UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T

X.446

(08/97)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE X: REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

Sistemas de tratamiento de mensajes

Interfaz de programas de aplicación para llamadas de mensajería común

Recomendación UIT-T X.446

(Anteriormente Recomendación del CCITT)

RECOMENDACIONES DE LA SERIE X DEL UIT-T

REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

REDES PÚBLICAS DE DATOS	X.1–X.199
Servicios y facilidades	X.1–X.19
Interfaces	X.20–X.49
Transmisión, señalización y conmutación	X.50–X.89
Aspectos de redes	X.90–X.149
Mantenimiento	X.150–X.179
Disposiciones administrativas	X.180–X.199
INTERCONEXIÓN DE SISTEMAS ABIERTOS	X.200–X.299
Modelo y notación	X.200–X.209
Definiciones de los servicios	X.210–X.219
Especificaciones de los protocolos en modo conexión	X.220–X.229
Especificaciones de los protocolos en modo sin conexión	X.230–X.239
Formularios para declaraciones de conformidad de implementación de protocolo	X.240–X.259
Identificación de protocolos	X.260–X.269
Protocolos de seguridad	X.270–X.279
Objetos gestionados de capa	X.280–X.289
Pruebas de conformidad	X.290–X.299
INTERFUNCIONAMIENTO ENTRE REDES	X.300–X.399
Generalidades	X.300–X.349
Sistemas de transmisión de datos por satélite	X.350–X.399
SISTEMAS DE TRATAMIENTO DE MENSAJES	**X.400–X.499**
DIRECTORIO	X.500–X.599
GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS	X.600–X.699
Gestión de redes	X.600–X.629
Eficacia	X.630–X.649
Denominación, direccionamiento y registro	X.650–X.679
Notación de sintaxis abstracta uno	X.680–X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	X.700–X.799
Marco y arquitectura de la gestión de sistemas	X.700–X.709
Servicio y protocolo de comunicación de gestión	X.710–X.719
Estructura de la información de gestión	X.720–X.729
Funciones de gestión	X.730–X.799
SEGURIDAD	X.800–X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	X.850–X.899
Cometimiento, concurrencia y recuperación	X.850–X.859
Tratamiento de transacciones	X.860–X.879
Operaciones a distancia	X.880–X.899
PROCESAMIENTO DISTRIBUIDO ABIERTO	X.900–X.999

Para más información, véase la Lista de Recomendaciones del UIT-T.

RECOMENDACIÓN UIT-T X.446

INTERFAZ DE PROGRAMAS DE APLICACIÓN PARA LLAMADAS DE MENSAJERÍA COMÚN

Resumen

Esta Recomendación especifica una interfaz de llamada simple a través de la cual las aplicaciones basadas en mensajería pueden invocar los servicios del sistema de tratamiento de mensajes (MHS) mediante una interfaz de programación normalizada. Esta Recomendación se elaboró en cooperación con la Asociación XAPI y define la interfaz de programación de aplicaciones que está siendo implementada, para el MHS, por los vendedores y proveedores de servicios más importantes en el mundo.

Orígenes

La Recomendación UIT-T X.446 ha sido preparada por la Comisión de Estudio 7 (1997-2000) del UIT-T y fue aprobada por el procedimiento de la Resolución N.° 1 de la CMNT el 9 de agosto de 1997.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución N.° 1 de la CMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT ha recibido/no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB.

ÍNDICE

Página

Página

INTERFAZ DE PROGRAMAS DE APLICACIÓN PARA LLAMADAS DE MENSAJERÍA COMÚN

(Ginebra, 1997)

1 Introducción

Esta cláusula introduce la interfaz de programas de aplicación (API, *application program interface*) para llamadas de mensajería común (CMC, *common messaging call*) y sus especificaciones. Expresa la finalidad de la interfaz, proporciona una visión de conjunto del mismo, indica las abreviaturas utilizadas, proporciona referencias a documentos, explica el nivel de abstracción de la interfaz, define convenios de denominación en el lenguaje de programación C y especifica requisitos de conformidad.

Esta Recomendación es un perfeccionamiento de la primera versión de API para CMC (brevemente CMC API) publicada en junio de 1993 por la Asociación API X.400 (o XAPIA). Esta Recomendación amplía el soporte de las aplicaciones conscientes de la mensajería (*messaging-aware application*) en el documento original con el soporte de aplicaciones basadas en la mensajería (*messaging-reliant application*).

1.1 Finalidad

Esta Recomendación tiene por finalidad especificar una interfaz de programas de aplicación de mensajería de alto nivel que pueda ser soportada por la mayor parte de los servicios de mensajería empleados en la actualidad. Se tiene la intención de que la API permita a los programadores de aplicaciones integrar fácilmente la mensajería, y por tanto las comunicaciones, en sus aplicaciones, creando un amplio cuerpo de *aplicaciones habilitadas para mensajería* (*messaging-enabled applications*).

Esta Recomendación está dirigida a los desarrolladores de servicios de mensajería que pudieran desear soportar esa interfaz de programas de aplicación. Esta Recomendación puede también orientar a los desarrolladores de aplicaciones para la comprensión de aquellas características de la API para llamadas de mensajería común que son independientes de la aplicación. Los desarrolladores de aplicaciones tienen que seguir los manuales proporcionados por el sistema que están utilizando para el soporte de mensajería.

1.2 Visión de conjunto

La interfaz de programas de aplicación para llamadas de mensajería común proporciona un conjunto de funciones de alto nivel que serán utilizadas por aplicaciones habilitadas para mensajería para enviar y recibir mensajes electrónicos.

Dentro de la gama de las aplicaciones habilitadas para mensajería hay aplicaciones conscientes de la mensajería y aplicaciones basadas en la mensajería.

Las aplicaciones conscientes de la mensajería son aquellas que pueden funcionar satisfactoriamente como aplicaciones autónomas, pero que podrían conectarse a un servicio de mensajería para proporcionar una funcionalidad realzada. Un ejemplo sería una aplicación de tratamiento de texto o de hoja de cálculo con la capacidad de enviar el documento o fichero utilizando una opción de envío de fichero (FILE-SEND) que apareciera en el menú.

Las aplicaciones basadas en la mensajería son aquellas que dependen intrínsecamente de la existencia de un servicio de mensajería para ejercer su funcionalidad. Son ejemplos de estas aplicaciones las de intercambio electrónico de datos (EDI, *electronic data exchange*), las aplicaciones de distribución de información, las aplicaciones de conferencias/colaboración, y posiblemente algunas bases de datos distribuidas.

Esta interfaz está concebida para que sea independiente del protocolo de mensajería real empleado entre el emisor y el recibiente. La interfaz soportará la creación y recepción de formatos de mensaje normalizados tales como X.400 y SMTP/MIME (RFC 822/RFC 1521) así como formatos de mensaje no normalizados. Esto se consigue mediante la definición genérica de capacidades comunes a la mayor parte de los protocolos de mensajería, más un mecanismo para definir extensiones, que pueden utilizarse para invocar servicios específicos de protocolo.

La interfaz está también concebida para que sea independiente del sistema operativo y del soporte físico subyacente utilizado por el servicio de mensajería.

Otra importante consideración en el diseño de esta API es la de permitir que se ejecuten acciones simples de aplicación con un número mínimo de llamadas a funciones, y que, al mismo tiempo, sea posible ejecutar acciones más complejas. Para alcanzar estos objetivos, a veces en pugna, la CMC API tiene dos interfaces: una interfaz CMC simple y una interfaz CMC completa. La interfaz CMC simple proporciona un número mínimo de llamadas a funciones, necesarias para que las aplicaciones conscientes de la mensajería envíen o reciban mensajes. La CMC completa proporciona un conjunto más amplio de llamadas a funciones en previsión de las aplicaciones basadas en la mensajería, que son más potentes.

La CMC API está concebida para que sea complementaria de las actuales API XAPIA-X/OPEN y de las API de XMHS y XMS.

La interfaz CMC está concebida para que permita una interconexión común a través de, prácticamente, cualquier servicio de mensajería electrónica. Para cada implementación de CMC, la visión/capacidades presentadas por la CMC tienen que hacerse corresponder con la visión/capacidades del servicio de mensajería subyacente.

Para maximizar la interoperabilidad entre aplicaciones CMC que utilizan servicios subyacentes similares de mensajería es de capital importancia que el segmento industrial que representa el protocolo o interfaz de mensajería en cuestión defina una correspondencia común.

Con ese fin:

- La presente Recomendación define la correspondencia común entre la CMC simple y el protocolo de memoria de mensaje X.400.
- Se invita a los organismos de normalización, vendedores, o grupos de vendedores que representen un protocolo o interfaz específico de mensajería, a que definan una correspondencia común entre la CMC y el protocolo o interfaz de mensajería en cuestión.

Para maximizar la interoperabilidad entre aplicaciones CMC que utilizan diferentes servicios de mensajería subyacentes es de capital importancia que las definiciones de las correspondencias se diseñen teniendo presente tal interoperabilidad.

Con ese fin se ofrecen las siguientes directrices. Siempre que sea posible, o apropiado:

- Las cadenas de texto de mensajes deben hacerse corresponder con juegos de caracteres internacionales.
- Los tipos de añadiduras de mensajes deben hacerse corresponder con los tipos de añadiduras comúnmente reconocidos.

Esta lista no es exhaustiva; se podrán ofrecer orientaciones adicionales una vez que se hayan situado las implementaciones.

1.3 Terminología

1.3.1 Definiciones

En esta Recomendación se definen los términos siguientes.

1.3.1.1 llamada de mensaje común completa: Una API habilitada para mensajería que proporciona las funciones para soportar aplicaciones basadas en la mensajería.

1.3.1.2 llamada de mensaje común simple: Una API habilitada para mensajería que proporciona las funciones para soportar aplicaciones conscientes de la mensajería.

1.3.1.3 T.611: Interfaz de programas de aplicación UIT-T para uso con servicios de facsímil télex y teletex.

1.3.2 Abreviaturas

En esta Recomendación se utilizan las siguientes siglas.

API	Interfaz de programas de aplicación (*application program interface*)
CMC	Llamada de mensajería común (*common messaging call*)
XAPIA	Asociación de interfaz de programas de aplicación X.400 (*X.400 application program interface Association*)
XMHS API	Interfaz de programas de aplicación X/OPEN con el correo electrónico (X.400) [*X/OPEN application program interface to electronic mail (X.400)*]

XMS API — Interfaz de programas de aplicación de memoria de mensajes X/OPEN (*X/OPEN message store application program interface*)

XOM API — Interfaz de programas de aplicación para la manipulación de datos abstractos OSI en X/OPEN (*X/OPEN OSI-abstract data manipulation API*)

UI — Interfaz de usuario (*user interface*)

1.4 Documentos a que se hace referencia

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes.

1.4.1 Recomendaciones | Normas Internacionales idénticas

– Recomendación UIT-T X.402 (1995) | ISO/CEI 10021-2:1996, *Tecnología de la información – Sistema de tratamiento de mensajes: Arquitectura global*.

– Recomendación UIT-T X.411 (1995) | ISO/CEI 10021-4:1997, *Tecnología de la información – Sistema de tratamiento de mensajes: Sistema de transferencia de mensajes: Definición del servicio abstracto y procedimientos*.

– Recomendación UIT-T X.413 (1995) | ISO/CEI 10021-5:1996, *Tecnología de la información – Sistemas de tratamiento de mensajes: Memoria de mensajes: Definición del servicio abstracto*.

– Recomendación UIT-T X.419 (1995) | ISO/CEI 10021-6:1996, *Tecnología de la información – Sistemas de tratamiento de mensajes: Especificaciones de protocolo*.

– Recomendación UIT-T X.420 (1996) | ISO/CEI 10021-7:1997, *Tecnología de la información – Sistemas de tratamiento de mensajes – Sistema de mensajería interpersonal*.

1.4.2 Pares de Recomendaciones | Normas Internacionales de contenido técnico equivalente

– Recomendación X.208 del CCITT (1988), *Especificación de la notación de sintaxis abstracta uno (ASN.1)*.

  ISO/CEI 8824:1990, *Information technology – Open Systems Interconnection – Specification of abstract syntax notation one (ASN.1)*.

– Recomendación X.209 del CCITT (1988), *Especificación de las reglas básicas de codificación de la notación de sintaxis abstracta uno (ASN.1)*.

  ISO/CEI 8825:1990, *Information technology – Open Systems Interconnection – Specification of abstract syntax notation one (ASN.1)*.

– Recomendación UIT-T X.400/F.400 (1996), *Visión de conjunto del sistema y del servicio de tratamiento de mensajes*.

  ISO/CEI 10021-1:1997, *Information technology – Text communications – Message-oriented text interchange systems (MOTIS) – Part 1: System and service overview*.

1.4.3 Referencias adicionales

– ISO 8601:1988, Data elements and interchange formats – Information interchange – Representations of dates and times.

– ISO 9070: 1991, Information Technology – SGML support facilities – Registration procedures for public text owner identifiers.

– ISO/IEC 10021-3:1990, *Information technology – Text Communication – Message-oriented Text Interchange Systems (MOTIS) – Part 3: Abstract Service Definition Conventions*.

– IMAP – "Internet Message Access Protocol", Version 4, RFC 1730, diciembre 1994.

– MIME – "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for specifying and Describing the format of Internet Message Bodies", RFC 1521, septiembre 1993.

– RFC 882 – "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 882, agosto 1982.

– SMTP – "Simple Mail Transfer Protocol", RFC 821, agosto 1982.

– XMHS API – API to Electronic Mail (X.400), CAE Specification, X/Open Company Limited and X.400 API Association, 1991.

– XMS API – Message Store API, Preliminary Specification, X/Open Company Limited and X.400 API Association, 1991.

– XOS API – OSI-Abstract-Data Manipulation, CAE Specification, X/Open Company Limited and X.400 API Association, 1991.

– ANSI C – American National Standard for Information Systems – Programming Language C, X.3.159-1989.

1.5 Niveles

Esta Recomendación define la CMC API en dos niveles de abstracción. Define una interfaz "genérica", independiente de cualquier lenguaje de programación particular, y una interfaz en lenguaje C, basada en la Norma nacional americana para el lenguaje de programación C (American National Standard for the C Programming Language). La interfaz "genérica" se incluye para orientar el desarrollo de otras especificaciones propias de otros lenguajes, por ejemplo PASCAL.

Para facilitar la lectura, la especificación de la interfaz genérica y la especificación de la interfaz en el lenguaje C se han combinado. En la cláusula 4, las estructuras de datos CMC se describen de manera general, pero incluyen una declaración en el lenguaje de programación C. En la cláusula 6, las funciones CMC se especifican de una manera general, pero incluyen una sinopsis escrita igualmente en lenguaje C. Para mayor claridad, las constantes y los códigos de error en esta Recomendación se han escrito siguiendo la sintaxis del lenguaje C descrita a continuación. En el anexo A se presenta una recapitulación de las declaraciones en lenguaje C y las constantes utilizadas en esta Recomendación.

1.6 Convenios de denominación

El modo de derivar un identificador para un elemento de la interfaz C a partir del nombre del elemento correspondiente de la interfaz genérica depende del tipo del elemento, especificado en el siguiente cuadro 1. El nombre genérico lleva como prefijo la cadena de caracteres indicada en la segunda columna del cuadro; los caracteres alfabéticos deberán escribirse o mayúsculas o minúsculas, según se indique en la tercera columna.

Cuadro 1/X.446 – Convenios para la derivación de nombres en lenguaje C

Tipo de elemento	Prefijo	Mayúscula/minúscula
Tipo de datos	CMC_	Minúsculas
Valor de datos	CMC_	Mayúsculas
Función	cmc_	Minúsculas
Argumento de función	ninguno	Minúsculas
Resultado de función	ninguno	Minúsculas
Constante	CMC_	Mayúsculas
Error	CMC_E_	Mayúsculas
Macro	CMC_	Mayúsculas
Clase de objeto	CMC_OC_	Mayúsculas
Tipo de contenido	CMC_CT_	Mayúsculas
Propiedad	CMC_PT_	Mayúsculas
Rótulo (tag) de estructura	CMC_TAG_	Mayúsculas
Reservado para conjuntos de extensiones	CMC_XS_	Mayúsculas o minúsculas
Reservado para extensiones	CMC_X_	Mayúsculas o minúsculas
Reservado para uso por implementadores	CMCP	Mayúsculas o minúsculas

Los elementos que llevan el prefijo "CMCP" (escritos con mayúsculas o minúsculas) están reservados para uso privado interno por implementadores del servicio CMC. No están destinados a ser utilizados directamente en programas escritos utilizando la interfaz CMC.

Los prefijos "CMC_XS" y "CMC_X_" (escritos con mayúsculas o minúsculas) están reservados para extensiones de la interfaz por vendedores o grupos de vendedores.

En el caso de valores de datos constantes, se suele agregar al final de "CMC_" una cadena adicional para indicar la estructura de datos o la función a que pertenece el valor de datos constante.

2 Arquitectura de la CMC

Esta cláusula describe la arquitectura funcional sobre la que descansa la interfaz API de la CMC. Define el modelo funcional de la CMC, el modelo de configuración de la CMC, el modelo computacional de la CMC API, y el modelo de objeto de la CMC. El modelo funcional define las funciones de mensajería normalizadas por esta Recomendación. El modelo de configuración define cómo múltiples implementaciones CMC pueden coexistir en una plataforma dada. El modelo computacional define características comunes de la interfaz de programación CMC. El modelo de objeto describe características de los objetos definidos por esta Recomendación.

2.1 Modelo funcional

La interfaz CMC se define entre una aplicación habilitada para mensajería y un servicio de mensajería. El servicio de mensajería, a su vez, puede soportar múltiples servicios de protocolo de mensajería, cada uno de los cuales utiliza diferentes formatos y protocolos de mensajería, por ejemplo, X.400, RFC 822 y RFC 1521. Todas las funciones de esta interfaz están diseñadas para que sean independientes de los servicios de protocolo de mensajería. Sin embargo, la API no permite invocar funciones específicas de protocolo mediante la definición de propiedades específicas de la implementación, ni mediante la utilización de extensiones (véase 2.2.5, Extensiones).

La interfaz CMC se representa en la figura 1.

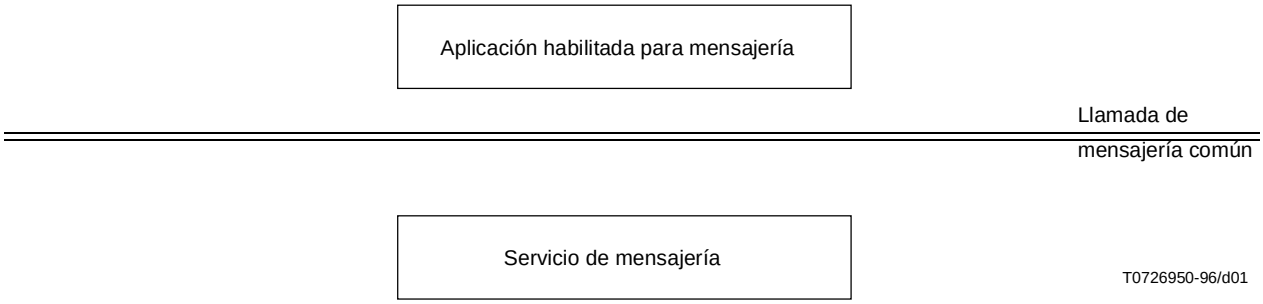


Figura 1/X.446 – Emplazamiento de la API para llamada de mensajería común

Los componentes funcionales subyacentes a la CMC API se muestran en la figura 2.

