obtener más información consulte la publicación *Communications Server para Linux - Guía de administración*.

Inicio de TP

Las conversaciones tienen lugar entre un TP que invoca y un TP invocado. En este apartado se explica cómo se inician los TP que invocan y los invocados.

TP que invocan

El TP que invoca se inicia mediante un mandato entrado por un usuario, un script de shell o un mandato de archivo de proceso por lotes.

TP invocados

El TP invocado puede iniciarlo un usuario, puede iniciarlo Communications Server para Linux, automáticamente o puede iniciarlo una aplicación de servidor de TP automáticamente. Cuando el administrador del sistema configura un TP invocado, el administrador del sistema debe especificar si el TP se inicia automáticamente o es iniciado por el usuario.

TP invocados iniciados por el usuario

Si se configura un TP invocado para ser iniciado por un usuario, éste puede iniciar el TP invocado antes o después del TP que invoca. Un TP iniciado de esta manera se denomina TP iniciado por el operador con cola:

- Si el usuario inicia en primer lugar el TP que invoca y no inicia el TP invocado antes de alcanzar el valor de tiempo de espera para iniciar el TP (consulte "Valores de tiempo de espera para TP invocados" en la página 61), la petición Allocate entrante falla.
- Si el usuario inicia el TP invocado antes de que el TP que invoca emita el verbo [MC_]ALLOCATE, el TP invocado espera hasta que llega la petición Attach del TP que invoca, o hasta que se alcanza el valor de tiempo de espera de RECEIVE_ALLOCATE (consulte "Valores de tiempo de espera para TP invocados" en la página 61).

TP invocados: iniciados automáticamente por el gestor de conexiones de Communications Server para Linux

Un TP invocado puede configurarse para iniciarse automáticamente en una de las siguientes condiciones:

- La primera vez que la LU que da servicio al TP invocado recibe una petición Attach (mensaje SNA de la LU remota que contiene la petición de asignación). Un TP que se inicia de esta manera se denomina TP de inicio automático con cola.

  Si el TP invocado no está en ejecución, la primera petición Allocate entrante lo inicia; se retiene una respuesta a la petición Allocate entrante hasta que se ejecuta el verbo RECEIVE_ALLOCATE en el TP invocado (o hasta que se excede un tiempo de espera; consulte "Valores de tiempo de espera para TP invocados" en la página 61). En ese momento, APPC asigna un identificador de conversación, que se devuelve a los dos TP como identificador de la conversación.

  Si el TP invocado ya está en ejecución, la petición Attach se coloca en la cola hasta que el TP invocado emite otro verbo RECEIVE_ALLOCATE o hasta que finaliza su ejecución y puede reiniciarse (o hasta que se excede un tiempo de espera; consulte "Valores de tiempo de espera para TP invocados" en la página 61).
- Cada vez que la LU que da servicio al TP invocado recibe una petición Attach. Se carga una nueva instancia del programa y se inicia con cada petición Attach entrante. Un TP que se inicia de esta manera se denomina TP de inicio automático sin cola.

  La petición Attach se coloca en la cola hasta que se ejecuta el verbo RECEIVE_ALLOCATE del TP invocado (o hasta que se excede un tiempo de espera; consulte "Valores de tiempo de espera para TP invocados" en la página 61). Cuando se ejecuta el verbo RECEIVE_ALLOCATE, APPC asigna un identificador de conversación, que se devuelve a los dos TP como identificador de la conversación.

Tras finalizar una conversación, el TP invocado puede finalizar, o puede emitir otro RECEIVE_ALLOCATE. En el caso de los programas usados con frecuencia, éste es un modo de evitar que el rendimiento general se vea afectado por el hecho de iniciar una nueva instancia del programa para cada conversación. Cada vez que se recibe una petición Attach para un TP de inicio automático sin cola, Communications Server para Linux comprueba si ya existe un verbo RECEIVE_ALLOCATE pendiente de una instancia de este TP. Si es así, este TP se utiliza para la conversación entrante; de lo contrario, Communications Server para Linux inicia una nueva instancia del programa.

TP invocados iniciados automáticamente por una aplicación de servidor de TP

AIX, LINUX

Cuando una petición Attach llega al nodo Communications Server para Linux, Communications Server para Linux distribuye las peticiones Attach a las aplicaciones de servidor de TP que se han registrado para recibir las peticiones Attach. El proceso que utiliza Communications Server para Linux para direccionar las peticiones Attach a un servidor de TP adecuado consta de las siguientes fases:

1. Una o varias aplicaciones se registran para recibir peticiones Attach para nombres de TP y LU. Una aplicación de servidor de TP puede utilizar un comodín para especificar el ámbito de las peticiones Attach que el servidor de TP puede recibir. Una aplicación de servidor de TP puede utilizar un comodín para cualquiera de los datos siguientes:
   - Alias de LU local
   - Nombre de TP
   - Nombre de LU asociada completamente calificado, que puede utilizar un comodín para cualquiera de los datos siguientes:
     - Nombre de red parcial
     - Nombre de red completo
     - Nombre de red parcial seguido del nombre de CP
     - Nombre de LU asociada completamente calificado

   Sólo puede registrarse una única aplicación de servidor de TP para una combinación de TP y LU determinada, incluyendo los comodines. Por ejemplo, una aplicación de servidor de TP puede registrar TPNAME1 y *, mientras que el mismo TP u otra aplicación de servidor de TP registra TPNAME1 y LUNAME1. Este tipo de registro está permitido, pero dos aplicaciones de servidor de TP no pueden registrar TPNAME1 y LUNAME1. El segundo intento de registro fallará.
2. Cuando una petición Attach llega a Communications Server para Linux, Communications Server para Linux intenta encontrar la aplicación de servidor de TP cuyo registro coincida más exactamente con el nombre de TP, el alias de LU y el nombre de LU completamente calificado recibidos en la petición Attach. Las coincidencias se examinan para encontrar la más exacta en el siguiente orden:
   a. Coincidencia de nombre de TP
   b. Coincidencia de alias de LU
   c. Coincidencia de nombre de LU asociada completamente calificado
   d. Coincidencia de nombre comodín de LU asociada completamente calificado

Cuando Communications Server para Linux encuentra una coincidencia, entrega la petición Attach a la aplicación de servidor de TP. El servidor de TP dispone de las siguientes opciones:

- Rechazar la petición Attach, en cuyo caso Communications Server para Linux devuelve la petición Attach al TP que invoca e incluye un código de error proporcionado por la aplicación de servidor de TP
- Aceptar la petición Attach, en cuyo caso Communications Server para Linux informa al TP que invoca de que la petición Attach ha sido aceptada

3. Si no se encuentra ninguna coincidencia después de probar todas las combinaciones de los criterios de búsqueda expuestos anteriormente, Communications Server para Linux rechaza la petición Attach y la devuelve al TP que invoca con el código de error pertinente.

Valores de tiempo de espera para TP invocados

La configuración de Communications Server para Linux especifica dos valores de tiempo de espera que definen cuánto tiempo espera APPC para establecer una conversación entre dos TP, como se indica a continuación:

Tiempo de espera para iniciar un TP
Este valor indica cuánto tiempo permanece una petición Attach en la cola en espera de que se inicie el TP invocado y se emita el verbo RECEIVE_ALLOCATE. Si no se emite RECEIVE_ALLOCATE dentro de este espacio de tiempo, el verbo [MC_]ALLOCATE del TP que invoca falla. Este tiempo de espera está definido en la configuración de la LU local utilizada por el TP.

Tiempo de espera para dar servicio a TP
Este valor indica cuánto tiempo un verbo RECEIVE_ALLOCATE emitido por el TP invocado espera a que se reciba una petición Attach del TP que invoca. Si no se recibe ninguna petición Attach en este espacio de tiempo, el verbo RECEIVE_ALLOCATE del TP invocado falla. La configuración puede especificar uno de los datos siguientes:

Tiempo de espera indefinido
RECEIVE_ALLOCATE espera indefinidamente.

Tiempo de espera cero
RECEIVE_ALLOCATE falla, a menos que ya se haya recibido la petición Attach.

Tiempo de espera definido
Se proporciona un valor de tiempo de espera específico.

Este tiempo de espera está definido para el TP invocado en el archivo de datos de TP invocable de Communications Server para Linux.

Si desea obtener información sobre la configuración de los TP invocados, consulte la publicación *Communications Server para Linux - Guía de administración*.

Sesiones LU-LU

Una sesión LU-LU es una conexión lógica entre dos LU. Las conversaciones entre TP tienen lugar dentro de las sesiones. Una conversación puede utilizar una sesión a la vez; muchas conversaciones pueden volver a utilizar la misma sesión en serie.

Communications Server para Linux permite que una LU de tipo 6.2 tenga sesiones múltiples (dos o más sesiones simultáneas con diferentes LU asociadas) y sesiones paralelas (dos o más sesiones simultáneas con la misma LU asociada).

Durante la configuración, el administrador del sistema determina cuántas sesiones soporta una LU determinada y si la LU soporta sesiones paralelas.

Contienda

Cuando ambas LU intentan asignar una conversación en la misma sesión a la vez, una debe ganar (la ganadora de la contienda) y otra debe perder (la perdedora de la contienda). La LU ganadora de la contienda y la LU perdedora de la contienda se determinan al establecer la sesión.

En una sesión, la LU perdedora de la contienda debe solicitar permiso a la LU ganadora de la contienda antes de asignar una conversación. La ganadora de la contienda puede otorgarle permiso o no. La LU ganadora de la contienda simplemente asigna una conversación cuando lo desea.

Durante la configuración, el administrador del sistema puede definir modalidades. Una modalidad es un conjunto de características de red. Entre las características que el administrador del sistema puede especificar en una definición de modalidad está el número de sesiones ganadoras de la contienda y perdedoras de la contienda para la LU local y la LU asociada que utilizan la modalidad. (El TP que emite el verbo [MC_]ALLOCATE especifica una modalidad, una LU local y una LU asociada.)

Conversaciones básicas

Los TP de servicio normalmente utilizan las conversaciones básicas. Los TP de servicio son programas que proporcionan servicios a otros programas locales. Son más complejas que las conversaciones correlacionadas pero ofrecen un mayor control sobre la transmisión y el manejo de datos a un programador de LU 6.2 experimentado. Este apartado resume las características de las conversaciones básicas en los aspectos en que difieren de las conversaciones correlacionadas.

Registros lógicos

En una conversación básica, los datos se envían en forma de registros lógicos. Un registro lógico es un registro que tiene la sintaxis de corriente de datos general (GDS) descrita en este apartado. Para ver más información sobre la sintaxis GDS, consulte la publicación de IBM *SNA Formats*.

El TP emisor debe dar formato a los datos en registros lógicos y el TP receptor debe decodificar los registros lógicos en datos que puedan utilizarse. Un TP puede enviar varios registros lógicos con un solo verbo SEND_DATA, o puede enviar un solo registro lógico en varias partes (llamadas segmentos) con varios verbos SEND_DATA. Un TP puede recibir varios registros lógicos con un solo verbo de recepción (RECEIVE_AND_WAIT, RECEIVE_IMMEDIATE o RECEIVE_AND_POST), o puede recibir un solo registro lógico en varias partes con varios verbos de recepción.

Si un registro lógico es un solo registro, consta de los campos siguientes:

- Un campo de longitud de registro (LL) de 2 bytes.
- Un campo de identificador (ID) de GDS de 2 bytes (por ejemplo, `0x12FF` identifica los datos como datos de aplicación).

- Un campo de datos cuya longitud puede estar comprendida entre 0 y 32.763 bytes.

Los 4 primeros bytes se denominan LLID.

Si un registro lógico tiene varias partes, la primera parte tiene el mismo formato que un solo registro y todas las partes posteriores constan de estos campos:

- Un campo de longitud de registro (LL) de 2 bytes.
- Un campo de datos cuya longitud puede estar comprendida entre 0 y 32.765 bytes.

El valor hexadecimal del campo LL incluye los 2 bytes del campo LL (y los 2 bytes del campo ID, si existe). Por ejemplo, una GDS de una sola parte sin bytes de datos tiene el valor `0x0004` para su campo LL. El campo LL debe tener el formato de más significativo a menos significativo, y no el formato de bytes intercambiados. Por ejemplo, una longitud de 230 bytes se representa como `0x00E6`, y no como `0xE600`.

El bit 0 del byte 0 del campo LL (el bit más significativo) se utiliza para indicar la continuación de longitud (segmentación). El ejemplo siguiente muestra 10 bytes de datos (cada byte de datos tiene el valor `DD`) divididos en 3 segmentos de GDS. Los segmentos primero y segundo contienen 4 bytes de datos cada uno y el último segmento contiene 2 bytes de datos.

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

Los siguientes valores del campo LL no son válidos (excepto para enviar una cabecera PS como se describe en "Envío de cabeceras PS en registros lógicos"):

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

En una conversación correlacionada, el TP emisor envía un registro de datos cada vez, y el TP receptor recibe un registro de datos cada vez. No se necesita ninguna conversión de registros para los TP.

Envío de cabeceras PS en registros lógicos

Si el nivel de sincronización de la conversación es `AP_SYNCPT`, la aplicación puede necesitar enviar y recibir datos en forma de cabeceras PS. El gestor de puntos de sincronización se encarga de configurar las cabeceras PS adecuadas para enviarlas a la aplicación asociada, según las funciones de punto de sincronización que requiere la aplicación, y de llevar a cabo el proceso de punto de sincronización necesario, según las cabeceras PS que recibe de la aplicación asociada.

Un campo LL con el valor `0x0001` indica que los datos son una cabecera PS; la aplicación emisora debe especificar un campo LL con el valor `0x0001` en lugar de especificar la longitud del campo de datos y la aplicación receptora debe interpretar los datos como una cabecera PS si recibe un campo LL con el valor `0x0001`. Si el nivel de sincronización de la conversación no es `AP_SYNCPT`, el valor `0x0001` no es un campo LL válido y será rechazado.

Información de errores y finalizaciones anormales

En una conversación básica, un TP puede indicar si un error o una finalización anormal (terminación anormal de un programa) es causado por un TP de servicio o por un programa que utiliza el TP de servicio. Esto permite que dos TP de servicio que se comunican distingan entre los errores que pueden haber causado y los errores que pueden haber sido ocasionados por los programas a los que dan servicio.

Anotación de error

En caso de error o finalización anormal de una conversación básica, un TP puede enviar un mensaje de error, en forma de variable de anotaciones de error de corriente de datos general (GDS), al archivo de anotaciones local y a la LU asociada.

Tiempos de espera frente a errores críticos

En una conversación básica, un TP puede indicar si una finalización anormal ha sido causada por un tiempo de espera excedido o por un error crítico.

Desarrollo de servidores de TP

AIX, LINUX

Utilice las siguientes directrices operativas para desarrollar servidores de TP:

1. Utilice REGISTER_TP_SERVER para registrar la aplicación como servidor de TP. El verbo REGISTER_TP_SERVER proporciona la dirección de una función callback utilizada en las notificaciones de petición Attach posteriores.
2. Utilice REGISTER_TP para registrar los nombres de TP y las LU locales y remotas para los que el servidor de TP quiere procesar peticiones Attach. El servidor de TP puede utilizar comodines para los nombres de TP y los nombres de LU, de manera que pueda elegir procesar las peticiones Attach de un TP por un par determinado de LU o procesar las peticiones Attach de todos los TP por todas las LU, o cualquier combinación de estas condiciones.
3. Utilice QUERY_ATTACH (con el identificador exclusivo recibido al efectuar la rutina callback de notificación después de recibir una petición Attach para la LU o el TP registrado) para consultar los parámetros de la petición Attach para decidir si se procesa la petición Attach y cómo hacerlo. El servidor de TP puede rechazar la petición Attach utilizando REJECT_ATTACH o aceptar la petición Attach con ACCEPT_ATTACH e iniciar una aplicación adecuada para procesar la petición Attach.
4. Emita la llamada RECEIVE_ALLOCATE estándar para recuperar la petición Attach. El parámetro *dload_id* reservado previamente se utiliza para especificar el identificador exclusivo de la petición Attach.
5. Utilice ABORT_ATTACH para cancelar cualquier proceso posterior si, después de emitir ACCEPT_ATTACH, se detecta un error.
6. Deshaga el registro del servidor de TP para cualquiera de los TP y las LU que se han registrado anteriormente utilizando UNREGISTER_TP.
7. Deshaga el registro de la aplicación como servidor de TP utilizando el verbo REGISTER_TP_SERVER.

Responsabilidades del servidor de TP

Cuando un servidor de TP maneja una petición Attach, el servidor de TP hereda una serie de responsabilidades que normalmente son realizadas por el gestor de peticiones Attach de Communications Server para Linux. Estas responsabilidades incluyen las siguientes acciones:

- Iniciar programas de transacciones para manejar las conversaciones si un TP está configurado para ser iniciado automáticamente.
- Manejar la seguridad de conversación, que incluye el análisis de los datos de petición Attach devueltos en QUERY_ATTACH.
- Transmitir información desde la petición Attach hacia el TP que el servidor de TP inicia para procesar la conversación resultante.

Servidor de TP por omisión

Communications Server para Linux proporciona un servidor de TP por omisión, **snatpsrvd**, que está instalado en todos los sistemas. Este servidor de TP utiliza el archivo **sna_tps** como el archivo fuente para la configuración de los TP que puede cargar. Otros servidores de TP pueden modificar y utilizar este archivo utilizando verbos DEFINE_TP_LOAD_INFO. Si desea obtener más información sobre el verbo DEFINE_TP_LOAD_INFO, consulte la publicación *Communications Server for Linux NOF Programmer's Guide*. APPC proporciona el parámetro *tp_file_updates* en un verbo REGISTER_TP para notificar a un servidor de TP que se ha efectuado un cambio en el archivo **sna_tps** y que puede emprender las acciones necesarias.

Desarrollo de TP portables

A continuación, se proporcionan directrices para desarrollar TP portables a otros entornos de sistema operativo:

- Incluya el archivo de cabecera APPC sin ningún prefijo de nombre de vía de acceso. Utilice opciones de inclusión en el compilador para localizar el archivo (consulte el apartado correspondiente a su sistema operativo que encontrará en una parte anterior de este capítulo). Esto permite utilizar el TP en un entorno con un sistema de archivos diferente.
- Utilice los nombres de constantes simbólicas para los valores de parámetro y códigos de retorno, no los valores numéricos mostrados en el archivo de cabecera; esto garantiza que se utilizará el valor correcto, independientemente de cómo estén almacenados estos valores en la memoria.
- Incluya una comprobación para otros códigos de retorno distintos a los correspondientes a su sistema operativo actual (por ejemplo, utilizando el caso "default" en una sentencia switch) y especifique el diagnóstico adecuado.
- Utilice el punto de entrada asíncrono.
- El verbo [MC_]RECEIVE_AND_POST no puede utilizarse si el TP debe ser completamente portable. Si utiliza este verbo, deberá volver a escribir algunas secciones del TP para utilizarlas en otros entornos. Puede restringir el uso de este verbo a unas cuantas rutinas específicas para facilitar las modificaciones.
- El formato ampliado del verbo RECEIVE_ALLOCATE es específico de la implementación de APPC de Communications Server para Linux y las demás implementaciones de APPC no lo proporcionan. Si utiliza este formato del verbo, deberá volver a escribir algunas secciones del TP para utilizarlas en otros

entornos. Puede restringir el uso de esta característica a unas cuantas rutinas específicas para facilitar las modificaciones.

Capítulo 3. Verbos de control APPC

Este capítulo contiene una descripción de cada uno de los verbos de control APPC. Para cada verbo se proporciona la siguiente información:

- Definición del verbo.
- Estructura que define el bloque de control de verbos (VCB) utilizado por el verbo. La estructura está definida en el archivo de cabecera APPC **/usr/include/sna/appc_c.h** (AIX), **/opt/ibm/sna/include/appc_c.h** (Linux) o **sdk/winappc.h** (Windows). Los parámetros que empiezan por *reserv* están reservados.
- Parámetros (campos del VCB) suministrados a APPC y devueltos por APPC. Para cada parámetro, se proporciona la siguiente información:
  - Descripción
  - Valores posibles
  - Información adicional
- Estado o estados de conversación en que el verbo puede emitirse.
- Estado o estados a que puede cambiar la conversación al volver el verbo. Las condiciones que no provocan un cambio de estado no se especifican. Por ejemplo, las comprobaciones de parámetros y las comprobaciones de estado no provocan un cambio de estado.
- Información adicional sobre el uso del verbo.

La mayoría de los parámetros suministrados a APPC y devueltos por APPC son valores hexadecimales. Para simplificar la codificación, estos valores están representados por constantes simbólicas significativas definidas en el archivo de cabecera **values_c.h**, incluido en el archivo de cabecera APPC **appc_c.h**. Por ejemplo, el parámetro *opcode* del verbo TP_STARTED es un valor hexadecimal representado por la constante simbólica `AP_TP_STARTED`. El archivo **values_c.h** también incluye definiciones de tipos de parámetro como `AP_UINT16`, que se utilizan en los VCB APPC.

Es importante que utilice la constante simbólica y no el valor hexadecimal cuando establezca valores para los parámetros suministrados o cuando compruebe los valores de los parámetros devueltos. Esto se debe a que los diferentes sistemas operativos guardan estos valores en la memoria de modo diferente, por lo que es posible que el valor mostrado no tenga el formato que su sistema reconoce.

WINDOWS

Para Windows, las constantes para los valores de los parámetros suministrados y devueltos están definidas en el archivo de cabecera APPC de Windows, **winappc.h**.

La notación "[MC_]verbo" se refiere tanto a la forma correlacionada como a la básica de un verbo APPC. Por ejemplo, [MC_]SEND_DATA se refiere a los verbos MC_SEND_DATA y SEND_DATA.

Nota: Los VCB APPC contienen muchos parámetros marcados como "reservados"; el software Communications Server para Linux utiliza internamente algunos de ellos y otros no se utilizan en esta versión, aunque pueden utilizarse en versiones futuras. Su aplicación no debe intentar acceder a ninguno de estos parámetros reservados, sino que debe establecer todo el contenido del VCB en cero para garantizar que todos estos parámetros tengan el valor cero, antes de que establezca otros parámetros utilizados por el verbo. Esto garantiza que Communications Server para Linux no interpretará de forma incorrecta ninguno de estos parámetros de uso interno y que su aplicación continuará funcionando con versiones posteriores de Communications Server para Linux en que estos parámetros pueden utilizarse para proporcionar nuevas funciones.

Para establecer el contenido del VCB en cero, utilice `memset`:

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

Los verbos de control se describen en el orden siguiente:

TP_STARTED
TP_ENDED
RECEIVE_ALLOCATE

AIX, LINUX

GET_LU_STATUS

GET_TP_PROPERTIES

AIX, LINUX

SET_TP_PROPERTIES

TP_STARTED

El verbo TP_STARTED es emitido por el TP que invoca. Notifica a APPC que el TP está iniciándose, y especifica la LU local que utilizará.

Si el TP utiliza LU dependientes para varias conversaciones simultáneas, debe emitir un verbo TP_STARTED (seguido de [MC_]ALLOCATE) diferente para cada conversación, a fin de obtener una LU diferente para cada conversación; esto es debido a que cada una de las LU dependientes sólo puede soportar una conversación a la vez.

Como respuesta a este verbo, APPC genera un identificador de TP para el TP que invoca. Este identificador es un parámetro necesario para los verbos APPC posteriores emitidos por el TP que invoca.

Estructura del VCB: TP_STARTED

AIX, LINUX

La definición de la estructura del VCB para el verbo TP_STARTED es la siguiente:

```
typedef struct tp_started
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reservado    */
  unsigned char    format;                    /* Reservado            */
  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;            /* Reservado            */
  unsigned char  enable_pool;            /* Reservado            */
  unsigned char  pip_dlen;               /* Reservado            */
} TP_STARTED;
```

Estructura del VCB: TP_STARTED (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo TP_STARTED es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_TP_STARTED

lu_alias

Alias por el que se conoce la LU local en el TP local. Este nombre debe coincidir con un alias de LU establecido durante la configuración.

Este parámetro es una cadena de caracteres ASCII de 8 bytes. Puede constar de cualquiera de los siguientes caracteres:

- Letras en mayúsculas
- Números del 0 al 9
- Blancos
- Los caracteres especiales $, #, % y @

El primer carácter de esta cadena no puede ser un blanco (salvo que toda la cadena conste de blancos).

Si el alias de LU tiene menos de ocho caracteres, rellénelo por la derecha con blancos ASCII (0x20).

En función de la configuración, puede especificar que la aplicación utilice una LU local por omisión (compruébelo con el administrador del sistema); para ello establezca *lu_alias* en una cadena de ocho ceros binarios. A fin de proporcionar compatibilidad con otras implementaciones de APPC, Communications Server para Linux también admite una cadena de ocho blancos ASCII para indicar la LU por omisión; sin embargo, las nuevas aplicaciones deben utilizar ceros binarios.

tp_name

Nombre del TP local. Los ocho primeros caracteres de este nombre se convierten a ASCII y los utilizan los programas de administración de Communications Server para Linux para identificar el TP en una lista de TP APPC en ejecución.

Este parámetro es una cadena de caracteres EBCDIC de 64 bytes sensible a las mayúsculas y minúsculas. El parámetro *tp_name* normalmente consta de caracteres del juego de caracteres EBCDIC de tipo AE (salvo que sea el nombre de un TP de servicio). Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

Si el nombre de TP tiene menos de 64 bytes, utilice blancos EBCDIC (0x40) para rellenarlo por la derecha.

El convenio SNA para denominar un TP de servicio es una excepción al parámetro *tp_name* normal; el nombre consta de hasta cuatro caracteres, el primero de los cuales es un byte hexadecimal entre 0x00 y 0x3F. Los demás caracteres pertenecen al juego de caracteres EBCDIC de tipo AE.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc

AP_OK

tp_id Identificador del TP local.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc

AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_LU_ALIAS
El valor del parámetro *lu_alias* no es válido.

AIX, LINUX

AP_INVALID_FORMAT
El parámetro reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

TP_STARTED debe ser el primer verbo APPC emitido por el TP que invoca. Por lo tanto, no hay conversaciones activas ni estados de conversación.

Un programa APPC único puede emitir más de un verbo TP_STARTED. Cada verbo crea un TP APPC lógicamente diferente, a pesar de que todos se ejecutan en el mismo proceso.

Cambio de estado

No aplicable (no se ha iniciado ninguna conversación, por lo tanto no hay ningún estado de conversación).

TP_ENDED

El verbo TP_ENDED es emitido por los TP que invocan y los TP invocados. Notifica a APPC que el TP está finalizando. Como respuesta a este verbo, APPC libera los recursos utilizados por el TP.

Si una conversación APPC todavía está en proceso, TP_ENDED realiza la función del verbo [MC_]DEALLOCATE con *dealloc_type* definido en AP_ABEND (para una conversación correlacionada) o AP_ABEND_PROG (para una conversación básica). Una vez que se ha ejecutado este verbo, el identificador de TP y el identificador de conversación ya no son válidos; el TP no puede emitir ningún verbo APPC más para la conversación.

Estructura del VCB: TP_ENDED

AIX, LINUX

La definición de la estructura del VCB para el verbo TP_ENDED es la siguiente:

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

Estructura del VCB: TP_ENDED (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo TP_ENDED es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_TP_ENDED

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED para el TP que invoca o por el verbo RECEIVE_ALLOCATE para el TP invocado.

type Especifica cómo finalizar el TP. Los valores posibles son:

AP_SOFT
Si hay conversaciones APPC activas, APPC lleva a cabo la función del verbo [MC_]DEALLOCATE para cada conversación, a fin de informar al TP asociado de que la conversación ha finalizado. El verbo TP_ENDED no vuelve hasta que [MC_]DEALLOCATE ha finalizado.

AP_HARD
APPC cierra todas las sesiones utilizadas por el TP, y TP_ENDED vuelve inmediatamente.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

APPC no devuelve un parámetro *secondary_rc* si el verbo se ejecuta de forma correcta.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_TP_ID
APPC no reconoce el parámetro *tp_id* como un identificador de TP asignado.

AP_BAD_TYPE
El valor del parámetro *type* no es válido.

AIX, LINUX

AP_INVALID_FORMAT
El parámetro reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback,

utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

La conversación (o las conversaciones, si el TP participa en más de una) puede encontrarse en cualquier estado cuando el TP emite este verbo.

Cambio de estado

Después de una ejecución correcta (el valor de *primary_rc* es AP_OK), no hay ningún estado APPC.

RECEIVE_ALLOCATE

El verbo RECEIVE_ALLOCATE es emitido por el TP invocado. Confirma que el TP invocado está preparado para empezar una conversación con el TP que invoca, que ha emitido el verbo [MC_]ALLOCATE.

Como respuesta a este verbo, APPC establece una conversación entre los dos TP, genera un identificador de TP para el TP invocado y genera un identificador de conversación. Estos identificadores son parámetros necesarios para los verbos APPC posteriores.

AIX, LINUX

La implementación de APPC de Communications Server para Linux proporciona el formato estándar del verbo RECEIVE_ALLOCATE, del mismo modo que lo proporcionan otras implementaciones de APPC, y un formato ampliado que permite que la aplicación reciba peticiones Attach entrantes de una LU local concreta. Los dos formatos se describen conjuntamente en este apartado, con referencias a "formato estándar" y "formato ampliado" cuando corresponde.

WINDOWS

El formato ampliado de RECEIVE_ALLOCATE no se proporciona en Windows.

Estructura del VCB: RECEIVE_ALLOCATE

AIX, LINUX

La definición de la estructura del VCB para el verbo RECEIVE_ALLOCATE es la siguiente:

```
typedef struct receive_allocate
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reservado    */
  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;
```

Estructura del VCB: RECEIVE_ALLOCATE (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo RECEIVE_ALLOCATE es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode `AP_RECEIVE_ALLOCATE` (formato estándar)

AIX, LINUX

`AP_RECEIVE_ALLOCATE_EX` (formato ampliado)

tp_name

Nombre del TP local. APPC hace coincidir este nombre con el nombre de TP especificado en la petición Attach entrante, que genera el verbo [MC_]ALLOCATE en el TP que invoca. Si Communications Server para Linux va a iniciar automáticamente el TP, este nombre de TP debe coincidir con un nombre de TP especificado en el archivo de datos de TP invocable.

Este parámetro es una cadena de caracteres EBCDIC de 64 bytes sensible a las mayúsculas y minúsculas. El parámetro *tp_name* normalmente consta de caracteres del juego de caracteres EBCDIC de tipo AE. Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

Si el nombre de TP tiene menos de 64 bytes, utilice blancos EBCDIC (`0x40`) para rellenarlo por la derecha.

El convenio SNA para denominar un TP de servicio es una excepción a lo mencionado anteriormente; el nombre consta de hasta 4 caracteres, el primero de los cuales es un byte hexadecimal entre `0x00` y `0x3F`. Los demás caracteres pertenecen al juego de caracteres EBCDIC de tipo AE.

AIX, LINUX

El TP puede especificar que aceptará peticiones Attach entrantes para cualquier nombre de TP definiendo este parámetro en 64 espacios EBCDIC. Para ver más información sobre cómo Communications Server para Linux direcciona las peticiones Attach entrantes a los TP, consulte "Direccionamiento de peticiones Attach entrantes" en la página 81.

lu_alias

Para el formato estándar de RECEIVE_ALLOCATE, este parámetro está reservado; defínalo en una cadena nula. Si Communications Server para Linux inicia automáticamente el TP, la entrada de este TP en el archivo de datos de TP invocable no debe especificar un alias de LU.

Para el formato ampliado de RECEIVE_ALLOCATE, especifique el alias de la LU local de la que el TP aceptará peticiones Attach entrantes. Se trata de una cadena de caracteres ASCII. Si Communications Server para Linux inicia automáticamente el TP, este alias de LU debe coincidir con el alias de LU especificado para el TP en el archivo de datos de TP invocable.

Para indicar que el TP aceptará peticiones Attach entrantes de cualquier LU local, defina este parámetro en ocho espacios ASCII. Si Communications Server para Linux inicia automáticamente el TP, la entrada de este TP en el archivo de datos de TP invocable no debe especificar un alias de LU.

Para ver más información sobre cómo Communications Server para Linux direcciona las peticiones Attach entrantes a los TP, consulte "Direccionamiento de peticiones Attach entrantes" en la página 81.

dload_id

Identificador de la petición Attach proporcionado por una aplicación de servidor de TP. Si el TP no trabaja en colaboración con un servidor de TP, establezca este campo en todo ceros.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc

AP_OK

AIX, LINUX

tp_name

Si la aplicación ha especificado un nombre de TP que sólo consta de espacios, Communications Server para Linux devolverá el nombre de TP suministrado por el TP que invoca en el verbo [MC_]ALLOCATE.

tp_id Identificador del TP local.

conv_id
Identificador de conversación.

Este valor identifica la conversación que APPC ha establecido entre los dos TP asociados.

sync_level
Nivel de sincronización de la conversación.

Este parámetro determina si los TP pueden solicitar la confirmación de recepción de datos y confirmar la recepción de datos. Los valores posibles son:

AP_CONFIRM_SYNC_LEVEL
Los TP asociados pueden utilizar el proceso de confirmación en esta conversación.

AP_SYNCPT
Los TP pueden utilizar las funciones de punto de sincronización de LU 6.2 en esta conversación. Para ver más información, consulte "Soporte de punto de sincronización" en la página 23.

AP_NONE
El proceso de confirmación no se utiliza en esta conversación.

conv_type
Tipo de conversación elegido por el TP asociado, utilizando el verbo [MC_]ALLOCATE. Los valores posibles son:
`AP_BASIC_CONVERSATION`
`AP_MAPPED_CONVERSATION`

user_id Si el TP asociado ha definido el parámetro *security* del verbo [MC_]ALLOCATE en `AP_PGM` o `AP_SAME`, este parámetro contiene el identificador de usuario enviado desde el TP asociado. El identificador de usuario es una cadena de caracteres EBCDIC de tipo AE, rellenada por la derecha con espacios EBCDIC hasta 10 caracteres si es necesario. Si el TP asociado ha definido el parámetro de seguridad del verbo [MC_]ALLOCATE en `AP_NONE`, este parámetro se define en 10 blancos EBCDIC.

lu_alias
Alias por el que se conoce la LU local en el TP local. Se trata de una cadena de caracteres ASCII.

plu_alias
Alias por el que se conoce la LU asociada (de la que se ha recibido la petición Allocate entrante) en el TP local. Se trata de una cadena de caracteres ASCII.

mode_name
Nombre de modalidad especificado por el verbo [MC_]ALLOCATE en el TP asociado. Este es el nombre de un conjunto de características de red definidas durante la configuración. Este nombre es una cadena de caracteres EBCDIC de tipo A.

conv_group_id
Identificador de grupo de conversación de la sesión que utiliza la nueva conversación.

fqplu_name
Nombre de la LU asociada completamente calificado.

Este parámetro contiene el nombre de red, un punto EBCDIC y el nombre de LU asociada. Cada uno de los dos nombres es una cadena de caracteres EBCDIC de 8 bytes que puede constar de caracteres del juego de caracteres EBCDIC de tipo A como los siguientes:

- Letras en mayúsculas
- Números del 0 al 9
- Los caracteres especiales $, # y @

AIX, LINUX

pip_incoming
Especifica si el TP asociado ha suministrado datos PIP (parámetros de inicialización de programa) en la petición [MC_]ALLOCATE. Los valores posibles son:

AP_YES El TP asociado ha suministrado datos PIP. El TP local debe emitir uno de los verbos [MC_]RECEIVE para recibir los datos; el primer registro de datos recibido serán los datos PIP.

AP_NO El TP asociado no ha suministrado datos PIP.

duplex_type
Tipo de dúplex de la nueva conversación. Los valores posibles son:
AP_HALF_DUPLEX
AP_FULL_DUPLEX

password
Si el TP asociado ha definido el parámetro *security* del verbo [MC_]ALLOCATE en AP_PGM, este parámetro contiene la contraseña especificada por el TP asociado en el verbo [MC_]ALLOCATE. La contraseña es una cadena de caracteres EBCDIC de tipo AE, rellenada por la derecha con espacios EBCDIC hasta 10 caracteres si es necesario. Si el TP asociado ha definido el parámetro *security* del verbo [MC_]ALLOCATE en AP_NONE o AP_SAME, este parámetro se define en 10 blancos EBCDIC.

Por razones de seguridad, Communications Server para Linux no guarda la contraseña una vez que se ha devuelto en el verbo RECEIVE_ALLOCATE. Si la aplicación necesita comprobar este parámetro, debe utilizar el valor devuelto en RECEIVE_ALLOCATE; la contraseña no se devuelve en ninguno de los verbos posteriores. La aplicación puede recuperar el identificador de usuario en cualquier momento de la conversación emitiendo el verbo GET_TP_PROPERTIES.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AIX, LINUX

AP_BAD_DLOAD_ID
El valor especificado para el parámetro *dload_id* no se reconoce.

AP_INVALID_FORMAT
El parámetro *format* está definido en un valor que no es válido.

AP_INVALID_LU_ALIAS
El parámetro *lu_alias* contiene un carácter que no es válido. (Este valor se devuelve para el formato ampliado de RECEIVE_ALLOCATE y no para el formato estándar.)

AIX, LINUX

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc

AP_ALLOCATE_NOT_PENDING
APPC no ha encontrado ninguna petición Allocate entrante (desde el TP que invoca) para hacer coincidir con la combinación de nombre de TP, alias de LU o ambos que ha suministrado por el verbo RECEIVE_ALLOCATE. El verbo RECEIVE_ALLOCATE esperaba la petición Allocate entrante y finalmente ha excedido el tiempo de espera. Para ver más información, consulte “Cómo evitar esperas” en la página 81 y “Direccionamiento de peticiones Attach entrantes” en la página 81.

Este código de retorno también se produce si intenta iniciar un TP que está definido en el archivo de datos de TP invocable como TP sin cola. Communications Server para Linux inicia automáticamente un TP sin cola como respuesta a una petición Attach entrante; si intenta iniciarlo manualmente, el verbo RECEIVE_ALLOCATE falla porque no hay ninguna petición Attach entrante que esté a la espera del TP.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde,

códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Este debe ser el primer verbo APPC emitido por el TP invocado. El estado inicial es Restablecer.

Un solo TP invocado puede emitir varios verbos RECEIVE_ALLOCATE; cada uno inicia un TP APPC lógicamente diferente, a pesar de que todos se ejecutan en el mismo proceso.

Cambio de estado

Si el verbo se ejecuta de forma correcta (el valor de *primary_rc* es AP_OK), el estado cambia a Recibir (en el caso de una conversación semidúplex) o a Enviar-Recibir (en el caso de una conversación dúplex).

Cómo evitar esperas

Si el TP invocado emite un verbo RECEIVE_ALLOCATE y no hay una petición Allocate entrante correspondiente (resultado del verbo [MC_]ALLOCATE emitido por el TP que invoca), el TP invocado espera hasta que llega la petición Allocate entrante o hasta que el verbo excede el tiempo de espera. El valor por omisión consiste en esperar indefinidamente una petición Allocate entrante; puede alterarse temporalmente este valor configurando el TP con un tiempo de espera 0 (cero) (RECEIVE_ALLOCATE falla salvo que ya esté en espera una petición Allocate entrante) o con un valor finito (RECEIVE_ALLOCATE falla salvo que llegue una petición Allocate entrante dentro del tiempo especificado). Para obtener más información consulte la publicación *Communications Server para Linux - Guía de administración*.

Direccionamiento de peticiones Attach entrantes

AIX, LINUX

Si la aplicación no especifica ningún *dload_id* en el verbo RECEIVE_ALLOCATE, puede utilizar los parámetros *tp_name* y *lu_alias* de RECEIVE_ALLOCATE para especificar el rango de peticiones Attach entrantes que aceptará. Al especificar un

nombre de TP, se indica que aceptará peticiones Attach entrantes de un TP asociado sólo si el TP asociado ha especificado este nombre de TP en el verbo [MC_]ALLOCATE; al utilizar el formato ampliado de RECEIVE_ALLOCATE y especificar un alias de LU, se indica que aceptará peticiones Attach entrantes sólo si llegan a una determinada LU local de Communications Server para Linux. En cualquier caso, el TP puede especificar un nombre en blanco para indicar que acepta peticiones Attach entrantes para cualquier nombre de TP o desde cualquier LU local.

Communications Server para Linux hace coincidir una petición Attach entrante con el verbo RECEIVE_ALLOCATE en el TP adecuado según el siguiente orden de prioridad:

1. Un TP que especifica un nombre de TP y un alias de LU y ambos coinciden con la petición Attach entrante.
2. Un TP que especifica un nombre de TP que coincide con la petición Attach entrante pero no especifica un alias de LU.
3. Un TP que especifica un alias de LU que coincide con la LU que ha recibido la petición Attach entrante pero no especifica un nombre de TP.
4. Un TP que no especifica un nombre de TP ni un alias de LU. Esta función sólo debe ser utilizada por un único TP en cada máquina Communications Server para Linux; si dos TP emiten el verbo RECEIVE_ALLOCATE sin el nombre de TP ni el alias de LU, no es posible determinar qué TP recibirá la petición Attach entrante.

Si el TP ha utilizado un nombre de TP en blanco, un alias de LU o ambos para aceptar un rango de peticiones Attach entrantes, puede comprobar los parámetros devueltos en este verbo para determinar el nombre de TP especificado en la petición Attach entrante, el alias de LU de la LU que la ha recibido o ambos. Esto significa que puede tener un solo TP para manejar todas las peticiones Attach entrantes, que lleva a cabo el proceso apropiado para cada uno de los nombres de TP, las LU o ambos. Si este TP recibe una petición Attach entrante de un TP asociado no reconocido o no autorizado, puede rechazar la nueva conversación si es necesario emitiendo el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* apropiado.

El TP que acepta la petición Attach entrante puede ser un TP iniciado por el operador que ya ha emitido el verbo RECEIVE_ALLOCATE o un TP de inicio automático indicado en el archivo de datos de TP invocable de Communications Server para Linux. Communications Server para Linux utiliza el nombre de TP, el alias de LU o ambos valores especificados en este archivo a fin de determinar si debe iniciar el TP para establecer una coincidencia con la petición Attach entrante. Si desea obtener información sobre la configuración de los TP invocados, consulte la publicación *Communications Server para Linux - Guía de administración*. El nombre de TP, el alias de LU o ambos valores especificados por un TP de inicio automático en el verbo RECEIVE_ALLOCATE deben coincidir con los especificados en el archivo para que Communications Server para Linux pueda direccionar correctamente la petición Attach entrante.

GET_LU_STATUS

AIX, LINUX

Este verbo se proporciona para los TP de punto de sincronización, que deben comprobar si han perdido la comunicación con sus TP asociados para que puedan volver a sincronizarse si es necesario.

