IBM Communications Server para Linux

Guía del programador para CPI-C

Versión 6.2.2

SC10-9861-01

IBM Communications Server para Linux

Guía del programador para CPI-C

Versión 6.2.2

SC10-9861-01

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 D, "Avisos", en la página 187.

Segunda edición (julio de 2006)

Esta publicación es la traducción del original inglés *IBM Communications Server for Linux, CPI-C Programmer's Guide*, (SC31-6774-01).

Esta edición se aplica 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 publicaciones a través del representante local de IBM o sucursal de IBM que preste servicio en su localidad. No hay existencias de publicaciones en la dirección indicada más abajo.

IBM agradece sus comentarios. Al final de esta publicación encontrará un formulario para los comentarios del lector. Si el formulario se ha extraído, puede enviar sus comentarios a la dirección siguiente:

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

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

Cuando envía información a IBM, 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 . . . . . **xv**

Figuras . . . . . **xvii**

Acerca de este manual . . . . . **xix**
A quién va dirigido este manual xix
Cómo utilizar este manual xx
Organización de este manual xx
Convenios tipográficos xx
Convenios gráficos. xxi
Publicaciones relacionadas xxi

Capítulo 1. Conceptos. **1**
Qué es CPI-C 1
Soporte de conjuntos de opciones de CPI-C de Communications Server para Linux. 1
Comunicación entre programas 2
Logical Unit 6.2 3
Sesiones 3
Conversaciones 3
Contienda 3
Características 4
Llamadas CPI-C 4
Proceso de la conversación 4
Tipos de conversación 4
Una conversación correlacionada sencilla 5
Inicio de una conversación. 5
Envío de datos 5
Recepción de datos 6
Finalización de una conversación 6
Proceso de confirmación 6
Establecimiento del nivel de sincronización 6
Envío de una petición de confirmación 7
Recepción de una petición de confirmación 7
Respuesta a una petición de confirmación 7
Desasignación de la conversación 7
Estados de conversación 7
Visión de la conversación por parte del programa 9
Cambios de estado 9
Comprobaciones de estado 9
Cambio de los estados de conversación 9
Estados iniciales 10
Cambio al estado Recibir 10
Cambio al estado Enviar 11
Información complementaria. 11
Conversaciones básicas 12
Registros lógicos. 12
Datos de anotaciones de error 13
Conversaciones múltiples. 13
Visión general de la seguridad de conversación 13
Seguridad de conversación para conversaciones múltiples 14
Seguridad de conversación ya verificada. 15
Funcionamiento sin bloqueo 15
CPI-C y LU 6.2 18

Capítulo 2. Desarrollo de aplicaciones CPI-C **19**

Resumen de llamadas CPI-C. 19
Inicio de una conversación 19
Envío de datos 21
Recepción de datos 22
Conversión de datos entre ASCII y EBCDIC 23
Confirmación de recepción de datos e información de errores 23
Emisión de llamadas en modalidad sin bloqueo 24
Emisión de llamadas en modalidad con bloqueo 25
Obtención de información 25
Finalización de una conversación 26
Administración de la información complementaria 27
Características iniciales de la conversación 27
Información complementaria 31
Alias de LU local 32
Nombre de LU asociada 32
Tipo y nombre de programa asociado 32
Nombre de modalidad 32
Tipo de seguridad de conversación 32
Identificador de usuario y contraseña de seguridad 32
Información complementaria especificada por la aplicación 33
Configuración 34
Especificación del nombre de TP local 34
Specify_Local_TP_Name 34
Contexto 35
Variable de entorno APPCTPN 35
Valor por omisión 35
Especificación de la LU local 35
Set_Local_LU_Name 36
Contexto 36
Variable de entorno APPCLLU 36
Información complementaria 37
LU local por omisión 37
LU de punto de control 37
Cómo se inician los programas 37
Programa invocado de inicio automático 37
Programa invocado iniciado por el usuario 38
Consideraciones sobre AIX o Linux 38
Archivo de cabecera CPI-C 39
Procesos múltiples 39
Compilación y enlace de la aplicación CPI-C 39
Consideraciones sobre CPI-C de Java 40
Utilización de clases de CPI-C de Java 40
Ejemplo de uso 41
Compilación y enlace de la aplicación CPI-C de Java 42
Ejecución de la aplicación CPI-C de Java 42
consideraciones sobre Windows 43
Archivos CPI-C de Windows 43
Prototipos de funciones 43
Procesos múltiples y conversaciones múltiples 43
Llamadas de función de Windows 43
Llamadas de bloqueo 44
Finalización de aplicaciones 46
Compilación y enlace de aplicaciones CPI-C 46
Desarrollo de aplicaciones portables 46

Capítulo 3. Llamadas CPI-C 49
Información proporcionada para las llamadas CPI-C. 49
Tipos de datos 49
Estructuras de datos 50
Constantes simbólicas 50
Cadenas 50

Validez de los parámetros devueltos 50
Información proporcionada para las llamadas de función de Windows 50
Accept_Conversation (cmaccp) 51
Llamada de función 51
Llamada de función para CPI-C de Java 51
Parámetros suministrados 51
Parámetros devueltos 52
Estado al emitirse 52
Cambio de estado 52
Notas de uso 52
Accept_Incoming (cmacci) 53
Llamada de función 53
Llamada de función para CPI-C de Java 54
Parámetros suministrados 54
Parámetros devueltos 54
Estado al emitirse 54
Cambio de estado 55
Notas de uso 55
Allocate (cmallc). 55
Llamada de función 56
Llamada de función para CPI-C de Java 56
Parámetros suministrados 56
Parámetros devueltos 56
Estado al emitirse 57
Cambio de estado 57
Notas de uso 57
Cancel_Conversation (cmcanc) 58
Llamada de función 58
Llamada de función para CPI-C de Java 58
Parámetros suministrados 58
Parámetros devueltos 59
Estado al emitirse 59
Cambio de estado 59
Notas de uso 59
Check_For_Completion (cmchck) 59
Llamada de función 60
Parámetros suministrados 60
Parámetros devueltos 60
Estado al emitirse 60
Cambio de estado 60
Notas de uso 60
Confirm (cmcfm) 61
Llamada de función 61
Llamada de función para CPI-C de Java 61
Parámetros suministrados 61
Parámetros devueltos 62
Estado al emitirse 63
Cambio de estado 63
Notas de uso 63
Confirmed (cmcfmd) 63
Llamada de función 64
Llamada de función para CPI-C de Java 64
Parámetros suministrados 64
Parámetros devueltos 64
Estado al emitirse 64
Cambio de estado 65
Notas de uso 65
Convert_Incoming (cmcnvi) 66
Llamada de función 66
Llamada de función para CPI-C de Java 66
Parámetros suministrados 66

Parámetros devueltos 66
Estado al emitirse 67
Cambio de estado 67
Nota de uso 67
Convert_Outgoing (cmcnvo) 67
Llamada de función 67
Llamada de función para CPI-C de Java 67
Parámetros suministrados 68
Parámetros devueltos 68
Estado al emitirse 68
Cambio de estado 68
Nota de uso 69
Deallocate (cmdeal) 69
Llamada de función 69
Llamada de función para CPI-C de Java 69
Parámetros suministrados 69
Parámetros devueltos 70
Estado al emitirse 70
Cambio de estado 71
Notas de uso 71
Delete_CPIC_Side_Information (xcmdsi) 71
Llamada de función 72
Parámetros suministrados 72
Parámetros devueltos 72
Estado al emitirse 72
Cambio de estado 72
Notas de uso 72
Extract_Conversation_Context (cmectx) 73
Llamada de función 73
Llamada de función para CPI-C de Java 73
Parámetros suministrados 73
Parámetros devueltos 73
Estado al emitirse 74
Cambio de estado 74
Notas de uso 74
Extract_Conversation_Security_Type (xcecst) 74
Llamada de función 74
Parámetros suministrados 74
Parámetros devueltos 75
Estado al emitirse 76
Cambio de estado 76
Extract_Conversation_Security_User_ID (cmecsu) 76
Extract_Conversation_Security_User_ID (xcecsu) 76
Extract_Conversation_State (cmecs) 76
Llamada de función 77
Llamada de función para CPI-C de Java 77
Parámetros suministrados 77
Parámetros devueltos 77
Estado al emitirse 78
Cambio de estado 78
Extract_Conversation_Type (cmect) 78
Llamada de función 78
Llamada de función para CPI-C de Java 78
Parámetros suministrados 78
Parámetros devueltos 78
Estado al emitirse 79
Cambio de estado 79
Extract_CPIC_Side_Information (xcmesi) 79
Llamada de función 79
Parámetros suministrados 79
Parámetros devueltos 80

Estado al emitirse . . . 81
Cambio de estado . . . 81
Notas de uso . . . 81
Extract_Local_LU_Name (cmelln) . . . 82
Llamada de función . . . 82
Llamada de función para CPI-C de Java . . . 82
Parámetros suministrados . . . 82
Parámetros devueltos . . . 82
Estado al emitirse . . . 83
Cambio de estado . . . 83
Notas de uso . . . 83
Extract_Maximum_Buffer_Size (cmembs) . . . 83
Llamada de función . . . 83
Llamada de función para CPI-C de Java . . . 83
Parámetros suministrados . . . 83
Parámetros devueltos . . . 83
Estado al emitirse . . . 84
Cambio de estado . . . 84
Extract_Mode_Name (cmemn) . . . 84
Llamada de función . . . 84
Llamada de función para CPI-C de Java . . . 84
Parámetros suministrados . . . 84
Parámetros devueltos . . . 85
Estado al emitirse . . . 85
Cambio de estado . . . 85
Extract_Partner_LU_Name (cmepln) . . . 85
Llamada de función . . . 85
Llamada de función para CPI-C de Java . . . 85
Parámetros suministrados . . . 86
Parámetros devueltos . . . 86
Estado al emitirse . . . 86
Cambio de estado . . . 86
Extract_Security_User_ID (cmesui o cmecsu) . . . 86
Llamada de función . . . 87
Llamada de función para CPI-C de Java . . . 87
Parámetros suministrados . . . 87
Parámetros devueltos . . . 87
Estado al emitirse . . . 88
Cambio de estado . . . 88
Notas de uso . . . 88
Extract_Sync_Level (cmesl) . . . 88
Llamada de función . . . 88
Llamada de función para CPI-C de Java . . . 88
Parámetros suministrados . . . 88
Parámetros devueltos . . . 89
Estado al emitirse . . . 89
Cambio de estado . . . 89
Extract_TP_Name (cmetpn) . . . 89
Llamada de función . . . 89
Llamada de función para CPI-C de Java . . . 90
Parámetros suministrados . . . 90
Parámetros devueltos . . . 90
Estado al emitirse . . . 90
Cambio de estado . . . 90
Flush (cmflus) . . . 91
Orígenes de los datos del almacenamiento intermedio . . . 91
Llamada de función . . . 91
Llamada de función para CPI-C de Java . . . 91
Parámetros suministrados . . . 91
Parámetros devueltos . . . 91
Estado al emitirse . . . 92

Cambio de estado 92
Initialize_Conversation (cminit) 92
Llamada de función 92
Llamada de función para CPI-C de Java 92
Parámetros suministrados 93
Parámetros devueltos 93
Estado al emitirse 93
Cambio de estado 93
Notas de uso 94
Initialize_For_Incoming (cminic) 94
Llamada de función 94
Llamada de función para CPI-C de Java 94
Parámetros suministrados 94
Parámetros devueltos 94
Estado al emitirse 94
Cambio de estado 95
Prepare_To_Receive (cmptr) 95
Llamada de función 95
Llamada de función para CPI-C de Java 95
Parámetros suministrados 96
Parámetros devueltos 96
Estado al emitirse 97
Cambio de estado 97
Notas de uso 97
Receive (cmrcv) 97
Recepción de los datos por parte de un programa 98
Llamada de función 98
Llamada de función para CPI-C de Java 98
Parámetros suministrados 99
Parámetros devueltos 99
Estado al emitirse 102
Cambio de estado 103
Notas de uso 105
Release_Local_TP_Name (cmrltp) 106
Llamada de función 106
Llamada de función para CPI-C de Java 106
Parámetros suministrados 106
Parámetros devueltos 106
Estado al emitirse 107
Cambio de estado 107
Notas de uso 107
Request_To_Send (cmrts) 107
Acción del programa asociado 107
Cuándo puede enviar datos el programa local 107
Llamada de función 108
Llamada de función para CPI-C de Java 108
Parámetros suministrados 108
Parámetros devueltos 108
Estado al emitirse 108
Cambio de estado 109
Notas de uso 109
Send_Data (cmsend) 109
Llamada de función 110
Llamada de función para CPI-C de Java 110
Parámetros suministrados 110
Parámetros devueltos 110
Estado al emitirse 112
Cambio de estado 112
Notas de uso 112
Send_Error (cmserr) 112
Llamada de función 113

Llamada de función para CPI-C de Java . . . 113
Parámetros suministrados . . . 113
Parámetros devueltos . . . 113
Estado al emitirse . . . 116
Cambio de estado . . . 116
Notas de uso . . . 116
Set_Conversation_Context (cmsctx) . . . 117
Llamada de función . . . 117
Llamada de función para CPI-C de Java . . . 117
Parámetros suministrados . . . 117
Parámetros devueltos . . . 118
Estado al emitirse . . . 118
Cambio de estado . . . 118
Notas de uso . . . 118
Set_Conversation_Security_Password (cmscsp) . . . 118
Llamada de función . . . 119
Llamada de función para CPI-C de Java . . . 119
Parámetros suministrados . . . 119
Parámetros devueltos . . . 119
Estado al emitirse . . . 120
Cambio de estado . . . 120
Notas de uso . . . 120
Set_Conversation_Security_Password (xcscsp) . . . 120
Set_Conversation_Security_Type (cmscst) . . . 121
Llamada de función . . . 121
Llamada de función para CPI-C de Java . . . 121
Parámetros suministrados . . . 121
Parámetros devueltos . . . 122
Estado al emitirse . . . 122
Cambio de estado . . . 122
Notas de uso . . . 122
Set_Conversation_Security_Type (xcscst) . . . 123
Set_Conversation_Security_User_ID (cmscsu) . . . 123
Llamada de función . . . 123
Llamada de función para CPI-C de Java . . . 123
Parámetros suministrados . . . 123
Parámetros devueltos . . . 124
Estado al emitirse . . . 124
Cambio de estado . . . 124
Notas de uso . . . 124
Set_Conversation_Security_User_ID (xcscsu) . . . 125
Set_Conversation_Type (cmsct) . . . 125
Llamada de función . . . 125
Llamada de función para CPI-C de Java . . . 125
Parámetros suministrados . . . 125
Parámetros devueltos . . . 126
Estado al emitirse . . . 126
Cambio de estado . . . 126
Notas de uso . . . 126
Set_CPIC_Side_Information (xcmssi) . . . 126
Llamada de función . . . 127
Parámetros suministrados . . . 127
Parámetros devueltos . . . 129
Estado al emitirse . . . 130
Cambio de estado . . . 130
Notas de uso . . . 130
Set_Deallocate_Type (cmsdt) . . . 130
Llamada de función . . . 130
Llamada de función para CPI-C de Java . . . 130
Parámetros suministrados . . . 131
Parámetros devueltos . . . 132

Estado al emitirse . . . 132
Cambio de estado . . . 132
Notas de uso . . . 132
Set_Error_Direction (cmsed) . . . 133
Llamada de función . . . 133
Llamada de función para CPI-C de Java . . . 133
Parámetros suministrados . . . 133
Parámetros devueltos . . . 133
Estado al emitirse . . . 134
Cambio de estado . . . 134
Notas de uso . . . 134
Set_Fill (cmsf) . . . 134
Llamada de función . . . 134
Llamada de función para CPI-C de Java . . . 135
Parámetros suministrados . . . 135
Parámetros devueltos . . . 135
Estado al emitirse . . . 136
Cambio de estado . . . 136
Notas de uso . . . 136
Set_Local_LU_Name (cmslln) . . . 136
Llamada de función . . . 136
Llamada de función para CPI-C de Java . . . 136
Parámetros suministrados . . . 136
Parámetros devueltos . . . 137
Estado al emitirse . . . 137
Cambio de estado . . . 137
Notas de uso . . . 137
Set_Log_Data (cmsld) . . . 137
Llamada de función . . . 138
Llamada de función para CPI-C de Java . . . 138
Parámetros suministrados . . . 138
Parámetros devueltos . . . 138
Estado al emitirse . . . 139
Cambio de estado . . . 139
Notas de uso . . . 139
Set_Mode_Name (cmsmn) . . . 139
Llamada de función . . . 139
Llamada de función para CPI-C de Java . . . 140
Parámetros suministrados . . . 140
Parámetros devueltos . . . 140
Estado al emitirse . . . 141
Cambio de estado . . . 141
Notas de uso . . . 141
Set_Partner_LU_Name (cmspln) . . . 141
Llamada de función . . . 141
Llamada de función para CPI-C de Java . . . 141
Parámetros suministrados . . . 142
Parámetros devueltos . . . 142
Estado al emitirse . . . 142
Cambio de estado . . . 143
Notas de uso . . . 143
Set_Prepare_To_Receive_Type (cmsptr) . . . 143
Llamada de función . . . 143
Llamada de función para CPI-C de Java . . . 143
Parámetros suministrados . . . 143
Parámetros devueltos . . . 144
Estado al emitirse . . . 144
Cambio de estado . . . 144
Notas de uso . . . 145
Set_Processing_Mode (cmspm) . . . 145
Llamada de función . . . 146

Parámetros suministrados . . . 146
Parámetros devueltos. . . . 146
Estado al emitirse . . . 146
Cambio de estado . . . 146
Notas de uso . . . 146
Set_Receive_Type (cmsrt) . . . 147
Llamada de función . . . 147
Llamada de función para CPI-C de Java . . . 147
Parámetros suministrados . . . 147
Parámetros devueltos. . . . 147
Estado al emitirse . . . 148
Cambio de estado . . . 148
Notas de uso . . . 148
Set_Return_Control (cmsrc). . . . 148
Llamada de función . . . 148
Llamada de función para CPI-C de Java . . . 148
Parámetros suministrados . . . 149
Parámetros devueltos. . . . 149
Estado al emitirse . . . 149
Cambio de estado . . . 149
Notas de uso . . . 149
Set_Send_Type (cmsst) . . . 150
Llamada de función . . . 150
Llamada de función para CPI-C de Java . . . 150
Parámetros suministrados . . . 150
Parámetros devueltos. . . . 151
Estado al emitirse . . . 151
Cambio de estado . . . 151
Notas de uso . . . 151
Set_Sync_Level (cmssl) . . . 151
Llamada de función . . . 152
Llamada de función para CPI-C de Java . . . 152
Parámetros suministrados . . . 152
Parámetros devueltos. . . . 152
Estado al emitirse . . . 153
Cambio de estado . . . 153
Notas de uso . . . 153
Set_TP_Name (cmstpn) . . . 153
Llamada de función . . . 153
Llamada de función para CPI-C de Java . . . 153
Parámetros suministrados . . . 154
Parámetros devueltos. . . . 154
Estado al emitirse . . . 154
Cambio de estado . . . 155
Notas de uso . . . 155
Specify_Local_TP_Name (cmsltp). . . . 155
Llamada de función . . . 155
Llamada de función para CPI-C de Java . . . 155
Parámetros suministrados . . . 155
Parámetros devueltos. . . . 156
Estado al emitirse . . . 156
Cambio de estado . . . 156
Notas de uso . . . 156
Specify_Windows_Handle (xchwnd). . . . 156
Llamada de función . . . 157
Parámetros suministrados . . . 157
Parámetros devueltos. . . . 157
Estado al emitirse . . . 158
Cambio de estado . . . 158
Test_Request_to_Send_Received (cmtrts) . . . 158
Llamada de función . . . 158

Llamada de función para CPI-C de Java . . . 158
Parámetros suministrados . . . 158
Parámetros devueltos . . . 159
Estado al emitirse . . . 159
Cambio de estado . . . 159
Wait_For_Conversation (cmwait) . . . 159
Llamada de función . . . 160
Parámetros suministrados . . . 160
Parámetros devueltos . . . 160
Estado al emitirse . . . 161
Cambio de estado . . . 161
Notas de uso . . . 161
WinCPICCleanup . . . 162
Llamada de función . . . 162
Parámetros suministrados . . . 162
Valores de retorno . . . 162
WinCPICIsBlocking . . . 162
Llamada de función . . . 163
Parámetros suministrados . . . 163
Valores de retorno . . . 163
WinCPICSetBlockingHook . . . 163
Llamada de función . . . 163
Parámetros suministrados . . . 163
Valores de retorno . . . 163
Uso . . . 163
WinCPICStartup . . . 164
Llamada de función . . . 164
Parámetros suministrados . . . 164
Valores de retorno . . . 164
WinCPICUnhookBlockingHook . . . 165
Llamada de función . . . 165
Parámetros suministrados . . . 166
Valores de retorno . . . 166
WinCPICSetEvent . . . 166
Llamada de función . . . 166
Parámetros suministrados . . . 166
Parámetros devueltos . . . 166
Notas de uso . . . 167
WinCPICExtractEvent . . . 167
Llamada de función . . . 167
Parámetros suministrados . . . 167
Parámetros devueltos . . . 167
Notas de uso . . . 167

Capítulo 4. Programas de transacciones CPI-C de ejemplo . . . 169
Visión general del proceso . . . 169
Pseudocódigo . . . 169
CSAMPLE1 (programa que invoca) . . . 169
CSAMPLE2 (TP invocado) . . . 170
Cómo probar los TP . . . 170

Capítulo 5. Programa de transacciones CPI-C de Java de ejemplo . . . 173
Visión general . . . 173
Compilación, enlace y ejecución del programa de ejemplo . . . 173

Apéndice A. Valores de código de retorno . . . 175

Apéndice B. Códigos de retorno comunes . . . 177
Códigos de retorno de cualquier programa asociado . . . 177
Programa asociado de LU 6.2 no CPI-C . . . 181

Apéndice C. Cambios de estado de conversación. 183

Apéndice D. Avisos 187
Marcas registradas. 189

Bibliografía 191
Publicaciones de Communications Server para Linux Versión 6.2.2 191
Publicaciones de SNA (Arquitectura de red de sistemas) 192
Publicaciones sobre la configuración de sistemas principales. 193
Publicaciones de z/OS Communications Server 193
Publicaciones sobre TCP/IP 193
Publicaciones sobre X.25. 193
Publicaciones de APPC 193
Publicaciones sobre programación 193
Otras publicaciones sobre redes de IBM 194

Índice. 195

Tablas

1. Convenios tipográficos xxi
2. Correlación entre funciones X/Open y funciones CPI-C 2.0 de IBM 2
3. Una conversación correlacionada sencilla 5
4. Proceso de confirmación 6
5. Cambio de los estados de conversación 9
6. Funcionamiento sin bloqueo 16
7. Llamadas Set_* para cambiar las características iniciales de la conversación 20
8. Llamadas Extract_* y acciones 25
9. Llamadas para añadir, sustituir, recuperar o suprimir información complementaria 27
10. Cambio de las características iniciales de la conversación 28
11. Constantes de CPI-C de Java 40
12. Cambios de estado para la llamada Allocate 57
13. Cambios de estado para la llamada Confirm 63
14. Cambios de estado para la llamada Confirmed 65
15. Estados de conversación al emitir la llamada Deallocate 71
16. Cambios de estado para la llamada Deallocate 71
17. Cambios de estado para la llamada Prepare_To_Receive 97
18. Cambios de estado cuando se emite la llamada Receive en estado Recibir 103
19. Cambios de estado cuando se emite la llamada Receive en estado Enviar 103
20. Cambios de estado cuando se emite la llamada Receive en estado Enviar-Pendiente 104
21. Cambios de estado cuando se emite la llamada Receive en cualquier estado permitido 104
22. Cambios de estado ocasionados por un error de transmisión de datos 105
23. Cambios de estado para la llamada Send_Data 112
24. Cambios de estado para la llamada Send_Error 116
25. Cambios de estado de conversación 184

Figuras

1. Comunicación entre programas 3
2. Conversaciones múltiples 13

Acerca de este manual

Esta publicación es una guía para desarrollar programas de aplicación de lenguaje C o Java que utilizan la interfaz común de programación para comunicaciones (CPI-C) 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 permite a un sistema ejecutar Linux para intercambiar información con otros nodos en una red SNA.

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

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

- 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 en System z
Communications Server para Linux en System z, número de producto de 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 una de las 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 cuando las diferencias se describan explícitamente.

La implementación de Communications Server para Linux de CPI-C se basa en la implementación de IBM de CPI-C en los productos de OS/2 (con modificaciones para el entorno Linux).

Los programas desarrollados para utilizar la implementación de Communications Server para Linux de CPI-C pueden intercambiar datos con los programas escritos para utilizar otras implementaciones de CPI-C 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

Este manual va dirigido a programadores expertos del lenguaje C o Java que desarrollan programas de transacciones SNA (Arquitectura de red de sistemas) para sistemas con Communications Server para Linux. Los programadores pueden tener o no experiencia previa con SNA o los recursos de comunicación de Communications Server para Linux.

Los programadores de aplicaciones diseñan y codifican programas de transacciones y aplicaciones 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 obtener información más detallada acerca de cómo desarrollar programas de aplicación en el manual correspondiente a cada 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 CPI-C. Está destinado a los programadores que no están familiarizados con CPI-C.
- El Capítulo 2, “Desarrollo de aplicaciones CPI-C”, en la página 19 contiene la información de carácter general que necesita un programador para CPI-C al desarrollar aplicaciones CPI-C.
- El Capítulo 3, “Llamadas CPI-C”, en la página 49 describe de forma detallada cada una de las llamadas CPI-C. Cada una de las descripciones incluye la siguiente información: finalidad, parámetros, estados de conversación en que puede emitirse la llamada y cambios de estado de conversación que se producen una vez que se ha ejecutado la llamada. Las diferencias entre las implementaciones de CPI-C para los distintos sistemas operativos se indican en los lugares en los que se producen.
- El Capítulo 4, “Programas de transacciones CPI-C de ejemplo”, en la página 169 describe los programas de ejemplo de CPI-C de Communications Server para Linux que ilustran el uso de las llamadas CPI-C en un programa C e incluye instrucciones para compilar, enlazar y ejecutar los programas.

  AIX, LINUX

- El Capítulo 5, “Programa de transacciones CPI-C de Java de ejemplo”, en la página 173 describe el programa de ejemplo de CPI-C de Java de Communications Server para Linux que ilustra el uso de las llamadas CPI-C en una aplicación Java e incluye instrucciones para compilar, enlazar y ejecutar el programa.

  

- El Apéndice A, “Valores de código de retorno”, en la página 175 lista todos los códigos de retorno posibles en la interfaz de CPI-C en orden numérico y proporciona sus significados.
- El Apéndice B, “Códigos de retorno comunes”, en la página 177 ofrece información sobre algunos códigos de retorno comunes a varias llamadas.
- El Apéndice C, “Cambios de estado de conversación”, en la página 183 proporciona información sobre los estados de conversación de CPI-C: qué llamadas pueden emitirse en cada uno de los estados y el estado al que cambia la conversación cuando vuelve de cada una de las llamadas.

Convenios tipográficos

La Tabla 1 en la página xxi 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	**cmc.h**
Mandato o programa de utilidad de AIX / Linux	**vi**
Opción o indicador	**-I**
Parámetro o campo de Motif	*data_received*; *request_to_send_received*
Valor literal o selección que el usuario puede entrar (incluidos los valores por omisión)	`0`; `32.767`
Constante o señal	`CM_NONE`
Valor de retorno	`CM_OK`; `CM_PRODUCT_SPECIFIC_ERROR`
Variable que representa un valor suministrado	*nombrefunción*
Variable de entorno	`APPCTPN`
Verbo de programación	RECEIVE
Datos entrados por el usuario	**cc -I**
Función, llamada o punto de entrada	`WinCPICSetEvent`
Estructura de datos	`WCPICDATA`
Valor hexadecimal	`0x20`

Convenios gráficos

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

Este símbolo se utiliza para indicar el comienzo de una sección de texto que corresponde a IBM Remote API Client en Windows.

Este símbolo indica el final de una sección de texto específica del sistema operativo. Se aplica la información que va a continuación de este símbolo independientemente del sistema operativo.

Publicaciones relacionadas

Para obtener información sobre la arquitectura SNA o LU 6.2, consulte estos documentos de IBM:

- *IBM Systems Network Architecture:*
  - *APPN Architecture Reference*, SC30-3422.
  - *LU 6.2 Reference: Peer Protocols*, SC31-6808

  - *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
- *IBM Common Programming Interface Communications Specification*, SC31-6180

Capítulo 1. Conceptos

Este capítulo presenta los conceptos básicos de CPI-C en un entorno de proceso distribuido. Se tratan los temas siguientes:

- Qué es CPI-C
- Ejemplo de una conversación correlacionada sencilla
- Proceso de confirmación
- Estados de conversación
- Cambio de los estados de conversación
- Información complementaria
- Conversaciones básicas
- Conversaciones múltiples
- Seguridad de conversación
- Funcionamiento sin bloqueo
- CPI-C y LU 6.2

Qué es CPI-C

CPI-C es la sigla inglesa que corresponde a la interfaz común de programación para comunicaciones. CPI-C es una interfaz de programación de aplicaciones (API) portable que habilita las comunicaciones de igual a igual entre programas en un entorno SNA.

CPI-C permite que programas de aplicación distribuidos en una red puedan trabajar juntos. Comunicándose unos con otros e intercambiando datos, pueden llevar a cabo una determinada tarea, como por ejemplo consultar una base de datos remota, copiar un archivo remoto o enviar o recibir correo electrónico.

Estos programas se comunican como iguales y no según una estructura jerárquica. Juntos, los programas distribuidos por una red de área local o de área amplia llevan a cabo procesos distribuidos.

Soporte de conjuntos de opciones de CPI-C de Communications Server para Linux

AIX, LINUX

Para programas C (no para programas Java), CPI-C de Communications Server para Linux implementa CPI-C 2.0 de IBM. Soporta la clase obligatoria para el cumplimiento de CPI-C 2.0, Conversations, y las siguientes clases opcionales:

- LU 6.2
- Funcionamiento sin bloqueo a nivel de conversación
- Servidor
- Rutinas de conversión de datos
- Seguridad

Asimismo, CPI-C de Communications Server para Linux proporciona soporte para funciones adicionales que se han definido como para de la implementación de X/Open CPI-C y se han incorporado a CPI-C 2.0 de IBM. Communications Server para Linux soporta estos puntos de entrada para garantizar la compatibilidad con versiones anteriores de aplicaciones CPI-C existentes. Siempre que sea posible, los programadores de CPI-C deberán utilizar las versiones CPI-C 2.0 de IBM. La correlación entre las funciones X/Open y las funciones CPI-C 2.0 de IBM se muestra en la Tabla 2.

Tabla 2. Correlación entre funciones X/Open y funciones CPI-C 2.0 de IBM

Función X/Open	Función CPI-C 2.0 de IBM
Extract_Conversation_Security_User_ID (xcecsu)	Extract_Security_User_ID (cmesui)
Set_Conversation_Security_Password (xcscsp)	Set_Conversation_Security_Password (cmscsp)
Set_Conversation_Security_Type (xcscst)	Set_Conversation_Security_Type (cmscst)
Set_Conversation_Security_User_ID (xcscsu)	Set_Conversation_Security_User_ID (cmscsu)

Para programas Java, Communications Server para Linux implementa CPI-C de Java como en el producto CS/Windows de IBM (el paquete COM.ibm.eNetwork.cpic). También incluye tres funciones CPI-C adicionales (Set_Conversation_Context, Set_Local_LU_Name y Extract_Local_LU_Name) que forman parte de la implementación estándar CPI-C de Communications Server para Linux pero que no se incluyen en CS/Windows.

WINDOWS

CPI-C de Communications Server para Linux en Windows implementa CPI-C de Windows (como define la especificación WOSA SNA).

Comunicación entre programas

Para que dos programas puedan comunicarse son necesarios muchos elementos de hardware y software en el entorno SNA. El diagrama siguiente muestra los elementos relevantes para los programadores.

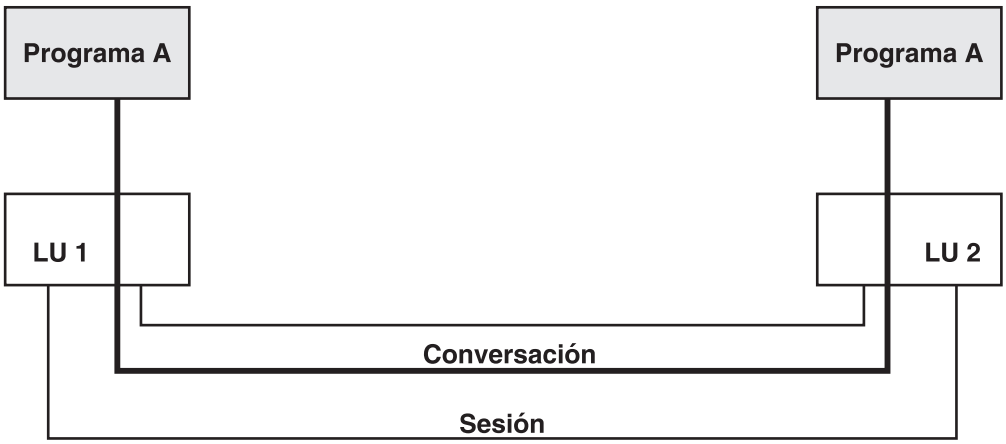

Figura 1. Comunicación entre programas

Logical Unit 6.2

Cada programa está asociado a una unidad lógica (LU), que es el punto de acceso del programa a la red. CPI-C utiliza el tipo de LU 6.2, que soporta las comunicaciones de igual a igual entre LU. Varios programas pueden estar asociados a la misma LU.

Sesiones

Para que dos programas puedan comunicarse, las LU deben estar conectadas a través de una sesión LU-LU (una conexión lógica entre las dos LU). La sesión se establece utilizando una modalidad concreta (un conjunto de características de red que determina cómo utilizan la sesión las LU).

Una LU de tipo 6.2 puede tener sesiones múltiples (dos o más sesiones simultáneas con distintas LU asociadas) y sesiones paralelas (dos o más sesiones simultáneas con la misma LU asociada). En la configuración, el administrador del sistema o el usuario determina cuántas sesiones soporta una LU concreta y si la LU proporciona soporte para sesiones paralelas.

Conversaciones

La comunicación entre los dos programas se produce como una conversación dentro de la sesión LU-LU. Un programa puede participar en varias conversaciones de forma simultánea.

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 modalidad utilizada por las dos LU especifica el número de sesiones ganadoras y perdedoras de contienda para cada LU; la LU ganadora de la contienda y la LU perdedora de la contienda se determinan cuando se establece la sesión.

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

Características

Una conversación tiene un conjunto de valores internos que controlan el funcionamiento global de la conversación así como el de las llamadas individuales. Estos valores se denominan características.

Llamadas CPI-C

Un programa accede a CPI-C mediante llamadas CPI-C. Cada una de las llamadas realiza una acción concreta, como por ejemplo iniciar o finalizar una conversación, enviar o recibir datos, establecer una opción que determine cómo operarán las llamadas CPI-C posteriores u obtener información sobre las opciones que se encuentren en uso en ese momento. En cada llamada, el programa proporciona parámetros a CPI-C, que lleva a cabo la función solicitada y devuelve nuevos parámetros al programa.

El programa que emite la llamada se denomina programa local y el otro programa, programa asociado. Del mismo modo, la LU que da servicio al programa local es la LU local y la LU que da servicio al programa asociado es la LU asociada.

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

Proceso de la conversación

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

- Un programa (el programa que invoca) indica a Communications Server para Linux que inicie otro programa (el programa invocado) y asigne una conversación entre los dos programas.
- El programa invocado notifica a Communications Server para Linux que ya está listo para comunicarse con el programa que invoca.

Durante la conversación, los dos programas intercambian información de estado y datos de aplicación. Generalmente, una conversación finaliza cuando un programa indica a Communications Server para Linux que desasigne la conversación.

Tipos de conversación

Una conversación puede ser correlacionada o básica.

En general, los programas de aplicación utilizan las conversaciones correlacionadas. Éstos son programas que efectúan tareas para usuarios finales. Las conversaciones correlacionadas son menos complejas que las conversaciones básicas. En una conversación correlacionada, los programas envían y reciben datos en un registro cada vez.

Los programas de servicio normalmente utilizan conversaciones básicas. Éstos 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 la sección “Conversaciones básicas” en la página 12.

Una conversación correlacionada sencilla

El ejemplo siguiente ilustra una conversación correlacionada sencilla. Muestra las llamadas CPI-C utilizadas para iniciar la conversación, intercambiar datos y finalizarla. La flecha indica el flujo de datos. Se muestran también entre paréntesis algunos parámetros de llamada y algunos códigos de retorno.

Tabla 3. Una conversación correlacionada sencilla

Programa que invoca		Programa invocado
Initialize_Conversation Allocate Send_Data Deallocate	→	Accept_Conversation Receive (*data_received*=CM_COMPLETE_DATA_RECEIVED) (*return_code*=CM_DEALLOCATED_NORMAL)

Inicio de una conversación

Para iniciar una conversación, el programa que invoca emite las llamadas siguientes:

- Initialize_Conversation, que solicita a CPI-C que establezca las características de la conversación.

  La llamada Initialize_Conversation especifica un nombre de destino simbólico, que está asociado a una entrada de información complementaria de CPI-C en la configuración de Communications Server para Linux. Esta entrada especifica el programa asociado, la LU asociada, la modalidad y la información sobre seguridad.
- Allocate, que solicita que Communications Server para Linux establezca una conversación entre el programa que invoca y el programa invocado.

El programa invocado emite la llamada Accept_Conversation, que informa a Communications Server para Linux que el programa invocado está preparado para iniciar una conversación con el programa que invoca.

Envío de datos

La llamada Send_Data coloca un registro de datos (con datos de aplicación para transmitir) en el almacenamiento intermedio de envío de la LU local que ya contiene la petición de asignación. La transmisión de los datos al programa asociado no se realiza hasta que se produce uno de los sucesos siguientes:

- El almacenamiento intermedio de envío se llena.
- El programa emite una llamada que fuerza a Communications Server para Linux a vaciar el almacenamiento intermedio (y enviar los datos al programa asociado)

La llamada Deallocate vacía el almacenamiento intermedio de envío enviando la petición de asignación y los datos al programa asociado.

Recepción de datos

La llamada Receive recibe el registro de datos y la información de estado del programa asociado. Si en ese momento no hay información de estado o datos disponibles, por omisión el programa local espera a que lleguen los datos.

El parámetro *data_received* de la llamada Receive indica al programa si ha recibido los datos y, en ese caso, si los datos están completos o no.

Finalización de una conversación

Para finalizar una conversación, uno de los programas emite la llamada Deallocate, lo que hace que Communications Server para Linux desasigne la conversación entre los dos programas.

Proceso de confirmación

Cuando un programa envía datos al programa asociado, también puede solicitar al programa asociado que confirme que ha recibido los datos correctamente. El programa receptor debe confirmar la recepción de los datos o indicar que se ha producido un error. Los dos programas se sincronizan cada vez que intercambian una petición de confirmación y una respuesta. Esto se ilustra en la Tabla 4.

Tabla 4. Proceso de confirmación

Programa que invoca		Programa invocado
Initialize_Conversation Set_Sync_Level (*sync_level*=CM_CONFIRM) Allocate Send_Data Confirm		
	→	
		Accept_Conversation Receive (*data_received*=CM_COMPLETE_DATA_RECEIVED) (*status_received*=CM_CONFIRM_RECEIVED) Confirmed
	←	
(*return_code*=CM_OK) Send_Data Deallocate		
	→	
		Receive (*status_received*=CM_CONFIRM_DEALLOC_RECEIVED) Confirmed
	←	
(*return_code*=CM_OK)		

Establecimiento del nivel de sincronización

El nivel de sincronización es una de las características de la conversación. Hay dos niveles de sincronización posibles:

- CM_NONE, el nivel por omisión, en el que no se produce el proceso de confirmación

- CM_CONFIRM, en el que los programas pueden solicitar la confirmación de la recepción de datos y responder a estas peticiones

El nivel de sincronización por omisión es CM_NONE; puede alterar temporalmente este valor utilizando la llamada Set_Sync_Level.

Envío de una petición de confirmación

La emisión de la llamada Confirm tiene el efecto siguiente:

- Vacía el almacenamiento intermedio de envío de la LU local (con lo cual se envían todos los datos contenidos en el almacenamiento intermedio al programa asociado).
- Envía una petición de confirmación, que recibe el programa asociado mediante el parámetro *status_received* de una llamada Receive.

Tras emitir la llamada Confirm, el programa que invoca espera la confirmación del programa invocado.

Recepción de una petición de confirmación

El parámetro *status_received* de la llamada Receive indica la próxima acción que requiere el programa local.

En el ejemplo anterior, la primera llamada Received tiene un parámetro *status_received* con el valor CM_CONFIRM_RECEIVED, lo que indica que se necesita una confirmación para que el programa asociado pueda continuar.

Respuesta a una petición de confirmación

El programa invocado emite la llamada Confirmed para confirmar la recepción de datos; esto libera el programa que invoca para reanudar el proceso.

Desasignación de la conversación

Como el nivel de sincronización de la conversación está establecido en CM_CONFIRM, la llamada Deallocate envía una petición de confirmación con los datos vaciados del almacenamiento intermedio.

En el caso de la segunda llamada Receive, el valor de *status_received* es CM_CONFIRM_DEALLOC_RECEIVED, lo que indica que el programa asociado necesita una confirmación, generada por la llamada Confirmed, para que se pueda desasignar la conversación.

Estados de conversación

El estado de la conversación rige qué llamadas CPI-C puede emitir el programa. Por ejemplo, un programa no puede emitir la llamada Send_Data si la conversación no se encuentra en estado Enviar o Enviar-Pendiente. Los estados de conversación posibles se resumen en la lista siguiente.

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

Inicializar
La conversación se ha inicializado correctamente.

Enviar El programa puede enviar datos al programa asociado y solicitar

confirmación. Cuando la conversación está en estado Enviar, el programa también puede empezar a recibir datos, lo que puede hacer que el estado cambie a Recibir.

Enviar-Pendiente
El programa ha emitido una llamada Receive y ha recibido datos así como un indicador de envío (*status_received*=CM_SEND_RECEIVED), que indica que el programa puede empezar a enviar datos. Este estado es parecido al estado Enviar, con la diferencia de que el programa puede proporcionar información adicional al informar de errores (para indicar si ha detectado un error en los datos recibidos o en su propio proceso).

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

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

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

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

AIX, LINUX

Inicializar-Entrante
El programa ha emitido correctamente Initialize_for_Incoming y ha obtenido un identificador de conversación. Ahora puede emitir Accept_Incoming para aceptar una conversación entrante.

WINDOWS

Pendiente-Anotar
El programa ha emitido satisfactoriamente la llamada Receive en modalidad sin bloqueo. Mientras la llamada está pendiente, puede emitir un rango limitado de llamadas CPI-C en esta conversación, emitir llamadas CPI-C en otras conversaciones o continuar con otros procesos.

La descripción de cada una de las llamadas CPI-C incluye información sobre los estados de conversación en que puede emitirse. Para ver una tabla de los verbos que pueden emitirse en cada uno de los estados de conversación, consulte el Apéndice C, “Cambios de estado de conversación”, en la página 183.

Visión de la conversación por parte del programa

Es la conversación, y no el programa, la que se encuentra en un estado determinado. Un programa puede mantener varias conversaciones, cada una de ellas en un estado diferente. Cuando se dice que una conversación está en estado Enviar, se habla desde el punto de vista del programa local. Para el programa asociado, la conversación está en otro estado (como, por ejemplo, Recibir).

Cambios de estado

Puede producirse un cambio en el estado de conversación como consecuencia de lo siguiente:

- Una llamada emitida por el programa local
- Una llamada emitida por el programa asociado
- Una condición de error

Comprobaciones de estado

Se produce una comprobación de estado cuando un programa emite una llamada CPI-C y la conversación no está en el estado apropiado. Por ejemplo, se produciría una comprobación de estado si un programa emitiera la llamada Send_Data mientras la conversación estuviera en estado Recibir. Cuando se produce una comprobación de estado, CPI-C no ejecuta la llamada y devuelve la información de comprobación de estado mediante el parámetro *return_code*.

Cambio de los estados de conversación

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

Tabla 5. Cambio de los estados de conversación

Estado	Programa que invoca	Programa invocado	Estado
Restablecer			
	Initialize_Conversation		
Inicializar			
	Set_Sync_Level (*sync_level*=CM_CONFIRM)		
	Allocate		
Enviar			
	Send_Data		
	Prepare_To_Receive		Restablecer
		→	
		Accept_Conversation	
			Recibir
		Receive (*status_received*=CM_CONFIRM_SEND_RECEIVED)	
			Confirmar-Enviar
		Confirmed	
		←	
			Enviar
	(*return_code*=CM_OK)		
Recibir			
		Send_Data	
		Confirm	

Tabla 5. Cambio de los estados de conversación (continuación)

Estado	Programa que invoca	Programa invocado	Estado
		←	
	Receive (*status_received*=CM_CONFIRM_RECEIVED)		
Confirmar			
	Request_To_Send Confirmed		
		→	
Recibir			
		(*return_code*=CM_OK) (*request_to_send_received*= CM_REQ_TO_SEND_RECEIVED) Prepare_To_Receive	
		←	
	Receive (*status_received*=CM_CONFIRM_SEND_RECEIVED)		
Confirmar-Enviar			
	Confirmed		
		→	
Enviar			
		(*return_code*=CM_OK)	
			Recibir
	Send_Data Deallocate		
		→	
		Receive (*status_received*=CM_CONFIRM_DEALLOC_RECEIVED)	
			Confirmar- Desasignar
		Confirmed	
		←	
			Restablecer
	(*return_code*=CM_OK)		
Restablecer			

Estados iniciales

Antes de asignarse la conversación, el estado de ambos programas es Restablecer.