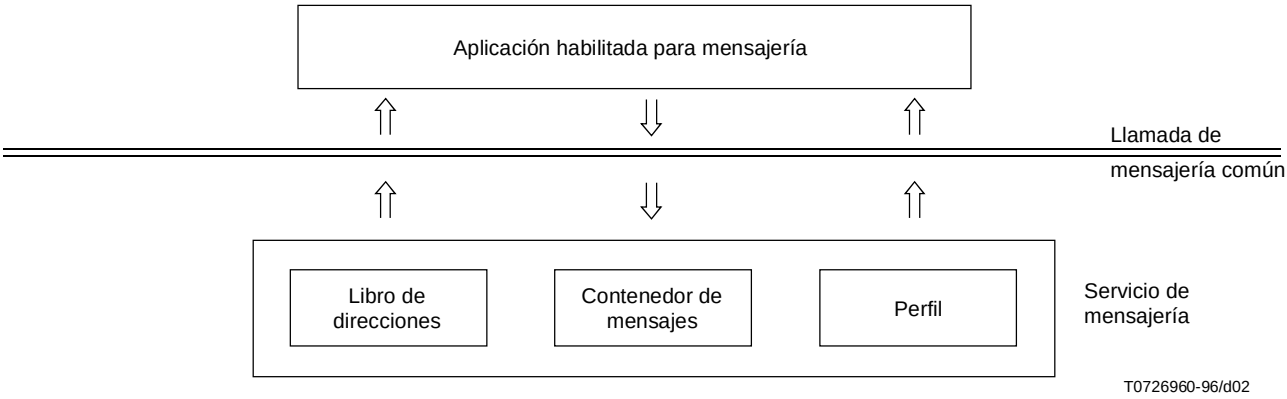


Figura 2/X.446 – Modelo de la API para llamada de mensajería común

La CMC API tiene tres componentes funcionales: libro de direcciones, contenedor de mensajes y perfil. El libro de direcciones contiene información relativa a los recibientes y a la lista de distribución para el direccionamiento de los mensajes. El contenedor de mensajes contiene mensajes. Los contenedores de mensajes usuales son el apartado de entrada (*inbox*), el apartado de salida (*outbox*), y el apartado de correo enviado (*sent mailbox*). El perfil contiene información relativa a la implementación CMC y el usuario.

La interacción de una aplicación habilitada para mensajería y estos componentes funcionales se especifica por el modelo computacional CMC.

2.2 Modelo computacional

El modelo computacional CMC define las interfaces definidas por la especificación y las características comunes de estas interfaces. Estas características comunes incluyen el concepto de una sesión CMC, el soporte del juego de caracteres, un mecanismo de extensión, y la notificación de eventos.

2.2.1 Interfaces

La API para la CMC (brevemente, CMC API) define dos interfaces: la interfaz CMC simple y la interfaz CMC completa. La CMC simple tiene por finalidad ofrecer una funcionalidad de mensajería básica para aplicaciones conscientes de la mensajería. La CMC completa está diseñada para ofrecer una funcionalidad de mensajería realzada para aplicaciones habilitadas para mensajería.

2.2.1.1 CMC simple

La interfaz CMC simple es retrocompatible con la implementación CMC 1.0. La CMC simple añade a CMC 1.0 un nuevo tipo de mensaje, CMC: REPORT, para permitir que los informes de entrega y de no entrega relacionados con un mensaje original se reúnan en mensaje de informe único, consistente con varias implementaciones X.400.

La interfaz CMC simple soporta tres tareas principales: envío de mensajes, lectura de mensajes, y consulta de información de direccionamiento. Las funciones de esta interfaz tienen por finalidad proporcionar soporte habilitado para mensajería a las aplicaciones conscientes de la mensajería. Estas son aplicaciones que no dependen de servicios de correo para realizar sus funciones básicas (por ejemplo, aplicaciones de tratamiento de texto, hoja de cálculo, imagen o gestión de documentos). El acceso a servicios de correo permite utilizar mejor estas aplicaciones dentro del entorno computacional de una empresa.

Para enviar un mensaje, la aplicación habilitada para mensajería debe primeramente establecer una sesión con el servicio de mensajería mediante la función **cmc_logon()**, o interactivamente fijando la bandera LOGON_UI_ALLOWED en el argumento extensiones de la función **cmc_send()**. Una aplicación deposita un mensaje en el servicio de mensajería de depósito por medio de una función **cmc_send()**. La aplicación habilitada para mensajería es responsable de llenar con elementos apropiados la estructura de mensaje CMC utilizada en la función **cmc_send()**. La aplicación habilitada para mensajería puede también utilizar una función **cmc_send_documents()**, más limitada, para enviar un mensaje. Esta función está principalmente destinada a llamadas desde un lenguaje de macro. La clausura de una sesión se efectúa mediante la función **cmc_logoff()**.

Para extraer un mensaje, la aplicación habilitada para mensajería establece una sesión mediante la función **cmc_logon()**. La aplicación puede entonces extraer un sumario de información de apartado de correo mediante la función **cmc_list()**. Se puede extraer mensajes individuales mediante la función **cmc_read()**. La función **cmc_act_on()** permite al usuario actuar sobre un mensaje en el apartado de correo (por ejemplo, suprimirlo). La memoria asignada por el sistema para estructuras se libera pasando a la función **cmc_free()** el puntero retornado. La clausura de una sesión se efectúa mediante la función **cmc_logoff()**. La CMC simple sólo normaliza el acceso al contenedor de mensajes apartado de entrada (*inbox*). El acceso a otros contenedores de mensajes a través de la interfaz CMC simple puede proporcionarse mediante extensiones específicas del vendedor.

Para consultar nombres, la aplicación habilitada para mensajería establece una sesión mediante la función **cmc_logon()**, o interactivamente fijando la bandera LOGON_UI_ALLOWED en el argumento extensiones de la función **cmc_look_up()**. La aplicación utiliza entonces la función **cmc_look_up()** para traducir un nombre cómodo para el usuario a una dirección de mensajería. La memoria asignada por el sistema para estructuras se libera pasando a la función **cmc_free()** el puntero retornado. La clausura de una sesión se efectúa mediante la función **cmc_logoff()**. Dependerá de la implementación los libros de direcciones que habrán de examinarse para las búsquedas por medio de la función **cmc_look_up()**. Las búsquedas en los libros de direcciones específicos mediante la CMC simple puede proporcionarse por medio de extensiones específicas del vendedor.

2.2.1.2 CMC completa

Los argumentos de la interfaz CMC completa aumentan las funciones conscientes de la mensajería proporcionadas por la interfaz CMC simple con funciones adicionales habilitadas para mensajería. Las principales tareas adicionales son: composición de mensajes, acceso y modificación de carpetas de mensajes (*message folders*), acceso, en forma de tren, a grandes volúmenes de información de contenido, y modificación del libro de direcciones. Las aplicaciones basadas en mensajes dependen de servicios de mensajería para realizar sus funciones básicas (por ejemplo, las aplicaciones de unidad de acceso o agente de un servicio de correos o las aplicaciones de gestión de flujos de trabajo). El acceso a servicios de correos es un requisito previo para el funcionamiento de estas aplicaciones.

Las funciones realzadas de la interfaz CMC completa se facilitan por varias estructuras de datos adicionales. Las capacidades proporcionadas por estas estructuras de datos incluyen: un modelo de datos basado en objetos, la definición de un modelo de propiedades de objetos que permite la definición extensible de objetos del servicio de mensajes, una denominación de contenido que facilita el soporte de contenido de multimedios en los mensajes, y un potente conjunto de tipos de mensajes (por ejemplo, calendarización (*calendaring*), itinerarización (*scheduling*) flujo de trabajo (*workflow*), intercambio electrónico de datos (EDI, *electronic data interchange*), mensajes activos), y objetos de contenedor anidados para soportar el registro en carpetas (*foldering*) en memorias de mensajes y libros de direcciones.

2.2.2 Sesión

Tanto en la interfaz CMC simple como en la interfaz CMC completa, las llamadas a funciones CMC se efectúan dentro del contexto de una sesión. Una sesión se establece con la función **cmc_logon()** y se termina con la función **cmc_logoff()**. La función **cmc_logon()** también autentica al usuario ante el servicio de mensajería y establece los atributos de la sesión. El contexto de una sesión se identifica por un identificador de sesión opaca retornado por la función **cmc_logon()**. Los atributos del contexto de la sesión incluyen el juego de caracteres y el número de la versión. Actualmente la compartición de sesiones entre aplicaciones no está soportada.

En el caso de aplicaciones cabeceras (*gateway applications*) un usuario único, que representa la cabecera, puede establecer sesiones a nombre de múltiples usuarios individuales y, por tanto, tener permisos más amplios que los de un usuario individual.

2.2.3 Soporte de caracteres formados por dos octetos

La interfaz CMC completa soporta cadenas de caracteres formados por dos octetos (por ejemplo, UNICODE). Esto se efectúa definiendo la constante CMC_WCHAR dentro de un entorno de desarrollo de aplicación, antes de incluir el fichero xcmc.h (es decir, fijar CMC_WCHAR=1). Si CMC_WCHAR no está definida, los caracteres estarán formados por un solo octeto. La definición de CMC_WCHAR obliga a que todas las cadenas de caracteres en la interfaz CMC completa estén formadas por caracteres de dos octetos. Las cadenas de caracteres de dos octetos son soportadas solamente por la CMC completa. El fichero xcmc.h contiene los prototipos de las definiciones de las funciones para los caracteres formados por un solo octeto y por dos octetos, para cada llamada API, según que la constante CMC_WCHAR esté o no definida.

Con esto se asegura la retrocompatibilidad con CMC 1.0 y se permite el soporte de cadenas de caracteres formados por dos octetos en la CMC simple y en la CMC completa. Las implementaciones exportan las funciones de dos octetos en una DLL separada. Las aplicaciones no están autorizadas a mezclar los dos paradigmas en una misma instancia de la aplicación. El soporte de las cadenas de caracteres de dos octetos no se requiere para la conformidad mínima.

2.2.4 Notificación de eventos

La interfaz CMC simple no soporta la notificación de eventos en el servicio subyacente, como la notificación de nuevos mensajes. En la interfaz CMC completa se han proporcionado cuatro funciones para soportar esta funcionalidad. Están soportados dos modos de notificación: la interrogación secuencial (*polling*) y llamada de retorno (*callback*).

En el modo de notificación por interrogación secuencial, la aplicación registra un interés en la interrogación secuencial sobre un evento con la función **cmc_register_event()**. La aplicación interroga entonces a la implementación con un periodo de temporización facultativo para verificar si ha ocurrido un evento, mediante la función **cmc_check_event()**. Si el evento ha ocurrido, la función retorna un resultado de éxito. Además, la función puede retornar datos específicos del evento. Si la aplicación deja de interesarse en un evento, invoca la función **cmc_unregister_event()**.

El segundo modo de interacción utiliza llamadas de retorno a funciones definidas en la aplicación. En este modo, la aplicación registra una función de llamada de retorno con la implementación, mediante la función **cmc_register_event()**. La función de llamada de retorno de la aplicación se invoca entonces automáticamente cuando ocurre el evento. La aplicación puede también desear forzar una llamada de retorno con la función **cmc_call_callbacks()**. Esta función es útil en entornos en que una aplicación sólo puede efectuar llamadas de retorno cuando se está ejecutando código de la implementación. Si la aplicación deja de estar interesada en un evento, invoca la función **cmc_unregister_event()**.

En esta Recomendación existe solamente un evento normalizado para señalar la llegada de un nuevo mensaje a un contenedor (CMC_EVENT_NEW_MESSAGES). Las estructuras de datos asociadas con este evento se especifican en 4.6 Estructuras de datos de llamadas de retorno. Cuando registra para el evento de nuevo mensaje, la aplicación indica los contenedores que se examinarán para comprobar si hay nuevos mensajes. Pueden examinarse múltiples contenedores. Si la aplicación no registra una llamada de retorno para el evento, puede interrogar secuencialmente para eventos de nuevo mensaje en el conjunto de contenedores especificados en la función **cmc_ckeck_event()**. Si la aplicación registra una función de llamada de retorno, se invoca dicha función cuando llega un nuevo mensaje a un contenedor especificado en la función **cmc_register_event()**.

Esta arquitectura de notificación de eventos permite añadir nuevos eventos en futuras extensiones de CMC, y mediante extensiones proporcionadas por los vendedores.

2.2.5 Extensiones

Tanto en la interfaz CMC simple como en la completa, las estructuras de datos y las funciones definidas en esta Recomendación pueden ampliarse metódicamente mediante la utilización de extensiones. Se utilizan extensiones para añadir campos a estructuras de datos y parámetros a una llamada de función. Para estas extensiones se ha definido una estructura de datos genérica normalizada. Consiste en un código de ítem (*item code*) que identifica la extensión, un dato de ítem (*item data*), que contiene la longitud de los datos de la extensión o de los propios datos, una referencia de ítem (*item reference*), que apunta al lugar en que está almacenado el valor de la extensión, o NULL, si no hay un almacenamiento de ítem correspondiente, y banderas para la extensión.

Las extensiones que son parámetros adicionales a una llamada de función pueden ser extensiones de entrada o extensiones de salida. Es decir, la extensión puede pasarse como parámetros de entrada de la aplicación al servicio CMC, o como parámetros de salida del servicio CMC a la aplicación. Si una extensión es un parámetro de entrada, la aplicación asigna memoria para la estructura de extensión y cualquier otra estructura asociada con la extensión. Si una extensión es un parámetro de salida, el servicio CMC asigna memoria para el resultado de la extensión, si es necesario. En este caso, la aplicación tiene que liberar la memoria asignada mediante una llamada a la función **cmc_free()**.

Las extensiones desempeñan un doble papel en esta Recomendación. En primer lugar, proporcionan un mecanismo en virtud del cual pueden ser acomodadas las características que no son comunes a todos los servicios de mensajería. En segundo lugar, proporcionan un mecanismo para ampliar la Recomendación en el futuro, minimizando toda posible cuestión de retrocompatibilidad.

La utilización de extensiones por el primer motivo, aunque es muy importante, debe hacerse con precaución. La dependencia con respecto a características específicas de servicios de mensajería particulares limita la portabilidad de la aplicación a través de servicios de mensajería; asimismo, es posible que dichas características no se conserven cuando atraviesen múltiples cabeceras en una red de mensajería mixta.

Para minimizar las cuestiones de portabilidad, se insta a los implementadores a especificar extensiones de la manera más general posible, y a aportar estas extensiones como propuestas de adiciones al conjunto de extensiones definidas por la CMC. Mediante este proceso, el conjunto de las interfaces API de la CMC evolucionará en un sentido positivo, lo que contribuirá a que se continúe acrecentando la portabilidad.

Para mas información sobre el registro de extensiones y las extensiones definidas en esta Recomendación, véanse los anexos.

2.3 Modelo de configuración

El modelo de configuración de CMC permite a múltiples implementaciones CMC coexistir en un entorno único especificando un gestor de CMC como un intermediario (*broker*) entre implementaciones CMC. La figura 3 muestra la relación entre el gestor de CMC y las implementaciones de CMC.

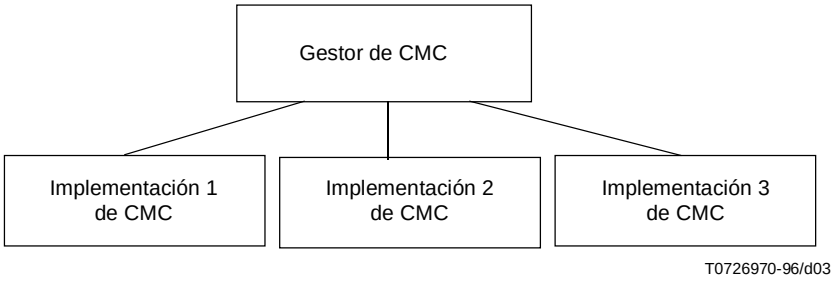


Figura 3/X.446 – Gestor de CMC e implementaciones de CMC

2.3.1 Gestor de CMC

El gestor de CMC (brevemente, gestor CMC) sirve de intermediario para devolver tablas de despacho a la aplicación mediante el empleo de la función **cmc_bind_implementation()**. La tabla de despacho representa una matriz de punteros de funciones CMC cuyas posiciones ordinales tienen que concordar con el orden especificado en el fichero de encabezamiento CMC (*CMC header file*). La aplicación invoca la función de implementación CMC apropiada a través de la correspondiente tabla de despacho de la implementación. Esto implica que la aplicación tiene que mantener copias de los punteros a las tablas de despacho para cada implementación CMC a que desea vincularse. Se utiliza la función **cmc_free()** para liberar la tabla de despacho creada por la llamada a la función **cmc_bind_implementation()**. La función **cmc_unbind_implementation()** se utiliza para hacer desaparecer todo lo demás que haya sido establecido por el gestor CMC o la implementación CMC. Las implementaciones CMC se denominan como identificadores globalmente únicos (*GUIDS, globally unique identifiers*). Las aplicaciones obtienen nombres de implementaciones y GUIDS de los ficheros de encabezamiento (*header files*) de los vendedores, de la documentación suministrada por los vendedores, o por convenios. El gestor CMC es responsable de hacer corresponder la tabla de despacho de la implementación con el espacio de direcciones de la aplicación en plataformas cuyas aplicaciones pueden residir en diferentes espacios de direcciones. El gestor CMC puede desear crear y gestionar copias locales de toda tabla de despacho que pase a través de él. Los gestores CMC tienen que crear y gestionar sus propias copias en el caso antes mencionado de los diferentes espacios de direcciones. A continuación se define el flujo de la ejecución:

1) La aplicación invoca la función **cmc_bind_implementation()** para obtener un puntero a una tabla de despacho para la implementación CMC deseada.

2) El gestor CMC recibe la llamada e invoca la función **cmc_bind_implementation()** de la aplicación CMC apropiada. El gestor CMC tiene que suministrar un medio específico de la plataforma para determinar qué implementaciones CMC existen y dónde residen.

3) La implementación CMC recibe la llamada de la función **cmc_bind_implementation()**, y crea y llena una tabla de despacho que se devuelve al gestor CMC. En tal situación, el gestor CMC puede estimar conveniente crear una copia local de la tabla de despacho.

4) El gestor CMC completa su función **cmc_bind_implementation()** recibida y retorna a la aplicación el puntero a la tabla de despacho, a menos que se deba efectuar primero a un nuevo establecimiento de las correspondencias.

5) La aplicación efectúa entonces llamadas a toda implementación CMC vinculada que exista concurrentemente en ese momento.

6) La aplicación puede invocar en cualquier punto la función **cmc_free()** para liberar la memoria asociada con las tablas de despacho creadas por el gestor CMC y/o la implementación CMC. La aplicación invoca la función **cmc_unbind_implementation()** para señalar al gestor CMC que proceda a la liberación general (*clean up*) de los datos asociados con la vinculación de una determinada implementación CMC. El gestor CMC tiene que llamar entonces a la implementación CMC especificada para que haga lo mismo.

7) Cuando todas las implementaciones CMC están desvinculadas, la aplicación puede salir del procedimiento o invocar de nuevo la función **cmc_bind_implementation()**. Las vinculaciones que no son desvinculadas simétricamente con **cmc_unbind_implementation()** corren el riesgo de sufrir pérdidas de memoria, lo que tiene por consecuencia comportamientos impredecibles.

Si las aplicaciones están usando CMC 1.0, deben llamar a la implementación CMC directamente y no a través del gestor CMC. CMC 1.0 no soporta el acceso a múltiples implementaciones.