Nota: Si dos o más TP utilizan la misma combinación de LU local y LU asociada, es importante que sólo uno de ellos emita este verbo. Communications Server para Linux mantiene el indicador de cero sesiones para cada par de LU independientemente de los TP que las utilizan y lo restablece cada vez que se emite este verbo. Esto significa que, si el número de sesiones disminuye hasta `0` (cero) y después vuelven a activarse sesiones y dos TP emiten consecutivamente el verbo GET_LU_STATUS, sólo se notificará el número de sesiones cero al primer TP. Si varios TP que utilizan las mismas LU necesitan comprobar el estado de las LU y de las sesiones, deben hacerlo utilizando verbos NOF; consulte la publicación *Communications Server for Linux NOF Programmer's Guide* para ver más información.

Estructura del VCB: GET_LU_STATUS

La definición de la estructura del VCB para el verbo GET_LU_STATUS es la siguiente:

```
typedef struct get_lu_status
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reservado    */
  unsigned char    format;                    /* Reservado            */
  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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode `AP_GET_LU_STATUS`

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

plu_alias

Alias por el que se conoce la LU asociada en el TP local. Se trata de una cadena de caracteres ASCII de 8 bytes.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

active_sess
Especifica el número de sesiones que están activas en este momento entre la LU local y la LU asociada especificada.

zero_sess
Especifica si el número de sesiones activas entre las dos LU ha descendido hasta 0 (cero) en algún momento desde que se ha emitido el último verbo GET_LU_STATUS. Los valores posibles son:

AP_YES El número de sesiones ha descendido hasta 0 (cero).

AP_NO Ha habido al menos una sesión activa todo el tiempo desde que se ha emitido el último verbo.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_INVALID_FORMAT
El parámetro reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los valores posibles son:

primary_rc
AP_COMM_SUBSYSTEM_ABENDED
AP_INVALID_VERB
AP_TP_BUSY

AP_UNEXPECTED_SYSTEM_ERROR

APPC no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

El estado de conversación no cambia con este verbo.

GET_TP_PROPERTIES

El verbo GET_TP_PROPERTIES devuelve información sobre los atributos del TP local y de la LUW (unidad lógica de trabajo) en que participa el TP. Una unidad lógica de trabajo es una transacción entre TP APPC para llevar a cabo una tarea determinada; puede comprender dos TP que se comunican o una secuencia de conversaciones entre varios TP.

Estructura del VCB: GET_TP_PROPERTIES

AIX, LINUX

La definición de la estructura del VCB para el verbo GET_TP_PROPERTIES es la siguiente:

```
typedef struct get_tp_properties
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reservado    */
  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;
```

Estructura del VCB: GET_TP_PROPERTIES (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo GET_TP_PROPERTIES es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_GET_TP_PROPERTIES

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

tp_name
Nombre de TP del TP local, como se ha especificado en el verbo TP_STARTED o RECEIVE_ALLOCATE. Este parámetro es una cadena de caracteres EBCDIC de 64 bytes.

lu_alias
Alias por el que se conoce la LU local en el TP local, como se ha especificado en el verbo TP_STARTED o RECEIVE_ALLOCATE. Se trata de una cadena de caracteres ASCII de 8 bytes.

AIX, LINUX

luw_id LUWID (identificador de unidad lógica de trabajo) no protegido para la

transacción en que participa el TP. El LUWID se asigna en nombre del TP que inicia la transacción, y permite correlacionar las diferentes conversaciones que forman la transacción. El LUWID no protegido se utiliza para correlacionar conversaciones no protegidas (aquellas con un parámetro *sync_level* con el valor `AP_NONE` o `AP_CONFIRM_SYNC_LEVEL`); para los TP que utilizan el proceso de punto de sincronización, existe un LUWID protegido adicional para las conversaciones con un parámetro *sync_level* con el valor `AP_SYNCPT` (para ver más información, consulte Tabla 9).

El LUWID consta de los siguientes parámetros:

luw_id.fq_length
Longitud (1–17 bytes) del nombre de LU completamente calificado asociado con la unidad lógica de trabajo (el parámetro *luw_id.fq_luw_name* especifica el nombre de LU en sí).

luw_id.fq_luw_name
Nombre de LU completamente calificado asociado con la unidad lógica de trabajo. Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC. Consta de un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.

luw_id.instance
Número de instancia asociado con la unidad lógica de trabajo (un número binario de 6 bytes).

luw_id.sequence
Número de secuencia del segmento actual de la unidad lógica de trabajo (un número binario de 2 bytes).

WINDOWS

luw_id LUWID (identificador de unidad lógica de trabajo) correspondiente a la transacción en que participa el TP. El LUWID se asigna en nombre del TP que inicia la transacción, y permite correlacionar las diferentes conversaciones que forman la transacción.

El LUWID es una cadena de 26 bytes que consta de los parámetros que se muestran en la Tabla 9:

Tabla 9. Parámetros de LUWID

Parámetro	Longitud	Descripción
fq_length	1	Longitud (1–17 bytes) del nombre de LU completamente calificado asociado con la unidad lógica de trabajo (el parámetro *fq_luw_name* especifica el nombre de LU en sí).
fq_luw_name	1–17	Nombre de LU completamente calificado asociado con la unidad lógica de trabajo. Este nombre es una cadena EBCDIC que consta de un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A. En este nombre, los espacios no se rellenan; el parámetro *fq_length* especifica el número de bytes del nombre y el parámetro *instance* va inmediatamente después de este número de bytes.
instance	6	Número de instancia asociado con la unidad lógica de trabajo (un número binario de 6 bytes).

Tabla 9. Parámetros de LUWID (continuación)

Parámetro	Longitud	Descripción
sequence	2	Número de secuencia del segmento actual de la unidad lógica de trabajo (un número binario de 2 bytes); siempre se establece en 1.

Si el parámetro *fq_length* indica que el nombre de LU tiene menos de 17 bytes, la longitud total de los parámetros anteriores será inferior a 26 bytes; los bytes restantes se rellenan con espacios EBCDIC.

fqlu_name
Nombre de LU completamente calificado de la LU local asociada al TP. Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC. Consta de un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.

AIX, LINUX

verified Especifica si se ha verificado la seguridad de conversación en esta conversación. Los valores posibles son:

AP_YES Se ha verificado la seguridad de conversación. El TP que invoca ha suministrado un identificador de usuario (que se ha devuelto como el parámetro *user_id* en este verbo) y o bien ha suministrado una contraseña válida o bien ha indicado que la seguridad de conversación ya se ha verificado.

AP_NO No se ha verificado la seguridad de conversación. El TP invocado no solicita ningún identificador de usuario ni contraseña.

user_id Identificador de usuario asociado con el TP. El identificador es una cadena de caracteres EBCDIC de tipo AE de 10 bytes, rellenada por la derecha con espacios EBCDIC si el identificador tiene menos de 10 bytes. La contraseña no se devuelve en este verbo, sino en el verbo RECEIVE_ALLOCATE.

AIX, LINUX

prot_luw_id
LUWID (identificador de unidad lógica de trabajo) protegido para la transacción en que participa el TP.

El LUWID protegido se utiliza para correlacionar conversaciones protegidas (aquellas con el parámetro *sync_level* definido en `AP_SYNCPT`). Consta de los siguientes parámetros:

prot_luw_id.fq_length
Longitud (1–17 bytes) del nombre de LU completamente calificado asociado con la unidad lógica de trabajo (el parámetro *prot_luw_id.fq_luw_name* especifica el nombre de LU en sí).

prot_luw_id.fq_luw_name
Nombre de LU completamente calificado asociado con la unidad lógica de trabajo. Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC. Consta de un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.

prot_luw_id.instance
Número de instancia asociado con la unidad lógica de trabajo (un número binario de 6 bytes).

prot_luw_id.sequence
Número de secuencia del segmento actual de la unidad lógica de trabajo (un número binario de 2 bytes).

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
`AP_PARAMETER_CHECK`

secondary_rc
Los valores posibles son:

`AP_BAD_TP_ID`
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AIX, LINUX

`AP_INVALID_FORMAT`
El parámetro *format* está definido en un valor que no es válido.

`AP_SYNC_NOT_ALLOWED`
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono `APPC`. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los valores posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

El estado de conversación no cambia con este verbo.

SET_TP_PROPERTIES

AIX, LINUX

El verbo SET_TP_PROPERTIES permite a la aplicación establecer las propiedades del TP local, que se utilizan cuando se asignan nuevas conversaciones para el TP. Este verbo da acceso a las propiedades siguientes:

- El identificador de usuario que se utilizará cuando se asigne una nueva conversación especificando que la seguridad ya se ha verificado. En general, un TP utiliza la función de seguridad ya verificada cuando ha sido invocado por otro TP con un identificador de usuario y una contraseña válidos, y ahora está invocando un tercer TP como parte de la misma transacción; en este caso, APPC envía el identificador de usuario del TP original sin que sea necesaria una contraseña. En cambio, si el TP no ha sido invocado por otro TP, APPC utiliza el nombre de usuario de AIX o Linux con el que se ejecuta la aplicación como el identificador de usuario para la seguridad de conversación.

  Sin embargo, si el TP ha obtenido y verificado el identificador de usuario y la contraseña por otros medios (por ejemplo, si pide al usuario que escriba un identificador de usuario y una contraseña explícitamente antes de asignar la conversación), debe proporcionar el identificador de usuario a APPC utilizando el verbo SET_TP_PROPERTIES antes de invocar otro TP utilizando la función de seguridad ya verificada.
- Identificadores para la unidad lógica de trabajo en que participa el TP. Una unidad lógica de trabajo es una transacción entre TP APPC para llevar a cabo una tarea determinada; puede comprender dos TP que se comunican o una secuencia de conversaciones entre varios TP. Existen dos identificadores de

unidad lógica de trabajo (LUWID) asociados con el TP: el LUWID no protegido, que se utiliza para las conversaciones con un parámetro *sync_level* definido en AP_NONE o AP_CONFIRM_SYNC_LEVEL, y el LUWID protegido, que se utiliza para las conversaciones con un parámetro *sync_level* definido en AP_SYNCPT.

Estructura del VCB: SET_TP_PROPERTIES

La definición de la estructura del VCB para el verbo SET_TP_PROPERTIES es la siguiente:

```
typedef struct set_tp_properties
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reservado    */
  unsigned char    format;                    /* Reservado            */
  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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_SET_TP_PROPERTIES

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

set_prot_id
Especifica si APPC debe modificar el identificador de unidad lógica de trabajo protegido. Los valores posibles son:

AP_YES Modificar el LUWID protegido para este TP.

AP_NO Dejar el LUWID protegido sin cambiar.

new_prot_id
Especifica si APPC debe generar un nuevo identificador de unidad lógica de trabajo protegido o utilizar el especificado en este verbo. Este parámetro está reservado si *set_prod_id* se define en AP_NO. Los valores posibles son:

AP_YES Generar un nuevo LUWID protegido.

AP_NO Establecer el LUWID protegido del TP en el suministrado en este verbo.

prot_id Si *set_prot_id* tiene el valor AP_YES y *new_prot_id* tiene el valor AP_NO, esta

estructura especifica el nuevo LUWID protegido para el TP; de lo contrario la estructura está reservada. La estructura contiene los parámetros siguientes:

prot_id.fq_length
Longitud (1–17 bytes) del nombre de LU completamente calificado asociado con la unidad lógica de trabajo (el parámetro siguiente especifica el nombre de LU en sí).

prot_id.fq_luw_name
Nombre de LU completamente calificado asociado con la unidad lógica de trabajo. Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC. Consta de un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.

prot_id.instance
Número de instancia asociado con la unidad lógica de trabajo (un número binario de 6 bytes).

prot_id.sequence
Número de secuencia del segmento actual de la unidad lógica de trabajo (un número binario de 2 bytes).

set_unprot_id
Especifica si APPC debe modificar el identificador de unidad lógica de trabajo no protegido. Los valores posibles son:

AP_YES Modificar el LUWID no protegido para este TP.

AP_NO Dejar el LUWID no protegido sin cambiar.

new_unprot_id
Especifica si APPC debe generar un nuevo identificador de unidad lógica de trabajo no protegido o utilizar el especificado en este verbo. Este parámetro está reservado si *set_unprot_id* se define en AP_NO. Los valores posibles son:

AP_YES Generar un nuevo LUWID no protegido.

AP_NO Establecer el LUWID no protegido del TP en el suministrado en este verbo.

unprot_id
Si *set_unprot_id* tiene el valor AP_YES y *new_unprot_id* tiene el valor AP_NO, esta estructura especifica el nuevo LUWID no protegido para el TP; de lo contrario la estructura está reservada. La estructura contiene los parámetros siguientes:

unprot_id.fq_length
Longitud (1–17 bytes) del nombre de LU completamente calificado asociado con la unidad lógica de trabajo (el parámetro siguiente especifica el nombre de LU en sí).

unprot_id.fq_luw_name
Nombre de LU completamente calificado asociado con la unidad lógica de trabajo. Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC. Consta de un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.

unprot_id.instance
Número de instancia asociado con la unidad lógica de trabajo (un número binario de 6 bytes).

unprot_id.sequence
Número de secuencia del segmento actual de la unidad lógica de trabajo (un número binario de 2 bytes).

set_user_id
Especifica si APPC debe modificar el identificador de usuario. Los valores posibles son:

AP_YES Modificar el identificador de usuario para este TP.

AP_NO Dejar el identificador de usuario sin cambiar.

set_password
Especifica si APPC debe modificar la contraseña asociada con el parámetro *new_password*. Los valores posibles son:

AP_YES APPC debe modificar la contraseña.

AP_NO APPC no debe modificar la contraseña.

user_id Si *set_user_id* tiene el valor AP_YES, este parámetro especifica el nuevo identificador de usuario; de lo contrario está reservado.

new_password
Si *set_password* tiene el valor AP_YES, este parámetro especifica la nueva contraseña; de lo contrario está reservado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

prot_id Si *set_prot_id* y *new_prot_id* tienen el valor AP_YES, esta estructura especifica el nuevo LUWID protegido para el TP, tal como lo genera APPC. La estructura contiene los parámetros siguientes:

prot_id.fq_length
Longitud (1–17 bytes) del nombre de LU completamente calificado asociado con la unidad lógica de trabajo (el parámetro *prot_id.fq_luw_name* especifica el nombre de LU en sí).

prot_id.fq_luw_name
Nombre de LU completamente calificado asociado con la unidad lógica de trabajo. Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC. Consta de un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.

prot_id.instance
Número de instancia asociado con la unidad lógica de trabajo (un número binario de 6 bytes).

prot_id.sequence
Número de secuencia del segmento actual de la unidad lógica de trabajo (un número binario de 2 bytes).

unprot_id
Si *set_unprot_id* y *new_unprot_id* tienen el valor AP_YES, esta estructura especifica el nuevo LUWID no protegido para el TP, tal como lo genera APPC. La estructura contiene los parámetros siguientes:

unprot_id.fq_length
Longitud (1–17 bytes) del nombre de LU completamente calificado asociado con la unidad lógica de trabajo (el parámetro *unprot_id.fq_luw_name* especifica el nombre de LU en sí).

unprot_id.fq_luw_name
Nombre de LU completamente calificado asociado con la unidad lógica de trabajo. Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC. Consta de un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.

unprot_id.instance
Número de instancia asociado con la unidad lógica de trabajo (un número binario de 6 bytes).

unprot_id.sequence
Número de secuencia del segmento actual de la unidad lógica de trabajo (un número binario de 2 bytes).

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_INVALID_FORMAT
El parámetro reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los valores posibles son:

primary_rc

```
AP_COMM_SUBSYSTEM_ABENDED
AP_INVALID_VERB
AP_TP_BUSY
AP_UNEXPECTED_SYSTEM_ERROR
```

APPC no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación puede encontrarse en cualquier estado.

Cambio de estado

El estado de conversación no cambia con este verbo.

Uso y restricciones

Para los TP que utilizan funciones de punto de sincronización, cuando la aplicación local cambia el LUWID protegido, el gestor de puntos de sincronización se encarga de enviar la cabecera PS apropiada a la aplicación asociada para informarle del nuevo LUWID protegido. De la misma forma, cuando el gestor de puntos de sincronización recibe una cabecera PS que contiene un nuevo LUWID protegido, debe emitir el verbo SET_TP_PROPERTIES para informar a la LU local del nuevo LUWID.

Capítulo 4. Verbos de conversación APPC

Este capítulo contiene una descripción de cada uno de los verbos de conversación APPC. Para cada verbo se proporciona la siguiente información:

- Definición del verbo.
- Estructura que define el bloque de control de verbos (VCB) utilizado por el verbo. La estructura está definida en el archivo de cabecera APPC **/usr/include/sna/appc_c.h** (AIX), **/opt/ibm/sna/include/appc_c.h** (Linux) o **sdk/winappc.h** (Windows). Los parámetros que empiezan por *reserv* están reservados.
- Parámetros (campos del VCB) suministrados a APPC y devueltos por APPC. Para cada parámetro, se proporciona la siguiente información:
  - Descripción
  - Valores posibles
  - Información adicional
- Estado o estados de conversación en que el verbo puede emitirse.
- Estado o estados a que puede cambiar la conversación al volver el verbo. Las condiciones que no provocan un cambio de estado no se especifican. Por ejemplo, las comprobaciones de parámetros y las comprobaciones de estado no provocan un cambio de estado.
- Información adicional sobre el uso del verbo.

La mayoría de los parámetros suministrados a APPC y devueltos por APPC son valores hexadecimales. Para simplificar la codificación, estos valores están representados por constantes simbólicas significativas definidas en el archivo de cabecera **values_c.h**, incluido en el archivo de cabecera APPC **appc_c.h**. Por ejemplo, el parámetro *opcode* del verbo MC_SEND_DATA es el valor hexadecimal representado por la constante simbólica `AP_M_SEND_DATA`.

Es importante que utilice la constante simbólica y no el valor hexadecimal cuando establezca valores para los parámetros suministrados o cuando compruebe los valores de los parámetros devueltos. Esto se debe a que los diferentes sistemas Linux guardan estos valores en la memoria de modo diferente, por lo que es posible que el valor mostrado no tenga el formato que su sistema reconoce.

WINDOWS

Para Windows, las constantes para los valores de los parámetros suministrados y devueltos están definidas en el archivo de cabecera APPC de Windows, **winappc.h**.

La notación “[MC_]verbo” se refiere tanto a la forma correlacionada como a la básica de un verbo APPC. Por ejemplo, [MC_]SEND_DATA se refiere a los verbos MC_SEND_DATA y SEND_DATA.

Nota: Los VCB APPC contienen muchos parámetros marcados como “reservados”; el software Communications Server para Linux utiliza internamente algunos

de ellos y otros no se utilizan en esta versión, aunque pueden utilizarse en versiones futuras. Su aplicación no debe intentar acceder a ninguno de estos parámetros reservados, sino que debe establecer todo el contenido del VCB en cero para garantizar que todos estos parámetros tengan el valor cero, antes de que establezca otros parámetros utilizados por el verbo. Esto garantiza que Communications Server para Linux no interpretará de forma incorrecta ninguno de estos parámetros de uso interno y que su aplicación continuará funcionando con versiones posteriores de Communications Server para Linux en que estos parámetros pueden utilizarse para proporcionar nuevas funciones.

Para establecer el contenido del VCB en cero, utilice `memset`:

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

Los verbos de conversación se describen en el orden siguiente:

GET_TYPE

[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

El verbo GET_TYPE devuelve el tipo de una conversación determinada (básica o correlacionada) y si la conversación funciona en dúplex o semidúplex.

Con esta información, el TP puede determinar los verbos correctos que se deben emitir en esta conversación.

Estructura del VCB: GET_TYPE

AIX, LINUX

La definición de la estructura del VCB para el verbo GET_TYPE es la siguiente:

```
typedef struct get_type
{
  AP_UINT16         opcode;
  unsigned char     opext;                  /* Reservado    */
  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;
```

Estructura del VCB: GET_TYPE (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo GET_TYPE es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_GET_TYPE

format Si crea una nueva aplicación APPC o vuelve a compilar una aplicación APPC existente con el archivo de cabecera APPC actual, deberá establecer este parámetro en 1. (Las aplicaciones existentes creadas con versiones anteriores del archivo de cabecera, en que este parámetro estaba reservado, seguirán funcionando sin ningún cambio y no será necesario volver a crearlas).

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de la conversación por la que pregunta este TP.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

conv_type
Tipo de conversación de la conversación identificada por *conv_id*.

Los valores posibles son:
AP_BASIC_CONVERSATION
AP_MAPPED_CONVERSATION

duplex_type
Tipo de dúplex de la conversación identificada por *conv_id*.

Los valores posibles son:
AP_HALF_DUPLEX
AP_FULL_DUPLEX

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AIX, LINUX

AP_INVALID_FORMAT
El parámetro *format* está definido en un valor que no es válido.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

El estado de conversación no cambia con este verbo.

MC_ALLOCATE y ALLOCATE

El verbo MC_ALLOCATE o ALLOCATE es emitido por el TP que invoca. Este verbo asigna una sesión entre la LU local y la LU asociada y (junto con el verbo RECEIVE_ALLOCATE) establece una conversación entre el TP que invoca y el TP invocado.

El verbo MC_ALLOCATE establece una conversación correlacionada. El verbo ALLOCATE puede establecer una conversación básica o correlacionada. El uso del

verbo ALLOCATE para establecer una conversación correlacionada permite al TP utilizar verbos de conversación básica para comunicarse con un TP asociado de conversación correlacionada.

Una vez ejecutado correctamente este verbo, APPC genera un identificador de conversación (*conv_id*). Este identificador es un parámetro necesario para todos los demás verbos de conversación APPC.

La petición [MC_]ALLOCATE normalmente no se enviará de inmediato a la LU asociada, sino que se colocará en cola en la LU local hasta que pueda enviarse un almacenamiento intermedio completo. Esto significa que normalmente no se informa de los errores de asignación de una conversación en el verbo [MC_]ALLOCATE sino en un verbo posterior.

Estructura del VCB: MC_ALLOCATE

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_ALLOCATE es la siguiente:

```
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;
```

Estructura del VCB: ALLOCATE

La definición de la estructura del VCB para el verbo ALLOCATE es la siguiente:

```
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;
```

Estructura del VCB: MC_ALLOCATE (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_ALLOCATE es la siguiente:

```
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;
```

Estructura del VCB: ALLOCATE (Windows)

La definición de la estructura del VCB para el verbo ALLOCATE es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_ALLOCATE
Para el verbo MC_ALLOCATE.

AP_B_ALLOCATE
Para el verbo ALLOCATE.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_ALLOCATE.

AP_BASIC_CONVERSATION
Para el verbo ALLOCATE.

Si el verbo se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con el valor AP_NON_BLOCKING.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED para un TP que invoca o por el verbo RECEIVE_ALLOCATE para un TP invocado.

conv_type
Tipo de conversación para asignar. Este parámetro sólo es utilizado por el verbo ALLOCATE.

Los valores posibles son:
AP_BASIC_CONVERSATION

AP_MAPPED_CONVERSATION

Si el verbo ALLOCATE establece una conversación correlacionada, el TP local debe emitir verbos de conversación básica y proporcionar su propia capa de correlación para convertir los registros de datos a registros lógicos y los registros lógicos a registros de datos. El TP asociado puede emitir verbos de conversación básica y proporcionar la capa de correlación, o puede utilizar verbos de conversación correlacionada (si la implementación de APPC que utiliza el TP asociado soporta los verbos de conversación correlacionada). Para obtener más información, consulte la publicación de IBM *Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2.*

sync_level

Nivel de sincronización de la conversación.

Este parámetro determina si los TP pueden solicitar la confirmación de recepción de datos y confirmar la recepción de datos. Los valores posibles son:

AP_NONE

El proceso de confirmación no se utilizará en esta conversación.

AP_CONFIRM_SYNC_LEVEL

Los TP pueden utilizar el proceso de confirmación en esta conversación. Este valor sólo se puede utilizar en las conversaciones semidúplex; el proceso de confirmación no se puede realizar en las conversaciones dúplex.

AIX, LINUX

AP_SYNCPT

Los TP pueden utilizar las funciones de punto de sincronización de LU 6.2 en esta conversación. Establezca este valor únicamente si tiene un SPM (Gestor de puntos de sincronización) y C-PRM (Gestor de recursos protegidos por conversación) además del producto Communications Server para Linux estándar. Para ver más información, consulte “Soporte de punto de sincronización” en la página 23.

rtn_ctl Especifica cuándo la LU local que participa en una petición de sesión del TP local debe devolver el control al TP local. Para ver más información sobre las sesiones, consulte “Sesiones LU-LU” en la página 61. Sea cual sea el valor de este parámetro, la LU devuelve inmediatamente el control al TP si encuentra determinados errores, tales como un límite de sesiones cero (lo que significa que nunca se asignará una sesión).

Los valores posibles son:

AP_IMMEDIATE

- Si el parámetro *auto_act* del verbo DEFINE_MODE o el mandato **define_mode** tiene el valor `0` (cero), Communications Server para Linux no intenta activar ninguna sesión. Si una sesión ganadora de la contienda está disponible de inmediato (activa y no utilizada por otra conversación), la LU le asigna esta conversación e inmediatamente devuelve el control al TP. Si una

sesión ganadora de la contienda no está inmediatamente disponible, se devuelve el control al TP de inmediato con el parámetro *primary_rc* con el valor AP_UNSUCCESSFUL.

- Si el parámetro *auto_act* del verbo DEFINE_MODE o el mandato **define_mode** tiene cualquier otro valor, Communications Server para Linux intentará activar una o varias sesiones.

Para ver más información, consulte la descripción del verbo DEFINE_MODE en la publicación *Communications Server for Linux NOF Programmer's Guide* o el mandato **define_mode** en la publicación *Communications Server for Linux Administration Command Reference*.

AP_WHEN_SESSION_ALLOCATED
Si una sesión está disponible de inmediato (activa y no utilizada por otra conversación), la LU le asigna esta conversación. Si no hay ninguna sesión disponible de inmediato pero puede activarse una, la LU la activa y le asigna la conversación; si no puede activar una sesión, espera a que se libere una.

AP_WHEN_SESSION_FREE
Si una sesión está disponible de inmediato (activa y no utilizada por otra conversación), la LU le asigna esta conversación. Si no hay ninguna sesión disponible de inmediato pero puede activarse una, la LU la activa y le asigna la conversación. Si no hay ninguna sesión activa libre y no puede activarse otra sesión, se devuelve el control al TP con el código de retorno primario AP_ALLOCATION_ERROR y el código de retorno secundario AP_ALLOCATION_FAILURE_RETRY. Es parecido a AP_WHEN_SESSION_ALLOCATED salvo que la LU no esperará a que se libere una sesión.

AP_WHEN_CONWINNER_ALLOC
Igual que AP_WHEN_SESSION_ALLOCATED, excepto que la LU siempre asigna la conversación a una sesión ganadora de la contienda; no utilizará una sesión perdedora de la contienda.

AP_WHEN_CONLOSER_ALLOC
Igual que AP_WHEN_SESSION_ALLOCATED, excepto que la LU siempre asigna la conversación a una sesión perdedora de la contienda; no utilizará una sesión ganadora de la contienda.

AP_WHEN_CONV_GROUP_ALLOC
Utilice este valor si quiere que la nueva conversación utilice la misma sesión que la de una conversación anterior; establezca el parámetro *conv_group_id* en el identificador de grupo de conversación de la conversación anterior, que se ha devuelto en el verbo [MC_]ALLOCATE o en RECEIVE_ALLOCATE.

Si la sesión identificada por el parámetro *conv_group_id* está disponible inmediatamente (activa y no utilizada por otra conversación), la LU le asigna esta conversación e inmediatamente devuelve el control al TP. Si otra conversación utiliza la sesión, la LU espera a que se libere. Si la sesión ya no está activa, se devuelve el control al TP con el código de retorno primario AP_ALLOCATION_ERROR y el código de retorno secundario AP_ALLOCATION_FAILURE_NO_RETRY.

duplex_type
Tipo de dúplex de la nueva conversación. Si desea obtener información

detallada sobre las diferencias entre las conversaciones dúplex y semidúplex, consulte el apartado “Conversaciones semidúplex y dúplex” en la página 4.

Los valores posibles son:

AP_HALF_DUPLEX
Conversación semidúplex.

AP_FULL_DUPLEX
Conversación dúplex.

conv_group_id
Identificador de grupo de conversación de la sesión solicitada para la conversación. Este parámetro sólo se utiliza si *rtn_ctl* tiene el valor AP_WHEN_CONV_GROUP_ALLOC; defínalo en ceros binarios para cualquier otro valor de *rtn_ctl*.

plu_alias
Alias por el que se conoce la LU asociada en el TP local.

Este parámetro es una cadena de caracteres ASCII de 8 bytes, rellenada por la derecha con blancos ASCII (0x20) si el alias tiene menos de ocho caracteres. Puede constar de cualquiera de los siguientes caracteres:

- Letras en mayúsculas
- Números del 0 al 9
- Blancos
- Los caracteres especiales $, #, % y @

El primer carácter de esta cadena no puede ser un blanco.

Para identificar la LU por su nombre de LU en lugar de por su alias de LU, establezca este parámetro en ocho ceros binarios y especifique el nombre de LU en el parámetro *fqplu_name*.

mode_name
Nombre de un conjunto de características de red definidas durante la configuración.

Este parámetro es una cadena de caracteres EBCDIC de 8 bytes. Puede constar de caracteres del juego de caracteres EBCDIC de tipo A. Estos caracteres son los siguientes:

- Letras en mayúsculas
- Números del 0 al 9
- Los caracteres especiales $, # y @

El primer carácter de la cadena debe ser una letra en mayúsculas o puede ser # para una de las modalidades definidas por SNA como, por ejemplo, #INTER. Si desea obtener información sobre las modalidades definidas por SNA, consulte la publicación *Communications Server para Linux - Guía de administración*. Si el nombre de la modalidad tiene menos de ocho caracteres, rellénelo por la derecha con blancos EBCDIC (0x40).

Un nombre de modalidad también puede estar formado por todo blancos EBCDIC (0x40).

En una conversación correlacionada, el nombre no puede ser SNASVCMG (un nombre de modalidad reservado utilizado internamente por APPC). No se recomienda el uso de este nombre en una conversación básica.

Si el nombre de modalidad especificado no se corresponde ni con una modalidad definida por SNA ni con una modalidad definida en la configuración de Communications Server para Linux, Communications Server para Linux crea una nueva modalidad basada en el valor por omisión especificado en la configuración (o en la modalidad definida por SNA con un nombre de modalidad en blanco, si no hay ninguna modalidad definida por omisión).

tp_name

Nombre del TP invocado.

El valor de *tp_name* especificado por el verbo [MC_]ALLOCATE en el TP que invoca debe coincidir con el valor de *tp_name* especificado por el verbo RECEIVE_ALLOCATE en el TP invocado.

Este parámetro es una cadena de caracteres EBCDIC de 64 bytes sensible a las mayúsculas y minúsculas. El parámetro *tp_name* normalmente consta de caracteres del juego de caracteres EBCDIC de tipo AE (salvo cuando se denomina un TP de servicio). Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

Si el nombre de TP tiene menos de 64 bytes, utilice blancos EBCDIC (`0x40`) para rellenarlo por la derecha.

El convenio SNA para denominar un TP de servicio es una excepción a lo mencionado anteriormente; el nombre consta de hasta cuatro caracteres, el primero de los cuales es un byte hexadecimal entre `0x00` y `0x3F`. Los demás caracteres pertenecen al juego de caracteres EBCDIC de tipo AE.

security

Especifica la información que necesita la LU asociada para validar el acceso al TP invocado.

En función de la seguridad de conversación establecida para el TP invocado durante la configuración, utilice uno de los valores siguientes:

AP_NONE

El TP invocado no utiliza la seguridad de conversación. (Si utiliza este valor, el TP invocado debe estar configurado para no utilizar la seguridad de conversación.)

AP_PGM El TP invocado utiliza la seguridad de conversación y por consiguiente requiere un identificador de usuario y una contraseña. Suministre esta información mediante los parámetros *user_id* y *pwd*.

AP_PGM_STRONG

El TP invocado utiliza la seguridad de conversación y por consiguiente requiere un identificador de usuario y una contraseña. Además, al establecer `AP_PGM_STRONG` se estipula que Communications Server para Linux cifra la contraseña cuando se envía a través de la red. Suministre el identificador de usuario y la contraseña mediante los parámetros *user_id* y *pwd*.

AP_SAME

Utilice este valor cuando su TP ha sido invocado por otro TP, utilizando un identificador de usuario y una contraseña válidos, y ahora invoca un tercer TP que también requiere la seguridad de conversación. (La situación en que un TP invoca un segundo TP que después invoca un tercer TP se ilustra en “Conversaciones

múltiples" en la página 3.) Este valor informa al tercer TP (el TP invocado) de que la seguridad de conversación ya ha sido verificada para el primer TP que invoca.

Si utiliza este valor, el *tp_id* proporcionado en este verbo [MC_]ALLOCATE debe ser el mismo que se ha devuelto en el verbo RECEIVE_ALLOCATE al invocar este TP.

AIX, LINUX

Este valor también se puede utilizar si el TP no ha sido invocado por otro TP pero ha obtenido y verificado la información de seguridad pertinente por otros medios (por ejemplo, del nombre de usuario de AIX o Linux y la contraseña suministrados durante el inicio de sesión). En este caso, APPC utiliza el nombre de usuario de AIX o Linux con el que se ejecuta la aplicación, truncado en 10 caracteres si es necesario, como el identificador de usuario para la seguridad de conversación; asegúrese de que este nombre consta de caracteres de cadena de tipo AE válidos (consulte la descripción del parámetro *user_id* para ver más información) y que es un nombre de usuario válido para el TP que se invoca.

Si el TP ha obtenido la seguridad de conversación por otros medios (por ejemplo, solicitando al usuario que escriba un identificador de usuario y una contraseña válidos antes de asignar la conversación), debe utilizar el verbo SET_TP_PROPERTIES para especificar este identificador de usuario antes de emitir [MC_]ALLOCATE.

pwd Contraseña asociada con *user_id*.

Este parámetro sólo es necesario si el parámetro *security* tiene el valor `AP_PGM` o `AP_PGM_STRONG`; de lo contrario está reservado.

Los parámetros *pwd* y *user_id* deben coincidir con un par de identificador de usuario y contraseña configurado en la máquina en que está ubicado el TP invocado.

Este parámetro es una cadena de caracteres EBCDIC de 10 bytes sensible a las mayúsculas y minúsculas. El parámetro *pwd* puede constar de caracteres del juego de caracteres EBCDIC de tipo AE. Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

Si la contraseña tiene menos de 10 bytes, utilice blancos EBCDIC (`0x40`) para rellenarla por la derecha.

user_id Identificador de usuario necesario para acceder al TP asociado.

Este parámetro sólo es necesario si el parámetro *security* tiene el valor `AP_PGM` o `AP_PGM_STRONG`; de lo contrario está reservado.

Los parámetros *pwd* y *user_id* deben coincidir con un par de identificador de usuario y contraseña configurado en la máquina en que está ubicado el TP invocado.

Este parámetro es una cadena de caracteres EBCDIC de 10 bytes sensible a las mayúsculas y minúsculas. El parámetro *user_id* puede constar de caracteres del juego de caracteres EBCDIC de tipo AE. Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

Si el identificador de usuario tiene menos de 10 bytes, utilice blancos EBCDIC (`0x40`) para rellenarlo por la derecha.

pip_dlen

Longitud de PIP (parámetros de inicialización de programa) para pasar al TP asociado. El rango de este valor es `0`–`32.767`.

No todas las implementaciones de APPC soportan los datos PIP. Defina *pip_dlen* en `0` (cero) si el TP asociado utiliza una implementación de APPC que no soporta datos PIP o si el TP asociado es una aplicación CPI-C.

pip_dptr

Dirección del almacenamiento intermedio que contiene datos PIP.

Utilice únicamente este parámetro cuando *pip_dlen* sea superior a `0` (cero).

Los datos PIP pueden ser parámetros de inicialización o información de configuración del entorno que requiere un TP asociado o un sistema operativo remoto. Los datos PIP deben tener el formato de corriente de datos general (GDS). Si desea obtener más información, consulte la publicación de IBM *Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*.

WINDOWS

El almacenamiento intermedio de datos PIP puede residir en un área de datos estática o en un área asignada globalmente. El almacenamiento intermedio de datos debe caber completamente en esta área.

fqplu_name

Nombre de LU completamente calificado de la LU asociada. Este parámetro sólo se utiliza si *plu_alias* está definido en ceros.

Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC, que contiene uno de los elementos siguientes:

- Un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.
- Un nombre de LU de 1–8 caracteres de cadena de tipo A (sin el identificador de red ni el punto EBCDIC).

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

conv_id
Identificador de conversación. Este valor identifica la conversación establecida entre los dos TP.

conv_group_id
Identificador de grupo de conversación de la sesión utilizada por la conversación.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_TYPE
(Valor devuelto únicamente para el verbo ALLOCATE de conversación básica.) El valor especificado para *conv_type* no es válido.

AP_BAD_DUPLEX_TYPE
El valor especificado para *duplex_type* no es válido.

AP_BAD_PARTNER_LU_ALIAS
Se ha producido una de las situaciones siguientes:

- El parámetro *plu_alias* no coincide con ningún alias de LU asociada definido.
- El valor especificado para *fqplu_name* no es válido.

AP_BAD_RETURN_CONTROL
El valor especificado para *rtn_ctl* no es válido.

AP_BAD_SECURITY
El valor especificado para la seguridad no es válido.

AP_BAD_SYNC_LEVEL
El valor especificado para *sync_level* no es válido.

AP_BAD_TP_ID
El valor especificado para *tp_id* no es válido.

WINDOWS

AP_INVALID_DATA_SEGMENT
Los datos PIP son más largos que el segmento de datos asignado o la dirección del almacenamiento intermedio de datos PIP es incorrecta.

AP_CONFIRM_INVALID_FOR_FDX
Se ha establecido AP_CONFIRM_SYNC_LEVEL como valor del parámetro *sync_level* en una conversación dúplex. Este valor sólo se puede utilizar en una conversación semidúplex.

AP_NO_USE_OF_SNASVCMG
(Valor devuelto únicamente para el verbo MC_ALLOCATE.) SNASVCMG no es un valor válido para *mode_name*.

AP_PIP_LEN_INCORRECT
El valor de *pip_dlen* es superior a 32.767.

AP_UNKNOWN_PARTNER_MODE
El valor especificado para *mode_name* no es válido.

AIX, LINUX

AP_INVALID_FORMAT
El parámetro *format* está definido en un valor que no es válido.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Sesión no disponible: En función del valor especificado para *rtn_ctl*, APPC puede devolver el parámetro siguiente:

primary_rc

AP_UNSUCCESSFUL
El parámetro suministrado *rtn_ctl* especifica el retorno inmediato del control (AP_IMMEDIATE) al TP, y la LU local no tiene una sesión ganadora de la contienda disponible.

Error de asignación: Si Communications Server para Linux no puede asignar la conversación, APPC devuelve los parámetros siguientes:

primary_rc
AP_ALLOCATION_ERROR

secondary_rc
Los valores posibles son:

AP_ALLOCATION_FAILURE_NO_RETRY
No puede asignarse la conversación debido a una condición permanente, como un error de configuración o un error de protocolo de sesión. Para determinar el error, el administrador del sistema debe examinar el archivo de anotaciones de error. No vuelva a intentar la asignación hasta que se haya corregido el error.

Este valor también se devuelve si la sesión que corresponde al identificador de grupo de conversación solicitado ya no está activa.

AP_ALLOCATION_FAILURE_RETRY
No se puede asignar la conversación debido a una condición temporal, como una anomalía de enlace. El motivo de la anomalía está anotado en el archivo de anotaciones de error del sistema. Vuelva a intentar la asignación, preferiblemente después de un tiempo de espera para dejar que la condición desaparezca.

AP_FDX_NOT_SUPPORTED_BY_LU
Se ha establecido AP_FULL_DUPLEX como valor del parámetro *duplex_type*, pero la LU que este TP utiliza no admite el funcionamiento dúplex.

Para ver información sobre estos códigos de retorno secundarios, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

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
Datos de detección SNA que dan más información sobre el motivo de la anomalía en la asignación.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo el estado de conversación es Restablecer. Puede emitirse durante una conversación existente que se encuentre en cualquier estado, puesto que siempre supone el inicio de una nueva conversación que se encuentra en estado Restablecer.

Cambio de estado

Si el verbo se ejecuta de forma correcta (el valor de *primary_rc* es AP_OK), el estado de la nueva conversación será Enviar (en el caso de una conversación semidúplex) o a Enviar-Recibir (en el caso de una conversación dúplex). Si el verbo falla, el estado no se modifica.

Conversión EBCDIC-ASCII y ASCII-EBCDIC

Algunos parámetros del verbo [MC_]ALLOCATE son cadenas EBCDIC o ASCII. Un TP puede utilizar el verbo CSV CONVERT para convertir una cadena de un juego de caracteres al otro. Si desea obtener más información, consulte la publicación *Communications Server for Linux CSV Programmer's Guide*.

Asignación inmediata

Para asegurarse de que la conversación con el elemento asociado se inicia de inmediato, el TP que invoca puede emitir el verbo [MC_]FLUSH o [MC_]CONFIRM inmediatamente después del verbo [MC_]ALLOCATE. ([MC_]CONFIRM sólo es válido para las conversaciones semidúplex). De lo contrario, la petición [MC_]ALLOCATE se acumula con otros datos en el almacenamiento intermedio de envío de la LU local hasta que se llena el almacenamiento intermedio.

Confirmación de la asignación (sólo conversaciones semidúplex)

Al emitir el verbo [MC_]CONFIRM después de [MC_]ALLOCATE, el TP que invoca puede determinar inmediatamente si la asignación se ha efectuado de forma correcta (si *sync_level* está definido en AP_CONFIRM_SYNC_LEVEL).

MC_CONFIRM y CONFIRM

El verbo MC_CONFIRM o CONFIRM envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al TP asociado.

Nota: Este verbo sólo se puede utilizar en las conversaciones semidúplex; no es válido para las conversaciones dúplex.

Como respuesta al verbo [MC_]CONFIRM, el TP asociado normalmente emite el verbo [MC_]CONFIRMED para confirmar que ha recibido los datos sin ningún error. (Si el TP asociado encuentra un error, emite el verbo [MC_]SEND_ERROR o desasigna anormalmente la conversación.)

El TP puede emitir el verbo [MC_]CONFIRM únicamente si el nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

Estructura del VCB: MC_CONFIRM

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_CONFIRM es la siguiente:

```
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;
```

Estructura del VCB: CONFIRM

La definición de la estructura del VCB para el verbo CONFIRM es la siguiente:

```
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;
```

Estructura del VCB: MC_CONFIRM (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_CONFIRM es la siguiente:

```
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;
```

Estructura del VCB: CONFIRM (Windows)

La definición de la estructura del VCB para el verbo CONFIRM es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_CONFIRM
Para el verbo MC_CONFIRM.

AP_B_CONFIRM
Para el verbo CONFIRM.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_CONFIRM.

AP_BASIC_CONVERSATION
Para el verbo CONFIRM.