Después de asignarse la conversación, el estado inicial del programa que invoca es Enviar y el del programa invocado es Recibir.

Cambio al estado Recibir

La llamada Prepare_To_Receive permite a un programa cambiar el estado de la conversación de Enviar a Recibir. Esta llamada lleva a cabo lo siguiente:

- Vacía el almacenamiento intermedio de envío de LU local.
- Si el nivel de sincronización establecido es CM_CONFIRM, la llamada Prepare_To_Receive envía un indicador CM_CONFIRM_SEND al programa asociado mediante el parámetro *status_received* de una llamada Receive. Este indicador informa al programa asociado de que se espera una respuesta Confirmed para que el programa asociado pueda empezar a enviar datos.

Cambio al estado Enviar

El estado de conversación de un programa cambia de Recibir a Enviar cuando su programa asociado empieza a recibir datos (emitiendo la llamada Prepare_To_Receive).

El programa local (para el que la conversación se encuentra en estado Recibir) puede informar al programa asociado de que desea enviar datos, emitiendo la llamada Request_To_Send. Esta petición se comunica al programa asociado mediante el parámetro *request_to_send_received*. (En el ejemplo anterior, este parámetro aparece en la llamada Confirm; también se devuelve a Send_Data y otras llamadas.)

La emisión de la llamada Request_To_Send no cambia el estado de la conversación, ya que el programa asociado puede no tenerla en cuenta. Cuando el programa asociado emite la llamada Prepare_To_Receive, el estado de conversación cambia a Recibir para el programa asociado. El programa local recibe la indicación SEND en un verbo RECEIVE posterior y a partir de ese momento puede enviar datos.

Información complementaria

La información necesaria para que dos programas se comuniquen está almacenada en entradas de información complementaria de CPI-C en el archivo de configuración de Communications Server para Linux. Cada entrada de información complementaria se identifica mediante un nombre de destino simbólico, que es el parámetro *sym_dest_name* especificado por la llamada Initialize_Conversation. El parámetro *sym_dest_name* es una cadena de caracteres ASCII de 8 bytes y puede contener cualquier carácter visualizable. Contiene los campos siguientes:

- Nombre de LU asociada
- Tipo y nombre de programa asociado
- Nombre de modalidad
- Tipo de seguridad de conversación (consulte “Conversaciones múltiples” en la página 13)
- Identificador de usuario y contraseña de seguridad necesarios para acceder al programa asociado

CPI-C también proporciona dos mecanismos para que una aplicación altere temporalmente las entradas de información complementaria configuradas, tal como se describe a continuación. Ambos mecanismos afectan sólo a la utilización de dicha información por parte de la aplicación y no modifican la versión original almacenada en el archivo de configuración.

- La aplicación puede utilizar las llamadas Set_CPIC_Side_Information, Extract_CPIC_Side_Information y Delete_CPIC_Side_Information para gestionar su propia copia local de las entradas completas de información complementaria. (Estas funciones no están disponibles en CPI-C para Java.)
- La aplicación puede utilizar las funciones CPI-C Set_* (como Set_Partner_LU_Name) para alterar temporalmente un parámetro concreto de la información complementaria antes de asignar la conversación.

Para ver más información, consulte “Información complementaria” en la página 31.

Conversaciones básicas

Los programas de servicio normalmente utilizan conversaciones básicas. Éstos 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.

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 obtener más información sobre la sintaxis de GDS, consulte la publicación *IBM Systems Network Architecture: Formats*.

El TP emisor debe dar formato a los datos en varios registros lógicos y el TP receptor debe decodificar los registros lógicos en datos que puedan utilizarse.

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 con una longitud de 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 siguientes constan de estos campos:

- Un campo de longitud de registro (LL) de 2 bytes
- Un campo de datos con una longitud de entre 0 y 32.765 bytes

La longitud registrada en el 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 datos tiene el valor `0x0004` para el campo LL. El campo LL debe tener el formato de más significativo a menos significativo, y no el formato de bytes intercambiados. Por ejemplo, la 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 diez bytes de datos (cada byte de datos tiene el valor `DD`) dividido en tres segmentos de GDS. Los segmentos primero y segundo contienen cuatro bytes de datos cada uno y el último segmento contiene dos bytes de datos.

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

No son válidos los valores siguientes para el campo LL:

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

Datos de anotaciones de error

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

Conversaciones múltiples

Un programa puede participar en varias conversaciones de forma simultánea. Cada conversación requiere una sesión LU-LU. Las conversaciones múltiples no están soportadas si la aplicación utiliza una LU dependiente (para ver más información, consulte "Especificación de la LU local" en la página 35).

Las conversaciones múltiples habitualmente se utilizan para que un programa invocado invoque otro programa que, a su vez, invoca otro programa, y así sucesivamente. En el diagrama siguiente, el programa A invoca al programa B y el programa B invoca al programa.

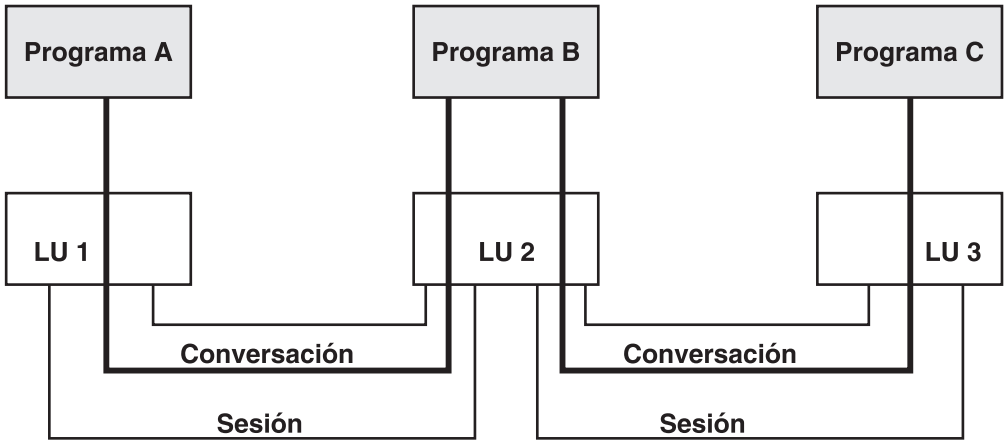

Figura 2. Conversaciones múltiples

Para ver más información sobre cómo funciona la seguridad de conversación de CPI-C con las conversaciones múltiples, 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 programa que invoca proporcione un identificador de usuario y una contraseña para que CPI-C 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. En tal caso, el TP que invoca debe proporcionar un identificador de usuario y una contraseña al asignar una conversación con el programa invocado. Estos datos se obtienen de la información complementaria, o el programa que invoca los especifica explícitamente, y deben coincidir con un identificador de usuario y una contraseña configurados para el programa que invoca.

Communications Server para Linux también da soporte a la seguridad de sesiones LU-LU, que proporciona comprobación de seguridad cuando se inicia la sesión entre las LU locales y remotas. La seguridad de sesiones LU-LU se especifica

durante la configuración y no requiere ninguna acción en los programas CPI-C. Para obtener más información, consulte el manual *Communications Server para Linux, Guía de administración*.

Seguridad de conversación para conversaciones múltiples

En el ejemplo que se muestra en "Conversaciones múltiples" en la página 13, cuando el programa A invoca el programa B y a continuación B invoca C como consecuencia de la conversación con A, la configuración de C puede indicar que aceptará una indicación de seguridad "already-verified" (ya verificada). En este caso, el identificador de usuario y la contraseña facilitados por A tienen que verificarse con la configuración de B. En cambio, cuando B invoca C, establece la característica *security_type* (tipo de seguridad) de la conversación en "same" (la misma) y CPI-C envía a C el identificador de usuario proporcionado por A y una indicación de que ya se ha verificado la seguridad. Para ver más información, consulte "Set_Conversation_Security_Type (cmscst)" en la página 121.

AIX, LINUX

Si el programa participa en más de un par de conversaciones entrantes y salientes de esta forma, tiene que indicar qué conversación entrante proporcionará el identificador de usuario para una conversación saliente. Para ello, CPI-C asocia a cada conversación un identificador de contexto específico. Éste se asigna y se utiliza de la forma siguiente:

- Cada vez que el programa emite correctamente Accept_Conversation o Accept_Incoming, CPI-C asigna un nuevo identificador de contexto a la conversación. El programa puede determinar el valor de este identificador de contexto emitiendo Extract_Conversation_Context con el identificador de conversación adecuado.
- El "contexto actual" del programa normalmente es el identificador de contexto asociado a la llamada Accept_Conversation o Accept_Incoming más reciente. El programa puede utilizar Set_Conversation_Context para establecer el contexto actual en el identificador de contexto de otra de sus conversaciones entrantes (conforme a la restricción descrita a continuación).
- Las llamadas Allocate se emiten en el contexto actual del programa. Esto significa que, si el tipo de seguridad de conversación es "same", el identificador de usuario de la conversación entrante asociada al identificador de contexto actual se enviará al programa asociado.

En el ejemplo anterior, el programa B debe comprobar, antes de emitir la llamada Allocate al programa C, que su contexto actual sea el contexto asociado a la conversación entrante del programa A. De esta forma se asegura que el identificador de usuario de A se envíe en la petición de asignación al programa C. El contexto actual normalmente será el correcto, salvo que B haya emitido otra llamada Accept_Conversation, Accept_Incoming o Set_Conversation_Context desde la aceptación de la conversación de A.

Cuando un programa utiliza Set_Conversation_Context para cambiar el contexto actual, Communications Server para Linux no conserva la información del contexto anterior a menos que siga existiendo una conversación activa asociada al mismo. Esto significa que, si B finaliza la conversación con A y después cambia su contexto actual para comunicarse con otro programa, no podrá volver al primer identificador de contexto para asignar la conversación con C. Si necesita finalizar la

conversación con A antes de asignar la conversación con C, debe asignar la conversación con C antes de cambiar su contexto actual por otro valor.

Seguridad de conversación ya verificada

AIX, LINUX

En algunos casos, un programa puede necesitar indicar la seguridad "ya verificada" cuando no ha sido invocado por otro programa 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 ésta de la siguiente manera:

- Si el programa que especifica la seguridad "already verified" (ya verificada) ha sido invocado por otro programa, tal como se describe en "Seguridad de conversación para conversaciones múltiples" en la página 14, CPI-C envía el identificador de usuario del contexto de conversación actual.
- De lo contrario, CPI-C toma el nombre de usuario de AIX o Linux con el que se ejecuta el programa, truncado a 10 caracteres si es necesario, y lo utiliza como el ID 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 programa que se invoca.
- Si la aplicación utiliza un método diferente para obtener la información de seguridad (por ejemplo, si solicita al usuario que especifique explícitamente un ID de usuario y una contraseña, en vez de confiar en la seguridad del sistema AIX o Linux), entonces puede utilizar cualquiera de las funciones de CPI-C, Set_Conversation_Security_User_ID o Set_CPIC_Side_Information para especificar este *user_id* en CPI-C antes de asignar la conversación.

Funcionamiento sin bloqueo

AIX, LINUX

Esta sección no se aplica a CPI-C de Java. Las funciones de CPI-C para Java siempre funcionan en modalidad con bloqueo; es decir, la función no devuelve el control a la aplicación hasta que el proceso solicitado se haya completado.

Por omisión, las funciones CPI-C operan en la modalidad con bloqueo; es decir, la función no devuelve el control a la aplicación hasta que el proceso solicitado haya finalizado. Por ejemplo, la función Confirm no vuelve hasta que CPI-C ha enviado una petición de confirmación a la aplicación asociada y ha recibido de ella una respuesta de ejecución correcta o de error.

Funcionamiento sin bloqueo

Las funciones CPI-C también pueden operar en la modalidad sin bloqueo; es decir, la función devuelve el control a la aplicación de inmediato, aunque el proceso solicitado no haya finalizado. De esta forma, la aplicación puede continuar con otro proceso que no esté relacionado con esta conversación y obtener los resultados del proceso del verbo posteriormente.

AIX, LINUX

La aplicación puede utilizar la función Check_For_Completion para determinar si ha finalizado una función sin bloqueo anterior o Wait_For_Conversation para esperar a que finalice. La Tabla 6 muestra un ejemplo de uso de la modalidad sin bloqueo.

Tabla 6. Funcionamiento sin bloqueo

Programa que invoca		Programa invocado
Initialize_Conversation Allocate Send_Data Set_Processing_Mode (CM_NON_BLOCKING) Confirm	→	
(*return_code*=CM_OPERATION_INCOMPLETE) (La aplicación puede realizar otro proceso no relacionado con esta conversación.)		Accept_Conversation
		Receive (*data_received*=CM_COMPLETE_DATA_RECEIVED) (*status_received*=CM_CONFIRM_RECEIVED)
Wait_For_Conversation (La aplicación se suspende hasta que finaliza el proceso de la función Confirm anterior.)		
		Confirmed
(Wait_For_Conversation vuelve, *return_code*=CM_OK, conversation_return_code=CM_OK) Send_Data Deallocate	←	
(*return_code*=CM_OPERATION_INCOMPLETE) (La aplicación realiza otro proceso no relacionado con esta conversación.)	→	
		Receive (*status_received*= CM_CONFIRM_DEALLOC_RECEIVED) Confirmed
Check _For_Completion (*return_code*=CM_OK) Wait_For_Conversation (*return_code*=CM_OK, conversation_return_code=CM_OK) (La conversación ahora está desasignada.)	←	

Los pasos siguientes describen el proceso mostrado en el ejemplo anterior.

1. Tras asignar la conversación y enviar algunos datos, el programa que invoca emite Set_Processing_Mode para establecer la modalidad de proceso en CM_NON_BLOCKING. Esto indica que las funciones posteriores de esta conversación pueden operar en la modalidad sin bloqueo.
2. A continuación el programa que invoca emite Confirm, que devuelve CM_OPERATION_INCOMPLETE. Esto indica que la función se ha emitido correctamente y opera en modalidad sin bloqueo.
3. El programa puede llevar a cabo otro proceso que no esté relacionado con esta conversación, como por ejemplo emitir funciones CPI-C en otras conversaciones. También puede emitir una serie limitada de funciones CPI-C por esta conversación (como, por ejemplo, las funciones Extract_*). Es diferente de la especificación CPI-C 2.0 de IBM, en la que el programa no puede emitir ninguna función en esta conversación, excepto Wait_For_Conversation o Cancel_Conversation.
4. Más adelante, el programa emite Wait_For_Conversation para esperar a que finalice la función sin bloqueo anterior. Dado que el programa asociado todavía no ha emitido Confirmed, el proceso de la función Confirm anterior no ha finalizado, por lo que se suspende el programa que invoca.
5. Cuando el programa asociado emite Confirmed, esta llamada finaliza el proceso de la función Confirm por parte del programa que invoca. A continuación vuelve la función Wait_For_Conversation. El parámetro *return_code* CM_OK indica que Wait_For_Conversation ha finalizado correctamente; parámetro *return_code* CM_OK de la conversación indica que la función Confirm (que se estaba esperando) ha finalizado correctamente.
6. Tras enviar datos adicionales, el programa que invoca emite Deallocate, que devuelve CM_OPERATION_INCOMPLETE. Esto indica que la función se ha emitido correctamente y opera en modalidad sin bloqueo. Como anteriormente, el programa puede llevar a cabo otro proceso que no esté relacionado con esta conversación, pero no puede emitir la mayoría de las funciones CPI-C en esta conversación.
7. El programa asociado recibe la petición Deallocate y responde con Confirmed. De esta forma se finaliza el proceso de la función Deallocate.
8. El programa que invoca emite Check_For_Completion para determinar si ha finalizado alguna de las funciones sin bloqueo anteriores de alguna de sus conversaciones. Como el proceso de Deallocate ya ha finalizado, Check_For_Completion vuelve con el identificador de conversación (*conversation_ID*) de esta conversación.
9. A continuación, el programa emite Wait_For_Conversation para obtener el resultado del proceso de Deallocate. Esta función vuelve de inmediato ya que el proceso de Deallocate ya ha finalizado.

WINDOWS

La aplicación debe utilizar la función Specify_Windows_Handle antes de emitir algún verbo en modalidad sin bloqueo. Esta función especifica un descriptor de Windows al que CPI-C envía un mensaje cuando el proceso del verbo ha finalizado. Este mensaje notifica a la aplicación que el verbo se ha completado; no es necesario que la aplicación emita una llamada adicional para esperar los resultados del proceso del verbo.

CPI-C puede utilizar un método alternativo para indicar que el verbo ha completado la señalización de un descriptor de sucesos. Si la aplicación registra un suceso con la conversación utilizando WinCPICSetEvent, la aplicación puede

invocarlas funciones de Windows `WaitForSingleObject` o `WaitForMultipleObjects` para esperar a que se le notifique la finalización del verbo.

Si la llamada pendiente es una llamada Receive, la aplicación puede emitir las siguientes llamadas mientras Receive está pendiente:

- Request_To_Send
- Send_Error
- Test_Request_to_Send_Received
- Cancel_Conversation
- Deallocate

Como alternativa al uso de Specify_Windows_Handle o `WinCPICSetEvent` tal como se ha descrito anteriormente, la aplicación puede utilizar Wait_For_Conversation, igual que para los sistemas Linux. Esta función se proporciona para que los sistemas Windows contribuyan a la migración de aplicaciones desde otros entornos de sistemas operativos. Sin embargo, no es nada aconsejable utilizar funciones de bloqueo, como por ejemplo, Wait_For_Conversation en el entorno Windows. Si escribe una aplicación nueva específicamente para el entorno Windows, utilice Specify_Windows_Handle en vez de Wait_For_Conversation.

Nota:

- Check_For_Completion, descrito anteriormente para sistemas AIX o Linux, no está soportado en sistemas Windows.
- Si la aplicación utiliza una de las llamadas listadas anteriormente en modalidad sin bloqueo mientras Receive está pendiente, debe utilizar Specify_Windows_Handle. No debe utilizar Wait_For_Conversation si hay otra llamada pendiente además de Receive; los resultados de esta llamada no están definidos si hay más de una llamada pendiente en la misma conversación.

CPI-C y LU 6.2

Las aplicaciones CPI-C pueden comunicarse con aplicaciones de LU 6.2 que no utilicen CPI-C, como por ejemplo APPC.

CPI-C no soporta las siguientes características incluidas en algunas implementaciones de LU 6.2:

- Proceso de punto de sincronización/restitución
- Datos PIP
- LOCKS=LONG
- MAP_NAME
- FMH_DATA

No deben utilizarse estas funciones en las aplicaciones de LU 6.2 si se va a usar CPI-C para comunicarse con ellas.

Capítulo 2. Desarrollo de aplicaciones CPI-C

Este capítulo contiene información necesaria para el desarrollo de programas de aplicación CPI-C. Se tratan los temas siguientes:

- Resumen de llamadas CPI-C
- Características iniciales de la conversación
- Información complementaria
- Configuración
- Especificación del nombre de TP y el nombre de LU local para un programa CPI-C
- Cómo se inician los programas

AIX, LINUX

- consideraciones sobre AIX o Linux
- Consideraciones sobre CPI-C de Java

WINDOWS

- Consideraciones sobre Windows

- Desarrollo de aplicaciones portables

Resumen de llamadas CPI-C

Este apartado describe brevemente cada una de las llamadas CPI-C. Están agrupadas por función. Para ver una descripción más detallada de una llamada concreta, consulte el Capítulo 3, “Llamadas CPI-C”, en la página 49.

Los “nombres” de las llamadas son seudónimos. Los nombres reales de las funciones en C aparecen entre paréntesis tras el seudónimo. Por ejemplo, Initialize_Conversation es el seudónimo de una llamada. El nombre real de la función es cminit.

También puede ser necesario establecer los nombres de TP y LU locales que utilizará el programa. Para ver más información al respecto, consulte “Especificación del nombre de TP local” en la página 34 y “Especificación de la LU local” en la página 35.

Inicio de una conversación

Las llamadas siguientes se utilizan para iniciar una conversación entre dos programas. Para ver más información sobre esta cuestión, consulte “Cómo se inician los programas” en la página 37.

También es posible que tenga que establecer el nombre de TP y el nombre de LU locales que utilizará el programa. Para ver información sobre cómo establecer estos valores, consulte “Especificación del nombre de TP local” en la página 34 y “Especificación de la LU local” en la página 35.

WinCPICStartup

WINDOWS

Esta llamada registra la aplicación como una aplicación CPI-C de Windows y determina si el software CPI-C soporta el nivel de función que la aplicación necesita. Una aplicación CPI-C de Windows debe utilizar esta llamada antes de emitir otras llamadas de CPI-C.

Initialize_Conversation (cminit)

El programa que invoca emite esta llamada para obtener un identificador de conversación y definir los valores iniciales de las características de la conversación. Los valores iniciales proceden de la información complementaria asociada al nombre de destino simbólico, o son valores por omisión de CPI-C.

Initialize_For_Incoming (cminic)

El programa que invoca emite esta llamada para obtener un identificador de conversación para una conversación entrante que posteriormente aceptará con Accept_Incoming. Esto permite al programa emitir Accept_Incoming en la modalidad sin bloqueo, en lugar de utilizar Accept_Conversation, que siempre opera en la modalidad con bloqueo.

Llamadas Set_* para cambiar las características iniciales de la conversación

Tras emitir la llamada Initialize_Conversation, el programa que invoca puede cambiar las características iniciales de la conversación emitiendo cualquiera de las llamadas que figuran en la Tabla 7. Estas llamadas sólo pueden emitirse en estado Inicializar.

Tabla 7. Llamadas Set_* para cambiar las características iniciales de la conversación

Llamada	Establece
Set_Conversation_Type (cmsct)	Tipo de conversación
Set_Mode_Name (cmsmn)	Nombre de modalidad
Set_Partner_LU_Name (cmspln)	Nombre de LU asociada
Set_TP_Name (cmstpn)	Nombre de TP del programa asociado
Set_Return_Control (cmsrc)	Control de retorno
Set_Sync_Level (cmssl)	Nivel de sincronización
AIX, LINUX	
Set_Conversation_Context (cmsctx)	Contexto de la conversación (agrupa esta conversación con otra anterior)
Set_Conversation_Security_Type (cmscst)	Tipo de seguridad de conversación
Set_Conversation_Security_User_ID (cmscsu)	Identificador de usuario de seguridad
Set_Conversation_Security_Password (cmscsp)	Contraseña de seguridad

Allocate (cmallc)
El programa que invoca emite esta llamada para asignar una conversación con el programa asociado, utilizando las características actuales de la conversación. El tipo de conversación asignado depende de la característica de tipo de conversación (correlacionada o básica).

Accept_Conversation (cmaccp)
El programa invocado emite esta llamada para aceptar la conversación entrante y definir determinadas características de conversación. Una vez ejecutada correctamente esta llamada, CPI-C genera y devuelve un identificador de conversación. Accept_Conversation siempre opera en la modalidad con bloqueo.

Accept_Incoming (cmacci)

AIX, LINUX

El programa invocado emite esta llamada para aceptar una conversación entrante para la que anteriormente ha emitido Initialize_For_Incoming. Es parecida a Accept_Conversation, pero puede operar en la modalidad sin bloqueo si es necesario (Accept_Conversation siempre opera en la modalidad con bloqueo).

Envío de datos
Las llamadas siguientes se utilizan para enviar datos al programa asociado.

Set_Send_Type (cmsst)
Esta llamada establece el tipo de envío de la conversación. El tipo de envío especifica cómo enviará los datos la llamada Send_Data. La llamada Send_Data puede incluir la función de la llamada Flush, Confirm, Prepare_To_Receive o Deallocate (que equivale a emitir Send_Data, seguida de la otra llamada) o puede simplemente enviar datos sin llevar a cabo ninguna otra función. El valor de tipo de envío afecta a todas las llamadas Send_Data posteriores. Puede cambiarse volviendo a emitir la llamada Set_Send_Type.

Send_Data (cmsend)
Esta llamada coloca datos en el almacenamiento intermedio de envío de la LU local para transmitirlos al programa asociado.

Si el tipo de envío (especificado por la llamada Set_Send_Type) incluye la función de la llamada Flush, Confirm, Prepare_To_Receive o Deallocate, los datos se transmiten a la LU asociada (y al programa asociado) de inmediato. De lo contrario, los datos se acumulan en el almacenamiento intermedio de envío de la LU local y se envían cuando se produce una de las situaciones siguientes:

- El almacenamiento intermedio de envío se llena.
- El programa local emite una de las llamadas siguientes, que vacían el almacenamiento intermedio de envío de la LU:
  - Flush
  - Confirm
  - Deallocate
  - Prepare_To_Receive
  - Receive (con el tipo de recepción establecido en `CM_RECEIVE_AND_WAIT`)

Flush (cmflus)
Esta llamada envía el contenido del almacenamiento intermedio de envío de la LU local a la LU asociada (y al programa asociado). Si el almacenamiento intermedio de envío está vacío, no se lleva a cabo ninguna acción.

Confirm (cmcfm)
Esta llamada envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al programa asociado y espera una confirmación.

Request_To_Send (cmrts)
Esta llamada notifica al programa asociado que el programa local desea enviar datos. El programa asociado puede responder a esta petición cambiando al estado Recibir para que el programa local cambie al estado Enviar o hacer caso omiso de la petición.

Recepción de datos
Las llamadas siguientes permiten a un programa recibir datos de su programa asociado.

Set_Prepare_To_Receive_Type (cmsptr)
Esta llamada establece el tipo de preparación para recepción de la conversación, que especifica si las llamadas Prepare_To_Receive posteriores incluirán la función Flush o Confirm. El tipo de preparación para recepción afecta a todas las llamadas Prepare_To_Receive posteriores. Puede cambiarse volviendo a emitir la llamada Set_Prepare_To_Receive_Type.

Prepare_To_Receive (cmptr)
Esta llamada cambia el estado de la conversación para el programa local de Enviar a Recibir, lo que permite al programa local empezar a recibir datos. Antes de cambiar el estado de conversación, esta llamada realiza el equivalente de la llamada Flush o Confirm.

Set_Receive_Type (cmsrt)
Esta llamada establece el tipo de recepción de la conversación, que especifica si un programa que emite una llamada Receive esperará a que lleguen los datos si no hay datos disponibles. El valor de tipo de recepción afecta a todas las llamadas Receive posteriores. Puede cambiarse volviendo a emitir la llamada Set_Receive_Type.

Receive (cmrcv)
Si se emite esta llamada mientras la conversación está en estado Recibir, el programa local recibe del programa asociado los datos que están actualmente disponibles. Si no hay datos disponibles y el tipo de recepción establecido es `CM_RECEIVE_AND_WAIT`, el programa local espera a que lleguen los datos. Si el tipo de recepción establecido es `CM_RECEIVE_IMMEDIATE`, el programa no espera.

Sólo se puede emitir esta llamada mientras la conversación se encuentra en estado Enviar o Enviar-Pendiente si el tipo de recepción establecido es `CM_RECEIVE_AND_WAIT`. De esta forma se vacía el almacenamiento intermedio de envío de la LU y se cambia el estado de conversación a Recibir. A continuación, el programa local empieza a recibir datos.

Set_Fill (cmsf)
Esta llamada establece el tipo de relleno de la conversación, que especifica si los programas recibirán los datos en forma de registros lógicos o como una longitud

de datos especificada. Sólo tiene efecto en las conversaciones básicas. El valor de fill afecta a todas las llamadas Receive posteriores. Puede cambiarse volviendo a emitir la llamada Set_Fill.

Conversión de datos entre ASCII y EBCDIC

Las llamadas siguientes permiten a un programa convertir los datos locales de ASCII a EBCDIC antes de enviarlos al programa asociado o convertir los datos recibidos del programa asociado de EBCDIC a ASCII. El programa únicamente necesita utilizar estas funciones si el programa asociado precisa que los datos estén en formato EBCDIC.

Convert_Incoming (cmcnvi)

Esta llamada convierte una cadena de datos EBCDIC a ASCII.

Convert_Outgoing (cmcnvo)

Esta llamada convierte una cadena de datos ASCII a EBCDIC.

WINDOWS

El programa también puede utilizar el verbo CSV CONVERT para convertir datos entre ASCII y EBCDIC. Para más información, consulte la publicación *Communications Server para Linux, Guía del programador para CSV*.

Confirmación de recepción de datos e información de errores

Las llamadas siguientes confirman la recepción de datos o informan de un error.

Confirmed (cmcfmd)

Esta llamada responde a una petición de confirmación del programa asociado. Informa al programa asociado de que el programa local no ha detectado ningún error en los datos recibidos. Como el programa que emite la petición de confirmación espera una confirmación, la llamada Confirmed sincroniza el proceso de los dos programas.

Set_Error_Direction (cmsed)

Esta llamada especifica si un programa ha detectado un error al recibir datos o prepararse para enviar datos. La dirección de error sólo es relevante cuando un programa emite la llamada Send_Error en estado Enviar-Pendiente.

Set_Log_Data (cmsld)

Esta llamada especifica un mensaje de anotaciones (datos de anotaciones) y la longitud del mensaje que se enviará a la LU asociada. Esta llamada sólo tiene efecto en las conversaciones básicas. En caso de existir, los datos de anotaciones se envían cuando se emite la llamada Send_Error o cuando se desasigna la conversación de forma anormal. Una vez enviados los datos de anotaciones, CPI-C los restablece en nulos y la longitud de los datos de anotaciones en `0` (cero).

Send_Error (cmserr)

Esta llamada notifica al programa asociado que el programa local ha encontrado un error a nivel de aplicación. El programa local puede utilizar la llamada Send_Error para fines tales como informar al programa asociado de un error

encontrado en los datos recibidos, rechazar una petición de confirmación o truncar un registro lógico incompleto que está enviando.

Emisión de llamadas en modalidad sin bloqueo

AIX, LINUX

Esta sección no se aplica a CPI-C de Java. Las funciones de CPI-C para Java siempre funcionan en modalidad con bloqueo; es decir, la función no devuelve el control a la aplicación hasta que el proceso solicitado se haya completado. Las funciones descritas en este apartado no están disponibles en CPI-C de Java.

Las llamadas siguientes permiten al programa especificar que las llamadas CPI-C posteriores pueden operar en la modalidad sin bloqueo, comprobar si una llamada sin bloqueo anterior ha finalizado o esperar a que finalice una llamada sin bloqueo.

Para información detallada sobre cómo utilizar la modalidad sin bloqueo, consulte los apartados “Consideraciones sobre AIX o Linux” en la página 38 y “consideraciones sobre Windows” en la página 43. (Consulte asimismo el apartado “Cancel_Conversation (cmcanc)” en la página 26; esta llamada cancela una llamada sin bloqueo anterior y también desasigna la conversación.)

Set_Processing_Mode (cmspm)

Esta llamada establece la modalidad de proceso de la conversación en la modalidad con bloqueo (las llamadas no vuelven hasta que haya finalizado el proceso) o en la modalidad sin bloqueo (las llamadas pueden volver de inmediato aunque no haya finalizado el proceso).

Check_For_Completion (cmchck)

AIX, LINUX

Esta llamada comprueba si hay una función sin bloqueo pendiente en alguna de las conversaciones del programa cuyo proceso ha finalizado. Si existe, devuelve el identificador de conversación de la conversación correspondiente; a continuación el programa llama Wait_For_Conversation para obtener los resultados de la función sin bloqueo. Esta llamada permite al programa comprobar la finalización de las funciones sin bloqueo sin quedar suspendidas (a diferencia de Wait_For_Conversation, que está en suspensión hasta que la función ha finalizado). Check_For_Completion no devuelve los resultados de la llamada anterior; para ello el programa debe utilizar Wait_For_Conversation antes de poder emitir más llamadas en esta conversación.

Wait_For_Conversation (cmwait)

Esta llamada espera a que finalice el proceso de una función sin bloqueo anterior. Si el programa participa en varias conversaciones simultáneas, esta llamada actúa en todas las conversaciones y vuelve cuando finaliza una función en cualquiera de ellas.

WINDOWS

La llamada Wait_For_Conversation recibe soporte en los sistemas Windows para obtener la compatibilidad con otras implementaciones CPI-C de Windows; sin embargo, nuevas aplicaciones de Windows deben utilizar Specify_Windows_Handle (se describe más abajo) en lugar de esta llamada.

Specify_Windows_Handle (xchwnd)

Esta llamada especifica un descriptor de Windows en el que CPI-C anota los resultados de las funciones sin bloqueo. La aplicación recibe un mensaje de CPI-C, lo envía a este descriptor de Windows, cuando una función sin bloqueo se completa; no necesita utilizar Wait_For_Conversation para obtener los resultados de la finalización del verbo.

Emisión de llamadas en modalidad con bloqueo

Las llamadas siguientes permiten que un programa de Windows gestione cómo funcionan las llamadas CPI-C posteriores en modalidad con bloqueo. (Consulte asimismo “Set_Processing_Mode (cmspm)” en la página 24; especifica si las llamadas posteriores funcionan en modalidad con bloqueo o modalidad sin bloqueo). Para obtener más información sobre las llamadas de bloqueo, consulte “Llamadas de bloqueo” en la página 44.

WinCPICIsBlocking
: Comprueba si hay una llamada CPI-C de bloqueo pendiente para esta aplicación.

WinCPICSetBlockingHook
: Especifica el procedimiento de bloqueo que CPI-C utiliza mientras procesa las llamadas de bloqueo; esto sustituye al procedimiento de bloqueo por omisión de CPI-C. Se invoca el procedimiento de bloqueo reiteradamente hasta que CPI-C haya finalizado el proceso de la llamada.

WinCPICUnhookBlockingHook
: Elimina el registro del procedimiento de bloqueo especificado por una llamada `WinCPICSetBlockingHook` anterior, de modo que CPI-C vuelve a utilizar el procedimiento de bloqueo por omisión.

Obtención de información

Las llamadas siguientes proporcionan información a los programas.

Llamadas Extract_*

Las llamadas Extract_*, que figuran en la Tabla 8, recuperan información sobre las características de una conversación especificada.

Tabla 8. Llamadas Extract_* y acciones

Llamada	Recupera
Extract_Conversation_Security_Type (xcecst)(no está disponible en CPI-C de Java)	Tipo de seguridad
Extract_Conversation_State (cmecs)	Estado de conversación
Extract_Conversation_Type (cmect)	Tipo de conversación

Tabla 8. Llamadas Extract_* y acciones (continuación)

Llamada	Recupera
AIX, LINUX	
Extract_Conversation_Context (cmectx)	Contexto de la conversación
Extract_Max_Buffer_Size (cmembs)	Tamaño máximo del almacenamiento intermedio de datos utilizado para las llamadas Send_Data y Receive
Extract_Security_User_ID (cmesui)	Identificador de usuario de seguridad
WINDOWS	
Extract_Conversation_Security_User_ID (cmecsu)	Identificador de usuario de seguridad
Extract_Mode_Name (cmemn)	Nombre de modalidad
Extract_Partner_LU_Name (cmepln)	Nombre de LU asociada
Extract_TP_Name (cmetpn)	Nombre de TP especificado con la petición Allocate entrante
Extract_Sync_Level (cmesl)	Nivel de sincronización

Test_Request_to_Send_Received (cmtrts)

Esta llamada determina si se ha recibido una notificación de petición de envío del programa asociado.

Finalización de una conversación

Las llamadas siguientes finalizan una conversación.

Set_Deallocate_Type (cmsdt)

Esta llamada especifica cómo debe desasignarse la conversación. Las instrucciones de desasignación especificadas por esta llamada entran en vigor cuando se emite la llamada Deallocate o cuando el tipo de envío establecido es `CM_SEND_AND_DEALLOCATE` y se emite la llamada Send_Data.

Deallocate (cmdeal)

Esta llamada desasigna una conversación entre dos programas. Antes de desasignar la conversación, esta llamada realiza el equivalente de la llamada Flush o Confirm, según el nivel de sincronización y el tipo de desasignación de la conversación actual.

Cancel_Conversation (cmcanc)

Esta llamada cancela las llamadas incompletas de una conversación y desasigna la conversación. (Una llamada incompleta es una llamada que se ha emitido en la modalidad sin bloqueo y ha devuelto `CM_OPERATION_INCOMPLETE`.)

En CPI-C de Java, las llamadas sin bloqueo no están soportadas y por lo tanto, no puede haber una llamada incompleta pendiente. Cancel_Conversation es equivalente a Deallocate con la diferencia de que no escribe datos de anotaciones en el archivo de anotaciones de error local.

WinCPICCleanup

WINDOWS

Esta llamada elimina el registro de la aplicación como una aplicación CPI-C de Windows, después de que haya finalizado de emitir llamadas CPI-C. Una aplicación CPI-C de Windows debe utilizar esta llamada antes de terminar y no debe emitir ninguna otra llamada CPI-C después de haberla emitido.

Administración de la información complementaria

Estas funciones no están disponibles en CPI-C de Java.

Las llamadas resumidas en la Tabla 9 permiten a las aplicaciones CPI-C añadir, sustituir, recuperar o suprimir entradas de información complementaria.

Tabla 9. Llamadas para añadir, sustituir, recuperar o suprimir información complementaria

Llamada	Acción
Set_CPIC_Side_Information (xcmssi)	Añade o sustituye una entrada de información complementaria.
Extract_CPIC_Side_Information (xcmesi)	Recupera una entrada de información complementaria.
Delete_CPIC_Side_Information (xcmdsi)	Suprime una entrada de información complementaria.

Características iniciales de la conversación

CPI-C mantiene un conjunto de valores internos, denominados características, para cada una de las conversaciones. Algunas características afectan al funcionamiento global de la conversación, como por ejemplo el tipo de conversación. Otras inciden en el funcionamiento de llamadas específicas, como por ejemplo el tipo de recepción.

Muchas de estas características derivan inicialmente de la información complementaria almacenada en el archivo de configuración de Communications Server para Linux; consulte el apartado “Información complementaria” en la página 31. La llamada Initialize_Conversation especifica el nombre de destino simbólico (el parámetro *sym_dest_name*) asociado a la entrada de la tabla de información complementaria deseada.

La Tabla 10 en la página 28 indica las características de conversación, cómo se establecen o modifican con las siguientes llamadas de inicio de conversación y qué llamada puede cambiar un valor determinado.

- Initialize_Conversation
- Accept_Conversation
- Initialize_For_Incoming
- Accept_Incoming

AIX, LINUX

Las llamadas Initialize_For_Incoming y Accept_Incoming siempre se usan juntas. Normalmente una característica se establece mediante una de estas llamadas y no se cambia con la otra.

WINDOWS

Las llamadas Initialize_For_Incoming y Accept_Incoming no están soportadas en los sistemas Windows. Deben ignorarsee todas las referencias a estas llamadas en los sistemas Windows.

Para ver completa información sobre una característica, consulte la descripción de la llamada Set_* asociada a ella en el Capítulo 3, "Llamadas CPI-C", en la página 49. Por ejemplo, el tipo de conversación se describe en el apartado de la llamada Set_Conversation_Type.

Tabla 10. Cambio de las características iniciales de la conversación

Estado de conversación	
Initialize_Conversation establece:	CM_INITIALIZE_STATE
Accept_Conversation establece:	CM_RECEIVE_STATE
Initialize_For_Incoming establece:	CM_INITIALIZE_INCOMING_STATE
Accept_Incoming establece:	CM_RECEIVE_STATE
Puede cambiarse mediante:	Muchas llamadas CPI-C; consulte los apartados *Cambio de estado* al final de las descripciones de las llamadas CPI-C en el Capítulo 3, "Llamadas CPI-C", en la página 49 para ver información sobre los cambios de estado producidos a consecuencia de la llamada.
Tipo de conversación	
Initialize_Conversation establece:	CM_MAPPED_CONVERSATION
Accept_Conversation establece:	El valor especificado por el programa que invoca.
Initialize_For_Incoming establece:	(No definido)
Accept_Incoming establece:	El valor especificado por el programa que invoca.
Puede cambiarse mediante:	Set_Conversation_Type
Tipo de desasignación	
Initialize_Conversation establece:	CM_DEALLOCATE_SYNC_LEVEL
Accept_Conversation establece:	CM_DEALLOCATE_SYNC_LEVEL
Initialize_For_Incoming establece:	CM_DEALLOCATE_SYNC_LEVEL
Accept_Incoming establece:	(Sin cambio)
Puede cambiarse mediante:	Set_Deallocate_Type
Dirección de error	
Initialize_Conversation establece:	CM_RECEIVE_ERROR
Accept_Conversation establece:	CM_RECEIVE_ERROR
Initialize_For_Incoming establece:	CM_RECEIVE_ERROR
Accept_Incoming establece:	(Sin cambio)
Puede cambiarse mediante:	Set_Error_Direction
Relleno	
Initialize_Conversation establece:	CM_FILL_LL
Accept_Conversation establece:	CM_FILL_LL

Tabla 10. Cambio de las características iniciales de la conversación (continuación)

Initialize_For_Incoming establece:	CM_FILL_LL
Accept_Incoming establece:	(Sin cambio)
Puede cambiarse mediante:	Set_Fill
Datos de anotaciones	
Initialize_Conversation establece:	Cadena nula
Accept_Conversation establece:	Cadena nula
Initialize_For_Incoming establece:	Cadena nula
Accept_Incoming establece:	(Sin cambio)
Puede cambiarse mediante:	Set_Log_Data
Nombre de LU local	
Initialize_Conversation establece:	El alias de LU local de uno de los distintos orígenes posibles (consulte "Especificación de la LU local" en la página 35).
Accept_Conversation establece:	El alias de LU de la sesión en que ha llegado la petición de inicio de conversación.
Initialize_For_Incoming establece:	(No definido)
Accept_Incoming establece:	El alias de LU de la sesión en que ha llegado la petición de inicio de conversación.
Puede cambiarse mediante:	Set_Local_LU_Name
Nombre de modalidad	
Initialize_Conversation establece:	El nombre de modalidad de la información complementaria, o una cadena nula si no se especifica ningún *sym_dest_name*.
Accept_Conversation establece:	El nombre de modalidad de la sesión en que ha llegado la petición de inicio de conversación.
Initialize_For_Incoming establece:	(No definido)
Accept_Incoming establece:	El nombre de modalidad de la sesión en que ha llegado la petición de inicio de conversación.
Puede cambiarse mediante:	Set_Mode_Name
Nombre de LU asociada	
Initialize_Conversation establece:	El nombre de LU asociada de la información complementaria, o un solo blanco si no se especifica ningún *sym_dest_name*.
Accept_Conversation establece:	El nombre de LU asociada de la sesión en que ha llegado la petición de inicio de conversación.
Initialize_For_Incoming establece:	(No definido)
Accept_Incoming establece:	El nombre de LU asociada de la sesión en que ha llegado la petición de inicio de conversación.
Puede cambiarse mediante:	Set_Partner_LU_Name
Tipo de preparación para recepción	
Initialize_Conversation establece:	CM_PREP_TO_RECEIVE_SYNC_LEVEL
Accept_Conversation establece:	CM_PREP_TO_RECEIVE_SYNC_LEVEL
Initialize_For_Incoming establece:	CM_PREP_TO_RECEIVE_SYNC_LEVEL
Accept_Incoming establece:	(Sin cambio)
Puede cambiarse mediante:	Set_Prepare_To_Receive_Type
Modalidad de proceso (con bloqueo o sin bloqueo)	

Tabla 10. Cambio de las características iniciales de la conversación (continuación)

Initialize_Conversation establece:	CM_BLOCKING
Accept_Conversation establece:	CM_BLOCKING
Initialize_For_Incoming establece:	CM_BLOCKING
Accept_Incoming establece:	(Sin cambio)
Puede cambiarse mediante:	Set_Processing_Mode
Tipo de recepción	
Initialize_Conversation establece:	CM_RECEIVE_AND_WAIT
Accept_Conversation establece:	CM_RECEIVE_AND_WAIT
Initialize_For_Incoming establece:	CM_RECEIVE_AND_WAIT
Accept_Incoming establece:	(Sin cambio)
Puede cambiarse mediante:	Set_Receive_Type
Control de retorno	
Initialize_Conversation establece:	CM_WHEN_SESSION_ALLOCATED
Accept_Conversation establece:	(No aplicable)
Initialize_For_Incoming establece:	(No aplicable)
Accept_Incoming establece:	(No aplicable)
Puede cambiarse mediante:	Set_Return_Control
Contraseña de seguridad	
Initialize_Conversation establece:	La contraseña incluida en la información complementaria, o un solo blanco si no se especifica ningún *sym_dest_name*.
Accept_Conversation establece:	(No aplicable)
Initialize_For_Incoming establece:	(No aplicable)
Accept_Incoming establece:	(No aplicable)
Puede cambiarse mediante:	Set_Conversation_Security_Password
Tipo de seguridad	
Initialize_Conversation establece:	El tipo de seguridad incluido en la información complementaria, o CM_SECURITY_SAME si no se especifica ningún *sym_dest_name*.
Accept_Conversation establece:	(No aplicable)
Initialize_For_Incoming establece:	(No aplicable)
Accept_Incoming establece:	(No aplicable)
Puede cambiarse mediante:	Set_Conversation_Security_Type
Identificador de usuario de seguridad	
Initialize_Conversation establece:	El identificador de usuario incluido en la información complementaria, o un solo blanco si no se especifica ningún *sym_dest_name*.
Accept_Conversation establece:	El valor especificado por el programa que invoca.
Initialize_For_Incoming establece:	(No definido)
Accept_Incoming establece:	El valor especificado por el programa que invoca.
Puede cambiarse mediante:	Set_Conversation_Security_User_ID
Tipo de envío	
Initialize_Conversation establece:	CM_BUFFER_DATA
Accept_Conversation establece:	CM_BUFFER_DATA

Tabla 10. Cambio de las características iniciales de la conversación (continuación)

Initialize_For_Incoming establece:	CM_BUFFER_DATA
Accept_Incoming establece:	(Sin cambio)
Puede cambiarse mediante:	Set_Send_Type
Nivel de sincronización	
Initialize_Conversation establece:	CM_NONE
Accept_Conversation establece:	El valor especificado por el programa que invoca.
Initialize_For_Incoming establece:	(No definido)
Accept_Incoming establece:	El valor especificado por el programa que invoca.
Puede cambiarse mediante:	Set_Sync_Level
Nombre de TP del programa invocado (desde el punto de vista del programa que invoca)	
Initialize_Conversation establece:	El nombre de TP incluido en la información complementaria, o un solo blanco si no se especifica ningún *sym_dest_name*.
Accept_Conversation establece:	(No aplicable)
Initialize_For_Incoming establece:	(No aplicable)
Accept_Incoming establece:	(No aplicable)
Puede cambiarse mediante:	Set_TP_Name
Nombre de TP del programa invocado (desde el punto de vista del programa invocado)	
Initialize_Conversation establece:	(No aplicable)
Accept_Conversation establece:	El valor especificado por el programa que invoca.
Initialize_For_Incoming establece:	(No definido)
Accept_Incoming establece:	El valor especificado por el programa que invoca.
Puede cambiarse mediante:	Specify_Local_TP_Name (para indicar uno o varios nombres para los que se aceptarán peticiones de asignación entrantes)

Información complementaria

La información necesaria para que dos programas se comuniquen está almacenada en entradas de información complementaria de CPI-C en el archivo de configuración de Communications Server para Linux. Tendrá que coordinar su trabajo con el administrador del sistema para asegurarse de que contenga lo que necesita. Para obtener información adicional sobre la configuración, consulte el manual *Communications Server para Linux, Guía de administración*.