2.3.2 Directrices para vinculaciones de plataformas

La CMC 2.0 soporta un gestor CMC y múltiples implementaciones de CMC en una sola plataforma. Para el soporte del gestor CMC y múltiples implementaciones CMC se necesitan las siguientes directrices:

- Cada vinculación de plataforma tiene que especificar un mecanismo que permita a las implementaciones de CMC registrarse ante el gestor CMC y anular dicho registro.
- El gestor CMC tiene que soportar por lo menos las funciones **cmc_bind_implementation()** y **cmc_unbind_implementation()** de la versión CMC 2.0.
- El gestor CMC puede también soportar cualquiera de las siguientes operaciones:
  - interfuncionamiento con implementaciones CMC en otro espacio de direcciones;
  - interfuncionamiento con implementaciones CMC en otra máquina;
  - examen de las implementaciones CMC registradas.
- Ciertas plataformas pueden requerir que las implementaciones CMC modifiquen los nombres de las funciones CMC para soportar múltiples implementaciones. El gestor CMC necesitará tramitar la obtención de las correspondencias con estos nombres de funciones modificados.

2.3.3 Consulta de la información de configuración

La configuración persistente del servicio está disponible para ser consultada por la aplicación habilitada para mensajería. La aplicación puede interrogar al servicio para determinar su soporte a una o más versiones diferentes de la CMC API, extensiones, y los parámetros de entorno que forman la configuración. En esta API no se define ninguna función para la modificación de esta información de configuración. La forma de almacenamiento de esta información (por ejemplo en formato de fichero) no está definida en esta Recomendación.

Se proporcionan dos mecanismos para consultar la información de configuración. La interfaz CMC simple utiliza una llamada a la función **cmc_query_configuration()**. La interfaz CMC completa utiliza sus funciones de enumeración para extraer la información de configuración de un contenedor de perfiles.

2.4 Modelo de objeto

La especificación de la CMC se basa en un eficaz modelo de datos orientado a objetos. Además, un grupo de funciones orientadas a la gestión de estos objetos dentro del servicio de mensajes define un método de acceso muy general. Estas funciones genéricas proporcionan un método muy eficaz, aunque simple, para crear y gestionar el objeto y las propiedades del objeto definidos por la especificación CMC.

El modelo de objeto de la especificación CMC es más bien transparente al usuario de la interfaz CMC simple. Este conjunto de funciones habilitadas para mensajería fue diseñado para simplificar el acceso de funciones del servicio de mensajes. Por otro lado, la interfaz CMC completa proporciona un conjunto realzado de funciones habilitadas para mensajería, que dan acceso a las potentes características de un servicio de mensajes y su modelo de objeto.

Esta subcláusula proporciona una visión de conjunto de los objetos de CMC, las clases de objetos, y presenta ejemplos de propiedades para cada objeto.

2.4.1 Componentes del modelo

En la especificación de la CMC, el modelo contiene objetos, clases de objetos, y propiedades. La figura 4 ilustra los componentes del modelo de objeto CMC. Un objeto es una colección de propiedades. Los objetos se clasifican a su tipo o a su clase. Una propiedad es un atributo del objeto.

2.4.1.1 Objetos

Los objetos se identifican por su asa (*handle*) de objeto específica de la sesión. El asa de objeto encapsula el identificador de sesión. La función **cmc_open_object_handle()** retorna un asa para un nuevo objeto. La información de contenido, que define el objeto de servicio de mensajería de que se trata, puede añadirse con la función **cmc_add_properties()**. Una propiedad individual sólo puede existir una vez en un objeto. Por tanto, esta misma función puede utilizarse para actualizar o modificar la información de contenido asociada a una propiedad determinada. La función **cmc_delete_properties()** puede utilizarse para suprimir una o más propiedades individuales de un objeto. Las propiedades de un objeto pueden ser listadas con la función **cmc_list_properties()**. La información de contenido para una o más propiedades puede leerse con la función **cmc_read_properties()**.

Esta Recomendación permite propiedades con múltiples valores; las propiedades con múltiples valores se utilizan junto con ciertos objetos en esta Recomendación.

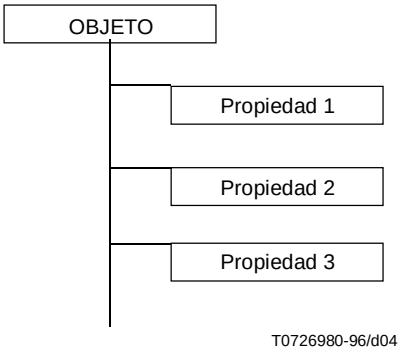


Figura 4/X.446 – Modelo objeto

Los contenedores son una modalidad de un objeto. Forman una colección, no solo de propiedades, sino también de otros objetos.

Una vez definido el objeto, tiene que ser añadido a un contenedor particular y comprometido (*committed*) al almacenamiento persistente de ese contenedor con las funciones **cmc_copy_object()** y **cmc_commit_object()**, respectivamente. Un objeto puede ser suprimido en un objeto contenedor por la función **cmc_delete_object()**.

La enumeración de los objetos del contenedor se facilita por el cursor del contenedor. Un cursor es una construcción específica de la implementación que se utiliza para clasificar y filtrar los elementos de un objeto de contenedor. El cursor puede utilizarse también para facilitar la visualización de una "corredera" (*thumb*) sobre una barra de desfile, que representa la posición relativa dentro de un contenedor. Se utiliza la función **cmc_open_cursor()** para definir un cursor. El contexto del cursor se mantiene referenciando el cursor por su asa de cursor opaco. El asa de cursor, así como los identificadores de objeto y de sesión, son asignados por el servicio de mensajes. Deben ser liberados por la función **cmc_free()** cuando dejen de necesitarse. La posición relativa del cursor puede ser leída por la función **cmc_read-cursor()** y actualizada por la función **cmc_update_cursor_position()**. El cursor puede también actualizarse llevándolo a una posición determinada por la posición relativa de un objeto dentro del contenedor, mediante la función **cmc_update_cursor_position_with_seed()**. El número de objetos en un contenedor que concuerdan con las restricciones del filtro de un cursor pueden ser listados por la función **cmc_list_number_matched()**.

Los objetos dentro del contenedor asociado con un cursor pueden ser listados con la función **cmc_list_objects()**. Los objetos son referenciados por las asas de objeto retornadas por esta función. La función **cmc_copy_object_handle()** puede utilizarse para hacer una copia de la referencia a uno de estos objetos.

Es posible que los contenedores no contengan ningún objeto, como por ejemplo en los casos de un contenedor de mensajes vacío o de un libro de direcciones vacío.

El soporte del anidamiento (*nesting*) de contenedores no es obligatorio en la interfaz CMC completa. No se requiere el anidamiento del contenedor de mensajes ni del libro de direcciones. El código de error que debe retornar una implementación en tal caso es CMC_E_UNSUPPORTED_ACTION. Cuando se recibe un mensaje con un mensaje incrustado (*embedded message*), la implementación no puede garantizar que se hará seguir el mensaje incrustado. Las implementaciones tienen que aceptar los mensajes anidados provenientes de la aplicación. Un anidamiento puede no ser preservado como tal después de que se haya liberado el asa del objeto anidado. Los objetos de mensaje creados por la implementación pueden no estar sujetos a anidamiento. Las implementaciones tienen la opción de generar objetos anidados o no anidados.

2.4.1.2 Clases de objetos

Las clases de objetos son los tipos de objetos definidos por esta Recomendación. Las clases de objetos en la CMC contienen propiedades y posiblemente otros objetos. La cláusula 3 describe las clases de objetos y la cláusula 5 describe las propiedades de cada clase de objeto. Los objetos que otra clase de objeto puede contener se indican en una jerarquía de contención. Esta jerarquía de contención se indica en la figura 5. La jerarquía de contención no ilustra toda la extensión de recursión de los objetos en la CMC.

En esta Recomendación no se define una jerarquía de clase explícita. En cambio, algunas propiedades pueden pertenecer a más de una clase de objetos.

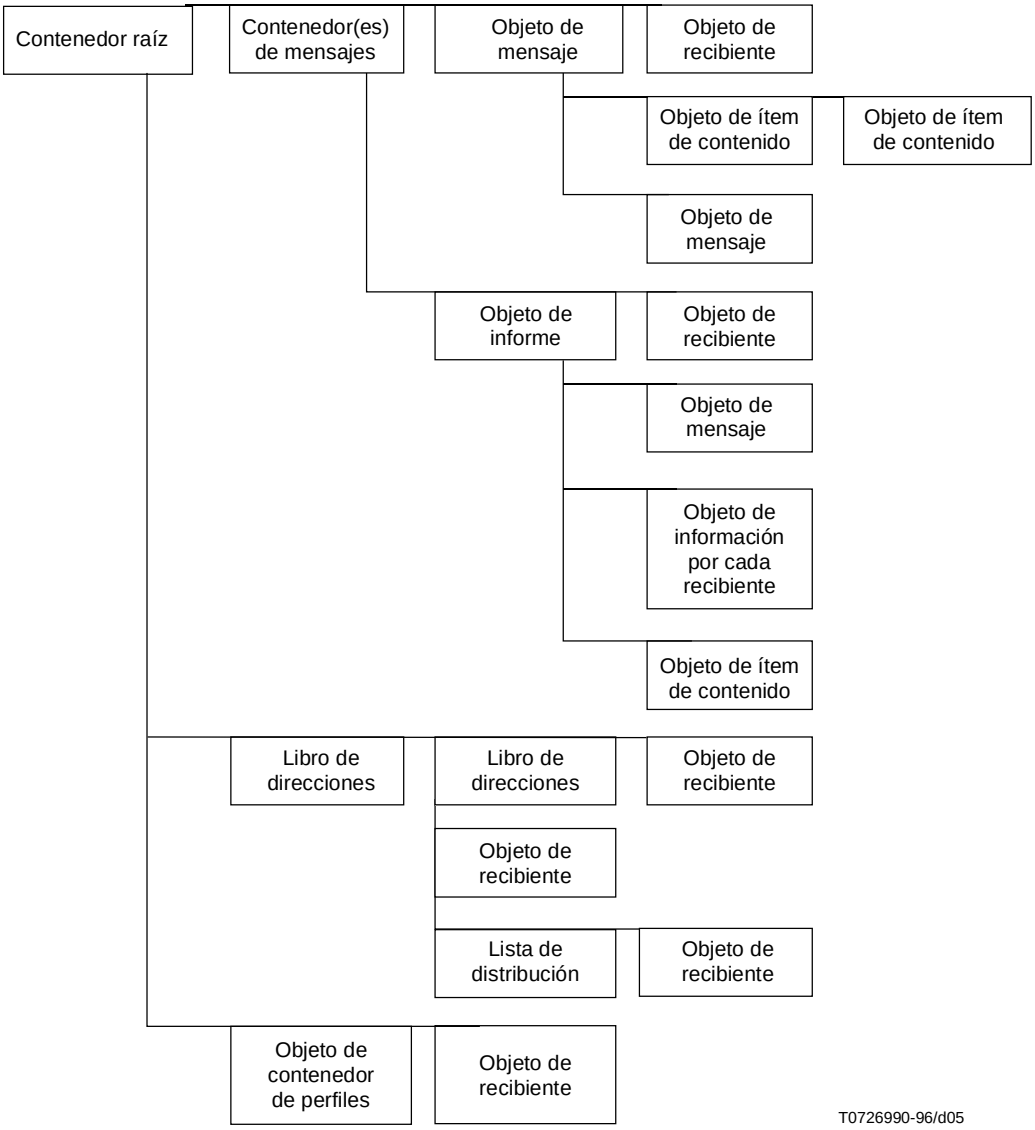


Figura 5/X.446 – Jerarquía de contención de la CMC 2.0

2.4.1.3 Propiedades de objetos

Las propiedades son los atributos de un determinado objeto. Las propiedades definen el objeto. Se designan por un nombre único o, si no, por un identificador específico de la implementación, un tipo de valor, y los datos de valor o información de contenido. Una propiedad se identifica unívocamente por un número entero y un nombre formado por una cadena de caracteres basado en el identificador público formal de ISO 9070.

Algunas implementaciones pueden prever propiedades definidas por el usuario. Esta capacidad permite una personalización del servicio subyacente. Las propiedades definidas por el usuario se distinguen por su nombre (de propiedad). Un mecanismo específico de la plataforma genera un identificador de propiedad único para una propiedad definida por el usuario. Se proporciona la función **cmc_identifier_to_name()** para establecer la correspondencia de un identificador de propiedad a su nombre de propiedad asociado. Se proporciona la función **cmc_name_to_identifier()** para establecer la correspondencia de un nombre de propiedad a su identificador de propiedad asociado. Los identificadores de propiedad y los nombres de propiedad los proporciona el servicio para permitir el acceso a las numerosas propiedades por el método más expeditivo. El espacio de números de identificador de propiedad se divide en espacios para números definidos por la Asociación XAPIA, números definidos por la implementación y números definidos por el usuario. Los números definidos por el usuario corren el riesgo de una posible duplicación en las distintas implementaciones o versiones.

Algunas propiedades de objetos consisten en grandes cantidades de información de contenido. Por ejemplo, el contenido de vídeo o audio de un mensaje multimedios podría ser de un megaocteto. Para facilitar la lectura y escritura de esta voluminosa información de contenido se han añadido funciones de tren (*stream functions*) a la interfaz CMC completa. El acceso a la información de contenido es similar al acceso normal a los ficheros en el lenguaje de programación C. Una propiedad se abre para operaciones de lectura o escritura por medio de la función **cmc_open_stream()**. Esta función retorna un asa de tren que mantiene el contexto de ese tren durante la sesión. Esta asa la asigna el servicio y debe liberarse por la función **cmc_free()** cuando deje de necesitarse. Las funciones **cmc_read_stream()** y **cmc_write_stream()** se utilizan para leer y escribir informaciones de contenido en forma de tren, respectivamente. Se utiliza la función **cmc_seek_stream()** para ir a una determinada posición de octeto dentro del tren. El tren se cierra como una operación secundaria de la llamada a la función **cmc_free()**.

La existencia de diferentes implementaciones puede entrañar costos diferentes para la lectura de las diferentes propiedades. La lectura de las propiedades que consisten en grandes volúmenes de información de contenido puede entrañar un costo importante. La lectura de las propiedades que consisten en pequeños volúmenes de información de contenido o en las que la información está fácilmente disponible por el servicio puede entrañar un costo pequeño o insignificante. La característica de costo relativo, específico de la implementación, para la lectura de cada propiedad, puede determinarse por la función **cmc_read_property_costs()**.

3 Clases de objetos de CMC

3.2 Clases de objetos API de CMC

Las subcláusulas siguientes definen las clases de objetos de la interfaz API de la CMC, indican los nombres de las clases, describen detalladamente los requisitos para el soporte de las clases de objetos, y explican la manera de crear, añadir y modificar objetos de cada clase.

El cuadro 2 recapitula las clases de objetos. La primera columna indica el nombre de la clase de objeto. La segunda columna indica si la clase de objeto es obligatoria o facultativa. La tercera columna especifica si la clase de objeto es o no de lectura solamente. Un "no" en esta columna significa que la clase de objeto puede ser añadida, suprimida o comprometida por las funciones **cmc_copy_object(), cmc_delete_object()**, o **cmc_commit_object()**, respectivamente, a menos que se indique lo contrario por un asterisco "*". La cuarta columna indica el número de las instancias permitidas de un objeto. La última columna indica el creador de la clase de objeto, que puede ser la implementación (I), el llamante (C) o cualquiera de los dos (E).

Las propiedades de cada clase de objeto se indican en la cláusula 5 de esta Recomendación.

Cuadro 2/X.446 – Recapitulación de las clases de objetos

Clase de objeto	M/O	Lectura solamente	Instancias	Creador
Libro de direcciones	O	No	Cualquier número	E
Ítem de contenido	O	No	Cualquier número	E
Lista de distribución	O	No	Cualquier número	E
Mensaje	M	No	Cualquier número	E
Contenedor de mensajes – Proyectos	O	No	Cualquier número	C
Contenedor de mensajes – Archivado	O	No	Cualquier número	C
Contenedor de mensajes – Apartado de entrada	O	No*	Cero o más	I
Contenedor de mensajes – Apartado de salida	O	No*	Una	I
Contenedor de mensajes – Enviado, suprimido	O	No	Una	I
Información por cada recibiente	O	No	Una o más	E
Contenedor de perfiles	M	Sí	Una	I
Recibiente	M	No	Cualquier número	E
Informe	O	No	Cualquier número	E
Contenedor raíz	O	No*	Una	I
M Obligatoria (*mandatory*) O Facultativa (*optional*)				

3.1.1 Libro de direcciones

NOMBRE

Libro de direcciones (*address book*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_ADDRESS_BOOK                    \
    "-//XAPIA/CMC/OBJECT CLASS//NONSGML Address Book//EN"
```

DESCRIPCIÓN

La clase contenedor de libros de direcciones incluye contenedores para mantener objetos de recibientes y de listas de distribución. Los contenedores de libros de direcciones pueden estar anidados, aunque las implementaciones no están obligadas a soportar el anidamiento de contenedores de libros de direcciones. Una implementación CMC no está obligada a soportar contenedores de libros de direcciones. Los contenedores de libros de direcciones pueden ser de dos subtipos: globales y personales. Los libros de direcciones contienen objetos de recibientes, objetos de listas de distribución, y facultativamente otros libros de direcciones.

3.1.2 Ítem de contenido

NOMBRE

Ítem de contenido (*content item*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_CONTENT_ITEM                    \
    "-//XAPIA/CMC/OBJECT CLASS//NONSGML Content Item//EN"
```

DESCRIPCIÓN

Esta clase de objeto identifica objetos asociados con el contenido de un mensaje. Se utiliza para representar añadiduras y notas, aunque no se distingue entre ellas en una interfaz de programación. Los objetos de ítem de contenido son tipificados por identificadores globalmente únicos. Los objetos de ítems de contenido pueden estar anidados, aunque el soporte del anidamiento es facultativo en una implementación.

Las implementaciones pueden imponer límites, sea al número de ítems de contenido en cada mensaje, sea al tamaño del ítem de contenido. Si en una llamada para añadir un ítem de contenido se excede el número de ítems de contenido permitidos, se puede generar el error CMC_E_TOO_MANY_CONTENT_ITEMS. Si el ítem de contenido excede el límite de tamaño impuesto por la implementación, se puede generar el error CMC_E_TEXT_TOO_LARGE.

3.1.3 Lista de distribución

NOMBRE

Lista de distribución (*distribution list*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_DISTRIBUTION_LIST               \
    "-//XAPIA/CMC/OBJECT CLASS//NONSGML Distribution List//EN"
```

DESCRIPCIÓN

La clase de objeto de lista de distribución identifica objetos que representan grupos de objetos de recibientes. Las listas de distribución contienen objetos de recibientes y facultativamente otras listas de distribución. El anidamiento de las listas de distribución puede no conservarse después de liberada el asa que apunta a la lista de distribución. Las implementaciones no están obligadas a soportar listas de distribución, ni el anidamiento de listas de distribución. Estas listas de distribución se identifican por objetos de recibientes cuya propiedad de tipo es "grupo".

La utilización de la CMC para construir una lista de distribución no implica que el sistema de mensajería tenga que soportar el acceso a listas de distribución cuyos miembros son administrados por un libro de direcciones o un servicio de directorio distinto del soportado por la implementación CMC.

3.1.4 Mensaje

NOMBRE

Mensaje (*message*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_MESSAGE                  \
   "-//XAPIA/CMC/OBJECT CLASS//NONSGML Message//EN"
```

DESCRIPCIÓN

Esta clase de objeto identifica objetos de mensajes que son vehículos para pasar información de contenido a través de un servicio de mensajería. Estos objetos de mensajes pueden ser mensajes de correo y mensajes de recibo. Los objetos de mensajes pueden ser anidados por las aplicaciones, y las implementaciones tienen que aceptar esos mensajes. El anidamiento puede no conservarse después de liberada el asa que apunta al objeto anidado. Las implementaciones no están obligadas a soportar el anidamiento de mensajes. Los objetos de mensajes pueden contener objetos de recibientes, objetos de ítems de contenido, y objetos de mensajes anidados.