Si el verbo se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con el valor AP_NON_BLOCKING.

format Si crea una nueva aplicación APPC o vuelve a compilar una aplicación APPC existente con el archivo de cabecera APPC actual, deberá establecer este parámetro en 1. (Las aplicaciones existentes creadas con versiones anteriores del archivo de cabecera, en que este parámetro estaba reservado, seguirán funcionando sin ningún cambio y no será necesario volver a crearlas).

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

rts_rcvd
Indicador de petición de envío recibida. Este parámetro sólo es válido para las conversaciones semidúplex; no se utiliza en las conversaciones dúplex.

Los valores posibles son:

AP_YES El TP asociado ha emitido el verbo [MC_]REQUEST_TO_SEND, que solicita que el TP local cambie la conversación al estado Recibir. Para cambiar al estado Recibir, el TP local puede utilizar el verbo [MC_]PREPARE_TO_RECEIVE, [MC_]RECEIVE_AND_WAIT o [MC_]RECEIVE_AND_POST.

AP_NO El TP asociado no ha emitido el verbo [MC_]REQUEST_TO_SEND.

expd_rcvd
Indicador de datos acelerados.

Los valores posibles son:

AP_YES El TP asociado ha enviado datos acelerados que el TP local aún no ha recibido. Para recibir estos datos, el TP local puede utilizar el verbo [MC_]RECEIVE_EXPEDITED_DATA.

Este indicador se puede establecer en diversos verbos APPC. Seguirá establecido en los verbos posteriores hasta que el TP local emita el verbo [MC_]RECEIVE_EXPEDITED_DATA para recibir los datos.

AP_NO No hay datos acelerados a la espera de ser recibidos.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_CONFIRM_INVALID_FOR_FDX
El TP local ha intentado utilizar el verbo [MC_]CONFIRM en una conversación dúplex. Este verbo sólo se puede utilizar en una conversación semidúplex.

AP_CONFIRM_ON_SYNC_LEVEL_NONE
El TP local ha intentado utilizar el verbo [MC_]CONFIRM en una

conversación con el nivel de sincronización AP_NONE. El nivel de sincronización, establecido por el verbo [MC_]ALLOCATE, debe ser AP_CONFIRM_SYNC_LEVEL.

AIX, LINUX

AP_INVALID_FORMAT
El parámetro *format* está definido en un valor que no es válido.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc
Los valores posibles son:

AP_CONFIRM_BAD_STATE
La conversación no se encuentra en estado Enviar ni Enviar-Pendiente.

AP_CONFIRM_NOT_LL_BDY
(Valor devuelto únicamente para el verbo CONFIRM de conversación básica.) La conversación para el TP local se encuentra en estado Enviar y el TP local no ha terminado de enviar un registro lógico.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

El verbo MC_CONFIRM devuelve el código de retorno primario siguiente:

primary_rc
AP_DEALLOC_ABEND

APPC no devuelve un código de retorno secundario con este código de retorno primario.

El verbo CONFIRM devuelve los códigos de retorno primarios siguientes:

primary_rc

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación debe encontrarse en estado Enviar o Enviar-Pendiente.

Cambio de estado

Los cambios de estado, resumidos en la tabla siguiente, dependen del valor del parámetro *primary_rc*.

primary_rc	**Estado nuevo**
AP_OK	Enviar
AP_PARAMETER_CHECK	Sin cambio
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	Recibir
AP_SVC_ERROR_PURGING	
AP_ALLOCATION_ERROR	Restablecer
AP_COMM_SUBSYSTEM_ABENDED	
AP_COMM_SUBSYSTEM_NOT_LOADED	
AP_CONV_FAILURE_RETRY	Restablecer
AP_CONV_FAILURE_NO_RETRY	
AP_DEALLOC_ABEND	Restablecer
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	

Sincronización con el TP asociado

El verbo [MC_]CONFIRM espera una respuesta del TP asociado. Una respuesta se genera mediante uno de los verbos siguientes en el TP asociado:

- [MC_]CONFIRMED
- [MC_]SEND_ERROR
- MC_DEALLOCATE con *dealloc_type* definido en AP_ABEND
- DEALLOCATE con *dealloc_type* definido en AP_ABEND_PROG, AP_ABEND_SVC o AP_ABEND_TIMER
- TP_ENDED

MC_CONFIRMED y CONFIRMED

El verbo MC_CONFIRMED o CONFIRMED responde a una petición de confirmación del TP asociado. Informa al TP asociado de que el TP local no ha detectado ningún error en los datos recibidos.

Nota: Este verbo sólo se puede utilizar en las conversaciones semidúplex; no es válido para las conversaciones dúplex.

Como el TP que emite la petición de confirmación espera una confirmación, el verbo [MC_]CONFIRMED sincroniza el proceso de los dos TP.

Orígenes de las peticiones de confirmación

Una petición de confirmación es emitida por uno de los siguientes verbos en el TP asociado:

- [MC_]CONFIRM
- [MC_]PREPARE_TO_RECEIVE si *ptr_type* está definido en `AP_SYNC_LEVEL` y el nivel de sincronización de la conversación (establecido por el verbo [MC_]ALLOCATE) es `AP_CONFIRM_SYNC_LEVEL`.
- [MC_]DEALLOCATE si *dealloc_type* está definido en `AP_SYNC_LEVEL` y el nivel de sincronización de la conversación (establecido por el verbo [MC_]ALLOCATE) es `AP_CONFIRM_SYNC_LEVEL`.
- [MC_]SEND_DATA si *type* está definido en `AP_SEND_DATA_CONFIRM` y el nivel de sincronización de la conversación (establecido por el verbo [MC_]ALLOCATE) es `AP_CONFIRM_SYNC_LEVEL`.

Recepción de peticiones de confirmación

El TP local recibe una petición de confirmación mediante el parámetro *what_rcvd* de uno de los verbos siguientes:

- [MC_]RECEIVE_IMMEDIATE
- [MC_]RECEIVE_AND_WAIT
- [MC_]RECEIVE_AND_POST

El TP local puede emitir el verbo MC_CONFIRMED o CONFIRMED únicamente si el campo *what_rcvd* contiene uno de los valores siguientes:

- `AP_CONFIRM_WHAT_RECEIVED`, `AP_DATA_CONFIRM` o `AP_DATA_COMPLETE_CONFIRM`.
- `AP_CONFIRM_SEND`, `AP_DATA_CONFIRM_SEND` o `AP_DATA_COMPLETE_CONFIRM_SEND`.
- `AP_CONFIRM_DEALLOCATE`, `AP_DATA_CONFIRM_DEALLOCATE` o `AP_DATA_COMPLETE_CONFIRM_DEALL`.

Estructura del VCB: MC_CONFIRMED

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_CONFIRMED es la siguiente:

```
typedef struct mc_confirmed
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado           */
  AP_UINT16         primary_rc;
```

```
  AP_UINT32      secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32        conv_id;
} MC_CONFIRMED;
```

Estructura del VCB: CONFIRMED

La definición de la estructura del VCB para el verbo CONFIRMED es la siguiente:

```
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;
```

Estructura del VCB: MC_CONFIRMED (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_CONFIRMED es la siguiente:

```
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;
```

Estructura del VCB: CONFIRMED (Windows)

La definición de la estructura del VCB para el verbo CONFIRMED es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_CONFIRMED
Para el verbo MC_CONFIRMED.

AP_B_CONFIRMED
Para el verbo CONFIRMED.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_CONFIRMED.

AP_BASIC_CONVERSATION
Para el verbo CONFIRMED.

Si el verbo se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con el valor AP_NON_BLOCKING.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AIX, LINUX

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

AP_CONFIRMED_INVALID_FOR_FDX
El TP local ha intentado utilizar el verbo [MC_]CONFIRMED en una conversación dúplex. Este verbo sólo se puede utilizar en una conversación semidúplex.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc

AP_CONFIRMED_BAD_STATE
La conversación no se encuentra en estado Confirmar, Confirmar-Enviar ni Confirmar-Desasignar.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación debe encontrarse en uno de los estados siguientes:

- Confirmar
- Confirmar-Enviar
- Confirmar-Desasignar

Cambio de estado

El estado nuevo viene determinado por el estado anterior (el estado de la conversación cuando el TP local ha emitido el verbo [MC_]CONFIRMED). El valor del parámetro *what_rcvd* del verbo de recepción precedente indica el estado anterior. Los cambios de estado posibles están resumidos en la tabla siguiente.

Estado anterior	Estado nuevo
Confirmar	Recibir
Confirmar-Enviar	Enviar
Confirmar-Desasignar	Restablecer

MC_DEALLOCATE y DEALLOCATE

El verbo MC_DEALLOCATE o DEALLOCATE desasigna una conversación entre dos TP.

Antes de desasignar la conversación, este verbo ejecuta el equivalente de uno de los verbos siguientes:

- El verbo [MC_]FLUSH, que envía el contenido del almacenamiento intermedio de envío de la LU local a la LU asociada (y al TP).
- El verbo [MC_]CONFIRM, que envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al TP asociado. ([MC_]CONFIRM sólo es válido para las conversaciones semidúplex).

Una vez que este verbo se ha ejecutado correctamente, el identificador de conversación ya no es válido.

Estructura del VCB: MC_DEALLOCATE

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_DEALLOCATE es la siguiente:

```
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;
```

Estructura del VCB: DEALLOCATE

La definición de la estructura del VCB para el verbo DEALLOCATE es la siguiente:

```
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;
```

Estructura del VCB: MC_DEALLOCATE (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_DEALLOCATE es la siguiente:

```
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;
```

Estructura del VCB: DEALLOCATE (Windows)

La definición de la estructura del VCB para el verbo DEALLOCATE es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_DEALLOCATE
Para el verbo MC_DEALLOCATE.

AP_B_DEALLOCATE
Para el verbo DEALLOCATE.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_DEALLOCATE.

AP_BASIC_CONVERSATION
Para el verbo DEALLOCATE.

Si el verbo se está utilizando en una conversación dúplex o se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con uno de los siguientes valores, o con ambos:

AP_FULL_DUPLEX_CONVERSATION
El verbo se está emitiendo en una conversación dúplex.

AP_NON_BLOCKING
El verbo se está emitiendo como un verbo de no bloqueo.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

dealloc_type
Especifica cómo realizar la desasignación. Los valores posibles se indican a continuación.

Si se utiliza AP_ABEND o cualquiera de los valores AP_ABEND_* la conversación se desasigna anormalmente. Si la conversación se encuentra en estado Enviar cuando el TP local emite el verbo [MC_]DEALLOCATE, APPC envía el contenido del almacenamiento intermedio de envío de la LU local al TP asociado antes de desasignar la conversación. Si la conversación se encuentra en estado Recibir o Pendiente-Envío, APPC elimina los datos entrantes innecesarios antes de desasignar la conversación.

AP_ABEND
Este valor únicamente es válido para el verbo MC_DEALLOCATE. Un TP debe especificar AP_ABEND cuando encuentra un error que impide la correcta finalización de una transacción.

AP_ABEND_PROG
Este valor únicamente es válido para el verbo DEALLOCATE. Una aplicación o un TP de servicio debe especificar AP_ABEND_PROG cuando encuentra un error que impide la correcta finalización de la transacción.

AP_ABEND_SVC
Un TP de servicio debe especificar AP_ABEND_SVC cuando encuentra un error causado por su TP de servicio asociado (por ejemplo, un error de formato en la información de control enviada por el TP de servicio asociado).

AP_ABEND_TIMER
Un TP de servicio debe especificar AP_ABEND_TIMER cuando encuentra un error que requiere la desasignación inmediata (por ejemplo un operador que finaliza el programa antes de lo debido).

AP_FLUSH
Envía el contenido del almacenamiento intermedio de envío de la LU local al TP asociado antes de desasignar la conversación. Este valor únicamente es válido si el estado de la conversación es Enviar o Enviar-Pendiente.

AP_CONFIRM_TYPE
Utilice este valor únicamente si el nivel de sincronización de la conversación es AP_SYNCPT. Indica que antes de desasignar la conversación se requiere la confirmación del TP asociado (pero no el proceso de punto de sincronización).

APPC envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al TP asociado. Una vez recibida la confirmación del TP asociado, APPC desasigna la conversación. Sin embargo, si el TP asociado informa de un error, la conversación sigue asignada.

AP_SYNC_LEVEL
Utiliza el nivel de sincronización de la conversación (establecido por el verbo [MC_]ALLOCATE) para determinar cómo desasignar la conversación. Este valor únicamente es válido si el estado de la conversación es Enviar o Enviar-Pendiente.

Si el nivel de sincronización de la conversación es AP_NONE, APPC envía el contenido del almacenamiento intermedio de envío de la LU local al TP asociado antes de desasignar la conversación.

Si el nivel de sincronización es AP_CONFIRM_SYNC_LEVEL, APPC envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al TP asociado. Una vez recibida la confirmación del TP asociado, APPC desasigna la conversación. Sin embargo, si el TP asociado informa de un error, la conversación sigue asignada.

AIX, LINUX

Si el nivel de sincronización de la conversación es AP_SYNCPT, APPC envía el contenido del almacenamiento intermedio de envío de la LU local al TP asociado antes de desasignar la conversación. El gestor de puntos de sincronización se encarga de lo siguiente:

- Interceptar el verbo [MC_]DEALLOCATE cuando se especifica el parámetro *dealloc_type* definido en AP_SYNC_LEVEL.
- Efectuar el proceso de punto de sincronización necesario.
- Pasar el verbo [MC_]DEALLOCATE original a Communications Server para Linux cuando el proceso de punto de sincronización ha finalizado.

Cuando Communications Server para Linux recibe el verbo [MC_]DEALLOCATE con *dealloc_type* definido en AP_SYNC_LEVEL en una conversación con *sync_level* definido en AP_SYNCPT, supone que el gestor de puntos de sincronización ya ha efectuado todo el proceso de punto de sincronización necesario y procesa el verbo igual que para el parámetro *sync_level* definido en AP_NONE.

AP_TP_NOT_AVAIL_RETRY
Este valor sólo debe ser utilizado por un TP que haya emitido RECEIVE_ALLOCATE con un nombre de TP en blanco (a fin de aceptar conversaciones entrantes para cualquier nombre de TP). Indica que el TP identificado por el nombre de TP especificado en la petición Attach entrante no está disponible debido a una condición temporal. Se informará del error al TP asociado con los códigos de retorno AP_ALLOCATION_FAILURE y AP_TRANS_PGM_NOT_AVAIL_RETRY; el TP asociado puede volver a intentar la petición de asignación.

AP_TP_NOT_AVAIL_NO_RETRY
Este valor sólo debe ser utilizado por un TP que haya emitido RECEIVE_ALLOCATE con un nombre de TP en blanco (a fin de aceptar conversaciones entrantes para cualquier nombre de TP). Indica que el TP identificado por el nombre de TP especificado en la petición Attach entrante no está disponible debido a una condición que necesita corregirse (como por ejemplo un problema de configuración). Se informará del error al TP asociado con los códigos de retorno AP_ALLOCATION_FAILURE y AP_TRANS_PGM_NOT_AVAIL_NO_RETRY; el TP asociado no debe volver a intentar la petición de asignación hasta que la condición que ha causado esta desasignación se haya corregido.

AP_TPN_NOT_RECOGNIZED
Este valor sólo debe ser utilizado por un TP que haya emitido RECEIVE_ALLOCATE con un nombre de TP en blanco (a fin de aceptar conversaciones entrantes para cualquier nombre de TP). Indica que el nombre de TP especificado en la petición Attach entrante no se ha reconocido como un nombre de TP válido. Se informará del error al TP asociado con los códigos de retorno AP_ALLOCATION_FAILURE y AP_TP_NAME_NOT_RECOGNIZED.

AP_PIP_DATA_NOT_ALLOWED
Este valor indica que el TP local está desasignando la conversación porque el TP asociado ha suministrado datos PIP en el verbo [MC_]ALLOCATE pero el TP local no esperaba recibirlos. Se informará del error al TP asociado con los códigos de retorno AP_ALLOCATION_FAILURE y AP_PIP_NOT_ALLOWED.

AP_PIP_DATA_INCORRECT
Este valor indica que el TP local está desasignando la conversación porque esperaba recibir datos PIP del TP asociado, pero el TP asociado ha suministrado datos PIP incorrectos o no ha proporcionado datos PIP. Se informará del error al TP asociado con los códigos de retorno AP_ALLOCATION_FAILURE y AP_PIP_NOT_SPECIFIED_CORRECTLY.

AP_RESOURCE_FAILURE_NO_RETRY
Este valor indica que el TP local está desasignando la conversación porque un recurso necesario para que el TP opere ha fallado.

AP_CONV_TYPE_MISMATCH
Este valor indica que el TP local está desasignando la conversación porque no soporta el tipo de conversación (correlacionada o básica) especificado por el TP asociado en el verbo [MC_]ALLOCATE. Se informará del error al TP asociado con los códigos de retorno AP_ALLOCATION_FAILURE y AP_CONVERSATION_TYPE_MISMATCH.

AP_SYNC_LVL_NOT_SUPPORTED
Este valor indica que el TP local está desasignando la conversación porque no soporta el nivel de sincronización especificado por el TP asociado en el verbo [MC_]ALLOCATE. Se informará del error al TP asociado con los códigos de retorno AP_ALLOCATION_FAILURE y AP_SYNC_LEVEL_NOT_SUPPORTED.

AP_SECURITY_PARAMS_INVALID
Este valor indica que el TP local está desasignando la conversación porque no ha aceptado los parámetros *security* especificados por el TP asociado en el verbo [MC_]ALLOCATE. Se informará del error al TP asociado con los códigos de retorno AP_ALLOCATION_FAILURE y AP_SECURITY_NOT_VALID.

log_dlen
Número de bytes de datos para enviar al archivo de anotaciones de error. El rango de este valor es 0–32.767.

Sólo el verbo DEALLOCATE utiliza este parámetro, con el parámetro *dealloc_type* definido en AP_ABEND_PGM, AP_ABEND_SVC o AP_ABEND_TIMER. Para el verbo MC_DEALLOCATE u otros valores de *dealloc_type*, este parámetro debe ser 0 (cero).

log_dptr
Dirección de almacenamiento intermedio de datos que contiene información de error. Estos datos se envían al archivo de anotaciones de error local y a la LU asociada.

El verbo DEALLOCATE utiliza este parámetro si *log_dlen* es superior a 0 (cero); de lo contrario está reservado.

El TP debe dar formato a los datos de error como una variable de anotaciones de error de corriente de datos general (GDS). Si desea obtener más información, consulte la publicación de IBM *Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*.

WINDOWS

El almacenamiento intermedio de datos de anotaciones puede residir en un área de datos estática o en un área asignada globalmente. El almacenamiento intermedio de datos debe caber completamente en esta área.

AIX, LINUX

Los parámetros siguientes se utilizan si el nivel de sincronización de la conversación es AP_SYNCPT; de lo contrario están reservados.

callback
Si el TP requiere la notificación de "FORGET" implícito, este parámetro especifica un puntero a una rutina callback que Communications Server para Linux llamará para proporcionar esta notificación. Si el TP no requiere esta notificación, no utiliza este parámetro. Si desea ver más información, consulte "Notificación de FORGET implícito" en la página 136.

correlator
Correlacionador opcional para ser usado por la aplicación. Este parámetro sólo se utiliza si el parámetro *callback* está especificado; de lo contrario está reservado.

Communications Server para Linux no utiliza este valor, sino que lo pasa como parámetro a la rutina callback con la notificación de "FORGET implícito". Este valor permite que la aplicación correlacione la información devuelta con su otro proceso.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

expd_rcvd
Indicador de datos acelerados. Este parámetro sólo se utiliza en las conversaciones dúplex, en que el TP puede seguir recibiendo datos acelerados después de emitir satisfactoriamente [MC_]DEALLOCATE.

Los valores posibles son:

AP_YES El TP asociado ha enviado datos acelerados que el TP local aún no ha recibido. Para recibir estos datos, el TP local puede utilizar el verbo [MC_]RECEIVE_EXPEDITED_DATA.

Este indicador se puede establecer en diversos verbos APPC. Seguirá establecido en los verbos posteriores hasta que el TP local emita el verbo [MC_]RECEIVE_EXPEDITED_DATA para recibir los datos.

AP_NO No hay datos acelerados a la espera de ser recibidos.

En una conversación semidúplex, siempre se establece AP_NO como valor de este parámetro, porque la conversación termina cuando el verbo [MC_]DEALLOCATE finaliza satisfactoriamente y, por lo tanto, el TP local no puede recibir más datos acelerados.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_DEALLOC_BAD_TYPE
El parámetro *dealloc_type* no está definido en un valor válido.

AP_DEALLOC_LOG_LL_WRONG
(Valor devuelto sólo para DEALLOCATE de conversación básica.) El campo LL de la variable de anotaciones de error GDS no coincide con la longitud real de los datos de anotaciones o el valor del parámetro *log_dlen* es incorrecto.

AIX, LINUX

AP_INVALID_FORMAT
El parámetro *format* está definido en un valor que no es válido.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

WINDOWS

AP_INVALID_DATA_SEGMENT
(Valor devuelto sólo para DEALLOCATE de conversación básica)

Los datos de anotaciones son más largos que el segmento de datos asignado o la dirección del almacenamiento intermedio de datos de anotaciones es incorrecta.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc
Los valores posibles son:

AP_DEALLOC_CONFIRM_BAD_STATE
La conversación no se encuentra en estado Enviar ni Enviar-Pendiente, y el TP ha intentado vaciar el almacenamiento intermedio de envío y enviar una petición de confirmación. Este intento es debido a que el valor del parámetro *dealloc_type* es AP_SYNC_LEVEL y el nivel de sincronización de la conversación es AP_CONFIRM_SYNC_LEVEL.

AP_DEALLOC_FLUSH_BAD_STATE
La conversación no se encuentra en estado Enviar ni Enviar-Pendiente, y el TP ha intentado vaciar el almacenamiento intermedio de envío. Este intento es debido a que el valor del parámetro *dealloc_type* es AP_FLUSH o el valor del parámetro *dealloc_type* es AP_SYNC_LEVEL y el nivel de sincronización de la conversación es AP_NONE.

AP_DEALLOC_NOT_LL_BDY
(Valor devuelto únicamente para el verbo DEALLOCATE de conversación básica.) La conversación se encuentra en estado Enviar y el TP no ha terminado de enviar un registro lógico. El parámetro *dealloc_type* está definido en AP_SYNC_LEVEL o AP_FLUSH.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

El verbo MC_DEALLOCATE devuelve el código de retorno primario siguiente:

primary_rc
AP_DEALLOC_ABEND

APPC no devuelve un código de retorno secundario con este código de retorno primario.

El verbo DEALLOCATE devuelve los códigos de retorno primarios siguientes:

primary_rc
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

En función del valor del parámetro *dealloc_type*, la conversación puede encontrarse en uno de los estados indicados en la tabla siguiente cuando el TP emite el verbo [MC_]DEALLOCATE.

dealloc_type	**Estado permitido**
AP_FLUSH	Enviar-Recibir (sólo conversaciones dúplex), Enviar o Enviar-Pendiente
AP_SYNC_LEVEL	Enviar-Recibir (sólo conversaciones dúplex), Enviar o Enviar-Pendiente
AP_ABEND AP_ABEND_PROG AP_ABEND_SVC AP_ABEND_TIMER	Cualquiera excepto Restablecer

Cambio de estado

Los cambios de estado, resumidos en la tabla siguiente, dependen del valor del parámetro *primary_rc*.

primary_rc	**Estado nuevo**
AP_OK	Sólo-Recibir (conversación dúplex con *dealloc_type* establecido en AP_FLUSH o AP_SYNC_LEVEL) o Restablecer (todos los demás casos)
AP_PARAMETER_CHECK 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	Sin cambio
AP_ALLOCATION_ERROR AP_CONV_FAILURE_RETRY AP_CONV_FAILURE_NO_RETRY	Restablecer
AP_DEALLOC_ABEND AP_DEALLOC_ABEND_PROG AP_DEALLOC_ABEND_SVC AP_DEALLOC_ABEND_TIMER	Restablecer
AP_PROG_ERROR_PURGING AP_SVC_ERROR_PURGING	Recibir

Notificación de FORGET implícito

AIX, LINUX

Los protocolos de punto de sincronización incluyen una función conocida como FORGET implícito, que significa que la cabecera PS FORGET (el último mensaje en un intercambio de punto de sincronización) no siempre es necesaria. Cuando el protocolo requiere un mensaje FORGET como el próximo mensaje que debe recibirse en una sesión, el próximo flujo de datos recibido en esta sesión implica que el mensaje FORGET se ha recibido.

Sin embargo, si el mensaje que precede el mensaje FORGET indica que se está desasignando la conversación, la aplicación ya no tiene acceso a la sesión y por consiguiente no puede saber cuándo se produce el próximo flujo de datos. Para proporcionar esta información, Communications Server para Linux permite que la aplicación especifique una rutina callback en el verbo [MC_]DEALLOCATE; Communications Server para Linux llama a esta rutina cuando se produce el próximo flujo de datos en la sesión o cuando finaliza la sesión (normal o anormalmente).

Si una aplicación utiliza esta función, debe esperar a que se llame a la rutina callback antes de emitir el verbo TP_ENDED para este TP. Communications Server para Linux no llamará a la rutina callback después de que se haya emitido el verbo TP_ENDED.

La rutina callback se define de la siguiente forma:

```
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 para Linux llama a la rutina con los siguientes parámetros:

vcb Puntero al VCB de [MC_]DEALLOCATE original suministrado por la aplicación. Si la aplicación necesita utilizar los parámetros del VCB en la rutina callback, no debe liberar ni volver a utilizar la memoria asociada con el VCB hasta que se haya llamado a la rutina callback.

tp_id Identificador de TP de 8 bytes del TP en que se ha emitido el verbo.

conv_id
Identificador de conversación de la conversación en que se ha emitido el verbo. La aplicación no puede emitir más verbos utilizando este identificador de conversación, puesto que ya no es válido una vez que ha finalizado el verbo [MC_]DEALLOCATE.

type Tipo de flujo de mensaje de que informa Communications Server para Linux. Los valores posibles son:

AP_DATA_FLOW
Flujo de datos normal en la sesión.

AP_UNBIND
La sesión ha finalizado normalmente.

AP_FAILURE
La sesión ha finalizado anormalmente. El gestor de puntos de sincronización puede que tenga que efectuar una resincronización.

corr Valor de correlacionador suministrado por la aplicación. Este valor permite que la aplicación correlacione la información devuelta con su otro proceso.

La rutina callback no tiene que utilizar todos estos parámetros. Puede ejecutar todo el proceso necesario en el VCB devuelto o simplemente puede establecer una variable para informar al programa principal de que el verbo ha finalizado.

Si la aplicación utiliza la planificación por señales, la rutina callback se ejecuta en el contexto de un receptor de señales. Esto significa que hay limitaciones en las llamadas del sistema operativo que se pueden utilizar dentro de la rutina; para obtener más información consulte la documentación del sistema operativo.

La aplicación puede emitir más verbos APPC desde dentro de la rutina callback si es necesario. Sin embargo, éstos deben ser verbos asíncronos. Los verbos síncronos emitidos desde una rutina callback se rechazarán con los códigos de retorno AP_PARAMETER_CHECK y AP_SYNC_NOT_ALLOWED.

MC_FLUSH y FLUSH

El verbo MC_FLUSH o FLUSH envía el contenido del almacenamiento intermedio de envío de la LU local a la LU asociada (y al TP). Si el almacenamiento intermedio de envío está vacío, no se lleva a cabo ninguna acción.

Orígenes de los datos del almacenamiento intermedio

Los datos procesados por el verbo [MC_]SEND_DATA y las peticiones de asignación generadas por el verbo [MC_]ALLOCATE se acumulan en el almacenamiento intermedio de envío de la LU local hasta que se produce una de las situaciones siguientes:

- El TP local emite el verbo [MC_]FLUSH (u otro verbo que vacíe el almacenamiento intermedio de envío de la LU).
- El almacenamiento intermedio se llena.

Estructura del VCB: MC_FLUSH

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_FLUSH es la siguiente:

```
typedef struct mc_flush
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado           */
  AP_UINT16         primary_rc;
```

```
  AP_UINT32      secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32         conv_id;
} MC_FLUSH;
```

Estructura del VCB: FLUSH

La definición de la estructura del VCB para el verbo FLUSH es la siguiente:

```
typedef struct flush
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  AP_UINT16         primary_rc;
  AP_UINT32      secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32         conv_id;
} FLUSH;
```

Estructura del VCB: MC_FLUSH (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_FLUSH es la siguiente:

```
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;
```

Estructura del VCB: FLUSH (Windows)

La definición de la estructura del VCB para el verbo FLUSH es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_FLUSH
Para el verbo MC_FLUSH.

AP_B_FLUSH
Para el verbo FLUSH.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_FLUSH.

AP_BASIC_CONVERSATION
Para el verbo FLUSH.

Si el verbo se está utilizando en una conversación dúplex o se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con uno de los siguientes valores, o con ambos:

AP_FULL_DUPLEX_CONVERSATION
El verbo se está emitiendo en una conversación dúplex.

AP_NON_BLOCKING
El verbo se está emitiendo como un verbo de no bloqueo.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AIX, LINUX

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc

AP_FLUSH_NOT_SEND_STATE
La conversación no se encuentra en estado Enviar ni Enviar-Pendiente.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación debe encontrarse en estado Enviar-Recibir (sólo en las conversaciones dúplex), Enviar o Enviar-Pendiente.

Cambio de estado

Después de una correcta ejecución, no hay ningún cambio de estado.

MC_GET_ATTRIBUTES y GET_ATTRIBUTES

El verbo MC_GET_ATTRIBUTES o GET_ATTRIBUTES devuelve los atributos de la conversación. Para obtener más detalles sobre estos atributos, consulte el Capítulo 1, “Conceptos”, en la página 1 de este manual o la publicación *Communications Server para Linux - Guía de administración*.

Estructura del VCB: MC_GET_ATTRIBUTES

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_GET_ATTRIBUTES es la siguiente:

```
typedef struct mc_get_attributes
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: GET_ATTRIBUTES

La definición de la estructura del VCB para el verbo GET_ATTRIBUTES es la siguiente:

```
typedef struct get_attributes
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: MC_GET_ATTRIBUTES (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_GET_ATTRIBUTES es la siguiente:

```
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;
```

Estructura del VCB: GET_ATTRIBUTES (Windows)

La definición de la estructura del VCB para el verbo GET_ATTRIBUTES es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_GET_ATTRIBUTES
Para el verbo MC_GET_ATTRIBUTES.

AP_B_GET_ATTRIBUTES
Para el verbo GET_ATTRIBUTES.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_GET_ATTRIBUTES.

AP_BASIC_CONVERSATION
Para el verbo GET_ATTRIBUTES.

Si el verbo se está emitiendo en una conversación dúplex, combine el valor anterior (utilizando un operador lógico OR) con el valor AP_FULL_DUPLEX_CONVERSATION.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes: Para obtener más información sobre el significado y los usos de estos parámetros, consulte la publicación *Communications Server para Linux - Guía de administración*.

primary_rc
`AP_OK`

sync_level
Nivel de sincronización de la conversación. Este parámetro determina si los TP pueden solicitar la confirmación de recepción de datos y confirmar la recepción de datos.

Los valores posibles son:

AP_CONFIRM_SYNC_LEVEL
Los TP pueden utilizar el proceso de confirmación en esta conversación.

AP_SYNCPT
Los TP pueden utilizar las funciones de punto de sincronización de LU 6.2 en esta conversación. Para ver más información, consulte “Soporte de punto de sincronización” en la página 23.

AP_NONE
El proceso de confirmación no se utilizará en esta conversación.

mode_name
Nombre de un conjunto de características de red.

Este parámetro es una cadena de caracteres EBCDIC de 8 bytes. Puede constar de caracteres del juego de caracteres EBCDIC de tipo A. Estos caracteres son los siguientes:

- Letras en mayúsculas
- Números del 0 al 9
- Los caracteres especiales $, # y @

net_name
Nombre de la red que contiene la LU local.

Este parámetro es una cadena de caracteres EBCDIC de 8 bytes. Puede constar de caracteres del juego de caracteres EBCDIC de tipo A. Estos caracteres son los siguientes:

- Letras en mayúsculas
- Números del 0 al 9

- Los caracteres especiales $, # y @

lu_name
Nombre de la LU local.

Este parámetro es una cadena de caracteres EBCDIC de 8 bytes. Puede constar de caracteres del juego de caracteres EBCDIC de tipo A. Estos caracteres son los siguientes:

- Letras en mayúsculas
- Números del 0 al 9
- Los caracteres especiales $, # y @

lu_alias
Alias por el que se conoce la LU local en el TP local. Se trata de una cadena de caracteres ASCII de 8 bytes.

plu_alias
Alias por el que se conoce la LU asociada en el TP local. Se trata de una cadena de caracteres ASCII de 8 bytes.

plu_un_name
Nombre sin interpretar de la LU asociada—el nombre de la LU asociada definido en el SSCP (punto de control de servicios del sistema). Se obtiene de la configuración de la LU remota en el archivo de configuración de Communications Server para Linux. Este parámetro es necesario en la configuración únicamente si la LU local es dependiente, por lo que el nombre devuelto para una LU independiente puede ser blanco o nulo.

Este parámetro es una cadena de caracteres EBCDIC de 8 bytes sensible a las mayúsculas y minúsculas. Puede constar de caracteres del juego de caracteres EBCDIC de tipo AE. Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

fqplu_name
Nombre de la LU asociada completamente calificado.

Este campo contiene el nombre de red, un punto EBCDIC y el nombre de LU asociada. Cada uno de los dos nombres es una cadena de caracteres de 8 bytes que puede constar de caracteres del juego de caracteres EBCDIC de tipo A. Estos caracteres son los siguientes:

- Letras en mayúsculas
- Números del 0 al 9
- Los caracteres especiales $, # y @

user_id Identificador de usuario enviado por el TP que invoca mediante el verbo [MC_]ALLOCATE para acceder al TP invocado (si corresponde).

Este parámetro es una cadena de caracteres EBCDIC de 10 bytes sensible a las mayúsculas y minúsculas. Puede constar de caracteres del juego de caracteres EBCDIC de tipo AE. Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

Este campo contiene el identificador de usuario si se cumplen las condiciones siguientes:

- El TP invocado necesita seguridad de conversación.
- Este verbo ha sido emitido por el TP invocado.

De lo contrario, este campo contiene blancos.

conv_group_id
Identificador de grupo de conversación de la sesión que utiliza esta conversación.

conv_corr_len
Longitud (0–8 bytes) del correlacionador de conversación (consulte la descripción del parámetro *conv_corr* para obtener más información).

conv_corr
Correlacionador de conversación asignado por el nodo del TP que invoca cuando se ha asignado la conversación.

AIX, LINUX

Para los TP que utilizan el proceso de punto de sincronización, el gestor de puntos de sincronización utiliza este parámetro para identificar la conversación durante el proceso de resincronización.

luw_id LUWID (identificador de unidad lógica de trabajo) para la transacción en que participa esta conversación. El LUWID se asigna en nombre del TP que inicia la transacción, y permite correlacionar las diferentes conversaciones que forman la transacción.

Un TP que utiliza el proceso de punto de sincronización tiene dos LUWID asociados; el LUWID no protegido se utiliza para conversaciones con *sync_level* definido en `AP_NONE` o `AP_CONFIRM_SYNC_LEVEL`, y el LUWID protegido se utiliza para conversaciones con *sync_level* definido en `AP_SYNCPT`. Un TP que no utiliza el proceso de punto de sincronización sólo tiene uno, el LUWID no protegido. Este verbo devuelve el LUWID que está asociado con esta conversación; es el LUWID protegido si la conversación tiene un parámetro *sync_level* definido en `AP_SYNCPT` y, de lo contrario, el LUWID no protegido. La aplicación puede utilizar el verbo GET_TP_PROPERTIES para obtener ambos LUWID para el TP.

El LUWID consta de los siguientes parámetros:

luw_id.fq_length
Longitud (1–17 bytes) del nombre de LU completamente calificado asociado con la unidad lógica de trabajo (el parámetro siguiente especifica el nombre de LU en sí).

luw_id.fq_luw_name
Nombre de LU completamente calificado asociado con la unidad lógica de trabajo. Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC. Consta de un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.

luw_id.instance
Número de instancia asociado con la unidad lógica de trabajo (un número binario de 6 bytes).

luw_id.sequence
Número de secuencia del segmento actual de la unidad lógica de trabajo (un número binario de 2 bytes).

sess_id Indicador de sesión de la sesión utilizada por esta conversación.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AIX, LINUX

AP_INVALID_FORMAT
El parámetro *format* está definido en un valor que no es válido.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

El estado de conversación no cambia con este verbo.

MC_PREPARE_TO_RECEIVE y PREPARE_TO_RECEIVE

El verbo MC_PREPARE_TO_RECEIVE o PREPARE_TO_RECEIVE cambia el estado de la conversación para el TP local de Enviar o Enviar-Pendiente a Recibir.

Nota: Este verbo sólo se puede utilizar en las conversaciones semidúplex; no es válido para las conversaciones dúplex.

Antes de cambiar el estado de conversación, este verbo realiza el equivalente de uno de los verbos siguientes, en función del parámetro *ptr_type* (tipo de preparación para recepción) como se describe a continuación:

- El verbo [MC_]FLUSH, que envía el contenido del almacenamiento intermedio de envío de la LU local a la LU asociada (y al TP).
- El verbo [MC_]CONFIRM, que envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al TP asociado.

Una vez que este verbo se ha ejecutado correctamente, el TP local puede recibir datos.

Estructura del VCB: MC_PREPARE_TO_RECEIVE

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_PREPARE_TO_RECEIVE es la siguiente:

```
typedef struct mc_prepare_to_receive
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                  /* Reservado           */
  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;
```

Estructura del VCB: PREPARE_TO_RECEIVE

La definición de la estructura del VCB para el verbo PREPARE_TO_RECEIVE es la siguiente:

```
typedef struct prepare_to_receive
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: MC_PREPARE_TO_RECEIVE (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_PREPARE_TO_RECEIVE es la siguiente:

```
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;
```

Estructura del VCB: PREPARE_TO_RECEIVE (Windows)

La definición de la estructura del VCB para el verbo PREPARE_TO_RECEIVE es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_PREPARE_TO_RECEIVE
Para el verbo MC_PREPARE_TO_RECEIVE.

AP_B_PREPARE_TO_RECEIVE
Para el verbo PREPARE_TO_RECEIVE.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_PREPARE_TO_RECEIVE.

AP_BASIC_CONVERSATION
Para el verbo PREPARE_TO_RECEIVE.

Si el verbo se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con el valor AP_NON_BLOCKING.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

ptr_type
Especifica cómo realizar el cambio de estado.

Los valores posibles son:

AP_FLUSH
Envía el contenido del almacenamiento intermedio de envío de la LU local a la LU asociada (y al TP) antes de cambiar el estado de la conversación a Recibir.

AIX, LINUX

AP_CONFIRM_TYPE
Utilice este valor únicamente si el nivel de sincronización de la conversación es AP_SYNCPT. Indica que antes de cambiar el estado de la conversación a Recibir se requiere la confirmación del TP asociado (pero no el proceso de punto de sincronización).

APPC envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al TP asociado. El estado de conversación no cambia a Recibir hasta que el TP asociado envía la confirmación solicitada (o informa de un error).

AP_SYNC_LEVEL
Utiliza el nivel de sincronización de la conversación (establecido por el verbo [MC_]ALLOCATE) para determinar cómo realizar el cambio de estado.

Si el nivel de sincronización de la conversación es AP_NONE, APPC envía el contenido del almacenamiento intermedio de envío de la LU local al TP asociado antes de cambiar el estado de la conversación a Recibir.

Si el nivel de sincronización es AP_CONFIRM_SYNC_LEVEL, APPC envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al TP asociado. Una vez recibida la confirmación del TP asociado, APPC cambia el estado de la conversación a Recibir. Sin embargo, si el TP asociado informa de un error, el estado cambia a Recibir o Restablecer; consulte "Cambio de estado" en la página 154.

AIX, LINUX

Si el nivel de sincronización de la conversación es AP_SYNCPT, APPC envía el contenido del almacenamiento intermedio de envío de la LU local al TP asociado antes de cambiar el estado de conversación. El gestor de puntos de sincronización se encarga de lo siguiente:

- Interceptar el verbo [MC_]PREPARE_TO_RECEIVE cuando se especifica el parámetro *ptr_type* definido en AP_SYNC_LEVEL.
- Efectuar el proceso de punto de sincronización necesario.
- Pasar el verbo [MC_]PREPARE_TO_RECEIVE original a Communications Server para Linux cuando el proceso de punto de sincronización ha finalizado.

Cuando Communications Server para Linux recibe el verbo [MC_]PREPARE_TO_RECEIVE con *ptr_type* definido en AP_SYNC_LEVEL en una conversación con *sync_level* definido en AP_SYNCPT, supone que el gestor de puntos de sincronización ya ha efectuado todo el proceso de punto de sincronización necesario y procesa el verbo igual que para *sync_level* definido en AP_NONE.

locks Especifica cuándo APPC debe devolver el control al TP local.

Utilice este parámetro únicamente si *ptr_type* está definido en AP_SYNC_LEVEL y el nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL. (De lo contrario, no se tiene en cuenta el parámetro.)

Los valores posibles son:

AP_LONG
APPC devuelve el control al TP local cuando la confirmación y los datos posteriores del TP asociado llegan a la LU local. (Con este método se consigue utilizar la red de forma más eficaz pero se tarda más en devolver el control al TP local.)

AP_SHORT
APPC devuelve el control al TP local cuando la confirmación del TP asociado llega a la LU local.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_P_TO_R_INVALID_FOR_FDX
El TP local ha intentado utilizar el verbo [MC_]PREPARE_TO_RECEIVE en una conversación dúplex. Este verbo sólo se puede utilizar en una conversación semidúplex.

AP_P_TO_R_INVALID_TYPE
El parámetro *ptr_type* no está definido en un valor válido.

AIX, LINUX

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc
Los valores posibles son:

AP_P_TO_R_NOT_LL_BDY
(Valor devuelto únicamente para el verbo PREPARE_TO_RECEIVE.) El TP local no ha terminado de enviar un registro lógico.

AP_P_TO_R_NOT_SEND_STATE
La conversación no se encuentra en estado Enviar ni Enviar-Pendiente.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

El verbo MC_PREPARE_TO_RECEIVE devuelve el código de retorno primario siguiente:

primary_rc

AP_DEALLOC_ABEND

El verbo PREPARE_TO_RECEIVE devuelve los códigos de retorno primarios siguientes:

primary_rc

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación debe encontrarse en estado Enviar o Enviar-Pendiente.