Cada entrada de información complementaria se identifica mediante un nombre de destino simbólico, que es el parámetro *sym_dest_name* especificado por la llamada Initialize_Conversation. El parámetro *sym_dest_name* es una cadena de caracteres ASCII de 8 bytes y puede contener cualquier carácter visualizable.

Si desarrolla programas comerciales o programas que se instalarán en varias máquinas dentro de su organización, puede ser recomendable incluir la lógica necesaria para utilizar un *sym_dest_name* distinto para cada copia del programa.

Cada una de las entradas de información complementaria contiene los campos siguientes:

- Alias de LU local

- Nombre de LU asociada
- Tipo y nombre de programa asociado
- Nombre de modalidad
- Tipo de seguridad de conversación
- Identificador de usuario y contraseña de seguridad
- Información complementaria especificada por la aplicación

Alias de LU local

Éste es el alias de la LU local que se utilizará para asignar conversaciones. Consta de hasta 8 caracteres ASCII. Para ver los caracteres permitidos, consulte "Set_Local_LU_Name (cmslln)" en la página 136.

Nombre de LU asociada

Éste es el nombre por el que se conoce la LU asociada en el programa local. Puede ser un alias de hasta 8 caracteres ASCII o un nombre de red completamente calificado de hasta 17 caracteres. Para ver los caracteres permitidos, consulte "Set_Partner_LU_Name (cmspln)" en la página 141.

Tipo y nombre de programa asociado

Estos campos indican si el programa asociado es un programa de aplicación o un programa de servicio SNA, así como el nombre del programa asociado. Un nombre de programa de aplicación puede contener hasta 64 caracteres ASCII. Un nombre de programa de servicio puede contener hasta 4 caracteres. Para ver los caracteres permitidos, consulte "Set_TP_Name (cmstpn)" en la página 153.

Nombre de modalidad

Este nombre representa un conjunto de características que se utilizarán en una sesión LU-LU. El nombre de modalidad puede contener hasta 8 caracteres ASCII. Para ver los caracteres permitidos, consulte "Set_Mode_Name (cmsmn)" en la página 139.

Tipo de seguridad de conversación

Este campo indica si se utilizará la seguridad y, en caso afirmativo, qué tipo de seguridad se usará. El tipo de seguridad puede especificar que CPI-C debe enviar un identificador de usuario y una contraseña al asignar una conversación con el programa invocado. En el caso de un programa invocado que a su vez invoca otro programa, el tipo de seguridad puede informar al segundo programa invocado de que ya se ha verificado la seguridad.

Para ver más información sobre la seguridad de conversación, consulte "Set_Conversation_Security_Type (cmscst)" en la página 121.

Identificador de usuario y contraseña de seguridad

Si el programa remoto utiliza la seguridad de conversación y no acepta una indicación de seguridad "ya verificada", se necesita una combinación válida de identificador de usuario y contraseña para acceder al programa invocado. El identificador de usuario y la contraseña pueden contener hasta 10 caracteres ASCII. Para ver los caracteres permitidos, consulte "Set_Conversation_Security_User_ID (cmscsu)" en la página 123 y "Set_Conversation_Security_Password (cmscsp)" en la página 118.

Información complementaria especificada por la aplicación

AIX, LINUX

Nota: Las funciones que se describen en esta sección no están disponibles en CPI-C de Java. Una aplicación CPI-C de Java no puede mantener sus propias entradas de información complementaria CPI-C. Sin embargo, puede alterar temporalmente parámetros individuales de la información complementaria o determinar sus valores, utilizando las funciones Set_* o Extract_* para cada parámetro necesario.

Una aplicación puede alterar temporalmente la información complementaria almacenada en el archivo de configuración para mantener sus propias entradas de información complementaria, mediante las siguientes entradas:

- Set_CPIC_Side_Information (para definir una entrada de información complementaria asociada a un *sym_dest_name* especificado; si el *sym_dest_name* ya está definido en el archivo de configuración, la nueva información altera temporalmente el archivo de configuración)
- Delete_CPIC_Side_Information (para indicar que una entrada definida por la aplicación, o una entrada definida en el archivo de configuración, ya no está disponible para ser utilizada por la aplicación)
- Extract_CPIC_Side_Information (para devolver el contenido de una entrada de información complementaria, ya sea una entrada definida por la aplicación, ya sea una entrada definida en el archivo de configuración)

A partir de este momento la información modificada se aplica únicamente a esta aplicación; no afecta a otras aplicaciones y no cambia el archivo de configuración. La información modificada se elimina cuando finaliza la aplicación.

Estas llamadas no forman parte de la especificación CPI-C 2.0 de IBM; se facilitan para la compatibilidad con X/Open CPI-C. Además, en la estructura de información complementaria utilizada por estas llamadas, los parámetros de ID de usuario y contraseña se definen como 8 caracteres (como en CPI-C de X/Open) en lugar de 10 (como en CPI-C 2.0) de IBM). Esto da lugar a las restricciones siguientes:

- Si la aplicación asociada necesita un identificador de usuario o una contraseña de más de 8 caracteres, no se puede especificar esta información utilizando Set_CPIC_Side_Information. Se debe utilizar una entrada de información complementaria definida en el archivo de configuración o definir una entrada con Set_CPIC_Side_Information y después alterar temporalmente el identificador de usuario o la contraseña con la llamada Set_Conversation_Security_User_ID o Set_Conversation_Security_Password.
- Si la entrada de información complementaria del archivo de configuración contiene un identificador de usuario de más de 8 caracteres, no se puede extraer esta información utilizando Extract_CPIC_Side_Information. Se debe utilizar la llamada Extract_Security_User_ID. (Esta restricción no se aplica a la contraseña, ya que CPI-C no permite a la aplicación extraerla.)

Configuración

A continuación, se indican algunas consideraciones que deben tenerse en cuenta al configurar Communications Server para Linux:

- Además de mantener la información complementaria (especificada por *sym_dest_name*), el Administrador del sistema debe definir las siguientes entidades durante la configuración para permitir que las aplicaciones CPI-C utilicen los servicios LU 6.2 de Communications Server para Linux:
  - Modalidades
  - LU locales
  - LU asociadas
  - TP invocables
  - Identificadores de usuario y contraseñas de seguridad

  Para obtener más información, consulte la publicación *Communications Server para Linux, Guía de administración*.
- Si desea activar las sesiones de inicio automático, establezca el parámetro *auto_act* en la modalidad. Para obtener más información sobre cómo definir modalidades, consulte la publicación *Communications Server para Linux, Guía de administración*.

Especificación del nombre de TP local

Cuando un programa emite la llamada Initialize_Conversation, Initialize_Conversation_For_Incoming o Accept_Conversation, la biblioteca de CPI-C genera una instancia de un programa de transacción (TP). Puede especificar el nombre de este TP de varias formas distintas, descritas a continuación.

Dichos métodos figuran por orden de prioridad. Esto significa que, si se especifica un nombre mediante el primer método, la biblioteca de CPI-C utiliza este nombre y no tiene en cuenta ningún nombre que se especifique mediante el segundo método o métodos posteriores. Si no se utiliza el primer método pero se especifica un nombre mediante el segundo método, la biblioteca de CPI-C utiliza este nombre y no tiene en cuenta ningún nombre que se especifique mediante el tercer método o métodos posteriores, y así sucesivamente.

- En el caso de los programas que invocan, el nombre de TP sólo se utiliza como identificador en los archivos de anotaciones y de rastreo.
- En el caso de los programas invocados iniciados por el operador, el nombre de TP debe establecerse correctamente ya que ese valor se utiliza para direccionar las peticiones de asignación entrantes al programa adecuado. La llamada Accept_Conversation o Accept_Incoming procedente del programa invocado finaliza cuando llega una petición de asignación entrante para este nombre de TP.
- En el caso de los programas invocados iniciados automáticamente, no es necesario especificar el nombre de TP porque se obtiene de la petición de asignación entrante.

Nota: El nombre de TP local es distinto del nombre de TP asociado establecido en la llamada Set_TP_Name.

Specify_Local_TP_Name

El programa puede utilizar esta llamada para especificar el nombre de TP.

Contexto

Si hay otro TP del que se copia el contexto, el nombre de TP se obtiene de ese otro TP. Para ver más información sobre el contexto, consulte “Conversaciones múltiples” en la página 13.

Variable de entorno APPCTPN

El nombre de TP puede especificarse utilizando la variable de entorno APPCTPN.

AIX, LINUX

En sistemas AIX o Linux el nombre TP se especifica en la variable de entorno APPCTPN. Esta variable de entorno puede definirse de las formas siguientes:

- El programa puede emitir una llamada putenv.
- Puede establecerla en el shell AIX o Linux. Por ejemplo, en el shell Korn emitiría el mandato siguiente:

  export APPCTPN=MYTP
- Si utiliza TP invocados de inicio automático, puede establecerla utilizando el campo de entorno del archivo de datos TP invocable de Communications Server para Linux.

WINDOWS

En sistemas Windows el nombre de TP se puede especificar utilizando la variable de entorno APPCTPN, o en el registro. CPI-C comprueba la variable de entorno en primer lugar y utiliza este nombre si se ha especificado; sólo utiliza la entrada de registro si la variable de entorno no se ha especificado. Tal vez tenga que utilizar variables de entorno si va a utilizar Windows Terminal Server y necesita ejecutar varias copias de la misma aplicación utilizando diferentes LU locales.

La clave de registro es

```
\\HKEY_LOCAL_MACHINE\SOFTWARE\SNA Client\SxClient\Parameters\MyExeName
```

donde MyExeName es el nombre de archivo del programa, sin la extensión **.exe**.

El valor APPCTPN bajo esta clave de registro especifica el nombre de TP.

Valor por omisión

Si no se establece el nombre de TP por ninguno de los métodos descritos en los apartados anteriores, éste se define en el valor por omisión CPIC_DEFAULT_TPNAME.

Especificación de la LU local

La LU local que utiliza un TP CPI-C que invoca puede especificarse de varias formas distintas, tal como se describe a continuación.

Nota: La LU local de un TP invocado no se especifica de esta forma, sino que la define el valor de LU asociada especificado en la petición de asignación.

Si la LU especificada es una LU dependiente, no puede haber varias conversaciones simultáneas (ya que las LU dependientes no soportan las sesiones múltiples).

En los apartados siguientes se describen los distintos métodos que puede utilizar para establecer el alias de LU local. Dichos métodos figuran por orden de prioridad. Esto significa que, si se especifica un alias de LU local mediante el primer método, la biblioteca de CPI-C utiliza este nombre y no tiene en cuenta ningún alias que se especifique mediante el segundo método o métodos posteriores. Si no se utiliza el primer método pero se especifica un alias de LU local mediante el segundo método, la biblioteca de CPI-C utiliza este alias y no tiene en cuenta ningún alias que se especifique mediante el tercer método o métodos posteriores, y así sucesivamente.

Set_Local_LU_Name

El programa puede emitir esta llamada para especificar el alias de LU local después de que haya finalizado la llamada Initialize_Conversation. Esta llamada sólo afecta al TP desde el que se emite. No modifica la información complementaria almacenada en el archivo de configuración.

Nota: Esta llamada no forma parte de la especificación CPI-C estándar y puede no estar disponible en otras implementaciones. Se recomienda evitar el uso de esta función o restringirla a unas cuantas rutinas que se puedan modificar fácilmente, si debe asegurarse de que la aplicación pueda utilizarse con otras implementaciones de CPI-C.

Contexto

Si hay otro TP del que se copia el contexto, el nombre de LU local se obtiene de ese otro TP. Para ver más información sobre el contexto, consulte “Conversaciones múltiples” en la página 13.

Variable de entorno APPCLLU

El alias de LU local puede especificarse utilizando la variable de entorno `APPCLLU`.

AIX, LINUX

En sistemas AIX o Linux esta variable de entorno se puede establecer de las formas siguientes:

- El programa puede emitir una llamada `putenv`.
- Puede establecerla en el shell AIX o Linux. Por ejemplo, en el shell Korn emitiría el mandato siguiente:

  export APPCLLU=MYLU

WINDOWS

En sistemas Windows el alias de LU local se puede especificar utilizando la variable de entorno `APPCLLU`, o en el registro. CPI-C comprueba la variable de entorno en primer lugar y utiliza este alias si se ha especificado; sólo utiliza la

entrada de registro si la variable de entorno no se ha especificado. Tal vez tenga que utilizar variables de entorno si va a utilizar Windows Terminal Server y necesita ejecutar varias copias de la misma aplicación utilizando diferentes LU locales.

La clave de registro es

```
\\HKEY_LOCAL_MACHINE\SOFTWARE\SNA Client\SxClient\Parameters\MyExeName
```

donde MyExeName es el nombre de archivo del programa, sin la extensión **.exe**.

El valor `APPCLLU` bajo esta clave de registro especifica el alias de LU local.

Información complementaria

El alias de LU local es parte de la información complementaria configurada para cada nombre de destino simbólico. Los TP seleccionan cuál de ellos utilizarán en la llamada Initialize_Conversation.

Nota: Los programas pueden modificar la información complementaria. Para ver más información, consulte "Administración de la información complementaria" en la página 27.

LU local por omisión

Las LU locales pueden configurarse como parte de la agrupación de LU APPC por omisión. Si no se especifica ningún otro alias de LU local, se utiliza cualquier LU adecuada de esta agrupación.

LU de punto de control

Normalmente, Communications Server para Linux tiene una LU de punto de control (CP) definida en cada nodo. Si no se define ningún otro alias de LU local, se utiliza la LU de punto de control.

Cómo se inician los programas

Una conversación tiene lugar entre un programa que invoca y un programa invocado. El programa que invoca se inicia mediante un mandato entrado por un usuario o mediante un mandato de proceso por lotes. El programa invocado lo puede iniciar manualmente un usuario o se puede iniciar automáticamente mediante Communications Server para Linux.

Programa invocado de inicio automático

Puede configurarse un programa invocado para que se inicie automáticamente en una de las siguientes condiciones:

- La primera vez que la LU que da servicio al programa invocado recibe una petición de asignación entrante. Un programa que se inicia de esta manera se denomina programa de inicio automático con cola (o TP de inicio automático con cola).

  Si el programa invocado no está en ejecución, la primera petición de asignación entrante lo inicia; se retiene una respuesta a la petición de asignación hasta que se ejecuta la llamada Accept_Conversation o Accept_Incoming en el programa invocado.

Si el programa invocado ya está en ejecución, la petición de asignación entrante espera hasta que el programa invocado emite otra llamada Accept_Conversation o Accept_Incoming o hasta que finaliza su ejecución y puede reiniciarse.

- La primera vez que la LU que da servicio al programa invocado recibe una petición de asignación entrante, se carga una nueva instancia del programa y se inicia. Un programa que se inicia de esta manera se denomina programa de inicio automático sin cola.

En general, la petición de asignación entrante espera hasta que el programa invocado se inicia y emite una llamada Accept_Conversation o Accept_Incoming. Sin embargo, la definición de la LU local del programa invocado incluye un valor de tiempo de espera, por lo que la petición de asignación entrante falla si se alcanza el tiempo de espera antes de que el programa invocado emita una llamada Accept_Conversation o Accept_Incoming.

La definición del TP invocado (en el archivo de datos TP invocable de Communications Server para Linux) incluye un segundo valor de tiempo de espera, que determina cuánto tiempo espera una llamada Accept_Conversation o Accept_Incoming a que llegue una petición de asignación entrante. La llamada falla si se alcanza este valor de tiempo de espera antes de que se reciba una petición de asignación entrante. Este valor de tiempo de espera no se aplica a un programa sin cola, ya que el programa siempre se inicia como respuesta a una petición de asignación entrante y, por consiguiente, siempre hay una pendiente.

Programa invocado iniciado por el usuario

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

Si el usuario inicia el programa que invoca antes de iniciar el programa invocado, la petición de asignación entrante para el programa invocado espera hasta que el programa invocado se inicia y emite una llamada Accept_Conversation o Accept_Incoming. Sin embargo, la definición de la LU local del programa invocado incluye un valor de tiempo de espera, por lo que la petición de asignación entrante falla si se alcanza el tiempo de espera antes de que el programa invocado se inicie y emita una llamada Accept_Conversation o Accept_Incoming.

Si el usuario inicia el programa invocado antes de que el programa que invoca emita la llamada Allocate, la llamada Accept_Conversation o Accept_Incoming emitida por el programa invocado espera una petición de asignación entrante. La definición del TP invocado (en el archivo de datos TP invocable de Communications Server para Linux) incluye un segundo valor de tiempo de espera, que determina cuánto tiempo espera una llamada Accept_Conversation o Accept_Incoming a que llegue una petición de asignación entrante. La llamada falla si se alcanza este valor de tiempo de espera antes de que se reciba una petición de asignación entrante.

Consideraciones sobre AIX o Linux

AIX, LINUX

Esta sección resume la información que debe tenerse en cuenta al desarrollar aplicaciones CPI-C para sistemas AIX o Linux.

Si desarrolla aplicaciones CPI-C de Java, consulte el apartado “Consideraciones sobre CPI-C de Java” en la página 40.

Archivo de cabecera CPI-C

El archivo de cabecera que se va a utilizar con aplicaciones CPI-C es **cmc.h**. Este archivo contiene las definiciones de todos los puntos de entrada CPI-C. También incluye el archivo de cabecera de interfaz común **values_c.h**; estos dos archivos contienen todas las constantes definidas para los valores de parámetro suministrados y devueltos en la interfaz CPI-C. Ambos archivos se almacenan en **/usr/include/sna** (AIX) o **/opt/ibm/sna/include** (Linux).

Procesos múltiples

Si el proceso que ha iniciado la conversación crea un subproceso, el subproceso no puede utilizar el `conversation_ID` devuelto al proceso superior. Sin embargo, puede emitir su propia llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation para obtener su propio `conversation_ID`.

Dos o más instancias del mismo programa se pueden ejecutar como procesos diferentes, pero a cada instancia se le asignará su propio `conversation_ID`.

Puede desarrollar una aplicación en la que un proceso contenga muchas conversaciones, cada una de ellas con su propio `conversation_ID`. Sin embargo, tiene que diseñar la aplicación con cuidado para evitar situaciones “de punto muerto”, en las que una llamada CPI-C no puede finalizar a causa del estado de otras conversaciones 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 conversación.

Compilación y enlace de la aplicación CPI-C

Aplicaciones AIX

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

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

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

-bimport:/usr/lib/sna/cpic_r64_5.exp -I
/usr/include/sna

Aplicaciones Linux

Antes de compilar y enlazar una aplicación CPI-C, especifique el directorio donde están almacenadas las bibliotecas compartidas, de modo que la aplicación pueda encontrarlas durante la ejecución. Para hacerlo, establezca la variable de entorno `LD_RUN_PATH` en **/opt/ibm/sna/lib** o bien en **/opt/ibm/sna/lib64** si va a compilar una aplicación de 64 bits.

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

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

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

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

Consideraciones sobre CPI-C de Java

Este apartado resume la información que debe tenerse en cuenta al desarrollar aplicaciones CPI-C para Java.

Utilización de clases de CPI-C de Java

El paquete de CPI-C para Java se denomina **COM.ibm.eNetwork.cpic**. Este paquete consiste en una clase Java que contiene:

- Un método para cada una de las llamadas CPI-C soportadas
- Las clases que deben utilizarse como parámetros para estas llamadas

Cuando desarrolle un programa Java para utilizar la clase CPIC, utilice la siguiente sentencia de importación en el archivo fuente Java para importar el paquete CPIC:

import COM.ibm.eNetwork.cpic.*;

Valores constantes

La clase de CPI-C para Java define un número de valores constantes para la longitud máxima en bytes de parámetros CPI-C específicos. En la Tabla 11 se muestran estas constantes. Utilice estas constantes en el programa en lugar de especificar explícitamente las longitudes.

Tabla 11. Constantes de CPI-C de Java

Longitud del parámetro	Constante de CPI-C de Java
Longitud del identificador de conversación	CM_CID_SIZE
Longitud del identificador de contexto	CM_CTX_SIZE
Tamaño de los datos de anotaciones	CM_LD_SIZE
Longitud del nombre de modalidad	CM_MN_SIZE
Longitud del nombre de LU asociada	CM_PLN_SIZE
Longitud de la contraseña de seguridad	CM_PW_SIZE
Longitud del identificador de usuario de seguridad	CM_UID_SIZE
Longitud del nombre de destino simbólico	CM_SDN_SIZE
Longitud del nombre del programa de transacción (TP)	CM_TPN_SIZE

Clases de tipos de parámetros

Muchos de los parámetros utilizados en las funciones CPI-C tienen un valor de entre un conjunto de dos o más valores definidos. En el paquete de CPI-C para Java, cada uno de estos tipos de parámetros está definido como una clase que contiene los valores válidos. Por ejemplo, la clase CPICSyncLevel se utiliza en las funciones Set_Sync_Level (cmssl) y Extract_Sync_Level (cmesl) y puede tener el valor CM_NONE o CM_CONFIRM.

La descripción de cada una de las funciones CPI-C que se facilita en el Capítulo 3, “Llamadas CPI-C”, en la página 49 proporciona el tipo de clase de parámetro CPI-C adecuado y los valores válidos. Por ejemplo, en Set_Sync_Level (cmssl), el

parámetro *sync_level* figura como de tipo CPICSyncLevel y la descripción de los parámetros de esta función señala CM_NONE y CM_CONFIRM como valores válidos.

Como los valores constantes asociados a una clase Java están definidos en la clase, debe acceder a ellos haciendo referencia tanto a la clase como al valor específico. Por ejemplo, para especificar que no haya ninguna confirmación de sincronización, debe establecer el parámetro *sync_level* de la función Set_Sync_Level en `CPICSyncLevel.CM_NONE`.

Cada una de estas clases tiene los métodos siguientes además del constructor:

int intValue()
Devuelve el valor almacenado en el objeto.

int intValue(*valor_entero***)**
Establece el valor almacenado en el objeto como el valor entero *valor_entero* proporcionado y devuelve el mismo valor.

Puede establecer también el valor almacenado en un objeto durante la construcción del objeto, pasando el valor como un parámetro al constructor.

boolean equals(*valor_entero***)**
Devuelve `true` si el valor almacenado en el objeto es igual al valor entero proporcionado (*valor_entero*).

boolean equals(*objeto_proporcionado***)**
Devuelve `true` si el valor almacenado en el objeto es equivalente al valor almacenado en el parámetro proporcionado *objeto_proporcionado*. *objeto_proporcionado* debe ser una instancia de una de las clases de parámetros CPI-C para Java.

La clase `CPICReturnCode` tiene el siguiente método adicional:

boolean isOK()
La aplicación debe llamar a este método para determinar si el valor almacenado en un objeto `CPICReturnCode` es `CM_OK`. La clase genera una excepción si el valor almacenado no es `CM_OK`.

Ejemplo de uso

El siguiente ejemplo muestra cómo configurar el programa Java para utilizar la clase de CPI-C para Java y cómo efectuar una llamada CPI-C individual.

Para importar el paquete de CPI-C para Java, incluya lo siguiente al inicio del código fuente del programa:

```
import COM.ibm.eNetwork.cpic.*;
```

Para utilizar CPI-C para Java en el programa, cree una instancia de la clase CPI-C para Java:

```
CPIC cpicObject = new CPIC();
```

En los pasos que se indican a continuación se muestra cómo efectuar la llamada a cada una de las funciones CPI-C de Java, utilizando la función Initialize_Conversation (cminit) a modo de ejemplo.

1. Crear e inicializar los parámetros para la función:

```
byte[] bConversationId = new byte[cpicObject.CM_CID_SIZE];
String sSymbolicDestination = "testprog";
CPICReturnCode cpicReturn = new CPICReturnCode(0);
```

Observe el uso de la constante CM_CID_SIZE para establecer el tamaño de la matriz de bytes del identificador de conversación y el uso de la clase CPICReturnCode para establecer en cero el valor inicial de este parámetro. La última línea de este ejemplo también se puede dividir en dos líneas, como se muestra a continuación:

```
CPICReturnCode cpicReturn = new CPICReturnCode();
cpicReturn.intValue(0);
```

2. Emitir la llamada de función:

```
cpicObject.cminit(bConversationId,
                  sSymbolicDestination,
                  cpicReturn);
```

3. Probar el código de retorno contra un valor específico:

```
if (cpicReturn.intValue() != CPICReturnCode.CM_PARAMETER_ERROR)
. . .
```

También se puede comprobar si el código de retorno es CM_OK:

```
try
{
      cpicReturn.isOK();
}
catch(CPICReturncode c)
{
. . . // cpicReturn is not set to CM_OK
}
```

Compilación y enlace de la aplicación CPI-C de Java

Antes de compilar y enlazar una aplicación CPI-C de Java, especifique el directorio donde están almacenadas las clases de Java. Para ello, establezca y exporte la variable de entorno CLASSPATH en **/usr/lib/sna/java/cpic.jar:.** (AIX) o **/opt/ibm/sna/java/cpic.jar:.** (Linux).

Compile y enlace la aplicación utilizando el compilador de Java **javac** del modo habitual. Las aplicaciones CPI-C para Java siempre se crean como aplicaciones de 32 bits; las aplicaciones de 64 bits no están soportadas.

Ejecución de la aplicación CPI-C de Java

Antes de ejecutar una aplicación CPI-C de Java, debe especificar el directorio donde están almacenadas las bibliotecas, de modo que la aplicación pueda localizarlas durante la ejecución. También debe establecer una variable de entorno adicional para asegurarse de que CPI-C de Java funcione correctamente con LiS Streams. Para ello, establezca y exporte las variables de entorno adecuadas tal como se indica a continuación:

```
export LD_LIBRARY_PATH=/opt/ibm/sna/lib
export LD_PRELOAD=/usr/lib/libpLiS.so
```

Es probable que también deba establecer y exportar la variable de entorno APPCTPN para especificar el nombre de TP local para la aplicación, tal como se describe en "Especificación del nombre de TP local" en la página 34.

Ejecute la aplicación mediante el intérprete de Java **java** del modo habitual.

consideraciones sobre Windows

WINDOWS

Esta sección resume las consideraciones de proceso que hay que tener en cuenta al desarrollar programas en un Remote API Client en Windows.

Archivos CPI-C de Windows

El archivo de cabecera que va a utilizarse con las aplicaciones CPI-C para Windows es **wincpic.h**, que contiene las definiciones de todos los puntos de entrada CPI-C y las constantes definidas para los valores de parámetro suministrados y devueltos en la interfaz CPI-C para Windows. Este archivo está instalado en el subdirectorio **\sdk** para aplicaciones de 32 bits o **\sdk64** para aplicaciones de 64 bits, dentro del directorio donde ha instalado el software Remote API Client en Windows.

La biblioteca que se utiliza para enlazar aplicaciones CPI-C de Windows es **\sdk\wcpic32.lib** para aplicaciones de 32 bits, o bien **\sdk64\wcpic32.lib** para aplicaciones de 64 bits.

Prototipos de funciones

Los prototipos de funciones para llamadas CPI-C que se muestran en el Capítulo 3, "Llamadas CPI-C", en la página 49 están en el formato que se utiliza para sistemas AIX o Linux. Para sistemas Windows, sustituya "void *nombrefunción*" por "void WINAPI *nombrefunción*" para cada llamada.

Procesos múltiples y conversaciones múltiples

Los procesos múltiples no pueden tener el mismo identificador de conversación. Sólo el proceso que emite la llamada Initialize_Conversation o Accept_Conversation puede utilizar el ID de conversación devuelto por la llamada. Otro proceso que desea utilizar CPI-C debe emitir una llamada Initialize_Conversation o Accept_Conversation para obtener su propio ID de conversación.

Un programa puede participar hasta en 64 conversaciones simultáneas.

Llamadas de función de Windows

Además de las llamadas de función CPI-C estándar y la llamada de función CPI-C específica de Windows, Specify_Windows_Handle, una aplicación Windows utiliza las funciones siguientes:

WinCPICStartup
: Registra la aplicación como un usuario CPI-C de Windows y determina si el software CPI-C soporta el nivel de función que la aplicación necesita.

WinCPICCleanup
: Elimina el registro de la aplicación cuando ha terminado de utilizar CPI-C.

WinCPICIsBlocking
Comprueba si hay una llamada de bloqueo pendiente para esta aplicación. Para más información sobre las circunstancias en las que esta llamada podría ser necesaria, consulte el apartado "Llamadas de bloqueo".

WinCPICSetBlockingHook
Especifica el procedimiento de bloqueo que CPI-C utiliza mientras procesa las llamadas de bloqueo; esto sustituye al procedimiento de bloqueo por omisión de CPI-C. Se invoca el procedimiento de bloqueo reiteradamente hasta que haya concluido el proceso de la llamada de bloqueo. Para ver más información, consulte el apartado "Llamadas de bloqueo".

WinCPICUnhookBlockingHook
Elimina el registro del procedimiento de bloqueo especificado por una llamada `WinCPICSetBlockingHook` anterior, de modo que CPI-C vuelve a utilizar el procedimiento de bloqueo por omisión.

WinCPICExtractEvent
Proporciona un método para que una aplicación determine el descriptor de sucesos de Windows que se utiliza para una conversación CPI-C.

WinCPICSetEvent
Asocia un descriptor de sucesos de Windows con la finalización de verbos para una conversación CPI-C.

La aplicación debe llamar a `WinCPICStartup` antes de intentar emitir alguna llamada CPI-C.

El apartado "Llamadas de bloqueo" proporciona más información sobre cómo funcionan las llamadas de bloqueo en el entorno Windows y cómo la aplicación debe utilizar las llamadas `WinCPICIsBlocking`, `WinCPICSetBlockingHook` y `WinCPICUnhookBlockingHook`.

Cuando la aplicación ha finalizado de emitir llamadas CPI-C, debe llamar `WinCPICCleanup` antes de terminar; no debe intentar emitir ninguna llamada CPI-C más después de llamar `WinCPICCleanup`.

Las llamadas de función de Windows se describen al final del Capítulo 3, "Llamadas CPI-C", en la página 49.

Llamadas de bloqueo

Esta sección describe cómo las llamadas CPI-C de bloqueo (llamadas emitidas con la modalidad de proceso de la conversación establecida en `CM_BLOCKING`) funcionan en el entorno Windows si la aplicación que llama es de un solo subproceso. (Normalmente, una aplicación Windows utiliza múltiples subprocesos para evitar el problema de un verbo de bloqueo que bloquea toda la aplicación.)

En el apartado también se proporciona información que hay que tener en cuenta cuando se escriben aplicaciones para utilizar llamadas de bloqueo.

Remote API Client proporciona soporte para llamadas de bloqueo en sistemas Windows para ayudar en la migración de aplicaciones desde otros entornos de sistemas operativos. Sin embargo, no es nada aconsejable utilizar llamadas de bloqueo en el entorno Windows. Si escribe una aplicación nueva específicamente para Windows, debe realizar las acciones siguientes:

- Utilice la función Specify_Windows_Handle para especificar un descriptor de Windows en el que CPI-C anota los resultados de una llamada efectuada

- Emita todas las llamadas CPI-C en modalidad sin bloqueo

Aunque aparentemente haya una llamada de bloqueo que ponga en suspenso la aplicación hasta que CPI-C haya finalizado de procesar la llamada, la biblioteca CPI-C ha de ceder el control del sistema mientras espera a que Communications Server para Linux complete el proceso, con el fin de permitir la ejecución de otros procesos. Para ello, utiliza una "función de bloqueo", que se invoca reiteradamente mientras la biblioteca está esperando; la función permite que se envíen mensajes de Windows a otros procesos. Para obtener más información sobre esta función, consulte el apartado "Función de bloqueo por omisión".

Es posible que la función de bloqueo envíe un mensaje a la aplicación que ha emitido la llamada de bloqueo original; en este caso, se puede volver a entrar la aplicación aunque tenga una llamada de bloqueo pendiente. En estas circunstancias, la aplicación puede continuar con otros procesos no relacionados con la emisión de llamadas CPI-C. Sin embargo, no puede emitir otra llamada de bloqueo mientras la primera está pendiente.

La aplicación puede comprobar si hay una llamada de bloqueo pendiente (es decir, si se ha vuelto a entrar como resultado de un mensaje recibido mientras la llamada estaba pendiente) mediante la función `WinCPICIsBlocking`, que se describe en el Capítulo 3, "Llamadas CPI-C", en la página 49. Si esta función indica que hay una llamada de bloqueo pendiente, la aplicación no debe intentar utilizar más llamadas CPI-C de bloqueo. Sin embargo, puede realizar las siguientes acciones:

- Continuar con otros procesos
- Emitir llamadas CPI-C en otras conversaciones para las cuales la modalidad de proceso es `CM_NON_BLOCKING`

Función de bloqueo por omisión

La función de bloqueo estándar por omisión que la biblioteca CPI-C de Windows utiliza es la siguiente:

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

Si es necesario que la aplicación realice otros procesos como parte de la función de bloqueo, puede especificar su propia función de bloqueo para sustituir la función por omisión proporcionada por CPI-C. Para ello, utiliza la llamada `WinCPICSetBlockingHook`, que se describe en el Capítulo 3, "Llamadas CPI-C", en la página 49.

Una función de bloqueo debe devolver `FALSE` si recibe un mensaje WM_QUIT; esto significa que CPI-C devuelve el control a la aplicación, que a continuación puede procesar el mensaje y terminar. De lo contrario, la función debe devolver `TRUE`.

Finalización de aplicaciones

CPI-C no puede indicar cuándo finaliza una aplicación bajo Windows. Por consiguiente, si una aplicación tiene que finalizar (por ejemplo, recibe un mensaje WM_CLOSE), la aplicación debe emitir la llamada `WinCPICCleanup`. Si no se emite la llamada, el sistema se queda en un estado indeterminado, sin embargo, se realiza la máxima limpieza posible cuando CPI-C detecta posteriormente que la aplicación ha terminado.

Compilación y enlace de aplicaciones CPI-C

En este apartado se proporciona información sobre la compilación y enlace de aplicaciones CPI-C en sistemas Windows.

Opciones del compilador para el empaquetado de las estructuras

Las estructuras suministradas y devueltas en algunas llamadas CPI-C no están empaquetadas. No utilice opciones del compilador que cambien este método de empaquetamiento. Los parámetros BYTE se encuentran en límites BYTE, los parámetros WORD se encuentran en límites WORD y los parámetros DWORD se encuentran en límites DWORD.

Archivos de cabecera

El archivo de cabecera que se va a incluir en las aplicaciones CPI-C de Windows se denomina **wincpic.h**. Este archivo está instalado en el subdirectorio **/sdk** dentro del directorio donde ha instalado el software Cliente de Windows.

Enlace durante la carga

Para enlazar la aplicación con CPI-C durante la carga, enlace la aplicación con la biblioteca **wincpic32.lib**.

Enlace durante la ejecución

Para enlazar la aplicación con CPI-C durante la ejecución, incluya las siguientes llamadas en la aplicación:

- `LoadLibrary` para cargar la biblioteca de enlace dinámico de CPI-C **wincpic32.dll**
- `GetProcAddress` para especificar `WinCPIC` como el punto de entrada a la biblioteca de enlace dinámico
- `FreeLibrary` cuando la biblioteca ya no es necesaria

Desarrollo de aplicaciones portables

A continuación, se proporcionan directrices para desarrollar aplicaciones CPI-C portables a otros entornos de sistema operativo o a otras implementaciones de CPI-C.

- Incluya el archivo de cabecera CPI-C sin ningún prefijo de nombre de vía de acceso. Utilice opciones de inclusión en el compilador para localizar el archivo (vea el apartado apropiado correspondiente a su sistema operativo, al principio de este capítulo). Esto permite utilizar la aplicación en un entorno que tenga otro sistema de archivos.
- 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; de este modo se garantiza que se utilizará el valor correcto, independientemente de cómo estén almacenados estos valores en la memoria.

- Incluya una marca para los códigos de retorno que no sean los que corresponden a su sistema operativo actual (por ejemplo, utilizando un caso "valor por omisión" en una sentencia switch) y proporcione los diagnósticos apropiados.
- Algunas de las funciones CPI-C proporcionadas por Communications Server para Linux son extensiones incluidas para lograr la compatibilidad con X/Open CPI-C o no forman parte de la especificación CPI-C estándar y puede que no estén disponibles en otras implementaciones. Cada una de estas funciones de extensión está identificada mediante notas en la introducción de la descripción de la función en el Capítulo 3, "Llamadas CPI-C", en la página 49.
  - Las funciones X/Open se incluyen para que pueda utilizar aplicaciones existentes desarrolladas para X/Open CPI-C con Communications Server para Linux. Se recomienda no utilizar estas funciones al desarrollar aplicaciones nuevas.
  - Si utiliza las funciones de extensión en la aplicación, es probable que deba volver a escribir secciones de la aplicación para poder utilizarla en otros entornos. Puede restringir el uso de estas funciones a unas cuantas rutinas específicas para facilitar las modificaciones.

AIX, LINUX

Las directrices siguientes se aplican a las aplicaciones CPI-C de Java:

- Las tres funciones Extract_Conversation_Context, Set_Conversation_Context y Set_Local_LU_Name no forman parte de la especificación CPI-C estándar y no están soportadas en CPI-C de Java de IBM para CS/Windows. Si utiliza estas funciones en su aplicación CPI-C para Java, es probable que deba volver a desarrollar secciones de la aplicación para poderla utilizar en otros entornos CPI-C de Java. Puede restringir el uso de estas funciones a unas cuantas rutinas específicas para facilitar las modificaciones.
- La clase CPI-C para Java incluye algunas funciones CPI-C que no se describen en este manual y que están definidas como parte de la clase Java, pero que no están soportadas. Si utiliza estas funciones no soportadas en la aplicación, la compilación podrá realizarse correctamente pero las funciones devolverán un código de retorno de error (`CM_CALL_NOT_SUPPORTED`) si la aplicación intenta utilizarlas.

Capítulo 3. Llamadas CPI-C

Este capítulo describe las llamadas de función de CPI-C y las llamadas de función adicionales específicas de Windows utilizadas por aplicaciones CPI-C. Se facilita la información siguiente:

- Descripción de la información proporcionada para las llamadas
- Descripciones de las llamadas

Información proporcionada para las llamadas CPI-C

Para cada una de las llamadas CPI-C descritas en este capítulo se proporciona la información siguiente:

- El seudónimo de la llamada, seguido del nombre real de la función en C entre paréntesis (esta información está en la cabecera de sección).
- El prototipo de función de la llamada, incluyendo los parámetros utilizados por la llamada y el tipo de datos de cada uno de los parámetros. El prototipo de cada función se declara en el archivo **cmc.h** (sistemas AIX o Linux) o **wincpic.h** (sistemas Windows).

  WINDOWS

  Los prototipos de funciones para llamadas CPI-C que se muestran en el Capítulo 3, "Llamadas CPI-C" están en el formato que se utiliza para sistemas AIX o Linux. Para sistemas Windows, sustituya "void *nombrefunción*" por "void WINAPI *nombrefunción*" para cada llamada.

  

- La definición del método Java para la función CPI-C, si está soportada en CPI-C para Java.
- Una descripción de cada uno de los parámetros suministrados y devueltos. Los nombres de parámetros son seudónimos. El programa de aplicación declara los nombres de variable reales de estos parámetros. La descripción incluye los valores posibles del parámetro.
- Los estados de conversación en que puede emitirse la llamada.
- El estado o estados a los que puede cambiar la conversación al volver la llamada. Las condiciones que no provocan un cambio de estado, como por ejemplo las comprobaciones de parámetros y las comprobaciones de estado.
- Información adicional sobre el uso de la llamada.

Tipos de datos

Para obtener información sobre tipos de datos en aplicaciones CPI-C para Java, consulte el apartado "Consideraciones sobre CPI-C de Java" en la página 40.

Con el objeto de mejorar la portabilidad de las aplicaciones CPI-C, los tipos de datos de los parámetros suministrados a CPI-C y recibidos de CPI-C están establecidos como constantes simbólicas mediante sentencias #define en el archivo de cabecera CPI-C. Por ejemplo, CM_INT32 representa un tipo de entero de 32 bits; CM_PTR representa un tipo de puntero.

Este capítulo utiliza estas constantes simbólicas para identificar los tipos de datos de los parámetros suministrados y devueltos. Al desarrollar aplicaciones, se recomienda utilizar estas constantes simbólicas en lugar de los tipos de datos reales.

Estructuras de datos

Este apartado no se aplica a las aplicaciones CPI-C para Java porque ninguna de las funciones CPI-C soportadas en CPI-C para Java utiliza estructuras de datos.

En el caso de algunas llamadas CPI-C, la aplicación suministra una estructura de datos en la que Communications Server para Linux puede rellenar parámetros que se devolverán a la aplicación. Estas estructuras de datos contienen parámetros marcados como "reservados"; algunos de estos parámetros reservados se utilizan internamente con el software Communications Server para Linux y otros no se utilizan en esta versión pero se pueden utilizar en futuras versiones. La aplicación no debe intentar acceder a ninguno de estos parámetros reservados; en cambio, debe establecer en el valor cero todo el contenido de la estructura de datos para garantizar que todos estos parámetros sean cero antes de definir otros parámetros utilizados por el verbo. Esto asegura que Communications Server para Linux no interpretará erróneamente ninguno de sus parámetros utilizados internamente, y también que la aplicación continuará funcionando con las futuras versiones de Communications Server para Linux, en las que estos parámetros se pueden utilizar para proporcionar nuevas funciones.

Para establecer el contenido de la estructura de datos en cero, utilice `memset`:

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

Constantes simbólicas

Para obtener información sobre valores de constantes simbólicas en aplicaciones CPI-C para Java, consulte el apartado "Consideraciones sobre CPI-C de Java" en la página 40.

La mayoría de los parámetros suministrados a CPI-C y devueltos por CPI-C son enteros de 32 bits. Para simplificar la codificación, los valores de estos parámetros se representan mediante constantes simbólicas significativas que están establecidas por medio de sentencias #define en el archivo de cabecera. Por ejemplo, el valor `CM_MAPPED_CONVERSATION` representa el entero 1. Para facilitar su portabilidad y lectura, utilice únicamente las constantes simbólicas cuando desarrolle programas.

Cadenas

Todas las cadenas están en formato ASCII cuando se pasan a través de la interfaz CPI-C.

Validez de los parámetros devueltos

Los parámetros devueltos por CPI-C sólo son válidos si la llamada CPI-C se ha ejecutado correctamente, como indica el código de retorno `CM_OK`.

Información proporcionada para las llamadas de función de Windows

WINDOWS

Se suministra la siguiente información para cada una de las llamadas de función específicas de Windows que se describen en este capítulo:

- El nombre de la llamada; a diferencia de las llamadas de función CPI-C, estas llamadas no contienen seudónimos.
- Una descripción de la llamada.
- El prototipo de función de la llamada, incluidos los parámetros utilizados por la llamada y el tipo de datos de cada uno de los parámetros. El prototipo de cada función se declara en el archivo **wincpic.h**.
- Una descripción de cada uno de los parámetros suministrados y devueltos. Los nombres de parámetros son seudónimos. El programa de aplicación declara los nombres de variable reales de estos parámetros. La descripción incluye los valores posibles del parámetro.
- Información adicional sobre el uso de la llamada.