3.1.5 Contenedor de mensajes

NOMBRE

Contenedor de mensajes (*message container*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_MESSAGE_CONTAINER            \
   "-//XAPIA/CMC/OBJECT CLASS//NONSGML Message Container//EN"
```

DESCRIPCIÓN

Los contenedores de mensajes son colecciones de propiedades de contenedores de mensajes, objetos de mensajes, y posiblemente otros contenedores de mensajes. Este objeto de contenedor proporciona también las mejoras para colecciones especializadas como por ejemplo un apartado de entrada, apartado de salida, carpetas de mensajes que habrán de suprimirse, y otras carpetas de mensajes definidas por el usuario.

La clase de objeto de contenedor de mensajes proporciona una capacidad de carpetas para contener objetos de mensajes y posiblemente objetos de informes y otros objetos de contenedores de mensajes. Los contenedores de mensajes pueden esta anidados, aunque las implementaciones no están obligadas a soportar el anidamiento de objetos de contenedores de mensajes. Los contenedores de mensajes definidos por la CMC incluyen los subtipos de proyecto (o borrador), suprimido, enviado, apartado de entrada y apartado de salida.

3.1.5.1 Clases de contenedores de mensajes: Proyecto (o borrador) de mensaje

El contenedor de proyectos (o borradores) de mensajes contiene mensajes que han sido creados, pero no enviados. El soporte del contenedor de proyectos de mensajes es facultativo.

3.1.5.2 Clases de contenedores de mensajes: Suprimido, enviado

El contenedor de mensajes suprimidos contiene mensajes que han sido suprimidos. El contendor de mensajes enviados contiene mensajes que han sido enviados. El soporte de los contenedores de mensajes enviados y de mensajes suprimidos es facultativo.

3.1.5.3 Clase de contenedores de mensajes: Archivado

El contenedor de mensajes archivados contiene mensajes que han sido archivados. El soporte de los contendores de mensajes archivados es facultativo.

3.1.5.4 Clase de contenedor de mensajes: Apartado de entrada

La clase de contenedores de mensajes del subtipo apartado de entrada almacena los mensajes entrantes. El soporte de un apartado de entrada es facultativo; puede haber más de un apartado de entrada.

3.1.5.5 Clase de contenedor de mensajes: Apartado de salida

El apartado de salida contiene mensajes que van a ser enviados. El soporte del apartado de salida es facultativo; sólo se permite un apartado de salida.

3.1.6 Información por cada recibiente

NOMBRE

Información por cada recibiente (*per recipient information*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_PER_RECIPIENT_INFORMATION          \
    "-//XAPIA/CMC/OBJECT CLASS//NONSGML Per Recipient Information//EN"
```

DESCRIPCIÓN

Esta clase de objeto identifica objetos que informan de la entrega o no entrega de un mensaje para un recibiente individual. Los objetos de esta clase están contenidos en objetos de informe. Por lo menos uno de estos objetos tiene que estar presente para el objeto de informa. El soporte de esta clase es facultativo en general, pero obligatorio para las implementaciones que soportan objetos de informe. Los objetos de información por cada recibiente no pueden estar anidados.

3.1.7 Contenedor de perfiles

NOMBRE

Contenedor de perfiles (*profile container*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_PROFILE_CONTAINER            \
    "-//XAPIA/CMC/OBJECT CLASS//NONSGML Profile Container//EN"
```

DESCRIPCIÓN

El contenedor de perfiles incluye información de contexto y de configuración de sesión. Sólo hay un contenedor de perfiles. Está situado debajo del objeto de contenedor raíz. El objeto de contenedor de perfiles lo crea el servicio de mensajería subyacente, es de lectura solamente, y no puede ser modificado por el usuario. El contenido del objeto de contenedor de perfiles también lo crea el servicio de mensajería subyacente, es de lectura solamente, y no puede ser modificado por el usuario. El soporte del objeto de contenedor de perfiles es obligatorio para las implementaciones que son conformes con esta Recomendación.

El contenedor de perfiles contiene un objeto de recibiente que corresponde al usuario que ha establecido una sesión. Si la implementación soporta el establecimiento de sesión compartido, el contenedor podrá contener objetos de recibientes adicionales que correspondan a los otros usuarios que han efectuado el logon. Esto permite el soporte de capacidades de tablero de boletines (*bulletin board*) o de foros de discusión. Además, el objeto de contenedor de perfiles contiene atributos y propiedades de contenedor de perfiles que corresponden a atributos individuales de contexto o configuración de sesión. Hay propiedades definidas para los atributos de configuración de la CMC simple y de la CMC completa. Las propiedades del contenedor de perfiles son de lectura solamente.

3.1.8 Recibiente

NOMBRE

Recibiente (*recipient*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_RECIPIENT                \
    "-//XAPIA/CMC/OBJECT CLASS//NONSGML Recipient//EN"
```

DESCRIPCIÓN

La clase de objeto de recibiente identifica usuarios dentro del servicio de mensajería. Los objetos de recibiente pueden incluir individuos y grupos. El tipo de recibiente puede ser un recibiente individual o un grupo de recibientes (por ejemplo, una lista de distribución), o un tipo desconocido. Una implementación individual puede proporcionar propiedades específicas de la implementación de un objeto de recibiente.

3.1.9 Informe

NOMBRE

Informe (*report*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_REPORT                   \
    "-//XAPIA/CMC/OBJECT CLASS//NONSGML Report//EN"
```

DESCRIPCIÓN

La clase de objeto de informe identifica objetos que informan sobre el estado de la entrega de un mensaje. Los objetos pertenecientes a esta clase incluyen notificaciones de entrega y de no entrega. Ciertos sistemas de transferencia de mensajes (por ejemplo SMTP) pueden no soportar la generación de informes. Los objetos de informes pueden contener objetos de recibientes, objetos de información por cada recibiente, objetos de ítems de contenido, y objetos de mensajes.

3.1.10 Contenedor raíz

NOMBRE

Contenedor raíz (*root container*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_OC_ROOT_CONTAINER                    \
    "-//XAPIA/CMC/OBJECT CLASS//NONSGML Root Container//EN"
```

DESCRIPCIÓN

La clase de contenedor raíz incluye contenedores de nivel superior para los objetos de mensajería de un usuario. Hay un solo tipo de contenedor raíz y un solo contenedor raíz por usuario. Tiene que ser soportado por las implementaciones CMC. El contenedor raíz contiene mensajes, un contenedor de perfiles, y facultativamente contenedores de libros de direcciones.

4 Estructuras de datos

Esta cláusula define las estructuras de datos utilizadas en la API de la CMC, que se indican en el cuadro 3.

Cuadro 3/X.446 – Estructuras de datos de la CMC

Nombre de tipo de datos	Descripción
Añadidura (*attachment*)	Estructura de añadidura de mensaje
Booleano (*boolean*)	Un valor que indica las condiciones lógicas de verdadero o falso
Memoria tampón (*buffer*)	Puntero a un ítem de datos
Estructuras de datos de llamadas de retorno (*callback data structures*)	Definiciones de tipos para los valores de datos de una función de llamada de retorno
Cadena contada (*counted string*)	Cadena cuya longitud se indica explícitamente
Asa de cursor (*cursor handle*)	Asa opaca para un cursor de contenedor
Restricción de cursor (*cursor restriction*)	Restricción para filtrar la enumeración de los objetos en un contenedor
Clave de clasificación de cursor (*cursor sort key*)	Define el orden de clasificación para elementos enumerados por un cursor en un contenedor
Tabla de despacho (*dispatch table*)	Una estructura que contiene punteros a las funciones en una implementación CMC
Enumerado (*enumerated*)	Tipo de datos que contiene un valor que corresponde a la posición en una enumeración
Eventos (*events*)	Tipo de datos para eventos del servicio de mensajería
Extensión (*extension*)	Estructura de extensión
Banderas (*flags*)	Contenedor para bits de bandera
Guid	Identificador globalmente único
Fecha y hora ISO (*ISO date and time*)	Un valor de cadena de fecha y hora con el formato de ISO 8601
Mensaje (*message*)	Estructura de mensaje

Cuadro 3/X.446 – Estructuras de datos de la CMC *(fin)*

Nombre de tipo de datos	Descripción
Referencia de mensaje (*message reference*)	Estructura de referencia de mensaje
Sumario de mensaje (*message summary*)	Estructura de sumario de mensaje
Asa de objeto (*object handle*)	Asa opaca para el objeto de CMC
Identificador de objeto (*object identifier*)	Estructura de identificador de objeto
Datos opacos (*opaque data*)	Cadena contada de octetos de datos específicos de la aplicación
Propiedad (*property*)	Una pieza de información de contenido de objeto
Identificador (*identifier*)	Identificador único, específico de la implementación
Nombre (*name*)	Nombre único
Recibiente (*recipient*)	Estructura de originador/recibiente
Informe (*report*)	Mensaje de estado para notificaciones de entrega, no entrega, recibos, etc.
Código de retorno (*return code*)	Valor de retornado por una función y que indica si ésta ha tenido éxito o la causa por la que fracasó
Identificador de sesión (*session id*)	Asa opaca para sesión
Asa de tren (*stream handle*)	Asa opaca para el tren de propiedad
Cadena (*string*)	Puntero a cadena de caracteres
Tiempo (*u hora*)	Estructura de tiempo (o de hora)
Identificador de interfaz de usuario (*user interface id*)	Asa de interfaz de usuario

4.1 Tipos de datos básicos

Algunos tipos de datos se definen en términos de los siguientes "tipos de datos intermedios", cuyas definiciones precisas en lenguaje C se definen por el sistema:

float32 Número de punto (o coma) flotante representado por 32 bits.

float64 Número de punto (o coma) flotante representado por 64 bits

sint16 Números enteros positivos y negativos que pueden representarse por 16 bits.

sint32 Números enteros positivos y negativos que pueden representarse por 32 bits.

uint8 Números enteros no negativos que pueden representarse por 8 bits.

uint16 Números enteros no negativos que pueden representarse por 16 bits.

uint32 Números enteros no negativos que pueden representarse por 32 bits.

DECLARACIÓN EN LENGUAJE C

```
typedef system-defined, e.g. float              CMC_float32;
typedef system-defined, e.g. double             CMC_float64;
typedef system-defined, e.g. int                CMC_sint16;
typedef system-defined, e.g. long int           CMC_sint32;
typedef system-defined, e.g. unsigned char      CMC_uint8;
typedef system-defined, e.g. unsigned int       CMC_uint16;
typedef system-defined, e.g. unsigned long int  CMC_uint32;
```

4.2 Tipos de datos de matriz

Esta Recomendación soporta propiedades de múltiples valores mediante la utilización de matrices (*arrays*) de tipos de datos básicos y no básicos. Los tipos de datos de matriz se definen como:

array_boolean Una matriz de booleanos.

array_buffer Una matriz de punteros a ubicaciones de almacenamiento en memoria.

array_counted_string Una matriz de cadenas cuya longitud está explícitamente indicada.

array_enum Una matriz de tipos de datos enumerados.

array_extension	Una matriz de tipos de datos de extensión.
array_float32	Una matriz de números de punto flotante representado por 32 bits.
array_float64	Una matriz de números de punto flotante representados en 64 bits.
array_guid	Una matriz de identificadores globalmente únicos.
array_iso_date_time	Una matriz de estructuras de fecha y hora ISO
array_object_handle	Una matriz de asas de objetos.
array_opaque_data	Una matriz de cadenas, formadas por números contados de octetos, de datos específicos de la aplicación.
array_return_code	Una matriz de códigos de retorno.
array_sint16	Una matriz de números enteros positivos y negativos que pueden representarse por 16 bits.
array_sint32	Una matriz de números enteros positivos y negativos que pueden representarse por 32 bits.
array_string	Una matriz de cadenas.
array_time	Una matriz de estructuras de tiempo.
array_uint16	Una matriz de números enteros no negativos que pueden representarse por 16 bits.
array_uint32	Una matriz de números enteros no negativos que pueden representarse por 32 bits.

DECLARACIÓN EN LENGUAJE C

```
typedef struct CMC_TAG_ARRAY_BOOLEAN {
   CMC_uint32                    count;
   CMC_boolean                   *bits;
} CMC_array_boolean;

typedef struct CMC_TAG_ARRAY_BUFFER {
   CMC_uint32                    count;
   CMC_buffer                    *buffer;
} CMC_array_buffer;

typedef struct CMC_TAG_ARRAY_COUNTED_STRING {
   CMC_uint32                    count;
   CMC_counted_string            *string;
} CMC_array_counted_string;

typedef struct CMC_TAG_ARRAY_ENUM {
   CMC_uint32                    count;
   CMC_enum                      *set;
} CMC_array_enum;

typedef struct CMC_TAG_ARRAY_EXTENSION {
   CMC_uint32                    count;
   CMC_extension                 *extension;
} CMC_array_extension;

typedef struct CMC_TAG_ARRAY_FLOAT32 {
   CMC_uint32                    count;
   CMC_float32                   *number;
} CMC_array_float32;

typedef struct CMC_TAG_ARRAY_FLOAT64 {
   CMC_uint32                    count;
   CMC_float64                   *number;
} CMC_array_float64;

typedef struct CMC_TAG_ARRAY_GUID {
   CMC_uint32                    count;
   CMC_guid                      *guid;
} CMC_array_guid;

typedef struct CMC_TAG_ARRAY_ISO_DATE_TIME {
   CMC_uint32                    count;
   CMC_date_time                 *time;
} CMC_array_iso_date_time;

typedef struct CMC_TAG_ARRAY_OBJECT_HANDLE {
   CMC_uint32                    count;
   CMC_object_handle             *ohandles;
} CMC_array_object_handle;

typedef struct CMC_TAG_ARRAY_OPAQUE_DATA {
   CMC_uint32                    count;
   CMC_opaque_data               *data;
} CMC_array_opaque_data;
```

```
typedef struct CMC_TAG_ARRAY_RETURN_CODE {
   CMC_uint32                    count;
   CMC_return_code               *code;
} CMC_array_return_code;

typedef struct CMC_TAG_ARRAY_SINT16 {
   CMC_uint32                    count;
   CMC_sint16                    *number;
} CMC_array_sint16;

typedef struct CMC_TAG_ARRAY_SINT32{
   CMC_uint32                    count;
   CMC_sint32                    *number;
} CMC_array_sint32;

typedef struct CMC_TAG_ARRAY_STRING {
   CMC_uint32                    count;
   CMC_string                    *string;
} CMC_array_string;

typedef struct CMC_TAG_ARRAY_TIME {
   CMC_uint32                    count;
   CMC_time                      *time;
} CMC_array_time;

typedef struct CMC_TAG_ARRAY_UINT16 {
   CMC_uint32                    count;
   CMC_uint16                    *number;
} CMC_array_uint16;

typedef struct CMC_TAG_ARRAY_UINT32 {
   CMC_uint32                    count;
   CMC_uint32                    *number;
} CMC_array_uint32;
```

DESCRIPCIÓN

Un valor de datos de estos tipos incluye un identificador de longitud que determina el tamaño de la matriz.

El soporte de propiedades de múltiples valores es facultativo para las implementaciones.

4.3 Añadidura

NOMBRE

Añadidura (*attachment*) – Definición de tipo para una estructura de añadidura a mensaje CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct {
   CMC_string                    attach_title;
   CMC_object_identifier         attach_type;
   CMC_string                    attach_filename;
   CMC_flags                     attach_flags;
   CMC_extension                 *attach_extensions;
} CMC_attachment;
```

DESCRIPCIÓN

Un valor de datos de este tipo es una añadidura. Esta estructura de datos se incluye para proporcionar el soporte de CMC 1.0 e implementaciones de la CMC simple. Una añadidura tiene los siguientes componentes:

1) *título de añadidura* (attach_title): Título facultativo para la añadidura, por ejemplo, nombre de fichero original de añadidura.

2) *tipo de añadidura* (attach_type): Identificador de objeto que específica el tipo de añadidura. El formato del identificador de objeto de CMC se define en 4.24. Un valor NULL designa un tipo de añadidura no definido.

   Se han predefinido dos identificadores de objetos para uso por aplicaciones e implementaciones CMC.

CMC_ATT_OID_BINARY	Los datos en ficheros deben tratarse como datos binarios. Éste es el valor por defecto.

CMC_ATT_OID_TEXT — Los datos en ficheros deben tratarse como una cadena de texto. Debe suponerse que la cadena está formada por caracteres pertenecientes al juego de caracteres para la sesión en entrada y que se han hecho corresponder con el juego de caracteres para la sesión en salida, si es posible.

3) *nombre de fichero de añadidura* (attach_filename): Nombre del fichero en que está situado el contenido de la añadidura. La ubicación del fichero depende de la implementación, pero se debe garantizar el acceso por la aplicación llamante.

4) *banderas de añadidura* (attach_flags): Bits para atributos booleanos. Los bits no utilizados deben estar despejados (o sea, puestos a cero).

   a) CMC_ATT_APP_OWNS_FILE

      Fijado: Indica en salida que la aplicación posee ahora el fichero y es responsable de su supresión. Se ignora en entrada.

      Despejado: Indica en salida que la implementación CMC posee el fichero y que la aplicación sólo puede leerlo.

   b) CMC_ATT_LAST_ELEMENT

      Fijado: Identifica la última estructura en una matriz de esas estructuras.

      Despejado: Indica que el elemento en cuestión no es el último elemento de la matriz.

5) *extensiones de añadiduras* (attach_extensions): Puntero al primer elemento de la matriz de extensiones por cada añadidura. Un valor NULL indica que no hay extensiones presentes.

4.4 Booleano

NOMBRE

Booleano (*boolean*) – Definición de tipo para un valor de dato booleano.

DECLARACIÓN EN LENGUAJE C

```
typedef   CMC_uint16   CMC_boolean;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un booleano, es decir, es falso o verdadero.

En la interfaz C, el valor booleano falso se representa por cero {CMC_FALSE}, y el valor booleano verdadero se representa por cualquier otro entero, aunque la constante simbólica {CMC_TRUE} tiene específicamente el valor entero uno.

4.5 Memoria tampón

NOMBRE

Memoria tampón (*buffer*) – Definición de tipo para espacio de almacenamiento en memoria de un tipo no definido.

DECLARACIÓN EN LENGUAJE C

```
typedef        void * CMC_buffer;
```

DESCRIPCIÓN

Un valor de datos de este tipo de datos es un puntero a una ubicación de almacenamiento en memoria de un tipo no definido. El tamaño de un valor de tipo void * es específico de la plataforma.

4.6 Estructuras de datos de llamada de retorno

NOMBRE

Estructuras de datos de llamada de retorno (*callback data structures*) – Definiciones de tipos para valores de datos de una función de llamada de retorno.

DECLARACIÓN EN LENGUAJE C

```
typedef struct CMC_TAG_NEW_MESSAGE_CB_DATA {
   CMC_object_handle                        *available;
} CMC_new_message_callback_data;

typedef struct CMC_TAG_NEW_MESSAGE_CHECK_DATA {
   CMC_uint32                               number_containers;
   CMC_object_handle                        *containers;
} CMC_new_message_check_data;

typedef CMC_new_message_check_data          CMC_new_message_register_data;

typedef CMC_new_message_check_data          CMC_new_message_unregister_data;

typedef void (*CMC callback) (
   CMC_session_id                           session,
   CMC_event                                event,
   CMC_buffer                               callback_data,
   CMC_buffer                               register_data,
   CMC_extension                            *callback_extensions
);
```

DESCRIPCIÓN

Los procedimientos de llamada de retorno permiten al servicio informar a las aplicaciones que ha ocurrido un evento. Todos los procedimientos de llamada de retorno son del tipo **cmc_callback**.