Cambio de estado

Los cambios de estado, resumidos en la tabla siguiente, dependen del valor del parámetro *primary_rc*.

primary_rc	**Estado nuevo**
AP_OK	Recibir
AP_PARAMETER_CHECK	Sin cambio
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	Restablecer
AP_CONV_FAILURE_RETRY	Restablecer
AP_CONV_FAILURE_NO_RETRY	
AP_DEALLOC_ABEND_RESET	Restablecer
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	
AP_PROG_ERROR_PURGING_RECEIVE	Recibir
AP_SVC_ERROR_PURGING	

Nota de uso

La conversación no cambia al estado Enviar o Enviar-Pendiente para el TP asociado hasta que éste recibe uno de los valores siguientes mediante el parámetro *what_rcvd* de un verbo de recepción posterior:

- AP_SEND, AP_DATA_SEND, AP_DATA_COMPLETE_SEND.
- AP_CONFIRM_SEND, AP_DATA_CONFIRM_SEND o AP_DATA_COMPLETE_CONFIRM_SEND (y responde con [MC_]CONFIRMED).

Los verbos RECEIVE son [MC_]RECEIVE_AND_WAIT, [MC_]RECEIVE_IMMEDIATE y [MC_]RECEIVE_AND_POST.

Verbos MC_RECEIVE y RECEIVE

APPC proporciona tres tipos de verbos diferentes que se utilizan para recibir datos del TP asociado. La mayoría de los parámetros y códigos de retorno son los mismos para los tres verbos, pero cada uno opera de modo distinto y realiza una función distinta. En este apartado se explica la información común para los tres verbos, y más adelante se explica cada verbo con más detalle.

Los tres verbos RECEIVE son:

- [MC_]RECEIVE_IMMEDIATE
- [MC_]RECEIVE_AND_WAIT
- [MC_]RECEIVE_AND_POST

Nota: El verbo [MC_]RECEIVE_EXPEDITED_DATA también recibe datos del TP asociado, pero recibe datos que se han enviado como datos de flujo acelerado en lugar de datos de flujo normal. Este verbo se describe por separado tras los demás verbos de recepción.

Cómo recibe datos un TP

El proceso por el que el TP local recibe datos es el siguiente:

1. El TP local emite un verbo de recepción hasta que termina de recibir una unidad de datos completa. Los datos recibidos pueden ser:
   - Un registro de datos transmitido en una conversación correlacionada.
   - Un registro lógico transmitido en una conversación básica.
   - Un almacenamiento intermedio de datos recibido aparte de su formato de registro lógico en una conversación básica.

   Puede que el TP local tenga que emitir varias veces el verbo RECEIVE para recibir una unidad de datos completa. Una vez que se ha recibido una unidad de datos completa, el TP local puede manipularla.
2. El TP local emite otro verbo de recepción. Esto tiene uno de los efectos siguientes:
   - Si el TP asociado ha enviado más datos, el TP local empieza a recibir una unidad de datos nueva.
   - Si el TP asociado ha terminado de enviar datos o está esperando la confirmación, la información de estado (disponible mediante el parámetro *what_rcvd*) indica la siguiente acción que normalmente lleva a cabo el TP local. Para ver más información, consulte "Parámetro what_rcvd".

El TP local también puede definir un parámetro *rtn_status* cuando emite el verbo de recepción; esto indica que se devolverá cualquier información de estado disponible con los datos. En este caso, el verbo de recepción que devuelve la última parte de los datos también devuelve la información de estado, y el TP local no tiene que emitir un verbo de recepción aparte para ésta. Para ver más información, consulte "Parámetro what_rcvd".

Parámetro what_rcvd

Después de emitir uno de los verbos [MC_]RECEIVE, un TP normalmente utilizará el parámetro *what_rcvd* para determinar su próxima acción. Los valores que hacen referencia a un tipo de datos de "control de usuario" se devolverán en una conversación correlacionada en el sistema AIX o Linux, y los valores referentes a un tipo de datos de "cabecera PS" se devolverán en una conversación correlacionada en el sistema AIX o Linux con el nivel de sincronización `AP_SYNCPT`.

En la lista siguiente se describen los valores posibles del parámetro *what_rcvd*, con la acción que el TP normalmente lleva a cabo para cada uno de ellos:

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
: El TP local ha recibido datos del TP asociado. Normalmente sigue emitiendo verbos RECEIVE hasta que recibe uno de los demás parámetros *what_rcvd* de esta lista.

AP_SEND (sólo conversaciones semidúplex)
: El TP asociado ha emitido el verbo [MC_]PREPARE_TO_RECEIVE sin solicitar confirmación o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío PREPARE_TO_RECEIVE. Ahora el TP local se encuentra en estado Enviar, por lo que normalmente empezará a enviar datos.

AP_CONFIRM_DEALLOCATE (sólo conversaciones semidúplex)
: El TP asociado ha emitido el verbo [MC_]DEALLOCATE con un parámetro *dealloc_type* que indicaba que se necesitaba la confirmación, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío DEALLOCATE. Ahora el TP

local se encuentra en estado Confirmar-Desasignar, por lo que normalmente emitirá el verbo [MC_]CONFIRMED para confirmar la desasignación de la conversación.

AP_CONFIRM_SEND (sólo conversaciones semidúplex)
El TP asociado ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con los parámetros *ptr_type* y *dealloc_type* que indicaban que se necesitaba una confirmación o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío PREPARE_TO_RECEIVE_CONFIRM. Ahora el TP local se encuentra en estado Confirmar-Enviar, por lo que normalmente emitirá el verbo [MC_]CONFIRMED para confirmar el cambio de estado y empezar a enviar datos.

AP_CONFIRM_WHAT_RECEIVED (sólo conversaciones semidúplex)
El TP asociado ha emitido el verbo [MC_]CONFIRM o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío CONFIRM. Ahora el TP local se encuentra en estado Confirmar, por lo que normalmente emitirá el verbo [MC_]CONFIRMED.

Los valores siguientes se devolverán únicamente si el TP local ha especificado `AP_YES` para el parámetro *rtn_status* (estado de retorno con datos):

AP_DATA_SEND AP_DATA_COMPLETE_SEND AP_UC_DATA_COMPLETE_SEND AP_PS_HDR_COMPLETE_SEND
El TP asociado ha enviado datos y después ha emitido el verbo [MC_]PREPARE_TO_RECEIVE sin solicitar confirmación, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío PREPARE_TO_RECEIVE. Ahora el TP local se encuentra en estado Enviar-Pendiente, por lo tanto empezará a enviar datos normalmente.

AP_DATA_CONFIRM_DEALLOCATE
AP_DATA_COMPLETE_CONFIRM_DEALL
AP_UC_DATA_COMPLETE_CNFM_DEALL
AP_PS_HDR_COMLETE_CNFM_DEALL
Todos estos valores sólo son válidos para las conversaciones semidúplex.

El TP asociado ha enviado datos y después ha emitido el verbo [MC_]DEALLOCATE con un parámetro *dealloc_type* que indicaba que se necesitaba una confirmación, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío DEALLOCATE. Ahora el TP local se encuentra en estado Confirmar-Desasignar, por lo que normalmente emitirá el verbo [MC_]CONFIRMED para confirmar la desasignación de la conversación.

AP_DATA_CONFIRM_SEND, AP_DATA_COMPLETE_CONFIRM_SEND, AP_UC_DATA_COMPLETE_CNFM_SEND, AP_PS_HDR_COMPLETE_CNFM_SEND
Todos estos valores sólo son válidos para las conversaciones semidúplex.

El TP asociado ha enviado datos y después ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con los parámetros *ptr_type* y *dealloc_type* que indicaban que se necesitaba una confirmación o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío PREPARE_TO_RECEIVE_CONFIRM. Ahora el TP local se encuentra en estado Confirmar-Enviar, por lo que normalmente emitirá el verbo [MC_]CONFIRMED para confirmar el cambio de estado y empezar a enviar datos.

AP_DATA_CONFIRM, AP_DATA_COMPLETE_CONFIRM, AP_UC_DATA_COMPLETE_CONFIRM, AP_PS_HDR_COMPLETE_CONFIRM
Todos estos valores sólo son válidos para las conversaciones semidúplex.

El TP asociado ha enviado datos y después ha emitido el verbo [MC_]CONFIRM, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío CONFIRM. Ahora el TP local se encuentra en estado Confirmar, por lo que normalmente emitirá el verbo [MC_]CONFIRMED.

En todos los casos de CONFIRM anteriores, el TP puede emitir el verbo [MC_]SEND_ERROR en lugar del verbo [MC_]CONFIRMED, para indicar que se ha detectado un error en los datos suministrados o en el proceso. Si emite [MC_]SEND_ERROR en estado Enviar-Pendiente (después de recibir AP_DATA_SEND, AP_DATA_COMPLETE_SEND, AP_UC_DATA_COMPLETE_SEND o AP_PS_HDR_COMPLETE_SEND), puede especificar si el error se ha detectado en los datos suministrados o en sus propios datos o proceso. Para ver más información, consulte la descripción del verbo [MC_]SEND_ERROR en "MC_SEND_ERROR y SEND_ERROR" en la página 228.

Fin de los datos

Si el TP local emite uno de los verbos RECEIVE de conversación básica y define el parámetro *fill* en AP_BUFFER, la recepción de los datos finaliza cuando se ha alcanzado el valor de *max_len* o el fin de los datos. El fin de los datos se indica mediante uno de los elementos siguientes:

- Un parámetro *primary_rc* con un valor diferente de AP_OK (por ejemplo, AP_DEALLOC_NORMAL).
- Un parámetro *what_rcvd* que incluye SEND, CONFIRM, CONFIRM_SEND o CONFIRM_DEALLOCATE.

Para determinar si se ha alcanzado el fin de los datos, el TP local vuelve a emitir uno de los verbos RECEIVE. Si el nuevo parámetro *primary_rc* contiene AP_OK y *what_rcvd* contiene AP_DATA o AP_DATA_INCOMPLETE, no se ha alcanzado el fin de los datos. Sin embargo, si se ha alcanzado el fin de los datos, el parámetro *primary_rc* o *what_rcvd* indicará el motivo del fin de los datos.

Comprobación del parámetro what_rcvd

El TP local puede utilizar cualquiera de los verbos [MC_]RECEIVE para determinar si el TP asociado tiene datos para enviar, requiere una confirmación o ha cambiado el estado de conversación, sin recibir datos. Para ello, emite el verbo [MC_]RECEIVE con el parámetro *max_len* definido en 0 (cero), y después (si el verbo vuelve con un parámetro *primary_rc* definido en AP_OK) comprueba el parámetro *what_rcvd*.

MC_RECEIVE_AND_POST y RECEIVE_AND_POST

El verbo MC_RECEIVE_AND_POST o RECEIVE_AND_POST recibe datos de aplicación e información de estado de manera asíncrona. De esta forma el TP local puede seguir procesando mientras llegan datos a la LU local.

Nota: Este verbo sólo se puede utilizar en las conversaciones semidúplex; no es válido para las conversaciones dúplex.

Estructura del VCB: MC_RECEIVE_AND_POST

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_RECEIVE_AND_POST es la siguiente:

```
typedef struct mc_receive_and_post
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: RECEIVE_AND_POST

La definición de la estructura del VCB para el verbo RECEIVE_AND_POST es la siguiente:

```
typedef struct receive_and_post
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: MC_RECEIVE_AND_POST (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_RECEIVE_AND_POST es la siguiente:

```
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;
```

Estructura del VCB: RECEIVE_AND_POST (Windows)

La definición de la estructura del VCB para el verbo RECEIVE_AND_POST es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_RECEIVE_AND_POST
Para el verbo MC_RECEIVE_AND_POST.

AP_B_RECEIVE_AND_POST
Para el verbo RECEIVE_AND_POST.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_RECEIVE_AND_POST.

AP_BASIC_CONVERSATION
Para el verbo RECEIVE_AND_POST.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

rtn_status
Indica si la información de estado y los datos pueden devolverse en el mismo verbo.

Los valores posibles son:

AP_YES La información de estado, si está disponible, se devuelve con la última parte de un registro de datos.

AP_NO La información de estado no se devuelve con los datos. Después de recibir el fin de un registro de datos, el TP local debe emitir otro verbo [MC_]RECEIVE para obtener la información de estado.

fill Indica la manera en que el TP local recibe datos.

Este parámetro sólo es utilizado por el verbo RECEIVE_AND_POST de conversación básica.

Los valores posibles son:

AP_BUFFER
El TP local recibe datos hasta que se alcanza el número de bytes especificado por el parámetro *max_len* o hasta el fin de los datos. Los datos se reciben sin tener en cuenta el formato de registro lógico.

AP_LL Los datos se reciben con el formato de registro lógico. Los datos recibidos pueden ser:

- Un registro lógico completo.
- Una parte de un registro lógico (el número de bytes especificado por *max_len*).
- El fin de un registro lógico.

max_len
Número máximo de bytes de datos que puede recibir el TP local.

El rango de este valor es `0–65.535`.

Este valor no debe sobrepasar la longitud del almacenamiento intermedio que contendrá los datos recibidos.

dptr Dirección del almacenamiento intermedio que contendrá los datos recibidos por el TP local.

AIX, LINUX

callback
Dirección de la rutina callback que APPC llamará cuando haya finalizado la operación de recepción asíncrona. Para ver más información, consulte "Notas de uso" en la página 170.

WINDOWS

sema Descriptor de contexto de sucesos de Windows que se obtiene llamando a una de las dos funciones siguientes de Windows: `CreateEvent` o `OpenEvent`.

APPC señala este descriptor de contexto de sucesos para informar al TP de que la operación de recepción asíncrona ha finalizado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Al emitir este verbo, éste vuelve inmediatamente con un parámetro *primary_rc* que indica si el verbo se ha emitido correctamente o no. Los únicos parámetros devueltos que son válidos en este punto son *primary_rc*, *secondary_rc* (si *primary_rc* no tiene el valor AP_OK), y *rts_rcvd*. Los valores posibles de *primary_rc* y *secondary_rc* se describen más adelante en este apartado.

Si este parámetro *primary_rc* está definido en AP_OK, el verbo ha empezado a recibir datos de manera asíncrona de forma correcta. Cuando el verbo ha finalizado (debido a que ha recibido los datos correctamente o bien porque ha finalizado por un error de conversación), APPC llama a la rutina callback suministrada. En este punto, los parámetros devueltos son los que se muestran a continuación. Los parámetros *primary_rc* y *secondary_rc* tendrán nuevos valores que indicarán si se han recibido o no los datos correctamente, y deberán volverse a examinar.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

what_rcvd
Información de estado recibida con los datos entrantes.

La siguiente acción que emprende el TP normalmente dependerá del valor de este parámetro. Para ver más información, consulte "Parámetro what_rcvd" en la página 156.

Los valores posibles son:

AP_CONFIRM_DEALLOCATE
El TP asociado ha emitido el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* establecido en AP_SYNC_LEVEL y el nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_CONFIRM_SEND
El TP asociado ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con el parámetro *ptr_type* establecido en AP_SYNC_LEVEL y el nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_CONFIRM_WHAT_RECEIVED
El TP asociado ha emitido el verbo [MC_]CONFIRM.

AP_DATA
Este valor puede ser devuelto por el verbo RECEIVE_AND_POST de conversación básica si el parámetro *fill* está definido en AP_BUFFER; no se aplica a MC_RECEIVE_AND_POST.

El TP local ha recibido datos hasta alcanzar *max_len* o el fin de los datos.

AP_DATA_COMPLETE
Para el verbo MC_RECEIVE_AND_POST, este valor indica que el TP local ha recibido un registro de datos completo o la última parte de un registro de datos. Para RECEIVE_AND_POST con el parámetro *fill* definido en AP_LL, este valor indica que el TP local ha recibido un registro lógico completo o el fin de un registro lógico.

AP_DATA_INCOMPLETE
Para MC_RECEIVE_AND_POST, este valor indica que el TP local ha recibido un registro de datos incompleto. El parámetro *max_len* ha especificado un valor inferior a la longitud del registro de datos (o inferior al registro de datos restante si no es el primer verbo de recepción que lee el registro).

Para RECEIVE_AND_POST con el parámetro *fill* definido en AP_LL, este valor indica que el TP local ha recibido un registro lógico incompleto.

AP_SEND
Para el TP asociado, la conversación ha entrado en estado Recibir. Para el TP local, la conversación se encuentra en estado Enviar.

Los valores siguientes sólo se devolverán si *rtn_status* estaba definido en AP_YES:

AP_DATA_CONFIRM
Es una combinación de AP_DATA y AP_CONFIRM_WHAT_RECEIVED. El TP asociado ha enviado datos y después ha emitido el verbo [MC_]CONFIRM, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío CONFIRM.

AP_DATA_COMPLETE_CONFIRM
Es una combinación de AP_DATA_COMPLETE y AP_CONFIRM_WHAT_RECEIVED. El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha emitido el verbo [MC_]CONFIRM, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío CONFIRM.

AP_DATA_CONFIRM_DEALLOCATE
Es una combinación de AP_DATA y AP_CONFIRM_DEALLOCATE. El TP asociado ha enviado datos y después ha emitido el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío DEALLOC_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_COMPLETE_CONFIRM_DEALL
Es una combinación de AP_DATA_COMPLETE y AP_CONFIRM_DEALLOCATE. El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha emitido el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío DEALLOC_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_CONFIRM_SEND
Es una combinación de AP_DATA y AP_CONFIRM_SEND. El TP asociado ha enviado datos y después ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con el parámetro *ptr_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío P_TO_R_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_COMPLETE_CONFIRM_SEND
Es una combinación de AP_DATA_COMPLETE y AP_CONFIRM_SEND. El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con el parámetro *ptr_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío P_TO_R_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_SEND
El TP asociado ha enviado datos y después ha pasado al estado Recibir. Para el TP local, la conversación se encuentra en estado Pendiente-Envío.

AP_DATA_COMPLETE_SEND
El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha pasado al estado Recibir. Para el TP local, la conversación se encuentra en estado Pendiente-Envío.

Los valores siguientes se devolverán en el verbo MC_RECEIVE_AND_POST:

AP_USER_CONTROL_DATA_INCMP
Igual que AP_DATA_INCOMPLETE, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_USER_CONTROL_DATA_COMPLETE
Igual que AP_DATA_COMPLETE, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_SEND
Igual que AP_DATA_COMPLETE_SEND, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_CONFIRM
Igual que AP_DATA_COMPLETE_CONFIRM, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_CNFM_DEALL
Igual que AP_DATA_COMPLETE_CONFIRM_DEALL, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_CNFM_SEND
Igual que AP_DATA_COMPLETE_CONFIRM_SEND, salvo que los datos recibidos tenían el formato de datos de control de usuario.

Los valores siguientes se devolverán en el verbo MC_RECEIVE_AND_POST con *sync_level* definido en AP_SYNCPT:

AP_PS_HEADER_INCOMPLETE
Igual que AP_DATA_INCOMPLETE, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HEADER_COMPLETE
Igual que AP_DATA_COMPLETE, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_SEND
Igual que AP_DATA_COMPLETE_SEND, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_CONFIRM
Igual que AP_DATA_COMPLETE_CONFIRM, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_CNFM_DEALL
Igual que AP_DATA_COMPLETE_CONFIRM_DEALL, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_CNFM_SEND
Igual que AP_DATA_COMPLETE_CONFIRM_SEND, salvo que los datos recibidos tenían el formato de cabecera PS.

rts_rcvd
Indicador de petición de envío recibida. Este parámetro sólo es válido para las conversaciones semidúplex; no se utiliza en las conversaciones dúplex.

Los valores posibles son:

AP_YES El TP asociado ha emitido el verbo [MC_]REQUEST_TO_SEND, que solicita que el TP local cambie la conversación al estado Recibir.

AP_NO El TP asociado no ha emitido el verbo [MC_]REQUEST_TO_SEND.

Para ver una explicación de por qué este indicador puede ser recibido por verbos de recepción, consulte "MC_REQUEST_TO_SEND y REQUEST_TO_SEND" en la página 203.

expd_rcvd
Indicador de datos acelerados.

Los valores posibles son:

AP_YES El TP asociado ha enviado datos acelerados que el TP local aún no ha recibido. Para recibir estos datos, el TP local puede utilizar el verbo [MC_]RECEIVE_EXPEDITED_DATA.

Este indicador se puede establecer en diversos verbos APPC. Seguirá establecido en los verbos posteriores hasta que el TP local emita el verbo [MC_]RECEIVE_EXPEDITED_DATA para recibir los datos.

AP_NO No hay datos acelerados a la espera de ser recibidos.

dlen Número de bytes de datos recibidos (los datos se guardan en el almacenamiento intermedio especificado por el parámetro *dptr*). La longitud 0 (cero) indica que no se han recibido datos. Este parámetro sólo se utiliza si el parámetro *what_rcvd* indica que se han recibido datos.

Conversación desasignada: Si el TP asociado ha desasignado la conversación sin solicitar confirmación, APPC devuelve los parámetros siguientes:

primary_rc

AP_DEALLOC_NORMAL
El TP asociado ha emitido el verbo [MC_]DEALLOCATE con *dealloc_type* establecido en uno de los siguientes valores:

- AP_FLUSH
- AP_SYNC_LEVEL con el nivel de sincronización de la conversación especificado como AP_NONE.

dlen Número de bytes de datos recibidos (los datos se guardan en el almacenamiento intermedio especificado por el parámetro *dptr*). La longitud 0 (cero) indica que no se han recibido datos. Este parámetro sólo se utiliza si *rtn_status* está definido en AP_YES.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_RETURN_STATUS_WITH_DATA
El parámetro *rtn_status* está definido en un valor que no es válido.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

AP_INVALID_CALLBACK_HANDLE
El parámetro *callback* se ha definido en un puntero nulo y el verbo se ha emitido utilizando el punto de entrada síncrono (o utilizando el punto de entrada asíncrono con un puntero nulo a una rutina callback). Para ver más información, consulte “Notas de uso” en la página 170.

AP_RCV_AND_POST_BAD_FILL
Este código de retorno sólo se aplica al verbo RECEIVE_AND_POST de conversación básica. El parámetro *fill* está definido en un valor que no es válido.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc
Los valores posibles son:

AP_RCV_AND_POST_BAD_STATE
La conversación no se encontraba en estado Recibir, Enviar o Enviar-Pendiente cuando el TP ha emitido este verbo.

AP_RCV_AND_POST_NOT_LL_BDY
Este código de retorno sólo se aplica al verbo RECEIVE_AND_POST de conversación básica. La conversación se encontraba en estado Enviar; el TP ha empezado pero no ha terminado de enviar un registro lógico.

Verbo cancelado: Este código de retorno no puede devolverse como el código de retorno inicial, sino sólo como el código de retorno posterior si el código de retorno inicial es AP_OK.

Si el verbo no se ha ejecutado porque ha sido cancelado por otro verbo emitido por el TP, APPC devuelve el siguiente parámetro:

primary_rc

AP_CANCELLED
El TP local ha emitido uno de los verbos siguientes mientras se encontraba en estado Pendiente-Envío:

- DEALLOCATE con *dealloc_type* definido en AP_ABEND_PROG, AP_ABEND_SVC o AP_ABEND_TIMER
- MC_DEALLOCATE con *dealloc_type* definido en AP_ABEND
- [MC_]SEND_ERROR
- TP_ENDED

La emisión de uno de estos verbos mientras se encuentra en estado Pendiente-Envío hace que el verbo [MC_]RECEIVE_RTS_AND_POST se cancele. No se llama a la rutina callback. El TP local ya no recibe datos de manera asíncrona del TP asociado.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

El verbo MC_RECEIVE_AND_POST devuelve el código de retorno primario siguiente:

primary_rc
AP_DEALLOC_ABEND

APPC no devuelve un código de retorno secundario con este código de retorno primario.

El verbo RECEIVE_AND_POST devuelve los códigos de retorno primarios siguientes:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

El TP puede emitir [MC_]RECEIVE_AND_POST cuando la conversación se encuentra en estado Recibir, Enviar o Enviar-Pendiente.

Emisión del verbo en estado Enviar

La emisión del verbo [MC_]RECEIVE_AND_POST mientras la conversación se encuentra en estado Enviar tiene los efectos siguientes:

- La LU local envía la información de su almacenamiento intermedio de envío y un indicador SEND al TP asociado.
- La conversación cambia al estado Pendiente-Envío; el TP local está listo para recibir información del TP asociado de manera asíncrona.

Cambio de estado

La conversación cambia de estado dos veces. Tras el retorno inicial del verbo, si el parámetro *primary_rc* tiene el valor AP_OK, la conversación cambia al estado Pendiente-Envío. Después de que APPC llame la rutina callback o borre el semáforo para indicar la finalización del verbo, se produce un cambio de estado tal como se describe en este apartado.

El cambio de estado al finalizar el verbo [MC_]RECEIVE_AND_POST depende del valor de los siguientes parámetros:

- El parámetro *primary_rc*.
- El parámetro *what_rcvd* si *primary_rc* tiene el valor AP_OK.

En la tabla de a continuación se resumen los posibles cambios de estado que pueden producirse cuando *primary_rc* tiene el valor AP_OK:

Parámetro *what_rcvd*	**Estado nuevo**
AP_CONFIRM_WHAT_RECEIVED AP_DATA_CONFIRM AP_DATA_COMPLETE_CONFIRM	Confirmar
AP_CONFIRM_DEALLOCATE AP_DATA_CONFIRM_DEALLOCATE AP_DATA_COMPLETE_CONFIRM_DEALL	Confirmar-Desasignar
AP_CONFIRM_SEND AP_DATA_CONFIRM_SEND AP_DATA_COMPLETE_CONFIRM_SEND	Confirmar-Enviar
AP_DATA AP_DATA_COMPLETE AP_DATA_INCOMPLETE	Recibir
AP_SEND	Enviar
AP_DATA_SEND AP_DATA_COMPLETE_SEND	Enviar-Pendiente

En la tabla de a continuación se resumen los posibles cambios de estado que pueden producirse cuando *primary_rc* no tiene el valor AP_OK:

primary_rc	**Estado nuevo**
AP_PARAMETER_CHECK AP_STATE_CHECK AP_CONVERSATION_TYPE_MIXED	Sin cambio (estos códigos de retorno sólo pueden producirse como el primer código de retorno y no como el segundo código de retorno)
AP_INVALID_VERB AP_INVALID_VERB_SEGMENT	Sin cambio
AP_STACK_TOO_SMALL AP_TP_BUSY AP_UNEXPECTED_DOS_ERROR	Sin cambio
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	Restablecer
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	Recibir
AP_CANCELLED	La conversación vuelve al estado Enviar o Recibir en que se ha emitido el verbo [MC_]RECEIVE_AND_POST. Como el código de retorno AP_CANCELLED se produce como consecuencia de otro verbo emitido por el mismo TP, el estado de conversación volverá a cambiar cuando este último verbo finalice.

Notas de uso

Este apartado contiene información de uso adicional sobre los siguientes temas:

- Datos de cabecera PS
- Rutina callback
- Proceso mientras el verbo está pendiente
- Compatibilidad con otras implementaciones de APPC
- Cómo utiliza el verbo el TP
- Cómo evitar esperas indefinidas

Datos de cabecera PS

En una conversación con el nivel de sincronización AP_SYNCPT, los datos recibidos pueden tener el formato de cabecera PS. En una conversación correlacionada, esto se indica mediante el valor del parámetro *what_rcvd*; en una conversación básica, esto se indica mediante un campo LL con el valor 0x0001 (consulte "Registros lógicos" en la página 62 para obtener más información). El gestor de puntos de sincronización se encarga de convertir los datos a los mandatos de punto de sincronización adecuados.

Rutina callback

AIX, LINUX

La aplicación suministra un puntero a una rutina callback como uno de los parámetros del VCB. En este apartado se describe cómo Communications Server para Linux utiliza esta rutina y las funciones que debe efectuar.

La rutina callback se define de la siguiente forma:

```
void (*callback) (
                    void *          vcb,
                    unsigned char   tp_id[8],
                    AP_UINT32       conv_id
                  );
```

Communications Server para Linux llama a la rutina con los siguientes parámetros:

vcb Puntero al VCB suministrado por la aplicación, incluidos los parámetros devueltos establecidos por Communications Server para Linux.

tp_id Identificador de TP de 8 bytes del TP en que se ha emitido el verbo.

conv_id
Identificador de conversación de la conversación en que se ha emitido el verbo.

La rutina callback no tiene que utilizar todos estos parámetros. Puede ejecutar todo el proceso necesario en el VCB devuelto o simplemente puede establecer una variable para informar al programa principal de que el verbo ha finalizado.

La aplicación puede emitir más verbos APPC desde dentro de la rutina callback si es necesario. Sin embargo, éstos deben ser verbos asíncronos. Los verbos síncronos emitidos desde una rutina callback se rechazarán con los códigos de retorno AP_PARAMETER_CHECK y AP_SYNC_NOT_ALLOWED.

Si la aplicación emite el verbo [MC_]RECEIVE_AND_POST utilizando el punto de entrada APPC asíncrono, hay dos rutinas callback especificadas: una en el VCB y otra suministrada como un parámetro en el punto de entrada. En general, APPC utiliza la rutina callback especificada en el VCB y no tiene en cuenta la del punto de entrada; sin embargo, si la aplicación suministra un puntero nulo para la rutina callback en el VCB, APPC utiliza la rutina callback del punto de entrada.

Cómo continuar con otro proceso mientras el verbo está pendiente

Puesto que el verbo [MC_]RECEIVE_AND_POST vuelve inmediatamente sin esperar a que lleguen datos, el TP puede continuar con otro proceso mientras espera a que aquél finalice. Sin embargo, deben tenerse en cuenta los siguientes puntos:

- El VCB suministrado al verbo [MC_]RECEIVE_AND_POST continúa utilizándose hasta que la rutina callback vuelve. El TP no debe cambiar ningún campo en el VCB durante este tiempo. Si emite algún otro verbo APPC mientras se encuentra en estado Pendiente-Envío, deberá utilizar otro VCB.
- Sólo puede haber un verbo [MC_]RECEIVE_AND_POST activo por conversación en todo momento.

Compatibilidad con otras implementaciones de APPC

AIX, LINUX

La implementación de AIX o Linux del verbo [MC_]RECEIVE_AND_POST difiere de la implementación de APPC de Windows. Además, este verbo no está disponible en ninguna implementación de APPC de DOS. Por ello, los TP que utilizan [MC_]RECEIVE_AND_POST no son totalmente portables a otros sistemas operativos; si su TP utiliza este verbo, deberá volver a escribir las secciones del TP que lo utilizan si quiere que el TP se ejecute en otros sistemas operativos.

Cómo utiliza el verbo el TP

Para utilizar el verbo [MC_]RECEIVE_AND_POST, el TP local ejecuta los siguientes pasos:

1. Emite el verbo [MC_]RECEIVE_AND_POST.
2. Comprueba el valor del código de retorno primario *primary_rc*.

   Si el código de retorno primario tiene el valor AP_OK, el almacenamiento intermedio de recepción (al que apunta el parámetro *dptr*) recibe datos de manera asíncrona del TP asociado. Mientras recibe datos de manera asíncrona, el TP local puede realizar las siguientes acciones:
   - Realizar tareas no relacionadas con esta conversación.
   - Emitir el verbo [MC_]REQUEST_TO_SEND.
   - Recopilar información sobre esta conversación emitiendo los verbos siguientes:
     - GET_TYPE
     - [MC_]GET_ATTRIBUTES
     - [MC_]TEST_RTS
   - Cancelar prematuramente el verbo [MC_]RECEIVE_AND_POST emitiendo uno de los verbos siguientes:
     - DEALLOCATE con *dealloc_type* definido en AP_ABEND_PROG, AP_ABEND_SVC o AP_ABEND_TIMER
     - MC_DEALLOCATE con *dealloc_type* definido en AP_ABEND
     - SEND_ERROR
     - TP_ENDED
3. Comprueba que APPC haya llamado a la rutina callback (suministrada como un parámetro en este verbo). Cuando el TP termina de recibir datos de manera asíncrona, APPC llama esta rutina.
4. Comprueba el nuevo valor del código de retorno primario *primary_rc*.

   Si el código de retorno primario es AP_OK, el TP local puede examinar los otros parámetros de retorno y manipular los datos recibidos de manera asíncrona.

   Si el código de retorno primario no es AP_OK, sólo los parámetros *secondary_rc* y *rts_rcvd* (petición de envío recibida) son significativos.

Cómo evitar esperas indefinidas

AIX, LINUX

Si el TP local emite el verbo [MC_]RECEIVE_AND_POST y posteriormente espera a que se llame a la rutina callback, se suspenderá hasta que se reciba información del TP asociado. Puede esperar indefinidamente si el TP asociado no envía ninguna información o no emite ningún verbo que haga que la LU asociada vacíe

su almacenamiento intermedio de envío. Si necesita que el TP esté operativo de forma continua, evite esperar en la rutina callback o utilice el verbo [MC_]RECEIVE_IMMEDIATE.

MC_RECEIVE_AND_WAIT y RECEIVE_AND_WAIT

El verbo MC_RECEIVE_AND_WAIT o RECEIVE_AND_WAIT recibe los datos que están disponibles actualmente del TP asociado. Si no hay datos actualmente disponibles, el TP local espera a que lleguen.

Mientras un verbo [MC_]RECEIVE_AND_WAIT asíncrono está pendiente, la aplicación puede emitir los verbos siguientes en la misma conversación:

- GET_TYPE
- [MC_]DEALLOCATE con el tipo de desasignación AP_ABEND, AP_ABEND_PROG, AP_ABEND_SVC o AP_ABEND_TIMER
- [MC_]GET_ATTRIBUTES
- Verbos [MC_]RECEIVE adicionales, siempre que se emitan en modalidad de no bloqueo
- [MC_]RECEIVE_EXPEDITED_DATA
- [MC_]REQUEST_TO_SEND
- [MC_]SEND_DATA (sólo conversaciones dúplex)
- [MC_]SEND_EXPEDITED_DATA
- [MC_]SEND_ERROR
- [MC_]TEST_RTS
- TP_ENDED

Estructura del VCB: MC_RECEIVE_AND_WAIT

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_RECEIVE_AND_WAIT es la siguiente:

```
typedef struct mc_receive_and_wait
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado           */
  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;
```

Estructura del VCB: RECEIVE_AND_WAIT

La definición de la estructura del VCB para el verbo RECEIVE_AND_WAIT es la siguiente:

```
typedef struct receive_and_wait
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: MC_RECEIVE_AND_WAIT (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_RECEIVE_AND_WAIT es la siguiente:

```
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;
```

Estructura del VCB: RECEIVE_AND_WAIT (Windows)

La definición de la estructura del VCB para el verbo RECEIVE_AND_WAIT es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_RECEIVE_AND_WAIT
Para el verbo MC_RECEIVE_AND_WAIT.

AP_B_RECEIVE_AND_WAIT
Para el verbo RECEIVE_AND_WAIT.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_RECEIVE_AND_WAIT.

AP_BASIC_CONVERSATION
Para el verbo RECEIVE_AND_WAIT.

Si el verbo se está utilizando en una conversación dúplex o se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con uno de los siguientes valores, o con ambos:

AP_FULL_DUPLEX_CONVERSATION
El verbo se está emitiendo en una conversación dúplex.

AP_NON_BLOCKING
El verbo se está emitiendo como un verbo de no bloqueo.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

rtn_status
Indica si la información de estado y los datos pueden devolverse en el mismo verbo. Los valores posibles son:

AP_YES La información de estado, si está disponible, se devuelve con la última parte de un registro de datos.

AP_NO La información de estado no se devuelve con los datos. Después

de recibir el fin de un registro de datos, el TP local debe emitir otro verbo [MC_]RECEIVE para obtener la información de estado.

fill Indica la manera en que el TP local recibe datos.

Este parámetro sólo es utilizado por el verbo RECEIVE_AND_WAIT de conversación básica. Los valores posibles son:

AP_BUFFER
El TP local recibe datos hasta que se alcanza el número de bytes especificado por el parámetro *max_len* o hasta el fin de los datos. Los datos se reciben sin tener en cuenta el formato de registro lógico.

AP_LL Los datos se reciben con el formato de registro lógico. Los datos recibidos pueden ser:

- Un registro lógico completo.
- Una parte de un registro lógico (el número de bytes especificado por *max_len*).
- El fin de un registro lógico.

max_len
Número máximo de bytes de datos que puede recibir el TP local.

El rango de este valor es `0–65.535`.

Este valor no debe sobrepasar la longitud del almacenamiento intermedio que contendrá los datos recibidos.

dptr Dirección del almacenamiento intermedio que contendrá los datos recibidos por el TP local.

WINDOWS

El almacenamiento intermedio de datos puede residir en un área de datos estática o en un área asignada globalmente. El almacenamiento intermedio de datos debe caber completamente en esta área.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
`AP_OK`

what_rcvd
Información de estado recibida con los datos entrantes.

La siguiente acción que emprende el TP normalmente dependerá del valor de este parámetro. Para ver más información, consulte “Parámetro what_rcvd” en la página 156.

Los valores posibles son:

AP_CONFIRM_DEALLOCATE
Este valor sólo se puede devolver en una conversación semidúplex.

El TP asociado ha emitido el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* establecido en AP_SYNC_LEVEL y el nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_CONFIRM_SEND
Este valor sólo se puede devolver en una conversación semidúplex.

El TP asociado ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con el parámetro *ptr_type* establecido en AP_SYNC_LEVEL y el nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_CONFIRM_WHAT_RECEIVED
Este valor sólo se puede devolver en una conversación semidúplex.

El TP asociado ha emitido el verbo [MC_]CONFIRM.

AP_DATA
Este valor puede ser devuelto por el verbo RECEIVE_AND_WAIT de conversación básica si el parámetro *fill* está definido en AP_BUFFER; no se aplica a MC_RECEIVE_AND_WAIT.

El TP local ha recibido datos hasta alcanzar *max_len* o el fin de los datos.

AP_DATA_COMPLETE
Para el verbo MC_RECEIVE_AND_WAIT de conversación correlacionada, este valor indica que el TP local ha recibido un registro de datos completo o la última parte de un registro de datos.

Para el verbo RECEIVE_AND_WAIT de conversación básica con el parámetro *fill* definido en AP_LL, este valor indica que el TP local ha recibido un registro lógico completo o el fin de un registro lógico.

AP_DATA_INCOMPLETE
Para el verbo MC_RECEIVE_AND_WAIT de conversación correlacionada, este valor indica que el TP local ha recibido un registro de datos incompleto. El parámetro *max_len* ha especificado un valor inferior a la longitud del registro de datos (o inferior al registro de datos restante si no es el primer verbo de recepción que lee el registro).

Para el verbo RECEIVE_AND_WAIT de conversación básica con el parámetro *fill* definido en AP_LL, este valor indica que el TP local ha recibido un registro lógico incompleto.

AP_SEND
Este valor sólo se puede devolver en una conversación semidúplex.

Para el TP asociado, la conversación ha entrado en estado Recibir. Para el TP local, la conversación se encuentra en estado Enviar.

Los valores siguientes sólo se devolverán en una conversación semidúplex y sólo si *rtn_status* se ha definido en AP_YES:

AP_DATA_CONFIRM
Es una combinación de AP_DATA y AP_CONFIRM_WHAT_RECEIVED. El TP

asociado ha enviado datos y después ha emitido el verbo [MC_]CONFIRM, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío CONFIRM.

AP_DATA_COMPLETE_CONFIRM
Es una combinación de AP_DATA_COMPLETE y AP_CONFIRM_WHAT_RECEIVED. El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha emitido el verbo [MC_]CONFIRM, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío CONFIRM.

AP_DATA_CONFIRM_DEALLOCATE
Es una combinación de AP_DATA y AP_CONFIRM_DEALLOCATE. El TP asociado ha enviado datos y después ha emitido el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío DEALLOC_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_COMPLETE_CONFIRM_DEALL
Es una combinación de AP_DATA_COMPLETE y AP_CONFIRM_DEALLOCATE. El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha emitido el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío DEALLOC_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_CONFIRM_SEND
Es una combinación de AP_DATA y AP_CONFIRM_SEND. El TP asociado ha enviado datos y después ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con el parámetro *ptr_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío P_TO_R_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_COMPLETE_CONFIRM_SEND
Es una combinación de AP_DATA_COMPLETE y AP_CONFIRM_SEND. El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con el parámetro *ptr_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío P_TO_R_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_SEND
El TP asociado ha enviado datos y después ha pasado al estado Recibir. Para el TP local, la conversación se encuentra en estado Pendiente-Envío.

AP_DATA_COMPLETE_SEND
El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha pasado al estado Recibir. Para el TP local, la conversación se encuentra en estado Pendiente-Envío.

AIX, LINUX

Los valores siguientes se devolverán en el verbo MC_RECEIVE_AND_WAIT:

AP_USER_CONTROL_DATA_INCMP
Igual que AP_DATA_INCOMPLETE, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_USER_CONTROL_DATA_COMPLETE
Igual que AP_DATA_COMPLETE, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_SEND
Igual que AP_DATA_COMPLETE_SEND, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_CONFIRM
Igual que AP_DATA_COMPLETE_CONFIRM, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_CNFM_DEALL
Igual que AP_DATA_COMPLETE_CONFIRM_DEALL, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_CNFM_SEND
Igual que AP_DATA_COMPLETE_CONFIRM_SEND, salvo que los datos recibidos tenían el formato de datos de control de usuario.

Los valores siguientes se devolverán en el verbo MC_RECEIVE_AND_WAIT con *sync_level* definido en AP_SYNCPT:

AP_PS_HEADER_INCOMPLETE
Igual que AP_DATA_INCOMPLETE, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HEADER_COMPLETE
Igual que AP_DATA_COMPLETE, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_SEND
Igual que AP_DATA_COMPLETE_SEND, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_CONFIRM
Igual que AP_DATA_COMPLETE_CONFIRM, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_CNFM_DEALL
Igual que AP_DATA_COMPLETE_CONFIRM_DEALL, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_CNFM_SEND
Igual que AP_DATA_COMPLETE_CONFIRM_SEND, salvo que los datos recibidos tenían el formato de cabecera PS.

rts_rcvd
Indicador de petición de envío recibida. Este parámetro sólo es válido para las conversaciones semidúplex; no se utiliza en las conversaciones dúplex.

Los valores posibles son:

AP_YES El TP asociado ha emitido el verbo [MC_]REQUEST_TO_SEND, que solicita que el TP local cambie la conversación al estado Recibir.

AP_NO El TP asociado no ha emitido el verbo [MC_]REQUEST_TO_SEND.

Para ver una explicación de por qué este indicador puede ser recibido por verbos RECEIVE, consulte "MC_REQUEST_TO_SEND y REQUEST_TO_SEND" en la página 203.

expd_rcvd
Indicador de datos acelerados.

Los valores posibles son:

AP_YES El TP asociado ha enviado datos acelerados que el TP local aún no ha recibido. Para recibir estos datos, el TP local puede utilizar el verbo [MC_]RECEIVE_EXPEDITED_DATA.

Este indicador se puede establecer en diversos verbos APPC. Seguirá establecido en los verbos posteriores hasta que el TP local emita el verbo [MC_]RECEIVE_EXPEDITED_DATA para recibir los datos.

AP_NO No hay datos acelerados a la espera de ser recibidos.

dlen Este parámetro sólo se utiliza si el parámetro *what_rcvd* indica que se han recibido datos.

Número de bytes de datos recibidos (los datos se guardan en el almacenamiento intermedio especificado por el parámetro *dptr*). La longitud `0` (cero) indica que no se han recibido datos.

Conversación desasignada: Si el TP asociado ha desasignado la conversación sin solicitar confirmación, APPC devuelve los parámetros siguientes:

primary_rc

AP_DEALLOC_NORMAL
El TP asociado ha emitido el verbo [MC_]DEALLOCATE con *dealloc_type* establecido en uno de los siguientes valores:

- `AP_FLUSH`
- `AP_SYNC_LEVEL` con el nivel de sincronización de la conversación especificado como `AP_NONE`.

dlen Número de bytes de datos recibidos (los datos se guardan en el almacenamiento intermedio especificado por el parámetro *dptr*). La longitud `0` (cero) indica que no se han recibido datos. Este parámetro sólo se utiliza si *rtn_status* está definido en `AP_YES`.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
`AP_PARAMETER_CHECK`

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_RETURN_STATUS_WITH_DATA
El parámetro *rtn_status* está definido en un valor que no es válido.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AIX, LINUX

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

WINDOWS

AP_INVALID_DATA_SEGMENT
Los datos son más largos que el segmento de datos asignado o la dirección del almacenamiento intermedio de datos es incorrecta.

AP_RCV_AND_WAIT_BAD_FILL
Este código de retorno sólo se aplica al verbo RECEIVE_AND_WAIT de conversación básica. El parámetro *fill* está definido en un valor que no es válido.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc
Los valores posibles son:

AP_RCV_AND_WAIT_BAD_STATE
La conversación no se encontraba en estado Recibir, Enviar o Enviar-Pendiente cuando el TP ha emitido este verbo.

AP_RCV_AND_WAIT_NOT_LL_BDY
Este código de retorno sólo se aplica al verbo RECEIVE_AND_WAIT de conversación básica. La conversación se encontraba en estado Enviar; el TP ha empezado pero no ha terminado de enviar un registro lógico.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde,

códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

El verbo MC_RECEIVE_AND_WAIT devuelve el siguiente código de retorno primario:

primary_rc