Accept_Conversation (cmaccp)

El programa invocado emite la llamada Accept_Conversation para aceptar la conversación entrante y definir determinadas características de conversación. Para ver una lista de las características iniciales de la conversación, consulte el Capítulo 2, “Desarrollo de aplicaciones CPI-C”, en la página 19.

Una vez ejecutada correctamente esta llamada, CPI-C genera un identificador de conversación de 8 bytes.. Este identificador es un parámetro necesario para todas las demás llamadas CPI-C emitidas por el programa invocado durante esta conversación.

Llamada de función

```
void cmaccp (
             unsigned char CM_PTR         conversation_ID,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmaccp (
                           byte[]             conversation_ID,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

No hay ningún parámetro suministrado para esta llamada.

Parámetros devueltos

Una vez que la llamada se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

conversation_ID
Identificador de la conversación. Las llamadas CPI-C posteriores lo utilizan.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_STATE_CHECK
Este valor indica una de las condiciones siguientes:

- No se ha recibido ninguna petición Allocate entrante dentro del período de tiempo de espera especificado en la configuración.
- La aplicación no ha especificado ningún nombre TP local (o para sistemas AIX o Linux, ha liberado todos los nombres especificados). La aplicación debe tener como mínimo un nombre de TP local antes de emitir esta llamada. Para ver más información sobre cómo especificar nombres de TP local, consulte "Especificación del nombre de TP local" en la página 34.
- La aplicación se ha iniciado manualmente, pero está definida en el archivo de datos de TP invocable como sin cola. Communications Server para Linux inicia automáticamente un TP sin cola como respuesta a una petición de conversación (una petición Attach entrante); si intenta iniciarlo manualmente, la llamada Accept_Conversation fallará porque no hay ninguna petición Attach entrante que espere la aplicación.

CM_PRODUCT_SPECIFIC_ERROR
Consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Restablecer.

Cambio de estado

Si la llamada se ejecuta correctamente, la conversación cambia al estado Recibir. Si la llamada falla, el estado no se modifica.

Notas de uso

El nombre de TP puede especificarse de varias formas. Para ver más información sobre cómo especificar nombres de TP local, consulte "Especificación del nombre de TP local" en la página 34. Antes de emitir Accept_Conversation, el programa puede emitir Specify_Local_TP_Name para indicar uno o varios nombres de TP para los que aceptará peticiones Allocate entrantes (estos nombres se añaden a los nombres definidos utilizando otros métodos como, por ejemplo, la variable de entorno `APPCTPN`). Si especifica más de un nombre de TP de esta forma, después puede utilizar la llamada Extract_TP_Name (después del retorno de Accept_Conversation) para determinar qué nombre de TP ha utilizado el programa que invoca.

AIX, LINUX

Si Accept_Conversation devuelve CM_OK, se crea un nuevo contexto de conversación para la conversación y éste pasa a ser el contexto actual del programa.

Accept_Conversation siempre funciona en la modalidad con bloqueo; es decir, siempre se suspende hasta que se recibe una petición Allocate entrante. Pueden utilizarse los métodos siguientes para evitar retardos innecesarios:

- Asegúrese de que la configuración del TP invocable de esta aplicación especifique un valor de tiempo de espera pequeño, de modo que la llamada Accept_Conversation vuelva rápidamente (con el parámetro *return_code* con el valor CM_PROGRAM_STATE_CHECK) si no hay ninguna petición Allocate entrante, y haga que la aplicación vuelva a intentar emitir Accept_Conversation más adelante. El valor de tiempo de espera se especifica en el archivo de datos de TP invocable; para más información, consulte la publicación *Communications Server para Linux, Guía de administración*.
- En lugar de utilizar Accept_Conversation, utilice Accept_Incoming, que puede operar en la modalidad sin bloqueo. Utilice la siguiente secuencia de llamadas:
  - Initialize_For_Incoming (para obtener un identificador de conversación para la conversación entrante)
  - Set_Processing_Mode (para establecer el parámetro *processing_mode* para este ID de conversación en CM_NON_BLOCKING)
  - Accept_Incoming

  Para más información, consulte las descripciones de estas llamadas.

Accept_Incoming (cmacci)

AIX, LINUX

El programa invocado emite la llamada Accept_Incoming para aceptar una conversación entrante que anteriormente se ha inicializado con Initialize_For_Incoming y definir determinadas características de conversación. Para ver una lista de las características iniciales de la conversación, consulte "Características iniciales de la conversación" en la página 27.

Antes de emitir esta llamada, el programa puede emitir Set_Processing_Mode para establecer la modalidad de proceso de la conversación en CM_NON_BLOCKING. De esta forma se asegura que la llamada Accept_Incoming y todas las llamadas CPI-C posteriores se emitan en la modalidad sin bloqueo.

Llamada de función

```
void cmacci (
             unsigned char CM_PTR         conversation_ID,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

```
public native void cmacci (
                            byte[]             conversation_ID,
                            CPICReturnCode     return_code
                           );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación que se ha devuelto en la llamada Initialize_For_Incoming anterior. Se utiliza para identificar las llamadas CPI-C posteriores por esta conversación.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
Se ha producido una de estas situaciones:

- La conversación especificada por *conversation_ID* no se encuentra en estado Inicializar-Entrante.
- No se ha recibido ninguna petición Allocate entrante dentro del período de tiempo de espera especificado en la configuración.
- La aplicación ha liberado el nombre de TP local especificado, por ejemplo, en la variable de entorno `APPCTPN`, y no ha especificado ningún nombre de TP local adicional. La aplicación debe tener como mínimo un nombre de TP local antes de emitir esta llamada. Para ver más información sobre cómo especificar nombres de TP local, consulte "Especificación del nombre de TP local" en la página 34.
- La aplicación se ha iniciado manualmente, pero está definida en el archivo de datos de TP invocable como sin cola. Communications Server para Linux inicia automáticamente un TP sin cola como respuesta a una petición Attach entrante; si intenta iniciarlo manualmente, la llamada Accept_Incoming fallará porque no hay ninguna petición Attach entrante para la aplicación.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

`CM_OPERATION_INCOMPLETE`
`CM_OPERATION_NOT_ACCEPTED`
`CM_PRODUCT_SPECIFIC_ERROR`

Estado al emitirse

La conversación debe encontrarse en estado Inicializar-Entrante.

Cambio de estado

Si la llamada se ejecuta correctamente, la conversación cambia al estado Recibir. Si la llamada falla, el estado no se modifica.

Notas de uso

Emitir Initialize_For_Incoming seguido de Accept_Incoming equivale a emitir Accept_Conversation. La diferencia entre los dos métodos de aceptación de una conversación consiste en que Accept_Conversation siempre opera en la modalidad con bloqueo, mientras que Accept_Incoming puede operar en la modalidad sin bloqueo. Para aceptar una conversación en la modalidad sin bloqueo, el programa emite la siguiente secuencia de llamadas:

Initialize_For_Incoming (para obtener un identificador de conversación para la conversación entrante)
Set_Processing_Mode (para establecer el parámetro *processing_mode* para este ID de conversación en `CM_NON_BLOCKING`)
Accept_Incoming

El nombre de TP especificado por la variable de entorno `APPCTPN` normalmente es el mismo nombre que se utiliza para hacer coincidir las peticiones Allocate entrantes con este programa. Antes de emitir Accept_Incoming, el programa puede emitir Specify_Local_TP_Name para indicar uno o varios nombres de TP para los que aceptará peticiones Allocate entrantes (estos nombres sustituyen al nombre que figura en `APPCTPN`). Si especifica más de un nombre de TP de esta forma, después puede utilizar la llamada Extract_TP_Name (una vez que vuelva Accept_Incoming) para determinar qué nombre de TP ha utilizado el programa que invoca. Para ver más información sobre cómo especificar nombres de TP local, consulte "Especificación del nombre de TP local" en la página 34.

Si Accept_Incoming devuelve `CM_OK`, se crea un nuevo contexto de conversación para la conversación y éste pasa a ser el contexto actual del programa. Si Accept_Incoming devuelve `CM_OPERATION_INCOMPLETE` y un programa Wait_For_Conversation posterior devuelve la finalización de Accept_Incoming como `CM_OK`, se crea un nuevo contexto de conversación para la conversación, pero no se cambia el contexto actual del programa. Para utilizar el nuevo contexto, el programa debe emitir Extract_Conversation_Context para este *conversation_ID* a fin de obtener el valor del contexto de la conversación, y Set_Conversation_Context a fin de establecer el contexto actual del programa en este valor.

Allocate (cmallc)

El programa que invoca emite la llamada Allocate para asignar una conversación con el programa asociado, utilizando las características actuales de la conversación. CPI-C también puede asignar una sesión entre la LU local y la LU asociada si todavía no existe ninguna.

El tipo de conversación asignado depende de la característica de tipo de conversación (correlacionada o básica).

Una vez que esta llamada ha asignado la conversación, no se pueden cambiar las características de conversación siguientes:

- Tipo de conversación

- Nombre de modalidad
- Nombre de LU asociada
- Nombre de programa asociado
- Control de retorno
- Nivel de sincronización
- Seguridad de conversación
- Identificador de usuario
- Contraseña

Llamada de función

```
void cmallc (
             unsigned char CM_PTR         conversation_ID,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmallc (
                           byte[]             conversation_ID,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PARAMETER_ERROR
Se ha producido una de estas situaciones:

- El nombre de modalidad procedente de la información complementaria o establecida por Set_Mode_Name no es válido.
- El nombre de modalidad es uno de los nombres reservados para el uso interno de SNA (como, por ejemplo, SNASVCMG); una aplicación no puede utilizarlo.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
La conversación no se encuentra en estado Inicializar.

CM_UNSUCCESSFUL
La característica de control de retorno de la conversación está definida en CM_IMMEDIATE y la LU local no tiene ninguna sesión ganadora de contienda disponible.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.
CM_ALLOCATE_FAILURE_NO_RETRY
CM_ALLOCATE_FAILURE_RETRY
CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

Los cambios de estado, resumidos en la Tabla 12, se basan en el valor del parámetro *return_code*.

Tabla 12. Cambios de estado para la llamada Allocate

return_code	**Estado nuevo**
CM_OK	Enviar
CM_ALLOCATE_FAILURE_NO_RETRY	Restablecer
CM_ALLOCATE_FAILURE_RETRY	
Todos los demás	Sin cambio

Notas de uso

Para enviar la petición de asignación de inmediato, el programa que invoca puede emitir la llamada Flush o Confirm inmediatamente después de la llamada Allocate. De lo contrario, la petición de asignación se acumula con otros datos en el almacenamiento intermedio de envío de la LU local hasta que se llena el almacenamiento intermedio.

Dado que la petición de asignación se coloca en el almacenamiento intermedio y no se envía de inmediato, la llamada Allocate puede devolver CM_OK, pero posteriormente la LU asociada puede rechazar la petición de asignación generada por la llamada Allocate. Este error se devuelve al programa que invoca en una llamada posterior.

Si el nivel de sincronización de la conversación está definido en CM_CONFIRM, el programa que invoca puede determinar de inmediato si la asignación ha sido correcta emitiendo la llamada Confirm después de la llamada Allocate.

AIX, LINUX

El contexto actual del programa en el momento en que se emite la llamada Allocate pasa a ser el contexto de la nueva conversación cuando Allocate devuelve

CM_OK. Si el programa utiliza varios contextos (como consecuencia de aceptar varias conversaciones), debe establecer el contexto actual en el valor adecuado antes de emitir la llamada Allocate.

Cancel_Conversation (cmcanc)

La llamada Cancel_Conversation finaliza una conversación especificada, cancelando las operaciones incompletas (una llamada anterior que ha vuelto con CM_OPERATION_INCOMPLETE) de esta conversación, y finaliza la sesión que utilizaba la conversación. Equivale a la llamada Deallocate con el parámetro *deallocate_type* establecido en CM_DEALLOCATE_ABEND, con las diferencias siguientes:

- No se puede utilizar Deallocate mientras haya una operación incompleta; se puede utilizar Cancel_Conversation, lo que cancela la llamada pendiente.
- Deallocate graba los datos de anotaciones, si existen, en el archivo de anotaciones de error local; Cancel_Conversation no.

Los resultados de la llamada pendiente son indefinidos y no se devuelven a la aplicación. Por ejemplo, si se utiliza Cancel_Conversation para cancelar una llamada Send_Data pendiente, puede que se hayan enviado algunos datos o todos; si se utiliza para cancelar Send_Error, puede que se haya enviado o no una indicación de error al programa asociado.

En CPI-C para Java, las llamadas sin bloqueo no están soportadas y por consiguiente, no puede haber una llamada incompleta pendiente. Cancel_Conversation es equivalente a Deallocate con la diferencia de que no escribe datos de anotaciones en el archivo de anotaciones de error local.

Llamada de función

```
void cmcanc (
             unsigned char CM_PTR         conversation_ID,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmcanc (
                            byte[]             conversation_ID,
                            CPICReturnCode     return_code
                           );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente. Se ha desasignado la conversación especificada y se han cancelado las llamadas pendientes de esta conversación.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

Si el código de retorno es CM_OK, el estado de conversación cambia a Restablecer.

Notas de uso

El programa asociado recibe la notificación del final de la conversación con el código de retorno CM_DEALLOCATED_ABEND.

Check_For_Completion (cmchck)

AIX, LINUX

Esta función no está disponible en CPI-C para Java.

La llamada Check_For_Completion comprueba si una llamada anterior que ha vuelto con CM_OPERATION_INCOMPLETE ha finalizado desde entonces. Esta llamada vuelve de inmediato tanto si la llamada anterior ha finalizado como si no; posteriormente la aplicación puede continuar con otro proceso si la llamada anterior todavía no ha terminado o llamar Wait_For_Conversation para obtener los resultados de la llamada anterior si ha finalizado.

Si la aplicación participa en varias conversaciones, esta llamada actúa en todas las conversaciones y devuelve un código de retorno de ejecución “correcta” si ha finalizado una llamada anterior en cualquiera de ellas.

Esta llamada no forma parte de la especificación CPI-C estándar y puede no estar disponible en otras implementaciones. El procedimiento estándar para obtener los resultados de una llamada pendiente consiste en emitir Wait_For_Conversation, que opera en la modalidad con bloqueo y espera a que finalice una llamada.

Llamada de función

```
void cmchck (
            unsigned char CM_PTR         conversation_ID,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Parámetros suministrados

No hay ningún parámetro suministrado para esta llamada.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

conversation_ID
: Identificador de la conversación en que ha finalizado una llamada pendiente anterior. Para ver más información, consulte "Notas de uso".

  Este valor sólo es relevante si el parámetro *return_code* tiene el valor CM_OK.

return_code
: Los valores posibles son:

  CM_OK La llamada se ha ejecutado correctamente. Una llamada que estaba pendiente en la conversación especificada por *conversation_ID* ha finalizado.

  CM_PROGRAM_STATE_CHECK
  : No hay ninguna llamada anteriormente incompleta pendiente. La aplicación no ha emitido ninguna llamada que haya devuelto CM_OPERATION_INCOMPLETE o ya haya emitido Wait_For_Conversation para obtener los resultados de todas esas llamadas.

  CM_UNSUCCESSFUL
  : Hay como mínimo una llamada anteriormente incompleta pendiente, pero todavía no hay ninguna que haya finalizado. La aplicación puede seguir con otro proceso y volver a intentar emitir Check_For_Completion más adelante. (Este código de retorno es distinto de CM_PROGRAM_STATE_CHECK.)

Estado al emitirse

La llamada no está asociada a ninguna conversación específica, por lo que el estado de conversación no es relevante. No obstante, la aplicación debe tener como mínimo una conversación con una operación incompleta pendiente.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el código de retorno de Check_For_Completion es CM_OK, la aplicación debe llamar Wait_For_Conversation para obtener los resultados de la llamada pendiente.

Si ha finalizado más de una llamada desde la última vez que la aplicación emitió Check_For_Completion o Wait_For_Conversation, el hecho de emitir Check_For_Completion más de una vez no necesariamente devuelve información

sobre las llamadas adicionales; simplemente indica que ha finalizado como mínimo una llamada, y por consiguiente una llamada Wait_For_Conversation posterior volverá de inmediato y no efectuará ningún bloqueo. Cada una de las llamadas Wait_For_Conversation devuelve una operación incompleta; si hay varias operaciones incompletas (en distintas conversaciones), la aplicación puede emitir otra llamada Check_For_Completion después de Wait_For_Conversation para comprobar si han finalizado más llamadas.

La llamada Wait_For_Conversation no devuelve necesariamente la información de la misma llamada de la que ha informado Check_For_Completion.

Confirm (cmcfm)

La llamada Confirm envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al programa asociado y espera una confirmación.

Como respuesta a la llamada Confirm, el programa asociado normalmente emite la llamada Confirmed para confirmar que ha recibido los datos sin ningún error. (Si el programa asociado encuentra un error, emite la llamada Send_Error o utiliza la llamada Deallocate para desasignar anormalmente la conversación.)

El programa únicamente puede emitir la llamada Confirm si el nivel de sincronización de la conversación es CM_CONFIRM.

Llamada de función

```
void cmcfm (
            unsigned char CM_PTR         conversation_ID,
            CM_Request_to_Send_Received CM_PTR   request_to_send_received,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmcfm (
                          byte[]             conversation_ID,
                          CPICControlInformationReceived  request_to_send_received,
                          CPICReturnCode     return_code
                         );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

request_to_send_received
Indicador de petición de envío recibida. Los valores posibles son:

CM_REQ_TO_SEND_RECEIVED
El programa asociado ha emitido la llamada Request_To_Send, que solicita al programa local que cambie la conversación al estado Recibir.

CM_REQ_TO_SEND_NOT_RECEIVED
El programa asociado no ha emitido la llamada Request_To_Send.

Este valor no es relevante si el parámetro *return_code* tiene uno de los valores siguientes:

- CM_PROGRAM_PARAMETER_CHECK
- CM_PROGRAM_STATE_CHECK

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente. El programa asociado ha emitido la llamada Confirmed.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* no es válido.
- El programa local ha intentado utilizar la llamada Confirm en una conversación con el nivel de sincronización CM_NONE. El nivel de sincronización debe ser CM_CONFIRM.

CM_PROGRAM_STATE_CHECK
Se ha producido una de estas situaciones:

- La conversación no se encontraba en estado Enviar ni Enviar-Pendiente.
- La conversación básica del programa local se encontraba en estado Enviar y el programa local no había terminado de enviar un registro lógico.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

CM_CONVERSATION_TYPE_MISMATCH
CM_DEALLOCATED_ABEND
CM_DEALLOCATED_ABEND_SVC
CM_DEALLOCATED_ABEND_TIMER
CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PIP_NOT_SPECIFIED_CORRECTLY
CM_PRODUCT_SPECIFIC_ERROR
CM_PROGRAM_ERROR_PURGING
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY
CM_SECURITY_NOT_VALID
CM_SVC_ERROR_PURGING
CM_SYNC_LVL_NOT_SUPPORTED_PGM

CM_SYNC_LVL_NOT_SUPPORTED_LU
CM_TP_NOT_AVAILABLE_NO_RETRY
CM_TP_NOT_AVAILABLE_RETRY
CM_TPN_NOT_RECOGNIZED

Estado al emitirse

La conversación puede encontrarse en estado Enviar o Enviar-Pendiente.

Cambio de estado

Los cambios de estado, resumidos en la Tabla 13, se basan en el valor del parámetro *return_code*.

Tabla 13. Cambios de estado para la llamada Confirm

return_code	**Estado nuevo**
CM_OK (llamada emitida en estado Enviar)	Sin cambio
CM_OK (llamada emitida en estado Enviar-Pendiente)	Enviar
CM_PROGRAM_ERROR_PURGING CM_SVC_ERROR_PURGING	Recibir
CM_CONVERSATION_TYPE_MISMATCH CM_PIP_NOT_SPECIFIED_CORRECTLY CM_SECURITY_NOT_VALID CM_SYNC_LEVEL_NOT_SUPPORTED_PGM CM_SYNC_LEVEL_NOT_SUPPORTED_LU CM_TPN_NOT_RECOGNIZED CM_TP_NOT_AVAILABLE_NO_RETRY CM_TP_NOT_AVAILABLE_RETRY CM_RESOURCE_FAILURE_NO_RETRY CM_RESOURCE_FAILURE_RETRY CM_DEALLOCATED_ABEND CM_DEALLOCATED_ABEND_SVC CM_DEALLOCATED_ABEND_TIMER	Restablecer
Todos los demás	Sin cambio

Notas de uso

La llamada Confirm espera una respuesta del programa asociado. Una respuesta se genera mediante una de las siguientes llamadas CPI-C en el programa asociado:

- Confirmed
- Send_Error
- Deallocate con el tipo de desasignación de la conversación definido en CM_DEALLOCATE_ABEND

Confirmed (cmcfmd)

La llamada Confirmed responde a una petición de confirmación del programa asociado. Informa al programa asociado de que el programa local no ha detectado ningún error en los datos recibidos.

Como el programa que emite la petición de confirmación espera una confirmación, la llamada Confirmed sincroniza el proceso de los dos programas.

Llamada de función

```
void cmcfmd (
            unsigned char CM_PTR         conversation_ID,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmcfmd (
                           byte[]             conversation_ID,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
Cuando el programa ha emitido esta llamada la conversación no se encontraba en estado Confirmar, Confirmar-Enviar ni Confirmar-Desasignar.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.
CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR

Estado al emitirse

La conversación debe encontrarse en uno de los estados siguientes cuando el programa emite esta llamada:

- Confirmar
- Confirmar-Enviar
- Confirmar-Desasignar

Cambio de estado

El estado nuevo viene determinado por el estado anterior (el estado de la conversación cuando el programa local ha emitido la llamada Confirmed). El valor del parámetro *status_received* de la llamada Receive precedente indica el estado anterior. La Tabla 14 resume los cambios de estado posibles cuando *return_code* tiene el valor CM_OK.

Tabla 14. Cambios de estado para la llamada Confirmed

Estado anterior	Estado nuevo
Confirmar	Recibir
Confirmar-Enviar	Enviar
Confirmar-Desasignar	Restablecer

Los demás códigos de retorno no producen ningún cambio de estado.

Notas de uso

Los apartados siguientes proporcionan información de uso adicional para la llamada Confirmed.

Orígenes de las peticiones de confirmación

Una petición de confirmación se emite mediante una de las llamadas siguientes en el programa asociado:

- Confirmar
- Prepare_To_Receive si el tipo de preparación para recepción definido es CM_PREP_TO_RECEIVE_CONFIRM o CM_PREP_TO_RECEIVE_SYNC_LEVEL y el nivel de sincronización de la conversación es CM_CONFIRM
- Deallocate si el tipo de desasignación definido es CM_DEALLOCATE_CONFIRM o CM_DEALLOCATE_SYNC_LEVEL y el nivel de sincronización de la conversación es CM_CONFIRM
- Send_Data en una de las circunstancias siguientes:
  - El tipo de envío definido es CM_SEND_AND_CONFIRM
  - El tipo de envío definido es CM_SEND_AND_PREP_TO_RECEIVE y el tipo de preparación para recepción definido es CM_PREP_TO_RECEIVE_CONFIRM
  - El tipo de envío definido es CM_SEND_AND_PREP_TO_RECEIVE, el tipo de preparación para recepción es CM_PREP_TO_RECEIVE_SYNC_LEVEL y el nivel de sincronización es CM_CONFIRM
  - El tipo de envío definido es CM_SEND_AND_DEALLOCATE y el tipo de desasignación es CM_DEALLOCATE_CONFIRM
  - El tipo de envío definido es CM_SEND_AND_DEALLOCATE, el tipo de desasignación es CM_DEALLOCATE_SYNC_LEVEL y el nivel de sincronización es CM_CONFIRM

Recepción de peticiones de confirmación

El programa local recibe una petición de confirmación mediante el parámetro *status_received* de la llamada Receive. El programa local únicamente puede emitir la llamada Confirmed si el parámetro *status_received* tiene uno de los valores siguientes:

- CM_CONFIRM_RECEIVED
- CM_CONFIRM_SEND_RECEIVED
- CM_CONFIRM_DEALLOC_RECEIVED

Convert_Incoming (cmcnvi)

La llamada Convert_Incoming convierte una cadena de caracteres de EBCDIC a ASCII. Si la aplicación asociada envía datos que consisten en cadenas de caracteres EBCDIC, la aplicación local puede utilizar Convert_Incoming para convertir estas cadenas a ASCII. (Todos los parámetros CPI-C excepto los datos de las llamadas Send_Data y Receive, como por ejemplo *mode_name* y *TP_name*, siempre se especifican en ASCII y no necesitan conversión.)

Llamada de función

```
void cmcnvi (
             unsigned char CM_PTR      string,
             CM_INT32 CM_PTR           string_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

```
public native void cmcnvi (
                             byte[]          string,
                             CPICLength      string_length,
                             CPICReturnCode     return_code
                           );
```

Parámetros suministrados

Los parámetros suministrados son:

string Cadena EBCDIC que se convertirá a ASCII. La especificación CPI-C establece que la cadena puede contener cualquiera de los caracteres siguientes (juego de caracteres 640):

A-Z en mayúsculas, a-z en minúsculas, 0–9, el punto (.) y los caracteres especiales < + (& *) ; - / , % _ > ? : ' = "

Además, CPI-C para Communications Server para Linuxtambién acepta los caracteres siguientes (que puede que no estén soportados en otras implementaciones CPI-C):

! # $ @ \ { } ~

` (comilla hacia atrás)

| (barra vertical continua)

¦ (barra vertical discontinua)

¬ (carácter NOT)

¢ (centavo)

El contenido de esta cadena (hasta el número de caracteres especificado en *string_length*) se sustituirá por la cadena ASCII obtenida de la conversión.

string_length
Número de caracteres que se convertirán (1–32.767).

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

string Cadena ASCII obtenida de la conversión. Es válida hasta el número de caracteres especificado en *string_length*.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente. El parámetro string ahora contiene la cadena ASCII convertida.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por el parámetro *buffer_length* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

Esta llamada no está asociada a una conversación.

Cambio de estado

No hay ningún cambio de estado.

Nota de uso

Cuando se reciben datos en formato de almacenamiento intermedio en una conversación básica (según lo especificado por la llamada Set_Fill), el almacenamiento intermedio de datos puede contener varios registros lógicos, cada uno de los cuales consta de una cabecera de 2 ó 4 bytes (LLID) seguida de datos. La aplicación debe extraer y convertir cada una de las cadenas de datos por separado (sin incluir las cabeceras). No debe intentar convertir todo el almacenamiento intermedio en una operación ya que, en ese caso, los valores de cabecera no serán válidos.

Convert_Outgoing (cmcnvo)

La llamada Convert_Outgoing convierte una cadena de caracteres de ASCII a EBCDIC. Si la aplicación asociada necesita datos que consistan en cadenas de caracteres EBCDIC, la aplicación local puede utilizar Convert_Outgoing para convertir los datos de ASCII a EBCDIC antes de enviarlos. (Todos los parámetros CPI-C excepto los datos de las llamadas Send_Data y Receive, como por ejemplo *mode_name* y *TP_name*, siempre se especifican en ASCII y no necesitan conversión.)

Llamada de función

```
void cmcnvo (
             unsigned char CM_PTR      string,
             CM_INT32 CM_PTR           string_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

```
public native void cmcnvo (
                            byte[]          string,
                            CPICLength      string_length,
                            CPICReturnCode     return_code
                           );
```

Parámetros suministrados

Los parámetros suministrados son:

string Cadena ASCII que se convertirá a EBCDIC. La especificación CPI-C establece que la cadena puede contener cualquiera de los caracteres siguientes (juego de caracteres 640):

A-Z en mayúsculas, a-z en minúsculas, 0–9, el punto (.) y los caracteres especiales < + (& *) ; - / , % _ > ? : ' = "

Además, CPI-C para Communications Server para Linuxtambién acepta los caracteres siguientes (que puede que no estén soportados en otras implementaciones CPI-C):

! # $ @ \ { } ~

` (comilla hacia atrás)

| (barra vertical continua)

¦ (barra vertical discontinua)

¬ (carácter NOT)

¢ (centavo)

El contenido de esta cadena (hasta el número de caracteres especificado en *string_length*) se sustituirá por la cadena EBCDIC obtenida de la conversión.

string_length
Número de caracteres que se convertirán (1–32.767).

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

string Cadena EBCDIC obtenida de la conversión. Es válida hasta el número de caracteres especificado en *string_length*.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente. El parámetro string ahora contiene la cadena EBCDIC convertida.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por el parámetro *buffer_length* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

Esta llamada no está asociada a una conversación.

Cambio de estado

No hay ningún cambio de estado.

Nota de uso

Cuando se envían datos en formato de almacenamiento intermedio en una conversación básica (según lo especificado por la llamada Set_Fill), el almacenamiento intermedio de datos puede contener varios registros lógicos, cada uno de los cuales consta de una cabecera de 2 ó 4 bytes (LLID) seguida de datos. La aplicación debe convertir cada una de las cadenas de datos por separado (sin incluir las cabeceras). No debe intentar convertir todo el almacenamiento intermedio en una operación ya que, en ese caso, los valores de cabecera no serán válidos.

Deallocate (cmdeal)

La llamada Deallocate desasigna una conversación entre dos programas.

Antes de desasignar la conversación, esta llamada realiza el equivalente de la llamada Flush o Confirmed, según el nivel de sincronización y el tipo de desasignación de la conversación actual. La llamada Set_Deallocate_Type establece el tipo de desasignación.

El programa asociado recibe la notificación de desasignación mediante uno de los parámetros siguientes:

- *status_received* = CM_CONFIRM_DEALLOC_RECEIVED
- *return_code* = CM_DEALLOCATED_NORMAL
- *return_code* = CM_DEALLOCATED_ABEND

Una vez que esta llamada se ha ejecutado correctamente, el identificador de conversación ya no es válido.

Llamada de función

```
void cmdeal (
             unsigned char CM_PTR         conversation_ID,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmdeal (
                            byte[]             conversation_ID,
                            CPICReturnCode     return_code
                           );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code

Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente y la conversación se ha desasignado.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
Pueden producirse los siguientes errores de estado cuando el tipo de desasignación indica una desasignación normal (CM_DEALLOCATE_SYNC_LEVEL, CM_DEALLOCATE_FLUSH, CM_DEALLOCATE_CONFIRM):

- La conversación no se encuentra en estado Enviar ni Enviar-Pendiente.
- La conversación se encuentra en estado Enviar, pero el programa no ha terminado de enviar un registro lógico.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

```
CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR
```

Pueden devolverse los códigos de retorno siguientes cuando el tipo de desasignación está definido en CM_DEALLOCATE_CONFIRM o cuando está definido en CM_DEALLOCATE_SYNC_LEVEL y el nivel de sincronización de la conversación establecido es CM_CONFIRM. Para ver una explicación de estos códigos de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

```
CM_CONVERSATION_TYPE_MISMATCH
CM_DEALLOCATED_ABEND
CM_DEALLOCATED_ABEND_SVC
CM_DEALLOCATED_ABEND_TIMER
CM_PIP_NOT_SPECIFIED_CORRECTLY
CM_SECURITY_NOT_VALID
CM_SVC_ERROR_PURGING
CM_SYNC_LVL_NOT_SUPPORTED_PGM
CM_SYNC_LVL_NOT_SUPPORTED_LU
CM_TP_NOT_AVAILABLE_NO_RETRY
CM_TP_NOT_AVAILABLE_RETRY
CM_TPN_NOT_RECOGNIZED
CM_PROGRAM_ERROR_PURGING
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY
```

Estado al emitirse

La conversación puede encontrarse en uno de los estados que se muestran en la Tabla 15 en la página 71 cuando el programa emite la llamada Deallocate. El estado depende del valor del parámetro *deallocate_type* de la conversación, establecido por la llamada Set_Deallocate_Type.

Tabla 15. Estados de conversación al emitir la llamada Deallocate

Tipo de desasignación	Estado permitido
CM_DEALLOCATE_FLUSH CM_DEALLOCATE_CONFIRM CM_DEALLOCATE_SYNC_LEVEL	Enviar o Enviar-Pendiente
CM_DEALLOCATE_ABEND	Cualquiera excepto Restablecer

Cambio de estado

Los cambios de estado, resumidos en la Tabla 16, se basan en el valor del parámetro *return_code*.

Tabla 16. Cambios de estado para la llamada Deallocate

return_code	Estado nuevo
CM_OK	Restablecer
CM_PROGRAM_ERROR_PURGING CM_SVC_ERROR_PURGING	Recibir
CM_CONVERSATION_TYPE_MISMATCH CM_PIP_NOT_SPECIFIED_CORRECTLY CM_SECURITY_NOT_VALID CM_SYNC_LEVEL_NOT_SUPPORTED_PGM CM_SYNC_LEVEL_NOT_SUPPORTED_LU CM_TPN_NOT_RECOGNIZED CM_TP_NOT_AVAILABLE_NO_RETRY CM_TP_NOT_AVAILABLE_RETRY	Restablecer
CM_RESOURCE_FAILURE_NO_RETRY CM_RESOURCE_FAILURE_RETRY	Restablecer
CM_DEALLOCATED_ABEND CM_DEALLOCATED_ABEND_SVC CM_DEALLOCATED_ABEND_TIMER	Restablecer
Todos los demás	Sin cambio

Notas de uso

Si el tipo de desasignación de la conversación está definido en CM_DEALLOCATE_ABEND y la longitud de los datos de anotaciones es superior a 0 (cero), la LU local graba los datos de anotaciones (especificados por la llamada Set_Log_Data) en el archivo de anotaciones de error local y en la LU asociada. Para ver información sobre los datos de anotaciones, consulte “Set_Log_Data (cmsld)” en la página 137.

Una vez ejecutada la llamada Deallocate, la longitud de los datos de anotaciones se establece en 0 (cero) y los datos de anotaciones se establecen en nulos.

Delete_CPIC_Side_Information (xcmdsi)

Esta función no está disponible en CPI-C para Java.

La llamada Delete_CPIC_Side_Information suprime una entrada de información complementaria que la aplicación ha especificado anteriormente con Set_CPIC_Side_Information o especifica que una entrada del archivo de configuración ya no está disponible para ser utilizada por esta aplicación. Esta entrada se identifica mediante el nombre de destino simbólico.

Esta llamada se proporciona para asegurar la compatibilidad con CPI-C para X/Open con la especificación CPI-C para Windows; no se incluye en CPI-C 2.0 para IBM.

Llamada de función

```
void xcmdsi (
             unsigned char CM_PTR      key,
             unsigned char CM_PTR        sym_dest_name,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Parámetros suministrados

Los parámetros suministrados son:

key Este parámetro no se tiene en cuenta.

sym_dest_name
Este parámetro especifica el nombre de destino simbólico de la entrada que se suprimirá. Es una cadena de caracteres ASCII de 8 bytes y puede contener cualquier carácter visualizable.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El parámetro *sym_dest_name* ha especificado una entrada de información complementaria que no existe.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La llamada no está asociada a una conversación.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Esta llamada no modifica la información complementaria del archivo de configuración; el cambio sólo se aplica a esta aplicación. Communications Server para Linux almacena la información modificada en memoria asociada a este proceso del sistema operativo; se elimina el cambio cuando finaliza el proceso. Para ver información más detallada al respecto, consulte “Información complementaria” en la página 31.

Extract_Conversation_Context (cmectx)

AIX, LINUX

La llamada Extract_Conversation_Context devuelve el contexto de una conversación especificada. Ello permite al programa establecer su contexto actual en el valor necesario (mediante Set_Conversation_Context) antes de iniciar una nueva conversación, para asegurarse de que la nueva conversación utilice el mismo contexto.

Llamada de función

```
void cmectx (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR      context_ID,
             CM_INT32 CM_PTR           context_ID_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

```
public native void cmectx (
                            byte[]             conversation_ID,
                            byte[]          context_ID,
                            CPICLength      context_ID_length,
                            CPICReturnCode     return_code
                           );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

context_ID
: Este parámetro contiene el contexto de la conversación especificada. Sólo es válido si el parámetro *return_code* tiene el valor `CM_OK`.

context_ID_length
: Este parámetro contiene la longitud de *context_ID* (1–32 bytes). Sólo es válido si el parámetro *return_code* tiene el valor `CM_OK`.

return_code
: Los valores posibles son:

  CM_OK La llamada se ha ejecutado correctamente.

  CM_PROGRAM_PARAMETER_CHECK
  : El valor especificado por *conversation_ID* no es válido.

  CM_PROGRAM_STATE_CHECK
  : La conversación especificada por *conversation_ID* se encuentra en estado Inicializar o Inicializar-Entrante.

Para ver una explicación del código de retorno siguiente, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.
CM_PRODUCT_SPECIFIC_ERROR

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer, Inicializar o Inicializar-Entrante.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Esta llamada no establece el contexto actual del programa en el valor extraído. Para ello el programa debe llamar Set_Conversation_Context.

Una aplicación utiliza la llamada Extract_Conversation_Context, seguida de Set_Conversation_Context, en las situaciones siguientes:

- Cuando participa en conversaciones múltiples y desea asignar una nueva conversación utilizando el mismo contexto que una conversación ya existente.
- Cuando una llamada CPI-C que asigna un nuevo contexto finaliza en la modalidad sin bloqueo. Por ejemplo, si Accept_Incoming finaliza de inmediato con *return_code* con el valor CM_OK, el contexto actual del programa se establece en el contexto de la nueva conversación; sin embargo, si Accept_Incoming devuelve CM_OPERATION_INCOMPLETE, una Wait_For_Conversation posterior que devuelva el resultado de Accept_Incoming no cambiará el contexto actual del programa. El programa debe utilizar Extract_Conversation_Context y Set_Conversation_Context para establecer el contexto actual en el valor correcto.

Extract_Conversation_Security_Type (xcecst)

Esta función no está disponible en CPI-C para Java.

La llamada Extract_Conversation_Security_Type devuelve el tipo de seguridad de una conversación especificada.

Esta llamada se proporciona para asegurar la compatibilidad con CPI-C para X/Open con la especificación CPI-C para Windows; no se incluye en CPI-C 2.0 para IBM.

Llamada de función

```
void xcecst (
             unsigned char CM_PTR         conversation_ID,
             XC_CONVERSATION_SECURITY_TYPE CM_PTR  conversation_security_type,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

conversation_security_type
Especifica la información que necesita la LU asociada a fin de validar el acceso al programa invocado. Los valores posibles son:

AIX, LINUX

CM_SECURITY_NONE
El programa invocado no utiliza ninguna seguridad de conversación.

CM_SECURITY_SAME
Este valor se utiliza cuando el programa invocado, que se ha invocado con un identificador de usuario y una contraseña válidos, invoca otro programa (como se muestra en el Capítulo 1, “Conceptos”, en la página 1). Suponga que el programa A invoca el programa B con un identificador de usuario y una contraseña válidos y, a su vez, el programa B invoca el programa C; si el programa B especifica el valor CM_SECURITY_SAME, CPI-C enviará un indicador de seguridad ya verificada a la LU del programa C. Este indicador ordena al programa C que no solicite la contraseña (si el programa C está configurado para aceptar un indicador de seguridad ya verificada).

CM_SECURITY_PROGRAM
El programa invocado utiliza la seguridad de conversación y por consiguiente requiere un identificador de usuario y una contraseña.

CM_SECURITY_PROGRAM_STRONG
Igual que para CM_SECURITY_PROGRAM, con la excepción de que el nodo local no debe enviar la contraseña a través de la red en formato de texto no cifrado. Este valor sólo se puede utilizar si el sistema remoto proporciona soporte para la sustitución de contraseñas.

WINDOWS

XC_SECURITY_NONE
Equivalente a CM_SECURITY_NONE

XC_SECURITY_SAME
Equivalente a CM_SECURITY_SAME

XC_SECURITY_PROGRAM
Equivalente a CM_SECURITY_PROGRAM

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Extract_Conversation_Security_User_ID (cmecsu)

WINDOWS

Esta llamada es la equivalente a CPI-C para Windows de la llamada Extract_Security_User_ID (cmesui) de CPI-C para AIX o Linux. Las dos llamadas se utilizan exactamente de la misma manera, salvo que los nombres son diferentes. Para obtener más información sobre Extract_Conversation_Security_User_ID, consulte la llamada "Extract_Security_User_ID (cmesui o cmecsu)" en la página 86, y sustituya el nombre de función y el seudónimo de AIX o Linux por el nombre de función y el seudónimo de Windows según lo indicado.

Extract_Conversation_Security_User_ID (xcecsu)

Esta función no está disponible en CPI-C para Java.

Esta llamada devuelve el identificador de usuario que se está utilizando en una conversación especificada.

La llamada proporciona compatibilidad para aplicaciones que utilizan la definición de X/Open CPI-C. Se ha incorporado en CPI-C 2.0 para IBM como la llamada Extract_Security_User_ID (cmesui). Utilice cmesui siempre que sea posible para lograr una mayor portabilidad del programa a otras plataformas.

Los parámetros de esta llamada son los mismos que los de la llamada cmesui. Para ver más información sobre cmesui, consulte la llamada "Extract_Security_User_ID (cmesui o cmecsu)" en la página 86.

Extract_Conversation_State (cmecs)

La llamada Extract_Conversation_State devuelve el estado de la conversación especificada.

Llamada de función

```
void cmecs (
            unsigned char CM_PTR         conversation_ID,
            CM_CONVERSATION_STATE CM_PTR     conversation_state,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmecs (
                           byte[]             conversation_ID,
                           CPICConversationState  conversation_state,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

conversation_state
Especifica el estado de conversación. Los valores posibles son:
CM_INITIALIZE_STATE
CM_INITIALIZE_INCOMING_STATE
CM_SEND_STATE
CM_RECEIVE_STATE
CM_SEND_PENDING_STATE
CM_CONFIRM_STATE
CM_CONFIRM_SEND_STATE
CM_CONFIRM_DEALLOCATE_STATE

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.
CM_OPERATION_INCOMPLETE
CM_PRODUCT_SPECIFIC_ERROR

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Extract_Conversation_Type (cmect)

La llamada Extract_Conversation_Type devuelve el tipo de conversación (correlacionada o básica) de la conversación especificada.