Los programadores que escriben procedimientos de llamada de retorno deben considerar el método específico de la plataforma seguido para efectuar la llamada de retorno y la influencia que ejerce las funciones de llamada de retorno sobre la calidad de funcionamiento. Las llamadas de retorno son invocadas en una secuencia específica de la implementación, por el servicio, cuando se produce la actividad de llamada de retorno especificada, o cuando se invoca la función **cmc_call_callbacks()**. Efectivamente, la aplicación CMC que está operando en el momento de la invocación de la llamada de retorno quedará bloqueada hasta que retorne la llamada de retorno. La aptitud de la aplicación CMC para reaccionar se verá afectada si la función de llamada de retorno no retorna rápidamente.

Los siguientes son componentes del prototipo de la función de llamada de retorno:

– **session** (sesión) – El asa opaca que representa una sesión con el servicio de mensajería.

– **event** (evento) – Una máscara de bits de eventos. Exactamente un bit será fijado, es decir, puesto a 1 lo que indica el evento que ha ocurrido y cómo interpretar el argumento **callback_data**. Se definen las siguientes banderas:

  CMC_EVENT_NEW_MESSAGES

  Para la definición de esta bandera, véase el tipo de dato evento.

– **callback_data** (datos de llamada de retorno) – Un puntero a la estructura de datos de llamada de retorno específico para el evento.

– **register_data** (datos de registro) – Un puntero a la estructura de datos pasado a la función **cmc_register()** específica del evento cuando se registra la llamada de retorno.

– **callback_extensions** (extensiones de llamada de retorno) – Un puntero a una matriz de estructuras de extensión CMC para esta función de llamada de retorno.

  Cada función de llamada de retorno devuelve un puntero a una de las estructuras de datos de llamada de retorno en su argumento callback_data. La estructura que se devuelve depende del contexto de la llamada de retorno y está determinada por el valor del argumento evento, descrito más abajo.

  La estructura de datos de llamada de retorno es el mecanismo que el servicio de mensajería utiliza para proporcionar, a la aplicación, información específica de la operación de actualización. La aplicación puede hacer pasar un contexto adicional a sus funciones de llamada de retorno mediante la utilización de la estructura de datos de registro. La estructura de datos de llamada de retorno la asigna la implementación CMC; la estructura de datos de registro la asigna la aplicación.

  Cuando se anula el registro de una llamada de retorno, se puede también especificar datos no registrados asociados con la función **cmc_unregister_event()** para proporcionar un contexto para la supresión de la operación de registro (por ejemplo para esperar nuevos mensajes en un conjunto más pequeño de contenedores). La estructura de datos de anular registro la asigna la aplicación. Los tipos válidos de los argumentos para cada evento se proporcionan más adelante.

En esta Recomendación, la aplicación puede interrogar también sobre eventos con la función **cmc_check_event()**. Los eventos pueden tener un contexto mediante una estructura de datos de comprobación dentro de la cual se puede invocar la función **cmc_ckeck_event()**. La estructura de datos de comprobación la asigna la aplicación. Los tipos válidos de los argumentos para cada evento se proporcionan aquí.

En esta Recomendación, el único evento especificado es CMC_EVENT_NEW_MESSAGES. Una aplicación puede interrogar sobre la existencia de nuevos mensajes utilizando **cmc_check_event()**, o registrar llamadas de retorno que se efectuarán cuando se reciban nuevos mensajes. Si se utiliza la interrogación secuencial, ésta puede estar limitada a determinados contenedores especificados en el argumento de datos de comprobación en la función **cmc_check_event()**, con la estructura CMC_TAG_new_message_check_data. Esta estructura tiene los siguientes elementos de datos:

– **number_containers** (número de contenedores) – El número de asas de contenedores en el argumento **containers**. Si el evento es independiente de la existencia de un contenedor, este argumento debe ser 0.

– **containers** (contenedores) – Una matriz de asas de contenedores que habrán de ser comprobadas para detectar eventos. Si el evento es independiente de la existencia de un contenedor, su argumento debe ser NULL.

Al retornar, **cmc_check_event()** devuelve la estructura CMC_TAG_new_message_callback_data. Los elementos de datos en esta estructura incluyen:

– **available** (disponible) – El asa de un contenedor de mensaje (entre los especificados por el argumento **containers**) a que corresponde el evento. Si no ocurrió ningún evento, el valor se fija a CMC_NULL_HANDLE.

Cuando se registra una llamada de retorno, la estructura CMC_TAG_new_message_register_data se pasa por referencia en el argumento register_data a la función **cmc_register_event()**. Los elementos de datos de esta estructura de datos son idénticos a los elementos de datos de la estructura CMC_TAG_new_message_check_data.

Si se registra una llamada de retorno y ocurre un evento, la estructura CMC_TAG_new_message_callback_data se pasa a la función de llamada de retorno. Además, la estructura CMC_TAG_new_message_register_data se pasa a la función de llamada de retorno.

Cuando se anula el registro de la llamada de retorno, la estructura CMC_TAG_new_message_unregister_data se pasa por referencia en el argumento register_data a la función cmc_unregister. Los elementos de datos de esta estructura de datos incluyen:

– **number_containers** (número de contenedores) – El número de asas de contenedores en el argumento **containers**. Si el evento es independiente de la existencia de un contenedor, este argumento debe ser 0.

– **containers** (contenedores) – Una matriz de asas de los contenedores con relación a los cuales la aplicación ya no está interesada en recibir notificaciones de nuevos mensajes. Esta matriz debe ser un subconjunto de las asas especificadas en el elemento containers del argumento register_data de la función **cmc_register()**. Si el evento es independiente de la existencia de un contenedor, este argumento debe ser NULL.

En todos los casos, el orden en que el servicio invoca funciones de llamada de retorno es específico de la implementación.

4.7 Cadena contada

NOMBRE

Cadena contada (*counted string*) – Definición de tipo para una estructura de cadena contada CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct {
   CMC_uint32       length;
   char             string[1];
} CMC_counted_string;
```

DESCRIPCIÓN

Un valor de datos de este tipo es una cadena contada cuya longitud se expresa explícitamente antes de la matriz de caracteres. No se requiere que esta cadena termine con el carácter nulo.

El soporte de un tipo de datos de cadena contada es facultativo. Tiene por finalidad soportar juegos de caracteres en que se permiten caracteres nulos incrustados.

Para información sobre la determinación del juego de caracteres, véase el tipo CMC_string.

Los componentes de una cadena contada son:

1) *length* (longitud): Longitud de la cadena que sigue, expresada en octetos.

2) *string* (cadena): Los caracteres que constituyen la cadena.

4.8 Asa de cursor

NOMBRE

Asa de cursor (*cursor handle*) – Definición de tipo para una estructura de asa de cursor CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef system-defined, e.g. uint32    CMC_cursor_handle;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un asa de cursor opaca. Las asas de cursor CMC se definen de una manera específica de la implementación. El asa mantiene un contexto de sesión con un cursor de contenedor. El cursor facilita la enumeración de los objetos en un contenedor. Se utiliza también para visualizar una corredera (*thumb*) sobre una barra de desfile, en el control de una ventana, para ilustrar la posición relativa dentro de una colección de objetos. Las asas de cursor no pueden copiarse.

4.9 Restricción de cursor

NOMBRE

Restricción de cursor (*cursor restriction*) – Definición de tipo para un tipo de datos de restricción de cursor CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct CMC_TAG_RESTRICTION_AND {
   CMC_uint32                              count;
   struct CMC_TAG_RESTRICTION_CURSOR       *restriction;
} CMC_restriction_and;

typedef struct CMC_TAG_RESTRICTION_OR {
   CMC_uint32                              count;
   struct CMC_TAG_RESTRICTION_CURSOR       *restriction;
} CMC_restriction_or;

typedef struct CMC_TAG_RESTRICTION_NOT {
   CMC_uint32                              count;
   struct CMC_TAG_RESTRICTION_CURSOR       *restriction;
} CMC_restriction_not;

typedef struct CMC_TAG_RESTRICTION_STRING {
   CMC_enum                      exactness;
   CMC_id                        property;
   CMC_string                    string_constant;
} CMC_restriction_string;

typedef struct CMC_TAG_RESTRICTION_CONTENT {
   CMC_enum                      logical;
   CMC_id                        property;
   CMC_buffer                    property_value;
} CMC_restriction_content;

typedef struct CMC_TAG_RESTRICTION_COMPARISON {
   CMC_enum                      logical;
   CMC_id                        property1;
   CMC_id                        property2;
} CMC_restriction_comparison;

typedef struct CMC_TAG_RESTRICTION_BITTEST {
   CMC_uint32                    comparison;
   CMC_id                        property;
   CMC_uint32                    bitmask;
} CMC_restriction_bitmask;
```

```
typedef struct CMC_TAG_RESTRICTION_SIZE {
   CMC_enum                      logical;
   CMC_id                        property;
   CMC_uint32                    byte_size;
} CMC_restriction_size;

typedef struct CMC_TAG_RESTRICTION_EXIST {
   CMC_id                        property;
} CMC_restriction_exist;

typedef struct CMC_TAG_RESTRICTION_CURSOR {
   CMC_enum                      type;
   union {
      CMC_restriction_and        restriction_and;
      CMC_restriction_or         restriction_or;
      CMC_restriction_not        restriction_not;
      CMC_restriction_string     restriction_string;
      CMC_restriction_content    restriction_content;
      CMC_restriction_comparison restriction_comparison;
      CMC_restriction_bitmask    restriction_bittest;
      CMC_restriction_size       restriction_size;
      CMC_restriction_exist      restriction_exist;
   } cr;
   CMC_extension                 *property_extensions;
} CMC_cursor_restriction;
```

DESCRIPCIÓN

Un valor de datos de este tipo es una restricción de cursor CMC. Una restricción de cursor es la definición de un filtro sobre la enumeración del contenido de un objeto de contenedor. Una restricción de cursor tiene los siguientes componentes:

1) *type* (tipo): El tipo de la restricción de cursor. Están soportados los siguientes tipos de restricción válidos:

   CMC_RESTRICTION_AND
   CMC_RESTRICTION_OR
   CMC_RESTRICTION_NOT
   CMC_RESTRICTION_STRING
   CMC_RESTRICTION_CONTENT
   CMC_RESTRICTION_COMPARISON
   CMC_RESTRICTION_BITTEST
   CMC_RESTRICTION_SIZE
   CMC_RESTRICTION_EXIST

CMC_RESTRICTION_AND – Filtra cuando todas las subrestricciones son verdaderas.

CMC_RESTRICTION_OR – Filtra cuando una o más de las subrestricciones son verdaderas.

CMC_RESTRICTION_NOT – Filtra cuando la subrestricción o subrestricciones son, todas ellas, falsas.

CMC_RESTRICTION_STRING – Filtra cuando una cadena concuerda exactamente con un valor de propiedad.

CMC_RESTRICTION_CONTENT – Filtra en base a una comparación lógica entre una constante y un valor de propiedad.

CMC_RESTRICTION_COMPARISON – Filtra en base a una comparación lógica entre dos valores de propiedade.

CMC_RESTRICTION_BITTEST – Filtra cuando un valor de propiedad concuerda con la máscara de bits especificada para la prueba.

CMC_RESTRICTION_EXIST – Filtra para determinar si una propiedad existe o no en el objeto.

2) *restriction* (restricción): Especifica el valor de restricción del cursor.

3) *property_extensions* (extensiones de propiedades): Puntero al primer elemento de una matriz de extensiones de propiedades.

El elemento de estructura exactitud tiene los siguientes valores enumerados válidos de exactitud de las cadenas:

   CMC_EXACTNESS_PRECISE
   CMC_EXACTNESS_STARTS_WITH
   CMC_EXACTNESS_MIXED_CASE

CMC_EXACTNESS_PRECISE – El valor de propiedad concuerda exactamente con la constante de cadena.
CMC_EXACTNESS_STARTS_WITH – El valor de propiedad comienza con la constante de cadena.
CMC_EXACTNESS_MIXED_CASE – El valor de propiedad concuerda sin tener en cuenta la escritura en mayúsculas/minúsculas.

El elemento de estructura lógica tiene los siguientes valores enumerados válidos de operador lógico:

CMC_LOGICAL_LT
CMC_LOGICAL_LE
CMC_LOGICAL_EQ
CMC_LOGICAL_NE
CMC_LOGICAL_GT
CMC_LOGICAL_GE

CMC_LOGICAL_LT – Menor que.
CMC_LOGICAL_LE – Menor que o igual a.
CMC_LOGICAL_EQ – Igual a.
CMC_LOGICAL_NE – No igual a.
CMC_LOGICAL_GT – Mayor que.
CMC_LOGICAL_GE – Mayor que o igual a.

El elemento de estructura de comparación tiene los siguientes valores enumerados válidos de comparación de máscara de bits:

CMC_COMPARISON_OR
CMC_COMPARISON_AND

CMC_COMPARISON_OR – El valor de propiedad es igual al resultado de aplicar el operador lógico OR a la máscara de bits.
CMC_COMPARISON_AND – El valor de propiedad es igual al resultado de aplicar el operador lógico AND a la máscara de bits.