```
AP_DEALLOC_ABEND
```

APPC no devuelve un código de retorno secundario con este código de retorno primario.

El verbo RECEIVE_AND_WAIT devuelve los siguientes códigos de retorno primarios:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

El TP puede emitir el verbo [MC_]RECEIVE_AND_WAIT cuando la conversación se encuentra en estado Enviar-Recibir (sólo en las conversaciones dúplex) Recibir, Enviar o Enviar-Pendiente.

Emisión del verbo en estado Enviar (sólo conversaciones semidúplex)

La emisión del verbo [MC_]RECEIVE_AND_WAIT mientras la conversación se encuentra en estado Enviar tiene los efectos siguientes:

- La LU local envía la información de su almacenamiento intermedio de envío y un indicador SEND al TP asociado.
- La conversación cambia al estado Recibir; el TP local espera recibir información del TP asociado.

Cambio de estado

WINDOWS

Cuando el verbo se emite al punto de entrada asíncrono, la conversación cambia de estado dos veces. Tras el retorno inicial del verbo, si el parámetro *primary_rc* tiene el valor AP_OK, la conversación cambia al estado Enviar-Pendiente. Después de que APPC indique la finalización del verbo, el estado cambia tal como se describe a continuación. Si desea obtener más información sobre las acciones que la aplicación puede llevar a cabo en estado Enviar-Pendiente, consulte el apartado "MC_RECEIVE_AND_POST y RECEIVE_AND_POST" en la página 158.

El cambio de estado después del verbo [MC_]RECEIVE_AND_WAIT depende del valor de los siguientes parámetros:

- El parámetro *primary_rc*.
- El parámetro *what_rcvd*.

Los cambios de estado posibles están resumidos en las tablas siguientes.

Parámetro *what_rcvd*	**Estado nuevo**
AP_CONFIRM_WHAT_RECEIVED AP_DATA_CONFIRM AP_DATA_COMPLETE_CONFIRM	Confirmar
AP_CONFIRM_DEALLOCATE AP_DATA_CONFIRM_DEALLOCATE AP_DATA_COMPLETE_CONFIRM_DEALL	Confirmar-Desasignar
AP_CONFIRM_SEND AP_DATA_CONFIRM_SEND AP_DATA_COMPLETE_CONFIRM_SEND	Confirmar-Enviar
AP_DATA AP_DATA_COMPLETE AP_DATA_INCOMPLETE	Recibir (conversación semidúplex) o sin cambios (conversación dúplex)
AP_SEND	Enviar
AP_DATA_SEND AP_DATA_COMPLETE_SEND	Enviar-Pendiente

primary_rc	**Estado nuevo**
AP_PARAMETER_CHECK 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	Sin cambio
AP_CONV_FAILURE_RETRY AP_CONV_FAILURE_NO_RETRY	Restablecer
AP_DEALLOC_ABEND AP_DEALLOC_ABEND_PROG AP_DEALLOC_ABEND_SVC AP_DEALLOC_ABEND_TIMER	Restablecer
AP_DEALLOC_NORMAL	Restablecer

primary_rc	**Estado nuevo**
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	Recibir (conversación semidúplex) o sin cambios (conversación dúplex)

Notas de uso

Este apartado contiene información de uso adicional sobre los siguientes temas:

- Datos de cabecera PS.
- Cómo evitar esperas indefinidas

Datos de cabecera PS

AIX, LINUX

En una conversación con el nivel de sincronización AP_SYNCPT, los datos recibidos pueden tener el formato de cabecera PS. En una conversación correlacionada, esto se indica mediante el valor del parámetro *what_rcvd*; en una conversación básica, esto se indica mediante un campo LL con el valor 0x0001 (para ver más información, consulte "Registros lógicos" en la página 62). El gestor de puntos de sincronización se encarga de convertir los datos a los mandatos de punto de sincronización adecuados.

Cómo evitar esperas indefinidas

Si el TP local emite el verbo [MC_]RECEIVE_AND_WAIT, se suspenderá hasta que se reciba información del TP asociado. Puede esperar indefinidamente si el TP asociado no envía ninguna información o no emite ningún verbo que haga que la LU asociada vacíe su almacenamiento intermedio de envío. Si necesita que el TP esté operativo de forma continua, utilice el verbo [MC_]RECEIVE_AND_POST pero evite esperar en la rutina callback, o utilice el verbo [MC_]RECEIVE_IMMEDIATE.

MC_RECEIVE_IMMEDIATE y RECEIVE_IMMEDIATE

El verbo MC_RECEIVE_IMMEDIATE o RECEIVE_IMMEDIATE recibe los datos y/o la información de estado que están disponibles actualmente del TP asociado. Si no hay nada disponible, el TP local vuelve inmediatamente y no espera.

WINDOWS

Aunque este verbo no espera a recibir información, aún es posible que la biblioteca APPC de Windows ceda para permitir que otros procesos continúen. No se debe presuponer que el verbo volverá sin ceder.

Estructura del VCB: MC_RECEIVE_IMMEDIATE

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_RECEIVE_IMMEDIATE es la siguiente:

```
typedef struct mc_receive_immediate
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: RECEIVE_IMMEDIATE

La definición de la estructura del VCB para el verbo RECEIVE_IMMEDIATE es la siguiente:

```
typedef struct receive_immediate
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: MC_RECEIVE_IMMEDIATE (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_RECEIVE_IMMEDIATE es la siguiente:

```
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;
```

Estructura del VCB: RECEIVE_IMMEDIATE (Windows)

La definición de la estructura del VCB para el verbo RECEIVE_IMMEDIATE es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_RECEIVE_IMMEDIATE
Para el verbo MC_RECEIVE_IMMEDIATE.

AP_B_RECEIVE_IMMEDIATE
Para el verbo RECEIVE_IMMEDIATE.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_RECEIVE_IMMEDIATE.

AP_BASIC_CONVERSATION
Para el verbo RECEIVE_IMMEDIATE.

Si el verbo se está utilizando en una conversación dúplex o se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con uno de los siguientes valores, o con ambos:

AP_FULL_DUPLEX_CONVERSATION
El verbo se está emitiendo en una conversación dúplex.

AP_NON_BLOCKING
El verbo se está emitiendo como un verbo de no bloqueo.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

rtn_status
Indica si la información de estado y los datos pueden devolverse en el mismo verbo. Los valores posibles son:

AP_YES La información de estado, si está disponible, se devuelve con la última parte de un registro de datos.

AP_NO La información de estado no se devuelve con los datos. Después de recibir el fin de un registro de datos, el TP local debe emitir otro verbo [MC_]RECEIVE para obtener la información de estado.

fill Indica la manera en que el TP local recibe datos.

Este parámetro sólo es utilizado por el verbo RECEIVE_IMMEDIATE de conversación básica. Los valores posibles son:

AP_BUFFER
El TP local recibe datos hasta que se alcanza el número de bytes especificado por el parámetro *max_len* o hasta el fin de los datos. Los datos se reciben sin tener en cuenta el formato de registro lógico.

AP_LL Los datos se reciben con el formato de registro lógico. Los datos recibidos pueden ser:

- Un registro lógico completo.
- Una parte de un registro lógico (el número de bytes especificado por *max_len*).
- El fin de un registro lógico.

max_len
Número máximo de bytes de datos que puede recibir el TP local.

El rango de este valor es 0–65.535.

Este valor no debe sobrepasar la longitud del almacenamiento intermedio que contendrá los datos recibidos.

dptr Dirección del almacenamiento intermedio que contendrá los datos recibidos por el TP local.

WINDOWS

El almacenamiento intermedio de datos puede residir en un área de datos estática o en un área asignada globalmente. El almacenamiento intermedio de datos debe caber completamente en esta área.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

what_rcvd
Información de estado recibida con los datos entrantes.

La siguiente acción que emprende el TP normalmente dependerá del valor de este parámetro. Para ver más información, consulte "Parámetro what_rcvd" en la página 156.

Los valores posibles son:

AP_CONFIRM_DEALLOCATE
El TP asociado ha emitido el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* establecido en AP_SYNC_LEVEL y el nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_CONFIRM_SEND
El TP asociado ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con el parámetro *ptr_type* establecido en AP_SYNC_LEVEL y el nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_CONFIRM_WHAT_RECEIVED
El TP asociado ha emitido el verbo [MC_]CONFIRM.

AP_DATA
Este valor puede ser devuelto por el verbo RECEIVE_IMMEDIATE de conversación básica si el parámetro *fill* está definido en AP_BUFFER; no se aplica a MC_RECEIVE_IMMEDIATE.

El TP local ha recibido datos hasta alcanzar *max_len* o el fin de los datos.

AP_DATA_COMPLETE
Para el verbo MC_RECEIVE_IMMEDIATE de conversación correlacionada, este valor indica que el TP local ha recibido un registro de datos completo o la última parte de un registro de datos.

Para el verbo RECEIVE_IMMEDIATE de conversación básica con el parámetro *fill* definido en AP_LL, este valor indica que el TP local ha recibido un registro lógico completo o el fin de un registro lógico.

AP_DATA_INCOMPLETE
Para el verbo MC_RECEIVE_IMMEDIATE de conversación correlacionada, este valor indica que el TP local ha recibido un registro de datos incompleto. El parámetro *max_len* ha especificado un valor inferior a la longitud del registro de datos (o inferior al registro de datos restante si no es el primer verbo de recepción que lee el registro).

Para el verbo RECEIVE_IMMEDIATE de conversación básica con el parámetro *fill* definido en AP_LL, este valor indica que el TP local ha recibido un registro lógico incompleto.

AP_SEND
Para el TP asociado, la conversación ha entrado en estado Recibir. Para el TP local, la conversación se encuentra en estado Enviar.

Los valores siguientes sólo se devolverán si *rtn_status* estaba definido en AP_YES:

AP_DATA_CONFIRM
Es una combinación de AP_DATA y AP_CONFIRM_WHAT_RECEIVED. El TP asociado ha enviado datos y después ha emitido el verbo [MC_]CONFIRM, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío CONFIRM.

AP_DATA_COMPLETE_CONFIRM
Es una combinación de AP_DATA_COMPLETE y AP_CONFIRM_WHAT_RECEIVED. El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha emitido el verbo [MC_]CONFIRM, o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío CONFIRM.

AP_DATA_CONFIRM_DEALLOCATE
Es una combinación de AP_DATA y AP_CONFIRM_DEALLOCATE. El TP asociado ha enviado datos y después ha emitido el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío DEALLOC_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_COMPLETE_CONFIRM_DEALL
Es una combinación de AP_DATA_COMPLETE y AP_CONFIRM_DEALLOCATE. El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha emitido el verbo [MC_]DEALLOCATE con el parámetro *dealloc_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío DEALLOC_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_CONFIRM_SEND
Es una combinación de AP_DATA y AP_CONFIRM_SEND. El TP asociado ha enviado datos y después ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con el parámetro *ptr_type* definido

en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío P_TO_R_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_COMPLETE_CONFIRM_SEND
Es una combinación de AP_DATA_COMPLETE y AP_CONFIRM_SEND. El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha emitido el verbo [MC_]PREPARE_TO_RECEIVE con el parámetro *ptr_type* definido en AP_SYNC_LEVEL o ha emitido el verbo [MC_]SEND_DATA con el tipo de envío P_TO_R_SYNC_LEVEL. El nivel de sincronización de la conversación, establecido por el verbo [MC_]ALLOCATE, es AP_CONFIRM_SYNC_LEVEL.

AP_DATA_SEND
El TP asociado ha enviado datos y después ha pasado al estado Recibir. Para el TP local, la conversación se encuentra en estado Pendiente-Envío.

AP_DATA_COMPLETE_SEND
El TP asociado ha enviado un registro de datos completo (o el fin de un registro de datos) y después ha pasado al estado Recibir. Para el TP local, la conversación se encuentra en estado Pendiente-Envío.

AIX, LINUX

Los valores siguientes se devolverán en el verbo MC_RECEIVE_IMMEDIATE:

AP_USER_CONTROL_DATA_INCMP
Igual que AP_DATA_INCOMPLETE, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_USER_CONTROL_DATA_COMPLETE
Igual que AP_DATA_COMPLETE, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_SEND
Igual que AP_DATA_COMPLETE_SEND, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_CONFIRM
Igual que AP_DATA_COMPLETE_CONFIRM, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_CNFM_DEALL
Igual que AP_DATA_COMPLETE_CONFIRM_DEALL, salvo que los datos recibidos tenían el formato de datos de control de usuario.

AP_UC_DATA_COMPLETE_CNFM_SEND
Igual que AP_DATA_COMPLETE_CONFIRM_SEND, salvo que los datos recibidos tenían el formato de datos de control de usuario.

Los valores siguientes se devolverán en el verbo MC_RECEIVE_IMMEDIATE con *sync_level* definido en AP_SYNCPT:

AP_PS_HEADER_INCOMPLETE
Igual que AP_DATA_INCOMPLETE, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HEADER_COMPLETE
Igual que AP_DATA_COMPLETE, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_SEND
Igual que AP_DATA_COMPLETE_SEND, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_CONFIRM
Igual que AP_DATA_COMPLETE_CONFIRM, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_CNFM_DEALL
Igual que AP_DATA_COMPLETE_CONFIRM_DEALL, salvo que los datos recibidos tenían el formato de cabecera PS.

AP_PS_HDR_COMPLETE_CNFM_SEND
Igual que AP_DATA_COMPLETE_CONFIRM_SEND, salvo que los datos recibidos tenían el formato de cabecera PS.

rts_rcvd
Indicador de petición de envío recibida. Este parámetro sólo es válido para las conversaciones semidúplex; no se utiliza en las conversaciones dúplex.

Los valores posibles son:

AP_YES El TP asociado ha emitido el verbo [MC_]REQUEST_TO_SEND, que solicita que el TP local cambie la conversación al estado Recibir.

AP_NO El TP asociado no ha emitido el verbo [MC_]REQUEST_TO_SEND.

Para ver una explicación de por qué este indicador puede ser recibido por verbos RECEIVE, consulte "MC_REQUEST_TO_SEND y REQUEST_TO_SEND" en la página 203.

expd_rcvd
Indicador de datos acelerados.

Los valores posibles son:

AP_YES El TP asociado ha enviado datos acelerados que el TP local aún no ha recibido. Para recibir estos datos, el TP local puede utilizar el verbo [MC_]RECEIVE_EXPEDITED_DATA.

Este indicador se puede establecer en diversos verbos APPC. Seguirá establecido en los verbos posteriores hasta que el TP local emita el verbo [MC_]RECEIVE_EXPEDITED_DATA para recibir los datos.

AP_NO No hay datos acelerados a la espera de ser recibidos.

dlen Este parámetro sólo se utiliza si el parámetro *what_rcvd* indica que se han recibido datos.

Número de bytes de datos recibidos (los datos se guardan en el almacenamiento intermedio especificado por el parámetro *dptr*). La longitud 0 (cero) indica que no se han recibido datos.

Conversación desasignada: Si el TP asociado ha desasignado la conversación sin solicitar confirmación, APPC devuelve los parámetros siguientes:

primary_rc

AP_DEALLOC_NORMAL
El TP asociado ha emitido el verbo [MC_]DEALLOCATE con *dealloc_type* establecido en uno de los siguientes valores:

- AP_FLUSH
- AP_SYNC_LEVEL con el nivel de sincronización de la conversación especificado como AP_NONE.

dlen Número de bytes de datos recibidos (los datos se guardan en el almacenamiento intermedio especificado por el parámetro *dptr*). La longitud 0 (cero) indica que no se han recibido datos. Este parámetro sólo se utiliza si *rtn_status* está definido en AP_YES.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_RETURN_STATUS_WITH_DATA
El parámetro *rtn_status* está definido en un valor que no es válido.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AIX, LINUX

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

WINDOWS

AP_INVALID_DATA_SEGMENT
Los datos son más largos que el segmento de datos asignado o la dirección del almacenamiento intermedio de datos es incorrecta.

AP_RCV_IMMD_BAD_FILL
Este código de retorno sólo se aplica al verbo RECEIVE_IMMEDIATE de conversación básica. El parámetro *fill* está definido en un valor que no es válido.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc
Los valores posibles son:

AP_RCV_IMMD_BAD_STATE
La conversación no se encontraba en estado Recibir cuando el TP ha emitido este verbo.

No hay datos disponibles: Si no hay datos inmediatamente disponibles del TP asociado, APPC devuelve el parámetro siguiente:

primary_rc
AP_UNSUCCESSFUL

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

El verbo MC_RECEIVE_IMMEDIATE devuelve el siguiente código de retorno primario:

primary_rc
AP_DEALLOC_ABEND

APPC no devuelve un código de retorno secundario con este código de retorno primario.

El verbo RECEIVE_IMMEDIATE devuelve los siguientes códigos de retorno primarios:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

El TP puede emitir el verbo [MC_]RECEIVE_IMMEDIATE sólo cuando la conversación se encuentra en estado Enviar-Recibir (sólo en las conversaciones dúplex) o Recibir.

Cambio de estado

El cambio de estado después del verbo [MC_]RECEIVE_IMMEDIATE depende del valor de los siguientes parámetros:

- El parámetro *primary_rc*.
- El parámetro *what_rcvd* si *primary_rc* tiene el valor AP_OK.

Los cambios de estado posibles están resumidos en las tablas siguientes.

Parámetro *what_rcvd*	**Estado nuevo**
AP_CONFIRM_WHAT_RECEIVED AP_DATA_CONFIRM AP_DATA_COMPLETE_CONFIRM	Confirmar
AP_CONFIRM_DEALLOCATE AP_DATA_CONFIRM_DEALLOCATE AP_DATA_COMPLETE_CONFIRM_DEALL	Confirmar-Desasignar
AP_CONFIRM_SEND AP_DATA_CONFIRM_SEND AP_DATA_COMPLETE_CONFIRM_SEND	Confirmar-Enviar
AP_DATA AP_DATA_COMPLETE AP_DATA_INCOMPLETE	Recibir (conversación semidúplex) o sin cambios (conversación dúplex)
AP_SEND	Enviar
AP_DATA_SEND AP_DATA_COMPLETE_SEND	Enviar-Pendiente

primary_rc	**Estado nuevo**
AP_PARAMETER_CHECK 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	Sin cambio
AP_CONV_FAILURE_RETRY AP_CONV_FAILURE_NO_RETRY	Restablecer
AP_DEALLOC_ABEND AP_DEALLOC_ABEND_PROG AP_DEALLOC_ABEND_SVC AP_DEALLOC_ABEND_TIMER AP_DEALLOC_NORMAL	Restablecer (conversación semidúplex) o Sólo-Enviar (conversación dúplex)

primary_rc	**Estado nuevo**
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	Recibir (conversación semidúplex) o sin cambios (conversación dúplex)

Datos de cabecera PS

AIX, LINUX

En una conversación con el nivel de sincronización AP_SYNCPT, los datos recibidos pueden tener el formato de cabecera PS. En una conversación correlacionada, esto se indica mediante el valor del parámetro *what_rcvd*; en una conversación básica, esto se indica mediante un campo LL con el valor 0x0001 (consulte "Registros lógicos" en la página 62 para obtener más información). El gestor de puntos de sincronización se encarga de convertir los datos a los mandatos de punto de sincronización adecuados.

MC_RECEIVE_EXPEDITED_DATA y RECEIVE_EXPEDITED_DATA

El verbo MC_RECEIVE_EXPEDITED_DATA o RECEIVE_EXPEDITED_DATA recibe los datos acelerados que están disponibles actualmente del TP asociado. Si no hay datos disponibles actualmente, el verbo puede volver inmediatamente o esperar a que lleguen datos.

Estructura del VCB: MC_RECEIVE_EXPEDITED_DATA

La definición de la estructura del VCB para el verbo MC_RECEIVE_EXPEDITED_DATA es la siguiente:

```
typedef struct mc_receive_expedited_data
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: RECEIVE_EXPEDITED_DATA

La definición de la estructura del VCB para el verbo RECEIVE_EXPEDITED_DATA es la siguiente:

```
typedef struct receive_expedited_data
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_RECEIVE_EXPEDITED_DATA
Para el verbo MC_RECEIVE_EXPEDITED_DATA.

AP_B_RECEIVE_EXPEDITED_DATA
Para el verbo RECEIVE_EXPEDITED_DATA.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_RECEIVE_EXPEDITED_DATA.

AP_BASIC_CONVERSATION
Para el verbo RECEIVE_EXPEDITED_DATA.

Si el verbo se está utilizando en una conversación dúplex o se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con uno de los siguientes valores, o con ambos:

AP_FULL_DUPLEX_CONVERSATION
El verbo se está emitiendo en una conversación dúplex.

AP_NON_BLOCKING
El verbo se está emitiendo como un verbo de no bloqueo.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

rtn_ctl Indica que el verbo debe devolver el control al TP si no hay datos acelerados disponibles al emitirlo. Los valores posibles son:

AP_IMMEDIATE
Si no hay datos acelerados disponibles, el verbo devolverá inmediatamente un código de retorno que indica esta situación.

AP_WHEN_EXPD_RCVD
Si no hay datos acelerados disponibles, el verbo espera a que lleguen. Puede esperar indefinidamente si el TP asociado no envía datos acelerados.

max_len
Número máximo de bytes de datos que puede recibir el TP local.

El rango de este valor es `0–86`.

Este valor no debe sobrepasar la longitud del almacenamiento intermedio que contendrá los datos recibidos.

dptr Dirección del almacenamiento intermedio que contendrá los datos recibidos por el TP local.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
`AP_OK`

expd_rcvd
Indicador de datos acelerados.

Los valores posibles son:

AP_YES El TP asociado ha enviado más datos acelerados que el TP local aún no ha recibido, además de los datos devueltos en este verbo. Para recibir estos datos, el TP local puede volver a emitir el verbo [MC_]RECEIVE_EXPEDITED_DATA.

Este indicador se puede establecer en diversos verbos APPC. Seguirá establecido en los verbos posteriores hasta que el TP local emita el verbo [MC_]RECEIVE_EXPEDITED_DATA para recibir los datos.

AP_NO No hay datos acelerados a la espera de ser recibidos.

dlen Número de bytes de datos recibidos (los datos se guardan en el almacenamiento intermedio especificado por el parámetro *dptr*). La longitud `0` (cero) indica que no se han recibido datos.

Los datos recibidos no tienen formato y no contienen un campo de longitud de dos bytes (LL).

No hay datos disponibles: Si se ha establecido `AP_IMMEDIATE` como valor del parámetro *rtn_ctl* y no había datos acelerados disponibles, APPC devolverá los siguientes parámetros:

primary_rc

AP_UNSUCCESSFUL

Conversación desasignada: Si el TP asociado ha desasignado la conversación, APPC devolverá uno de los siguientes valores:

primary_rc

AP_DEALLOC_NORMAL
El TP asociado ha emitido el verbo [MC_]DEALLOCATE con *dealloc_type* establecido en uno de los siguientes valores:

- AP_FLUSH
- AP_SYNC_LEVEL con el nivel de sincronización de la conversación especificado como AP_NONE.

primary_rc

AP_CONVERSATION_ENDED
Este verbo se ha emitido como un verbo de no bloqueo y se ha puesto en cola detrás de un verbo anterior. El TP asociado ha emitido el verbo [MC_]DEALLOCATE del mismo modo que para el verbo AP_DEALLOC_NORMAL antes mencionado, y el primer verbo de la cola ha vuelto con *primary_rc* establecido en AP_DEALLOC_NORMAL, que indica el fin de la conversación. Los verbos posteriores de la cola vuelven con *primary_rc* establecido en AP_CONVERSATION_ENDED, que indica que la conversación ya había finalizado antes de que el verbo se pudiese procesar.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

No se admiten los datos acelerados: Si el verbo no se ejecuta porque la LU remota no admite datos acelerados, APPC devolverá el parámetro siguiente:

primary_rc

AP_EXPD_NOT_SUPPORTED_BY_LU

El almacenamiento intermedio de datos es demasiado pequeño: Si el verbo no se ejecuta porque el almacenamiento intermedio de datos del TP es demasiado pequeño para contener todos los datos acelerados disponibles, APPC devolverá el parámetro siguiente:

primary_rc

AP_BUFFER_TOO_SMALL

dlen Número de bytes de datos acelerados disponibles en la LU remota.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_EXPD_BAD_RETURN_CONTROL
El parámetro *rtn_ctl* está definido en un valor que no es válido.

AP_RCV_EXPD_INVALID_LENGTH
El parámetro *max_len* está definido en un valor que no es válido.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc
Los valores posibles son:

AP_EXPD_DATA_BAD_CONV_STATE
La conversación se encontraba en estado Restablecer cuando el TP ha emitido este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

El verbo MC_RECEIVE_EXPEDITED_DATA devuelve el siguiente código de retorno primario:

primary_rc

AP_DEALLOC_ABEND

APPC no devuelve un código de retorno secundario con este código de retorno primario.

El verbo RECEIVE_EXPEDITED_DATA devuelve los siguientes códigos de retorno primarios:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

El TP puede emitir el verbo [MC_]RECEIVE_EXPEDITED_DATA cuando la conversación está en cualquier estado salvo Restablecer.

Cambio de estado

El cambio de estado después del verbo [MC_]RECEIVE_EXPEDITED_DATA depende del parámetro *primary_rc*. Los cambios de estado posibles están resumidos en la tabla siguiente.

primary_rc	**Estado nuevo**
AP_OK	Sin cambio
AP_PARAMETER_CHECK	Sin cambio
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	Restablecer
AP_CONV_FAILURE_NO_RETRY	
AP_CONVERSATION_ENDED	
AP_DEALLOC_ABEND	Restablecer
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	
AP_DEALLOC_NORMAL	Restablecer

MC_REQUEST_TO_SEND y REQUEST_TO_SEND

El verbo MC_REQUEST_TO_SEND_ERROR o REQUEST_TO_SEND notifica al TP asociado que el TP local quiere enviar datos.

Nota: Este verbo sólo se puede utilizar en las conversaciones semidúplex; no es válido para las conversaciones dúplex.

Acción del TP asociado

Como respuesta a esta petición, el TP asociado puede cambiar la conversación a uno de los estados siguientes:

- Estado Recibir emitiendo el verbo [MC_]PREPARE_TO_RECEIVE o [MC_]RECEIVE_AND_WAIT.
- Estado Pendiente-Envío emitiendo el verbo [MC_]RECEIVE_AND_POST.

El TP asociado también puede hacer caso omiso de la petición de envío.

Cuándo puede enviar datos el TP local

El estado de conversación cambia a Enviar para el TP local cuando recibe uno de los valores siguientes mediante el parámetro *what_rcvd* de un verbo de recepción posterior:

- AP_CONFIRM_SEND, AP_DATA_CONFIRM_SEND o AP_DATA_COMPLETE_CONFIRM_SEND (y responde con [MC_]CONFIRMED).
- AP_SEND

El estado de conversación cambia a Enviar-Pendiente para el TP local cuando el TP local recibe uno de los valores siguientes mediante el parámetro *what_rcvd* de un verbo de recepción posterior:

- AP_DATA_SEND
- AP_DATA_COMPLETE_SEND

Los verbos RECEIVE son [MC_]RECEIVE_AND_WAIT, [MC_]RECEIVE_IMMEDIATE y [MC_]RECEIVE_AND_POST.

Estructura del VCB: MC_REQUEST_TO_SEND

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_REQUEST_TO_SEND es la siguiente:

```
typedef struct mc_request_to_send
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  AP_UINT16         primary_rc;
  AP_UINT32      secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32         conv_id;
} MC_REQUEST_TO_SEND;
```

Estructura del VCB: REQUEST_TO_SEND

La definición de la estructura del VCB para el verbo REQUEST_TO_SEND es la siguiente:

```
typedef struct request_to_send
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  AP_UINT16         primary_rc;
  AP_UINT32      secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32         conv_id;
} REQUEST_TO_SEND;
```

Estructura del VCB: MC_REQUEST_TO_SEND (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_REQUEST_TO_SEND es la siguiente:

```
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;
```

Estructura del VCB: REQUEST_TO_SEND (Windows)

La definición de la estructura del VCB para el verbo REQUEST_TO_SEND es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_REQUEST_TO_SEND
Para el verbo MC_REQUEST_TO_SEND.

AP_B_REQUEST_TO_SEND
Para el verbo REQUEST_TO_SEND.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_REQUEST_TO_SEND.

AP_BASIC_CONVERSATION
Para el verbo REQUEST_TO_SEND.

Si el verbo se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con el valor AP_NON_BLOCKING.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id

Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Conversación desasignada: Si el TP asociado ha desasignado la conversación, APPC devolverá el siguiente valor:

primary_rc

AP_CONVERSATION_ENDED
Este verbo se ha emitido como un verbo de no bloqueo y se ha puesto en cola detrás de un verbo anterior. El TP asociado ha emitido el verbo [MC_]DEALLOCATE del mismo modo que para el verbo AP_DEALLOC_NORMAL antes mencionado, y el primer verbo de la cola ha vuelto con *primary_rc* establecido en AP_DEALLOC_NORMAL, que indica el fin de la conversación. Los verbos posteriores de la cola vuelven con *primary_rc* establecido en AP_CONVERSATION_ENDED, que indica que la conversación ya había finalizado antes de que el verbo se pudiese procesar.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AIX, LINUX

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

AP_R_TO_S_INVALID_FOR_FDX
El TP local ha intentado utilizar el verbo [MC_]REQUEST_TO_SEND en una conversación dúplex. Este verbo sólo se puede utilizar en una conversación semidúplex.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc

AP_R_T_S_BAD_STATE
La conversación no se encontraba en un estado permitido cuando el TP ha emitido este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación puede encontrarse en cualquiera de los estados siguientes:

- Recibir
- Confirmar
- Pendiente-Envío

Cambio de estado

El estado de conversación no cambia con este verbo.

Recepción de notificación de petición de envío

El programa asociado recibe la notificación de petición de envío mediante el parámetro *rts_rcvd* de los verbos siguientes:

- [MC]CONFIRM
- [MC_]RECEIVE_AND_POST
- [MC_]RECEIVE_AND_WAIT
- [MC_]RECEIVE_IMMEDIATE
- [MC_]SEND_DATA
- [MC_]SEND_ERROR

También se indica mediante un parámetro *primary_rc* definido en AP_OK en el verbo [MC_]TEST_RTS o mediante una devolución de llamada al verbo [MC_]TEST_RTS_AND_POST.

La notificación de petición de envío se envía al TP asociado de inmediato; APPC no espera a que el almacenamiento intermedio de envío se llene o se vacíe. Por consiguiente, la notificación de petición de envío puede llegar fuera de secuencia. Por ejemplo, si el TP asociado se encuentra en estado Enviar y emite el verbo [MC_]PREPARE_TO_RECEIVE seguido del verbo [MC_]REQUEST_TO_SEND, el TP asociado, en estado Recibir, puede recibir la notificación de petición de envío antes de recibir la notificación de envío. Por ello puede informarse a un TP de la notificación de petición de envío en un verbo de recepción.

MC_SEND_CONVERSATION y SEND_CONVERSATION

El verbo MC_SEND_CONVERSATION o SEND_CONVERSATION establece una conversación con el TP asociado, envía un único registro de datos en esta conversación y la desasigna. Equivale a emitir los tres verbos [MC_]ALLOCATE, [MC_]SEND_DATA y [MC_]DEALLOCATE(FLUSH).

Estructura del VCB: MC_SEND_CONVERSATION

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_SEND_CONVERSATION es la siguiente:

```
typedef struct mc_send_conversation
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado           */
  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;
```

Estructura del VCB: SEND_CONVERSATION

La definición de la estructura del VCB para el verbo SEND_CONVERSATION es la siguiente:

```
typedef struct send_conversation
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado           */
```

```
  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;
```

Estructura del VCB: MC_SEND_CONVERSATION (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_SEND_CONVERSATION es la siguiente:

```
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;
```

Estructura del VCB: SEND_CONVERSATION (Windows)

La definición de la estructura del VCB para el verbo SEND_CONVERSATION es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_SEND_CONVERSATION
Para el verbo MC_SEND_CONVERSATION.

AP_B_SEND_CONVERSATION
Para el verbo SEND_CONVERSATION.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_SEND_CONVERSATION.

AP_BASIC_CONVERSATION
Para el verbo SEND_CONVERSATION.

Si el verbo se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con el valor AP_NON_BLOCKING.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

rtn_ctl Especifica cuándo la LU local que participa en una petición de sesión del TP local debe devolver el control al TP local. Para ver más información sobre las sesiones, consulte “Sesiones LU-LU” en la página 61. Sea cual sea el valor de este parámetro, la LU devuelve inmediatamente el control al TP

si encuentra determinados errores, tales como un límite de sesiones cero (lo que significa que nunca se asignará una sesión).

Los valores posibles son:

AP_IMMEDIATE
Si una sesión ganadora de la contienda está disponible de inmediato (activa y no utilizada por otra conversación), la LU le asigna esta conversación e inmediatamente devuelve el control al TP. Si una sesión ganadora de la contienda no está inmediatamente disponible, se devuelve el control al TP de inmediato con el parámetro *primary_rc* con el valor AP_UNSUCCESSFUL.

AP_WHEN_SESSION_ALLOCATED
Si una sesión está disponible de inmediato (activa y no utilizada por otra conversación), la LU le asigna esta conversación. Si no hay ninguna sesión disponible de inmediato pero puede activarse una, la LU la activa y le asigna la conversación; si no puede activar una sesión, espera a que se libere una.

AP_WHEN_SESSION_FREE
Si una sesión está disponible de inmediato (activa y no utilizada por otra conversación), la LU le asigna esta conversación. Si no hay ninguna sesión disponible de inmediato pero puede activarse una, la LU la activa y le asigna la conversación. Si no hay ninguna sesión activa libre y no puede activarse otra sesión, se devuelve el control al TP con el código de retorno primario AP_ALLOCATION_ERROR y el código de retorno secundario AP_ALLOCATION_FAILURE_RETRY. Es parecido a AP_WHEN_SESSION_ALLOCATED salvo que la LU no esperará a que se libere una sesión.

AP_WHEN_CONWINNER_ALLOC
Igual que AP_WHEN_SESSION_ALLOCATED, excepto que la LU siempre asigna la conversación a una sesión ganadora de la contienda; no utilizará una sesión perdedora de la contienda.

AP_WHEN_CONLOSER_ALLOC
Igual que AP_WHEN_SESSION_ALLOCATED, excepto que la LU siempre asigna la conversación a una sesión perdedora de la contienda; no utilizará una sesión ganadora de la contienda.

AP_WHEN_CONV_GROUP_ALLOC
Utilice este valor si quiere que la nueva conversación utilice la misma sesión que la de una conversación anterior; establezca el parámetro *conv_group_id* en el identificador de grupo de conversación de la conversación anterior, que se ha devuelto en el verbo [MC_]ALLOCATE o en RECEIVE_ALLOCATE.

Si la sesión identificada por el parámetro *conv_group_id* está disponible inmediatamente (activa y no utilizada por otra conversación), la LU le asigna esta conversación e inmediatamente devuelve el control al TP. Si otra conversación utiliza la sesión, la LU espera a que se libere. Si la sesión ya no está activa, se devuelve el control al TP con el código de retorno primario AP_ALLOCATION_ERROR y el código de retorno secundario AP_ALLOCATION_FAILURE_NO_RETRY

conv_group_id
Identificador de grupo de conversación de la sesión solicitada para la

conversación. Este parámetro sólo se utiliza si *rtn_ctl* tiene el valor AP_WHEN_CONV_GROUP_ALLOC; defínalo en ceros binarios para cualquier otro valor de *rtn_ctl*.

plu_alias
Alias por el que se conoce la LU asociada en el TP local. Este nombre debe coincidir con el nombre de una LU asociada establecido durante la configuración.

Este parámetro es una cadena de caracteres ASCII de 8 bytes, rellenada por la derecha con blancos ASCII (0x20) si el alias tiene menos de ocho caracteres. Puede constar de cualquiera de los siguientes caracteres:

- Letras en mayúsculas
- Números del 0 al 9
- Blancos
- Los caracteres especiales $, #, % y @

El primer carácter de esta cadena no puede ser un blanco.

Para identificar la LU por su nombre de LU en lugar de por su alias de LU, establezca este parámetro en 8 ceros binarios y especifique el nombre de LU en el parámetro *fqplu_name*.

mode_name
Nombre de un conjunto de características de red definidas durante la configuración.

El valor de *mode_name* debe coincidir con el nombre de una modalidad relacionada con la LU asociada durante la configuración.

Este parámetro es una cadena de caracteres EBCDIC de 8 bytes. Puede constar de caracteres del juego de caracteres EBCDIC de tipo A. Estos caracteres son los siguientes:

- Letras en mayúsculas
- Números del 0 al 9
- Los caracteres especiales $, # y @

El primer carácter de la cadena debe ser una letra en mayúsculas o un carácter especial. Si el nombre de la modalidad tiene menos de ocho caracteres, rellénelo por la derecha con blancos EBCDIC (0x40).

Un nombre de modalidad también puede estar formado por todo blancos EBCDIC (0x40).

En una conversación correlacionada, el nombre no puede ser SNASVCMG (un nombre de modalidad reservado utilizado internamente por APPC). No se recomienda el uso de este nombre en una conversación básica.

tp_name
Nombre del TP invocado.

El valor de *tp_name* especificado por el verbo [MC_]ALLOCATE en el TP que invoca debe coincidir con el valor de *tp_name* especificado por el verbo RECEIVE_ALLOCATE en el TP invocado.

Este parámetro es una cadena de caracteres EBCDIC de 64 bytes sensible a las mayúsculas y minúsculas. El parámetro *tp_name* normalmente consta de caracteres del juego de caracteres EBCDIC de tipo AE (salvo cuando se denomina un TP de servicio). Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas

- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

Si el nombre de TP tiene menos de 64 bytes, utilice blancos EBCDIC (0x40) para rellenarlo por la derecha.

El convenio SNA para denominar un TP de servicio es una excepción a lo mencionado anteriormente; el nombre consta de hasta cuatro caracteres, el primero de los cuales es un byte hexadecimal entre 0x00 y 0x3F. Los demás caracteres pertenecen al juego de caracteres EBCDIC de tipo AE.

security

Especifica la información que necesita la LU asociada para validar el acceso al TP invocado.

En función de la seguridad de conversación establecida para el TP invocado durante la configuración, utilice uno de los valores siguientes:

AP_NONE
El TP invocado no utiliza la seguridad de conversación. (Si utiliza este valor, el TP invocado debe estar configurado para no utilizar la seguridad de conversación.)

AP_PGM El TP invocado utiliza la seguridad de conversación y por consiguiente requiere un identificador de usuario y una contraseña. Suministre esta información mediante los parámetros *user_id* y *pwd*.

AP_PGM_STRONG
El TP invocado utiliza la seguridad de conversación y por consiguiente requiere un identificador de usuario y una contraseña. Además, al establecer AP_PGM_STRONG se estipula que Communications Server para Linux cifra la contraseña cuando se envía a través de la red. Suministre el identificador de usuario y la contraseña mediante los parámetros *user_id* y *pwd*.

AP_SAME
Utilice este valor cuando su TP ha sido invocado por otro TP, utilizando un identificador de usuario y una contraseña válidos, y ahora invoca un tercer TP que también requiere la seguridad de conversación. (La situación en que un TP invoca un segundo TP que después invoca un tercer TP se ilustra en “Conversaciones múltiples” en la página 3.) Este valor informa al tercer TP (el TP invocado) de que la seguridad de conversación ya ha sido verificada para el primer TP que invoca.

Si utiliza este valor, el parámetro *tp_id* proporcionado en este verbo [MC_]SEND_CONVERSATION debe ser el mismo que se ha devuelto en el verbo RECEIVE_ALLOCATE cuando se ha invocado este TP.

AIX, LINUX

También puede utilizarse este valor si su TP no ha sido invocado por otro TP pero ha obtenido y verificado la información de seguridad pertinente por otros medios (por ejemplo, del nombre de usuario de Linux y la contraseña suministrados durante el inicio de sesión). En este caso, APPC utiliza el nombre de usuario de Linux con el que se ejecuta la aplicación, truncado en 10 caracteres si es necesario, como el identificador de usuario para la seguridad

de conversación; asegúrese de que este nombre consta de caracteres de cadena de tipo AE válidos (consulte la descripción del parámetro *user_id*) y que es un nombre de usuario válido para el TP que se invoca.

Si el TP ha obtenido la seguridad de conversación por otros medios (por ejemplo, solicitando al usuario que escriba un identificador de usuario y una contraseña válidos antes de asignar la conversación), debe utilizar el verbo SET_TP_PROPERTIES para especificar este identificador de usuario antes de emitir [MC_]SEND_CONVERSATION.

pwd Contraseña asociada con *user_id*.

Este parámetro sólo es necesario si el parámetro *security* tiene el valor `AP_PGM` o `AP_PGM_STRONG`; de lo contrario está reservado.

Los parámetros *pwd* y *user_id* deben coincidir con un par de identificador de usuario y contraseña configurado en la máquina en que está ubicado el TP invocado.

Este parámetro es una cadena de caracteres EBCDIC de 10 bytes sensible a las mayúsculas y minúsculas. El parámetro *pwd* puede constar de caracteres del juego de caracteres EBCDIC de tipo AE. Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

Si la contraseña tiene menos de 10 bytes, utilice blancos EBCDIC (`0x40`) para rellenarla por la derecha.

user_id Identificador de usuario necesario para acceder al TP asociado.

Este parámetro sólo es necesario si el parámetro *security* tiene el valor `AP_PGM` o `AP_PGM_STRONG`; de lo contrario está reservado.

Los parámetros *pwd* y *user_id* deben coincidir con un par de identificador de usuario y contraseña configurado en la máquina en que está ubicado el TP invocado.

Este parámetro es una cadena de caracteres EBCDIC de 10 bytes sensible a las mayúsculas y minúsculas. El parámetro *user_id* puede constar de caracteres del juego de caracteres EBCDIC de tipo AE. Estos caracteres son los siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales $, #, @ y el punto (.)

Si el identificador de usuario tiene menos de 10 bytes, utilice blancos EBCDIC (`0x40`) para rellenarlo por la derecha.

pip_dlen

Longitud de PIP (parámetros de inicialización de programa) para pasar al TP asociado.

El rango de este valor es `0–32.767`.

No todas las implementaciones de APPC pueden recibir datos PIP (aunque sí pueden enviarlos); además, CPI-C no soporta datos PIP. Defina *pip_dlen* en `0` (cero) si el TP asociado utiliza una implementación de APPC que no soporta datos PIP o si el TP asociado es una aplicación CPI-C.

pip_dptr
Dirección del almacenamiento intermedio que contiene datos PIP.

Utilice únicamente este parámetro cuando *pip_dlen* sea superior a `0` (cero).

Los datos PIP pueden ser parámetros de inicialización o información de configuración del entorno que requiere un TP asociado o un sistema operativo remoto. Los datos PIP deben tener el formato de corriente de datos general (GDS). Si desea obtener más información, consulte la publicación de IBM *Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2.*

fqplu_name
Nombre de LU completamente calificado de la LU asociada. Este parámetro sólo se utiliza si *plu_alias* está definido en ceros. Este nombre debe coincidir con el nombre de una LU asociada establecido durante la configuración.

Este nombre es una cadena EBCDIC de 17 bytes, rellenada por la derecha con espacios EBCDIC, que contiene uno de los elementos siguientes:

- Un identificador de red de 1–8 caracteres de cadena de tipo A, un carácter de punto EBCDIC y un nombre de LU de 1–8 caracteres de cadena de tipo A.
- Un nombre de LU de 1–8 caracteres de cadena de tipo A (sin el identificador de red ni el punto EBCDIC).

dlen Número de bytes de datos para enviar. El rango de este valor es `0`–`65.535`.

dptr Dirección del almacenamiento intermedio que contiene los datos que se deben enviar.

WINDOWS

El almacenamiento intermedio de datos puede residir en un área de datos estática o en un área asignada globalmente. El almacenamiento intermedio de datos debe caber completamente en esta área.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
`AP_OK`

conv_group_id
Identificador de grupo de conversación de la sesión utilizada por la conversación.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_LL
Este código de retorno sólo se aplica al verbo SEND_CONVERSATION. El campo de longitud de registro lógico de un registro lógico contiene un valor que no es válido (0x0000, 0x0001, 0x8000 o 0x8001). Para ver más información, consulte "Registros lógicos" en la página 62.

AP_BAD_RETURN_CONTROL
El valor especificado para *rtn_ctl* no es válido.

AP_BAD_SECURITY
El valor especificado para la seguridad no es válido.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_PIP_LEN_INCORRECT
El valor de *pip_dlen* es superior a 32.767.

AP_UNKNOWN_PARTNER_MODE
El valor especificado para *plu_alias* o *mode_name* no es válido.

AIX, LINUX

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Sesión no disponible: En función del valor especificado para *rtn_ctl*, APPC puede devolver el parámetro siguiente:

primary_rc

AP_UNSUCCESSFUL
El parámetro suministrado *rtn_ctl* especifica el retorno inmediato del control (AP_IMMEDIATE) al TP, y la LU local no tiene una sesión ganadora de la contienda disponible.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo el estado de conversación es Restablecer. Puede emitirse durante una conversación existente que se encuentre en cualquier estado, puesto que siempre supone el inicio de una nueva conversación que se encuentra en estado Restablecer.

Cambio de estado

El estado de conversación no cambia con este verbo.

MC_SEND_DATA y SEND_DATA

El verbo MC_SEND_DATA o SEND_DATA coloca los datos en el almacenamiento intermedio de envío de la LU local para transmitirlos al TP asociado.

Los datos recopilados en el almacenamiento intermedio de envío de la LU local se transmiten a la LU asociada (y al TP asociado) cuando se produce una de las situaciones siguientes:

- El almacenamiento intermedio de envío se llena.
- El TP local emite un verbo que vacía el almacenamiento intermedio de envío de la LU. Los verbos que realizan esta acción son [MC_]CONFIRM, [MC_]DEALLOCATE, [MC_]FLUSH, [MC_]PREPARE_TO_RECEIVE, [MC_]RECEIVE_AND_WAIT, [MC_]RECEIVE_AND_POST y [MC_]SEND_ERROR. ([MC_]CONFIRM, [MC_]PREPARE_TO_RECEIVE y [MC_]RECEIVE_AND_POST sólo son válidos para las conversaciones semidúplex).

El verbo MC_SEND_DATA o SEND_DATA también incluye opciones que le permiten ejecutar la función del verbo [MC_]CONFIRM, [MC_]DEALLOCATE, [MC_]FLUSH o [MC_]PREPARE_TO_RECEIVE, además de enviar los datos. Equivale a emitir [MC_]SEND_DATA seguido de otro verbo. ([MC_]CONFIRM y [MC_]PREPARE_TO_RECEIVE sólo son válidos para las conversaciones semidúplex).

Estructura del VCB: MC_SEND_DATA

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_SEND_DATA es la siguiente:

```
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;
```

Estructura del VCB: SEND_DATA

La definición de la estructura del VCB para el verbo SEND_DATA es la siguiente:

```
typedef struct send_data
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: MC_SEND_DATA (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_SEND_DATA es la siguiente:

```
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;
```

Estructura del VCB: SEND_DATA (Windows)

La definición de la estructura del VCB para el verbo SEND_DATA es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_SEND_DATA
Para el verbo MC_SEND_DATA.

AP_B_SEND_DATA
Para el verbo SEND_DATA.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_SEND_DATA.

AP_BASIC_CONVERSATION
Para el verbo SEND_DATA.

Si el verbo se está utilizando en una conversación dúplex o se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con uno de los siguientes valores, o con ambos:

AP_FULL_DUPLEX_CONVERSATION
El verbo se está emitiendo en una conversación dúplex.

AP_NON_BLOCKING
El verbo se está emitiendo como un verbo de no bloqueo.

format Este parámetro sólo se aplica al verbo MC_SEND_DATA de conversación correlacionada.

Si crea una nueva aplicación APPC o vuelve a compilar una aplicación APPC existente con el archivo de cabecera APPC actual para Communications Server para Linux, deberá establecer este parámetro en 1. (Las aplicaciones existentes creadas con versiones anteriores del archivo de cabecera, en que este parámetro estaba reservado, seguirán funcionando sin ningún cambio con Communications Server para Linux y no será necesario volver a crearlas).

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

dlen Número de bytes de datos que se deben colocar en el almacenamiento intermedio de envío de la LU local. El rango de este valor es 0–65.535.

dptr Dirección del almacenamiento intermedio que contiene los datos que se deben colocar en el almacenamiento intermedio de envío de la LU local.

WINDOWS

El almacenamiento intermedio de datos puede residir en un área de datos estática o en un área asignada globalmente. El almacenamiento intermedio de datos debe caber completamente en esta área.

type Indica si se debe ejecutar la función de otro verbo APPC además del verbo [MC_]SEND_DATA. Los valores posibles son:

AP_NONE
Sólo envíe datos; no ejecute ninguna otra función.

AP_SEND_DATA_CONFIRM
Esta opción sólo es válida en una conversación semidúplex. No la utilice si el parámetro *opext* incluye la opción AP_FULL_DUPLEX_CONVERSATION.

Efectúe la función del verbo [MC_]CONFIRM. Equivale a emitir [MC_]SEND_DATA seguido de [MC_]CONFIRM. Para el verbo SEND_DATA de conversación básica, los datos enviados en este verbo deben ser un registro lógico completo o el final de un registro lógico; este valor no puede utilizarse si se envía un registro lógico incompleto.

AP_SEND_DATA_FLUSH
Efectúe la función del verbo [MC_]FLUSH. Equivale a emitir [MC_]SEND_DATA seguido de [MC_]FLUSH. Para el verbo SEND_DATA de conversación básica, los datos enviados en este verbo deben ser un registro lógico completo o el final de un registro lógico; este valor no puede utilizarse si se envía un registro lógico incompleto.

AP_SEND_DATA_P_TO_R_FLUSH
Esta opción sólo es válida en una conversación semidúplex. No la utilice si el parámetro *opext* incluye la opción AP_FULL_DUPLEX_CONVERSATION.

Efectúe la función del verbo [MC_]PREPARE_TO_RECEIVE con *ptr_type* definido en AP_FLUSH. Equivale a emitir [MC_]SEND_DATA seguido de [MC_]PREPARE_TO_RECEIVE. Para el verbo SEND_DATA de conversación básica, los datos enviados en este verbo deben ser un registro lógico completo o el final de un registro lógico; este valor no puede utilizarse si se envía un registro lógico incompleto.

AP_SEND_DATA_P_TO_R_CONFIRM
Esta opción sólo es válida en una conversación semidúplex. No la utilice si el parámetro *opext* incluye la opción AP_FULL_DUPLEX_CONVERSATION.

Efectúe la función del verbo [MC_]PREPARE_TO_RECEIVE con *ptr_type* definido en AP_CONFIRM_TYPE. Equivale a emitir [MC_]SEND_DATA seguido de [MC_]PREPARE_TO_RECEIVE. Para el verbo SEND_DATA de conversación básica, los datos

enviados en este verbo deben ser un registro lógico completo o el final de un registro lógico; este valor no puede utilizarse si se envía un registro lógico incompleto.

AP_SEND_DATA_P_TO_R_SYNC_LEVEL
Esta opción sólo es válida en una conversación semidúplex. No la utilice si el parámetro *opext* incluye la opción AP_FULL_DUPLEX_CONVERSATION.

Efectúe la función del verbo [MC_]PREPARE_TO_RECEIVE con *ptr_type* definido en AP_SYNC_LEVEL. Equivale a emitir [MC_]SEND_DATA seguido de [MC_]PREPARE_TO_RECEIVE. Para el verbo SEND_DATA de conversación básica, los datos enviados en este verbo deben ser un registro lógico completo o el final de un registro lógico; este valor no puede utilizarse si se envía un registro lógico incompleto.

AP_SEND_DATA_DEALLOC_FLUSH
Efectúe la función del verbo [MC_]DEALLOCATE con *dealloc_type* definido en AP_FLUSH. Equivale a emitir [MC_]SEND_DATA seguido de [MC_]DEALLOCATE. Para el verbo SEND_DATA de conversación básica, los datos enviados en este verbo deben ser un registro lógico completo o el final de un registro lógico; este valor no puede utilizarse si se envía un registro lógico incompleto.

AP_SEND_DATA_DEALLOC_CONFIRM
Esta opción sólo es válida en una conversación semidúplex. No la utilice si el parámetro *opext* incluye la opción AP_FULL_DUPLEX_CONVERSATION.

Efectúe la función del verbo [MC_]DEALLOCATE con *dealloc_type* definido en AP_CONFIRM_TYPE. Equivale a emitir [MC_]SEND_DATA seguido de [MC_]DEALLOCATE. Para el verbo SEND_DATA de conversación básica, los datos enviados en este verbo deben ser un registro lógico completo o el final de un registro lógico; este valor no puede utilizarse si se envía un registro lógico incompleto.

AP_SEND_DATA_DEALLOC_SYNC_LEVEL
Efectúe la función del verbo [MC_]DEALLOCATE con *dealloc_type* definido en AP_SYNC_LEVEL. Equivale a emitir [MC_]SEND_DATA seguido de [MC_]DEALLOCATE. Para el verbo SEND_DATA de conversación básica, los datos enviados en este verbo deben ser un registro lógico completo o el final de un registro lógico; este valor no puede utilizarse si se envía un registro lógico incompleto.

AP_SEND_DATA_DEALLOC_ABEND
Efectúe la función del verbo MC_DEALLOCATE con *dealloc_type* definido en AP_ABEND o el verbo DEALLOCATE con *dealloc_type* definido en AP_ABEND_PROG. Equivale a emitir [MC_]SEND_DATA seguido de [MC_]DEALLOCATE.

No puede utilizar [MC_]SEND_DATA para efectuar la función de [MC_]DEALLOCATE en los siguientes casos:

- El tipo de conversación es AP_BASIC_CONVERSATION y el valor del parámetro *dealloc_type* necesario es AP_ABEND_SVC o AP_ABEND_TIMER.
- El nivel de sincronización de la conversación es AP_SYNCPT y el TP requiere una notificación de FORGET implícito.

En estos casos, tiene que emitir [MC_]SEND_DATA y [MC_]DEALLOCATE por separado. Para ver más información, consulte la descripción del verbo [MC_]DEALLOCATE en el Capítulo 4, "Verbos de conversación APPC", en la página 97.

AIX, LINUX

data_type
Especifica el formato de los datos que se envían. Este parámetro sólo es utilizado por el verbo MC_SEND_DATA de conversación correlacionada. Los valores posibles son:

AP_APPLICATION
Datos de aplicación de APPC estándar. Communications Server para Linux envía los datos a la LU asociada en variables GDS de datos de aplicación.

AP_USER_CONTROL_DATA
Datos de control de usuario. Communications Server para Linux envía los datos a la LU asociada en variables GDS de datos de control de usuario. No active esta opción si la LU asociada no puede aceptar los datos en este formato.

AP_PS_HEADER
Datos de cabecera PS. Este formato de datos es utilizado sólo por los TP de punto de sincronización. No lo active si el nivel de sincronización de la conversación no es AP_SYNCPT. El gestor de puntos de sincronización se encarga de convertir los mandatos en cabeceras PS adecuadas.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

rts_rcvd
Indicador de petición de envío recibida. Este parámetro sólo es válido para las conversaciones semidúplex; no se utiliza en las conversaciones dúplex.

Los valores posibles son:

AP_YES El TP asociado ha emitido un verbo [MC_]REQUEST_TO_SEND, que solicita que el TP local cambie la conversación al estado Recibir. Para cambiar al estado Recibir, el TP local puede utilizar el verbo [MC_]PREPARE_TO_RECEIVE, [MC_]RECEIVE_AND_WAIT o [MC_]RECEIVE_AND_POST.

AP_NO El TP asociado no ha emitido el verbo [MC_]REQUEST_TO_SEND.

expd_rcvd
Indicador de datos acelerados.

Los valores posibles son:

AP_YES El TP asociado ha enviado datos acelerados que el TP local aún no ha recibido. Para recibir estos datos, el TP local puede utilizar el verbo [MC_]RECEIVE_EXPEDITED_DATA.

Este indicador se puede establecer en diversos verbos APPC. Seguirá establecido en los verbos posteriores hasta que el TP local emita el verbo [MC_]RECEIVE_EXPEDITED_DATA para recibir los datos.

AP_NO No hay datos acelerados a la espera de ser recibidos.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_LL
Este código de retorno sólo se aplica al verbo SEND_DATA. El campo de longitud de registro lógico de un registro lógico contiene un valor que no es válido (0x0000, 0x0001, 0x8000 o 0x8001). Para ver más información, consulte "Registros lógicos" en la página 62.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

WINDOWS

AP_INVALID_DATA_SEGMENT
Los datos son más largos que el segmento de datos asignado o la dirección del almacenamiento intermedio de datos es incorrecta.

AP_SEND_DATA_INVALID_TYPE
El parámetro *type* está definido en un valor que no es válido.

AIX, LINUX

AP_INVALID_FORMAT
El parámetro *format* está definido en un valor que no es válido.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

AP_SEND_TYPE_INVALID_FOR_FDX
La aplicación ha emitido este verbo en una conversación dúplex pero el parámetro *type* ha especificado un tipo de envío que no es válido en una conversación dúplex.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc
Los valores posibles son:

AP_SEND_DATA_NOT_SEND_STATE
El TP local ha emitido el verbo [MC_]SEND_DATA, pero la conversación no se encuentra en estado Enviar ni Enviar-Pendiente.

AP_SEND_DATA_CONFIRM_SYNC_NONE
El TP local ha emitido el verbo [MC_]SEND_DATA con el parámetro *type* definido en AP_SEND_DATA_CONFIRM, pero el nivel de sincronización de la conversación es AP_NONE. La función CONFIRM sólo es válida si el nivel de sincronización es AP_CONFIRM_SYNC_LEVEL.

AP_SEND_DATA_NOT_LL_BDY
(Valor devuelto sólo para el verbo SEND_DATA de conversación básica.) El TP local ha emitido el verbo SEND_DATA para enviar un registro lógico incompleto y ha utilizado un parámetro *type* diferente de AP_NONE o AP_SEND_DATA_DEALLOC_ABEND.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

El verbo MC_SEND_DATA devuelve el siguiente código de retorno primario:

primary_rc
AP_DEALLOC_ABEND

APPC no devuelve un código de retorno secundario con este código de retorno primario.

El verbo SEND_DATA devuelve los siguientes códigos de retorno primarios:

primary_rc

```
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING
```

APPC no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación debe encontrarse en estado Enviar-Recibir (sólo en las conversaciones dúplex), Enviar o Enviar-Pendiente.

Cambio de estado

Los cambios de estado, resumidos en la tabla siguiente, dependen del parámetro *primary_rc*.

primary_rc	**Estado nuevo**
AP_OK	Enviar (conversación semidúplex) o sin cambios (conversación dúplex)
AP_STATE_CHECK 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	Sin cambio
AP_ALLOCATION_ERROR	Restablecer
AP_CONV_FAILURE_RETRY AP_CONV_FAILURE_NO_RETRY	Restablecer
AP_DEALLOC_ABEND AP_DEALLOC_ABEND_PROG AP_DEALLOC_ABEND_SVC AP_DEALLOC_ABEND_TIMER	Restablecer
AP_PROG_ERROR_PURGING AP_SVC_ERROR_PURGING	Recibir (conversación semidúplex), Enviar-Recibir (conversación dúplex, verbo emitido en estado Enviar-Recibir) o Restablecer (conversación dúplex, verbo emitido en estado Sólo-Enviar)

En espera del TP asociado

El verbo [MC_]SEND_DATA puede esperar indefinidamente porque el TP asociado no ha emitido ningún verbo de recepción. Esto se debe a que el almacenamiento intermedio de envío puede llenarse y APPC no puede transmitir su contenido a la LU asociada porque ésta no tiene ningún almacenamiento intermedio para recibir los datos.

Registros lógicos en conversaciones básicas

Al utilizar el verbo SEND_DATA de conversación básica, la aplicación debe suministrar datos en forma de registros lógicos (con un campo LLID al inicio de cada registro de datos). Para ver más información, consulte “Registros lógicos” en la página 62.

AIX, LINUX

En una conversación con el nivel de sincronización AP_SYNCPT, los datos que se deben enviar pueden estar en formato de cabecera PS; esto se indica mediante un campo de longitud de 0x0001. El gestor de puntos de sincronización se encarga de configurar las cabeceras PS adecuadas según las funciones de punto de sincronización que requiere la aplicación.

MC_SEND_ERROR y SEND_ERROR

El verbo MC_SEND_ERROR o SEND_ERROR notifica al TP asociado que el TP local ha encontrado un error a nivel de aplicación.

El TP local envía la notificación de error inmediatamente al TP asociado; no retiene la información en el almacenamiento intermedio de envío de la LU local.

Para una conversación semidúplex, después de la correcta ejecución de este verbo, la conversación está en estado Enviar para el TP local y en estado Recibir para el TP asociado. Para una conversación dúplex, después de la correcta ejecución del verbo no hay ningún cambio de estado.

Estructura del VCB: MC_SEND_ERROR

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_SEND_ERROR es la siguiente:

```
typedef struct mc_send_error
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: SEND_ERROR

La definición de la estructura del VCB para el verbo SEND_ERROR es la siguiente:

```
typedef struct send_error
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: MC_SEND_ERROR (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_SEND_ERROR es la siguiente:

```
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;
```

Estructura del VCB: SEND_ERROR (Windows)

La definición de la estructura del VCB para el verbo SEND_ERROR es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_SEND_ERROR
Para el verbo MC_SEND_ERROR.

AP_B_SEND_ERROR
Para el verbo SEND_ERROR.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_SEND_ERROR.

AP_BASIC_CONVERSATION
Para el verbo SEND_ERROR.

Si el verbo se está utilizando en una conversación dúplex o se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con uno de los siguientes valores, o con ambos:

AP_FULL_DUPLEX_CONVERSATION
El verbo se está emitiendo en una conversación dúplex.

AP_NON_BLOCKING
El verbo se está emitiendo como un verbo de no bloqueo.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

err_type
Indica el tipo de error que se está notificando. Esto determina el código de retorno que APPC envía al TP asociado para informar del error; todos estos códigos de retorno se describen en el Apéndice B, "Códigos de retorno comunes", en la página 283.

WINDOWS

Este parámetro sólo lo utiliza el verbo de conversación básica SEND_ERROR.

Los valores posibles son:

AP_PROG
El error debe notificarse a un programa de aplicación que no utiliza puntos de sincronización. Este valor hace que APPC envíe uno de los siguientes códigos de retorno al TP asociado:

- AP_PROG_ERROR_TRUNC (si el verbo SEND_ERROR se emite en estado Enviar después de enviar parte de un registro lógico).
- AP_PROG_ERROR_NO_TRUNC (si el verbo MC_SEND_ERROR se emite en estado Enviar o si el verbo SEND_ERROR se emite en estado Enviar pero no se ha enviado un registro lógico incompleto).
- AP_PROG_ERROR_PURGING (si el verbo se emite en cualquier estado salvo Enviar).

AP_SVC El error debe notificarse a un programa de servicio. Este valor sólo es utilizado por el verbo SEND_ERROR. Este valor hace que APPC envíe uno de los siguientes códigos de retorno al TP asociado:

- AP_SVC_ERROR_TRUNC (si el verbo SEND_ERROR se emite en estado Enviar después de enviar parte de un registro lógico).
- AP_SVC_ERROR_NO_TRUNC (si el verbo SEND_ERROR se emite en estado Enviar pero no se ha enviado un registro lógico incompleto).
- AP_SVC_ERROR_PURGING (si el verbo SEND_ERROR se emite en cualquier estado salvo Enviar).

AIX, LINUX

AP_BACKOUT_NO_RESYNC
Este valor sólo está permitido si el nivel de sincronización de la conversación es AP_SYNCPT. El TP local (u otro TP que participa en la misma unidad lógica de trabajo) ha emitido una petición BACKOUT; el TP local ha completado la restitución de sus recursos. El gestor de puntos de sincronización se encarga de emitir [MC_]SEND_ERROR con este valor definido al recibir la petición BACKOUT. Este valor hace que APPC envíe los códigos de retorno primarios y secundarios AP_BACKED_OUT y AP_BO_NO_RESYNC al TP asociado.

AP_BACKOUT_RESYNC
Este valor sólo está permitido si el nivel de sincronización de la conversación es AP_SYNCPT. El TP local (u otro TP que participa en la misma unidad lógica de trabajo) ha emitido una petición de restitución; la resincronización todavía está en curso. El gestor de puntos de sincronización se encarga de emitir [MC_]SEND_ERROR con este valor definido al recibir la petición BACKOUT. Este valor hace que APPC envíe los códigos de retorno primarios y secundarios AP_BACKED_OUT y AP_BO_RESYNC al TP asociado.

err_dir Indica si el error del que se está informando está en los datos recibidos del TP asociado o en los datos que estaba a punto de enviar el TP local.

En una conversación dúplex, se debe establecer AP_SEND_DIR_ERROR como valor de este parámetro. En una conversación semidúplex, este parámetro sólo se utiliza cuando el verbo [MC_]SEND_ERROR se emite en estado Enviar-Pendiente.

Los valores posibles son:

AP_RCV_DIR_ERROR
El TP local ha detectado un error en los datos que ha recibido del TP remoto.

AP_SEND_DIR_ERROR
El TP local ha detectado un error en sus propios datos (por ejemplo, no puede leer datos del disco) o en su propio proceso.

log_dlen
Número de bytes de datos para enviar al archivo de anotaciones de error. Este parámetro sólo es utilizado por el verbo SEND_ERROR.

El rango de este valor es `0–32.767`. La longitud `0` (cero) indica que no hay datos de anotaciones de error.

log_dptr
Dirección de almacenamiento intermedio de datos que contiene información de error. Estos datos se envían al archivo de anotaciones de error local y a la LU asociada.

Este parámetro es utilizado por el verbo SEND_ERROR si *log_dlen* es mayor que `0` (cero).

El TP debe dar formato a los datos de error como una variable de anotaciones de error de corriente de datos general (GDS). Para obtener más información, consulte la publicación de IBM *Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2.*

WINDOWS

El almacenamiento intermedio de datos puede residir en un área de datos estática o en un área asignada globalmente. El almacenamiento intermedio de datos debe caber completamente en esta área.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
`AP_OK`

rts_rcvd
Indicador de petición de envío recibida. Este parámetro sólo es válido para las conversaciones semidúplex; no se utiliza en las conversaciones dúplex. Los valores posibles son:

AP_YES El TP asociado ha emitido el verbo [MC_]REQUEST_TO_SEND, que solicita que el TP local cambie la conversación al estado Recibir. Para cambiar al estado Recibir, el TP local puede utilizar el verbo [MC_]PREPARE_TO_RECEIVE, [MC_]RECEIVE_AND_WAIT o [MC_]RECEIVE_AND_POST.

AP_NO El TP asociado no ha emitido el verbo [MC_]REQUEST_TO_SEND.

expd_rcvd
Indicador de datos acelerados.

Los valores posibles son:

AP_YES El TP asociado ha enviado datos acelerados que el TP local aún no ha recibido. Para recibir estos datos, el TP local puede utilizar el verbo [MC_]RECEIVE_EXPEDITED_DATA.

Este indicador se puede establecer en diversos verbos APPC. Seguirá establecido en los verbos posteriores hasta que el TP local emita el verbo [MC_]RECEIVE_EXPEDITED_DATA para recibir los datos.

AP_NO No hay datos acelerados a la espera de ser recibidos.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_ERROR_DIRECTION
El valor de *err_dir* no es válido.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_SEND_ERROR_BAD_TYPE
El valor de *err_type* no es válido.

AIX, LINUX

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

WINDOWS

AP_INVALID_DATA_SEGMENT
Los datos de anotaciones son más largos que el segmento de datos asignado o la dirección del almacenamiento intermedio de datos de anotaciones es incorrecta.

El siguiente valor de *secondary_rc* sólo puede devolverse en el verbo SEND_ERROR:

AP_SEND_ERROR_LOG_LL_WRONG
El campo LL de la variable de anotaciones de error GDS no coincide con la longitud real de los datos.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Verbo emitido en cualquier estado permitido: Los siguientes códigos de retorno pueden generarse cuando el verbo [MC_]SEND_ERROR se emite en cualquier estado permitido:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Verbo emitido en estado Enviar: Los siguientes códigos de retorno sólo pueden generarse si el verbo [MC_]SEND_ERROR se emite en estado Enviar:

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 no devuelve un código de retorno secundario con este código de retorno primario.

El siguiente código de retorno sólo puede generarse si el verbo MC_SEND_ERROR se emite en estado Enviar:

primary_rc
AP_DEALLOC_ABEND

APPC no devuelve un código de retorno secundario con este código de retorno primario.

Los siguientes códigos de retorno sólo pueden generarse si el verbo SEND_ERROR se emite en estado Enviar:

primary_rc
AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Verbo emitido en estado Recibir: El siguiente código de retorno sólo puede generarse si el verbo se emite en estado Recibir:

primary_rc
AP_DEALLOC_NORMAL

APPC no devuelve un código de retorno secundario con este código de retorno primario.

Estado al emitirse

Cuando el TP emite este verbo, la conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

El estado nuevo viene determinado por el código de retorno primario *primary_rc*. Los cambios de estado posibles se resumen en la tabla siguiente.

primary_rc	**Estado nuevo**
AP_OK	Enviar (conversación semidúplex) o sin cambios (conversación dúplex)
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	Sin cambio
AP_ALLOCATION_ERROR	Restablecer
AP_CONV_FAILURE_RETRY AP_CONV_FAILURE_NO_RETRY	Restablecer
AP_DEALLOC_ABEND AP_DEALLOC_ABEND_PROG AP_DEALLOC_ABEND_SVC AP_DEALLOC_ABEND_TIMER AP_DEALLOC_NORMAL	Restablecer
AP_PROG_ERROR_PURGING AP_SVC_ERROR_PURGING	Recibir (conversación semidúplex), Enviar-Recibir (conversación dúplex, verbo emitido en estado Enviar-Recibir) o Restablecer (conversación dúplex, verbo emitido en estado Sólo-Enviar)

Datos innecesarios eliminados

Si la conversación se encuentra en estado Recibir cuando el TP emite el verbo [MC_]SEND_ERROR, APPC elimina los datos entrantes innecesarios. Estos datos incluyen lo siguiente:

- Datos enviados por el verbo [MC_]SEND_DATA.
- Indicadores de código de retorno.
- Peticiones de confirmación.
- Peticiones de desasignación.

APPC no elimina ningún indicador REQUEST_TO_SEND entrante innecesario.

Indicadores de código de retorno eliminados

Los siguientes códigos de retorno primarios indican que el TP remoto o la LU remota ha detectado un error y que normalmente se informará del mismo en el próximo verbo APPC emitido por el TP local. Sin embargo, cuando el TP local emite el verbo [MC_]SEND_ERROR, estos códigos de retorno se eliminan y sustituyen por otros códigos de retorno.

El código de retorno primario AP_OK sustituye los siguientes indicadores de código de retorno eliminados:

```
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
```

El código de retorno primario AP_DEALLOC_NORMAL sustituye los siguientes indicadores de código de retorno eliminados:

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 y SEND_EXPEDITED_DATA

El verbo MC_SEND_EXPEDITED_DATA o SEND_EXPEDITED_DATA coloca los datos en el almacenamiento intermedio de envío acelerado de la LU local para transmitirlos al TP asociado.

Los datos recopilados en el almacenamiento intermedio de envío de la LU local se transmiten a la LU asociada (y al TP asociado) del mismo modo que en el caso del verbo [MC_]SEND_DATA. No obstante, puesto que los datos se envían por la red como datos acelerados, es posible que lleguen antes que los datos que se han enviado antes utilizando [MC_]SEND_DATA.

Estructura del VCB: MC_SEND_EXPEDITED_DATA

La definición de la estructura del VCB para el verbo MC_SEND_EXPEDITED_DATA es la siguiente:

```
typedef struct mc_send_expedited_data
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: SEND_EXPEDITED_DATA

La definición de la estructura del VCB para el verbo SEND_EXPEDITED_DATA es la siguiente:

```
typedef struct send_expedited_data
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_SEND_EXPEDITED_DATA
Para el verbo MC_SEND_EXPEDITED_DATA.

AP_B_SEND_EXPEDITED_DATA
Para el verbo SEND_EXPEDITED_DATA.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_SEND_EXPEDITED_DATA.

AP_BASIC_CONVERSATION
Para el verbo SEND_EXPEDITED_DATA.

Si el verbo se está utilizando en una conversación dúplex o se está emitiendo como un verbo de no bloqueo, combine el valor anterior (utilizando un operador lógico OR) con uno de los siguientes valores, o con ambos:

AP_FULL_DUPLEX_CONVERSATION
El verbo se está emitiendo en una conversación dúplex.

AP_NON_BLOCKING
El verbo se está emitiendo como un verbo de no bloqueo.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

dlen Número de bytes de datos que se deben colocar en el almacenamiento intermedio de envío de la LU local. El rango de este valor es `0`–`86`.

dptr Dirección del almacenamiento intermedio que contiene los datos que se deben colocar en el almacenamiento intermedio de envío de la LU local.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
`AP_OK`

rts_rcvd
Indicador de petición de envío recibida. Este parámetro sólo es válido para las conversaciones semidúplex; no se utiliza en las conversaciones dúplex.

Los valores posibles son:

AP_YES El TP asociado ha emitido un verbo [MC_]REQUEST_TO_SEND, que solicita que el TP local cambie la conversación al estado Recibir. Para cambiar al estado Recibir, el TP local puede utilizar el verbo [MC_]PREPARE_TO_RECEIVE, [MC_]RECEIVE_AND_WAIT o [MC_]RECEIVE_AND_POST.

AP_NO El TP asociado no ha emitido el verbo [MC_]REQUEST_TO_SEND.

expd_rcvd
Indicador de datos acelerados.

Los valores posibles son:

AP_YES El TP asociado ha enviado datos acelerados que el TP local aún no ha recibido. Para recibir estos datos, el TP local puede utilizar el verbo [MC_]RECEIVE_EXPEDITED_DATA.

Este indicador se puede establecer en diversos verbos APPC. Seguirá establecido en los verbos posteriores hasta que el TP local emita el verbo [MC_]RECEIVE_EXPEDITED_DATA para recibir los datos.

AP_NO No hay datos acelerados a la espera de ser recibidos.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

No se admiten los datos acelerados: Si el verbo no se ejecuta porque la LU remota no admite datos acelerados, APPC devolverá el parámetro siguiente:

primary_rc

AP_EXPD_NOT_SUPPORTED_BY_LU

Conversación desasignada: Si el TP asociado ha desasignado la conversación, APPC devolverá el siguiente valor:

primary_rc

AP_CONVERSATION_ENDED
Este verbo se ha emitido como un verbo de no bloqueo y se ha puesto en cola detrás de un verbo anterior. El TP asociado ha emitido el verbo [MC_]DEALLOCATE del mismo modo que para el verbo AP_DEALLOC_NORMAL antes mencionado, y el primer verbo de la cola ha vuelto con *primary_rc* establecido en AP_DEALLOC_NORMAL, que indica el fin de la conversación. Los verbos posteriores de la cola vuelven con *primary_rc* establecido en AP_CONVERSATION_ENDED, que indica que la conversación ya había finalizado antes de que el verbo se pudiese procesar.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_SEND_EXPD_INVALID_LENGTH
El parámetro *dlen* está definido en un valor que no es válido.

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

Comprobación de estado: Si la conversación se encuentra en un estado incorrecto cuando el TP emite este verbo, APPC devuelve los parámetros siguientes:

primary_rc
AP_STATE_CHECK

secondary_rc
Los valores posibles son:

AP_EXPD_DATA_BAD_CONV_STATE
El TP local ha emitido el verbo [MC_]SEND_EXPEDITED_DATA, pero la conversación se encuentra en estado Restablecer.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

El verbo MC_SEND_EXPEDITED_DATA devuelve el siguiente código de retorno primario:

primary_rc
AP_DEALLOC_ABEND

APPC no devuelve un código de retorno secundario con este código de retorno primario.

El verbo SEND_EXPEDITED_DATA devuelve los siguientes códigos de retorno primarios:

primary_rc

AP_DEALLOC_ABEND_PROG
AP_DEALLOC_ABEND_SVC
AP_DEALLOC_ABEND_TIMER
AP_SVC_ERROR_PURGING

APPC no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación debe encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

Los cambios de estado, resumidos en la tabla siguiente, dependen del parámetro *primary_rc*.

primary_rc	**Estado nuevo**
AP_OK	Sin cambio
AP_STATE_CHECK	Sin cambio
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	Restablecer
AP_CONV_FAILURE_RETRY	Restablecer
AP_CONV_FAILURE_NO_RETRY	
AP_CONVERSATION_ENDED	
AP_DEALLOC_ABEND	Restablecer
AP_DEALLOC_ABEND_PROG	
AP_DEALLOC_ABEND_SVC	
AP_DEALLOC_ABEND_TIMER	

En espera del TP asociado

Del mismo modo que sucede con [MC_]SEND_DATA, el verbo [MC_]SEND_EXPEDITED_DATA puede esperar indefinidamente porque el TP asociado no ha emitido un verbo [MC_]RECEIVE_EXPEDITED_DATA. Esto se debe a que el almacenamiento intermedio de envío acelerado puede llenarse y APPC no puede transmitir su contenido a la LU asociada porque ésta no tiene ningún almacenamiento intermedio para recibir los datos.

MC_TEST_RTS y TEST_RTS

El verbo MC_TEST_RTS o TEST_RTS determina si se ha recibido una notificación REQUEST_TO_SEND del TP asociado.

Nota: Este verbo sólo se puede utilizar en las conversaciones semidúplex; no es válido para las conversaciones dúplex.

Generalmente, si el TP asociado emite un verbo [MC_]REQUEST_TO_SEND, al TP local se le notificará de ello mediante el parámetro *rts_rcvd* en un verbo posterior (este parámetro se recibe en varios verbos). Sólo se informa del mismo en el primer verbo posterior que puede devolver este parámetro, y no en los demás verbos posteriores. El verbo [MC_]TEST_RTS permite que el TP local verifique si se ha recibido una notificación de petición de envío en algún momento desde que el TP local estaba por última vez en estado Recibir.

En lugar de emitir repetidamente [MC_]TEST_RTS, la aplicación puede utilizar [MC_]TEST_RTS_AND_POST, que se describe en "MC_TEST_RTS_AND_POST y TEST_RTS_AND_POST" en la página 246. Este verbo vuelve de manera asíncrona cuando se recibe una notificación REQUEST_TO_SEND del TP asociado. [MC_]TEST_RTS_AND_POST opera de manera asíncrona del mismo modo que [MC_]RECEIVE_AND_POST y, por consiguiente, la aplicación puede emitir otros verbos APPC mientras está pendiente.

Estructura del VCB: MC_TEST_RTS

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_TEST_RTS es la siguiente:

```
typedef struct mc_test_rts
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  AP_UINT16         primary_rc;
  AP_UINT32      secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     reserv3;
} MC_TEST_RTS;
```

Estructura del VCB: TEST_RTS

La definición de la estructura del VCB para el verbo TEST_RTS es la siguiente:

```
typedef struct test_rts
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  AP_UINT16         primary_rc;
  AP_UINT32      secondary_rc;
  unsigned char   tp_id[8];
  AP_UINT32         conv_id;
  unsigned char     reserv3;
} TEST_RTS;
```

Estructura del VCB: MC_TEST_RTS (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_TEST_RTS es la siguiente:

```
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;
```

Estructura del VCB: TEST_RTS (Windows)

La definición de la estructura del VCB para el verbo TEST_RTS es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_TEST_RTS
Para el verbo MC_TEST_RTS.

AP_B_TEST_RTS
Para el verbo TEST_RTS.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_TEST_RTS.

AP_BASIC_CONVERSATION
Para el verbo TEST_RTS.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
Indica si se ha recibido una notificación REQUEST_TO_SEND del TP asociado. Los valores posibles son:

AP_OK Se ha recibido la notificación REQUEST_TO_SEND.

AP_UNSUCCESSFUL
No se ha recibido la notificación REQUEST_TO_SEND.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AIX, LINUX

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

AP_TEST_INVALID_FOR_FDX
El TP local ha intentado utilizar el verbo [MC_]TEST_RTS en una conversación dúplex. Este verbo sólo se puede utilizar en una conversación semidúplex.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

Los códigos de retorno posibles son:

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 no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

Cuando el TP emite este verbo, la conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

El estado de conversación no cambia con este verbo.

MC_TEST_RTS_AND_POST y TEST_RTS_AND_POST

El verbo MC_TEST_RTS_AND_POST o TEST_RTS_AND_POST informa a la aplicación de que se ha recibido una notificación REQUEST_TO_SEND del TP asociado.

Nota: Este verbo sólo se puede utilizar en las conversaciones semidúplex; no es válido para las conversaciones dúplex.

Generalmente, si el TP asociado emite un verbo [MC_]REQUEST_TO_SEND, se informará de ello al TP local mediante el parámetro *rts_rcvd* en un verbo posterior (parámetro recibido en varios verbos) o mediante un código de retorno correcto en el verbo [MC_]TEST_RTS. El verbo [MC_]TEST_RTS_AND_POST permite que el TP local reciba la notificación REQUEST_TO_SEND de manera asíncrona cuando llega, en lugar de tener que emitir verbos repetidas veces para obtener dicha notificación.