Llamada de función

```
void cmect (
            unsigned char CM_PTR         conversation_ID,
            CM_CONVERSATION_TYPE CM_PTR   conversation_type,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmect (
                           byte[]             conversation_ID,
                           CPICConversationType  conversation_type,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

conversation_type
: Este parámetro especifica el tipo de conversación. Los valores posibles son:
  CM_BASIC_CONVERSATION
  CM_MAPPED_CONVERSATION

return_code
: Los valores posibles son:

  CM_OK La llamada se ha ejecutado correctamente.

  CM_PROGRAM_PARAMETER_CHECK
  : El valor especificado por *conversation_ID* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Extract_CPIC_Side_Information (xcmesi)

Esta función no está disponible en CPI-C para Java.

La llamada Extract_CPIC_Side_Information devuelve la información complementaria de un número o nombre de destino simbólico de entrada.

Esta llamada se proporciona para asegurar la compatibilidad con CPI-C para X/Open con la especificación CPI-C para Windows; no se incluye en CPI-C 2.0 para IBM.

Llamada de función

```
  void xcmesi  (
                CM_INT32 CM_PTR             entry_number,
                unsigned char CM_PTR        sym_dest_name,
                SIDE_INFO CM_PTR          side_info_entry,
                CM_INT32 CM_PTR           side_info_entry_length,
                CM_RETURN_CODE CM_PTR        return_code
               );
typedef struct side_info_entry
{
 unsigned char     sym_dest_name[8];      /* Nombre de destino simbólico    */
 unsigned char    partner_LU_name[17];        /* Nombre de LU asociada       */
                                              /* completamente calificado    */
 unsigned char     reserved[3];           /* Reservado                      */
 XC_TP_NAME_TYPE   TP_name_type;          /* Tipo de nombre de TP           */
 unsigned char     TP_name[64];           /* Nombre de TP                   */
 unsigned char     mode_name[8];          /* Nombre de modalidad            */
 XC_CONVERSATION_SECURITY_TYPE
                  conversation_security_type; /* Tipo seguridad conversación */
 unsigned char     security_user_ID[8];   /* Identificador de usuario       */
 unsigned char     security_password[8];  /* Contraseña                     */
} SIDE_INFO;
```

Parámetros suministrados

Los parámetros suministrados son:

entry_number
Este parámetro no se tiene en cuenta.

sym_dest_name
Este parámetro especifica el nombre de destino simbólico que se debe buscar. Es una cadena de caracteres ASCII de 8 bytes y puede contener cualquier carácter visualizable.

side_info_entry_length

AIX, LINUX

Este valor siempre debe estar definido en `sizeof(SIDE_INFO)`.

WINDOWS

Este valor siempre debe establecerse en `124`.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

side_info_entry
Este parámetro especifica el contenido de una entrada de información complementaria, como se indica a continuación.

side_info_entry.sym_dest_name
Nombre de destino simbólico que identifica la entrada de información complementaria. El parámetro *sym_dest_name* es una cadena de caracteres ASCII de 8 bytes y puede contener cualquier carácter visualizable.

side_info_entry.partner_LU_name
Nombre de la LU asociada completamente calificado. Este nombre consta de dos cadenas de caracteres, cada una de ellas de 1–8 bytes, concatenadas por un punto.

side_info_entry.TP_name_type
Tipo de TP de destino (los caracteres válidos para un nombre de TP vienen determinados por el tipo de TP). Los valores posibles son:

XC_APPLICATION_TP
TP de aplicación. Todos los caracteres del nombre de TP deben ser caracteres ASCII válidos.

XC_SNA_SERVICE_TP
TP de servicio. El nombre de TP se debe especificar como una cadena ASCII de 8 caracteres que represente los dígitos hexadecimales de un nombre de 4 caracteres. Por ejemplo, si la representación hexadecimal del nombre es 0x21F0F0F8, defina el parámetro *TP_name* en la cadena de 8 caracteres "21F0F0F8".

El primer carácter (representado por dos bytes) debe ser un valor hexadecimal en el rango `0x0`–`0x3F`, salvo `0x0E` y `0x0F`; los caracteres restantes (cada uno representado por dos bytes) deben ser caracteres EBCDIC válidos.

side_info_entry.TP_name
Nombre de TP del TP de destino.

side_info_entry.mode_name
Nombre de la modalidad utilizada para acceder al TP de destino.

side_info_entry.conversation_security_type
Especifica si el TP de destino utiliza la seguridad de conversación. Los valores posibles son:

XC_SECURITY_NONE
El TP de destino no utiliza la seguridad de conversación.

XC_SECURITY_PROGRAM
El TP de destino utiliza la seguridad de conversación. Los parámetros *security_user_ID* y *security_password* especificados más abajo se utilizarán para acceder al TP de destino.

XC_SECURITY_SAME
El TP de destino utiliza la seguridad de conversación y puede aceptar un indicador de seguridad "ya verificada" del TP local. (Esto indica que el propio TP local ha sido invocado por otro TP y ha verificado el identificador de usuario y la contraseña de seguridad proporcionados por este TP.) El parámetro *security_user_ID* especificado más abajo se utilizará para acceder al TP de destino; no se necesita ninguna contraseña.

side_info_entry.security_user_ID
Identificador de usuario que se utiliza para acceder al TP asociado. Este parámetro no es necesario si el parámetro *conversation_security_type* está establecido en `XC_SECURITY_NONE`.

Para asegurar la compatibilidad con la implementación de CPI-C de X/Open, este verbo sólo devuelve 8 caracteres para el identificador de usuario, aunque los identificadores de usuario de seguridad pueden tener hasta 10 caracteres. Para asegurarse de que obtiene el ID de usuario completo, debe extraerlo explícitamente utilizando la llamada Extract_Security_User_ID (Extract_Conversation_Security_User_ID para sistemas Windows), en lugar del valor devuelto aquí.

side_info_entry.security_password
Este parámetro está reservado; la información de contraseña nunca se devuelve a la aplicación.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:
- El parámetro *sym_dest_name* no es válido.
- El parámetro *side_info_entry_length* no está establecido en `sizeof(SIDE_INFO)`

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

Esta llamada no está asociada a una conversación.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el identificador de usuario de seguridad de la información complementaria no está definido, el campo de identificador de usuario de seguridad se devuelve rellenado con espacios.

Extract_Local_LU_Name (cmelln)

La llamada Extract_Local_LU_Name devuelve el alias de la LU local para una conversación especificada.

Esta llamada no forma parte de la especificación CPI-C estándar y puede no estar disponible en otras implementaciones. En especial, no está soportada en otras implementaciones de CPI-C para Java.

Llamada de función

```
void cmelln   (
               unsigned char CM_PTR         conversation_ID,
               unsigned char CM_PTR         lu_alias,
               CM_RETURN_CODE CM_PTR        return_code
              );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmelln (
                             byte[]             conversation_ID,
                             byte[]          lu_alias,
                             CPICReturnCode     return_code
                           );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

lu_alias
El alias LU de la LU local.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

El alias de LU devuelto por esta llamada no tiene que establecerse mediante Set_Local_LU_Name, como se describe en "Set_Local_LU_Name (cmslln)" en la página 136. Puede utilizarse cualquiera de los métodos descritos en "Especificación de la LU local" en la página 35.

Extract_Maximum_Buffer_Size (cmembs)

AIX, LINUX

La llamada Extract_Maximum_Buffer_Size devuelve el tamaño máximo de un almacenamiento intermedio de datos CPI-C. De esta forma se define la cantidad máxima de datos que puede enviarse en una llamada Send_Data o recibirse en una llamada Receive.

CPI-C de Communications Server para Linux utiliza siempre un tamaño de almacenamiento intermedio de datos de 32.767 bytes. Sin embargo, por motivos de compatibilidad con otras implementaciones de CPI-C (o con futuras versiones de Communications Server para Linux), una aplicación no debe confiar en este valor y debe utilizar esta llamada para determinar el mayor tamaño de almacenamiento intermedio que puede utilizar.

Llamada de función

```
void cmembs (
             CM_INT32 CM_PTR           maximum_buffer_size,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

```
public native void cmembs (
                            CPICLength      maximum_bufer_size,
                            CPICReturnCode     return_code
                          );
```

Parámetros suministrados

No hay ningún parámetro suministrado para esta llamada.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

maximum_buffer_size
: Este parámetro especifica la longitud del almacenamiento intermedio de datos.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

Esta llamada no está asociada a ninguna conversación.

Cambio de estado

No hay ningún cambio de estado.

Extract_Mode_Name (cmemn)

La llamada Extract_Mode_Name devuelve el nombre de modalidad y la longitud del nombre de modalidad de una conversación especificada.

Llamada de función

```
void cmemn (
            unsigned char CM_PTR         conversation_ID,
            unsigned char CM_PTR       mode_name,
            CM_INT32 CM_PTR            mode_name_length,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmemn (
                           byte[]             conversation_ID,
                           byte[]          mode_name,
                           CPICLength      mode_name_length,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

mode_name
Este parámetro especifica la dirección inicial del nombre de modalidad.

mode_name_length
Este parámetro especifica la longitud del nombre de modalidad.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Extract_Partner_LU_Name (cmepln)

La llamada Extract_Partner_LU_Name devuelve el nombre de LU asociada y la longitud del nombre de LU asociada de una conversación especificada. Puede ser un nombre de alias de hasta 8 bytes o un nombre de red completamente calificado de hasta 17 bytes.

Llamada de función

```
void cmepln (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR      partner_LU_name,
             CM_INT32 CM_PTR           partner_LU_name_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmepln (
                           byte[]             conversation_ID,
                           byte[]          partner_LU_name,
                           CPICLength      partner_LU_name_length,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

partner_LU_name
Este parámetro especifica la variable que contiene el nombre de LU asociada. (El programa debe proporcionar un puntero a una variable disponible.)

partner_LU_name_length
Este parámetro especifica la longitud del nombre de LU asociada.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
La conversación especificada por *conversation_ID* se encuentra en estado Inicializar-Entrante.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer o Inicializar-Entrante.

Cambio de estado

No hay ningún cambio de estado.

Extract_Security_User_ID (cmesui o cmecsu)

La llamada Extract_Security_User_ID devuelve el identificador de usuario que se utiliza en una conversación especificada.

WINDOWS

Esta llamada se denomina Extract_Conversation_Security_User_ID, con el seudónimo cmecsu, para obtener compatibilidad con la interfaz CPI-C de Windows.

Llamada de función

```
void cmesui (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR      security_user_ID,
             CM_INT32 CM_PTR           security_user_ID_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

WINDOWS

Para los sistemas Windows, sustituya cmesui por cmecsu.

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmesui (
                           byte[]             conversation_ID,
                           byte[]          security_user_ID,
                           CPICLength      security_user_ID_length,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

security_user_ID
: Especifica el identificador de usuario utilizado para establecer la conversación.

security_user_ID_length
: Especifica la longitud de *security_user_ID*.

  El rango de este valor es de 1 a 10 caracteres (sistemas AIX o Linux), o bien 1 a 8 caracteres (sistemas Windows). Si *security_user_ID_length* está definido en `0` (cero), el parámetro *security_user_ID_length* no se tiene en cuenta; esto equivale a establecer *security_user_ID* en una cadena nula.

return_code

Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

El valor de *security_user_ID* no está rellenado con blancos. Sólo es significativo hasta *security_user_ID_length*.

Extract_Sync_Level (cmesl)

La llamada Extract_Sync_Level devuelve el nivel de sincronización de una conversación especificada.

Llamada de función

```
void cmesl (
            unsigned char CM_PTR         conversation_ID,
            CM_INT32 CM_PTR             sync_level,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmesl (
                           byte[]             conversation_ID,
                           CPICSyncLevel    sync_level,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID

Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

sync_level
Este parámetro indica el nivel de sincronización de la conversación. Los valores posibles son:

CM_NONE
Los programas no llevarán a cabo el proceso de confirmación.

CM_CONFIRM
Los programas pueden llevar a cabo el proceso de confirmación.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
La conversación especificada por *conversation_ID* se encuentra en estado Inicializar-Entrante.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer o Inicializar-Entrante.

Cambio de estado

No hay ningún cambio de estado.

Extract_TP_Name (cmetpn)

La llamada Extract_TP_Name devuelve el nombre de TP y la longitud del nombre de TP del TP invocado para una conversación especificada.

Si una aplicación ha utilizado la llamada Specify_Local_TP_Name para aceptar peticiones Allocate entrantes para más de un nombre de TP y posteriormente ha emitido Accept_Conversation o Accept_Incoming para aceptar una petición Allocate entrante, puede utilizar esta llamada para determinar qué nombre de TP se ha especificado en la petición Allocate entrante.

Llamada de función

```
void cmetpn (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR      TP_name,
             CM_INT32 CM_PTR           TP_name_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmetpn (
                             byte[]             conversation_ID,
                             byte[]          TP_name,
                             CPICLength      TP_name_length,
                             CPICReturnCode     return_code
                           );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

TP_name
Este parámetro especifica la dirección inicial del nombre de TP.

TP_name_length
Este parámetro especifica la longitud del nombre de TP.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
La conversación se encuentra en estado Restablecer o Inicializar-Entrante.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer o Inicializar-Entrante.

Cambio de estado

No hay ningún cambio de estado.

Flush (cmflus)

La llamada Flush envía el contenido del almacenamiento intermedio de envío de la LU local a la LU asociada (y al programa). 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 la llamada Send_Data se acumulan en el almacenamiento intermedio de envío de la LU local hasta que se produce una de las situaciones siguientes:

- El programa local emite la llamada Flush u otra llamada que vacía el almacenamiento intermedio de envío de la LU. (Algunos tipos de envío, definidos por la llamada Set_Send_Type, incluyen la función de vaciar.)
- El almacenamiento intermedio se llena.

La petición de asignación generada por la llamada Allocate y la información de error generada por la llamada Send_Error también se guardan en el almacenamiento intermedio.

Llamada de función

```
void cmflus (
             unsigned char CM_PTR         conversation_ID,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmflus (
                             byte[]             conversation_ID,
                             CPICReturnCode     return_code
                           );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación.

La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
La conversación no se encontraba en estado Enviar ni Enviar-Pendiente cuando el programa ha emitido esta llamada.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR

Estado al emitirse

La conversación debe encontrarse en estado Enviar o Enviar-Pendiente.

Cambio de estado

Si la llamada finaliza correctamente (*return_code* = CM_OK), la conversación se encuentra en estado Enviar.

Los demás códigos de retorno no producen ningún cambio de estado.

Initialize_Conversation (cminit)

El programa que invoca emite la llamada Initialize_Conversation para obtener un identificador de conversación de 8 bytes y definir los valores iniciales de las características de la conversación.

Los valores iniciales son valores por omisión de CPI-C o proceden de la información complementaria asociada al nombre de destino simbólico. Para ver más información sobre los valores iniciales y la información complementaria, consulte el Capítulo 2, "Desarrollo de aplicaciones CPI-C", en la página 19.

Una vez ejecutada correctamente esta llamada, CPI-C genera un identificador de conversación. Este identificador es un parámetro necesario para todas las demás llamadas CPI-C emitidas por el programa invocado para esta conversación.

Los valores iniciales pueden modificarse mediante las llamadas Set_*.

Llamada de función

```
void cminit (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR        sym_dest_name,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cminit (
                           byte[]             conversation_ID,
                           String          sym_dest_name,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

El parámetro suministrado es:

sym_dest_name
: Este parámetro especifica el nombre de destino simbólico—el nombre asociado a una entrada de información complementaria cargada desde el archivo de configuración de Communications Server para Linux o definida por las llamadas Set_CPIC_Side_Information).

  El parámetro es una cadena de caracteres ASCII de 8 bytes y puede contener cualquier carácter visualizable. Este parámetro también puede definirse en 8 espacios. En este caso, el programa que invoca debe utilizar las llamadas siguientes antes de emitir la llamada Allocate:

  - Set_Mode_Name
  - Set_Partner_LU_Name
  - Set_TP_Name

  Para ver información más detallada sobre la entrada de información complementaria, consulte “Set_CPIC_Side_Information (xcmssi)” en la página 126.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

conversation_ID
: Identificador de la conversación. Las llamadas CPI-C posteriores lo utilizan.

return_code
: Los valores posibles son:

  CM_OK La llamada se ha ejecutado correctamente.

  CM_PROGRAM_PARAMETER_CHECK
  : Se ha producido una de estas situaciones:

    - El valor especificado por *sym_dest_name* no coincide con un nombre de destino simbólico definido en el archivo de configuración o con uno especificado por el programa mediante Set_CPIC_Side_Information.
    - El parámetro *conversation_ID* no es válido.

  CM_PRODUCT_SPECIFIC_ERROR
  : Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación se encuentra en estado Restablecer.

Cambio de estado

Si *return_code* es `CM_OK`, la conversación cambia al estado Inicializar. En el caso de los demás códigos de retorno, el estado de conversación no se modifica.

Notas de uso

Si la información complementaria contiene un valor que no es válido para una característica de conversación, o si la llamada Set_* la define en un valor que no es válido, el error se devuelve en la llamada Allocate.

Initialize_For_Incoming (cminic)

AIX, LINUX

El programa invocado emite la llamada Initialize_For_Incoming para obtener un identificador de conversación de 8 bytes. Después el programa acepta la conversación mediante la llamada Accept_Incoming.

Emitir Initialize_For_Incoming seguido de Accept_Incoming equivale a emitir Accept_Conversation. La diferencia consiste en que Set_Processing_Mode puede emitirse entre Initialize_For_Incoming y Accept_Incoming para asegurarse de que Accept_Incoming opere en la modalidad sin bloqueo, mientras que Accept_Conversation siempre opera en la modalidad con bloqueo.

Llamada de función

```
void cminic (
             unsigned char CM_PTR         conversation_ID,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

```
public native void cminic (
                             byte[]             conversation_ID,
                             CPICReturnCode     return_code
                           );
```

Parámetros suministrados

No hay ningún parámetro suministrado para esta llamada.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

conversation_ID
Identificador de la conversación. Las llamadas CPI-C posteriores lo utilizan.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación se encuentra en estado Restablecer.

Cambio de estado

Si *return_code* es CM_OK, la conversación cambia al estado Inicializar. De lo contrario, el estado de conversación no se modifica.

Prepare_To_Receive (cmptr)

La llamada Prepare_To_Receive cambia el estado de la conversación para el programa local de Enviar a Recibir. Antes de cambiar el estado de conversación, esta llamada realiza el equivalente a una de las llamadas siguientes:

- La llamada Flush, con lo que envía el contenido del almacenamiento intermedio de envío de la LU local a la LU asociada (y al programa), si se cumple alguna de las condiciones siguientes:
  - El tipo de preparación para recepción de la conversación está definido en CM_PREP_TO_RECEIVE_FLUSH
  - El tipo de preparación para recepción está definido en CM_PREP_TO_RECEIVE_SYNC_LEVEL y el nivel de sincronización de la conversación es CM_NONE
- La llamada Confirm, con lo que envía el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación al programa asociado, si se cumple alguna de las condiciones siguientes:
  - El tipo de preparación para recepción de la conversación está definido en CM_PREP_TO_RECEIVE_CONFIRM
  - El tipo de preparación para recepción está definido en CM_PREP_TO_RECEIVE_SYNC_LEVEL y el nivel de sincronización de la conversación es CM_CONFIRM

El tipo de preparación para recepción se define mediante la llamada Set_Prepare_To_Receive_Type; el nivel de sincronización se define mediante la llamada Set_Sync_Level.

Una vez que esta llamada se ha ejecutado correctamente, el programa local puede recibir datos.

Llamada de función

```
void cmptr (
            unsigned char CM_PTR         conversation_ID,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmptr (
                           byte[]             conversation_ID,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
Se ha producido una de estas situaciones:

- El estado de conversación no es Enviar ni Enviar-Pendiente.
- La conversación básica se encuentra en estado Enviar. Sin embargo, el programa no ha terminado de enviar un registro lógico.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

```
CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR
```

Los códigos de retorno siguientes pueden producirse si el tipo de preparación para recepción de la conversación está definido en CM_PREP_TO_RECEIVE_CONFIRM o si el tipo de preparación para recepción está definido en CM_PREP_TO_RECEIVE_SYNC_LEVEL, y el nivel de sincronización de la conversación es CM_CONFIRM. Para ver una descripción de los mismos, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

```
CM_CONVERSATION_TYPE_MISMATCH
CM_DEALLOCATED_ABEND
CM_DEALLOCATED_ABEND_SVC
CM_DEALLOCATED_ABEND_TIMER
CM_PIP_NOT_SPECIFIED_CORRECTLY
CM_PROGRAM_ERROR_PURGING
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY
CM_SECURITY_NOT_VALID
CM_SVC_ERROR_PURGING
CM_SYNC_LVL_NOT_SUPPORTED_PGM
CM_SYNC_LVL_NOT_SUPPORTED_LU
CM_TPN_NOT_RECOGNIZED
CM_TP_NOT_AVAILABLE_NO_RETRY
CM_TP_NOT_AVAILABLE_RETRY
```

Estado al emitirse

La conversación puede encontrarse en estado Enviar o Enviar-Pendiente.

Cambio de estado

Los cambios de estado resumidos en la Tabla 17 se basan en el valor del parámetro *return_code*.

Tabla 17. Cambios de estado para la llamada Prepare_To_Receive

return_code	**Estado nuevo**
CM_OK	Recibir
CM_PROGRAM_ERROR_PURGING	Recibir
CM_SVC_ERROR_PURGING	
CM_CONVERSATION_TYPE_MISMATCH	Restablecer
CM_PIP_NOT_SPECIFIED_CORRECTLY	
CM_SECURITY_NOT_VALID	
CM_SYNC_LEVEL_NOT_SUPPORTED_PGM	
CM_SYNC_LEVEL_NOT_SUPPORTED_LU	
CM_TPN_NOT_RECOGNIZED	
CM_TP_NOT_AVAILABLE_NO_RETRY	
CM_TP_NOT_AVAILABLE_RETRY	
CM_DEALLOCATED_ABEND	Restablecer
CM_RESOURCE_FAILURE_NO_RETRY	
CM_RESOURCE_FAILURE_RETRY	
CM_DEALLOCATED_ABEND_SVC	Restablecer
CM_DEALLOCATED_ABEND_TIMER	
Todos los demás	Sin cambio

Notas de uso

La conversación no cambia al estado Enviar (o Enviar-Pendiente) para el programa asociado hasta que éste recibe uno de los valores siguientes mediante el parámetro *status_received* de la llamada Receive:

- CM_SEND_RECEIVED
- CM_CONFIRM_SEND_RECEIVED y responde con la llamada Confirmed o Send_Error

Receive (cmrcv)

La llamada Receive recibe del programa asociado los datos que están actualmente disponibles.

Si no hay datos disponibles y el tipo de recepción (definido por la llamada Set_Receive_Type) es CM_RECEIVE_AND_WAIT, el programa local espera a que lleguen los datos. Si el tipo de recepción establecido es CM_RECEIVE_IMMEDIATE, el programa local no espera.

WINDOWS

Si se emite la llamada Receive en modalidad sin bloqueo (especificada por una llamada Set_Processing_Mode anterior), la aplicación puede emitir las siguientes llamadas mientras Receive está pendiente:

- Request_To_Send
- Send_Error

- Test_Request_to_Send_Received
- Cancel_Conversation
- Deallocate

Si la aplicación utiliza una de estas llamadas en modalidad sin bloqueo mientras la llamada Receive está pendiente, debe utilizar Specify_Windows_Handle para permitir que CPI-C devuelva los resultados de llamadas sin bloqueo. No emite Wait_For_Conversation si hay otra llamada pendiente además de Receive; los resultados de esta llamada no están definidos si hay más de una llamada pendiente en la misma conversación.

Recepción de los datos por parte de un programa

El proceso para recibir datos es el siguiente:

- El programa local emite una llamada Receive hasta que termina de recibir una unidad de datos completa. Puede que el programa local tenga que emitir varias veces la llamada Receive para recibir una unidad de datos completa. El parámetro *data_received* indica si ha finalizado la recepción de datos.

  Los datos recibidos pueden ser:

  – Un registro de datos transmitido en una conversación correlacionada
  – Un registro lógico transmitido en una conversación básica con la característica de relleno de la conversación definida en `CM_FILL_LL`
  – Un almacenamiento intermedio de datos recibido aparte de su formato de registro lógico en una conversación básica con la característica de relleno definida en `CM_FILL_BUFFER`

  Una vez que se ha recibido una unidad de datos completa, el programa local puede manipularla.
- El programa local determina la siguiente acción que debe llevarse a cabo a partir de la información de control recibida mediante el parámetro *status_received*. Puede que el programa local tenga que volver a emitir la llamada Receive para recibir la información de control.

El tipo de conversación se define mediante la llamada Set_Conversation_Type; la característica de relleno se define mediante la llamada Set_Fill.

Llamada de función

```
void cmrcv (
            unsigned char CM_PTR         conversation_ID,
            unsigned char CM_PTR                buffer,
            CM_INT32 CM_PTR               requested_length,
            CM_DATA_RECEIVED_TYPE CM_PTR  data_received,
            CM_INT32 CM_PTR               received_length,
            CM_STATUS_RECEIVED CM_PTR     status_received,
            CM_INT32 CM_PTR               request_to_send_received,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmrcv (
                           byte[]             conversation_ID,
                           byte[]                            buffer,
                           CPICLength                        requested_length,
                           CPICDataReceivedType              data_received,
                           CPICLength                        received_length,
                           CPICStatusReceived                status_received,
                           CPICControlInformationReceived    request_to_send_received,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

requested_length
Número máximo de bytes de datos que recibirá el programa local.

El rango de este valor es `0–32.767`.

buffer Dirección del almacenamiento intermedio que contendrá los datos recibidos por el programa local.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

buffer El almacenamiento intermedio de datos de la aplicación contiene datos si se cumplen las condiciones siguientes:

- El parámetro *data_received* está definido en un valor distinto de `CM_NO_DATA_RECEIVED`
- El parámetro *return_code* está definido en `CM_OK` o en `CM_DEALLOCATED_NORMAL`

data_received
Este parámetro indica si el programa ha recibido datos. A continuación figuran los valores posibles. Estos códigos no son relevantes salvo que *return_code* esté definido en `CM_OK` o `CM_DEALLOCATED_NORMAL`.

Puede devolverse `CM_DATA_RECEIVED` si la característica de relleno de la conversación está definida en `CM_FILL_BUFFER`, lo que indica que el programa recibe datos aparte de su formato lógico. El programa local ha recibido datos hasta que ha alcanzado *requested_length* o el fin de los datos.

El fin de los datos se indica mediante uno de los elementos siguientes:

- Un cambio a otro estado de conversación, en función de los parámetros *return_code*, *status_received* y *data_received*
- Una condición de error

Si el tipo de recepción de la conversación está definido en `CM_RECEIVE_IMMEDIATE`, los datos recibidos pueden ser menos de *requested_length* si ha llegado una cantidad de datos inferior del programa asociado.

CM_COMPLETE_DATA_RECEIVED
En una conversación correlacionada, este parámetro indica que el programa local ha recibido un registro de datos completo o la última parte de un registro de datos.

En una conversación básica con la característica de relleno definida en CM_FILL_LL, este valor indica que el programa local ha recibido un registro lógico completo o el final de un registro lógico.

CM_INCOMPLETE_DATA_RECEIVED
En una conversación correlacionada, este valor indica que el programa local ha recibido un registro de datos incompleto; el parámetro *requested_length* ha especificado un valor inferior a la longitud del registro de datos (o inferior al registro de datos restante si no es la primera llamada Receive que lee el registro). La cantidad de datos recibidos es igual al parámetro *requested_length*.

En una conversación básica con la característica de relleno definida en CM_FILL_LL, este valor indica que el programa local ha recibido un registro lógico incompleto. La cantidad de datos recibidos es igual al parámetro *requested_length*. (Si los datos recibidos se han truncado, la longitud de los datos será inferior a *requested_length*.)

Al recibir este valor, el programa local normalmente vuelve a emitir la llamada Receive para recibir la parte siguiente del registro.

CM_NO_DATA_RECEIVED
El programa no ha recibido datos.

Nota: Si el parámetro *return_code* tiene el valor CM_OK, puede haber información de estado disponible mediante el parámetro *status_received*.

received_length
Indica el número máximo de bytes de datos que ha recibido el programa local en esta llamada Recibir. Si el parámetro *return_code* o *data_received* indica que el programa no ha recibido datos, este valor no es relevante.

status_received
Este parámetro indica cambios en el estado de la conversación. Estos códigos no son relevantes salvo que *return_code* esté definido en CM_OK. Los valores posibles son:

CM_NO_STATUS_RECEIVED
No se ha recibido ningún cambio de conversación en esta llamada.

CM_SEND_RECEIVED
Para el programa asociado, la conversación ha pasado al estado Recibir. Para el programa local, la conversación ahora se encuentra en estado Enviar si no se han recibido datos en esta llamada o en estado Enviar-Pendiente si se han recibido datos en esta llamada.

Al recibir este valor, el programa local normalmente utiliza la llamada Send_Data para empezar a enviar datos.

CM_CONFIRM_DEALLOC_RECEIVED
El programa asociado ha emitido la llamada Deallocate con confirmación solicitada. Para el programa local, la conversación ahora se encuentra en estado Confirmar-Desasignar.

Al recibir este valor, el programa local normalmente emite la llamada Confirmed.

CM_CONFIRM_RECEIVED
El programa asociado ha emitido la llamada Confirm. Para el programa local, la conversación se encuentra en estado Confirmar.

Al recibir este valor, el programa local normalmente emite la llamada Confirmed.

CM_CONFIRM_SEND_RECEIVED
Para el programa asociado, la conversación ha pasado al estado Recibir y el programa local ha recibido una petición de confirmación. Para el programa local, la conversación ahora se encuentra en estado Confirmar-Enviar.

El programa normalmente responde emitiendo la llamada Confirmed. Una vez ejecutada correctamente la llamada Confirmed, la conversación cambia al estado Enviar para el programa local.

request_to_send_received
Indicador de petición de envío recibida. Los valores posibles son:

CM_REQ_TO_SEND_RECEIVED
El programa asociado ha emitido la llamada Request_To_Send, que solicita al programa local que cambie la conversación al estado Recibir.

CM_REQ_TO_SEND_NOT_RECEIVED
El programa asociado no ha emitido la llamada Request_To_Send.

Este valor no es relevante si el parámetro *return_code* tiene uno de los valores siguientes:

- CM_PROGRAM_PARAMETER_CHECK
- CM_PROGRAM_STATE_CHECK

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_UNSUCCESSFUL
El tipo de recepción definido es CM_RECEIVE_IMMEDIATE y actualmente no hay datos ni información de estado disponible del programa asociado.

CM_DEALLOCATED_NORMAL
La conversación se ha desasignado normalmente. El programa asociado ha emitido la llamada Deallocate con el tipo de desasignación de la conversación definido en uno de los siguientes:

- CM_DEALLOCATE_FLUSH
- CM_DEALLOCATE_SYNC_LEVEL con el nivel de sincronización de la conversación especificado como CM_NONE

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* no es válido.
- El valor especificado por *requested_length* está fuera del rango.

Si el programa recibe este código de retorno, los demás parámetros devueltos no son válidos.

CM_PROGRAM_STATE_CHECK
Se ha producido una de estas situaciones:

- El tipo de recepción establecido es CM_RECEIVE_AND_WAIT y el estado de conversación no es Recibir, Enviar ni Enviar-Pendiente.
- El tipo de recepción establecido es CM_RECEIVE_IMMEDIATE y el estado de conversación no es Recibir.
- La conversación básica se encuentra en estado Enviar, el tipo de recepción establecido es CM_RECEIVE_AND_WAIT y el programa no había terminado de enviar un registro lógico.

Si el programa recibe este código de retorno, los demás parámetros devueltos no son válidos.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

CM_CONVERSATION_TYPE_MISMATCH
CM_DEALLOCATED_ABEND
CM_DEALLOCATED_ABEND_SVC (sólo conversación básica)
CM_DEALLOCATED_ABEND_TIMER (sólo conversación básica)
CM_OPERATION_INCOMPLETE (sólo si *receive_type* = CM_RECEIVE_AND_WAIT)
CM_OPERATION_NOT_ACCEPTED
CM_PIP_NOT_SPECIFIED_CORRECTLY
CM_PRODUCT_SPECIFIC_ERROR
CM_PROGRAM_ERROR_NO_TRUNC
CM_PROGRAM_ERROR_PURGING
CM_PROGRAM_ERROR_TRUNC (sólo conversación básica)
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY
CM_SECURITY_NOT_VALID
CM_SYNC_LVL_NOT_SUPPORTED_PGM
CM_SYNC_LVL_NOT_SUPPORTED_LU
CM_TP_NOT_AVAILABLE_NO_RETRY
CM_TP_NOT_AVAILABLE_RETRY
CM_TPN_NOT_RECOGNIZED
CM_SVC_ERROR_NO_TRUNC (sólo conversación básica)
CM_SVC_ERROR_PURGING (sólo conversación básica)
CM_SVC_ERROR_TRUNC (sólo conversación básica)

Estado al emitirse

La conversación puede encontrarse en estado Recibir, Enviar o Enviar-Pendiente.

Si *receive_type* está definido en CM_RECEIVE_IMMEDIATE, la conversación debe encontrarse en estado Recibir.

WINDOWS

Si la aplicación emite satisfactoriamente la llamada Receive en modalidad sin bloqueo, el estado de la conversación cambia dos veces. En el retorno inicial de la llamada, la conversación cambia al estado Pendiente-Anotar. Después de que CPI-C devuelva los resultados del proceso de la llamada, el estado de la conversación es el que se describe a continuación.

Emisión de la llamada en estado Enviar o Enviar-Pendiente

Si se emite la llamada Receive mientras la conversación se encuentra en estado Enviar o Enviar-Pendiente, la LU local envía la información de su almacenamiento intermedio de envío y un indicador de envío al programa asociado. En función de los parámetros *data_received* y *status_received*, la conversación puede cambiar al estado Recibir para el programa local. Para ver más información, consulte "Cambio de estado".

Cambio de estado

El nuevo estado de conversación viene determinado por los factores siguientes:

- El estado en que se encuentra la conversación cuando el programa emite la llamada
- El parámetro *return_code*
- Los parámetros *data_received* y *status_received*

Llamada emitida en estado Recibir

Los cambios de estado que se muestran en la Tabla 18 pueden producirse cuando la llamada Receive se emite con la conversación en estado Recibir y el parámetro *return_code* tiene el valor CM_OK.

Tabla 18. Cambios de estado cuando se emite la llamada Receive en estado Recibir

data_received	*status_received*	**Estado nuevo**
CM_DATA_RECEIVED CM_COMPLETE_DATA_RECEIVED CM_INCOMPLETE_DATA_RECEIVED	CM_NO_STATUS_RECEIVED	Sin cambio
CM_DATA_RECEIVED CM_COMPLETE_DATA_RECEIVED	CM_SEND_RECEIVED	Enviar-Pendiente
CM_NO_DATA_RECEIVED	CM_SEND_RECEIVED	Enviar

Si *return_code* tiene el valor CM_UNSUCCESSFUL, lo que significa que *receive_type* está definido en CM_RECEIVE_IMMEDIATE y no existen datos disponibles, no hay ningún cambio de estado.

Llamada emitida en estado Enviar

Los cambios de estado que se muestran en la Tabla 19 pueden producirse cuando la llamada Receive se emite con la conversación en estado Enviar y el parámetro *return_code* tiene el valor CM_OK.

Tabla 19. Cambios de estado cuando se emite la llamada Receive en estado Enviar

data_received	*status_received*	**Estado nuevo**
CM_DATA_RECEIVED CM_COMPLETE_DATA_RECEIVED CM_INCOMPLETE_DATA_RECEIVED	CM_NO_STATUS_RECEIVED	Recibir
CM_DATA_RECEIVED CM_COMPLETE_DATA_RECEIVED	CM_SEND_RECEIVED	Enviar-Pendiente
CM_NO_DATA_RECEIVED	CM_SEND_RECEIVED	Sin cambio

Llamada emitida en estado Enviar-Pendiente

Los cambios de estado que se muestran en la Tabla 20 pueden producirse cuando la llamada Receive se emite con la conversación en estado Enviar-Pendiente y el parámetro *return_code* tiene el valor CM_OK.

Tabla 20. Cambios de estado cuando se emite la llamada Receive en estado Enviar-Pendiente

data_received	*status_received*	**Estado nuevo**
CM_DATA_RECEIVED CM_COMPLETE_DATA_RECEIVED CM_INCOMPLETE_DATA_RECEIVED	CM_NO_STATUS_RECEIVED	Recibir
CM_DATA_RECEIVED CM_COMPLETE_DATA_RECEIVED	CM_SEND_RECEIVED	Sin cambio
CM_NO_DATA_RECEIVED	CM_SEND_RECEIVED	Enviar

Llamada emitida en cualquier estado permitido

Los apartados siguientes resumen los cambios de estado que pueden producirse cuando se emite la llamada Receive en cualquiera de los estados permitidos.

Proceso de confirmación

Los cambios de estado siguientes se producen en las condiciones siguientes:

- El parámetro *return_code* tiene el valor CM_OK.
- El parámetro *data_received* está definido en CM_DATA_RECEIVED, CM_COMPLETE_DATA_RECEIVED o CM_NO_DATA_RECEIVED.
- El parámetro *status_received* indica un cambio a un estado de confirmación, tal como se muestra en la Tabla 21.

Tabla 21. Cambios de estado cuando se emite la llamada Receive en cualquier estado permitido

status_received	**Estado nuevo**
CM_CONFIRM_DEALLOC_RECEIVED	Confirmar-Desasignar
CM_CONFIRM_SEND_RECEIVED	Confirmar-Enviar
CM_CONFIRM_RECEIVED	Confirmar

Desasignación normal

Si el parámetro *return_code* tiene el valor CM_DEALLOCATED_NORMAL, la conversación cambia al estado Restablecer.

Finalizaciones anormales

Las siguientes condiciones de finalización anormal, indicadas por el parámetro *return_code*, hacen que la conversación cambie al estado Restablecer:

CM_CONVERSATION_TYPE_MISMATCH
CM_PIP_NOT_SPECIFIED_CORRECTLY
CM_SECURITY_NOT_VALID
CM_SYNC_LVL_NOT_SUPPORTED_PGM
CM_SYNC_LVL_NOT_SUPPORTED_LU
CM_TPN_NOT_RECOGNIZED
CM_TP_NOT_AVAILABLE_NO_RETRY
CM_TP_NOT_AVAILABLE_RETRY
CM_DEALLOCATED_ABEND
CM_DEALLOCATED_ABEND_SVC
CM_DEALLOCATED_ABEND_TIMER

CM_SVC_ERROR_TRUNC
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY

Errores

Los cambios de estado que se muestran en la Tabla 22 pueden producirse cuando se encuentra un error de transmisión. (Éste se indica mediante uno de los códigos de retorno siguientes: CM_PROGRAM_ERROR_PURGING, CM_PROGRAM_ERROR_NO_TRUNC, CM_SVC_ERROR_PURGING o CM_SVC_ERROR_NO_TRUNC).

Tabla 22. Cambios de estado ocasionados por un error de transmisión de datos

return_code	**Estado anterior**	**Estado nuevo**
CM_PROGRAM_ERROR_PURGING	Recibir	Sin cambio
CM_PROGRAM_ERROR_NO_TRUNC	Recibir	Sin cambio
CM_SVC_ERROR_PURGING	Enviar	Recibir
CM_SVC_ERROR_NO_TRUNC	Enviar-Pendiente	Recibir

Notas de uso

Los apartados siguientes proporcionan información de uso adicional para la llamada Receive.

Registros truncados

Si el programa asociado trunca un registro lógico, el programa local recibe la notificación del truncamiento mediante el parámetro *return_code* en la siguiente llamada Receive.

Definición del parámetro Requested_Length en cero

Si un programa emite la llamada Receive con el parámetro *requested_length* definido en 0 (cero), la llamada se ejecuta como de costumbre.

Sin embargo, los parámetros *data_received* y *status_received* no se definen en la misma llamada Receive. (El registro nulo enviado por una conversación correlacionada, que se describe en el parámetro siguiente, constituye una excepción a esta situación.)

En una conversación correlacionada en que hay datos disponibles del programa asociado, el parámetro *data_received* tiene el valor CM_INCOMPLETE_DATA_RECEIVED. Si hay un registro nulo disponible (el parámetro *send_length* de la llamada Send_Data emitida por el programa asociado tiene el valor 0), el parámetro *data_received* se define en CM_COMPLETE_DATA_RECEIVED con el parámetro *received_length* definido en 0 (cero).

En una conversación básica en que hay datos disponibles y la característica de relleno definida es CM_FILL_LL, el parámetro *data_received* tiene el valor CM_INCOMPLETE_DATA_RECEIVED. Si la característica de relleno se define en CM_FILL_BUFFER, el parámetro *data_received* tiene el valor CM_DATA_RECEIVED.

Conversión de cadenas

La LU no efectúa automáticamente ninguna conversión entre EBCDIC y ASCII en la cadena de datos recibida antes de colocarla en el almacenamiento intermedio.

Si el programa remoto envía datos en EBCDIC, el programa local puede utilizar la llamada Convert_Incoming para convertir los datos recibidos a ASCII.

WINDOWS

El programa también puede utilizar el verbo CSV CONVERT para convertir los datos recibidos a ASCII. Para más información, consulte la publicación *Communications Server para Linux, Guía del programador para CSV*.

Release_Local_TP_Name (cmrltp)

AIX, LINUX

Un programa emite la llamada Release_Local_TP_Name para indicar que ya no aceptará peticiones Allocate entrantes para un nombre de TP. El nombre de TP puede haberse especificado mediante cualquiera de los métodos descritos en "Especificación del nombre de TP local" en la página 34.

Llamada de función

```
void cmrltp (
             unsigned char CM_PTR      TP_name,
             CM_INT32 CM_PTR           TP_name_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

```
public native void cmrltp (
                             byte[]          TP_name,
                             CPICLength      TP_name_length,
                             CPICReturnCode     return_code
                           );
```

Parámetros suministrados

Los parámetros suministrados son:

TP_name
Este parámetro especifica la dirección inicial del nombre de TP. Debe ser un nombre de TP que el programa haya especificado anteriormente en una llamada Specify_Local_TP_Name.

TP_name_length
Este parámetro especifica la longitud del nombre (1–64 caracteres).

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *TP_name* no es un nombre de TP asociado a este programa.
- El valor especificado por *TP_name_length* está fuera del rango.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

Esta llamada no está asociada a una conversación.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, los nombres asociados a este programa no se modifican.

Si en el momento en que se emite esta llamada hay pendiente una llamada Accept_Incoming, ésta puede aceptar una petición Allocate entrante para el nombre especificado en esta llamada. No obstante, las llamadas Accept_Conversation o Accept_Incoming posteriores no aceptarán peticiones Allocate entrantes para este nombre.

Si un programa libera todos sus nombres de TP, incluido el nombre especificado por la variable de entorno APPCTPN (si existe), no puede emitir ninguna otra llamada Accept_Conversation o Accept_Incoming salvo que primero especifique un nuevo nombre de TP local. Para ver más información, consulte "Especificación del nombre de TP local" en la página 34.

Request_To_Send (cmrts)

La llamada Request_To_Send notifica al programa asociado que el programa local desea enviar datos.

Acción del programa asociado

Como respuesta a esta petición, el programa asociado puede cambiar la conversación al estado Recibir emitiendo una de las llamadas siguientes:

- Receive con *receive_type* definido en CM_RECEIVE_AND_WAIT
- Prepare_To_Receive
- Send_Data con *send_type* definido en CM_SEND_AND_PREP_TO_RECEIVE

El programa asociado también puede hacer caso omiso de la petición de envío.

Cuándo puede enviar datos el programa local

El estado de conversación cambia a Enviar para el programa local cuando éste recibe uno de los valores siguientes mediante el parámetro *status_received* de una llamada Receive posterior:

- CM_SEND_RECEIVED

- CM_CONFIRM_SEND_RECEIVED y responde con una llamada Confirmed

Llamada de función

```
void cmrts (
            unsigned char CM_PTR         conversation_ID,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmrts (
                            byte[]             conversation_ID,
                            CPICReturnCode     return_code
                          );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
La conversación no se encuentra en estado Recibir, Enviar, Enviar-Pendiente, Confirmar, Confirmar-Enviar ni Confirmar-Desasignar.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.
CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR

Estado al emitirse

La conversación puede estar en cualquiera de los estados siguientes: Recibir, Enviar, Enviar-Pendiente, Confirmar, Confirmar-Enviar, Confirmar-Desasignar o Pendiente-Anotar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

El programa asociado recibe la notificación de petición de envío mediante el parámetro *request_to_send_received* de las llamadas siguientes:

- Confirmed
- Receive
- Send_Data
- Send_Error
- Test_Request_to_Send_Received

La notificación de petición de envío se envía al programa asociado de inmediato; CPI-C 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 programa local se encuentra en estado Enviar y emite la llamada Prepare_To_Receive seguida de la llamada Request_To_Send, el programa asociado (en estado Recibir) puede recibir la notificación de petición de envío antes de recibir la notificación de envío. Por ello puede informarse de la notificación de petición de envío a un programa mediante la llamada Receive.

Al recibir una notificación de petición de envío, la LU asociada retiene la notificación hasta que el programa asociado emite una llamada que devuelve el parámetro *request_to_send_received*. La LU retiene únicamente una notificación de petición de envío por conversación, de modo que no se envía una notificación al programa asociado de todas las llamadas Request_To_Send emitidas por el programa local.

Send_Data (cmsend)

La llamada Send_Data coloca datos en el almacenamiento intermedio de envío de la LU local para transmitirlos al programa asociado.

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

- El almacenamiento intermedio de envío se llena.
- El programa local emite una llamada Flush, Confirm o Deallocate u otra llamada que vacía el almacenamiento intermedio de envío de la LU. (Algunos tipos de envío, definidos por la llamada Set_Send_Type, incluyen la función de vaciar.)

Los datos que se enviarán pueden ser:

- Un registro de datos completo en una conversación correlacionada. Un registro de datos completo es una cadena de la longitud especificada por el parámetro *send_length*.
- Un registro lógico completo, o parte de un registro lógico, en una conversación básica. La longitud de un registro lógico completo viene determinada por el valor de LL. (Un registro lógico puede terminar y uno nuevo puede empezar en medio de la cadena de datos que se va a enviar.)

Llamada de función

```
void cmsend (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR                buffer,
             CM_INT32 CM_PTR                     send_length,
             CM_Request_to_Send_Received CM_PTR   request_to_send_received,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsend (
                           byte[]             conversation_ID,
                           byte[]                          buffer,
                           CPICLength                      buffer_length,
                           CPICControlInformationReceived  request_to_send_received,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

buffer Este parámetro especifica la dirección del almacenamiento intermedio que contiene los datos que se colocarán en el almacenamiento intermedio de envío de la LU local.

send_length
: Número de bytes de datos que se colocarán en el almacenamiento intermedio de envío de la LU local.

  El rango de este valor es `0`–`32.767`.

  Para las conversaciones correlacionadas, si *send_length* tiene el valor `0`, se envía un registro de datos nulo al programa asociado.

  En el caso de las conversaciones básicas, si *send_length* tiene el valor `0` (cero), no se envía ningún dato. El parámetro de almacenamiento intermedio no se tiene en cuenta. Sin embargo, los demás parámetros son válidos.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

request_to_send_received
: Indicador de petición de envío recibida. Los valores posibles son:

CM_REQ_TO_SEND_RECEIVED
El programa asociado ha emitido la llamada Request_To_Send, que solicita al programa local que cambie la conversación al estado Recibir.

CM_REQ_TO_SEND_NOT_RECEIVED
El programa asociado no ha emitido la llamada Request_To_Send.

Este valor no es relevante si el parámetro *return_code* está establecido en CM_PROGRAM_PARAMETER_CHECK o CM_PROGRAM_STATE_CHECK.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* no es válido.
- El valor especificado por *send_length* está fuera del rango.
- Ésta es una conversación básica y los 2 primeros bytes del parámetro *buffer* contienen una longitud de registro lógico que no es válida (0x0000, 0x0001, 0x8000 ó 0x8001).