4.10 Clave de clasificación de cursor

NOMBRE

Clave de clasificación de cursor (*cursor sort key*) – Definición de tipo para un tipo de datos de clave de clasificación de cursor CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct CMC_TAG_CURSOR_SORT_KEY {
    CMC_id                          property;
    CMC_enum                        order;
} CMC_cursor_sort_key;
```

DESCRIPCIÓN

Un valor de datos de este tipo es una clave de clasificación de cursor CMC. Una clave de clasificación de cursor CMC define el orden en que se clasifican los elementos de un contenedor cuando son enumerados por un cursor. Una implementación puede tener una matriz de claves de clasificación de cursor. Una clave de clasificación de cursor tiene los siguientes componentes:

1) *property* (propiedad): Especifica la propiedad en base a la cual se van a clasificar los elementos enumerados.

2) *order* (orden): Especifica el orden en el cual se van a clasificar los elementos enumerados. El orden de clasificación válido es uno de los siguientes:

CMC_SORT_DEFAULT
CMC_SORT_ASCEND
CMC_SORT_DESCEND

CMC_SORT_DEFAULT – Los elementos del contenedor no serán necesariamente clasificados, sino que se dejarán en su orden por defecto. El resultado de este orden es específico de la implementación.
CMC_SORT_ASCEND – Clasifica los elementos del objeto de contenedor en orden ascendente. Los objetos que no tienen la propiedad indicada por la clave de clasificación se colocan en último lugar.
CMC_SORT_DESCEND – Clasifica los elementos del objeto de contenedor en orden descendente. Los objetos que no tienen la propiedad indicada por la clave de clasificación se colocan en último lugar.

4.11 Tabla de despacho

NOMBRE

Tabla de despacho (*dispatch table*) – Definición de tipo para una estructura con punteros a las funciones de una implementación CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct {
CMC_extension                 *dispatch_table_extensions;

/* SEND */
CMC_return_code
(*cmc_send)(
   CMC_session_id                 session,
   CMC_message                    *message,
   CMC_flags                      send_flags,
   CMC_ui_id                      ui_id,
   CMC_extension                  *send_extensions
);

/* SEND DOCUMENT */
CMC_return_code
(*cmc_send_documents)(
   CMC_string                     recipient_addresses,
   CMC_string                     subject,
   CMC_string                     text_note,
   CMC_flags                      send_doc_flags,
   CMC_string                     file_paths,
   CMC_string                     file_names,
   CMC_string                     delimiter,
   CMC_ui_id                      ui_id
);

/* ACT ON */
CMC_return_code
(*cmc_act_on)(
   CMC_session_id                 session,
   CMC_message_reference          *message_reference,
   CMC_enum                       operation,
   CMC_flags                      act_on_flags,
   CMC_ui_id                      ui_id,
   CMC_extension                  *act_on_extensions
);

/* LIST */
CMC_return_code
(*cmc_list)(
   CMC_session_id                 session,
   CMC_string                     message_type,
   CMC_flags                      list_flags,
   CMC_message_reference          *seed,
   CMC_uint32                     *count,
   CMC_ui_id                      ui_id,
   CMC_message_summary            **result,
   CMC_extension                  *list_extensions
);

/* READ */
CMC_return_code
(*cmc_read)(
   CMC_session_id                 session,
   CMC_message_reference          *message_reference,
   CMC_flags                      read_flags,
   CMC_message                    **message,
   CMC_ui_id                      ui_id,
   CMC_extension                  *read_extensions
);
```

```
/* LOOK UP */
CMC_return_code
(*cmc_look_up)(
   CMC_session_id                 session,
   CMC_recipient                  *recipient_in,
   CMC_flags                      look_up_flags,
   CMC_ui_id                      ui_id,
   CMC_uint32                     *count,
   CMC_recipient                  **recipient_out,
   CMC_extension                  *look_up_extensions
);

/* FREE */
CMC_return_code
(*cmc_free)(
   CMC_buffer                     memory
);

/* LOGOFF */
CMC_return_code
(*cmc_logoff)(
   CMC_session_id                 session,
   CMC_ui_id                      ui_id,
   CMC_flags                      logoff_flags,
   CMC_extension                  *logoff_extensions
);

/* LOGON */
CMC_return_code
(*cmc_logon)(
   CMC_string                     service,
   CMC_string                     user,
   CMC_string                     password,
   CMC_object_identifier          character_set,
   CMC_ui_id                      ui_id,
   CMC_uint16                     caller_cmc_version,
   CMC_flags                      logon_flags,
   CMC_session_id                 *session,
   CMC_extension                  *logon_extensions
);

/* QUERY CONFIGURATION */
CMC_return_code
(*cmc_query_configuration)(
   CMC_session_id                 session,
   CMC_enum                       item,
   CMC_buffer                     reference,
   CMC_extension                  *config_extensions
);

/* FULL CMC */

/* COPY OBJECT */
CMC_return_code
(*cmc_copy_object)(
   CMC_object_handle              container,
   CMC_object_handle              source_object,
   CMC_object_handle              *new_object,
   CMC_extension                  *copy_object_extensions
);

/* ADD PROPERTIES */
CMC_return_code
(*cmc_add_properties)(
   CMC_object_handle              object,
   CMC_uint32                     number_properties,
   CMC_property                   *properties,
   CMC_extension                  *add_properties_extensions
);

/* COMMIT OBJECT */
CMC_return_code
(*cmc_commit_object)(
   CMC_object_handle              source_object,
   CMC_extension                  *commit_object_extensions
);
```

```
/* COPY OBJECT HANDLE */
CMC_return_code
(*cmc_copy_object_handle)(
   CMC_object_handle             source_object,
   CMC_object_handle             *new_object,
   CMC_extension                 *copy_object_handle_extensions
);

/* CREATE DERIVED MESSAGE OBJECT */
CMC_return_code
(*cmc_create_derived_message_object)(
   CMC_object_handle             original_message,
   CMC_enum                      derived_action,
   CMC_boolean                   inherit_contents,
   CMC_object_handle             *derived_message,
   CMC_boolean                   modified_message,
   CMC_extension                 *create_derived_object_extensions
);

/* DELETE OBJECTS */
CMC_return_code
(*cmc_delete_objects)(
   CMC_uint32                    number_objects,
   CMC_object_handle             *object,
   CMC_extension                 *delete_objects_extensions
);

/* DELETE PROPERTIES */
CMC_return_code
(*cmc_delete_properties)(
   CMC_object_handle             object,
   CMC_uint32                    number_properties,
   CMC_id                        *property_ids,
   CMC_extension                 *delete_properties_extensions
);

/* GET ROOT HANDLE */
CMC_return_code
(*cmc_get_root_handle)(
   CMC_session_id                session,
   CMC_object_handle             *root_object_handle,
   CMC_extension                 *get_root_handle_extensions
);

/* IDENTIFIER TO NAME */
CMC_return_code
(*cmc_identifier_to_name)(
   CMC_id                        identifier,
   CMC_name                      *name,
   CMC_extension                 *identifier_to_name_extensions
);

/* LIST CONTAINED PROPERTIES */
CMC_return_code
(*cmc_list_contained_properties)(
   CMC_cursor_handle             *cursor,
   CMC_sint32                    *number_object,
   CMC_sint32                    *number_properties,
   CMC_id                        *property_ids,
   CMC_property                  **properties,
   CMC_extension                 *list_contained_properties_extensions
);

/* LIST NUMBER MATCHED */
CMC_return_code
(*cmc_list_number_matched)(
   CMC_cursor_handle             *cursor,
   CMC_uint32                    *number_matches,
   CMC_extension                 *list_number_matched_extensions
);
```

```
/* LIST OBJECTS */
CMC_return_code
(*cmc_list_objects)(
   CMC_cursor_handle             *cursor,
   CMC_sint32                    *number_objects,
   CMC_object_handle             *objects,
   CMC_extension                 *list_objects_extensions
);

/* LIST PROPERTIES */
CMC_return_code
(*cmc_list_properties)(
   CMC_object_handle             *object,
   CMC_uint32                    *number_properties,
   CMC_id                        *property_ids,
   CMC_extension                 *list_properties_extensions
);

/* NAME TO IDENTIFIER */
CMC_return_code
(*cmc_name_to_identifier)(
   CMC_name                      name,
   CMC_id                        *identifier,
   CMC_extension                 *name_to_identifier_extensions
);

/* OPEN CURSOR */
CMC_return_code
(*cmc_open_cursor)(
   CMC_object_handle             object,
   CMC_cursor_restriction        *restrictions,
   CMC_uint32                    number_sort_orders,
   CMC_cursor_sort_key           *sort_keys,
   CMC_cursor_handle             *cursor,
   CMC_extension                 *open_cursor_extensions
);

/* OPEN OBJECT HANDLE */
CMC_return_code
(*cmc_open_object_handle)(
   CMC_session_id                session,
   CMC_object_handle             *new_object,
   CMC_id                        object_class,
   CMC_extension                 *open_object_handle_extensions
);

/* READ CURSOR */
CMC_return_code
(*cmc_read_cursor)(
   CMC_cursor_handle             *cursor,
   CMC_uint32                    *position_numerator,
   CMC_uint32                    *position_denominator,
   CMC_extension                 *read_cursor_extensions
);

/* READ PROPERTIES */
CMC_return_code
(*cmc_read_properties)(
   CMC_object_handle             object,
   CMC_uint32                    *number_properties,
   CMC_id                        *property_ids,
   CMC_property                  **properties,
   CMC_extension                 *read_properties_extensions
);

/* READ PROPERTY COSTS */
CMC_return_code
(*cmc_read_property_costs)(
   CMC_object_handle             object,
   CMC_uint32                    *number_properties,
   CMC_id                        *property_ids,
   CMC_enum                      *costs,
   CMC_extension                 *read_property_costs_extensions
);
```

```
/* RESTORE OBJECT */
CMC_return_code
(*cmc_restore_object)(
   CMC_object_handle             container,
   CMC_string                    file_specification,
   CMC_object_handle             *restored_object,
   CMC_flags                     restore_flags,
   CMC_extension                 *restore_object_extensions
);

/* SAVE OBJECT */
CMC_return_code
(*cmc_save_object)(
   CMC_object_handle             object,
   CMC_string                    file_specification,
   CMC_flags                     save_flags,
   CMC_extension                 *save_object_extensions
);

/* SEND MESSAGE OBJECT */
CMC_return_code
(*cmc_send_message_object)(
   CMC_object_handle             message_to_send,
   CMC_extension                 *send_message_object_extensions
);

/* UPDATE CURSOR POSITION */
CMC_return_code
(*cmc_update_cursor_position)(
   CMC_cursor_handle             *cursor,
   CMC_uint32                    position_numerator,
   CMC_uint32                    position_denominator,
   CMC_extension                 *update_cursor_position_extensions
);

/* UPDATE CURSOR POSITION WITH SEED */
CMC_return_code
(*cmc_update_cursor_position_with_seed)(
   CMC_cursor_handle             cursor,
   CMC_object_handle             seed,
   CMC_extension                 *update_cursor_position_with_seed_extensions
);

/* CHECK EVENT */
CMC_return_code
(*cmc_check_event)(
   CMC_session_id                session,
   CMC_event                     event_type,
   CMC_uint32                    minimum_timeout,
   CMC_buffer                    check_event_data,
   CMC_buffer                    *callback_data,
   CMC_extension                 *check_event_extensions
);

/* REGISTER EVENT */
CMC_return_code
(*cmc_register_event)(
   CMC_session_id                session,
   CMC_event                     event_type,
   CMC_callback                  callback,
   CMC_buffer                    register_data,
   CMC_extension                 *register_event_extensions
);

/* UNREGISTER EVENT */
CMC_return_code
(*cmc_unregister_event)(
   CMC_session_id                session,
   CMC_flags                     event_type,
   CMC_callback                  callback,
   CMC_buffer                    unregister_data,
   CMC_extension                 *unregister_event_extensions
);
```

```
/* CALL CALLBACKS */
CMC_return_code
(*cmc_call_callbacks)(
   CMC_session_id                 session,
   CMC_event                      event_type,
   CMC_extension                  *call_callbacks_extensions
);

/* EXPORT STREAM */
CMC_return_code
(*cmc_export_stream)(
   CMC_stream_handle              stream,
   CMC_string                     file_specification,
   CMC_uint32                     count,
   CMC_flags                      export_flags,
   CMC_extension                  *export_stream_extensions
);

/* IMPORT FILE TO STREAM */
CMC_return_code
(*cmc_import_file_to_stream)(
   CMC_stream_handle              stream,
   CMC_string                     file_specification,
   CMC_uint32                     file_offset,
   CMC_extension                  *import_file_to_stream_extensions
);

/* OPEN STREAM */
CMC_return_code
(*cmc_open_stream)(
   CMC_object_handle              object,
   CMC_property                   *property,
   CMC_enum                       operation,
   CMC_stream_handle              **stream,
   CMC_extension                  *open_stream_extensions
);

/* READ STREAM */
CMC_return_code
(*cmc_read_stream)(
   CMC_stream_handle              stream,
   CMC_uint32                     *count,
   CMC_buffer                     content_information,
   CMC_extension                  *read_stream_extensions
);

/* SEEK STREAM */
CMC_return_code
(*cmc_seek_stream)(
   CMC_stream_handle              stream,
   CMC_enum                       operation,
   CMC_uint32                     *location,
   CMC_extension                  *seek_stream_extensions
);

/* WRITE STREAM */
CMC_return_code
(*cmc_write_stream)(
   CMC_stream_handle              *stream,
   CMC_uint32                     *count,
   CMC_buffer                     *content_information,
   CMC_extension                  *write_stream_extensions
);

/* GET LAST ERROR */
CMC_return_code
(*cmc_get_last_error)(
   CMC_session_id                 session,
   CMC_object_handle              objRef,
   CMC_string                     **error_buffer,
   CMC_extension                  *get_last_error_extensions
);

} CMC_dispatch_table;
```

```
/* BIND IMPLEMENTATION     */
CMC_return_code
cmc_bind_implementation (
   CMC_guid                      implementation_name,
   CMC_dispatch_table            **dispatch_table,
   CMC_extension                 *cmc_bind_extensions
);

/* UNBIND IMPLEMENTATION */
CMC_return_code
cmc_unbind_implementation (
   CMC_guid                      implementation_name,
   CMC_extension                 *cmc_unbind_implementation_extensions
);
```

DESCRIPCIÓN

Un valor de datos de este tipo es una tabla de despacho para una implementación CMC. La tabla de despacho incluye una entrada para cada función en una implementación CMC. Para el uso de la tabla de despacho, véanse los ejemplos en C.2 (bind.c y cmc_bind.c).

4.12 Enumerado

NOMBRE

Enumerado (*enumerated*) – Definición de tipo para un valor de datos enumerados.

DECLARACIÓN EN LENGUAJE C

```
typedef  CMC_sint32  CMC_enum;
```

DESCRIPCIÓN

Un valor de datos de este tipo contiene un valor seleccionado de una lista enumerada.

4.13 Eventos

NOMBRE

Eventos (*events*) – Definición de tipo para un evento CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef   CMC_uint32       CMC_event;
```

DESCRIPCIÓN

Un valor de datos de este tipo contiene 32 bits de evento. Los eventos no documentados están reservados. Los bits de evento puestos a cero se dicen que están "despejados" ("*clear*"). Los bits de evento de valor diferente de cero se dicen que están "fijados" ("*set*"). Los bits de evento no especificados tienen siempre que estar despejados.

Fijado: Han llegado nuevos mensajes a un contenedor de mensajes.

Despejado: No ha llegado ningún nuevo mensaje a un contenedor de mensajes.

En esta Recomendación, el único tipo de evento válido es:

CMC_EVENT_NEW_MESSAGES

4.14 Extensión

NOMBRE

Extensión (*extension*) – Definición de tipo para una estructura de extensión CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct {
    CMC_uint32      item_code;
    CMC_uint32      item_data;
    CMC_buffer      item_reference;
    CMC_flags           extension_flags;
} CMC_extension;
```

DESCRIPCIÓN

Un valor de datos de este tipo es una extensión. Se utiliza la misma estructura de extensión para especificar y recibir información de extensión relacionada con llamadas a funciones CMC *y* estructuras de datos CMC.

En general, las llamadas a funciones *y* las estructuras de datos pueden permitir extensiones de entrada y extensiones de salida, expresándose implícitamente el sentido de entrada o salida por el código de ítem de extensión. Las extensiones de entrada pueden hacer referencia a un almacenamiento asignado por la aplicación y las extensiones de salida pueden hacer referencia a un almacenamiento asignado por el servicio CMC. Por ejemplo, algunas implementaciones de **cmc_act_on()** podrían permitir que los mensajes parcialmente completados se guardaran en el apartado de entrada, para ulterior lectura y envío, utilizando la extensión CMC_X_COM_SAVE_MESSAGE para introducir la estructura de mensaje y recibir en retorno la referencia de mensaje resultante. Para la lista completa de las extensiones de mensajes comunes especificadas en esta Recomendación, véanse 4.11 y 4.14.

En el caso de matrices de extensiones CMC que pueden contener un almacenamiento de extensión de salida asignado por el servicio CMC, los llamantes tienen que utilizar **cmc_free()** para liberar el puntero retornado en el campo de referencia de ítem. Estas estructuras se identifican por la bandera de salida CMC_EXT_OUTPUT fijada y un valor de referencia de ítem diferente de NULL. Los llamantes solicitan explícitamente extensiones de funciones de salida cuando invocan funciones, fijando el código de ítem de extensión apropiado. Todas las subestructuras contenidas en la memoria asignada serán liberadas cuando se libere el puntero a la estructura de base.

Las extensiones de datos no tienen que ser liberadas explícitamente, pues se liberan con la estructura en que están contenidas. Por ejemplo, la matriz de extensiones de mensajes resultante de la función **cmc_read()** se libera implícitamente cuando se invoca **cmc_free()** para liberar la estructura de mensaje circundante.

Una extensión tiene los siguientes componentes:

1) *item_code* (código de ítem): Un código que identifica unívocamente la extensión en cuestión.

2) *item_data* (datos de ítem): Según sea el código de ítem, los datos de ítem pueden contener la longitud del valor de ítem, el propio valor de ítem, u otra información sobre el ítem. La especificación de la extensión describe cómo debe interpretarse este campo.

3) *item reference* (referencia de ítem): Según sea el código de ítem, la referencia de ítem puede contener un puntero al lugar en que está almacenado el valor de ítem, o NULL si no hay un correspondiente almacenamiento de ítem. La especificación de la extensión describe cómo debe interpretarse este campo.

4) *extension flags* (banderas de extensión): Bits para atributos booleanos. Los 16 bits superiores están reservados para definición por la especificación CMC. De estos bits, todos los que no se utilicen tienen que estar despejados. Los 16 bits inferiores de bandera están reservados para definición por la extensión.

   a) CMC_EXT_REQUIRED

      Fijado: Retorna un error si la extensión no puede ser soportada.

      Despejado: Permitir el soporte de "mejor esfuerzo", incluida la ausencia de soporte, de esta extensión.

   b) CMC_EXT_OUTPUT

      Fijado: En extensiones de salida indica que esta extensión contiene un puntero a memoria asignada por la implementación CMC, que debe liberarse con **cmc_free()**.

      Despejado: La implementación no asignó memoria para la extensión que la aplicación debe liberar. Esta bandera está siempre despejada en extensiones de datos, como se ha dicho anteriormente.

   c) CMC_EXT_LAST_ELEMENT

      Fijado: Identifica la última estructura en una matriz de tales estructuras. Debe estar al final de la matriz de extensiones.

      Despejado: Este elemento no es el último de la matriz.

4.15 Banderas

NOMBRE

Banderas (*flags*) – Definición de tipo para una bandera CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef   CMC_uint32       CMC_flags;
```

DESCRIPCIÓN

Un valor de datos de este tipo contiene 32 bits de bandera. El significado de los bits depende del contexto en que se utiliza el valor de datos de las banderas. Las banderas no documentadas están reservadas. Las banderas puestas a cero se dicen que están "despejadas" (*clear*). Las banderas fijadas a valores diferentes de cero se dicen que están "fijadas" (*set*). Las banderas no especificadas deben siempre estar despejadas.

4.16 GUID

NOMBRE

GUID – Definición de tipo para una estructura de identificador globalmente único en la CMC (GUID, *globally unique identifier*).

DECLARACIÓN EN LENGUAJE C

```
typedef CMC_string     CMC_guid
```

DESCRIPCIÓN

Un valor de datos de este tipo es un identificador globalmente único. El formato de la cadena se ajusta al texto de identificador público formal de ISO 9070 para garantizar su unicidad. Los GUID de la CMC tienen el siguiente formato:

–//XAPIA/CMC/*name type*/NONSGML *name*//EN

donde *name type* es el tipo del nombre y *name* es el nombre del objeto a que se asigna el GUID. Por ejemplo, la clase de objeto CONTENT ITEM es:

```
–//XAPIA/CMC/OBJECT CLASS//NONSGML Content Item//EN
```

Algunos de los valores del GUID de la CMC pueden definirse en términos de un identificador de objeto (OID, *Object Identifier*) para la interconexión de sistemas abiertos de ISO. El OID puede estar encapsulado en un identificador público formal de ISO 9070. La encapsulación en un identificador público formal (FPI, *formal public identifier*) se efectúa como sigue:

```
–//XAPIA/CMC/OID//NONSGML <oid>//EN
```

donde <**oid**> es la forma numérica del OID para la OSI definido por el tipo de datos de identificador de objeto en 4.24.

4.17 Identificador

NOMBRE

Identificador (*identifier*) – Definición de tipo para un identificador único específico de la implementación.

DECLARACIÓN EN LENGUAJE C

```
typedef system-defined, e.g. uint32     CMC_id;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un identificador único específico de la implementación. Este tipo de datos se utiliza para los identificadores localmente únicos como son el identificador de propiedad y el identificador de clase de objeto.

4.18 Fecha y hora ISO

NOMBRE

Fecha y hora ISO (*ISO date and time*) – Definición de tipo para un valor de datos de fecha y hora con el formato ISO 8601.

DECLARACIÓN EN LENGUAJE C

```
typedef CMC_string     CMC_date_time;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un valor de fecha y hora combinado con la representación de la fecha y hora del día de ISO 8601. El formato de este tipo de datos admite la hora del día representada sea como hora local u hora de reloj para uso público localmente, sea como tiempo universal coordinado (UTC, *coordinated universal time*) o la escala horaria mantenida por el Bureau International de l'Heure en que se basa la difusión coordinada de frecuencias patrón y señales horarias, sea como la diferencia entre la hora local y UTC.

El valor de datos es una concatenación de las representaciones de **fecha** y **hora**. El carácter [T] se utiliza como designador de tiempo para indicar el comienzo de la representación de hora del día en la expresión de cadena combinada de fecha y hora del día. Si el tiempo (u hora) se expresa en UTC, el carácter [Z] se utiliza como designador de zona horaria para UTC. Si el designador de zona horaria está ausente, la hora es hora local. Por ejemplo, ccyymmddThhmmssZ, donde [cc] es la cadena de siglo [yy] es la cadena de año [mm] es la cadena de mes, [dd] es la cadena de día del mes, [hh] es la cadena de hora en el formato de 24 horas, [mm] es la cadena de minuto para los minutos transcurridos después de la hora, y [ss] es la cadena de segundo para los segundos transcurridos después de la cadena de minuto.

Para expresar la hora como la diferencia entre la hora local y UTC, la fecha y la hora se representan por una de las cadenas ccyymmddThhmmss+hhmm, ccyymmddThhmmss+hh, ccyymmddThhmmss–hhmm, o ccyymmddThhmmss–hh, donde [cc] es la cadena de siglo, [yy] es la cadena de año, [mm] es la cadena de mes, [dd] es la cadena de día del mes, [hh] es la cadena de hora en el formato de 24 horas, [mm] es la cadena de minuto para los minutos transcurridos después de la cadena de hora, y [ss] es la cadena de segundo para los segundos transcurridos después de la cadena de minuto. El designador de zona horaria está ausente y la cadena de fecha y hora se concatena con la diferencia de hora y minuto, o diferencia de hora con respecto a UTC. La diferencia entre la hora local y UTC se expresa en horas y minutos, o en horas solamente, independientemente de la precisión de la expresión de hora local asociada con ella. Se expresa como positiva (es decir, precedida de un signo más [+]) si la hora local está más adelantada que UTC y como negativa (es decir, precedida de un signo menos [–]) si está más atrasada que UTC. Por ejemplo, 19850414T152746+0100 correspondería a la fecha 14 de abril de 1985 y a la hora 15 horas 27 minutos 46 segundos, como hora local en un punto que normalmente tiene una hora de adelanto con respecto a UTC. La cadena 19850414T152746–05 correspondería a la fecha 14 de abril de 1985 y a la hora 15 horas 27 minutos y 46 segundos hora local en un punto que normalmente tiene cinco horas de atraso con respecto a UTC.

1) **date** (fecha) – La fecha calendario, expresada como la representación completa, formato básico, definidos en ISO 8601, 5.2.1.1. Por ejemplo, 14 de abril de 1985 se representaría por la cadena 19850414.

2) **time** (hora, o tiempo) – La hora del día expresada sea como hora local, tiempo universal coordinado (UTC) equivalente, o diferencia de hora local con respecto a UTC. El formato de hora es la representación completa, formato básico, definidos en ISO 8601 5.3.3 y 5.3.3.1. Por ejemplo, el tiempo UTC de 23 horas, 20 minutos y 30 segundos se representaría por la cadena 232030Z. La hora local 10 minutos y 15 segundos pasadas las 12 horas se representaría por la cadena 121510. La misma hora local como diferencia con respecto a UTC se representaría por la cadena 121510–06 ó 121510–0600 si la hora local tiene seis horas de atraso con respecto a UTC.

4.19 Mensaje

NOMBRE

Mensaje (*message*) – Definición de tipo para una estructura de mensaje CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct {
    CMC_message_reference    *message_reference;
    CMC_string               message_type;
    CMC_string               subject;
    CMC_time                 time_sent;
    CMC_string               text_note;
    CMC_recipient            *recipients;
    CMC_attachment           *attachments;
    CMC_flags                message_flags;
    CMC_extension            *message_extensions;
} CMC_message;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un mensaje. Esta estructura de datos se incluye para proporcionar soporte para las implementaciones CMC 1.0 y CMC simple. Un mensaje tiene los siguientes componentes:

1) *message_reference* (referencia de mensaje): Identifica el mensaje. La referencia de mensaje es única en un apartado de correo.

2) *message_type* (tipo de mensaje): Cadena que identifica el tipo de mensaje. Pueden utilizarse tres identificadores de cadena diferentes:

   a) Identificadores de objeto – Utilizados para tipos identificados por identificadores de objetos, definidos en la Recomendación X.208.

   b) Valores registrados CMC – Utilizados para tipos definidos en esta Recomendación.

   c) Valores definidos bilateralmente – Utilizados para tipos definidos que no están registrados.

NOTA – No se garantiza que los valores definidos bilateralmente sean únicos.

A continuación se indica el formato de cada tipo. Espacio blanco puede ser cualquier combinación de caracteres de tabulador o de espacio. "*" indica que uno o más de los elementos indicados (separados por espacio en blanco) son válidos. Las cadenas entre comillas tienen el mismo valor independientemente de que estén escritas en mayúsculas o minúsculas.