Estructura del VCB: MC_TEST_RTS_AND_POST

AIX, LINUX

La definición de la estructura del VCB para el verbo MC_TEST_RTS_AND_POST es la siguiente:

```
typedef struct mc_test_rts_and_post
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado           */
  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;
```

Estructura del VCB: TEST_RTS_AND_POST

La definición de la estructura del VCB para el verbo TEST_RTS_AND_POST es la siguiente:

```
typedef struct test_rts_and_post
{
  AP_UINT16         opcode;
  unsigned char   opext;
  unsigned char    format;                    /* Reservado            */
  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;
```

Estructura del VCB: MC_TEST_RTS_AND_POST (Windows)

WINDOWS

La definición de la estructura del VCB para el verbo MC_TEST_RTS_AND_POST es la siguiente:

```
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;
```

Estructura del VCB: TEST_RTS_AND_POST (Windows)

La definición de la estructura del VCB para el verbo TEST_RTS_AND_POST es la siguiente:

```
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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode Los valores posibles son:

AP_M_TEST_RTS_AND_POST
Para el verbo MC_TEST_RTS_AND_POST.

AP_B_TEST_RTS_AND_POST
Para el verbo TEST_RTS_AND_POST.

opext Los valores posibles son:

AP_MAPPED_CONVERSATION
Para el verbo MC_TEST_RTS_AND_POST.

AP_BASIC_CONVERSATION
Para el verbo TEST_RTS_AND_POST.

tp_id Identificador del TP local.

El valor de este parámetro ha sido devuelto por el verbo TP_STARTED en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

conv_id
Identificador de conversación.

El valor de este parámetro ha sido devuelto por el verbo [MC_]ALLOCATE en el TP que invoca o por el verbo RECEIVE_ALLOCATE en el TP invocado.

AIX, LINUX

callback
Dirección de la rutina callback que APPC debe llamar cuando se recibe una notificación REQUEST_TO_SEND. Para ver más información, consulte "Notas de uso" en la página 250.

WINDOWS

sema Descriptor de contexto de sucesos de Windows que se obtiene llamando a una de las dos funciones siguientes de Windows: `CreateEvent` o `OpenEvent`. APPC señala este descriptor de contexto de sucesos para informar al TP de que se ha recibido la notificación de REQUEST_TO_SEND.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Nota: Al emitir este verbo, éste vuelve inmediatamente con un parámetro *primary_rc* que indica si el verbo se ha emitido correctamente o no. Los únicos parámetros devueltos que son válidos en este punto son *primary_rc* y *secondary_rc* (si *primary_rc* no tiene el valor `AP_OK`). Los valores posibles de *primary_rc* y *secondary_rc* se describen más adelante en este apartado.

Si el valor de este parámetro *primary_rc* es AP_OK, el verbo ha empezado a esperar correctamente la notificación REQUEST_TO_SEND. Cuando el verbo ha finalizado (o bien porque se ha recibido la notificación o bien porque ha terminado debido al fin de la conversación o a un error), APPC llama a la rutina callback suministrada. En este punto, los parámetros devueltos son los que se muestran a continuación. Los parámetros *primary_rc* y *secondary_rc* tendrán nuevos valores que indican si se ha recibido o no la notificación REQUEST_TO_SEND, y deberán volverse a examinar.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc

AP_OK Se ha recibido la notificación REQUEST_TO_SEND.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_CONV_ID
El valor de *conv_id* no coincide con un identificador de conversación asignado por APPC.

AP_BAD_TP_ID
El valor de *tp_id* no coincide con un identificador de TP asignado por APPC.

AP_INVALID_FORMAT
El campo reservado *format* tiene un valor distinto de cero.

AP_SYNC_NOT_ALLOWED
La aplicación ha emitido este verbo dentro de una rutina callback, utilizando el punto de entrada síncrono APPC. Los verbos emitidos desde una rutina callback deben utilizar el punto de entrada asíncrono.

AP_INVALID_CALLBACK_HANDLE
El parámetro *callback* se ha definido en un puntero nulo y el verbo se ha emitido utilizando el punto de entrada síncrono (o utilizando el punto de entrada asíncrono con un puntero nulo a una rutina callback). Para ver más información, consulte “Notas de uso” en la página 250.

AP_TEST_INVALID_FOR_FDX
El TP local ha intentado utilizar el verbo [MC_]TEST_RTS_AND_POST en una conversación dúplex. Este verbo sólo se puede utilizar en una conversación semidúplex.

Comprobación de estado: No hay errores de comprobación de estado para este verbo.

Verbo cancelado: Este código de retorno no puede devolverse como el código de retorno inicial, sino sólo como el código de retorno posterior si el código de retorno inicial es AP_OK. Si el verbo no se ha ejecutado porque ha sido cancelado por otro verbo emitido por el TP, APPC devuelve el siguiente parámetro:

primary_rc

AP_CANCELLED
El TP local ha emitido uno de los verbos siguientes mientras [MC_]TEST_RTS_AND_POST estaba pendiente:

- DEALLOCATE con *dealloc_type* definido en AP_ABEND_PROG, AP_ABEND_SVC o AP_ABEND_TIMER
- MC_DEALLOCATE con *dealloc_type* definido en AP_ABEND
- [MC_]SEND_ERROR
- TP_ENDED

La emisión de uno de estos verbos hace que el verbo [MC_]TEST_RTS_AND_POST se cancele. No se llama a la rutina callback.

Conversación finalizada: Si el verbo vuelve porque la conversación ha finalizado, APPC devuelve el siguiente parámetro:

primary_rc
AP_UNSUCCESSFUL

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve códigos de retorno primarios (y, si corresponde, códigos de retorno secundarios). Para ver información acerca de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

Los códigos de retorno posibles son:

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR
AP_CONVERSATION_TYPE_MIXED
AP_INVALID_VERB
AP_TP_BUSY

APPC no devuelve códigos de retorno secundarios con estos códigos de retorno primarios.

Estado al emitirse

El TP puede emitir [MC_]TEST_RTS_AND_POST cuando la conversación está en cualquier estado salvo Restablecer.

Cambio de estado

El estado de conversación no cambia con este verbo.

Notas de uso

Este apartado contiene información de uso adicional sobre los siguientes temas:

- Rutina callback
- Proceso mientras el verbo está pendiente

- Cómo utiliza el verbo el TP
- Cómo evitar esperas indefinidas

Rutina callback

AIX, LINUX

La aplicación suministra un puntero a una rutina callback como uno de los parámetros del VCB. En este apartado se describe cómo Communications Server para Linux utiliza esta rutina y las funciones que debe efectuar.

La rutina callback se define de la siguiente forma:

```
void (*callback) (
                   void *          vcb,
                   unsigned char   tp_id[8],
                   AP_UINT32       conv_id
                 );
```

Communications Server para Linux llama a la rutina con los siguientes parámetros:

vcb Puntero al VCB suministrado por la aplicación, incluidos los parámetros devueltos establecidos por Communications Server para Linux.

tp_id Identificador de TP de 8 bytes del TP en que se ha emitido el verbo.

conv_id
Identificador de conversación de la conversación en que se ha emitido el verbo.

La rutina callback no tiene que utilizar todos estos parámetros. Puede ejecutar todo el proceso necesario en el VCB devuelto o simplemente puede establecer una variable para informar al programa principal de que el verbo ha finalizado.

La aplicación puede emitir más verbos APPC desde dentro de la rutina callback si es necesario. Sin embargo, éstos deben ser verbos asíncronos. Los verbos síncronos emitidos desde una rutina callback se rechazarán con los códigos de retorno AP_PARAMETER_CHECK y AP_SYNC_NOT_ALLOWED.

Nota: Si la aplicación emite el verbo [MC_]TEST_RTS_AND_POST utilizando el punto de entrada APPC asíncrono, hay dos rutinas callback especificadas: una en el VCB y otra suministrada como un parámetro en el punto de entrada. En general, APPC utiliza la rutina callback especificada en el VCB y no tiene en cuenta la del punto de entrada; sin embargo, si la aplicación suministra un puntero nulo para la rutina callback en el VCB, APPC utiliza la rutina callback del punto de entrada.

Cómo continuar con otro proceso mientras el verbo está pendiente

Puesto que el verbo [MC_]TEST_RTS_AND_POST vuelve inmediatamente sin esperar a que lleguen datos, el TP puede continuar con otro proceso mientras espera a que aquél finalice. Sin embargo, deben tenerse en cuenta los siguientes puntos:

- El VCB suministrado al verbo [MC_]TEST_RTS_AND_POST continúa utilizándose hasta que la rutina callback vuelve. El TP no debe cambiar ningún

campo en el VCB durante este tiempo. Si emite algún otro verbo APPC mientras [MC_]TEST_RTS_AND_POST está pendiente, deberá utilizar otro VCB para el nuevo verbo.

- Sólo puede haber un verbo [MC_]TEST_RTS_AND_POST activo por conversación en todo momento.

Cómo utiliza el verbo el TP

Para utilizar el verbo [MC_]TEST_RTS_AND_POST, el TP local ejecuta los siguientes pasos:

Utilización de [MC_]TEST_RTS_AND_POST

1. Emite el verbo [MC_]TEST_RTS_AND_POST.
2. Comprueba el valor del código de retorno primario *primary_rc*:
   - Si el código de retorno primario es AP_OK, el verbo espera una notificación REQUEST_TO_SEND del TP asociado. Mientras recibe datos de manera asíncrona, el TP local puede realizar las siguientes acciones:
     – Realizar tareas no relacionadas con esta conversación.
     – Emitir otros verbos APPC en esta conversación.
     – Cancelar antes de lo debido el verbo [MC_]TEST_RTS_AND_POST emitiendo uno de los verbos siguientes:
       - DEALLOCATE con *dealloc_type* definido en AP_ABEND_PROG, AP_ABEND_SVC o AP_ABEND_TIMER
       - MC_DEALLOCATE con *dealloc_type* definido en AP_ABEND
       - SEND_ERROR
       - TP_ENDED
   - En cambio, si el código de retorno primario no es AP_OK, el verbo [MC_]TEST_RTS_AND_POST ha fallado. En ese caso, el TP local no realiza los pasos 3 y 4.
3. Comprueba que APPC haya llamado a la rutina callback (suministrada como un parámetro en este verbo). Cuando se recibe una notificación REQUEST_TO_SEND del TP asociado, APPC llama esta rutina.
4. Comprueba el nuevo valor del código de retorno primario *primary_rc*.
   - Si el código de retorno primario es AP_OK, el TP asociado ha emitido [MC_]REQUEST_TO_SEND.
   - Si el código de retorno primario no es AP_OK, la aplicación debe comprobar los parámetros *primary_rc* y *secondary_rc* para determinar la acción que debe realizar.

Cómo evitar esperas indefinidas

Si el TP local emite el verbo [MC_]TEST_RTS_AND_POST y posteriormente espera a que se llame a la rutina callback, se suspenderá hasta que reciba la notificación REQUEST_TO_SEND del TP asociado. Puede esperar indefinidamente si el TP asociado no emite [MC_]REQUEST_TO_SEND. Si necesita que el TP esté operativo de forma continua, evite esperar en la rutina callback o utilice el verbo [MC_]TEST_RTS.

Capítulo 5. Verbos de servidor de TP

AIX, LINUX

Este capítulo contiene una descripción de cada uno de los verbos de servidor de TP APPC. Para cada verbo se proporciona la siguiente información:

- Definición del verbo.
- Estructura que define el bloque de control de verbos (VCB) utilizado por el verbo. La estructura está definida en el archivo de cabecera TP **/usr/include/sna/tpsrv_c.h** (AIX) o **/opt/ibm/sna/include/tpsrv_c.h** (Linux). (Los parámetros que empiezan por *rsrvd* están reservados).
- Parámetros (campos del VCB) suministrados a APPC y devueltos por APPC. Para cada parámetro, se proporciona la siguiente información:
  - Descripción
  - Valores posibles
  - Información adicional
- Información adicional sobre el uso del verbo.

Nota:

1. Los verbos de servidor de TP se deben emitir utilizando el punto de entrada asíncrono `APPC_Async` y no el punto de entrada síncrono `APPC`. Para ver más información sobre estos puntos de entrada, consulte el Capítulo 2, “Desarrollo de programas de transacciones”, en la página 27.
2. Los verbos de servidor de TP no afectan a las conversaciones o estados APPC.

La mayoría de los parámetros suministrados a APPC y devueltos por APPC para el servidor de TP son valores hexadecimales. Para simplificar la codificación, estos valores están representados por constantes simbólicas significativas definidas en el archivo de cabecera **values_c.h**, incluido en el archivo de cabecera del servidor de TP **tpsrv_c.h**. Por ejemplo, el parámetro *opcode* del verbo REGISTER_TP_SERVER es el valor hexadecimal representado por la constante simbólica `AP_REGISTER_TP_SERVER`.

Es importante que utilice la constante simbólica y no el valor hexadecimal cuando establezca valores para los parámetros suministrados o cuando compruebe los valores de los parámetros devueltos. Esto se debe a que los diferentes sistemas Linux guardan estos valores en la memoria de modo diferente, por lo que es posible que el valor mostrado no tenga el formato que su sistema reconoce.

Los verbos de servidor de TP se describen en el orden siguiente:

REGISTER_TP_SERVER
UNREGISTER_TP_SERVER
REGISTER_TP
UNREGISTER_TP
QUERY_ATTACH
ACCEPT_ATTACH
REJECT_ATTACH
ABORT_ATTACH

REGISTER_TP_SERVER

El verbo REGISTER_TP_SERVER se utiliza para notificar a Communications Server para Linux que la aplicación puede iniciar automáticamente programas de transacciones (TP).

Estructura del VCB: REGISTER_TP_SERVER

La definición de la estructura del VCB para el verbo REGISTER_TP_SERVER es la siguiente:

```
typedef struct mc_receive_immediate
{
   AP_UINT16         opcode;
   unsigned char     rsrvd1;                 /* Reservado            */
   unsigned char     rsrvd2;                 /* Reservado            */
   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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_REGISTER_TP_SERVER

tp_file_updates
Solicita si la aplicación debe recibir una notificación cuando se actualice el archivo de configuración del TP **sna_tps**. Los valores posibles son:

AP_YES La aplicación solicita que las rutinas callback le notifiquen que se ha cambiado el archivo **sna_tps**.

AP_NO La aplicación no necesita la notificación de los cambios producidos en el archivo **sna_tps**.

notify_cb
Dirección de la función callback de notificación. APPC utiliza esta función junto con el valor del parámetro *app_corr* especificado en el verbo REGISTER_TP para notificar a un servidor de TP que se ha producido una de las siguientes situaciones:

- Hay disponible una petición Attach adecuada.
- El archivo de configuración **sna_tps** del TP ha cambiado (si la aplicación ha solicitado esta notificación definiendo el parámetro *tp_file_updates* en AP_YES).

Para ver más información acerca del uso de la función callback de notificación, consulte “Rutina callback” en la página 255.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

tps_id Identificador exclusivo para este servidor de TP. Después de que una aplicación se registre como servidor de TP, el valor del parámetro *tps_id* es válido sólo para ese proceso. El valor del parámetro *tps_id* no es válido fuera de los límites del proceso. Si otra aplicación intenta utilizar este valor del parámetro *tps_id* en otro verbo, ese verbo se rechazará con el parámetro *primary_rc* con el valor AP_PARAMETER_CHECK y el parámetro *secondary_rc* con el valor AP_BAD_TPS_ID.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_INVALID_CALLBACK
La dirección de la función callback no es válida.

Anomalía de registro: Si la aplicación no puede registrarse como servidor de TP, APPC devuelve los siguientes parámetros:

primary_rc
AP_REGISTER_FAIL

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve el siguiente código de retorno primario. Para ver una lista de los códigos de retorno comunes a todos los verbos, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

Notas de uso

Este apartado contiene información adicional sobre el uso de la rutina callback.

Rutina callback

La aplicación suministra un puntero a una rutina callback como uno de los parámetros del VCB. En este apartado se describe cómo Communications Server para Linux utiliza esta rutina y las funciones que debe efectuar.

La rutina callback se define de la siguiente forma:

```
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 para Linux llama a la rutina con los siguientes parámetros:

reason Tipo de notificación. Los valores posibles son:

AP_ATTACH
Se ha recibido una petición Attach para un TP registrado por este servidor de TP. En este caso, el parámetro *attach_id* se pasa a la rutina callback de notificación porque se utiliza la llegada de una petición Attach para realizar una o más de las siguientes acciones:

- Consultar opcionalmente más información sobre el inicio automático de un TP.
- Rechazar la petición Attach, si es necesario.
- Identificar qué TP se debe iniciar automáticamente para procesar RECEIVE_ALLOCATE.

AP_TP_FILE_CHANGE
El archivo de configuración **sna_tps** del TP ha sido modificado.

attach_id
Identificador de la petición Attach, tal como se ha devuelto en la rutina callback de notificación de petición Attach.

app_corr
Valor de correlacionador suministrado por la aplicación. Este valor permite que la aplicación correlacione la información devuelta con su otro proceso. El significado del correlacionador pasado a la rutina callback de notificación depende del tipo de notificación según especifica el valor del indicador *reason*:

- Si *reason* está definido en AP_ATTACH, el correlacionador es el que especifica la aplicación en el verbo REGISTER_TP. Esto permite que la aplicación correlacione la petición Attach con el TP registrado adecuado.
- Si *reason* está definido en AP_TP_FILE_CHANGE, el correlacionador es el valor del parámetro *tps_id* en el verbo REGISTER_TP_SERVER.

La rutina callback no tiene que utilizar todos estos parámetros. Puede ejecutar todo el proceso necesario en el VCB devuelto o simplemente establecer una variable para informar al programa principal de que el verbo ha finalizado.

UNREGISTER_TP_SERVER

El verbo UNREGISTER_TP_SERVER se utiliza cuando una aplicación ya no quiere recibir notificaciones de conexión.

Estructura del VCB: UNREGISTER_TP_SERVER

La definición de la estructura del VCB para el verbo UNREGISTER_TP_SERVER es la siguiente:

```
typedef struct unregister_tp_server
{
   AP_UINT16         opcode;
   unsigned char     rsrvd1;                 /* Reservado           */
   unsigned char     rsrvd2;                 /* Reservado           */
   AP_UINT16         primary_rc;
   AP_UINT32      secondary_rc;
   AP_UINT32         tps_id;
} UNREGISTER_TP_SERVER;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_UNREGISTER_TP_SERVER

tps_id Identificador del servidor de TP del que se debe deshacer el registro, tal como se ha devuelto en un verbo REGISTER_TP_SERVER anterior.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_TPS_ID
El valor especificado del parámetro *tps_id* no se reconoce.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve el siguiente código de retorno primario. Para ver una lista de los códigos de retorno comunes a todos los verbos, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 283.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

REGISTER_TP

El verbo REGISTER_TP se utiliza para comunicar al gestor de servicios el nombre de un TP cuyas conexiones deben ser manejadas por el servidor de TP. También puede utilizarse para cambiar el tipo de TP o el tiempo de espera de recepción de asignación para un TP que ya se ha registrado.

Estructura del VCB: REGISTER_TP

La definición de la estructura del VCB para el verbo REGISTER_TP es la siguiente:

```
typedef struct register_tp
{
   AP_UINT16         opcode;
   unsigned char     rsrvd1;                 /* Reservado            */
   unsigned char     rsrvd2;                 /* Reservado            */
   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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_REGISTER_TP

tps_id Identificador de un servidor de TP, tal como se ha devuelto en un verbo REGISTER_TP_SERVER anterior.

res_id Si se utiliza REGISTER_TP para cambiar un registro de TP existente (el parámetro *modify_existing* está definido en AP_YES), este parámetro especifica el identificador exclusivo de este recurso que se ha devuelto en el verbo REGISTER_TP original. De lo contrario, este parámetro está reservado.

tp_name
Nombre del TP que se registra. Especifique este nombre en formato EBCDIC rellenado con espacios EBCDIC, si es necesario, hasta una longitud de 64 caracteres. Especifique un valor de 64 espacios EBCDIC (0x40) para un TP para el que se manejarán todas las conexiones.

lu_alias
Alias de LU local. Especifique este nombre en formato ASCII rellenado con espacios ASCII, si es necesario, hasta una longitud de ocho caracteres. Especifique un valor de ocho espacios ASCII (0x20) para una LU para la que se manejarán todas las conexiones.

fqplu_name
Nombre completamente calificado de la LU asociada. Especifique una de

las siguientes cadenas EBCDIC rellenadas con espacios EBCDIC, si es necesario, hasta una longitud de 17 caracteres:

- Un nombre completamente calificado en EBCDIC para indicar que sólo debe establecerse la coincidencia con una petición Attach que tenga el mismo nombre completamente calificado.
- Un valor de todo espacios EBCDIC (0x40) para indicar que cualquier nombre de LU asociada se considera una coincidencia.
- Un nombre parcial, seguido de un * EBCDIC (0x5C), para indicar un nombre de LU comodín.

tp_type

Tipo de TP que se registra. Los valores posibles son:

AP_TP_TYPE_QUEUED
Las conexiones entrantes se colocan en la cola de las copias en ejecución del TP antes de intentar iniciar un nuevo TP o colocar la petición Attach en la cola en espera de un TP adecuado.

AP_TP_TYPE_QUEUED_BROADCAST
Igual que AP_TP_TYPE_QUEUED, a excepción de que la existencia del TP se difunde por el dominio Communications Server para Linux. La difusión de la existencia de este TP obvia la necesidad de configurar datos de direccionamiento de conexión para muchas LU si las maneja el mismo TP en una sola máquina.

AP_TP_TYPE_NON_QUEUED
Se inicia una nueva instancia del TP para cada petición Attach recibida, a menos que una instancia en ejecución tenga un verbo RECEIVE_ALLOCATE pendiente.

rcv_alloc_timeout

El tiempo en segundos que el verbo RECEIVE_ALLOCATE del TP debe esperar un TP iniciado automáticamente. Los valores posibles son:

0 (cero)
No esperar. Este es normalmente el valor especificado, ya que un servidor de TP inicia programas de transacciones como respuesta a una petición Attach entrante, de modo que siempre debe haber una petición Attach disponible para el verbo RECEIVE_ALLOCATE de un TP. La única excepción a esta norma se produce si la petición Attach ha excedido el tiempo de espera mientras el servidor de TP está iniciando el TP.

-1 Esperar indefinidamente.

x donde x es mayor que 0
Esperar el número de segundos indicado por x.

modify_existing

Indica si este verbo se utiliza para cambiar un registro existente o para registrar un nuevo TP. Los valores posibles son:

AP_YES Este verbo se utiliza para cambiar el parámetro *rcv_alloc_timeout*, el parámetro *type* o ambos parámetros para un registro existente. Deben tenerse en cuenta las restricciones siguientes:

- El verbo debe ser emitido por el mismo programa de servidor de TP que ha emitido el verbo REGISTER_TP original.
- El parámetro *res_id* debe estar especificado y debe coincidir con el valor devuelto en el verbo REGISTER_TP original.

- El parámetro *rcv_alloc_timeout*, el parámetro *type* o ambos parámetros pueden modificarse respecto del verbo REGISTER_TP original, pero el valor de todos los demás parámetros suministrados debe coincidir con el valor utilizado en el verbo REGISTER_TP original.

AP_NO Este verbo se utiliza para registrar un nuevo TP.

app_corr
Correlacionador proporcionado por la aplicación pasado a la rutina callback de notificación de petición Attach. Para ver más información, consulte "Notas de uso" en la página 255.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
AP_OK

res_id Identificador exclusivo para este recurso.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_INVALID_TP_NAME
El valor especificado para el parámetro *tp_name* no es válido.

AP_INVALID_LU_ALIAS
El valor especificado para el parámetro *lu_alias* no es válido.

AP_INVALID_FQ_LU_NAME
El valor especificado para el parámetro *fqplu_name* no es válido.

AP_INVALID_TIMEOUT
El valor especificado para el parámetro *rcv_alloc_timeout* no es válido.

AP_BAD_TPS_ID
El valor especificado para el parámetro *tps_id* no se reconoce.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve el siguiente código de retorno primario. Para ver una lista de los códigos de retorno comunes a todos los verbos, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

UNREGISTER_TP

El verbo UNREGISTER_TP se utiliza para informar al gestor de servicios de que la aplicación no quiere recibir notificaciones de petición Attach para el TP especificado.

Estructura del VCB: UNREGISTER_TP

La definición de la estructura del VCB para el verbo UNREGISTER_TP es la siguiente:

```
typedef struct unregister_tp
{
   AP_UINT16         opcode;
   unsigned char     rsrvd1;                 /* Reservado           */
   unsigned char     rsrvd2;                 /* Reservado           */
   AP_UINT16         primary_rc;
   AP_UINT32      secondary_rc;
   AP_UINT32         tps_id;
   AP_UINT32        res_id;
} UNREGISTER_TP;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_UNREGISTER_TP

tps_id Identificador del servidor de TP, tal como se ha devuelto en un verbo REGISTER_TP_SERVER anterior.

res_id Identificador exclusivo para este recurso, tal como se ha devuelto en el verbo REGISTER_TP anterior.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_TPS_ID
El valor especificado para el parámetro *tps_id* no se reconoce.

AP_BAD_RES_ID
El valor especificado para el parámetro *res_id* no se reconoce.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve el siguiente código de retorno primario. Para ver una lista de los códigos de retorno comunes a todos los verbos, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

QUERY_ATTACH

El verbo QUERY_ATTACH se utiliza para recuperar información sobre una petición Attach que Communications Server para Linux ha notificado a la aplicación. Este verbo es opcional. Si los datos representados por el correlacionador del servidor de TP pasado a la rutina callback de petición Attach son suficientes para su uso por parte del servidor de TP, el servidor de TP no necesita emitir este verbo.

Por razones de seguridad, la información de identificador de usuario y de contraseña de la petición Attach sólo está disponible para un servidor de TP cuyo identificador de usuario efectivo es `root`. Para las aplicaciones que no se ejecutan como `root`, a la petición Attach devuelta se le han eliminado los subcampos de seguridad de acceso.

Este verbo puede emitirse tantas veces como requiera el servidor de TP. Sin embargo, los datos PIP pueden extraerse sólo una vez. Para recuperar información de petición Attach sin recuperar los datos PIP, emita QUERY_ATTACH con *max_pip_len* definido en `0` (cero).

Estructura del VCB: QUERY_ATTACH

La definición de la estructura del VCB para el verbo QUERY_ATTACH es la siguiente:

```
typedef struct query_attach
{
   AP_UINT16         opcode;
   unsigned char     rsrvd1;                 /* Reservado            */
   unsigned char     rsrvd2;                 /* Reservado            */
   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;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode `AP_QUERY_ATTACH`

tps_id Identificador del servidor de TP, tal como se ha devuelto en un verbo REGISTER_TP_SERVER anterior.

attach_id
Identificador de la petición Attach, tal como se ha devuelto en la rutina callback de notificación de petición Attach.

max_pip_len
Espacio máximo de almacenamiento intermedio disponible para los datos PIP.

pip_dptr
Puntero al almacenamiento intermedio asignado por el emisor de la llamada para el almacenamiento intermedio de datos PIP de la petición Attach devuelta.

max_fmh5_len
Espacio máximo de almacenamiento intermedio disponible para los datos FMH5.

fmh5_dptr
Puntero al almacenamiento intermedio asignado por el emisor de la llamada para el almacenamiento intermedio de los datos FMH5 de la petición Attach devuelta.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve los parámetros siguientes:

primary_rc
`AP_OK`

tp_name
Nombre del TP de conexión.

lu_alias
Alias de LU local de la conexión.

fq_plu_name
Nombre de LU asociada completamente calificado de la conexión.

mode_name
Nombre de la modalidad de la conexión.

pip_dlen
Número real de bytes de los datos PIP devueltos.

fmh5_dlen
Número real de bytes de los datos FMH5 devueltos.

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_ATTACH_ID
El valor especificado para el parámetro *attach_id* no se reconoce.

AP_BAD_TPS_ID
El valor especificado para el parámetro *tps_id* no se reconoce.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve el siguiente código de retorno primario. Para ver una lista de los códigos de retorno comunes a todos los verbos, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

ACCEPT_ATTACH

El verbo ACCEPT_ATTACH se utiliza para que el servidor de TP continúe el proceso de la conexión.

Estructura del VCB: ACCEPT_ATTACH

La definición de la estructura del VCB para el verbo ACCEPT_ATTACH es la siguiente:

```
typedef struct accept_attach
{
   AP_UINT16         opcode;
   unsigned char     rsrvd1;                 /* Reservado            */
   unsigned char     rsrvd2;                 /* Reservado            */
   AP_UINT16         primary_rc;
   AP_UINT32      secondary_rc;
   AP_UINT32         tps_id;
   unsigned char    attach_id[8];
} ACCEPT_ATTACH;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_ACCEPT_ATTACH

tps_id Identificador del servidor de TP, tal como se ha devuelto en un verbo REGISTER_TP_SERVER anterior.

attach_id
Identificador de la petición Attach, tal como se ha devuelto en la rutina callback de notificación de petición Attach.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_ATTACH_ID
El valor especificado para el parámetro *attach_id* no se reconoce.

AP_BAD_TPS_ID
El valor especificado para el parámetro *tps_id* no se reconoce.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve el siguiente código de retorno primario. Para ver una lista de los códigos de retorno comunes a todos los verbos, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

REJECT_ATTACH

El verbo REJECT_ATTACH se utiliza para que este servidor de TP finalice el proceso de la conexión.

Estructura del VCB: REJECT_ATTACH

La definición de la estructura del VCB para el verbo REJECT_ATTACH es la siguiente:

```
typedef struct reject_attach
{
   AP_UINT16         opcode;
   unsigned char     rsrvd1;                 /* Reservado           */
   unsigned char     rsrvd2;                 /* Reservado           */
   AP_UINT16         primary_rc;
   AP_UINT32      secondary_rc;
   AP_UINT32         tps_id;
   unsigned char    attach_id[8];
   AP_UINT32         reason;
} REJECT_ATTACH;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_REJECT_ATTACH

tps_id Identificador del servidor de TP, tal como se ha devuelto en un verbo REGISTER_TP_SERVER anterior.

attach_id
Identificador de la petición Attach, tal como se ha devuelto en la rutina callback de notificación de petición Attach.

reason Motivo por el que se rechaza el inicio automático. El valor es un código de detección SNA, como el que se muestra en "Códigos de detección SNA".

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_ATTACH_ID
El valor especificado para el parámetro *attach_id* no se reconoce.

AP_BAD_TPS_ID
El valor especificado para el parámetro *tps_id* no se reconoce.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve el siguiente código de retorno primario. Para ver una lista de los códigos de retorno comunes a todos los verbos, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

Códigos de detección SNA: La Tabla 10 en la página 267 muestra los códigos de detección SNA comunes utilizados para rechazar una petición Attach:

Tabla 10. Códigos de detección SNA comunes

Símbolo	Valor	Significado
AP_SECURITY_INVALID	080F6051	Seguridad no válida.
AP_SEC_BAD_PASSWORD_EXPIRED	080FFF00	La contraseña ha caducado.
AP_SEC_BAD_PASSWORD_INVALID	080FFF01	La contraseña no es válida.
AP_SEC_BAD_USERID_REVOKED	080FFF02	El identificador de usuario ha sido revocado.
AP_SEC_BAD_USERID_INVALID	080FFF03	El identificador de usuario no es válido.
AP_SEC_BAD_USERID_MISSING	080FFF04	Falta el identificador de usuario.
AP_SEC_BAD_PASSWORD_MISSING	080FFF05	Falta la contraseña.
AP_SEC_BAD_GROUP_INVALID	080FFF06	El grupo no es válido.
AP_SEC_BAD_UID_REVOKED_IN_GRP	080FFF07	El identificador de usuario se revoca en el grupo especificado.
AP_SEC_BAD_UID_NOT_DEFD_TO_GRP	080FFF08	El identificador de usuario no está definido en el grupo especificado.
AP_SEC_BAD_UNAUTHRZD_AT_RLU	080FFF09	El identificador de usuario no está definido para utilizar la LU remota.
AP_SEC_BAD_UNAUTHRZD_FROM_LLU	080FFF0A	El identificador de usuario no está definido para utilizar la LU remota desde la LU local.
AP_SEC_BAD_UNAUTHRZD_TO_TP	080FFF0B	El identificador de usuario no está definido para utilizar el TP en la LU remota.
AP_SEC_BAD_INSTALL_EXIT_FAILED	080FFF0C	El proceso de salida de la instalación en la LU remota ha fallado.
AP_SEC_BAD_PROCESSING_FAILURE	080FFF0D	El proceso ha fallado entre la LU local y la LU remota, pero la condición es temporal.

Tabla 10. Códigos de detección SNA comunes (continuación)

Símbolo	Valor	Significado
AP_SEC_BAD_PROTOCOL_VIOLATION	080F6058	Una violación de protocolo ha producido una anomalía de validación de seguridad.
AP_TRANS_PGM_NOT_AVAIL_RETRY	084B6031	TP no disponible—reintento.
AP_TRANS_PGM_NOT_AVAIL_NO_RETRY	084C0000	TP no disponible—sin reintento.
AP_PIP_INVALID	1008201D	Datos PIP no válidos.
AP_ATTACH_LEN_INVALID	10086000	Longitud de conexión no válida.
AP_SECURITY_LEN_INVALID	10086005	Longitud de conexión no válida.
AP_PARM_LEN_INVALID	10086009	Longitud de parámetro no válida.
AP_LUWID_LEN_INVALID	10086011	Longitud LUWID no válida.
AP_TP_NAME_NOT_RECOGNIZED	10086021	Nombre de TP no reconocido.
AP_PIP_NOT_ALLOWED	10086031	Datos PIP no permitidos.
AP_PIP_FIELDS_REQUIRED	10086032	Datos PIP necesarios.
AP_CONVERSATION_TYPE_MISMATCH	10086034	Discrepancia de tipo de conversación.
AP_LU_CAPABILITY_CONFLICT	10086040	Las posibilidades de la LU de conexión entran en conflicto con el enlace.
AP_SYNC_LEVEL_NOT_SUPPORTED	10086041	Nivel de sincronización no soportado por el TP.

Nota: Communications Server para Linux puede sustituir el código de detección genérico AP_SECURITY_INVALID (080F651) por los códigos de detección del rango 080FFF00–080FFFFF si la LU remota no desea información de seguridad ampliada.

ABORT_ATTACH

El verbo ABORT_ATTACH se utiliza para que este servidor de TP finalice el proceso de la conexión después de que ésta haya sido aceptada con el verbo ACCEPT_ATTACH porque el servidor de TP o el TP ha detectado un error durante el proceso posterior. Por ejemplo, el servidor de TP no ha podido crear un subproceso para el TP. El verbo ABORT_ATTACH puede ser emitido por el servidor de TP y los procesos del TP.

Estructura del VCB: ABORT_ATTACH

La definición de la estructura del VCB para el verbo ABORT_ATTACH es la siguiente:

```
typedef struct abort_attach
{
   AP_UINT16         opcode;
   unsigned char     rsrvd1;                 /* Reservado           */
   unsigned char     rsrvd2;                 /* Reservado           */
   AP_UINT16         primary_rc;
   AP_UINT32      secondary_rc;
   AP_UINT32         tps_id;
   unsigned char    attach_id[8];
   AP_UINT32        reason;
} ABORT_ATTACH;
```

Parámetros suministrados

El TP suministra los siguientes parámetros a APPC:

opcode AP_ABORT_ATTACH

tps_id Identificador del servidor de TP, tal como se ha devuelto en un verbo REGISTER_TP_SERVER anterior.

attach_id
Identificador de la conexión que se debe terminar anormalmente, tal como lo devuelve la rutina callback de notificación de conexión.

reason Motivo por el que se termina anormalmente el inicio automático. El valor es un código de detección SNA, como el que se muestra en "Códigos de detección SNA" en la página 266.

Parámetros devueltos

Después de que el verbo se ejecute, APPC devuelve parámetros para indicar si la ejecución se ha llevado a cabo correctamente y, si no, para indicar el motivo por el que la ejecución no se ha llevado a cabo correctamente.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, APPC devuelve el parámetro siguiente:

primary_rc
AP_OK

Ejecución incorrecta

Si el verbo no se ejecuta de forma correcta, APPC devuelve un parámetro de código de retorno primario para indicar el tipo de error y un parámetro de código de retorno secundario para proporcionar datos específicos sobre el motivo de la ejecución incorrecta.

Comprobación de parámetros: Si el verbo no se ejecuta debido a un error de parámetro, APPC devuelve los parámetros siguientes:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Los valores posibles son:

AP_BAD_ATTACH_ID
El valor especificado para el parámetro *attach_id* no se reconoce.

AP_BAD_TPS_ID
El valor especificado para el parámetro *tps_id* no se reconoce.

Otras condiciones: Si el verbo no se ejecuta debido a que existen otras condiciones, APPC devuelve el siguiente código de retorno primario. Para ver una lista de los códigos de retorno comunes a todos los verbos, consulte el Apéndice B, "Códigos de retorno comunes", en la página 283.

primary_rc
AP_UNEXPECTED_SYSTEM_ERROR

Capítulo 6. Programas de transacciones de ejemplo

Los programas de transacciones (TP) APPC de ejemplo de Communications Server para Linux Linux ilustran el uso de los verbos APPC en una conversación correlacionada.

Los programas se suministran con Communications Server para Linux con los nombres **asample1.c** y **asample2.c**, situados en el directorio **/usr/lib/sna/samples** (AIX) o **/opt/ibm/sna/samples** (Linux).

En este capítulo se proporciona la siguiente información:

- Visión general del proceso de los TP de ejemplo.
- Pseudocódigo para cada TP.
- Instrucciones para compilar, enlazar y ejecutar los TP.

Visión general del proceso

Los TP presentados en este capítulo permiten al usuario examinar un archivo de otro sistema. El usuario ve un solo bloque de datos a la vez, en formato hexadecimal y de caracteres. Después de cada bloque, el usuario puede solicitar el siguiente bloque, el anterior o salir.

asample1 (el TP que invoca) envía un nombre de archivo a **asample2** (el TP invocado). Si **asample2** localiza el archivo, devuelve el primer bloque de datos a **asample1**; de lo contrario, desasigna la conversación y finaliza.

Si **asample1** recibe un bloque, visualiza el bloque en la pantalla y espera a que el usuario entre **F** para avanzar, **B** para retroceder o **Q** para salir. Si el usuario selecciona avanzar o retroceder, **asample1** envía la petición a **asample2**, que a su vez envía el bloque adecuado. Este proceso continúa hasta que el usuario selecciona la opción de salir, momento en que **asample1** desasigna la conversación y ambos programas finalizan.

Si el usuario solicita el siguiente bloque y **asample2** ha enviado el último, **asample2** vuelve al inicio del archivo. Igualmente, si el usuario solicita el bloque anterior y se está visualizando el primer bloque, **asample2** retrocede para enviar el último bloque.

Ningún programa intenta efectuar la recuperación de errores. Un código de retorno incorrecto de APPC hace que el programa finalice con un mensaje explicativo.

Pseudocódigo

Este apartado contiene el pseudocódigo para los TP **asample1** y **asample2**.

asample1 (TP que invoca)

El pseudocódigo para **asample1** (el TP que invoca) es el siguiente:

```
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 (TP invocado)

El pseudocódigo para **asample2** (el TP invocado) es el siguiente:

```
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
```

Cómo probar los TP

Tras examinar el código fuente de los dos programas, puede que desee probarlos.

Aunque APPC se utiliza normalmente para las comunicaciones entre una máquina local y otra remota, puede resultarle cómodo ejecutar ambos TP en la misma máquina de Communications Server para Linux para realizar la comprobación.

Para compilar y enlazar los TP, lleve a cabo los pasos siguientes.

1. Copie los dos archivos **asample1.c** y **asample2.c** del directorio **/opt/ibm/sna/samples** en un directorio privado.
2. Para compilar y enlazar los programas para AIX, utilice los mandatos siguientes:

```
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
```

   Para compilar y enlazar los programas para Linux, utilice los mandatos siguientes:

```
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
```

Para ejecutar los TP, siga los pasos que se describen a continuación. Tenga en cuenta que algunos de estos pasos suponen actualizar la configuración de Communications Server para Linux, lo que normalmente lleva a cabo el administrador del sistema.

Los TP pueden ejecutarse en la misma máquina o en máquinas diferentes. En los pasos siguientes, la "máquina de origen" es la máquina en que se ejecuta el TP que invoca **asample1** y la "máquina de destino" es la máquina en que se ejecuta el TP invocado **asample2**.

1. Si ejecuta los TP en máquinas diferentes, configure el enlace de comunicaciones para dar soporte a las sesiones CP-CP entre las máquinas de origen y de destino. Si desea obtener más información, consulte la publicación *Communications Server para Linux - Guía de administración*.
2. Configure una modalidad. Especifique LOCMODE como el nombre de la modalidad. Acepte los valores por omisión para el resto de parámetros.
3. Configure una unidad lógica (LU) en la máquina de origen para **asample1** (el TP que invoca). Establezca el nombre de LU y el alias de LU en **TPLU1** (el alias de LU especificado en el programa **asample1**). Acepte los valores por omisión para el resto de parámetros.
4. Si ejecuta los TP en máquinas diferentes, configure un alias de LU asociada en la máquina de origen para identificar la LU de destino. Establezca el nombre de LU asociada como *nombre_red***.TPLU2**, donde *nombre_red* es el nombre de red SNA de la máquina de destino.
5. Configure una LU en la máquina de destino para el TP invocado. Establezca el nombre de LU y el alias de LU en **TPLU2** (el alias que utiliza el programa **asample1** para hacer referencia a la LU que da servicio a **asample2**). Acepte los valores por omisión para el resto de parámetros.
6. Configure el TP invocado en el archivo de datos de TP invocable de Communications Server para Linux en la máquina de destino. Si desea obtener más información, consulte la publicación *Communications Server para Linux - Guía de administración*.
   - Para el parámetro *Nombre de TP*, especifique **TPNAME2** (nombre especificado por el TP que invoca).
   - Para *Vía de acceso completa al ejecutable del TP*, entre el nombre completo de la vía de acceso del archivo ejecutable **asample2**.
   - Para el parámetro *Identificador de usuario*, especifique su identificador de usuario de Linux en la máquina de destino.
   - Acepte los valores por omisión para otros parámetros.
7. Si el TP invocado va a ejecutarse con el parámetro *user_id* definido en `root`, cambie los permisos del archivo ejecutable para permitir esta posibilidad. Utilice el mandato siguiente:

   chmod +s asample2
8. Inicie el programa que invoca, **asample1**. Este programa requiere un parámetro, el nombre completo de vía de acceso (en la máquina de destino) del archivo que se visualizará. Por ejemplo:

   asample1 /usr/john/myfile.text
9. Entre **F** o **B** para visualizar bloques del archivo solicitado.
10. Utilice **Q** para finalizar el programa 1; el programa 2 también finalizará.

Apéndice A. Valores de código de retorno

Este apéndice lista todos los códigos de retorno posibles de la interfaz APPC en orden numérico. Los valores están definidos en el archivo de cabecera **values_c.h**(para AIX o Linux) o **winappc.h** (para Windows).

Puede utilizar este apéndice como referencia para verificar el significado de un código de retorno recibido por la aplicación.

Códigos de retorno primarios

En las aplicaciones APPC se utilizan los siguientes códigos de retorno primarios.

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

Códigos de retorno secundarios

En las aplicaciones APPC se utilizan los siguientes códigos de retorno secundarios.

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

Códigos de retorno secundarios

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

Apéndice B. Códigos de retorno comunes

En este apéndice se describen los códigos de retorno primarios (y, si corresponde, los códigos de retorno secundarios) que son comunes a varios verbos APPC.

Los códigos de retorno específicos de los verbos se describen en la documentación de los distintos verbos.

Los códigos de retorno comunes se describen en los siguientes apartados.

AP_ALLOCATION_ERROR

Los códigos de retorno primarios y secundarios son:

primary_rc

AP_ALLOCATION_ERROR
APPC no ha podido asignar una conversación. El estado de conversación está definido en Restablecer. Este código puede devolverse mediante un verbo emitido después de [MC_]ALLOCATE.

secondary_rc
Los valores posibles son:

AP_ALLOCATION_FAILURE_NO_RETRY
No puede asignarse la conversación debido a una condición permanente, como un error de configuración o un error de protocolo de sesión. Para determinar el error, el administrador del sistema debe examinar el archivo de anotaciones de error. No vuelva a intentar la asignación hasta que se haya corregido el error.

AP_ALLOCATION_FAILURE_RETRY
No se puede asignar la conversación debido a una condición temporal, como una anomalía de enlace. El motivo de la anomalía está anotado en el archivo de anotaciones de error del sistema. Vuelva a intentar la asignación, preferiblemente después de un tiempo de espera para dejar que la condición desaparezca.

AP_CONVERSATION_TYPE_MISMATCH
La LU asociada o el TP asociado no da soporte al tipo de conversación (básica o correlacionada) especificado en la petición de asignación.

AP_PIP_NOT_ALLOWED
La petición de asignación especifica datos PIP, pero el TP asociado no los acepta. Esto puede deberse a que el TP asociado no requiere estos datos, porque está utilizando una implementación de APPC que no da soporte a la recepción de datos PIP, o porque el elemento asociado es una aplicación CPI-C (CPI-C no da soporte a los datos PIP).