CM_PROGRAM_STATE_CHECK
Se ha producido una de estas situaciones:

- El estado de conversación no es Enviar ni Enviar-Pendiente.
- La conversación básica se encuentra en estado Enviar y el tipo de envío definido es CM_SEND_AND_CONFIRM, CM_SEND_AND_DEALLOCATE o CM_SEND_AND_PREP_TO_RECEIVE. Sin embargo, los datos no terminan en un límite de registro lógico. Únicamente puede emitirse Send_Data en medio de un registro lógico si el tipo de envío definido es CM_SEND_AND_DEALLOCATE y el tipo de desasignación es CM_DEALLOCATE_ABEND.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

CM_CONVERSATION_TYPE_MISMATCH
CM_DEALLOCATED_ABEND
CM_DEALLOCATED_ABEND_SVC
CM_DEALLOCATED_ABEND_TIMER
CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PIP_NOT_SPECIFIED_CORRECTLY
CM_PRODUCT_SPECIFIC_ERROR
CM_PROGRAM_ERROR_PURGING
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY
CM_SECURITY_NOT_VALID
CM_SVC_ERROR_PURGING
CM_SYNC_LVL_NOT_SUPPORTED_PGM
CM_SYNC_LVL_NOT_SUPPORTED_LU
CM_TP_NOT_AVAILABLE_NO_RETRY
CM_TP_NOT_AVAILABLE_RETRY
CM_TPN_NOT_RECOGNIZED

Estado al emitirse

Cuando el programa emite esta llamada la conversación debe encontrarse en estado Enviar o Enviar-Pendiente.

Cambio de estado

Cuando el parámetro *return_code* tiene el valor CM_OK, el nuevo estado de conversación depende del parámetro *send_type*, tal como se muestra en la Tabla 23.

Tabla 23. Cambios de estado para la llamada Send_Data

send_type	**Estado nuevo**
CM_BUFFER_DATA	Enviar
CM_SEND_AND_FLUSH	Enviar
CM_SEND_AND_CONFIRM	Enviar
CM_SEND_AND_PREP_TO_RECEIVE	Recibir
CM_SEND_AND_DEALLOCATE	Restablecer

Si el parámetro *return_code* tiene el valor CM_PROGRAM_ERROR_PURGING o CM_SVC_ERROR_PURGING, el estado de la conversación cambia a Recibir. Para los demás valores de ejecución incorrecta, el estado de la conversación cambia a Restablecer.

Notas de uso

La LU no efectúa automáticamente ninguna conversión entre ASCII y EBCDIC en la cadena de datos que se va a enviar.

Si el programa remoto necesita que se envíen datos en EBCDIC, el programa local puede utilizar la llamada Convert_Outgoing para convertir los datos a EBCDIC antes de enviarlos.

WINDOWS

El programa también puede utilizar el verbo CSV CONVERT para convertir los datos a EBCDIC antes de enviarlos. Para más información, consulte la publicación *Communications Server para Linux, Guía del programador para CSV*.

Send_Error (cmserr)

La llamada Send_Error notifica al programa asociado que el programa local ha encontrado un error a nivel de aplicación. El programa local puede utilizar la llamada Send_Error para fines tales como informar al programa asociado de un error encontrado en los datos recibidos, rechazar una petición de confirmación o truncar un registro lógico incompleto que está enviando.

La llamada Send_Error vacía el almacenamiento intermedio de envío de la LU local y envía al programa asociado el contenido del almacenamiento intermedio de envío seguido de la notificación de error.

La notificación de error se envía al programa asociado como uno de los siguientes valores de *return_code*:

- CM_PROGRAM_ERROR_TRUNC
- CM_PROGRAM_ERROR_NO_TRUNC
- CM_PROGRAM_ERROR_PURGING

Una vez ejecutada correctamente esta llamada, la conversación se encuentra en estado Enviar para el programa local y en estado Recibir para el programa asociado.

Llamada de función

```
void cmserr (
             unsigned char CM_PTR         conversation_ID,
             CM_Request_to_Send_Received CM_PTR   request_to_send_received,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmserr (
                            byte[]             conversation_ID,
                            CPICControlInformationReceived  request_to_send_received,
                            CPICReturnCode     return_code
                           );
```

Parámetros suministrados

El parámetro suministrado es:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation ha devuelto el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

request_to_send_received
Indicador de petición de envío recibida. Los valores posibles son:

CM_REQ_TO_SEND_RECEIVED
El programa asociado ha emitido la llamada Request_To_Send, que solicita al programa local que cambie la conversación al estado Recibir.

CM_REQ_TO_SEND_NOT_RECEIVED
El programa asociado no ha emitido la llamada Request_To_Send.

Este valor no es relevante si el parámetro *return_code* está establecido en CM_PROGRAM_PARAMETER_CHECK o CM_STATE_CHECK.

return_code

Los códigos de retorno posibles varían en función del estado de conversación en el momento de emitirse la llamada. **Estado Enviar**

Si el programa emite la llamada con la conversación en el estado Enviar, pueden devolverse los códigos de retorno siguientes:

CM_OK La llamada se ha ejecutado correctamente.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

```
CM_CONVERSATION_TYPE_MISMATCH
CM_DEALLOCATED_ABEND
CM_DEALLOCATED_ABEND_SVC
CM_DEALLOCATED_ABEND_TIMER
CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PIP_NOT_SPECIFIED_CORRECTLY
CM_PRODUCT_SPECIFIC_ERROR
CM_PROGRAM_ERROR_PURGING
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY
CM_SECURITY_NOT_VALID
CM_SVC_ERROR_PURGING
CM_SYNC_LVL_NOT_SUPPORTED_PGM
CM_SYNC_LVL_NOT_SUPPORTED_LU
CM_TP_NOT_AVAILABLE_NO_RETRY
CM_TP_NOT_AVAILABLE_RETRY
CM_TPN_NOT_RECOGNIZED
```

Estado Recibir o estado Pendiente-Anotar

Si se emite la llamada en estado Recibir o el estado Pendiente-Anotar, son posibles los siguientes códigos de retorno:

CM_OK Dado que la información entrante se elimina cuando se emite la llamada Send_Error en estado Recibir o Pendiente-Anotar, se genera CM_OK en lugar de los códigos siguientes:

```
CM_PROGRAM_ERROR_NO_TRUNC
CM_PROGRAM_ERROR_PURGING
CM_SVC_ERROR_NO_TRUNC
CM_SVC_ERROR_PURGING
CM_PROGRAM_ERROR_TRUNC
CM_SVC_ERROR_TRUNC
```

Códigos de retorno comunes

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

```
CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY
```

CM_DEALLOCATED_NORMAL

Dado que la información entrante se elimina cuando se emite la

llamada Send_Error en estado Recibir o el estado Pendiente-Anotar, se genera CM_DEALLOCATED_NORMAL en lugar de los códigos siguientes:

CM_CONVERSATION_TYPE_MISMATCH
CM_DEALLOCATED_ABEND
CM_DEALLOCATED_ABEND_SVC
CM_DEALLOCATED_ABEND_TIMER
CM_PIP_NOT_SPECIFIED_CORRECTLY
CM_SECURITY_NOT_VALID
CM_SYNC_LVL_NOT_SUPPORTED_PGM
CM_SYNC_LVL_NOT_SUPPORTED_LU
CM_TPN_NOT_RECOGNIZED
CM_TP_NOT_AVAILABLE_NO_RETRY
CM_TP_NOT_AVAILABLE_RETRY

Estado Enviar-Pendiente

Si la llamada se emite en estado Enviar-Pendiente, pueden devolverse los códigos de retorno siguientes:

CM_OK La llamada se ha ejecutado correctamente.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_DEALLOCATED_ABEND
CM_DEALLOCATED_ABEND_SVC
CM_DEALLOCATED_ABEND_TIMER
CM_PRODUCT_SPECIFIC_ERROR
CM_PROGRAM_ERROR_PURGING
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY
CM_SVC_ERROR_PURGING

Estado Confirmar, Confirmar-Enviar o Confirmar-Desasignar

Si la llamada se emite en estado Confirmar, Confirmar-Enviar o Confirmar-Desasignar, pueden devolverse los códigos de retorno siguientes:

CM_OK La llamada se ha ejecutado correctamente.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

CM_OPERATION_INCOMPLETE
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR
CM_RESOURCE_FAILURE_NO_RETRY
CM_RESOURCE_FAILURE_RETRY

Otros estados

No está permitido emitir la llamada Send_Error con la conversación en estado Restablecer, Inicializar o Inicializar-Entrante. Pueden devolverse los códigos de retorno siguientes:

CM_OPERATION_NOT_ACCEPTED
Consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
El estado de conversación no es Enviar, Recibir, Confirmar, Confirmar-Enviar, Confirmar-Desasignar ni Enviar-Pendiente.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Inicializar, Inicializar-Entrante o Restablecer.

Cambio de estado

El estado nuevo viene determinado por el parámetro *return_code*. Los cambios de estado posibles están resumidos en la Tabla 24.

Tabla 24. Cambios de estado para la llamada Send_Error

return_code	**Estado nuevo**
CM_OK	Enviar
CM_CONVERSATION_TYPE_MISMATCH CM_PIP_NOT_SPECIFIED_CORRECTLY CM_SECURITY_NOT_VALID CM_SYNC_LEVEL_NOT_SUPPORTED_PGM CM_SYNC_LEVEL_NOT_SUPPORTED_LU CM_TPN_NOT_RECOGNIZED CM_TP_NOT_AVAILABLE_NO_RETRY CM_TP_NOT_AVAILABLE_RETRY	Restablecer
CM_RESOURCE_FAILURE_RETRY CM_RESOURCE_FAILURE_NO_RETRY	Restablecer
CM_DEALLOCATED_ABEND CM_DEALLOCATED_ABEND_SVC CM_DEALLOCATED_ABEND_TIMER	Restablecer
CM_DEALLOCATED_NORMAL	Restablecer
CM_PROGRAM_ERROR_PURGING CM_SVC_ERROR_PURGING	Recibir
Todos los demás	Sin cambio

Notas de uso

Los apartados siguientes proporcionan información de uso adicional para la llamada Send_Error.

Envío de datos de anotaciones

En las conversaciones básicas, el programa local puede utilizar la llamada Set_Log_Data para especificar datos de anotaciones de error que se enviarán a la LU asociada. Si la característica de longitud de los datos de anotaciones de la conversación básica es mayor que 0 (cero), la LU da formato a los datos y los almacena en el almacenamiento intermedio de envío.

Una vez finalizada la llamada Send_Error, la longitud de los datos de anotaciones se establece en 0 (cero) y los datos de anotaciones se establecen en nulos.

Datos innecesarios eliminados

Si la conversación está en estado Recibir o Pendiente-Anotar cuando el programa emite la llamada Send_Error, los datos entrantes los elimina CPI-C. Estos datos incluyen lo siguiente:

- Datos enviados por la llamada Send_Data
- Peticiones de confirmación
- Peticiones de desasignación si el tipo de desasignación de la conversación está definido en CM_DEALLOCATE_CONFIRM o CM_DEALLOCATE_SYNC_LEVEL con el nivel de sincronización definido en CM_CONFIRM

CPI-C no elimina los indicadores de petición de envío entrantes innecesarios.

Estado Enviar-Pendiente

Si la conversación se encuentra en estado Enviar-Pendiente, el programa local puede emitir la llamada Set_Error_Direction para especificar si el error del que se informa se ha producido debido a los datos recibidos o al proceso del programa local una vez recibidos correctamente los datos.

Set_Conversation_Context (cmsctx)

AIX, LINUX

La llamada Set_Conversation_Context define el contexto actual del programa en un valor devuelto anteriormente en una llamada Extract_Conversation_Context. De esta forma el programa puede iniciar una nueva conversación utilizando el mismo contexto que una conversación anterior.

Para ver más información sobre los contextos de conversación, consulte "Conversaciones múltiples" en la página 13.

Llamada de función

```
void cmsctx (
             unsigned char CM_PTR      context_ID,
             CM_INT32 CM_PTR           context_ID_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

```
public native void cmsctx (
                            byte[]          context_ID,
                            CPICLength      context_ID_length,
                            CPICReturnCode     return_code
                           );
```

Nota: Esta llamada no forma parte de la especificación CPI-C para Java y no está soportada en otras implementaciones de CPI-C para Java.

Parámetros suministrados

Los parámetros suministrados son:

context_ID
Este parámetro especifica el contexto necesario.

context_ID_length
Este parámetro especifica la longitud de *context_ID* (1–32 bytes).

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Este código de retorno indica uno de los casos siguientes:

- El valor especificado por *context_ID* no es el contexto de ninguna de las conversaciones actuales del programa ni de sus conversaciones más recientes.
- El valor especificado por *context_ID_length* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Una aplicación utiliza Set_Conversation_Context en las situaciones siguientes:

- Cuando participa en conversaciones múltiples y desea asignar una nueva conversación utilizando el mismo contexto que una conversación ya existente.
- Cuando una llamada CPI-C que asigna un nuevo contexto finaliza en la modalidad sin bloqueo. Por ejemplo, si Accept_Incoming finaliza de inmediato con *return_code* con el valor CM_OK, el contexto actual del programa se establece en el contexto de la nueva conversación; sin embargo, si Accept_Incoming devuelve CM_OPERATION_INCOMPLETE, una Wait_For_Conversation posterior que devuelva el resultado de Accept_Incoming no cambiará el contexto actual del programa. El programa debe utilizar Extract_Conversation_Context y Set_Conversation_Context para establecer el contexto actual en el valor correcto.

Set_Conversation_Security_Password (cmscsp)

El programa que invoca emite la llamada Set_Conversation_Security_Password para especificar la contraseña necesaria con el fin de acceder al programa invocado. Esta llamada únicamente afecta a la conversación si el tipo de seguridad de es CM_SECURITY_PROGRAM o CM_SECURITY_PROGRAM_STRONG (sistemas AIX o Linux) o XC_SECURITY_PROGRAM (sistemas Windows). Altera temporalmente la contraseña

inicial de la información complementaria especificada por la llamada Initialize_Conversation. Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate.

Llamada de función

```
void cmscsp (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR      security_password,
             CM_INT32 CM_PTR           security_password_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmscsp (
                            byte[]             conversation_ID,
                            byte[]          security_password,
                            CPICLength      security_password_length,
                            CPICReturnCode     return_code
                           );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

security_password
Especifica la contraseña necesaria para acceder al programa asociado. Este parámetro es una cadena de caracteres de 1 a 10 caracteres (sistemas AIX or Linux) o bien de 1 a 8 caracteres (sistemas Windows) y distingue entre mayúsculas y minúsculas. Debe coincidir con la contraseña del identificador de usuario configurado para el programa asociado.

Están permitidos los caracteres siguientes:

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

security_password_length
Especifica la longitud de *security_password*.

El rango de este valor es de 1 a 10 caracteres (sistemas AIX o Linux), o bien 1 a 8 caracteres (sistemas Windows). Si *security_password_length* está definido en `0` (cero), el parámetro *security_password* no se tiene en cuenta; esto equivale a establecer *security_password* en una cadena nula.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:
- El valor especificado por *conversation_ID* no es válido.
- El valor especificado por *security_password_length* está fuera del rango.

CM_PROGRAM_STATE_CHECK
Se ha producido una de estas situaciones:
- La conversación no se encuentra en estado Inicializar.
- El tipo de seguridad de la conversación no está definido en CM_SECURITY_PROGRAM ni CM_SECURITY_PROGRAM_STRONG.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Además de la contraseña se necesita un identificador de usuario. Se puede obtener de la entrada de información complementaria especificada en la anterior llamada Initialize_Conversation o el programa puede especificarlo mediante Set_Conversation_Security_User_ID.

Una contraseña que no es válida no se detecta hasta que se envía la petición de asignación, generada por la llamada Allocate, a la LU asociada. El error se devuelve al programa que invoca en una llamada posterior.

Si el código de retorno no es CM_OK, las características de conversación *security_password* y *security_password_length* no se modifican.

Set_Conversation_Security_Password (xcscsp)

Esta función no está disponible en CPI-C para Java.

El programa que invoca emite esta llamada para especificar la contraseña necesaria con el fin de acceder al programa invocado.

La llamada xcscsp proporciona compatibilidad para aplicaciones que utilizan la definición CPI-C de X/Open. Se ha incorporado en CPI-C 2.0 de IBM como la llamada Set_Conversation_Security_Password (cmscsp). Utilice cmscsp siempre que sea posible para lograr una mayor portabilidad del programa a otras plataformas.

Los parámetros de esta llamada son los mismos que los de la llamada cmscsp. Para ver más información sobre cmscsp, consulte "Set_Conversation_Security_Password (cmscsp)" en la página 118.

Set_Conversation_Security_Type (cmscst)

El programa que invoca emite la llamada Set_Conversation_Security_Type para especificar la información que necesita la LU asociada a fin de validar el acceso al programa invocado. Esta llamada altera temporalmente el tipo de seguridad inicial de la información complementaria especificada por la llamada Initialize_Conversation. Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate.

Llamada de función

```
void cmscst (
             unsigned char CM_PTR         conversation_ID,
             XC_CONVERSATION_SECURITY_TYPE CM_PTR  conversation_security_type,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmscst (
                           byte[]             conversation_ID,
                           CPICConversationSecurityType  conversation_security_type,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

conversation_security_type
Especifica la información que necesita la LU asociada a fin de validar el acceso al programa invocado. En función de la seguridad de conversación establecida para el programa invocado durante la configuración, utilice uno de los valores siguientes:

AIX, LINUX

CM_SECURITY_NONE
El programa invocado no utiliza ninguna seguridad de conversación.

CM_SECURITY_SAME
El programa invocado utiliza la seguridad de conversación y está configurado para aceptar un indicador de seguridad ya verificada (como se describe en "Visión general de la seguridad de conversación" en la página 13). El identificador de usuario del contexto actual del programa local (en el momento de emitirse la llamada Allocate) se enviará al programa invocado, junto con un indicador de seguridad ya verificado. Este indicador ordena al programa invocado que no solicite la contraseña.

CM_SECURITY_PROGRAM
El programa invocado utiliza la seguridad de conversación y por consiguiente requiere un identificador de usuario y una contraseña. La información de seguridad se obtendrá de las características de la conversación actual (en el momento de emitirse la llamada Allocate).

CM_SECURITY_PROGRAM_STRONG
Igual que para CM_SECURITY_PROGRAM, con la excepción de que el nodo local no debe enviar la contraseña a través de la red en formato de texto no cifrado. Este valor sólo se puede utilizar si el sistema remoto proporciona soporte para la sustitución de contraseñas.

WINDOWS

XC_SECURITY_NONE
Equivalente a CM_SECURITY_NONE

XC_SECURITY_SAME
Equivalente a CM_SECURITY_SAME

XC_SECURITY_PROGRAM
Equivalente a CM_SECURITY_PROGRAM

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_STATE_CHECK
La conversación no se encuentra en estado Inicializar.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* o *conversation_security_type* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, el parámetro *conversation_security_type* no se modifica.

Set_Conversation_Security_Type (xcscst)

Esta función no está disponible en CPI-C para Java.

Esta llamada se emite mediante el programa que invoca para especificar la información que la LU asociada necesita a fin de validar el acceso al programa invocado. Esta llamada altera temporalmente el tipo de seguridad inicial de la información complementaria especificada por la llamada Initialize_Conversation.

La llamada proporciona compatibilidad para aplicaciones que utilizan la definición de X/Open CPI-C. Se ha incorporado en CPI-C 2.0 de IBM como la llamada Set_Conversation_Security_Type (cmscst). Utilice cmscst siempre que sea posible para lograr una mayor portabilidad del programa a otras plataformas.

Los parámetros de esta llamada son los mismos que los de la llamada cmscst. Para ver más información sobre cmscst, consulte "Set_Conversation_Security_Type (cmscst)" en la página 121.

Set_Conversation_Security_User_ID (cmscsu)

El programa que invoca emite la llamada Set_Conversation_Security_User_ID para especificar el identificador de usuario necesario para acceder al programa invocado. Altera temporalmente el identificador de usuario inicial de la información complementaria especificada por la llamada Initialize_Conversation.

Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate. Esta llamada no es válida si el tipo de seguridad de conversación es CM_SECURITY_NONE (sistemas AIX o Linux) o XC_SECURITY_NONE (sistemas Windows).

Llamada de función

```
void cmscsu (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR      security_user_ID,
             CM_INT32 CM_PTR           security_user_ID_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmscsu (
                            byte[]             conversation_ID,
                            byte[]          security_user_ID
                            CPICLength      security_user_ID_length,
                            CPICReturnCode     return_code
                           );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

security_user_ID
Especifica el identificador de usuario necesario para acceder al programa asociado. Este parámetro es una cadena de caracteres de 1 a 10 caracteres (sistemas AIX or Linux) o bien de 1 a 8 caracteres (sistemas Windows) y distingue entre mayúsculas y minúsculas.

Están permitidos los caracteres siguientes:

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

security_user_ID_length
Especifica la longitud de *security_user_ID*. El rango para este valor es de 1 a 10 caracteres (sistemas AIX o Linux), o bien 1 a 8 caracteres (sistemas Windows). Si la longitud es `0` (cero), el parámetro *security_user_ID* no se tiene en cuenta; esto equivale a establecer *security_user_ID* en una cadena nula.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* no es válido.
- El valor especificado por *security_user_ID_length* está fuera del rango.

CM_PROGRAM_STATE_CHECK
Se ha producido una de estas situaciones:

- La conversación no se encuentra en estado Inicializar.
- El tipo de seguridad de la conversación se define en `CM_SECURITY_NONE`.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el código de retorno no es `CM_OK`, las características de conversación *security_user_ID* y *security_user_ID_length* no se modifican.

Un identificador de usuario que no es válido no se detecta hasta que se envía la petición de asignación, generada por la llamada Allocate, a la LU asociada. El error se devuelve al programa que invoca en una llamada posterior.

Set_Conversation_Security_User_ID (xcscsu)

Esta función no está disponible en CPI-C para Java.

El programa que invoca emite esta llamada para especificar el identificador de usuario necesario para acceder al programa invocado.

La llamada xcscsu proporciona compatibilidad para aplicaciones que utilizan la definición CPI-C de X/Open. Se ha incorporado en CPI-C 2.0 de IBM como la llamada Set_Conversation_Security_User_ID (cmscsu). Utilice cmscsu siempre que sea posible para lograr una mayor portabilidad del programa a otras plataformas.

Los parámetros de esta llamada son los mismos que los de la llamada cmscsu. Para ver más información sobre cmscsu, consulte "Set_Conversation_Security_User_ID (cmscsu)" en la página 123.

Set_Conversation_Type (cmsct)

El programa que invoca emite la llamada Set_Conversation_Type para definir una conversación como correlacionada o básica. Esta llamada altera temporalmente el tipo de conversación por omisión establecido por la llamada Initialize_Conversation. El tipo de conversación por omisión es CM_MAPPED_CONVERSATION. Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate.

Llamada de función

```
void cmsct (
            unsigned char CM_PTR         conversation_ID,
            CM_CONVERSATION_TYPE CM_PTR   conversation_type,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsct (
                           byte[]             conversation_ID,
                           CPICConversationType  conversation_type,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

conversation_type
: Este parámetro especifica el tipo de conversación que asignará la llamada Allocate. Los valores posibles son:
  CM_BASIC_CONVERSATION
  CM_MAPPED_CONVERSATION

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code

Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_STATE_CHECK

La conversación no se encuentra en estado Inicializar.

CM_PROGRAM_PARAMETER_CHECK

Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* o *conversation_type* no es válido.
- El parámetro *conversation_type* especifica una conversación correlacionada, pero la característica de relleno está definida en CM_FILL_BUFFER, que es incompatible con las conversaciones correlacionadas. Antes de cambiar el tipo de conversación a correlacionada, debe emitir la llamada Set_Fill para cambiar el tipo de relleno a CM_FILL_LL.
- El parámetro *conversation_type* especifica una conversación correlacionada. Sin embargo, todavía está vigente una llamada Set_Log_Data anterior, que sólo está permitida en las conversaciones básicas.

CM_PRODUCT_SPECIFIC_ERROR

Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el código de retorno no es CM_OK, la característica de la conversación *conversation_type* no se modifica.

Set_CPIC_Side_Information (xcmssi)

Esta función no está disponible en CPI-C para Java.

La llamada Set_CPIC_Side_Information especifica una entrada de información complementaria para que la utilice esta aplicación. Una entrada de información complementaria de CPI-C asocia un conjunto de características de conversación con un nombre de destino simbólico.

Las entradas de información complementaria están definidas en el archivo de configuración de Communications Server para Linux. Esta llamada especifica una entrada adicional para que la utilice esta aplicación o altera temporalmente la definición del archivo de configuración (o la definición local de la aplicación) si el nombre de destino simbólico especificado ya existe.

Esta llamada se proporciona para asegurar la compatibilidad con CPI-C para X/Open con la especificación CPI-C para Windows; no se incluye en CPI-C 2.0 para IBM.

Llamada de función

```
     void xcmssi (
                  unsigned char CM_PTR      key,
                  SIDE_INFO CM_PTR          side_info_entry,
                  CM_INT32 CM_PTR           side_info_entry_length,
                  CM_RETURN_CODE CM_PTR        return_code
                 );
typedef struct side_info_entry
{
  unsigned char     sym_dest_name[8];      /* Nombre de destino simbólico    */
  unsigned char     partner_LU_name[17];   /* Nombre LU asociada compl. cal. */
  unsigned char     reserved[3];           /* Reservado                      */
  XC_TP_NAME_TYPE   TP_name_type;          /* Tipo de nombre de TP           */
  unsigned char     TP_name[64];           /* Nombre de TP                   */
  unsigned char     mode_name[8];          /* Nombre de modalidad            */
  XC_CONVERSATION_SECURITY_TYPE
                    conversation_security_type; /* Tipo seguridad conversac. */
  unsigned char     security_user_ID[8];   /* Identificador de usuario       */
  unsigned char     security_password[8];  /* Contraseña                     */
} SIDE_INFO;
```

Parámetros suministrados

Los parámetros suministrados son:

key Este parámetro no se tiene en cuenta.

side_info_entry
Este parámetro especifica el contenido de una entrada de información complementaria, tal como se indica a continuación. Cada uno de los campos de la estructura debe estar justificado a la izquierda. Si es necesario, rellene los campos con espacios por la derecha.

side_info_entry.sym_dest_name
Nombre de destino simbólico que identifica la entrada de información complementaria. El parámetro *sym_dest_name* es una cadena de caracteres ASCII de 8 bytes y puede contener cualquier carácter visualizable.

side_info_entry.partner_LU_name
Nombre de la LU asociada completamente calificado. Este nombre consta de dos cadenas de caracteres concatenadas por un punto. Cada uno de los nombres puede tener una longitud máxima de 8 bytes sin espacios intercalados; los caracteres válidos son los caracteres `A–Z` mayúsculas y los números `0–9`.

side_info_entry.TP_name_type
Tipo de TP de destino (los caracteres válidos para un nombre de TP vienen determinados por el tipo de TP). Los valores permitidos son los siguientes:

XC_APPLICATION_TP
TP de aplicación. Todos los caracteres del nombre de TP deben ser caracteres ASCII válidos.

XC_SNA_SERVICE_TP
TP de servicio. El nombre de TP se debe especificar como una cadena ASCII de 8 caracteres que represente los dígitos hexadecimales de un nombre de 4 caracteres. Por ejemplo, si la

representación hexadecimal del nombre es 0x21F0F0F8, defina el parámetro *tp_name* en la cadena de 8 caracteres "21F0F0F8".

El primer carácter (representado por dos bytes) debe ser un valor hexadecimal en el rango `0x0–0x3F`, salvo `0x0E` y `0x0F`; los caracteres restantes (cada uno representado por dos bytes) deben ser caracteres EBCDIC válidos.

side_info_entry.TP_name
Nombre de TP del TP de destino.

Set_CPIC_Side_Information es la única llamada CPI-C que permite especificar un TP de servicio SNA como programa asociado. Consulte la descripción del parámetro *TP_name_type* que aparece más arriba para obtener más información sobre cómo especificar el nombre de TP.

side_info_entry.mode_name
Nombre de la modalidad que se utiliza para acceder al TP de destino.

Para una conversación correlacionada, el nombre de modalidad SNASVCMG está reservado para uso interno de SNA; la llamada Allocate fallará si utiliza este nombre. Se recomienda no utilizar SNASVCMG en una conversación básica, ni CPSVCMG (otro nombre reservado de SNA) en ninguno de los dos tipos de conversación.

side_info_entry.conversation_security_type
Especifica si el TP de destino utiliza la seguridad de conversación. Los valores permitidos son los siguientes:

AIX, LINUX

CM_SECURITY_NONE
El TP de destino no utiliza la seguridad de conversación.

CM_SECURITY_PROGRAM
El TP de destino utiliza la seguridad de conversación. Los parámetros *security_user_ID* y *security_password* especificados más abajo se utilizarán para acceder al TP de destino.

CM_SECURITY_SAME
El TP de destino utiliza la seguridad de conversación y puede aceptar un indicador de seguridad "ya verificada" del TP local. (Esto indica que el propio TP local ha sido invocado por otro TP y ha verificado el identificador de usuario y la contraseña de seguridad proporcionados por este TP.) El parámetro *security_user_ID* especificado más abajo se utilizará para acceder al TP de destino; no se necesita ninguna contraseña.

CM_SECURITY_PROGRAM_STRONG
Igual que para `CM_SECURITY_PROGRAM`, con la excepción de que el nodo local no debe enviar la contraseña a través de la red en formato de texto no cifrado. Este valor sólo se puede utilizar si el sistema remoto proporciona soporte para la sustitución de contraseñas.

WINDOWS

XC_SECURITY_NONE
Equivalente a `CM_SECURITY_NONE`

XC_SECURITY_SAME
Equivalente a CM_SECURITY_SAME

XC_SECURITY_PROGRAM
Equivalente a CM_SECURITY_PROGRAM

side_info_entry.security_user_ID
Identificador de usuario que se utiliza para acceder al TP asociado. Este parámetro no es necesario si el parámetro *conversation_security_type* está establecido en CM_SECURITY_NONE.

side_info_entry.security_password
Contraseña que se utiliza para acceder al TP asociado. Este parámetro es necesario sólo si el parámetro *conversation_security_type* está definido en CM_SECURITY_PROGRAM o CM_SECURITY_PROGRAM_STRONG.

AIX, LINUX

Para asegurar la compatibilidad con la implementación de CPI-C de X/Open, este verbo sólo admite 8 caracteres para el identificador de usuario y la contraseña, aunque los identificadores de usuario de seguridad pueden tener hasta 10 caracteres. Si el TP asociado requiere un identificador de usuario o una contraseña de 9 ó 10 caracteres, debe especificar explícitamente esta información mediante la llamada Set_Conversation_Security_User_ID o Set_Conversation_Security_Password.

side_info_entry_length
Este valor siempre debe estar definido en sizeof(SIDE_INFO).

WINDOWS

side_info_entry_length
Este valor siempre debe establecerse en 124.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- Un valor especificado en la estructura *side_info_entry* no es válido.
- El primer carácter de *side_info_entry* contiene un espacio.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Esta llamada no modifica la información complementaria del archivo de configuración; el cambio sólo se aplica a esta aplicación. Communications Server para Linux almacena la información modificada en la memoria asociada a este proceso del sistema operativo; el cambio se elimina cuando el proceso finaliza (o cuando la aplicación emite la llamada Delete_CPIC_Side_Information para eliminar la entrada). Para ver información más detallada al respecto, consulte "Información complementaria" en la página 31.

Si el parámetro *return_code* no tiene el valor CM_OK, la información complementaria no se modifica.

Los parámetros string de la información complementaria que no son válidos (por ejemplo, los que especifican una LU asociada inexistente) no se detectan hasta que se emite la llamada Allocate. El error se devuelve en una llamada a continuación de Allocate.

Set_Deallocate_Type (cmsdt)

La llamada Set_Deallocate_Type especifica cómo debe desasignarse la conversación. Esta llamada altera temporalmente el tipo de desasignación por omisión establecido por la llamada Initialize_Conversation o Accept_Conversation. El tipo de desasignación por omisión es CM_DEALLOCATE_SYNC_LEVEL.

Las instrucciones de desasignación especificadas por esta llamada entran en vigor cuando se emite la llamada Deallocate o cuando el tipo de envío establecido es CM_SEND_AND_DEALLOCATE y se emite la llamada Send_Data.

Llamada de función

```
void cmsdt (
            unsigned char CM_PTR         conversation_ID,
            CM_DEALLOCATE_TYPE CM_PTR     deallocate_type,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsdt (
                           byte[]             conversation_ID,
                           CPICDeallocateType  deallocate_type,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

deallocate_type
Este parámetro especifica cómo realizar la desasignación. Los valores posibles son:

CM_DEALLOCATE_ABEND
La conversación debe desasignarse de forma anormal e incondicional. Un programa debe especificar CM_DEALLOCATE_ABEND cuando encuentra un error que impide la correcta finalización de una transacción.

Si la conversación se encuentra en estado Enviar, CPI-C envía el contenido del almacenamiento intermedio de envío de la LU local al programa asociado antes de desasignar la conversación. Si la conversación se encuentra en estado Recibir, es posible que se eliminen los datos entrantes innecesarios. En el caso de una conversación básica en estado Enviar, puede producirse el truncamiento del registro lógico.

CM_DEALLOCATE_CONFIRM
Este valor envía al programa asociado el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación de la desasignación. La aplicación no puede utilizar este valor si el nivel de sincronización de la conversación es CM_NONE.

Esta petición de confirmación de la desasignación se envía mediante la llamada Deallocate o mediante la llamada Send_Data con el tipo de envío definido en CM_SEND_AND_DEALLOCATE. La conversación se desasigna normalmente cuando el programa asociado emite la llamada Confirmed, como respuesta a la petición de confirmación.

CM_DEALLOCATE_FLUSH
Este valor envía el contenido del almacenamiento intermedio de envío de la LU local al programa asociado antes de desasignar la conversación normalmente.

CM_DEALLOCATE_SYNC_LEVEL
Este valor utiliza el nivel de sincronización de la conversación para determinar cómo desasignar la conversación. La llamada Initialize_Conversation establece un nivel de sincronización por omisión que se puede alterar temporalmente mediante la llamada Set_Sync_Level.

Si el nivel de sincronización de la conversación está definido en el valor por omisión, CM_NONE, el contenido del almacenamiento intermedio de envío de la LU local se envía al programa asociado y la conversación se desasigna normalmente.

Si el nivel de sincronización de la conversación es CM_CONFIRM, se envía al programa asociado el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación de la desasignación. Esta petición de confirmación de la desasignación se envía mediante la llamada Deallocate o mediante la llamada Send_Data con el tipo de envío definido en CM_SEND_AND_DEALLOCATE. La conversación se desasigna normalmente cuando el programa asociado emite la llamada Confirmed, como respuesta a la petición de confirmación.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* o *deallocate_type* no es válido.
- El parámetro *deallocate_type* especifica CM_DEALLOCATE_CONFIRM, pero el nivel de sincronización de la conversación está definido en CM_NONE

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *deallocate_type* no se modifica.

Puede definir el parámetro *deallocate_type* en CM_FLUSH si el nivel de sincronización de la conversación está definido en CM_NONE o CM_CONFIRM.

El valor CM_DEALLOCATE_FLUSH equivale a CM_DEALLOCATE_SYNC_LEVEL con el nivel de sincronización de la conversación definido en CM_NONE.

El valor CM_DEALLOCATE_CONFIRM equivale a CM_DEALLOCATE_SYNC_LEVEL con el nivel de sincronización de la conversación definido en CM_CONFIRM.

Set_Error_Direction (cmsed)

La llamada Set_Error_Direction especifica si un programa ha detectado un error al recibir datos o prepararse para enviar datos. Esta llamada altera temporalmente la dirección de error por omisión establecido por la llamada Initialize_Conversation o Accept_Conversation. La dirección de error por omisión es CM_RECEIVE_ERROR.

La dirección de error únicamente es relevante cuando un programa emite la llamada Send_Error en estado Enviar-Pendiente inmediatamente después de emitir la llamada Receive y recibir datos (*data_received* es un valor distinto de CM_NO_DATA_RECEIVED) y un indicador de envío (*status_received* = CM_SEND_RECEIVED).

Llamada de función

```
void cmsed (
            unsigned char CM_PTR         conversation_ID,
            CM_ERROR_DIRECTION CM_PTR     error_direction,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsed (
                           byte[]             conversation_ID,
                           CPICErrorDirection  error_direction,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

error_direction
: Este parámetro especifica la dirección en la que fluían los datos cuando el programa ha encontrado un error. Los valores posibles son:

  CM_RECEIVE_ERROR
  : Se ha producido un error en los datos recibidos del programa asociado.

  CM_SEND_ERROR
  : Se ha producido un error mientras el programa local se preparaba para enviar datos al programa asociado.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* o *error_direction* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *error_direction* no se modifica.

Cuando la conversación se encuentra en estado Enviar-Pendiente, el programa emite la llamada Send_Error si detecta errores en los datos recibidos o si se ha producido un error mientras el programa local se preparaba para enviar datos. El programa debe proporcionar la información de dirección de error mediante la llamada Set_Error_Direction antes de emitir la llamada Send_Error ya que la LU no puede saber qué tipo de error se ha producido (de recepción o de envío). La nueva dirección de error permanece en vigor hasta que la modifica una llamada Set_Error_Direction posterior.

Cuando se emite la llamada Send_Error, el programa asociado recibe uno de los códigos de retorno siguientes:

- CM_PROGRAM_ERROR_PURGING si *error_direction* está definido en CM_RECEIVE_ERROR
- CM_PROGRAM_ERROR_NO_TRUNC si *error_direction* está definido en CM_SEND_ERROR

Set_Fill (cmsf)

La llamada Set_Fill especifica si los programas recibirán los datos en forma de registros lógicos o como una longitud de datos especificada. Esta llamada sólo está permitida en las conversaciones básicas. Altera temporalmente el relleno por omisión establecido por la llamada Initialize_Conversation o Accept_Conversation. El relleno por omisión es CM_FILL_LL.

El valor de fill afecta a todas las llamadas Receive posteriores. Puede cambiarse volviendo a emitir la llamada Set_Fill.

Llamada de función

```
void cmsf  (
           unsigned char CM_PTR         conversation_ID,
           CM_FILL CM_PTR             fill,
           CM_RETURN_CODE CM_PTR        return_code
          );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsf (
                           byte[]             conversation_ID,
                           CPICFill        fill,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

fill Este parámetro especifica la forma en que los programas recibirán los datos. Los valores posibles son:

CM_FILL_BUFFER
El programa local recibe datos hasta que se alcanza el número de bytes especificado por el parámetro *requested_length* de la llamada Receive o hasta el final de los datos. Los datos se reciben sin tener en cuenta el formato de registro lógico.

CM_FILL_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 igual al parámetro *requested_length* de la llamada Receive
- El final de un registro lógico

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por el parámetro *conversation_ID* o *fill* no es válido.
- La conversación actual está correlacionada. El parámetro *fill* no es válido para las conversaciones correlacionadas.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *fill* no se modifica.

Set_Local_LU_Name (cmslln)

El programa que invoca emite la llamada Set_Local_LU_Name para especificar la LU local de una conversación. Esta llamada altera temporalmente la LU local definida por el sistema que se ha obtenido de la información complementaria al emitirse Initialize_Conversation, y la LU local especificada por la variable de entorno APPCLLU. Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate. La emisión de esta llamada no afecta a la información complementaria en sí.

Esta llamada no forma parte de la especificación CPI-C estándar y puede no estar disponible en otras implementaciones. En especial, no está soportada en otras implementaciones de CPI-C para Java.

Llamada de función

```
void cmslln   (
               unsigned char CM_PTR         Conversation_ID,
               unsigned char CM_PTR         lu_alias,
               CM_INT32 CM_PTR              lu_alias_length,
               CM_RETURN_CODE CM_PTR        return_code
              );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmslln (
                             byte[]             conversation_ID,
                             byte[]          lu_alias,
                             CPICLength      lu_alias_length,
                             CPICReturnCode     return_code
                           );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

lu_alias
Este parámetro especifica la dirección inicial del alias de LU. El alias de LU puede contener hasta 8 caracteres ASCII.

lu_alias_length
Este parámetro especifica la longitud del alias de LU. El rango de este valor es 0–8 (número de bytes). Si *lu_alias_length* es `0` (cero), el alias de LU se define en todo ceros.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_STATE_CHECK
La conversación no se encuentra en estado Inicializar.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:
- El valor especificado por *conversation_ID* no es válido.
- El valor especificado por *lu_alias_length* está fuera del rango (mayor que `8` o menor que `0`).

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor `CM_OK`, la característica de conversación *lu_alias* no se modifica.

Si se especifica un valor para *lu_alias* que no es válido (un nombre que no está permitido por el archivo de configuración), no se detecta hasta que se emite la llamada Allocate.

Set_Log_Data (cmsld)

La llamada Set_Log_Data especifica un mensaje de anotaciones (datos de anotaciones) y la longitud del mensaje que se enviará a la LU asociada. Esta llamada sólo está permitida en las conversaciones básicas. Altera temporalmente los datos de anotaciones por omisión, que son nulos, y la longitud de los datos de anotaciones por omisión, que es `0` (cero).

Llamada de función

```
void cmsld (
            unsigned char CM_PTR         conversation_ID,
            unsigned char CM_PTR       log_data,
            CM_INT32 CM_PTR            log_data_length,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsld (
                            byte[]             conversation_ID,
                            byte[]           log_data,
                            CPICLength       log_data_length,
                            CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

log_data
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.

La llamada Send_Error utiliza este parámetro si *log_data_length* es superior a `0` (cero).

El programa debe dar formato a los datos de error como una variable de anotaciones de error de corriente de datos general (GDS). Para más información, consulte la publicación de IBM: *IBM Systems Network Architecture: LU 6.2 Reference: Peer Protocols*.

log_data_length
Este parámetro especifica la longitud de los datos de anotaciones.

El rango de este valor es `0`–`512` (número de bytes).

La longitud `0` (cero) indica que no hay datos de anotaciones. El parámetro *log_data* no se tiene en cuenta y la característica de conversación *log_data* se establece en una cadena nula.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:
- El valor especificado por *conversation_ID* no es válido.
- El tipo de conversación definido se establece en correlacionada.
- El valor especificado por *log_data_length* está fuera del rango (mayor que 512 o menor que 0).

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, las características de conversación *log_data* y *log_data_length* no se modifican.

Los datos de anotaciones especificados por la llamada Set_Log_Data se envían a la LU asociada cuando el programa local emite una de las llamadas siguientes:
- Send_Error
- Deallocate con el parámetro *deallocate_type* de la conversación definido en CM_DEALLOCATE_ABEND
- Send_Data con el parámetro *send_type* de la conversación definido en CM_SEND_AND_DEALLOCATE y el parámetro *deallocate_type* definido en CM_DEALLOCATE_ABEND

Una vez enviados los datos de anotaciones a la LU asociada, la LU local restablece los datos de anotaciones en nulos y la longitud de los datos de anotaciones en 0 (cero).

CPI-C automáticamente convierte los datos de anotaciones de ASCII a EBCDIC según sea necesario.

Set_Mode_Name (cmsmn)

El programa que invoca emite la llamada Set_Mode_Name para especificar el nombre de modalidad de una conversación. Esta llamada altera temporalmente el nombre de modalidad definido por el sistema que se ha obtenido de la información complementaria al emitirse la llamada Initialize_Conversation. Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate. La emisión de esta llamada no afecta a la información complementaria en sí.

Llamada de función

```
void cmsmn (
           unsigned char CM_PTR         conversation_ID,
           unsigned char CM_PTR       mode_name,
           CM_INT32 CM_PTR            mode_name_length,
           CM_RETURN_CODE CM_PTR        return_code
          );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsmn (
                            byte[]            conversation_ID,
                            byte[]          mode_name,
                            CPICLength      mode_name_length,
                            CPICReturnCode    return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

mode_name
Este parámetro especifica la dirección inicial del nombre de modalidad (el nombre de un conjunto de características de red definidas durante la configuración). El nombre de modalidad puede contener hasta 8 caracteres ASCII. Están permitidos los caracteres siguientes:

- Letras en mayúsculas
- Números del `0` al `9`

El primer carácter del nombre debe ser una letra o puede ser `#` para una de las modalidades definidas por SNA, como por ejemplo, `#INTER`. For information about SNA-defined modes, see the *Communications Server for Linux Administration Guide*.

El valor de *mode_name* debe coincidir con el nombre de una modalidad asociada durante la configuración a la LU asociada.

Para una conversación correlacionada, el nombre de modalidad SNASVCMG está reservado para uso interno de SNA; la llamada Allocate fallará si utiliza este nombre. Se recomienda no utilizar SNASVCMG en una conversación básica, ni CPSVCMG (otro nombre reservado de SNA) en ninguno de los dos tipos de conversación.

mode_name_length
Este parámetro especifica la longitud del nombre de modalidad.

El rango de este valor es 0–8 (número de bytes).

Si el parámetro *mode_name_length* tiene el valor `0` (cero), la llamada Set_Mode_Name no se tiene en cuenta.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_STATE_CHECK
La conversación no se encuentra en estado Inicializar.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* no es válido.
- El valor especificado por *mode_name_length* está fuera del rango (mayor que 8 o menor que 0).

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *mode_name* no se modifica.

Si se especifica un valor para *mode_name* que no es válido (un nombre que no está permitido por el archivo de configuración), no se detecta hasta que se emite la llamada Allocate.

Set_Partner_LU_Name (cmspln)

El programa que invoca emite la llamada Set_Partner_LU_Name para especificar el nombre de LU asociada. Esta llamada altera temporalmente el nombre de LU asociada obtenido de la información complementaria al emitirse la llamada Initialize_Conversation. Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate. La emisión de esta llamada no afecta a la información complementaria en sí.

Llamada de función

```
void cmspln (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR      partner_LU_name,
             CM_INT32 CM_PTR           partner_LU_name_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmspln (
                            byte[]             conversation_ID,
                            byte[]          partner_LU_name,
                            CPICLength      partner_LU_name_length,
                            CPICReturnCode     return_code
                           );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

partner_LU_name
Este parámetro especifica la dirección inicial del nombre de LU asociada. Están permitidos los caracteres siguientes:

- Letras en mayúsculas
- Números del 0 al 9

El nombre de LU asociada puede ser:

- Un alias compuesto por 1–8 caracteres ASCII.
- Un nombre de red completamente calificado compuesto por 2–17 caracteres ASCII. Un punto (.) separa el identificador de red (que puede tener 0–8 caracteres) del nombre de LU de red (que puede tener 1–8 caracteres). Aunque el identificador de red tenga cero caracteres de longitud, el punto sigue siendo necesario.