```
message_type_value      ::= oid | cmc_reg | bilat_def
oid                     ::= "OID:" object_identifier
cmc_reg                 ::= "CMC:" cmc_registered_value
bilat_def               ::= "BLT:" string
object_identifier       ::= object_id_component*
object_id_component     ::= integer
cmc_registered_value    ::= "IPM" | "IP RN" | "IP NRN" | "DR" | "NDR" | "REPORT"
```

Estos valores registrados se definen como sigue:

"CMC: IPM" Mensaje interpersonal: Un mensaje interpersonal es un mensaje en forma de memorándum que contiene una lista de recibientes, un asunto facultativo, una nota textual facultativa, y cero o más añadiduras. La estructura "Message" está optimizada para acomodar un mensaje de tipo IPM.

"CMC: IP RN" Notificación de recepción para un mensaje interpersonal: Una notificación de recepción indica que un mensaje ha sido leído por el recibiente.

"CMC: IP NRN" Notificación de no recepción para un mensaje interpersonal: Una notificación de no recepción indica que un mensaje ha sido retirado del apartado de correos del recibiente sin haber sido leído. Por ejemplo, el mensaje ha sido descartado por el usuario o el servicio, o ha sido reenviado automáticamente a otro recibiente.

"CMC: DR" Informe de entrega: Un informe de entrega indica que el servicio pudo entregar un mensaje al recibiente.

"CMC: NDR" Informe de no entrega: Un informe de no entrega indica que el servicio no pudo entregar un mensaje al recibiente.

"CMC: REPORT" Informes de entrega y de no entrega cuando el mensaje está destinado a múltiples recibientes: Indica que un servicio de mensajería subyacente puede entregar el mensaje a algunos recibientes, pero no a otros.

El formato de estos tipos de mensajes en las estructuras definidas depende de los protocolos de mensajería empleados por el servicio de mensajería. A menudo, los mensajes que no son IPM adoptan la forma de un mensaje generado por programa, que sigue un formato como el de un memorándum (similar al de un IPM), pero que tiene por finalidad transportar información sobre un mensaje anteriormente enviado.

NOTA – Estos tipos de mensajes corresponden a los tipos de mensajes X.400, pero pueden utilizarse con servicios de mensajería diferentes de X.400. Por tanto, estos tipos de mensajes CMC están destinados a aplicarse de manera general y no específicamente a X.400.

Son ejemplos de identificadores válidos:

OID: 1 2 840 113556 3 2 850

CMC: IPM

BLT: mi tipo de mensaje especial

Se define también una forma canónica de estos tipos para permitir que una aplicación compare fácilmente estas cadenas. La implementación CMC retornará siempre la forma canónica. En la forma canónica:

1) todos los espacios en blanco consecutivos se convierten en un solo espacio en blanco, y todos los elementos serán separados por un espacio en blanco;

2) los identificadores de tipos (es decir OID, CMC, BLT) se convierten a la escritura con mayúsculas.

Algunas implementaciones CMC sólo soportarán el tipo de mensaje interpersonal (CMC: IPM). En estas implementaciones, otros tipos de mensajes pueden tratarse como mensajes IPM, o pueden generar un error.

El comportamiento de la implementación cuando aparecen cadenas que no corresponden a uno de estos formatos no está definido.

3) *subject* (asunto): Cadena del asunto del mensaje.

4) *time_sent* (hora de envío): Fecha/hora en que se envió/(depositó) el mensaje.

5) *text_note* (nota textual o nota de texto): Cadena de nota textual del mensaje. Si el valor es NULL, no hay nota textual. Si no se fija la bandera CMC_TEXT_NOTE_AS_FILE, la nota textual está en la primera añadidura.

   El formato de la nota textual, independientemente de que se haya pasado en memoria o en un fichero, es una secuencia de párrafos, cada un de los cuales es terminado por el terminador de línea apropiado para la plataforma (CR para Macintosh, LF para Unix, CR/LF para DOS y Windows, etc.). La implementación CMC puede dividir las líneas largas y llevarlas sucesivamente a la línea siguiente (formando párrafos).

NOTA – La fidelidad no está garantizada (por ejemplo, un párrafo largo puede ser retornado por las funciones CMC de lectura como una serie de párrafos más cortos).

6) *recipients* (recibientes): Puntero al primer elemento de una matriz de recibientes del mensaje.

7) *attachments* (añadiduras): Puntero al primer elemento de una matriz de añadiduras al mensaje.

8) *message_flags* (banderas de mensaje): Bits para atributos booleanos. Los bits no utilizados deben estar despejados.

   a) CMC_MSG_READ

      Fijado: Se ha leído el mensaje

      Despejado: No se ha leído el mensaje.

   b) CMC_MSG_TEXT_NOTE_AS_FILE

      Fijado: El campo de nota textual se ignora y el texto de la nota textual está contenido en el fichero a que hace referencia la primera añadidura.

      Despejado: El texto de la nota textual está contenido en la cadena de nota de textual.

   c) CMC_MSG_UNSENT

      Fijado: El mensaje no ha sido enviado (es decir, es un proyecto o borrador de mensaje). Este tipo de mensaje puede crearse con la extensión CMC_X_COM_SAVE_MESSAGE.

      Despejado: Se ha enviado el mensaje.

   d) CMC_MSG_LAST_ELEMENT

      Fijado: Identifica la última estructura de una matriz de esas estructuras.

      Despejado: El elemento en cuestión no es el último elemento de la matriz.

9) *message_extensions* (extensiones de mensaje): Puntero al primer elemento de una matriz de extensiones por cada mensaje.

4.20 Referencia de mensaje

NOMBRE

Referencia de mensaje (*message reference*) – Definición de tipo para una estructura de referencia de mensaje CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef   CMC_counted_string  CMC_message_reference;
```

DESCRIPCIÓN

Un valor de datos de este tipo es una cadena contada que es el asa de mensaje utilizada por el apartado de correos. Esta estructura de datos se incluye para el soporte de las implementaciones CMC 1.0 y CMC simple. La validez de una referencia de mensaje sólo está garantizada durante el tiempo de vida de la sesión y su correspondencia con cualquier otro identificador de mensaje utilizado no está garantizada por el sistema de mensajería subyacente. Durante el tiempo de vida de la sesión puede ser copiada por el programa de aplicación.

4.21 Sumario de mensaje

NOMBRE

Sumario de mensaje (*message summary*) – Definición de tipo para una estructura de sumario de mensaje CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct {
    CMC_message_reference    *message_reference;
    CMC_string               message_type;
    CMC_string               subject;
    CMC_time                 time_sent;
    CMC_uint32               byte_length;
    CMC_recipient            *originator;
    CMC_flags                summary_flags;
    CMC_extension            *message_summary_extensions;
} CMC_message_summary;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un sumario de mensaje. Esta estructura de datos se incluye para el soporte de las implementaciones CMC 1.0 y CMC simple. Un sumario de mensaje tiene los siguientes componentes:

1) *message_reference* (referencia de mensaje): Véase la definición en la estructura de mensaje.

2) *message_type* (tipo de mensaje): Véase la definición en estructura de mensaje.

3) *subject* (asunto): Véase la definición en la estructura de mensaje.

4) *time_sent* (hora de envío): Véase la definición en la estructura de mensaje.

5) *byte_length* (longitud en octetos): Tamaño del mensaje. El valor debe incluir todas las características asociadas del mensaje (añadiduras, campos de sobre y de encabezamiento, etc.). Las implementaciones pueden retornar un valor aproximado, o la constante CMC_LENGTH_UNKNOWN si la longitud se desconoce o no está disponible.

6) *originator* (originador): Originador del mensaje

7) *summary_flags* (banderas de sumario): Bits para atributos booleanos. Los bits no utilizados deben estar despejados.

   a) CMC_SUM_READ

      Fijado: El mensaje se ha leído.

      Despejado: El mensaje no se ha leído.

   b) CMC_SUM_UNSENT

      Fijado: El mensaje no se ha enviado (es decir, es un proyecto o borrador de mensaje).

      Despejado: El mensaje se ha enviado.

   c) CMC_SUM_LAST_ELEMENT

      Fijado: Identifica la última estructura de una matriz de esas estructuras.

      Despejado: El elemento en cuestión no es el último de la matriz.

   d) CMC_SUM_HAS_ATTACHMENTS

      Fijado: El mensaje tiene añadiduras.

      Despejado: El mensaje no tiene añadiduras.

8) *message_summary_extensions* (extensiones de sumario de mensajes): Puntero al primer elemento de una matriz de extensiones de sumario por cada mensaje.

4.22 Nombre

NOMBRE

Nombre (*name*) – Definición de tipo para un nombre CMC 2.0 único.

DECLARACIÓN EN LENGUAJE C

```
typedef CMC_string     CMC_name;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un nombre único. La cadena se formatea de acuerdo con el texto de identificador público formal de ISO 9070 para garantizar la unicidad. Los nombres CMC tienen el siguiente formato:

`–//XAPIA/CMC/`*`name type`*`//NONSGML `*`name`*`//EN`

donde *name type* es el tipo de nombre *name* es el nombre del objeto. Por ejemplo, la clase de objeto CONTENT ITEM es:

```
–//XAPIA/CMC/OBJECT CLASS//NONSGML Content Item//EN
```

4.23 Asa de objeto

NOMBRE

Asa de objeto (*object handle*) – Definición de tipo para una estructura de asa de objeto CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef system-defined, e.g. uint32     CMC_object_handle;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un asa de objeto opaca. Las asas de objetos CMC son únicas para el servicio de mensajes. Las asas son persistentes durante la existencia de la sesión o hasta que sean destruidas. El asa proporciona el contexto a un objeto CMC. El asa encapsula el identificador de sesión. Para copiar un asa de objeto se utiliza la función **cmc_copy_object_handle()**.

La noción de "ningún asa" debe almacenarse en un asa de objeto. En este caso se utiliza la constante CMC_NULL_HANDLE, que está definida por el sistema.

4.24 Identificador de objeto

NOMBRE

Identificador de objeto (*object identifier*) – Definición de tipo para una estructura de identificador de objeto CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef   CMC_string      CMC_object_identifier;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un identificador de objeto, definido en la Recomendación X.208. Esta estructura de datos se incluye para el soporte de implementaciones CMC 1.0 y CMC simple. Es globalmente inequívoco. Su sintaxis, tal como se utiliza en esta Recomendación, deberá concordar con la forma de número en la Recomendación X.208. Esta sintaxis es:

```
object_identifier          ::= object_id_component*
```

object_id_component ::= integer

Un ejemplo de un identificador de objeto es:

1 2 840 113556 3 2 850

NOTA – El formato de la cadena de identificador de objeto es el mismo utilizado en el tipo de mensaje identificador de objeto (OID).

4.25 Datos opacos

NOMBRE

Datos opacos (*opaque data*) – Definición de tipo para un valor de datos opacos.

DECLARACIÓN EN LENGUAJE C

```
typedef struct CMC_TAG_OPAQUE_DATA {
    CMC_size                    size;
    CMC_byte                    *data;
} CMC_opaque_data;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un valor de datos opacos. La estructura de datos opacos consta de los siguientes componentes:

1) **size** (tamaño) – Específica el número de octetos de datos opacos señalado por el puntero **data**.

2) **data** (datos) – Un puntero a una matriz de valores representados por 8 bits. Estos datos no tienen una semántica explícita.

4.26 Propiedad

NOMBRE

Propiedad (*property*) – Definición de tipo para un tipo de datos de propiedad CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct CMC_TAG_PROPERTY{
    CMC_id                          property id;
    CMC_enum                            type;
    union {
        CMC_boolean                 CMC_pv_boolean;
        CMC_byte                        CMC_pv_byte;
        CMC_buffer                  CMC_pv_buffer;
        CMC_counted_string          CMC_pv_counted_string;
        CMC_enum                        CMC_pv_enum;
        CMC_extension                   CMC_pv_extension;
        CMC_float32                 CMC_pv_float32;
        CMC_float64                 CMC_pv_float64;
        CMC_flags                       CMC_pv_flags;
        CMC_guid                        CMC_pv_guid;
        CMC_iso_date_time       CMC_pv_iso_date_time;
        CMC_object_handle           CMC_pv_object_handle;
        CMC_opaque_data             CMC_pv_opaque_data;
        CMC_return_code             CMC_pv_return_code;
        CMC_sint16                  CMC_pv_sint16;
        CMC_sint32                  CMC_pv_sint32;
        CMC_string                  CMC_pv_string;
        CMC_time                        CMC_pv_time;
        CMC_uint16                  CMC_pv_uint16;
        CMC_uint32                  CMC_pv_uint32;
        CMC_array_boolean           CMC_pv_array_boolean;
        CMC_array_buffer            CMC_pv_array_buffer;
        CMC_array_counted_string CMC_pv_array_counted_string;
        CMC_array_enum              CMC_pv_array_enum;
        CMC_array_extension         CMC_pv_array_extension;
        CMC_array_float32       CMC_pv_array_float32;
        CMC_array_float64       CMC_pv_array_float64;
        CMC_array_guid              CMC_pv_array_guid;
        CMC_array_iso_date_time CMC_pv_array_iso_date_time;
        CMC_array_object_handle     CMC_pv_array_object_handle;
        CMC_array_opaque_data       CMC_pv_array_opaque_data;
        CMC_array_return_code       CMC_pv_array_return_code;
        CMC_array_sint16            CMC_pv_array_sint16;
        CMC_array_sint32            CMC_pv_array_sint32;
        CMC_array_string            CMC_pv_array_string;
        CMC_array_time              CMC_pv_array_time;
        CMC_array_uint16            CMC_pv_array_uint16;
        CMC_array_uint32            CMC_pv_array_uint32;
    } value
    } CMC_property;
```

DESCRIPCIÓN

Un valor de datos de este tipo es una propiedad de matriz CMC. Una propiedad es el método para especificar una información de contenido específica de una matriz CMC. Una propiedad tiene los siguientes componentes:

1) *id* (identificador): Identifica unívocamente la propiedad.

2) *type* (tipo): Específica el tipo de datos para la propiedad.

3) *value* (valor): Define el valor para la propiedad.

4) *property_extensions* (extensiones de propiedad): Puntero al primer elemento de una matriz de extensiones de propiedad.

4.27 Recibiente

NOMBRE

Recibiente (*recipient*) – Definición de tipo para la estructura originador/recibiente.

DECLARACIÓN EN LENGUAJE C

```
typedef struct {
    CMC_string      name;
    CMC_enum        name_type;
    CMC_string      address;
    CMC_enum        role;
    CMC_flags       recip_flags;
    CMC_extension   *recip_extensions;
} CMC_recipient;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un originador o un recibiente. Esta estructura de datos se incluye para el soporte de implementaciones CMC 1.0 y CMC simple. Esta estructura tiene los siguientes componentes:

1) *name* (nombre): Nombre para la visualización del recibiente. La interpretación del nombre primero como un individuo y después como un grupo, si tal individuo no se encuentra, o viceversa, se deja para que la determine la implementación cuando resuelva el nombre para obtener una dirección.

2) *name_type* (tipo de nombre): Tipo recibiente, enumerado:

CMC_TYPE_UNKNOWN (= 0)	Tipo de recibiente desconocido.
CMC_TYPE_INDIVIDUAL	El recibiente es un individuo.
CMC_TYPE_GROUP	El nombre es un grupo de recibientes.

NOTA – Es significativo solamente si el nombre está presente. La implementación lo fija en salida. En entrada puede utilizarse como una sugerencia para optimizar la resolución del nombre.

3) *address* (dirección): Dirección del recibiente que es aceptable por el servicio de mensajería subyacente. El formato de la cadena de dirección no está definido en esta Recomendación. Deberá admitir cualesquiera notaciones de cadenas soportadas por una implementación dada, tal como están configuradas en una instalación determinada. Los usuarios de extremo deben consultar al gestor de su servicio local para averiguar la notación o notaciones de cadenas que están soportadas por sus instalaciones.

4) *role* (rol, o papel): Rol de recibiente, enumerado:

CMC_ROLE_TO	Recibiente TO (recibiente primario).
CMC_ROLE_CC	Recibiente CC (recibiente de copia).
CMC_ROLE_BCC	Recibiente BCC (recibiente de copia ciega).
CMC_ROLE_ORIGINATOR	Originador de mensaje.
CMC_ROLE_AUTHORIZING_USER	Usuario autorizante de mensaje.
CMC_ROLE_REPLY_TO	Recibiente para la recepción de respuestas.

Un recibiente de copia (CC) puede convertirse automáticamente en un recibiente primario (TO) si el servicio de mensajería subyacente no admite los recibientes de copia. Los servicios que no admiten las copias ciegas deben rechazar los mensajes que las contengan. Para que un mismo recibiente pueda desempeñar más de un rol, tiene que haber múltiples inscripciones de ese recibiente con roles diferentes.

La implementación CMC debe retornar la matriz de recibientes en el orden siguiente, en salida. El originador debe ser el primer elemento de la matriz, seguido por los recibientes de respuestas (REPLY TO), los recibientes primarios (TO), los recibientes de copias (CC) y los recibientes de copias ciegas (BCC) agrupados en ese orden. El usuario autorizante, si existe, deberá ser el recibiente final en la matriz. No se requiere ordenación en entrada.

5) *recip_flags* (banderas de recibientes): Bits para atributos booleanos. Los bits no utilizados deben estar despejados.

   a) CMC-RECIP_IGNORE

      Fijado: Se ignora este recibiente (es útil para reutilizar la lista de recibientes de un mensaje entrante, para una respuesta).

      Despejado: No se ignora este recibiente.

   b) CMC_RECIP_LIST_TRUNCATED

      Fijado: Indica que no todas las estructuras de recibiente solicitadas fueron retornadas por el sistema. Esto sólo se utiliza en la función **cmc_look_up()** cuando no pudo retornar la lista completa de recibientes que concuerdan con el nombre de búsqueda. Esta bandera sólo se fija en la última estructura de la matriz.

      Despejado: Se retornó íntegramente la matriz de estructuras de recibientes.

   c) CMC_RECIP_LAST_ELEMENT

      Fijado: Identifica la última estructura de una matriz de estructuras de recibientes.

      Despejado: El elemento en cuestión no es el último de la estructura.

6) *recip_extensions* (extensiones de recibientes): Puntero al primer elemento de una matriz de extensiones por cada recibiente.

4.28 Informe

NOMBRE

Informe (*report*) – Definición de tipo para una combinación de estructura de informe y de estructura de informe de no entrega.