AP_PIP_NOT_SPECIFIED_CORRECTLY
El TP asociado requiere datos PIP, pero la petición de asignación indica que no hay datos PIP o que el número de parámetros es incorrecto.

AP_SECURITY_NOT_VALID
La LU asociada no acepta el identificador de usuario o la contraseña que se ha especificado en la petición de asignación.

AP_SYNC_LEVEL_NOT_SUPPORTED
El TP asociado no da soporte al parámetro *sync_level* (AP_NONE, AP_CONFIRM_SYNC_LEVEL o AP_SYNCPT) especificado en la petición de asignación, o no se reconoce el parámetro *sync_level*.

AP_TP_NAME_NOT_RECOGNIZED
La LU asociada no reconoce el nombre de TP especificado en la petición de asignación.

AP_TRANS_PGM_NOT_AVAIL_NO_RETRY
La LU remota ha rechazado una petición de asignación porque no ha podido iniciar el TP asociado solicitado. La condición es permanente. El motivo del error puede estar anotado en el nodo remoto. No vuelva a intentar la asignación hasta que se haya corregido el motivo del error.

AP_TRANS_PGM_NOT_AVAIL_RETRY
La LU remota ha rechazado una petición de asignación porque no ha podido iniciar el TP asociado solicitado. La condición puede ser temporal, como un tiempo de espera excedido. El motivo del error puede estar anotado en el nodo remoto. Vuelva a intentar la asignación, preferiblemente después de un tiempo de espera para dejar que la condición desaparezca.

AP_SEC_BAD_PROTOCOL_VIOLATION
La LU remota ha rechazado la petición de asignación debido a una violación de protocolo.

AP_SEC_BAD_PASSWORD_EXPIRED
La LU remota ha rechazado la petición de asignación porque la contraseña proporcionada ha dejado de ser válida.

AP_SEC_BAD_PASSWORD_INVALID
La LU remota ha rechazado la petición de asignación porque la contraseña no es válida.

AP_SEC_BAD_USERID_REVOKED
La LU remota ha rechazado la petición de asignación porque el identificador de usuario ha dejado de ser válido.

AP_SEC_BAD_USERID_INVALID
La LU remota ha rechazado la petición de asignación porque el identificador de usuario no es válido.

AP_SEC_BAD_USERID_MISSING
La LU remota ha rechazado la petición de asignación porque no se ha especificado ningún identificador de usuario, y éste es necesario.

AP_SEC_BAD_PASSWORD_MISSING
La LU remota ha rechazado la petición de asignación porque no se ha especificado ninguna contraseña, y ésta es necesaria.

AP_SEC_BAD_GROUP_INVALID
La LU remota ha rechazado la petición de asignación porque el grupo no es válido.

AP_SEC_BAD_UID_REVOKED_IN_GRP
La LU remota ha rechazado la petición de asignación porque el identificador de usuario ya no está en el grupo.

AP_SEC_BAD_UID_NOT_DEFD_TO_GRP
La LU remota ha rechazado la petición de asignación porque el identificador de usuario no está en el grupo.

AP_SEC_BAD_UNAUTHRZD_AT_RLU
La LU remota ha rechazado la petición de asignación porque el identificador de usuario no está autorizado para iniciar este TP en la LU remota.

AP_SEC_BAD_UNAUTHRZD_FROM_LLU
La LU remota ha rechazado la petición de asignación porque el identificador de usuario no está autorizado para iniciar este TP desde la LU local.

AP_SEC_BAD_UNAUTHRZD_TO_TP
La LU remota ha rechazado la petición de asignación porque el identificador de usuario no está autorizado para iniciar este TP.

AP_SEC_BAD_INSTALL_EXIT_FAILED
La LU remota ha rechazado la petición de asignación porque no ha podido instalar una salida necesaria.

AP_SEC_BAD_PROCESSING_FAILURE
La LU remota ha rechazado la petición de asignación a causa de una anomalía de proceso en la LU remota.

Puesto que proporcionar información detallada sobre las anomalías de seguridad constituye un riesgo potencial para la seguridad, se puede desactivar el soporte para estos códigos de retorno `AP_SEC_BAD_*`. En ese caso, se informa a la aplicación de todos estos errores como `AP_SECURITY_NOT_VALID`. Para conocer más detalles, consulte la información sobre el mandato **define_defaults** en la publicación *Communications Server for Linux Administration Command Reference* y la información sobre el verbo DEFINE_DEFAULTS NOF en la publicación *Communications Server for Linux NOF Programmer's Guide*.

AP_BACKED_OUT

AIX, LINUX

Los códigos de retorno primarios y secundarios son:

primary_rc

AP_BACKED_OUT
El TP asociado (u otro TP que participa en la misma unidad lógica de trabajo) ha emitido una petición de restitución. El gestor de puntos de sincronización se encarga de ejecutar el proceso de punto de sincronización apropiado, según el código de retorno secundario, que es uno de los siguientes:

secondary_rc
Los valores posibles son:

AP_BO_NO_RESYNC
El TP asociado ha completado la restitución de sus recursos.

AP_BO_RESYNC
Se ha producido una anomalía mientras el TP asociado intentaba restituir sus recursos; la resincronización todavía se está ejecutando.

AP_CANCELLED

WINDOWS

El código de retorno primario es:

primary_rc

AP_CANCELLED
El verbo se ha emitido utilizando el punto de entrada `WinAsyncAPPC` y posteriormente se ha cancelado utilizando el punto de entrada `WinAPPCCancel`. Para ver más información sobre estos puntos de entrada, consulte el "Puntos de entrada APPC: Sistemas Windows" en la página 38.

No se devuelve ningún código de retorno secundario.

AP_COMM_SUBSYSTEM_ABENDED

El código de retorno primario es:

primary_rc

AP_COMM_SUBSYSTEM_ABENDED
El código de retorno indica que el software Communications Server para Linux ha finalizado anormalmente o que hay un problema con la LAN. El administrador del sistema debe examinar el archivo de anotaciones de error para determinar el motivo de la finalización anormal.

No se devuelve ningún código de retorno secundario.

AP_COMM_SUBSYSTEM_NOT_LOADED

El código de retorno primario es:

primary_rc

AP_COMM_SUBSYSTEM_NOT_LOADED
Este código de retorno indica que no puede aceptarse un intento de iniciar un TP utilizando el verbo TP_STARTED o RECEIVE_ALLOCATE a causa de una de las siguientes condiciones.

AIX, LINUX

- No se ha cargado el software de Communications Server para Linux o no se ha iniciado el nodo local que posee la LU

utilizada por este TP. Póngase en contacto con el administrador del sistema para corregir el error.

- El número máximo de usuarios permitidos por la licencia de Communications Server para Linux ya están utilizando Communications Server para Linux. En estos momentos no puede utilizar este TP, porque se sobrepasaría el límite de número de usuarios; puede iniciarlo más tarde, cuando haya menos usuarios en el sistema.

WINDOWS

- No se ha cargado el software de Communications Server para Linux o un componente de comunicaciones que utiliza la LU APPC que se ha especificado está inactivo. Póngase en contacto con el administrador del sistema para corregir el error.
- El alias de LU especificado en un verbo TP_STARTED no se ha reconocido. Compruebe que el alias de LU especificado coincida con un alias de LU local APPC en el archivo de configuración.
- El nodo local al que pertenece la LU local APPC que se está utilizando ya lo está utilizando el número máximo de usuarios de Communications Server para Linux que permite la licencia de Communications Server para Linux. En estos momentos no puede utilizar este TP, porque se sobrepasaría el límite de número de usuarios; puede iniciarlo más tarde, cuando haya menos usuarios en el sistema.

No se devuelve ningún código de retorno secundario.

AP_CONV_FAILURE_NO_RETRY

El código de retorno primario es:

primary_rc

AP_CONV_FAILURE_NO_RETRY
La conversación se ha finalizado a causa de una condición permanente, como un error de protocolo de sesión. El administrador del sistema debe examinar el archivo de anotaciones de error del sistema para determinar el motivo del error. No vuelva a intentar establecer la conversación hasta que se haya corregido el error.

No se devuelve ningún código de retorno secundario.

AP_CONV_FAILURE_RETRY

El código de retorno primario es:

primary_rc

AP_CONV_FAILURE_RETRY
La conversación se ha finalizado a causa de un error temporal. Vuelva a iniciar el TP para ver si se repite el problema. Si es así, el

administrador del sistema debe examinar el archivo de anotaciones de error para determinar el motivo del error.

No se devuelve ningún código de retorno secundario.

AP_CONVERSATION_TYPE_MIXED

El código de retorno primario es:

primary_rc

AP_CONVERSATION_TYPE_MIXED
El TP ha emitido verbos básicos y correlacionados. Sólo puede emitirse un tipo en una sola conversación.

No se devuelve ningún código de retorno secundario.

AP_DEALLOC_ABEND

El código de retorno primario es:

primary_rc

AP_DEALLOC_ABEND
La conversación se ha desasignado por uno de los siguientes motivos:

- El TP asociado ha emitido el verbo MC_DEALLOCATE con el *dealloc_type* definido en AP_ABEND.
- El TP asociado ha finalizado anormalmente, lo que ha hecho que la LU asociada envíe una petición MC_DEALLOCATE.

No se devuelve ningún código de retorno secundario.

AP_DEALLOC_ABEND_PROG

El código de retorno primario es:

primary_rc

AP_DEALLOC_ABEND_PROG
La conversación se ha desasignado por uno de los siguientes motivos:

- El TP asociado ha emitido el verbo DEALLOCATE con *dealloc_type* definido en AP_ABEND_PROG.
- El TP asociado ha finalizado anormalmente, lo que ha hecho que la LU asociada envíe una petición DEALLOCATE.

No se devuelve ningún código de retorno secundario.

AP_DEALLOC_ABEND_SVC

El código de retorno primario es:

primary_rc

AP_DEALLOC_ABEND_SVC
La conversación se ha desasignado porque el TP asociado ha emitido el verbo DEALLOCATE con *dealloc_type* definido en AP_ABEND_SVC.

No se devuelve ningún código de retorno secundario.

AP_DEALLOC_ABEND_TIMER

El código de retorno primario es:

primary_rc

AP_DEALLOC_ABEND_TIMER
La conversación se ha desasignado porque el TP asociado ha emitido el verbo DEALLOCATE con *dealloc_type* definido en AP_ABEND_TIMER.

No se devuelve ningún código de retorno secundario.

AP_DEALLOC_NORMAL

El código de retorno primario es:

primary_rc

AP_DEALLOC_NORMAL
Este código de retorno no indica un error.

El TP asociado ha emitido el verbo [MC_]DEALLOCATE con *dealloc_type* con uno de los siguientes valores:

- AP_FLUSH
- AP_SYNC_LEVEL con el nivel de sincronización de la conversación especificado como AP_NONE.

No se devuelve ningún código de retorno secundario.

AP_DUPLEX_TYPE_MIXED

El código de retorno primario es:

primary_rc

AP_DUPLEX_TYPE_MIXED
El TP ha emitido un verbo de conversación con un tipo de dúplex que no coincide con el de la conversación. Si la conversación es dúplex (tal como especifica el parámetro *duplex_type* de [MC_]ALLOCATE or RECEIVE_ALLOCATE), el TP deberá establecer la opción AP_FULL_DUPLEX_CONVERSATION en el parámetro *opext* de todos los demás verbos de la conversación. Si la conversación es semidúplex, no deberá establecer esta opción.

No se devuelve ningún código de retorno secundario.

AP_INVALID_VERB

El código de retorno primario es:

primary_rc

AP_INVALID_VERB
El código de operación proporcionado para el verbo no es válido. El verbo no se ha ejecutado.

Este código de retorno también se devuelve si intenta emitir el verbo [MC_]RECEIVE_AND_POST en una conversación dúplex. [MC_]RECEIVE_AND_POST sólo se puede utilizar en una conversación semidúplex.

No se devuelve ningún código de retorno secundario.

AP_INVALID_VERB_SEGMENT

WINDOWS

El código de retorno primario es:

primary_rc

AP_INVALID_VERB_SEGMENT
El bloque de control del verbo se ha extendido más allá del final de un segmento de datos. El verbo no se ha ejecutado.

No se devuelve ningún código de retorno secundario.

AP_PROG_ERROR_NO_TRUNC

El código de retorno primario es:

primary_rc

AP_PROG_ERROR_NO_TRUNC
El TP asociado ha emitido uno de los siguientes verbos mientras la conversación estaba en estado Enviar:

- SEND_ERROR con *err_type* definido en AP_PROG.
- MC_SEND_ERROR

  Los datos no se han truncado.

No se devuelve ningún código de retorno secundario.

AP_PROG_ERROR_PURGING

El código de retorno primario es:

primary_rc

AP_PROG_ERROR_PURGING
El TP asociado ha emitido uno de los siguientes verbos:

- SEND_ERROR con *err_type* definido en AP_PROG.
- MC_SEND_ERROR

  mientras la conversación estaba en estado Recibir, Pendiente-Envío, Confirmar, Confirmar-Enviar o Confirmar-Desasignar. Se eliminan los datos que se han enviado pero que aún no se han recibido.

No se devuelve ningún código de retorno secundario.

AP_PROG_ERROR_TRUNC

El código de retorno primario es:

primary_rc

AP_PROG_ERROR_TRUNC
En estado Enviar, después de enviar un registro lógico incompleto, el TP asociado ha emitido un verbo SEND_ERROR con *err_type* definido en AP_PROG. El TP local puede haber recibido la primera parte del registro lógico mediante un verbo de recepción.

No se devuelve ningún código de retorno secundario.

AP_SVC_ERROR_NO_TRUNC

El código de retorno primario es:

primary_rc

AP_SVC_ERROR_NO_TRUNC
Mientras estaba en estado Enviar, el TP asociado (o la LU asociada) ha emitido un verbo SEND_ERROR con *err_type* definido en AP_SVC. Los datos no se han truncado.

No se devuelve ningún código de retorno secundario.

AP_SVC_ERROR_PURGING

El código de retorno primario es:

primary_rc

AP_SVC_ERROR_PURGING
El TP asociado (o la LU asociada) ha emitido un verbo SEND_ERROR con *err_type* definido en AP_SVC mientras estaba en estado Recibir, Pendiente-Envío, Confirmar, Confirmar-Enviar o Confirmar-Desasignar. Los datos enviados al TP asociado pueden haber sido eliminados.

No se devuelve ningún código de retorno secundario.

AP_SVC_ERROR_TRUNC

El código de retorno primario es:

primary_rc

AP_SVC_ERROR_TRUNC
En estado Enviar, después de enviar un registro lógico incompleto, el TP asociado (o la LU asociada) ha emitido un verbo SEND_ERROR. El TP local puede haber recibido la primera parte del registro lógico.

No se devuelve ningún código de retorno secundario.

AP_THREAD_BLOCKING

WINDOWS

El código de retorno primario es:

primary_rc

AP_THREAD_BLOCKING
El verbo se ha emitido utilizando el punto de entrada APPC (de bloqueo), pero ya había pendiente otro verbo APPC de bloqueo. Para ver más información sobre estos puntos de entrada, consulte el “Puntos de entrada APPC: Sistemas Windows” en la página 38.

No se devuelve ningún código de retorno secundario.

AP_TP_BUSY

El código de retorno primario es:

primary_rc

AP_TP_BUSY
El TP local ha emitido una llamada a APPC mientras APPC estaba procesando otra llamada para el mismo TP. Esto puede ocurrir si el TP local ha iniciado varios procesos y más de uno emite llamadas APPC utilizando el mismo *tp_id*. Sin embargo, debe asegurarse de que cada proceso emita su propio verbo TP_STARTED o RECEIVE_ALLOCATE para obtener su propio *tp_id*; las consecuencias del uso del mismo *tp_id* por varios procesos son impredecibles.

WINDOWS

Este código de retorno también puede indicar que la aplicación que emite el verbo se ha invocado utilizando la función SendMessage de Windows en lugar de la función PostMessage; la aplicación no puede emitir ningún verbo en este estado. Si desea ver más información, consulte “Consideraciones sobre Windows” en la página 55.

No se devuelve ningún código de retorno secundario.

AP_UNEXPECTED_SYSTEM_ERROR

El código de retorno primario es:

primary_rc

AP_UNEXPECTED_SYSTEM_ERROR
El sistema operativo ha detectado un error al procesar una llamada

APPC del TP local. El código de retorno del sistema operativo se devuelve mediante *secondary_rc*. Si el problema persiste, consulte a su administrador del sistema.

AIX, LINUX

Para ver el significado del código de retorno del sistema operativo, consulte el archivo **/usr/include/errno.h** en la máquina en que se ha producido el error.

WINDOWS

Para ver el significado del código de retorno del sistema operativo, consulte la documentación del sistema operativo.

No se devuelve ningún código de retorno secundario.

Apéndice C. Cambios de estado APPC

Las tablas siguientes muestran los estados de conversación en que puede emitirse cada uno de los verbos APPC y el cambio de estado que se produce cuando el verbo finaliza. En algunos casos, el cambio de estado depende del parámetro *primary_rc* devuelto al verbo; en este caso, los valores de *primary_rc* aplicables se muestran en la misma columna que el verbo. Cuando no se muestra ningún valor de *primary_rc*, los cambios de estado son los mismos para todos los códigos de retorno, excepto en los casos que se indican en las notas que hay después de cada tabla.

Los estados de conversación posibles se muestran como cabeceras de columna. Para cada combinación de verbo y valor de *primary_rc*, se proporcionan las siguientes abreviaturas y símbolos bajo cada estado para indicar los resultados de emitir el verbo en ese estado:

X El verbo no puede emitirse en este estado.

T, E, R, ...
Estado de la conversación una vez que el verbo ha finalizado.

Para conversaciones semidúplex:

T	Restablecer
E	Enviar
EP	Enviar-Pendiente
R	Recibir
C	Confirmar
CE	Confirmar-Enviar
CD	Confirmar-Desasignar
PE	Pendiente-Envío

Para conversaciones dúplex:

T	Restablecer
ER	Enviar-Recibir
E	Sólo-Enviar
R	Sólo-recibir

- No hay ningún estado de conversación después de emitirse el verbo.

/ No es válido para considerar el estado anterior, porque el verbo inicia una conversación nueva como si lo hiciera desde el estado Restablecer; no hay ningún efecto sobre ninguna conversación existente.

(blanco)
El código de retorno mostrado no puede producirse en este estado.

Conversaciones semidúplex

Verbo y valores de *primary_rc*	**Estado en que se emite**							
	Resta-blecer (T)	**Enviar (E)**	**Enviar Pend. (EP)**	**Rec. (R)**	**Conf. (C)**	**Conf. Enviar (CE)**	**Conf. Des. (CD)**	**Pend. Env. (PE)**
TP_STARTED								
AP_OK	T	/	/	/	/	/	/	/
otros valores de *primary_rc*	-							
TP_ENDED								
AP_OK	-	-	-	-	-	-	-	-
otros valores de *primary_rc*	T	E	EP	R	C	CE	CD	PE
RECEIVE_ALLOCATE								
AP_OK	R	/	/	/	/	/	/	/
otros valores de *primary_rc*	-							
GET_LU_STATUS	X	E	EP	R	C	CE	CD	PE
GET_TP_PROPERTIES	T	E	EP	R	C	CE	CD	PE
SET_TP_PROPERTIES	T	E	EP	R	C	CE	CD	PE
GET_TYPE	X	E	EP	R	C	CE	CD	PE
[MC_]ALLOCATE								
AP_OK	E	/	/	/	/	/	/	/
otros valores de *primary_rc*	T							
[MC_]CONFIRM								
AP_OK	X	E	E	X	X	X	X	X
AP_ERROR		R	R					
[MC_]CONFIRMED	X	X	X	X	R	E	T	X
[MC_]DEALLOCATE								
valores de *dealloc_type* AP_ABEND_*	X	T	T	T	T	T	T	T
otros valores de *dealloc_type*								
AP_ERROR	X	R	R	X	X	X	X	X
otros valores de *primary_rc*	T	T						
[MC_]FLUSH	X	E	E	X	X	X	X	X
[MC_]GET_ ATTRIBUTES	X	E	EP	R	C	CE	CD	PE
[MC_]PREPARE_TO_RECEIVE	X	R	R	X	X	X	X	X
[MC_]RECEIVE_AND_POST (Veáse la nota 4)	X	PE	PE	PE	X	X	X	X
[MC_]RECEIVE_AND_WAIT	X	Veáse la nota 5	Veáse la nota 5	Veáse la nota 5	X	X	X	X
[MC_]RECEIVE_IMMEDIATE	X	X	X	Veáse la nota 5	X	X	X	X

Verbo y valores de *primary_rc*	**Estado en que se emite**							
	Resta-blecer (T)	**Enviar (E)**	**Enviar Pend. (EP)**	**Rec. (R)**	**Conf. (C)**	**Conf. Enviar (CE)**	**Conf. Des. (CD)**	**Pend. Env. (PE)**
[MC_]RECEIVE_EXPEDITED_DATA	X	X	X	R	C	X	X	PE
[MC_]REQUEST_TO_SEND	X	X	X	R	C	X	X	PE
[MC_]SEND_CONVERSATION	T	/	/	/	/	/	/	/
[MC_]SEND_DATA AP_OK AP_ERROR	X	E R	E	X	X	X	X	X
[MC_]SEND_ERROR AP_OK AP_ERROR	X	E R	E	E	E	E	E	E
[MC_]SEND_EXPEDITED_DATA	X	X	X	R	C	X	X	PE
[MC_]TEST_RTS	X	E	E	R	C	CE	CD	PE
[MC_]TEST_RTS_AND_POST	X	E	E	R	C	CE	CD	PE

Nota:

1. En la columna de códigos de retorno de la tabla, la abreviatura AP_ERROR se utiliza para:
   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. La conversación siempre entra en estado Restablecer si se recibe uno de los códigos de retorno siguientes.
   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. Los siguientes códigos de retorno de ejecución incorrecta no provocan ningún cambio de estado. La conversación siempre permanece en el estado en que se ha emitido el verbo.
   AP_CONVERSATION_TYPE_MIXED
   AP_INVALID_VERB
   AP_PARAMETER_CHECK
   AP_STATE_CHECK
   AP_TP_BUSY
   AP_UNEXPECTED_SYSTEM_ERROR

AP_UNSUCCESSFUL

4. Después de emitirse [MC_]RECEIVE_AND_POST y de recibirse el valor de *primary_rc* AP_OK inicial, la conversación cambia al estado Pendiente-Envío. Una vez que se ha llamado a la rutina callback suministrada, para indicar que el verbo ha finalizado, el nuevo estado de conversación depende de los parámetros *primary_rc* y *what_rcvd*, como se indica en la nota 5.
5. El cambio de estado después de uno de los verbos RECEIVE depende de los parámetros *primary_rc* y *what_rcvd*.

   Si el parámetro *primary_rc* es AP_PROG_ERROR, AP_SVC_ERROR o (sólo para [MC_]RECEIVE_IMMEDIATE) AP_UNSUCCESSFUL, el nuevo estado es Recibir.

   Si el parámetro *primary_rc* es AP_DEALLOC, el nuevo estado es Restablecer.

   Si el parámetro *primary_rc* es AP_OK, el nuevo estado depende del valor del parámetro *what_rcvd*:

   AP_DATA, AP_DATA_COMPLETE, AP_DATA_INCOMPLETE
   Estado Recibir

   AP_SEND
   Estado Enviar

   AP_DATA_SEND, AP_DATA_COMPLETE_SEND
   Estado Enviar-Pendiente.

   AP_CONFIRM_WHAT_RCVD, AP_DATA_CONFIRM, AP_DATA_COMPLETE_CONFIRM
   Estado Confirmar

   AP_CONFIRM_SEND, AP_DATA_CONFIRM_SEND, AP_DATA_COMPLETE_CONFIRM_SEND
   Estado Confirmar-Enviar.

   AP_CONFIRM_DEALLOCATE, AP_DATA_CONFIRM_DEALLOCATE, AP_DATA_COMPLETE_CONFIRM_DEALL
   Estado Confirmar-Desasignar.

Conversaciones dúplex

Verbo y valores de *primary_rc*	**Estado en que se emite**			
	Restablecer (T)	**Enviar Recibir (ER)**	**Sólo Enviar (E)**	**Sólo Recibir (R)**
TP_STARTED				
AP_OK	T	/	/	/
otros valores de *primary_rc*	-			
TP_ENDED				
AP_OK	-	-	-	-
otros valores de *primary_rc*	T	ER	E	R
RECEIVE_ALLOCATE				
AP_OK	ER	/	/	/
otros valores de *primary_rc*	-			
GET_LU_STATUS	X	ER	E	R

Verbo y valores de *primary_rc*	**Estado en que se emite**			
	Restablecer (T)	**Enviar Recibir (ER)**	**Sólo Enviar (E)**	**Sólo Recibir (R)**
GET_TP_PROPERTIES	T	ER	E	R
SET_TP_PROPERTIES	T	ER	E	R
GET_TYPE	X	ER	E	R
[MC_]ALLOCATE				
AP_OK	ER	/	/	/
otros valores de *primary_rc*	T			
[MC_]DEALLOCATE				
valores de *dealloc_type* AP_ABEND_*	X	T	T	T
otros valores de *dealloc_type*	X	R	T	X
[MC_]FLUSH	X	ER	E	X
[MC_]GET_ ATTRIBUTES	X	ER	E	R
[MC_]RECEIVE_AND_WAIT				
AP_OK	X	ER	X	R
AP_ERROR	X	ER	X	R
AP_DEALLOC_NORMAL	X	E	X	T
[MC_]RECEIVE_IMMEDIATE				
AP_OK	X	ER	X	R
AP_ERROR	X	ER	X	R
AP_DEALLOC_NORMAL	X	E	X	T
[MC_]RECEIVE_ EXPEDITED_DATA	X	ER	E	R
[MC_]SEND_DATA				
AP_OK	X	ER	E	X
AP_ERROR	X	ER	T	X
[MC_]SEND_ERROR				
AP_OK	X	ER	E	X
AP_ERROR	X	ER	T	X
[MC_]SEND_ EXPEDITED_DATA	X	ER	E	R

Nota:

1. En la columna de códigos de retorno de la tabla, la abreviatura AP_ERROR se utiliza para:
   AP_BACKED_OUT
   AP_PROG_ERROR_TRUNC
   AP_PROG_ERROR_NO_TRUNC
   AP_SVC_ERROR_TRUNC
   AP_SVC_ERROR_NO_TRUNC
2. La conversación siempre entra en estado Restablecer si se recibe uno de los códigos de retorno siguientes.
   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. Los siguientes códigos de retorno de ejecución incorrecta no provocan ningún cambio de estado. La conversación siempre permanece en el estado en que se ha emitido el verbo.
   AP_CONVERSATION_TYPE_MIXED
   AP_INVALID_VERB
   AP_PARAMETER_CHECK
   AP_STATE_CHECK
   AP_TP_BUSY
   AP_UNEXPECTED_SYSTEM_ERROR
   AP_UNSUCCESSFUL

Apéndice D. Soporte de LU 6.2 SNA

Este apéndice explica en detalle cómo se relaciona la implementación de APPC de Communications Server para Linux con la arquitectura de LU 6.2. Incluye la siguiente información:

- Un resumen de los conjuntos de opciones de LU 6.2 soportados por Communications Server para Linux.
- Una lista de los verbos de operador de control que se incluyen en la implementación de APPC de Communications Server para Linux.
- Una lista de los verbos de operador de control cuyas funciones llevan a cabo en Communications Server para Linux las herramientas de administración o la API de NOF.

Soporte de conjuntos de opciones de LU 6.2

APPC de Communications Server para Linux da soporte al conjunto base de funciones de LU 6.2 y a algunos conjuntos de opciones. Algunos de estos conjuntos de opciones están soportados por verbos APPC y otros están soportados por las herramientas de administración o por la API de NOF.

Las siguientes tablas contienen los conjuntos de opciones soportados por Communications Server para Linux, con el número de referencia del conjunto de opciones especificado en la publicación *Transaction Programmer's Reference Manual for LU Type 6.2* de IBM. (Las versiones anteriores de este manual de IBM utilizan números de referencia diferentes).

Conjuntos de opciones de LU 6.2 soportados por verbos APPC

Número de referencia	Conjunto de opciones
101	Vaciar el almacenamiento intermedio de envío de la LU.
102	GET_ATTRIBUTES
103	POST_ON_RECEIPT con comprobación para envío. *
104	POST_ON_RECEIPT con espera. *
105	PREPARE_TO_RECEIVE
106	RECEIVE_IMMEDIATE
109	Obtener nombre de TP e identificador de instancia.
110	GET_CONVERSATION_TYPE.
112	Conversaciones dúplex y datos acelerados
113	Soporte de no bloqueo
201	Asignación en cola de una sesión ganadora de contienda.
203	Asignación inmediata de una sesión.
204	Conversaciones entre programas situados en la misma LU.
205	Asignación en cola en espera de una sesión libre.
211	Verificación de LU-LU a nivel de sesión.
212	Verificación de identificador de usuario.
213	Identificador de usuario y contraseña suministrados por el programa.
214	Autorización de identificador de usuario.
241	Envío de datos PIP.

Número de referencia	Conjunto de opciones
242	Recepción de datos PIP.
243	Contabilidad.
244	Bloqueos largos.
245	Comprobación de recepción de REQUEST_TO_SEND.
247	Datos de control de usuario.
290	Anotación de datos en un archivo de anotaciones del sistema.
291	Componente de servicios de LU de conversación correlacionada.
401	Brackets de un solo sentido fiables.
616	Soporte de nombre de modalidad CPSVCMG.

*Las opciones 103 y 104 están soportadas por el verbo [MC_]RECEIVE_AND_POST.

Conjuntos de opciones de LU 6.2 soportados por las herramientas de administración y por la API de NOF

Número de referencia	Conjunto de opciones
501	CHANGE_SESSION_LIMIT.
502	ACTIVATE_SESSION
504	DEACTIVATE_SESSION
505	Verbos de definición de LU.
601	Parámetro *min_conwinners_target*.
602	Parámetro *responsible* (TARGET).
603	Parámetro *drain_target* (NO).
604	Parámetro *force*.
605	Límite de sesiones LU-LU.
606	Nombres de LU localmente conocidos.
607	Nombres de LU sin interpretar.
610	Límites de tamaño máximo de RU.
611	Cifrado obligatorio a nivel de sesión.
612	Límite de activación automática del ganador de la contienda.
613	Límite máximo de sesiones locales (LU, modalidad).

Soporte de verbos de operador de control

Las funciones de los siguientes verbos de operador de control se proporcionan como parte de la implementación de APPC de Communications Server para Linux:

RECEIVE_ALLOCATE
TP_STARTED
TP_ENDED

Los programas de administración de Communications Server para Linux y la API de NOF proporcionan las funciones de los siguientes verbos de operador de control.

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

Soporte de verbos de operador de control

Apéndice E. Avisos

Esta información ha sido desarrollada para los productos y servicios que se ofrecen en los EE.UU. Es posible que IBM no ofrezca los productos, servicios o funciones que se tratan en este documento en otros países. Póngase en contacto con el representante de IBM de su zona para obtener información sobre los productos y servicios que actualmente se encuentran disponibles en su región. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que solamente sea posible utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ningún derecho de propiedad de intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de todo producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patente en tramitación que abarquen los temas descritos en este documento. La posesión de este documento no le concede ningún tipo de licencia sobre dichas patentes. Puede enviar sus consultas relativas a las licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para cualquier consulta sobre licencias relacionada con la información sobre DBCS (juego de caracteres de doble byte), póngase en contacto con el departamento de propiedad intelectual de IBM de su país o envíe su consulta por escrito a la siguiente dirección:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún otro país en el que estas disposiciones sean contrarias a la legislación del país:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN “TAL CUAL” SIN GARANTÍA DE NINGÚN TIPO, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIABILIDAD O ADECUACIÓN PARA UN FIN DETERMINADO. Algunos estados no permiten la renuncia de garantías expresas ni implícitas en determinadas transacciones, por lo que esta declaración puede no ser aplicable a su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en esta información; tales modificaciones se incorporarán en nuevas ediciones de la publicación. IBM puede realizar mejoras y cambios en los productos y programas descritos en esta publicación en todo momento, sin previo aviso.

Las referencias hechas en esta publicación a sitios Web que no sean de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de

dichos sitios Web. La información contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de dichos sitios Web.

IBM puede utilizar o distribuir cualquier información que se le facilite del modo que IBM considere oportuno, sin incurrir por ello en ninguna obligación con el remitente.

Los poseedores de licencias de este programa que deseen disponer de información acerca del mismo con la finalidad de permitir: (i) el intercambio de información entre programas desarrollados independientemente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Corporation
P.O. Box 12195
3039 Cornwallis Road
Research Triangle Park, NC 27709-2195
EE.UU.

Dicha información puede estar disponible, sujeta a los términos y condiciones pertinentes, lo que puede incluir, en algunos casos, el pago de una cuota.

IBM proporciona el programa bajo licencia que se describe en esta información y todo el material bajo licencia disponible para él conforme a los términos del contrato de cliente de IBM, el acuerdo internacional de licencia de programas de IBM o cualquier acuerdo equivalente entre las partes.

Los datos de rendimiento contenidos en este documento se han obtenido en un entorno controlado. Por lo tanto, los resultados que se obtengan en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información relacionada con productos que no sean de IBM se ha obtenido de los proveedores de dichos productos, de sus anuncios publicados o de otras fuentes de información públicamente disponibles. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni ninguna otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de esos productos.

Este manual contiene ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

LICENCIA DE COPYRIGHT: Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que muestran técnicas de programación en varias plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin pago alguno a IBM, con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa

para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas. El usuario puede copiar, modificar y distribuir estos programas de ejemplo en cualquier forma, sin pago alguno a IBM, con el fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación que se ajustan a las interfaces de programación de aplicaciones de IBM.

Cada copia total o parcial de estos programas de ejemplo o cualquier trabajo derivado deben incluir un aviso de copyright como el siguiente: ® (nombre de la empresa) (año). Partes de este código proceden de programas de ejemplo de IBM Corp. ® Copyright IBM Corp. 2000, 2005, 2006. Reservados todos los derechos.

Marcas registradas

Los términos siguientes son marcas registradas de IBM Corporation en Estados Unidos y/o en otros países:

AIX
IBM
OS/2
pSeries
z/OS

Los términos siguientes son marcas registradas de otras empresas:

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc., en Estados Unidos y/o en otros países.

UNIX es una marca registrada en Estados Unidos y en otros países cuyas licencias concede exclusivamente The Open Group.

Intel y EM64T son marcas registradas de Intel Corporation.

AMD64 es una marca registrada de Advanced Micro Devices, Inc.

Linux es una marca registrada de Linus Torvalds.

RedHat y RPM son marcas registradas de Red Hat, Inc.

SuSE Linux es una marca registrada de Novell.

Microsoft, Windows, Windows 2003, Windows XP y el logotipo de Windows son marcas registradas de Microsoft Corporation en Estados Unidos y/o en otros países.

Otras empresas, productos y nombres de servicios pueden ser marcas registradas de terceros.

Bibliografía

Las siguientes publicaciones de IBM proporcionan información sobre los temas descritos en esta biblioteca. Las publicaciones se dividen en las siguientes grandes áreas temáticas:

- Communications Server para Linux, versión 6.2.2
- SNA (Arquitectura de red de sistemas)
- Configuración de sistema principal
- z/OS Communications Server
- Transmission Control Protocol/Internet Protocol (TCP/IP)
- X.25
- APPC (comunicación avanzada programa a programa)
- Programación
- Otros temas de redes de IBM

Se proporcionan breves descripciones de los manuales de la biblioteca de Communications Server para Linux. Del resto de los manuales, sólo se muestra el título, el número de pedido y, en algunos casos, el título abreviado que se utiliza en el texto de este manual.

Publicaciones de Communications Server para Linux Versión 6.2.2

La biblioteca de Communications Server para Linux consta de los manuales siguientes. Además, se proporcionan versiones en copia software de estos documentos en el CD-ROM. Si desea obtener información sobre cómo acceder a los archivos en copia software del CD-ROM, consulte la publicación *IBM Communications Server para Linux, Guía rápida de iniciación*. Para instalar en el sistema estos manuales en copia software, necesitará 9–15 MB de espacio de disco duro (según la versión de idioma que instale).

- *IBM Communications Server para Linux, Guía rápida de iniciación* (GC10-9852-02)

  Este manual ofrece una introducción general a Communications Server para Linux, conteniendo información sobre características de red soportadas, instalación, configuración y funcionamiento.
- *IBM Communications Server para Linux, Guía de administración* (SC10-9853-02)

  Este manual proporciona una visión general de SNA y Communications Server para Linux e información sobre la configuración y el funcionamiento de Communications Server para Linux.
- *IBM Communications Server for Linux Administration Command Reference* (SC31-6770-02)

  Este manual proporciona información sobre los mandatos de SNA y Communications Server para Linux.
- *IBM Communications Server para Linux, Guía del programador para CPI-C* (SC10-9861-02)

  Este manual proporciona información para programadores expertos de “C” o Java acerca de cómo escribir programas de transacciones SNA utilizando la API de comunicaciones de CPI-C de Communications Server for Linux.
- *IBM Communications Server para Linux, Guía del programador para APPC* (SC10-9854-02)

Este manual contiene la información necesaria para desarrollar programas de aplicación mediante APPC (comunicación avanzada programa a programa).

- *IBM Communications Server para Linux, Guía del programador para LUA* (SC10-9855-02)

  Este manual contiene la información necesaria para desarrollar aplicaciones utilizando la interfaz de programas de aplicación de LU (LUA) convencional.
- *IBM Communications Server for Linux, CSV Programmer's Guide* (SC31-6775-02)

  Este manual contiene la información necesaria para desarrollar programas de aplicación utilizando la interfaz de programas de aplicación (API) de CSV (Common Service Verbs).
- *IBM Communications Server for Linux, MS Programmer's Guide* (SC31-67770-02)

  Este manual contiene la información necesaria para desarrollar aplicaciones utilizando la API de MS (Management Services).
- *IBM Communications Server for Linux, NOF Programmer's Guide* (SC31-6778-02)

  Este manual contiene la información necesaria para desarrollar aplicaciones utilizando la API de NOF (Node Operator Facility).
- *IBM Communications Server para Linux, Guía de diagnósticos* (SC11-3348-02)

  Este manual proporciona información sobre la resolución de problemas en redes SNA.
- *IBM Communications Server for Linux, APPC Application Suite User's Guide* (SC31-6772-02)

  Este manual proporciona información sobre las aplicaciones APPC utilizadas con Communications Server para Linux.
- *IBM Communications Server for Linux, Glossary* (GC31-6780-02)

  Este manual contiene una lista completa de los términos y definiciones que se utilizan en la biblioteca de IBM Communications Server para Linux.

Publicaciones de SNA (Arquitectura de red de sistemas)

Los manuales siguientes contienen información sobre las redes SNA:

- *Systems Network Architecture: Format and Protocol Reference Manual—Architecture Logic for LU Type 6.2* (SC30-3269)
- *Systems Network Architecture: Formats* (GA27-3136)
- *Systems Network Architecture: Guide to SNA Publications* (GC30-3438)
- *Systems Network Architecture: Network Product Formats* (LY43-0081)
- *Systems Network Architecture: Technical Overview* (GC30-3073)
- *Systems Network Architecture: APPN Architecture Reference* (SC30-3422)
- *Systems Network Architecture: Sessions between Logical Units* (GC20-1868)
- *Systems Network Architecture: LU 6.2 Reference—Peer Protocols* (SC31-6808)
- *Systems Network Architecture: Transaction Programmer's Reference Manual for LU Type 6.2* (GC30-3084)
- *Systems Network Architecture: 3270 Datastream Programmer's Reference* (GA23-0059)
- *Networking Blueprint Executive Overview* (GC31-7057)
- *Systems Network Architecture: Management Services Reference* (SC30-3346)

Publicaciones sobre la configuración de sistemas principales

Los manuales siguientes contienen información sobre la configuración de sistemas principales:

- *ES/9000, ES/3090 IOCP User's Guide Volume A04* (GC38-0097)
- *3174 Establishment Controller Installation Guide* (GG24-3061)
- *3270 Information Display System 3174 Establishment Controller: Planning Guide* (GA27-3918)
- *OS/390 Hardware Configuration Definition (HCD) User's Guide* (SC28-1848)

Publicaciones sobre z/OS Communications

Los manuales siguientes contienen información sobre z/OS Communications Server:

- *z/OS V1R7 Communications Server: SNA Network Implementation Guide* (SC31-8777)
- *z/OS V1R7 Communications Server: SNA Diagnostics* (Vol 1: GC31-6850, Vol 2: GC31-6851)
- *z/OS V1R6 Communications Server: Resource Definition Reference* (SC31-8778)

Publicaciones sobre TCP/IP

Los manuales siguientes contienen información sobre el protocolo de red TCP/IP (Transmission Control Protocol/Internet Protocol):

- *z/OS V1R7 Communications Server: IP Configuration Guide* (SC31-8775)
- *z/OS V1R7 Communications Server: IP Configuration Reference* (SC31-8776)
- *z/VM V5R1 TCP/IP Planning and Customization* (SC24-6125)

Publicaciones sobre X.25

Los manuales siguientes contienen información sobre el protocolo de red X.25:

- *Communications Server for OS/2 Version 4 X.25 Programming* (SC31-8150)

Publicaciones sobre APPC

Los manuales siguientes contienen información sobre APPC (comunicación avanzada programa a programa):

- *APPC Application Suite V1 User's Guide* (SC31-6532)
- *APPC Application Suite V1 Administration* (SC31-6533)
- *APPC Application Suite V1 Programming* (SC31-6534)
- *APPC Application Suite V1 Online Product Library* (SK2T-2680)
- *APPC Application Suite Licensed Program Specifications* (GC31-6535)
- *z/OS V1R2.0 Communications Server: APPC Application Suite User's Guide* (SC31-8809)

Publicaciones de programación

Los manuales siguientes contienen información sobre programación:

- *Common Programming Interface Communications CPI-C Reference* (SC26-4399)
- *Communications Server for OS/2 Version 4 Application Programming Guide* (SC31-8152)

Otras publicaciones sobre redes de IBM

Los manuales siguientes contienen información sobre otros temas relacionados con Communications Server para Linux:

- *SDLC Concepts* (GA27-3093)
- *Local Area Network Concepts and Products: LAN Architecture* (SG24-4753)
- *Local Area Network Concepts and Products: LAN Adapters, Hubs and ATM* (SG24-4754)
- *Local Area Network Concepts and Products: Routers and Gateways* (SG24-4755)
- *Local Area Network Concepts and Products: LAN Operating Systems and Management* (SG24-4756)
- *IBM Network Control Program Resource Definition Guide* (SC30-3349)

Índice

A

C

D

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

V

Número de Programa: 5724-i33, 5724-i34

SC10-9854-01