Si la LU asociada se especifica mediante su alias, éste debe coincidir con el alias definido por una LU asociada en la configuración de Communications Server para Linux.

partner_LU_name_length
Este parámetro especifica la longitud del nombre de LU asociada.

El rango de este valor es 1–17.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_STATE_CHECK
La conversación no se encuentra en estado Inicializar.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* no es válido.
- El valor especificado por *partner_LU_name_length* está fuera del rango.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *partner_LU_name* no se modifica.

Si se especifica un valor para *partner_LU_name* que no es válido (un nombre no permitido por la configuración), no se detecta hasta que se emite la llamada Allocate.

Set_Prepare_To_Receive_Type (cmsptr)

La llamada Set_Prepare_To_Receive_Type especifica cómo se ejecutarán las llamadas Prepare_To_Receive posteriores. Altera temporalmente el proceso de preparación para recepción por omisión establecido por la llamada Initialize_Conversation o Accept_Conversation. Por omisión, el proceso de preparación para recepción se basa en el nivel de sincronización de la conversación.

El tipo de preparación para recepción afecta a todas las llamadas Prepare_To_Receive posteriores. Se puede cambiar volviendo a emitir la llamada Set_Prepare_To_Receive_Type.

Llamada de función

```
void cmsptr (
             unsigned char CM_PTR         conversation_ID,
             CM_PREPARE_TO_RECEIVE_TYPE CM_PTR      prepare_to_receive_type,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsptr (
                           byte[]             conversation_ID,
                           CPICPrepareToReceiveType  prepare_to_receive_type,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

prepare_to_receive_type
: Este parámetro especifica cómo se ejecutarán las llamadas Prepare_To_Receive posteriores. Los valores posibles son:

CM_PREP_TO_RECEIVE_CONFIRM
Este valor envía al programa asociado el contenido del almacenamiento intermedio de envío de la LU y una petición de confirmación. Al recibir la confirmación, la conversación cambia al estado Recibir.

CM_PREP_TO_RECEIVE_FLUSH
Este valor envía al programa asociado el contenido del almacenamiento intermedio de envío de la LU local y cambia el estado de la conversación a Recibir.

CM_PREP_TO_RECEIVE_SYNC_LEVEL
Este valor utiliza el nivel de sincronización de la conversación para determinar el proceso de preparación para recepción. La llamada Initialize_Conversation establece un nivel de sincronización por omisión que se puede alterar temporalmente mediante la llamada Set_Sync_Level.

Si el nivel de sincronización de la conversación está definido en el valor por omisión, CM_NONE, el contenido del almacenamiento intermedio de envío de la LU local se envía al programa asociado y el estado de la conversación cambia a Recibir.

Si el nivel de sincronización de la conversación es CM_CONFIRM, se envía al programa asociado el contenido del almacenamiento intermedio de envío de la LU local y una petición de confirmación. El estado de la conversación cambia a Recibir cuando el programa asociado emite la llamada Confirmed, como respuesta a la petición de confirmación.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por el parámetro *prepare_to_receive_type* o *conversation_ID* no es válido.
- El parámetro *prepare_to_receive_type* está definido en CM_PREP_TO_RECEIVE_CONFIRM, pero el nivel de sincronización de la conversación está definido en CM_NONE.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *prepare_to_receive_type* no se modifica.

Set_Processing_Mode (cmspm)

Esta función no está disponible en CPI-C para Java. Las funciones de CPI-C para Java siempre funcionan en modalidad con bloqueo; es decir, la función no devuelve el control a la aplicación hasta que el proceso solicitado se haya completado.

La llamada Set_Processing_Mode especifica si las llamadas CPI-C posteriores volverán cuando la operación solicitada haya finalizado (modalidad con bloqueo) o volverán de inmediato aunque la operación no haya finalizado (modalidad sin bloqueo). La modalidad de proceso por omisión, establecida por la llamada Initialize_Conversation o Accept_Conversation, es CM_BLOCKING (modalidad con bloqueo).

AIX, LINUX

Si la modalidad de proceso de la conversación es sin bloqueo, las llamadas CPI-C emitidas en esta conversación pueden volver de inmediato con el código de retorno CM_OPERATION_INCOMPLETE para indicar que la operación solicitada no ha finalizado. A partir de ese momento la aplicación puede llevar a cabo otro proceso que no esté relacionado con esta conversación o emitir cualquiera de las llamadas siguientes:

- Check_For_Completion, para determinar si ha finalizado alguna llamada pendiente (en esta conversación o en cualquier otra)
- Wait_For_Conversation, para esperar a que finalice esta llamada
- Cancel_Conversation, para cancelar la llamada pendiente y desasignar la conversación

WINDOWS

Una aplicación de Windows puede utilizar la llamada Wait_For_Conversation, tal como se ha descrito anteriormente. Sin embargo, el método recomendado para manejar llamadas sin bloqueo es utilizar Specify_Windows_Handle. Esta función, que debe emitirse antes que cualquier llamada sin bloqueo, especifica un descriptor de Windows al que CPI-C envía un mensaje cuando el proceso del verbo ha finalizado. La aplicación comprueba los resultados de la llamada cuando recibe este mensaje y no utiliza Wait_For_Conversation. Check_For_Completion, descrito anteriormente para sistemas AIX o Linux, no está soportado en sistemas Windows.

Si la llamada pendiente es una llamada Receive, una aplicación de Windows puede emitir llamadas Request_To_Send, Send_Error, Test_Request_to_Send_Received o Deallocate además de las listadas anteriormente. Para ver más información, consulte la llamada “Receive (cmrcv)” en la página 97.

La modalidad de proceso afecta a todas las llamadas CPI-C posteriores. Puede cambiarse volviendo a emitir la llamada Set_Processing_Mode.

Llamada de función

```
void cmspm (
            unsigned char CM_PTR         conversation_ID,
            CM_INT32 CM_PTR            processing_mode,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation o Accept_Conversation devuelve el valor de este parámetro.

processing_mode
Este parámetro especifica si las llamadas CPI-C posteriores se ejecutarán en la modalidad con bloqueo o en la modalidad sin bloqueo. Los valores posibles son:

CM_BLOCKING
Las llamadas CPI-C posteriores no volverán hasta que haya finalizado la operación.

CM_NON_BLOCKING
Las llamadas CPI-C posteriores volverán inmediatamente después de que se inicie la operación, tanto si ha finalizado como si no.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por el parámetro *processing_mode* o *conversation_ID* no es válido.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *processing_mode* no se modifica.

Set_Receive_Type (cmsrt)

La llamada Set_Receive_Type especifica cómo recibirá los datos el programa en las llamadas Receive posteriores. Altera temporalmente el tipo de recepción por omisión establecido por la llamada Initialize_Conversation o Accept_Conversation. Por omisión, el programa espera a que lleguen datos si no hay datos disponibles cuando se emite la llamada Receive.

El valor de tipo de recepción afecta a todas las llamadas Receive posteriores. Puede cambiarse volviendo a emitir la llamada Set_Receive_Type.

Llamada de función

```
void cmsrt (
            unsigned char CM_PTR         conversation_ID,
            CM_RECEIVE_TYPE CM_PTR     receive_type,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsrt (
                           byte[]             conversation_ID,
                           CPICReceiveType  receive_type,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
: Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

receive_type
: Este parámetro especifica cómo recibirá el programa los datos en las llamadas Receive posteriores. Los valores posibles son:

  CM_RECEIVE_AND_WAIT
  : El programa local recibe del programa asociado los datos que están actualmente disponibles. Si no hay datos disponibles, el programa local espera a que lleguen los datos.

  CM_RECEIVE_IMMEDIATE
  : El programa local recibe del programa asociado los datos que están actualmente disponibles. Si no hay datos disponibles, el programa local no espera.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* o *receive_type* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *receive_type* no se modifica.

Set_Return_Control (cmsrc)

El programa que invoca emite la llamada Set_Return_Control para especificar si la llamada Allocate vuelve de inmediato si no hay ninguna sesión disponible o espera a que se asigne una sesión.

Esta llamada altera temporalmente el control de retorno por omisión establecido por la llamada Initialize_Conversation. Por omisión, CPI-C espera a que se asigne la sesión. Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate.

Para ver más información sobre las sesiones, consulte el Capítulo 2, "Desarrollo de aplicaciones CPI-C", en la página 19.

Llamada de función

```
void cmsrc (
            unsigned char CM_PTR         conversation_ID,
            CM_RETURN_CONTROL CM_PTR     return_control,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsrc (
                           byte[]             conversation_ID,
                           CPICReturnControl  return_control,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

return_control
Este parámetro especifica cuándo debe devolver el control al programa la LU local, que participa en la llamada Allocate. Los valores permitidos son los siguientes:

CM_IMMEDIATE
La LU asigna una sesión ganadora de la contienda, si hay una disponible de inmediato, y devuelve el control al programa.

CM_WHEN_SESSION_ALLOCATED
La LU no devuelve el control al programa hasta que asigna una sesión o encuentra determinados errores. Si no existe ninguna sesión disponible, el programa espera a que haya una. (Si el límite de sesiones es `0`, la LU devuelve el control de inmediato.)

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_STATE_CHECK
La conversación no se encuentra en estado Inicializar.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* o *return_control* no es válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor `CM_OK`, la característica de conversación *return_control* no se modifica.

Si la LU no puede asignar una sesión, la notificación se devuelve en la llamada Allocate.

Set_Send_Type (cmsst)

La llamada Set_Send_Type especifica cómo enviará los datos la siguiente llamada Send_Data. Altera temporalmente el tipo de envío por omisión establecida por la llamada Initialize_Conversation o Accept_Conversation. El tipo de envío por omisión es CM_BUFFER_DATA, lo que indica que sólo se enviarán datos (y no información de control).

El valor de tipo de envío afecta a todas las llamadas Send_Data posteriores. Puede cambiarse volviendo a emitir la llamada Set_Send_Type.

Llamada de función

```
void cmsst (
            unsigned char CM_PTR         conversation_ID,
            CM_SEND_TYPE CM_PTR        send_type,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsst (
                           byte[]             conversation_ID,
                           CPICSendType    send_type,
                           CPICReturnCode     return_code
                         );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

send_type
Este parámetro especifica cómo enviarán los datos las llamadas Send_Data posteriores. Los valores posibles son:

CM_BUFFER_DATA
Los datos a los que apunta la llamada Send_Data se guardan en un almacenamiento intermedio hasta que éste se llena o se vacía.

CM_SEND_AND_FLUSH
Los datos a los que apunta la llamada Send_Data se enviarán de inmediato. Equivale a la llamada Send_Data, con el parámetro *send_type* definido en CM_BUFFER_DATA, seguida de Flush.

CM_SEND_AND_CONFIRM
Los datos se enviarán de inmediato con una petición de confirmación. Equivale a la llamada Send_Data, con el parámetro *send_type* definido en CM_BUFFER_DATA, seguida de Confirm.

CM_SEND_AND_PREP_TO_RECEIVE
Los datos se enviarán de inmediato junto con una notificación al

programa asociado de que el estado de conversación para el programa emisor cambia a Recibir. Equivale a la llamada Send_Data, con el parámetro *send_type* definido en CM_BUFFER_DATA, seguido de Prepare_To_Receive.

CM_SEND_AND_DEALLOCATE
Los datos se enviarán de inmediato junto con una notificación de desasignación. Equivale a la llamada Send_Data, con el parámetro *send_type* definido en CM_BUFFER_DATA, seguido de Deallocate.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* o *send_type* no es válido.
- El parámetro *send_type* está definido en CM_SEND_AND_CONFIRM, pero el nivel de sincronización de la conversación está definido en CM_NONE.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación puede encontrarse en cualquier estado salvo Restablecer.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *send_type* no se modifica.

El uso de valores de *send_type* distintos de CM_BUFFER_DATA permite reducir el número de llamadas emitidas, ya que con estos valores una llamada Send_Data puede incluir la función de otra llamada CPI-C.

Set_Sync_Level (cmssl)

El programa que invoca emite la llamada Set_Sync_Level para especificar el nivel de sincronización de la conversación. El nivel de sincronización determina si los programas sincronizan sus procesos mediante las llamadas Confirm y Confirmed.

Esta llamada altera temporalmente el nivel de sincronización establecido por la llamada Initialize_Conversation. El nivel de sincronización por omisión es CM_NONE, lo que indica que no hay sincronización. Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate.

Llamada de función

```
void cmssl (
            unsigned char CM_PTR         conversation_ID,
            CM_SYNC_LEVEL CM_PTR       sync_level,
            CM_RETURN_CODE CM_PTR        return_code
           );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmssl (
                           byte[]             conversation_ID,
                           CPICSyncLevel   sync_level,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

sync_level
Este parámetro especifica el nivel de sincronización de la conversación. Los valores posibles son:

CM_NONE
Los programas no llevarán a cabo el proceso de confirmación.

CM_CONFIRM
Los programas pueden llevar a cabo el proceso de confirmación.

Algunas implementaciones CPI-C proporcionan un tercer nivel, de punto de sincronismo pero no está soportado por CPI-C de Communications Server for Linux.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_STATE_CHECK
La conversación no se encuentra en estado Inicializar.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* o *sync_level* no es válido.

- El parámetro *sync_level* especifica CM_NONE pero se ha producido una de las situaciones siguientes:
  - El parámetro *send_type* está definido en CM_SEND_AND_CONFIRM.
  - El parámetro *prepare_to_receive_type* está definido en CM_PREP_TO_RECEIVE_CONFIRM.
  - El parámetro *deallocate_type* está definido en CM_DEALLOCATE_CONFIRM.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *sync_level* no se modifica.

Set_TP_Name (cmstpn)

El programa que invoca emite la llamada Set_TP_Name para especificar el nombre de programa asociado. Esta llamada altera temporalmente el nombre de programa asociado obtenido de la información complementaria al emitirse la llamada Initialize_Conversation. Esta llamada no puede emitirse después de que se haya emitido la llamada Allocate. La emisión de esta llamada no afecta a la información complementaria en sí.

Esta llamada funciona de forma distinta a Specify_Local_TP_Name. El programa que invoca emite Set_TP_Name para especificar el nombre del programa con el que desea asignar una conversación; el programa invocado emite Specify_Local_TP_Name para especificar un nombre para el que aceptará peticiones Allocate entrantes.

Llamada de función

```
void cmstpn (
             unsigned char CM_PTR         conversation_ID,
             unsigned char CM_PTR      TP_name,
             CM_INT32 CM_PTR           TP_name_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmstpn (
                           byte[]             conversation_ID,
                           byte[]          TP_name,
                           CPICLength      TP_name_length,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación. La llamada Initialize_Conversation devuelve el valor de este parámetro.

TP_name
Este parámetro especifica la dirección inicial del nombre de programa asociado. El nombre de programa puede contener hasta 64 caracteres. Están permitidos los caracteres siguientes:

- Letras en mayúsculas y en minúsculas
- Números del `0` al `9` y `.` (punto)
- Los caracteres especiales siguientes: `< > ( ) + - & *; / , % _ ? : ' = "` (válidos sólo si el programa asociado es un programa CPI-C) `$ # @` (válidos sólo si el programa asociado es un programa APPC)

No se puede utilizar la llamada Set_TP_Name para especificar el nombre de un TP de servicio SNA, que contiene caracteres no permitidos para esta llamada. Para ello, sin embargo, se puede utilizar la llamada Set_CPIC_Side_Information.

Los juegos de caracteres de doble byte, como por ejemplo Kanji, no están soportados.

TP_name_length
Este parámetro especifica la longitud del nombre de programa asociado.

El rango de este valor es `1–64`.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_STATE_CHECK
La conversación no se encuentra en estado Inicializar.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:

- El valor especificado por *conversation_ID* no es válido.
- El valor especificado por *TP_name_length* está fuera del rango.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

Estado al emitirse

La conversación debe encontrarse en estado Inicializar.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, la característica de conversación *TP_name* no se modifica.

Specify_Local_TP_Name (cmsltp)

Una aplicación CPI-C emite la llamada Specify_Local_TP_Name para especificar un nombre de TP local para el que aceptará peticiones Allocate entrantes.

En lugar de utilizar esta llamada, puede establecer el nombre de TP local de otras formas, como por ejemplo mediante la variable de entorno APPCTPN. Para ver más información sobre cómo establecer el nombre de TP local, consulte "Especificación del nombre de TP local" en la página 34. La llamada Specify_Local_TP_Name sólo es necesaria cuando una aplicación desea aceptar peticiones Allocate entrantes para más de un nombre de TP local; puede utilizar APPCTPN para un nombre, pero debe utilizar esta llamada para especificar nombres adicionales. (Tras emitir la llamada Accept_Conversation o Accept_Incoming para aceptar una petición Allocate entrante, puede utilizar Extract_TP_Name para determinar cuál de los nombres ha especificado la aplicación asociada.)

Esta llamada funciona de forma distinta a Set_TP_Name. El programa que invoca emite Set_TP_Name para especificar el nombre del programa con el que desea asignar una conversación; el programa invocado emite Specify_Local_TP_Name para especificar un nombre para el que aceptará peticiones Allocate entrantes.

Llamada de función

```
void cmsltp (
             unsigned char CM_PTR      TP_name,
             CM_INT32 CM_PTR           TP_name_length,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmsltp (
                            byte[]          TP_name,
                            CPICLength      TP_name_length,
                            CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

TP_name

Este parámetro especifica la dirección inicial del nombre de TP. El nombre puede contener hasta 64 caracteres. Están permitidos los caracteres siguientes:

- Letras en mayúsculas y en minúsculas
- Números del 0 al 9
- Los caracteres especiales: . < > ( ) + - & *; / , % _ ? : ' = "

No se puede utilizar la llamada Specify_Local_TP_Name para especificar el nombre de un TP de servicio SNA, que contiene caracteres no permitidos para esta llamada.

Los juegos de caracteres de doble byte, como por ejemplo Kanji, no están soportados.

TP_name_length
Este parámetro especifica la longitud del nombre.

El rango de este valor es 1–64.

Parámetros devueltos

Una vez que el verbo se ejecuta, Communications Server para Linux devuelve los parámetros siguientes:

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
Se ha producido una de estas situaciones:
- El valor especificado por *TP_name* es un nombre reservado o contiene uno o varios caracteres que no son válidos.
- El valor especificado por *TP_name_length* está fuera del rango.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

Esta llamada no está asociada a una conversación.

Cambio de estado

No hay ningún cambio de estado.

Notas de uso

Si el parámetro *return_code* no tiene el valor CM_OK, los nombres de TP para los que este programa aceptará peticiones Allocate entrantes no se modifican.

Si en el momento en que se emite esta llamada hay pendiente una llamada Accept_Incoming, ésta no aceptará una petición Allocate entrante para el nombre especificado en esta llamada. No obstante, las llamadas Accept_Conversation o Accept_Incoming posteriores aceptarán peticiones Allocate entrantes para este nombre.

Specify_Windows_Handle (xchwnd)

WINDOWS

Una aplicación CPI-C emite la llamada Specify_Windows_Handle para especificar un descriptor de Windows al que CPI-C enviará un mensaje cada vez que una función CPI-C sin bloqueo se complete. Esto proporciona un mecanismo alternativo al de la utilización de Wait_For_Conversation (como en los sistemas AIX o Linux) para esperar a que concluya la función. Si escribe una aplicación CPI-C nueva para los sistemas Windows, debe utilizar este mecanismo y no Wait_For_Conversation.

Para utilizar llamadas sin bloqueo y recibir mensajes para indicar que han finalizado, la aplicación debe emitir las llamadas siguientes antes de emitir una llamada sin bloqueo:

- `RegisterWindowMessage` para obtener el identificador de mensaje que CPI-C utilizará para los mensajes que indican la conclusión de una función CPI-C sin bloqueo. Es una llamada de función estándar de Windows, no específica de CPI-C; para más información, consulte la documentación de Windows. La aplicación debe pasar el valor `WIN_CPIC_ASYNC_COMPLETE_MESSAGE` a la función; el valor de retorno es un identificador de mensaje, tal como se describe a continuación. (No es necesario emitir de nuevo la llamada antes de posteriores llamadas CPI-C; el valor de retorno será el mismo para todas las llamadas emitidas por la aplicación).
- Set_Processing_Mode, para establecer la modalidad de proceso de la conversación en `CM_NON_BLOCKING`.
- Specify_Windows_Handle, para especificar el descriptor al que se envía el mensaje de finalización.

Cada vez que se completa una función CPI-C sin bloqueo, CPI-C anota un mensaje en el descriptor de ventana especificado en la llamada Specify_Windows_Handle. El formato del mensaje es el siguiente:

- El identificador de mensaje es el valor de retorno de la llamada `RegisterWindowMessage`.
- El argumento *lParam* contiene el ID de conversación de la llamada CPI-C que se ha completado.
- El argumento *wParam* contiene el parámetro *return_code* de conversación de la llamada CPI-C que se ha completado. Los valores posibles para este parámetro dependen de la llamada individual.

Llamada de función

```
void xchwnd (
             HWND                       hwnd,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Parámetros suministrados

El parámetro suministrado es:

hwnd Un descriptor de ventana que CPI-C utilizará para anotar un mensaje indicando que se ha completado una función sin bloqueo.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El parámetro suministrado no era un descriptor de Windows válido.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, "Códigos de retorno comunes", en la página 177.

Estado al emitirse

Esta llamada no está asociada a una conversación.

Cambio de estado

No hay ningún cambio de estado asociado a esta llamada.

Cuando CPI-C envía un mensaje para indicar que se ha completado una llamada sin bloqueo, el cambio de estado depende de la función que se ha completado y de su código de retorno.

Test_Request_to_Send_Received (cmtrts)

La llamada Test_Request_to_Send_Received determina si se ha recibido una notificación de petición de envío del programa asociado.

Llamada de función

```
void cmtrts (
             unsigned char CM_PTR         conversation_ID,
             CM_Request_to_Send_Received CM_PTR   request_to_send_received,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Llamada de función para CPI-C de Java

AIX, LINUX

```
public native void cmtrts (
                           byte[]             conversation_ID,
                           CPICControlInformationReceived  request_to_send_received,
                           CPICReturnCode     return_code
                          );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Identificador de la conversación.

La llamada Initialize_Conversation, Initialize_For_Incoming o Accept_Conversation devuelve el valor de este parámetro.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

request_to_send_received
Indicador de petición de envío recibida. Los valores posibles son:

CM_REQ_TO_SEND_RECEIVED
El programa asociado ha emitido la llamada Request_To_Send, que solicita al programa local que cambie la conversación al estado Recibir.

CM_REQ_TO_SEND_NOT_RECEIVED
El programa asociado no ha emitido la llamada Request_To_Send.

Este valor no es relevante si el parámetro *return_code* contiene un valor distinto de CM_OK.

return_code
Los valores posibles son:

CM_OK La llamada se ha ejecutado correctamente.

CM_PROGRAM_PARAMETER_CHECK
El valor especificado por *conversation_ID* no es válido.

CM_PROGRAM_STATE_CHECK
La conversación se encuentra en un estado no válido.

Para ver una explicación de los códigos de retorno siguientes, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.
CM_OPERATION_NOT_ACCEPTED
CM_PRODUCT_SPECIFIC_ERROR

Estado al emitirse

La conversación debe encontrarse en estado Recibir, Enviar, Enviar-Pendiente o Pendiente-Anotar.

Cambio de estado

No hay ningún cambio de estado.

Wait_For_Conversation (cmwait)

Esta función no está disponible en CPI-C para Java. Las funciones de CPI-C para Java siempre funcionan en modalidad con bloqueo; es decir, la función no devuelve el control a la aplicación hasta que el proceso solicitado se haya completado.

La llamada Wait_For_Conversation espera a que finalice una llamada CPI-C anterior que ha devuelto CM_OPERATION_INCOMPLETE.

Si el proceso de la llamada anterior ya ha finalizado cuando se emite Wait_For_Conversation, esta llamada vuelve de inmediato; de lo contrario, bloquea la aplicación hasta que CPI-C ha terminado de procesar la operación incompleta. Si

la aplicación participa en varias conversaciones, esta llamada espera en todas las conversaciones y vuelve tan pronto como finaliza una llamada en cualquiera de ellas.

WINDOWS

Las nuevas aplicaciones escritas para sistemas Windows deben utilizar Specify_Windows_Handle para obtener los resultados de llamadas sin bloqueo, en lugar de utilizar Wait_For_Conversation. Consulte el apartado "Specify_Windows_Handle (xchwnd)" en la página 156. La llamada Wait_For_Conversation se proporciona por motivo de compatibilidad con otras implementaciones CPI-C pero no se recomienda su uso en aplicaciones Windows.

En especial, si la aplicación emite la llamada Receive en modalidad sin bloqueo y a continuación emite otras llamadas en modalidad sin bloqueo en la misma conversación mientras Receive está pendiente, debe utilizar Specify_Windows_Handle. No debe emitir Wait_For_Conversation mientras hay más de una llamada pendiente en la misma conversación; los resultados de Wait_For_Conversation en esta situación no están definidos.

Llamada de función

```
void cmwait (
             unsigned char CM_PTR         conversation_ID,
             CM_INT32 CM_PTR           conversation_return_code,
             CM_RETURN_CODE CM_PTR        return_code
            );
```

Parámetros suministrados

No hay ningún parámetro suministrado para esta llamada.

Parámetros devueltos

Una vez que el verbo se ha ejecutado, Communications Server para Linux devuelve parámetros para indicar si la ejecución ha sido satisfactoria y, en caso contrario, para indicar la razón por la que la ejecución no lo ha sido.

conversation_ID
: Identificador de la conversación en que ha finalizado la llamada pendiente.

conversation_return_code
: Código de retorno de la llamada finalizada (que anteriormente ha devuelto CM_OPERATION_INCOMPLETE). Los valores posibles de este parámetro dependen de la llamada que estaba pendiente. Para ver más información, consulte la descripción de la llamada específica.

  Este valor no es relevante si el parámetro *return_code* contiene un valor distinto de CM_OK.

return_code
: Los valores posibles son:

  CM_OK La llamada Wait_For_Conversation se ha ejecutado correctamente. El parámetro *conversation_return_code* indica si la operación incompleta anterior ha finalizado correctamente.

CM_PROGRAM_STATE_CHECK
No había ninguna operación incompleta pendiente.

CM_PRODUCT_SPECIFIC_ERROR
Para ver una explicación de este código de retorno, consulte el Apéndice B, “Códigos de retorno comunes”, en la página 177.

WINDOWS

CM_SYSTEM_EVENT
Un suceso del sistema operativo ha terminado la llamada en lugar de que la complete una llamada CPI-C anterior.

Estado al emitirse

La llamada no está asociada a ninguna conversación específica, por lo que el estado de conversación no es relevante. No obstante, la aplicación debe tener como mínimo una conversación con una operación incompleta pendiente.

Cambio de estado

Si *return_code* está definido en CM_OK, el cambio de estado depende de la llamada pendiente que ha finalizado y del código de retorno de esa llamada (el parámetro *conversation_return_code* de esa llamada). Para ver más información, consulte la descripción de la llamada específica. Si *return_code* no está definido en CM_OK, no hay ningún cambio de estado.

Notas de uso

Esta llamada no cambia el contexto actual del programa (aunque normalmente la operación pendiente que ha finalizado lo cambiaría, como por ejemplo Accept_Incoming). Si es necesario, el programa puede utilizar Extract_Conversation_Context para el *conversation_ID* devuelto en esta llamada con el fin de obtener el valor del contexto de la conversación y Set_Conversation_Context para establecer el contexto actual en este valor.

Si no ha finalizado ninguna llamada pendiente anterior, esta llamada bloquea la aplicación (y se suspende el proceso de la misma) hasta que alguna finaliza.

AIX, LINUX

Para comprobar si ha finalizado alguna llamada sin bloquear la aplicación, ésta puede utilizar Check_For_Completion (que siempre vuelve de inmediato) para determinar si una llamada ha finalizado, y la llamada Wait_For_Conversation únicamente cuando Check_For_Completion indique que ha finalizado una llamada (por lo que Wait_For_Conversation volverá de inmediato).

Si hay varias llamadas pendientes (en distintas conversaciones), cada una de las llamadas Wait_For_Conversation devuelve una llamada pendiente. Tras emitir Wait_For_Conversation, la aplicación puede comprobar si ha finalizado alguna otra llamada emitiendo Check_For_Completion.

WINDOWS

Una aplicación de Windows puede utilizar Wait_For_Conversation, tal como se ha descrito anteriormente. Sin embargo, el método recomendado para manejar llamadas sin bloqueo es utilizar Specify_Windows_Handle. Esta función, que debe emitirse antes que cualquier llamada sin bloqueo, especifica un descriptor de Windows al que CPI-C envía un mensaje cuando el proceso del verbo ha finalizado. La aplicación comprueba los resultados de la llamada cuando recibe este mensaje y no utiliza Wait_For_Conversation. Check_For_Completion, descrito anteriormente para sistemas AIX o Linux, no está soportado en sistemas Windows.

WinCPICCleanup

WINDOWS

La aplicación utiliza esta función para eliminar el registro de un usuario CPI-C de Windows, después de que haya finalizado de emitir llamadas CPI-C.

Llamada de función

```
BOOL WINAPI WinCPICCleanup (void);
```

Parámetros suministrados

No hay ningún parámetro suministrado para esta llamada.

Valores de retorno

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

TRUE La aplicación ha eliminado el registro de forma satisfactoria.

FALSE Se ha producido un error durante el proceso de la llamada y la aplicación no se ha registrado. Compruebe en los archivos de anotaciones los mensajes que indican la causa del error.

WinCPICIsBlocking

WINDOWS

La aplicación utiliza esta función para comprobar si hay una llamada pendiente CPI-C de bloqueo (una llamada emitida con la modalidad de proceso de la conversación establecida en CM_BLOCKING). Para obtener más información sobre cómo bloquear llamadas, consulte el apartado “consideraciones sobre Windows” en la página 43.

Llamada de función

```
BOOL WINAPI WinCPICIsBlocking (void);
```

Parámetros suministrados

No hay ningún parámetro suministrado para esta función.

Valores de retorno

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

TRUE Hay una llamada CPI-C de bloqueo pendiente. Si es necesario, la aplicación puede utilizar Cancel_Conversation o Deallocate para cancelar la llamada y finalizar la conversación.

FALSE No hay ninguna llamada CPI-C de bloqueo pendiente.

WinCPICSetBlockingHook

WINDOWS

La aplicación utiliza esta llamada para especificar su propia función de bloqueo, que CPI-C utilizará en lugar de la función de bloqueo por omisión. Para obtener más información sobre el funcionamiento de la función de bloqueo y sobre las funciones que debe realizar, consulte el apartado “Llamadas de bloqueo” en la página 44.

Llamada de función

```
FARPROC WINAPI WinCPICSetBlockingHook (FARPROC lpBlockFunc);
```

Parámetros suministrados

El parámetro suministrado es:

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

Valores de retorno

El valor de retorno es la dirección de la instancia de procedimiento de la función de bloqueo anterior. Si la aplicación utiliza más de una función de bloqueo y tiene que restaurar más tarde la función de bloqueo anterior, deberá guardar esta dirección; a continuación, puede emitir de nuevo `WinCPICSetBlockingHook` utilizando el valor guardado, para restaurar la función de bloqueo anterior. Si sólo utiliza una función de bloqueo o va a necesitar restaurar el valor anterior, puede ignorar el valor de retorno de esta llamada.

Uso

La nueva función de bloqueo permanece vigente hasta que la aplicación emita una de las llamadas que se indican a continuación:

- `WinCPICSetBlockingHook` (con una dirección de instancia de procedimiento diferente) para especificar una función de bloqueo nueva o restaurar una anterior
- `WinCPICUnhookBlockingHook` (que se describe a continuación) para dejar de utilizar la nueva función de bloqueo anterior y volver a la función de bloqueo por omisión.

WinCPICStartup

WINDOWS

La aplicación utiliza esta función para registrarse como usuario CPI-C de Windows y para determinar si el software CPI-C soporta la versión CPI-C de Windows que necesita.

Llamada de función

```
int WINAPI WinCPICStartup (
                            WORD              wVersionRequired;
                            LPWCPICDATA       lpData;
                           )
typedef struct
{
  WORD        wVersion;
  char        szDescription[128];
} WCPICDATA;
```

Parámetros suministrados

El parámetro suministrado es:

wVersionRequired
La versión de CPI-C para Windows que la aplicación requiere. Communications Server para Linux soporta la versión 1.0.

El byte de orden inferior de este parámetro especifica el número de versión principal y el byte de orden superior especificar el número de versión menor. Por ejemplo:

Versión	wVersionRequired
1.0	0x0001
1.1	0x0101
2.0	0x0002

Si la aplicación puede utilizar más de una versión, debe especificar la versión superior que puede utilizar.

Valores de retorno

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

0 (cero)
La aplicación se ha registrado con éxito y el software CPI-C de Windows da soporte al número de versión especificado por la aplicación o una

versión inferior. La aplicación debe comprobar el número de versión en la estructura WCPICDATA (véase la descripción que viene a continuación) para asegurarse de que sea suficientemente alto.

WCPICVERNOTSUPPORTED
El número de versión especificado por la aplicación era menor que la versión inferior soportada por el software CPI-C de Windows. No se ha registrado la aplicación.

WCPICSYSNOTREADY
No se ha iniciado el software Communications Server para Linux o el nodo local no está activo. No se ha registrado la aplicación.

Si el valor de retorno de WinCPICStartup es 0 (cero), la estructura WCPICDATA contiene información sobre el soporte proporcionado por el software CPI-C Windows. Si el valor de retorno es distinto de cero, el contenido de esta estructura no está definido y la aplicación no debería comprobarlo. Los parámetros de esta estructura son los siguientes:

wVersion
El número de versión CPI-C de Windows que el software soporta, en el mismo formato que el parámetro *wVersionRequired* (vea la explicación anterior). Communications Server para Linux soporta la versión 1.0.

Si el software soporta el número de versión solicitado, este parámetro se establece en el mismo valor que el parámetro *wVersionRequired*; de lo contrario, se establece en la versión superior que el software soporta, que será menor que el número de versión suministrado por la aplicación. La aplicación debe comprobar el valor de retorno y realiza la acción siguiente:

- Si el número de versión de retorno es el mismo que el número de versión solicitado, la aplicación puede utilizarlo como implementación CPI-C de Windows.
- Si el número de versión de retorno es menor que el número de versión solicitado, la aplicación puede utilizarlo como implementación CPI-C de Windows pero no debe intentar utilizar funciones que no estén soportadas por el número de versión devuelto. Si no puede conseguirlo porque precisa de funciones que no están disponibles en la versión inferior, la inicialización debería fallar y no intentará emitir ninguna llamada CPI-C.

szDescription
Una cadena de texto que describe el software CPI-C de Windows.

WinCPICUnhookBlockingHook

WINDOWS

La aplicación utiliza esta llamada para eliminar su propia función de bloqueo, que se ha especificado anteriormente mediante WinCPICSetBlockingHook y utiliza de nuevo la función de bloqueo por omisión de CPI-C.

Llamada de función

```
BOOL WINAPI WinCPICUnhookBlockingHook (void);
```

Parámetros suministrados

No hay ningún parámetro suministrado para esta función.

Valores de retorno

El valor de retorno es uno de los siguientes:

TRUE La función de bloqueo se ha eliminado con éxito; cualquier otra llamada de bloqueo posterior utilizará la función de bloqueo por omisión.

FALSE La llamada no se ha completado correctamente.

WinCPICSetEvent

WINDOWS

La aplicación utiliza esta función para asociar un descriptor de sucesos con la terminación del verbo para la conversación especificada.

Llamada de función

```
VOID WINAPI WinCPICSetEvent (
                             unsigned char CM_PTR         conversation_ID,
                             HANDLE CM_PTR         event_handle,
                             CM_INT32 CM_PTR       return_code
                            );
```

Parámetros suministrados

Los parámetros suministrados son:

conversation_ID
Es el identificador de la conversación para la cual se utiliza el suceso. Este parámetro lo devuelve la llamada Accept_Conversation inicial.

event_handle
Es el descriptor del suceso que se va borrar cuando se completa un verbo asíncrono en la conversación. Este parámetro puede sustituir un suceso ya definido o eliminar un suceso ya definido (teniendo como parámetro NULL).

Parámetros devueltos

return_code
Los valores posibles son:

CM_OK La función `WinCPICSetEvent` se ha ejecutado satisfactoriamente.

CM_PROGRAM_PARAMETER_CHECK
Uno o más de los parámetros pasados a esta función no son válidos.

CM_OPERATION_NOT_ACCEPTED
Este valor indica que una operación anterior en esta conversación está incompleta y no se ha aceptado la llamada `WinCPICSetEvent`.

Notas de uso

Cuando se emite un verbo en una conversación sin bloqueo, devuelve CM_OPERATION_INCOMPLETE si va a completarla asíncronamente. Si se ha registrado un suceso con la conversación, la aplicación puede llamar WaitForSingleObject o WaitForMultipleObjects para que se le notifique que ha finalizado el verbo. Cuando se ha completado el verbo, la aplicación debe llamar Wait_for_Conversation para determinar el código de retorno del verbo asíncrono.

Es responsabilidad de la aplicación restablecer el suceso.

WinCPICExtractEvent

WINDOWS

La aplicación utiliza esta función para determinar el descriptor de sucesos que se utiliza para una conversación CPI-C.

Llamada de función

```
VOID WINAPI WinCPICExtractEvent (
                                 unsigned char CM_PTR         conversation_ID,
                                 HANDLE CM_PTR         event_handle,
                                 CM_INT32 CM_PTR       return_code
                                );
```

Parámetros suministrados

El parámetro suministrado para esta función es:

conversation_ID
: Es el identificador de la conversación para la cual se utiliza el suceso. Este parámetro lo devuelve la llamada Accept_Conversation inicial.

Parámetros devueltos

event_handle
: Es el descriptor del suceso que esta conversación utiliza. Si no se ha registrado ningún descriptor de suceso, este parámetro devuelve un valor NULL.

return_code
: Los valores posibles son:

CM_OK La función WinCPICExtractEvent se ha ejecutado satisfactoriamente.

CM_PROGRAM_PARAMETER_CHECK
: Uno o más de los parámetros pasados a esta función no son válidos.

Notas de uso

Cuando se emite un verbo en una conversación sin bloqueo, devuelve CM_OPERATION_INCOMPLETE si va a completarla asíncronamente. Si se ha registrado un suceso con la conversación, la aplicación puede llamar WaitForSingleObject o WaitForMultipleObjects para que se le notifique que ha finalizado el verbo.

`WinCPICExtractEvent` permite que una aplicación CPI-C determine este descriptor de sucesos. Cuando se ha completado el verbo, la aplicación debe llamar Wait_for_Conversation para determinar el código de retorno del verbo asíncrono.

Se puede invocar la función Cancel_Conversation para cancelar una operación y una conversación.

Si no se ha registrado ningún suceso, se completa el verbo asíncrono anotando un mensaje en la ventana que ha registrado la aplicación con la biblioteca CPI-C.

Capítulo 4. Programas de transacciones CPI-C de ejemplo

Este capítulo describe los programas de transacciones CPI-C de ejemplo de Communications Server para Linux, que ilustran el uso de llamadas CPI-C en aplicaciones AIX or Linux. Para obtener información sobre cómo utilizar llamadas CPI-C en una aplicación Java, consulte el Capítulo 5, "Programa de transacciones CPI-C de Java de ejemplo", en la página 173.

Se facilita la información siguiente:

- Visión general del proceso de los dos programas
- Pseudocódigo para cada programa
- Instrucciones para compilar, enlazar y ejecutar los dos programas

Visión general del proceso

Los programas 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 bloque siguiente, el anterior o salir.

CSAMPLE1 (el programa que invoca) envía un nombre de archivo a CSAMPLE2 (el programa invocado). Si CSAMPLE2 localiza el archivo, devuelve el primer bloque de datos a CSAMPLE1; de lo contrario, desasigna la conversación y finaliza.

Si CSAMPLE1 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, CSAMPLE1 envía la petición a CSAMPLE2, que a su vez envía el bloque adecuado. Este proceso continúa hasta que el usuario selecciona la opción de salir, momento en el que CSAMPLE1 desasigna la conversación y ambos programas finalizan.

Si el usuario solicita el bloque siguiente y CSAMPLE2 ha enviado el último, CSAMPLE2 vuelve al inicio del archivo. Igualmente, si el usuario solicita el bloque anterior y se está visualizando el primer bloque, CSAMPLE2 retrocede para enviar el último bloque.

Ningún programa intenta efectuar la recuperación de errores. Un código de retorno incorrecto de CPI-C hace que el programa finalice con un mensaje explicativo.

Pseudocódigo

Este apartado contiene el pseudocódigo para los programas de transacciones CSAMPLE1 y CSAMPLE2.

Los programas de ejemplo se proporcionan como **csample1.c** y **csample2.c**, en el directorio **/usr/lib/sna/samples** (AIX) o **/opt/ibm/sna/samples** (Linux).

CSAMPLE1 (programa que invoca)

El pseudocódigo para CSAMPLE1 (el programa que invoca) es el siguiente:

```
initialize
allocate
send data (data = filename)
do while no error and prompt not Q
    receive
    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 */
            send data (data = prompt)
        endif
    endif
end do
deallocate
```

CSAMPLE2 (TP invocado)

El pseudocódigo para CSAMPLE2 (el TP invocado) es el siguiente:

```
initialize
do while conversing
    receive
    if data received
        if first time (data = filename)
            open file
            if file not found
                deallocate
                set conversing false
            endif
        else (data = prompt)
            read and store prompt
        endif
        if (conversing)
            read file block
            send data (file block)
        endif
    else if deallocate received
        set conversing false
    endif
 end while conversing
close file
```

Cómo probar los TP

Tras examinar el código fuente de CSAMPLE1 y CSAMPLE2, es posible que desee probar los programas.

Aunque CPI-C se utiliza normalmente para las comunicaciones entre programas en máquinas diferentes, puede que le resulte cómodo ejecutar ambos programas en la misma máquina para realizar la comprobación.

Para compilar y enlazar los programas para un sistema AIX o Linux, realice los pasos siguientes:

1. Copie los dos archivos **csample1.c** y **csample2.c** desde el directorio **/usr/lib/sna/samples** (AIX) o **/opt/ibm/sna/samples** (Linux) en un directorio privado.
2. To compile and link the programs for AIX, use the following commands:

```
cc -o csample1 -I /usr/include/sna -bimport:/usr/lib/sna/cpic_r.exp csample1.c

cc -o csample2 -I /usr/include/sna -bimport:/usr/lib/sna/cpic_r.exp csample2.c
```

To compile and link the programs for Linux, use the following commands:

```
gcc -o csample1 -I /opt/ibm/sna/include -L /opt/ibm/sna/lib -lcpic -lappc -lnof -lsna_r -lpLiS -lpthread -lpLis csample1.c

gcc -o csample2 -I /opt/ibm/sna/include -L /opt/ibm/sna/lib -lcpic -lappc -lnof -lsna_r -lpLiS -lpthread -lpLis csample2.c
```

Para ejecutar los programas, siga los pasos que se describen a continuación. Note that some of these steps involve updating the Communications Server for Linux configuration, which is usually performed by the System Administrator.

Los programas se pueden ejecutar en la misma máquina o en máquinas diferentes. En los pasos siguientes, la "máquina de origen" es la máquina en que se ejecuta el programa que invoca CSAMPLE1 y la "máquina de destino" es la máquina en que se ejecuta el programa invocado CSAMPLE2.

1. Si ejecuta los programas en máquinas diferentes, configure el enlace de comunicaciones para dar soporte a las sesiones CP-CP entre las máquinas de origen y de destino. See *Communications Server for Linux Administration Guide* for more information.
2. Configure una modalidad con el nombre de modalidad LOCMODE.
3. Configure una unidad lógica (LU) en la máquina de origen para CSAMPLE1 (el programa que invoca). Especifique `TPLU1` como nombre de LU y como alias de LU. Acepte los valores por omisión para el resto de parámetros.
4. Configure un nombre de destino simbólico en la máquina de origen. Lleve a cabo las acciones siguientes:
   - Para *Nombre*, especifique `CPICTEST`.
   - Para *LU local*, seleccione *Alias de LU local* y especifique `TPLU1` como el alias de LU.
   - Para *LU asociada*, especifique el nombre completamente calificado *nombre_red***.TPLU2**, donde *nombre_red* es el nombre de red SNA de la máquina de destino.
   - Para *Modalidad*, especifique `LOCMODE`.
   - Para *TP asociado*, especifique `TPNAME2`.

   Acepte los valores por omisión para otros parámetros.
5. Configure una LU en la máquina de destino para CSAMPLE2 (el programa invocado). Especifique `TPLU2` como nombre de LU y como alias de LU. Acepte los valores por omisión para el resto de parámetros.
6. Configure the invoked TP in the Communications Server for Linux invokable TP data file on the target computer. Refer to the *Communications Server for Linux Administration Guide* for more information.
   - 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 **csample2**.
   - For the *User ID* parameter, specify your Linux user ID on the target computer.
   - 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 csample2
8. Inicie el software Communications Server for Linux mediante este archivo de configuración.
9. Defina las variables de entorno siguientes:

- APPCLLU en TPLU1 (el nombre de la LU local para **csample1**)
- APPCTPN en TPNAME1

10. Inicie el programa que invoca, **csample1**. Este programa requiere un parámetro, el nombre completo de la vía de acceso (en la máquina de destino) del archivo que se visualizará. Por ejemplo:

```
csample1 /usr/jim/myfile
```

11. Entre F o B para visualizar bloques del archivo solicitado. Utilice Q para finalizar el programa invocado; el programa que invoca también finalizará.

Capítulo 5. Programa de transacciones CPI-C de Java de ejemplo

AIX, LINUX

Este capítulo describe el programa de transacciones CPI-C de Java de ejemplo de Communications Server para Linux, **JPing**, que ilustra el uso de llamadas CPI-C en una aplicación Java. Para ver información sobre la utilización de llamadas CPI-C en un programa C estándar, consulte el Capítulo 4, "Programas de transacciones CPI-C de ejemplo", en la página 169.

Se facilita la información siguiente:

- Visión general del programa
- Instrucciones para compilar, enlazar y ejecutar el programa

Visión general

El programa CPI-C para Java de ejemplo, **JPing** (en el archivo **/usr/lib/sna/samples/JPing.java** (AIX) o **/opt/ibm/sna/samples/JPing.java** (Linux)) es una implementación simple de Java de la función APPC estándar **aping**, que se utiliza para comprobar la conectividad con un nodo remoto. Para obtener más información sobre **aping**, consulte la publicación *Communications Server for Linux APPC Application Suite User's Guide* o la publicación *Communications Server for Linux Administration Command Reference*.

Puede especificar de manera opcional un nombre de destino simbólico que identifique la LU asociada con la que se debe establecer contacto, el número de iteraciones de ping que se debe intentar y el tamaño de la información enviada en cada iteración.

Para ver más información sobre el funcionamiento del programa, consulte los comentarios del archivo fuente del programa.

Compilación, enlace y ejecución del programa de ejemplo

Tras examinar el código fuente de **JPing**, puede que desee generar y probar el programa.

Antes de compilar y enlazar una aplicación CPI-C para Java, especifique el directorio donde están almacenadas las clases de Java. Para ello, establezca y exporte la variable de entorno `CLASSPATH` en **/usr/lib/sna/java/cpic.jar:.** (AIX) o **/opt/ibm/sna/java/cpic.jar:.** (Linux).

Las aplicaciones CPI-C para Java siempre se crean como aplicaciones de 32 bits; las aplicaciones de 64 bits no están soportadas.

Para compilar y enlazar el programa, realice los pasos siguientes.

1. Copie el archivo **JPing.java** desde el directorio **/usr/lib/sna/samples** (AIX) o **/opt/ibm/sna/samples** (Linux) en un directorio privado.

2. Desde el directorio privado, compile y enlace la aplicación mediante el compilador de Java, **javac** del modo habitual, utilizando el mandato siguiente:

   javac JPing.java

Debe observar que se haya generado el archivo **JPing.class**.

Antes de ejecutar una aplicación CPI-C para Java, debe especificar el directorio donde están almacenadas las bibliotecas, de modo que la aplicación pueda localizarlas durante la ejecución. También debe establecer una variable de entorno adicional para asegurarse de que CPI-C para Java funcione correctamente con LiS Streams. Para ello, establezca y exporte las variables de entorno adecuadas tal como se indica a continuación:

export LD_LIBRARY_PATH=/opt/ibm/sna/lib
export LD_PRELOAD=/usr/lib/libpLiS.so

Es probable que también deba establecer y exportar la variable de entorno `APPCTPN` para especificar el nombre de TP local para la aplicación, tal como se describe en "Especificación del nombre de TP local" en la página 34.

La ejecución del programa implica la actualización de la configuración de Communications Server for Linux para que incluya un nombre de destino simbólico que identifique la LU asociada. El administrador del sistema realiza habitualmente esta tarea. Deben llevarse a cabo los pasos siguientes:

- Para Nombre de destino simbólico, especifique `JPING`
- Para Tipo de nombre de TP asociado, especifique `Programa de aplicación`.
- Para Nombre de TP asociado, especifique `APINGD`
- Para LU asociada, especifique el nombre completamente calificado de la LU asociada con la que desea ponerse en contacto.
- Par Nombre de modalidad, especifique `#INTER`.

Acepte los valores por omisión para otros parámetros.

Ejecute la aplicación mediante el intérprete de Java, **java** del modo habitual. Utilice el mandato siguiente:

java JPing [*nombre_dest_sim*] [**-i** *núm_iteraciones*] [**-s** *long_datos*]

nombre_dest_sim indica el nombre de destino simbólico que el programa debe utilizar. Si no especifica esta opción, el valor por omisión es `JPING`.

La opción **-i** indica el número de iteraciones de ping que deben llevarse a cabo. Si no especifica esta opción, el valor por omisión es `2`.

La opción **-s** indica el número de bytes de datos que deben enviarse al programa asociado. Si no especifica esta opción, el valor por omisión es `100`.

Para más información sobre cómo se utilizan el número de iteraciones de ping y la longitud de datos, consulte la descripción de **aping** en la publicación *Communications Server for Linux APPC Application Suite User's Guide* o la publicación *Communications Server for Linux Administration Command Reference*.

Apéndice A. Valores de código de retorno

Este apéndice lista todos los códigos de retorno posibles en la interfaz de CPI-C en orden numérico. Los valores están definidos en el archivo de cabecera **cmc.h**(para AIX o Linux) o **wincpic.h** (para Windows).

Puede utilizar este apéndice como referencia para comprobar el significado de un código de retorno recibido por la aplicación.

```
CM_OK                                0
CM_ALLOCATE_FAILURE_NO_RETRY         1
CM_ALLOCATE_FAILURE_RETRY            2
CM_CONVERSATION_TYPE_MISMATCH        3
CM_PIP_NOT_SPECIFIED_CORRECTLY       5
CM_SECURITY_NOT_VALID                6
CM_SYNC_LVL_NOT_SUPPORTED_LU         7
CM_SYNC_LVL_NOT_SUPPORTED_PGM        8
CM_TPN_NOT_RECOGNIZED                9
CM_TP_NOT_AVAILABLE_NO_RETRY        10
CM_TP_NOT_AVAILABLE_RETRY           11
CM_DEALLOCATED_ABEND                17
CM_DEALLOCATED_NORMAL               18
CM_PARAMETER_ERROR                  19
CM_PRODUCT_SPECIFIC_ERROR           20
CM_PROGRAM_ERROR_NO_TRUNC           21
CM_PROGRAM_ERROR_PURGING            22
CM_PROGRAM_ERROR_TRUNC              23
CM_PROGRAM_PARAMETER_CHECK          24
CM_PROGRAM_STATE_CHECK              25
CM_RESOURCE_FAILURE_NO_RETRY        26
CM_RESOURCE_FAILURE_RETRY           27
CM_UNSUCCESSFUL                     28
CM_DEALLOCATED_ABEND_SVC            30
CM_DEALLOCATED_ABEND_TIMER          31
CM_SVC_ERROR_NO_TRUNC               32
CM_SVC_ERROR_PURGING                33
CM_SVC_ERROR_TRUNC                  34
CM_OPERATION_INCOMPLETE             35
CM_SYSTEM_EVENT                     36
CM_OPERATION_NOT_ACCEPTED           37
CM_CONVERSATION_ENDING              38
CM_SEND_RCV_MODE_NOT_SUPPORTED      39
CM_BUFFER_TOO_SMALL                 40
CM_EXP_DATA_NOT_SUPPORTED           41
CM_DEALLOC_CONFIRM_REJECT           42
CM_ALLOCATION_ERROR                 43
CM_RETRY_LIMIT_EXCEEDED             44
CM_NO_SECONDARY_INFORMATION         45
CM_SECURITY_NOT_SUPPORTED           46
CM_SECURITY_MUTUAL_FAILED           47
CM_CALL_NOT_SUPPORTED               48
CM_PARM_VALUE_NOT_SUPPORTED         49
CM_TAKE_BACKOUT                    100
CM_DEALLOCATED_ABEND_BO            130
CM_DEALLOCATED_ABEND_SVC_BO        131
CM_DEALLOCATED_ABEND_TIMER_BO      132
CM_RESOURCE_FAIL_NO_RETRY_BO       133
CM_RESOURCE_FAILURE_RETRY_BO       134
CM_DEALLOCATED_NORMAL_BO           135
CM_CONV_DEALLOC_AFTER_SYNCPT       136
CM_INCLUDE_PARTNER_REJECT_BO       137
```

Apéndice B. Códigos de retorno comunes

Este apéndice describe los códigos de retorno comunes a varias llamadas CPI-C. Los códigos de retorno figuran en orden alfabético. Los códigos de retorno que se generan cuando el programa asociado es un programa de LU 6.2 que no es CPI-C se indican aparte.

Los códigos de retorno específicos de llamadas se describen en la documentación de las llamadas individuales en el Capítulo 3, "Llamadas CPI-C", en la página 49.

Códigos de retorno de cualquier programa asociado

Los códigos de retorno siguientes se pueden producir con cualquier programa asociado. (Otros códigos de retorno, que sólo se pueden producir cuando el programa asociado no es un programa CPI-C, se indican aparte.)

CM_ALLOCATION_FAILURE_NO_RETRY
No se puede asignar 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.

CM_ALLOCATION_FAILURE_RETRY
No se puede asignar la conversación debido a una condición temporal, como una anomalía del enlace. El motivo de la anomalía está anotado en el archivo de anotaciones de error del sistema. Vuelva a intentar la asignación.

AIX, LINUX

CM_CALL_NOT_SUPPORTED
Este código de retorno sólo se utiliza en aplicaciones CPI-C de Java.

La aplicación ha utilizado una función CPI-C que está definida en la clase CPI-C de Java, pero que no está soportada.

CM_CONVERSATION_TYPE_MISMATCH
La LU asociada o el programa asociado no da soporte al tipo de conversación (básica o correlacionada) especificado en la petición de asignación.

CM_DEALLOCATED_ABEND
La conversación se ha desasignado por uno de los motivos siguientes:

- El programa asociado ha emitido la llamada Deallocate con el tipo de desasignación definido en `CM_DEALLOCATE_ABEND`. Si la conversación se encuentra en estado Recibir para el programa asociado cuando el programa local emite esta llamada, se eliminan los datos enviados por el programa local que el programa asociado todavía no ha recibido.
- El programa asociado ha finalizado normalmente pero no ha desasignado la conversación antes de terminar.

- El programa local ha emitido la llamada Cancel_Conversation, que cancela todas las llamadas CPI-C asíncronas pendientes de la conversación.

CM_DEALLOCATED_NORMAL
Este código de retorno no indica un error.

El programa asociado ha emitido la llamada Deallocate con el tipo de desasignación definido en uno de los siguientes:

- CM_DEALLOCATE_FLUSH
- CM_DEALLOCATE_SYNC_LEVEL con el nivel de sincronización de la conversación especificado como CM_NONE

CM_OK La llamada se ha ejecutado correctamente.

CM_OPERATION_INCOMPLETE
La llamada se ha emitido correctamente y opera en la modalidad sin bloqueo (es decir, se ha devuelto el control al programa aunque el proceso de la llamada todavía no haya finalizado).

El programa puede llevar a cabo otro proceso que no esté relacionado con esta conversación (como por ejemplo emitir llamadas CPI-C en otras conversaciones). Puede emitir una serie limitada de llamadas CPI-C por esta conversación (como, por ejemplo, las llamadas Extract_*). Es diferente de la especificación CPI-C 2.0 de IBM en la que el programa no puede emitir ninguna llamada en esta conversación, excepto Wait_For_Conversation o Cancel_Conversation.

AIX, LINUX

Posteriormente, la aplicación puede emitir Check_For_Completion para determinar si ha finalizado la llamada sin bloqueo pendiente, Wait_For_Conversation para esperar a que finalice o Cancel_Conversation para cancelar la llamada pendiente y finalizar la conversación.

WINDOWS

Si la aplicación ha utilizado Specify_Windows_Handle para recibir la notificación de terminación de llamada asíncrona, no debe emitir más llamadas en esta conversación hasta que haya recibido esta notificación. De lo contrario, la aplicación puede emitir Wait_For_Conversation para esperar a que finalice la llamada sin bloqueo o bien Cancel_Conversation para cancelar la llamada pendiente y finalizar la conversación.

CM_OPERATION_NOT_ACCEPTED
No puede emitirse la llamada debido a una de las condiciones siguientes:

- Hay una llamada sin bloqueo pendiente en esta conversación. El programa puede llevar a cabo otro proceso que no esté relacionado con esta conversación (como por ejemplo emitir llamadas CPI-C en otras conversaciones), pero no puede emitir la mayor parte de las llamadas CPI-C en esta conversación.

AIX, LINUX

Posteriormente, la aplicación puede emitir Check_For_Completion para determinar si ha finalizado una llamada sin bloqueo pendiente, Wait_For_Conversation para esperar a que finalice o Cancel_Conversation para cancelar la llamada pendiente y finalizar la conversación.

- El programa se ejecuta en un entorno de múltiples subprocesos DCE y hay una llamada pendiente en esta conversación de otro subproceso del programa. No puede haber más de una llamada pendiente simultáneamente para cada conversación.

WINDOWS

Si la aplicación ha utilizado Specify_Windows_Handle para recibir la notificación de terminación de llamada asíncrona, no debe emitir más llamadas en esta conversación hasta que haya recibido esta notificación. De lo contrario, la aplicación puede emitir Wait_For_Conversation para esperar a que finalice la llamada sin bloqueo o bien Cancel_Conversation para cancelar la llamada pendiente y finalizar la conversación.

CM_PARAMETER_ERROR
Un parámetro al que ha hecho referencia CPI-C no es válido. El parámetro no válido es un parámetro que puede suministrar el programa u otro componente ajeno al control del programa (como por ejemplo el archivo de configuración). Por ejemplo, el parámetro *mode_name* puede haber sido especificado por el programa mediante Set_Mode_Name o puede haberse obtenido de la entrada de información complementaria especificada por el parámetro *sym_dest_name*.

CM_PRODUCT_SPECIFIC_ERROR
Cuando CPI-C genera un código de retorno CM_PRODUCT_SPECIFIC_ERROR, crea una entrada en el archivo de anotaciones que indica el motivo del error y si es preciso efectuar alguna acción. Para más información sobre cómo interpretar estos mensajes, consulte la publicación *Communications Server para Linux, Guía de administración*.

CM_PROGRAM_ERROR_NO_TRUNC
El programa asociado ha emitido la llamada Send_Error mientras se encontraba en estado Enviar o Enviar-Pendiente con la dirección de error definida en CM_SEND_ERROR. Los datos no se han truncado.

CM_PROGRAM_ERROR_PURGING
Se ha producido una de las condiciones siguientes:

- El programa asociado ha emitido la llamada Send_Error mientras se encontraba en estado Recibir o Confirmar. Los datos enviados que aún no se han recibido se eliminan.
- El programa asociado ha emitido la llamada Send_Error mientras se encontraba en estado Enviar-Pendiente con la dirección de error definida en CM_RECEIVE_ERROR. Los datos no se han eliminado.

CM_PROGRAM_ERROR_TRUNC
El programa asociado de una conversación básica ha emitido una llamada Send_Error mientras se encontraba en estado Enviar, antes de terminar de enviar un registro lógico completo. El programa local puede haber recibido la primera parte del registro lógico a través de una llamada Receive.

CM_PROGRAM_PARAMETER_CHECK
El programa ha proporcionado un parámetro que no es válido para la llamada. Para ver información detallada al respecto, consulte las llamadas individuales en el Capítulo 3, "Llamadas CPI-C", en la página 49.

CM_PROGRAM_STATE_CHECK
La llamada emitida no está permitida en el estado actual de la conversación o no es adecuada debido al valor actual de una característica de conversación. Para ver información detallada al respecto, consulte las llamadas individuales en el Capítulo 3, "Llamadas CPI-C", en la página 49.

CM_RESOURCE_FAILURE_NO_RETRY
Se ha producido una de las condiciones siguientes:

- La conversación se ha finalizado de forma prematura a causa de una condición permanente. No vuelva a intentarlo hasta que se haya corregido el error.
- El programa asociado no ha desasignado la conversación antes de terminar normalmente.

CM_RESOURCE_FAILURE_RETRY
La conversación se ha finalizado de forma prematura a causa de una condición temporal, como una anomalía del módem. Vuelva a intentar la conversación.

CM_SECURITY_NOT_VALID
La LU asociada no acepta el identificador de usuario o la contraseña que se ha especificado en la petición de asignación.

CM_SYNC_LVL_NOT_SUPPORTED_PGM
El programa asociado no da soporte al nivel de sincronización especificado en la petición de asignación.

CM_SYNC_LVL_NOT_SUPPORTED_LU
La LU asociada no da soporte al nivel de sincronización especificado en la petición de asignación.

CM_TP_NOT_AVAILABLE_NO_RETRY
La LU asociada no puede iniciar el programa especificado en la petición de asignación debido a una condición permanente. El motivo del error puede estar registrado en el nodo remoto. No vuelva a intentar la asignación hasta que se haya corregido el motivo del error.

CM_TP_NOT_AVAILABLE_RETRY
La LU asociada no puede iniciar el programa especificado en la petición de asignación debido a una condición temporal. El motivo del error puede estar registrado en el nodo remoto. Vuelva a intentar la asignación.

CM_TPN_NOT_RECOGNIZED
La LU asociada no reconoce el nombre de programa especificado en la petición de asignación.

CM_UNSUCCESSFUL
La llamada no se ha ejecutado correctamente. Este código de retorno se produce en los casos siguientes:

- El programa ha emitido Allocate con el parámetro *return_control* establecido en `CM_IMMEDIATE` y Communications Server for Linux no ha podido asignar una versión para la conversación de inmediato.
- El programa ha emitido Receive con el parámetro *receive_type* definido en `CM_RECEIVE_IMMEDIATE` y actualmente no hay datos ni información de control disponible del programa asociado.

AIX, LINUX

- El programa ha emitido Check_For_Completion y no ha finalizado ninguna función sin bloqueo pendiente en ninguna de las conversaciones del programa.

Programa asociado de LU 6.2 no CPI-C

Cuando el programa asociado es un programa de LU 6.2 que no es CPI-C (por ejemplo, un TP APPC) pueden producirse los códigos de retorno siguientes. Los verbos descritos en estos párrafos son verbos de LU 6.2.

CM_DEALLOCATED_ABEND_SVC
La conversación se ha desasignado por uno de los motivos siguientes:

- El programa asociado ha emitido el verbo DEALLOCATE con *TYPE* definido en ABEND_SVC.
- El programa asociado no ha desasignado la conversación antes de terminar.

Si la conversación se encuentra en estado Recibir para el programa asociado cuando el programa local emite esta llamada, se eliminan los datos enviados por el programa local que el programa asociado todavía no ha recibido.

CM_DEALLOCATED_ABEND_TIMER
La conversación se ha desasignado porque el programa asociado ha emitido el verbo DEALLOCATE con *TYPE* definido en ABEND_TIMER. Si la conversación se encuentra en estado Recibir para el programa asociado cuando el programa local emite esta llamada, se eliminan los datos enviados por el programa local que el programa asociado todavía no ha recibido.

CM_PIP_NOT_SPECIFIED_CORRECTLY
Un programa de LU 6.2 no CPI-C ha rechazado la petición de asignación. El programa asociado requiere una o más variables de datos PIP, y CPI-C no proporciona soporte para los datos PIP.

CM_SVC_ERROR_NO_TRUNC
El programa asociado (o la LU asociada) ha emitido un verbo SEND_ERROR con el parámetro *TYPE* definido en SVC durante una conversación básica mientras se encontraba en estado Enviar. Los datos no se han truncado.

CM_SVC_ERROR_PURGING
Mientras se encontraba en estado Enviar, el programa asociado (o la LU asociada) ha emitido un verbo SEND_ERROR con *TYPE* definido en SVC. Los datos enviados al programa asociado pueden haber sido eliminados.

CM_SVC_ERROR_TRUNC
El programa asociado (o la LU asociada) de una conversación básica ha emitido un verbo SEND_ERROR con el parámetro *TYPE* definido en SVC mientras se encontraba en estado Recibir o Confirmar, antes de terminar de enviar un registro lógico completo. El programa local puede haber recibido la primera parte del registro lógico.

Apéndice C. Cambios de estado de conversación

La Tabla 25 en la página 184 muestra los estados de conversación en los que puede emitirse cada una de las llamadas CPI-C y el cambio de estado que se produce cuando la llamada finaliza.

En algunos casos, el cambio de estado depende del código de retorno de la llamada; en la mayoría de los casos, no se produce ningún cambio de estado para los códigos de retorno de ejecución incorrecta. En los casos en que no se muestra ningún código de retorno, el código de retorno CM_OK produce el cambio de estado indicado y los códigos de retorno de ejecución incorrecta no producen ningún cambio de estado (salvo en los casos descritos en la nota que figura a continuación). Cuando los cambios de estado varían según el código de retorno, los valores aplicables se indican en la columna de códigos de retorno.

Los estados de conversación posibles se muestran como cabeceras de columna. Para cada llamada se facilita la información siguiente bajo cada una de las cabeceras para indicar el resultado de emitir la llamada en ese estado:

X La llamada no puede emitirse en este estado.

T, I, II, S, SP, R, C, CS, CD o PP
Indica el estado de la conversación una vez finalizada la llamada: Restablecer (R), Inicializar (I), Inicializar-Entrante (II), Enviar (S), Enviar-Pendiente (SP), Recibir (R), Confirmar (C), Confirmar-Enviar (CS), Confirmar-Desasignar (CD), o Pendiente-Anotar (PP).

(blanco)
El código de retorno mostrado no puede se producir en este estado.

Vea función
Consulte la descripción de esta función en el Capítulo 3, “Llamadas CPI-C”, en la página 49. Los cambios del estado de conversación dependen de los parámetros devueltos de la llamada.

Nota: La conversación siempre entrará en estado Restablecer si se recibe alguno de los códigos de retorno siguientes:

- CM_ALLOCATION_FAILURE_NO_RETRY, CM_ALLOCATION_FAILURE_RETRY
- CM_CONVERSATION_TYPE_MISMATCH
- CM_DEALLOCATED_NORMAL, CM_DEALLOCATED_ABEND
- CM_PIP_NOT_SPECIFIED_CORRECTLY
- CM_RESOURCE_FAILURE_RETRY, CM_RESOURCE_FAILURE_NO_RETRY
- CM_SECURITY_NOT_VALID, CM_SYNC_LVL_NOT_SUPPORTED_PGM, CM_SYNC_LVL_NOT_SUPPORTED_LU
- CM_TPN_NOT_RECOGNIZED, CM_TP_NOT_AVAILABLE_RETRY, CM_TP_NOT_AVAILABLE_NO_RETRY

AIX, LINUX

El estado Pendiente-Anotar no se aplica a los sistemas AIX o Linux. Deberán ignorarse todas las referencias a este estado.

Cambios de estado de conversación

WINDOWS

El estado Inicializar-Entrante no se aplica a los sistemas Windows. Deberán ignorarse todas las referencias a este estado.

Las llamadas de función específicas de Windows no están asociadas a una conversación específica y no tienen efecto en los estados de conversación. No aparecen listadas en este apéndice.

Tabla 25. Cambios de estado de conversación

Llamada CPI-C y valores de *primary_rc*	**Estado al emitirse**									
	Resta-blecer (T)	**Inic. (I)**	**Inic. Ent. (II)**	**Enviar (S)**	**Enviar Pend. (SP)**	**Rec. (R)**	**Conf. (C)**	**Conf. Send (CS)**	**Conf. Des. (CD)**	**Pend. Anot. (PP)**
Accept_Conversation	R	X	X	X	X	X	X	X	X	X
Accept_Incoming	X	X	R	X	X	X	X	X	X	X
Allocate	X		X	X	X	X	X	X	X	X
CM_OK		S								
(Anomalía de Allocate)		T								
Cancel_Conversation	X	T	T	T	T	T	T	T	T	T
Check_For_Completion	T	I	IE	S	SP	R	C	CS	CD	X
Confirm	X	X	X			X	X	X	X	X
CM_OK				S	S					
(Error de programa,error de SVC)				R	R					
Confirmed	X	X	X	X	X	X	R	S	T	X
Convert_Incoming,Convert_Outgoing	T	I	IE	S	SP	R	C	CS	CD	X
Deallocate (finaliz. anormal)	X									
CM_OK		T	T	T	T	T	T	T	T	T
(Error de programa,error de SVC)		R	R	R	R	R	R	R	R	R
Deallocate (otros)	X	X	X			X	X	X	X	X
CM_OK				T	T					
(Error de programa,error de SVC)				R	R					
Delete_CPIC_Side_Information	T	I	IE	S	SP	R	C	CS	CD	X
Extract_Conversation_Context	X	X	X	S	SP	R	C	CS	CD	X
Extract_Conversation_Security_Type	X	I	IE	S	SP	R	C	CS	CD	X
Extract_Conversation_State	X	I	IE	S	SP	R	C	CS	CD	X
Extract_Conversation_Type	X	I	IE	S	SP	R	C	CS	CD	X
Extract_CPIC_Side_Information	T	I	IE	S	SP	R	C	CS	CD	X
Extract_Local_LU_Name	X	I	IE	S	SP	R	C	CS	CD	X
Extract_Maximum_Buffer_Size	T	I	IE	S	SP	R	C	CS	CD	X
Extract_Mode_Name	X	I	IE	S	SP	R	C	CS	CD	X
Extract_Partner_LU_Name	X	I	X	S	SP	R	C	CS	CD	X
Extract_Security_User_ID	X	I	IE	S	SP	R	C	CS	CD	X
Extract_Sync_Level	X	I	X	S	SP	R	C	CS	CD	X
Extract_TP_Name	X	I	X	S	SP	R	C	CS	CD	X
Flush	X	X	X	S	S	X	X	X	X	X
Initialize_Conversation	I	X	X	X	X	X	X	X	X	X
Initialize_For_Incoming	IE	X	X	X	X	X	X	X	X	X

Tabla 25. Cambios de estado de conversación (continuación)

Llamada CPI-C y valores de *primary_rc*	**Estado al emitirse**									
	Resta-blecer (T)	**Inic. (I)**	**Inic. Ent. (II)**	**Enviar (S)**	**Enviar Pend. (SP)**	**Rec. (R)**	**Conf. (C)**	**Conf. Send (CS)**	**Conf. Des. (CD)**	**Pend. Anot. (PP)**
Prepare_To_Receive	X	X	X			X	X	X	X	X
CM_OK, error del programa, error de SVC				R	R					
Receive (tipo de recepción CM_RECEIVE_IMMEDIATE)	X	X	X	X	X	Vea función	X	X	X	X
Receive (tipo de recepción CM_RECEIVE_AND_WAIT)	X	X	X	Vea función	Vea función	Vea función	X	X	X	X
Release_Local_TP_Name	I	X	X	X	X	X	X	X	X	X
Request_To_Send	X	X	X	S	SP	R	C	CS	CD	PE
Send_Data	X	X	X	Vea función	Vea función	X	X	X	X	X
Send_Error	X	X	X							
CM_OK				S	S	S	S	S	S	S
Error del programa, error de SVC				R	R	R	R	R	R	PE
Set_Conversation_Context	X	I	IE	S	SP	R	C	CS	CD	X
Set_Conversation_Security_Password	X	I	X	X	X	X	X	X	X	X
Set_Conversation_Security_Type	X	I	X	X	X	X	X	X	X	X
Set_Conversation_Security_User_ID	X	I	X	X	X	X	X	X	X	X
Set_Conversation_Type	X	I	X	X	X	X	X	X	X	X
Set_CPIC_Side_Information	T	I	IE	S	SP	R	C	CS	CD	X
Set_Deallocate_Type	X	I	IE	S	SP	R	C	CS	CD	X
Set_Error_Direction	X	I	IE	S	SP	R	C	CS	CD	X
Set_Fill	X	I	IE	S	SP	R	C	CS	CD	X
Set_Local_LU_Name	X	I	X	X	X	X	X	X	X	X
Set_Log_Data	X	I	IE	S	SP	R	C	CS	CD	X
Set_Mode_Name	X	I	X	X	X	X	X	X	X	X
Set_Partner_LU_Name	X	I	X	X	X	X	X	X	X	X
Set_Prepare_To_Receive_Type	X	I	IE	S	SP	R	C	CS	CD	X
Set_Processing_Mode	X	I	IE	S	SP	R	C	CS	CD	X
Set_Receive_Type	X	I	IE	S	SP	R	C	CS	CD	X
Set_Return_Control	X	I	X	X	X	X	X	X	X	X
Set_Send_Type	X	I	IE	S	SP	R	C	CS	CD	X
Set_Sync_Level	X	I	X	X	X	X	X	X	X	X
Set_TP_Name	X	I	X	X	X	X	X	X	X	X
Specify_Local_TP_Name	T	I	IE	S	SP	R	C	CS	CD	X
Test_Request_To_Send_Received	X	X	X	S	SP	R	X	X	X	PE
Wait_For_Conversation	Puede emitirse en cualquier estado; el nuevo estado depende de la llamada pendiente que ha finalizado y del código de retorno de esa llamada. Consulte la información de la llamada correspondiente.									

Cambios de estado de conversación

Apéndice D. 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. Consulte al representante local de IBM para obtener información sobre los productos y servicios que se pueden adquirir actualmente en su zona geográfica. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que solamente se pueda utilizar ese producto, programa o servicio de IBM. En su lugar, puede utilizarse cualquier producto, programa o servicio de funciones equivalentes que no infrinja los derechos de propiedad 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 patentes en tramitación sobre los temas descritos en este documento. El suministro de este documento no le otorga ninguna licencia sobre esas patentes. Puede enviar solicitudes de información sobre licencias, por escrito, a esta dirección:

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
Tokio 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 realizan cambios en la información aquí contenida; estos cambios se incorporarán a las 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.

Cualquier referencia en esta información a sitios Web que no sean de IBM se ofrece para su comodidad y no supone una recomendación de dichos sitios Web. Los

materiales de dichos sitios Web no forman parte de los materiales correspondientes a este producto de IBM y el uso de dichos sitios Web se realiza bajo la responsabilidad del usuario.

IBM puede utilizar o distribuir cualquier información que el usuario le proporcione de la manera que IBM considere apropiada, sin contraer ninguna obligación con el usuario.

Los licenciatarios de este programa que deseen tener información sobre él con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido el presente programa) y (ii) el uso recíproco de la información que se ha intercambiado, 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 apropiados, incluido en algunos casos, el pago de una tarifa.

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 sobre rendimiento aquí contenidos se han determinado en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Es posible que algunas mediciones se hayan tomado en sistemas a nivel de desarrollo y no se garantiza que se obtengan los mismos resultados en sistemas disponibles a nivel general. Además, puede que algunas mediciones se hayan estimado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos pertinentes correspondientes a su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha comprobado dichos productos y no puede confirmar la precisión de rendimiento, la compatibilidad y otras afirmaciones relacionadas con productos no IBM. Las preguntas sobre las funciones de productos que no son de IBM deben enviarse a los proveedores de dichos productos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de particulares, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con los nombres y direcciones que se utilizan en una empresa comercial real son pura coincidencia.

LICENCIA DE COPYRIGHT: Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que muestran técnicas de programación en varias plataformas operativas. El cliente puede copiar, modificar y distribuir estos programas de ejemplo en la forma que crea conveniente sin pagar a IBM, con el objeto de desarrollar, utilizar, comercializar o distribuir programas de aplicación que son compatibles con la interfaz de programación de aplicaciones para la

plataforma operativa para la cual están escritos los programas de ejemplo. Estos ejemplos no se han probado de forma exhaustiva en todas las condiciones. Por consiguiente, IBM no puede garantizar ni presuponer la fiabilidad, el servicio o la función de estos programas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier forma sin pago alguno a IBM, con el fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación de acuerdo con las interfaces de programación de aplicaciones de IBM.

Cada copia o cualquier porción de estos programas de ejemplo o cualquier trabajo derivado debe incluir un aviso de copyright, de la manera siguiente: ® (nombre de su empresa) (año). Partes de este código proceden de programas de ejemplo de IBM Corp. ® Copyright IBM Corp. 2000, 2005, 2006. Todos los derechos reservados.

Marcas registradas

Los términos siguientes son marcas registradas de IBM Corporation en Estados Unidos y/o en otros países:

Advanced Peer-to-Peer Networking	Power5
AIX	pSeries
Application System/400	S/390
AS/400	SP
CICS	System/370
IBM	System/390
MQSeries	SAA
MVS	Systems Application Architecture
MVS/ESA	VTAM
MVS/XA	WebSphere
NetView	z/OS
OpenPower	z9
OS/2	zSeries

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 los Estados Unidos y/o en otros países.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Intel Centrino, el logotipo de Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus subsidiarias en los Estados Unidos y en otros países.

UNIX es una marca registrada de The Open Group en los Estados Unidos y en otros países.

Linux es una marca registrada de Linus Torvalds en los 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 que se describen 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

Para los manuales en la biblioteca de Communications Server para Linux, se proporcionan breves descripciones. Para los demás manuales, sólo se muestra el título, el número de pedido y, en algunos casos, el título abreviado que se utiliza en el texto del presente manual.

Publicaciones de Communications Server para Linux Versión 6.2.2

La biblioteca de Communications Server para Linux contiene los manuales siguientes. Además, se proporcionan versiones en copia software de estos documentos en el CD-ROM. Para obtener información sobre cómo acceder a los archivos en copia software del CD-ROM, consulte la publicación *IBM Communications Server para Linux, Guía de iniciación rápida*. Para instalar en el sistema estos manuales en copia software, necesitará 9–15 MB de espacio de disco duro (según las versiones de idioma que instale).

- *IBM Communications Server para Linux, Guía de iniciación rápida* (GC10-9852-01)

  Este manual ofrece una introducción general a Communications Server para Linux, incluyendo información sobre las características de red soportadas, la instalación, la configuración y el funcionamiento.
- *IBM Communications Server para Linux, Guía de administración* (SC10-9853-01)

  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 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 del lenguaje “C” o Java acerca de cómo desarrollar programas de transacciones SNA utilizando la API de CPI de Communications Server para Linux.
- *IBM Communications Server para Linux, Guía del programador para APPC* (SC10-9854-01)

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* (SC10-9855-01)

  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* (SC31-6770-02)

  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* (SC11-3348-00)

  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 aplicaciones APPC que se utilizan con Communications Server para Linux.
- *IBM Communications Server for Linux, Glossary* (GC31-6780-02)

  Este manual proporciona una lista completa de términos y definiciones utilizados 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 de z/OS Communications Server

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 de 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 sobre 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

Accept_Conversation 51
Accept_Incoming 53
Allocate, llamada 55
aplicaciones AIX
  compilación y enlace 39
aplicaciones Linux
  compilación y enlace 39
ASCII-EBCDIC, conversión de datos 23
asignación de sesión, esperar 148
asignación inmediata de una conversación 57
asignar una conversación
  confirmar la asignación 57
  errores 57
  utilizando la llamada Allocate 55

C

cambios de estado 183
Cancel_Conversation 58
característica de relleno de la conversación 134
características de conversación
  asociadas a un nombre de destino simbólico 126
  consideraciones sobre Allocate 55
  establecer con Accept_Conversation 51
  establecer con Accept_Incoming 53
  valores iniciales 20, 92, 94
Check_For_Completion 59
CM_ALLOCATION_FAILURE_NO_RETRY 177
CM_ALLOCATION_FAILURE_RETRY 177
CM_CALL_NOT_SUPPORTED 177
CM_CONVERSATION_TYPE_MISMATCH 177
CM_DEALLOCATED_ABEND 177
CM_DEALLOCATED_ABEND_SVC 181
CM_DEALLOCATED_ABEND_TIMER 181
CM_DEALLOCATED_NORMAL 178
CM_OK 178
CM_OPERATION_INCOMPLETE 178
CM_OPERATION_NOT_ACCEPTED 178
CM_PARAMETER_ERROR 179
CM_PIP_NOT_SPECIFIED_CORRECTLY 181
CM_PRODUCT_SPECIFIC_ERROR 179
CM_PROGRAM_ERROR_NO_TRUNC 179
CM_PROGRAM_ERROR_PURGING 179
CM_PROGRAM_ERROR_TRUNC 179
CM_PROGRAM_PARAMETER_CHECK 180
CM_PROGRAM_STATE_CHECK 180
CM_RESOURCE_FAILURE_NO_RETRY 180
CM_RESOURCE_FAILURE_RETRY 180
CM_SECURITY_NOT_VALID 180
CM_SVC_ERROR_NO_TRUNC 181
CM_SVC_ERROR_PURGING 181
CM_SVC_ERROR_TRUNC 181
CM_SYNC_LVL_NOT_SUPPORTED_LU 180
CM_SYNC_LVL_NOT_SUPPORTED_PGM 180
CM_TP_NOT_AVAILABLE_NO_RETRY 180
CM_TP_NOT_AVAILABLE_RETRY 180
CM_TPN_NOT_RECOGNIZED 180
CM_UNSUCCESSFUL 180
códigos de retorno 175
códigos de retorno comunes 177
códigos de retorno de error 177
compilación de aplicaciones AIX 39
compilación de aplicaciones Linux 39
compilación y enlace 46
comunicaciones entre TP 2
con cola, programa iniciado automáticamente 37
con cola, programa iniciado por el operador 38
Confirm, llamada 61
Confirmar, estado 8
Confirmar-Desasignar, estado 8
Confirmar-Enviar, estado 8
confirmed 63
consideraciones sobre Windows 43
constantes simbólicas 49
contexto 73, 117
contienda, ganadoras y perdedoras 3
contraseña, seguridad de conversación 119
control de retorno 148
conversación
  asignación 4
  básica 4
  contienda 3
  correlacionada 4
  desasignación 4, 26
  desasignar 7, 69
  estado 7
  finalización 6, 26
  inicio 5
  nivel de sincronización 7
  seguridad 13
  visión de la conversación por parte del TP 9
conversación básica
  características de 12
  tipos 4
conversación correlacionada 4, 125
conversaciones múltiples 13
conversión (EBCDIC-ASCII) 105, 112
conversión de datos entre ASCII y EBCDIC 23
conversión entre ASCII y EBCDIC 112
conversión entre EBCDIC y ASCII 105
Convert_Incoming, llamada 66
Convert_Outgoing, llamada 67
CPI-C de Java
  clases 40
  compilar y enlazar una aplicación 42, 173
  constantes 40
  desarrollar programas 40
  ejecutar una aplicación 42
  ejemplo de uso 41
  tipos de parámetros 40

D

datos
  envío 5
  recibir 6
datos, recibir 97
datos de anotaciones 71, 116, 137
datos de anotaciones de error 13, 71, 116, 137
Deallocate, llamada 69

Delete_CPIC_Side_Information 71
desasignar, recibir notificación del programa asociado 69
desasignar una conversación 69
dirección de error 133

E

EBCDIC-ASCII, conversión de datos 23
enlace de aplicaciones AIX 39
enlace de aplicaciones Linux 39
Enviar, estado
  cambio a 11
  definición 8
enviar datos 91
  utilizando la llamada Request_To_Send 107
  utilizando la llamada Send_Data 109
Enviar-Pendiente, estado 8
envío de datos
  llamadas utilizadas para 21
  utilización de la llamada Send_Data 5
errores, informar 112
esperar asignación de sesión 148
estado de conversación
  cambio 9
  cambios 9, 183
  descripción 7
  inicial 10
  obtener 76
estado de una conversación 76
estado Pendiente-Anotar, Windows 8
Extract_Conversation_Context 73
Extract_Conversation_Security_Type 74
Extract_Conversation_Security_User_ID 76
Extract_Conversation_State 76
Extract_Conversation_Type 78
Extract_CPIC_Side_Information 79
Extract_Local_LU_Name 82
Extract_Maximum_Buffer_Size 83
Extract_Mode_Name 84
Extract_Partner_LU_Name 85
Extract_Security_User_ID 86
Extract_Sync_Level 88
Extract_TP_Name 89

F

Flush 91
funcionamiento sin bloqueo 15

I

identificador de conversación 51, 92, 94
identificador de usuario, seguridad de conversación 81, 86, 123
información complementaria 71, 79, 126
información de configuración 34, 72, 126, 130
informar de errores 112
Inicializar, estado 7
Inicializar-Entrante, estado 8
Initialize_Conversation 92
Initialize_For_Incoming 94
interfaz de programación de aplicaciones 1
llamada WinCPICCleanup 162
llamada WinCPICIsBlocking 162
llamada WinCPICStartup 164
llamadas CPI-C
  resumidas por función 19
  visión general 4
llamadas de bloqueo,Windows 44
llamadas de función para CPI-C, específicas de Windows 43

L

LU asociada 4
LU asociada, nombre 85, 141
LU local 4
LU remota 4

M

mensajes de error 179
modalidad 3
modalidad, nombre 84, 139
modalidad con bloqueo 15
modalidad sin bloqueo 16

N

nivel de sincronización
  establecer 151
  establecimiento 7
  y Extract_Sync_Level 88
nombre de destino simbólico 31, 71, 126
nombre de LU asociada 85
nombre de TP 89
notificación de petición de envío
  consultar 158
  en la llamada Request_To_Send 109

P

petición de confirmación
  envío 7
  recibir 7, 65
  respuesta a 7, 63
  y llamada Confirm 61
Prepare_To_Receive 95
proceso, modalidad 145
proceso de confirmación 6
proceso de transacciones distribuido 1
procesos múltiples 39
programa asociado, nombre 153
programa CPI-C de Java de ejemplo 173
programa invocado
  con cola, iniciado automáticamente 37
  con cola, iniciado por el operador 38
  sin cola, iniciado automáticamente 38
  starting 37
programa que invoca, iniciar 37
programas de ejemplo
  pseudocódigo 169
  visión general 169
programas de transacciones (TP)
  TP asociado 4
  TP invocado 4
  TP local 4
  TP que invoca 4
  TP remoto 4
Publicaciones de (Arquitectura de red de sistemas) xxi

R

recepción de datos
  con la llamada Receive 6
  llamadas que permiten 22
Recibir 97
Recibir, estado
  cambiar a 95
  cambio a 10
  definición 8
recibir datos
  con la llamada Receive 97
  esperar datos 147
registro de datos 5, 134
registros lógicos 12, 134
Release_Local_TP_Name 106
Request_To_Send 107
Restablecer, estado 7

S

seguridad de conversación
  contraseña 119
  identificador de usuario 123
  tipo 121, 123
  visión general 13
Send_Data 109
Send_Error 112
servicio, TP 4
sesiones, LU-LU 3
sesiones LU-LU 3
sesiones múltiples 3
sesiones paralelas 3
Set_Conversation_Context 117
Set_Conversation_Security_Password 118, 120
Set_Conversation_Security_Type 121, 123
Set_Conversation_Security_User_ID 123, 125
Set_Conversation_Type 125
Set_CPIC_Side_Information 126
Set_Deallocate_Type 130
Set_Error_Direction 133
Set_Fill 134
Set_Local_LU_Name 136
Set_Log_Data 137
Set_Mode_Name 139
Set_Partner_LU_Name 141
Set_Prepare_To_Receive_Type 143
Set_Processing_Mode 145
Set_Receive_Type 147
Set_Return_Control 148
Set_Send_Type 150
Set_Sync_Level 151
Set_TP_Name 153
sin cola, programa iniciado automáticamente 38
sincronización con el programa asociado 63
Specify_Local_TP_Name 155
Specify_Windows_Handle 157

T

tamaño del almacenamiento intermedio 83
tamaño del almacenamiento intermedio de datos 83
tamaño máximo del almacenamiento intermedio 83
Test_Request_to_Send_Received 158
tipo de conversación
  básica 4
  con la llamada Allocate 55
tipo de conversación *(continuación)*
  con la llamada Extract_Conversation_Type 78
  correlacionada 4
  establecer 125
tipo de desasignación 70, 130
tipo de envío 150
tipo de preparación para recepción 143
tipo de recepción 147
tipo de seguridad 74
tipos de datos 49
TP, comunicaciones 2
TP asociado 4
TP asociado, nombre 153
TP de aplicación 4
TP invocado 4
TP local 4
TP que invoca 4
TP remoto 4

U

unidad lógica (LU)
  LU 6.2 xxi, 3
  LU asociada 4
  LU local 4
  LU remota 4

V

vaciado del almacenamiento intermedio de envío de LU local 5
vaciar almacenamiento intermedio de envío de LU local 91

W

Wait_For_Conversation 159

Número de Programa: 5724-i33, 5724-i34

SC10-9861-01