DECLARACIÓN EN LENGUAJE C

```
typedef struct {
    CMC_recipient       *msg_recipient;
    CMC_enum            report_type;
    CMC_time            delivered_time;
    CMC_uint32          reason_code;
    CMC_flags           report_flags;
} CMC_report;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un informe, o un informe de no entrega, o ambos. Esta estructura de datos se incluye para el soporte de implementaciones CMC 1.0 y CMC simple. Un informe tiene los siguientes componentes:

1) *report_type* (tipo de informe): Valor enumerado que identifica el tipo de informe. El tipo de informe puede ser:

   CMC_X400_DR ((CMC_enum) 0)

   CMC_X400_NDR ((CMC_enum) 1)

2) *delivered_time* (hora de entrega): Fecha/hora en que se entregó el mensaje original al recibiente. Es NULL para CMC_X400_NDR, u hora de entrega del informe para CMC_X400_DR.

3) *reason_code* (código de motivo, o razón): El motivo por el cual no se entregó un mensaje. El valor es ZERO para CMC_X400_DR o, el siguiente valor para CMC_X400_NDR:

   reason_code.<higher order 16 bits> = X.411.NonDeliveryReasonCode.

   reason_code.<lower order 16 bits> = X.411.NonDeliveryDiagnosticCode.

4) *report_flags* (banderas de informe): Bits para atributos booleanos. Los bits no utilizados tienen que estar despejados.

– CMC_REPORT_LAST_ELEMENT

Fijado: Identifica la última estructura de una matriz de esas estructuras.

Despejado: El elemento en cuestión no es el último de la matriz.

NOTA – La CMC define tipos de mensajes específicos para informes de entrega ("CMC:DR") e informes de no entrega ("CMC:NDR") que pueden ser tratados independientemente, pues se perciben como mensajes distintos. En la Recomendación X.400 tanto la información de entrega como la de no entrega se transporta sobre una base de información genérica de informe. Es posible que un informe X.400 contenga informes de entrega para algunos recibientes e informes de no entrega para otros recibientes, cuando un mensaje está destinado a múltiples recibientes controlados por un mismo MTA. Esto no corresponde bien con los tipos CMC:DR o CMC:NDR en salida (X.400 a CMC) porque la Recomendación X.400 no los percibe, ni los almacena como base de información distinta y, en consecuencia, no pueden ser tratados individualmente. Por tanto, se ha añadido un nuevo tipo de mensaje "CMC: REPORT" para satisfacer los requisitos del informe X.400.

4.29 Código de retorno

NOMBRE

Código de retorno (*return code*) – Definición de tipo para un valor retornado por todas las funciones CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef   CMC_uint32       CMC_return_code;
```

DESCRIPCIÓN

Un código de retorno se define como un valor representado por 32 bits. Un valor diferente de cero indica un error, indicándose el código de error por el valor retornado. Un valor de retorno de cero indica éxito. Los valores contenidos en los 16 bits de orden inferior están reservados para códigos de error definidos en esta Recomendación. Los valores contenidos en los 16 bits de orden superior están reservados para códigos de error definidos por la implementación, mientras que los 16 bits de orden inferior deben fijarse a un error CMC apropiado.

Los errores pueden resolverse dentro del ámbito de una llamada CMC utilizando, por ejemplo, diálogos disponibles a través de la interfaz de usuario. Si se inicia un dialogo para resolver el error, pero el error subsiste después de finalizado el dialogo, la bandera de un bit definida en CMC_ERROR_UI_DISPLAYED se fija en el error para indicar que ya se ha visualizado el error al usuario.

4.30 Identificador de sesión

NOMBRE

Identificador de sesión (*session id*) – Definición de tipo para un identificador de sesión CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef   system-defined, e.g. uint32CMC_session_id;
```

DESCRIPCIÓN

Identificador de sesión opaca. El contexto identificado por el identificador de sesión contiene información por cada sesión como la relativa al juego de caracteres que se está utilizando y las asas para la sesión o sesiones que estén abiertas con uno o más servicios de mensajería subyacentes. El identificador de sesión CMC se crea por la función de establecimiento de sesión (*logon*) de la CMC y se elimina por la función de terminación de sesión (*logoff*) de la CMC.

Para la definición apropiada para una plataforma específica, véase B.2.4.

4.31 Asa de tren

NOMBRE

Asa de tren (*stream handle*) – Definición de tipo para una estructura de asa de tren CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef system-defined, e.g. uint32 CMC_stream_handle;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un asa de un tren opaca. Las asas de trenes CMC son únicas con respecto al servicio de mensajes. Las asas mantienen su validez durante la existencia de la sesión o hasta que sean eliminadas. El asa proporciona el contexto a un tren de información de contenido. El tren encapsula al identificador de sesión y las asas de objetos. Las asas de trenes no pueden ser copiadas.

4.32 Cadena

NOMBRE

Cadena (*string*) – definición de tipo para una cadena de caracteres CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef   cmc_string*      CMC_string;
```

DESCRIPCIÓN

Un valor de datos de este tipo es una cadena. La matriz de tipo carácter a que se apunta se interpreta como una matriz de tipo carácter terminada por el carácter nulo, por defecto. Todas las implementaciones tienen que soportar cadenas terminadas en el carácter nulo. La anchura de un carácter y el correspondiente carácter nulo de terminación se determinan por el juego de caracteres elegidos.

Si una aplicación desea utilizar cadenas contadas en lugar de cadenas terminadas por el carácter nulo y la implementación CMC lo admite, la aplicación fijará la bandera CMC_COUNTED_STRING_TYPE cuando establezca la sesión. En este caso, se supondrá que los datos a que apunta CMC_string_ están en el formato de datos de cadena contada CMC. Si se establece una sesión implícitamente al invocar una función, esta bandera debe estar fijada en el parámetro flags (banderas).

Para determinar el juego de caracteres a que pertenecen los caracteres que forman la cadena, la implementación CMC examina el contexto de la sesión. Si no se ha creado un contexto de sesión antes de la llamada, se interpretará que la cadena utiliza el juego de caracteres por defecto de la implementación. La implementación deberá siempre tratar de establecer la correspondencia de todas las cadenas pasadas a la aplicación del cliente, con el juego de caracteres para la sesión.

4.33 Tiempo (u hora)

NOMBRE

Tiempo, u hora (*time*) – Definición de tipo para una estructura CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef struct {
    CMC_sint8        second;
    CMC_sint8        minute;
    CMC_sint8        hour;
    CMC_sint8        day;
    CMC_sint8        month;
    CMC_sint8        year;
    CMC_sint8        isdst;
    CMC_sint16       tmzone;
} CMC_time;
```

DESCRIPCIÓN

Un valor de datos de este tipo es un valor de tiempo (u hora). Esta estructura de datos se incluye para el soporte de implementaciones CMC 1.0 y CMC simple. Un valor de tiempo tiene los siguientes componentes:

1) *second* (segundo): Segundos; gama 0..59.

2) *minute* (minuto): Minutos; gama 0..59.

3) *hour* (hora): Horas a partir de la medianoche; gama 0..23.

4) *day* (día): Día del mes; gama 1..31.

5) *month* (mes): Meses a partir de enero; gama 0..11.

6) *year* (año): Años desde 1900.

7) *isdst*: Bandera de hora de verano; un valor diferente de cero representa hora de verano.

8) *tmzone*: Zona horaria en minutos con relación al tiempo medio de Greenwich. El valor definido, CMC_NO_TIMEZONE, indica que la zona horaria no está disponible.

Todos los valores de tiempo (u hora) se expresan en tiempo (u hora) local apropiado. Por ejemplo, el campo de hora de envío en las estructuras de mensaje CMC y de sumario de mensaje CMC se expresa en la hora local del emisor.

NOTA – Si el campo de zona horaria está fijado a cualquier valor diferente de CMC_NO_TIMEZONE, el valor de tiempo puede convertirse en la hora local del llamante, si bien la funcionalidad efectivamente empleada para la conversión está fuera del ámbito de la CMC.

4.34 Identificador de interfaz de usuario

NOMBRE

Identificador de interfaz de usuario (*user interface identifier*) – Definición de tipo para un asa de interfaz de usuario CMC.

DECLARACIÓN EN LENGUAJE C

```
typedef   system-defined, e.g. uint32CMC_ui_id;
```

DESCRIPCIÓN

Valor utilizado para pasar información de interfaz de usuario a funciones CMC. Por ejemplo, en un entorno basado en ventanas, éste sería el asa de la ventana progenitora para la aplicación llamante.

Un valor de NULL siempre es válido; el correspondiente comportamiento por defecto lo define la implementación.

NOTA – Las implementaciones CMC no están obligadas a proporcionar interfaz de usuario, y la provisión de una interfaz de usuario para una característica no implica necesariamente que haya interfaces de usuario disponibles para todas las características de la CMC.

Para la definición apropiada para una plataforma específica, véase B.2.4.

5 Propiedades de objetos

Esta cláusula define las propiedades de objetos para clases de objetos de la interfaz de programas de aplicación (API) para llamadas de mensajería común (CMC). Cada objeto es una colección de propiedades. Se definen aquí propiedades de objetos con el fin de uniformar su representación dentro de esta Recomendación.

Las definiciones de propiedades de objetos van precedidas de cuadros que recapitulan las propiedades para cada clase de objeto. Los siguientes cuadros de recapitulación de propiedades de objetos indican el nombre de la propiedad y el tipo de valor de todas las propiedades de objeto que figuran en las columnas uno y dos. La tercera columna proporciona una descripción de la propiedad. La cuarta columna indica posibles valores para cada propiedad. Los valores señalados con asteriscos son valores por defecto. Si ningún valor está marcado con un asterisco, la propiedad no tiene valor por defecto. La quinta columna expresa si la propiedad es obligatoria (M, *mandatory*) o facultativa (O, *optional*). La sexta columna expresa si la propiedad es de lectura solamente. Un "no" en esta columna significa que la propiedad puede ser modificada, actualizada o suprimida por una llamada a las funciones **cmc_update_properties(), cmc_add_properties(),** o **cmc_delete_properties()** respectivamente, a menos que se exprese otra cosa. La última columna específica el creador de la propiedad, que puede ser la implementación (I), el llamante (C) o cualquiera de los dos (E).

Se puede asociar valores por defecto a propiedades. Sin embargo, cuando una implementación crea un objeto, debe surtir el objeto con valores explícitos para todas las propiedades soportadas que tengan valores por defecto. Esto simplificará las enumeraciones de propiedades por las aplicaciones.

Cuadro 4/X.446 – Sumario de las propiedades del libro de direcciones CMC

Libro de direcciones					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Vástago permitido	booleano	CMC_TRUE, CMC_FALSE	O	No	CMC_FALSE
Comentario	cadena	Cualquier cadena válida	O	No	Ninguno
Ubicación	enum	LOCAL SERVER UNKNOWN[a)]	O	No	UNKNOWN[a)]
Nombre	cadena	Cualquier cadena válida	O	No	Cadena nula
Clase de objeto	enum	ADDRESS BOOK[b)]	M	Sí	No aplicable
Progenitor	asa de objeto	Cualquier asa de objeto válida	M, si está anidada	No	Ninguno
Nombre del servidor	cadena	Cualquier cadena válida	O	No	Ninguno
Compartido	booleano	CMC_TRUE, CMC_FALSE	O	No	CMC_FALSE
Tipo	enum	GLOBAL, PERSONAL[c)]	O	No	PERSONAL[b)]

a) Prefijo de valor "CMC_ADDRESS_BOOK_LOCATION_".

b) Prefijo de valor "CMC_OBJECT_TYPE_".

c) Prefijo de valor "CMC_ADDRESS_BOOK_TYPE_".

Cuadro 5/X.446 – Sumario de las propiedades del ítem de contenido CMC

Ítem de contenido					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Conjunto de caracteres	guid	GUID para cualquier conjunto de caracteres	O	No	Dependiente de la plataforma
Información de contenido	datos opacos	Cualquier dato	O	No	Ninguno
Tipo de contenido	guid	GUID para cualquier tipo de contenido	O	No	Ninguno
Hora de creación	iso_date_time	Cualquier fecha y hora ISO 8601	O	Sí	Ninguno
Tipo de codificación	guid	GUID para cualquier tipo de codificación	O	No	7-BIT[a)]
Directorio de ficheros	cadena	Cualquier directorio de ficheros válido	O	No	Ninguno
Nombre de fichero	cadena	Cualquier nombre de fichero válido	O	No	Ninguno
Número de ítem	uint32	Hasta un máximo definido por la implementación	M, para más de un ítem de contenido	No	Ninguno
Tipo de ítem	enum	NOTE ATTACHMENT ANNOTATION[b)]	O	No	NOTE
Último modificado	iso_date_time	Cualquier fecha y hora ISO 8601	O	Sí	Ninguno
Clase de objeto	enum	CONTENT ITEM[c)]	M	Sí	No aplicable
Posición de reproducción	uint32	Posición de octeto dentro del contenedor	O	No	Ninguno
Tamaño	uint32	Tamaño en octetos	O	No	Ninguno
Título	cadena	Cualquier cadena válida	O	No	Ninguno

a) Prefijo de valor "CMC_ADDRESS_BOOK_LOCATION_".

b) Prefijo de valor "CMC_IT_".

c) Prefijo de valor "CMC_OBJECT_TYPE_".

Cuadro 6/X.446 – Sumario de las propiedades de la lista de distribución CMC

Lista de distribución					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Dirección	cadena	Cualquier dirección válida	O	Sí	Ninguno
Comentario	cadena	Cualquier cadena válida	O	No	Ninguno
Hora de la última modificación	iso_date_time	Cualquier fecha y hora ISO 8601	O	Sí	Ninguno
Nombre	cadena	Cualquier cadena válida	M	No	Cadena nula
Clase de objeto	enum	DISTRIBUTION LIST[a)]	M	Sí	No aplicable
Progenitor	asa de objeto	Cualquier asa de objeto válida	M, si está anidada	No	Ninguno
Compartido	booleano	CMC_TRUE, CMC_FALSE	O	No	CMC_FALSE
a) Prefijo de valor "CMC_OBJECT_TYPE_".					

Cuadro 7/X.446 – Sumario de las propiedades del mensaje CMC

Mensaje					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
ID de aplicación	cadena	Cualquier cadena válida	O	No	Ninguno
Estado de mensaje de aplicación	banderas	Proyecto	O	No	Ninguno
Acción automática	banderas	CMC_AA_ DELETE	O	No	Ninguno
Hora de entrega diferida	iso_date_time	Cualquier fecha y hora ISO 8601	O	No	Ninguno
Estado de mensaje en entrada	banderas	NEW, READ, CHANGED[a)]	O	Sí	Ninguno
ID (Identificador)	cadena	Cualquier cadena válida	M	Sí	Ninguno
En respuesta a	cadena	Cualquier cadena válida	O	No	Ninguno
Cuenta de ítems	uint32	Hasta un máximo definido por la implementación	M	Sí	Ninguno
Diagnóstico NRN	cadena	Cualquier cadena válida	O	No	Ninguno

Cuadro 7/X.446 – Sumario de las propiedades del mensaje CMC *(fin)*

Mensaje					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Motivo NRN	cadena	Cualquier cadena válida	M, si el tipo de mensaje es recibido	No	Ninguno
Clase de objeto	object_class	MESSAGE[b)]	M	Sí	No aplicable
Estado fuera de mensaje	banderas	DELETED, SUBMITTED, SENT[a)]	O	Sí	Ninguno
Prioridad	enum	URGENT NORMAL LOW[c)]	O	No	Normal
Recibo solicitado	booleano	CMC_TRUE CMC_FALSE	O	No	CMC_FALSE
Tipo de recibo	enum	RN, NRN[d)]	O	No	Ninguno
Informe solicitado	enum	DR, NDR, BOTH, NONE[e)]	O	No	Ninguno
Rol (o papel)	enum	ORIGINAL RETURNED FORWARDED REPLIED OBSOLETED RESENT[f)]	O	No	Ninguno
Sensibilidad	enum	PERSONAL PRIVATE CONFIDENTIAL NONE[g)]	O	No	Ninguno
Tamaño	uint32	Cualquier valor válido, en octetos	O	No	Ninguno
Asunto	cadena	Cualquier cadena válida	O	No	Ninguno
Hora de recepción	iso_date_time	Cualquier fecha y hora ISO 8601	M	Sí	Ninguno
Hora de envío	iso_date_time	Cualquier fecha y hora ISO 8601	M	Sí	Ninguno
Tipo	enum	IPM, REPORT[h)]	M	No	IPM

a) Prefijo de valor "CMC_MESSAGE_STATUS_".

b) Prefijo de valor "CMC_OBJECT_TYPE_".

c) Prefijo de valor "CMC_PRIORITY_".

d) Prefijo de valor "CMC_RECEIPT_".

e) Prefijo de valor "CMC_REPORT_".

f) Prefijo de valor "CMC_MESSAGE_ROLE_".

g) Prefijo de valor "CMC_MESSAGE_SENSITIVITY_".

h) Prefijo de valor "CMC_MT_".

Cuadro 8/X.446 – Sumario de las propiedades del contenedor de mensajes CMC

Contenedor de mensajes					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Acción automática	banderas	CMC-AA_Delete	O	Sí	Despejado (*clear*)
Vástago permitido	booleano	CMC_TRUE CMC_FALSE	O	No	CMC_FALSE
Comentario	cadena	Cualquier cadena válida	O	No	Ninguno
Ubicación	enum	LOCAL, SERVER, UNKOWN[a)]	O	No	Ninguno
Nombre	cadena	Cualquier cadena válida	O	No	Ninguno
Clase de objeto	enum	MESSAGE CONTAINER[b)]	M	Sí	No aplicable
Progenitor	asa de objeto	Cualquier asa de objeto válida	M, si está anidada	No	Ninguno
Nombre del servidor	cadena	Cualquier cadena válida	O	No	Ninguno
Compartido	booleano	CMC_TRUE CMC_FALSE	O	No	CMC_FALSE
Tipo	enum	DELETED DRAFTS INBOX OUTBOX SENT[c)]	O	Sí	Ninguno

a) Prefijo de valor "CMC_MESSAGE_CONTAINER_".

b) Prefijo de valor "CMC_OBJECT_TYPE_".

c) Prefijo de valor "CMC_MCT_".

Cuadro 9/X.446 – Sumario de las propiedades de la información por cada recibiente CMC

Información por cada recibiente					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Comentario	cadena	Cualquier cadena válida	O	No	Ninguno
Hora de entrega	iso_date_time	Cualquier fecha y hora ISO 8601	O	No	Ninguno
Diagnóstico	cadena	Cualquier cadena válida	O	No	Ninguno
Clase de objeto	enum	PER RECIPIENT INFORMATION[a)]	M	Sí	No aplicable
Motivo (o razón)	cadena	Cualquier cadena válida	O	No	Ninguno
Dirección de recibiente	cadena	Cualquier cadena válida	M	Sí	No aplicable
Nombre de recibiente	cadena	Cualquier cadena válida	M	No	No aplicable
Tipo	enum	DR, NDR, UNKNOWN[b)]	M	Sí	No aplicable

a) Prefijo de valor "CMC_OBJECT_TYPE_".

b) Prefijo de valor "CMC_PRI_".

Cuadro 10/X.446 – Sumario de las propiedades del contenedor de perfiles CMC

Contenedor de perfiles					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Acción automática	banderas	CMC-AA_Delete	O	Sí	Despejado (*clear*)
Juego de caracteres	array_of_guid	Uno o más GUID para cualquier juego de caracteres	M	Sí	No aplicable
Comentario	cadena	Cualquier cadena válida	O	No	Ninguno
Conformidad	enum	SIMPLE_CMC, FULL_CMC[a)]	M	Sí	No aplicable
Servicio por defecto	cadena	Cualquier cadena válida	M	Sí	No aplicable
Usuario por defecto	cadena	Cualquier cadena válida	M	Sí	No aplicable
Terminador de línea	enum	CRLF, LF, CR[b)]	M	Sí	No aplicable
Clase de objeto	enum	Contenedor de perfiles[c)]	M	Sí	No aplicable

Cuadro 10/X.446 – Sumario de las propiedades del contenedor de perfiles CMC *(fin)*

Contenedor de perfiles					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Extensiones de objetos soportadas	array_of_guid	Uno o más GUID para objetos CMC	M	Sí	No aplicable
Objetos soportados	array_of_guid	Uno o más GUID para objetos CMC	M	Sí	No aplicable
Propiedades soportadas	array_of_guid	Uno o más GUID para propiedades CMC	M	Sí	No aplicable
Extensiones de propiedades soportadas	array_of_guid	Uno o más GUID para propiedades CMC	M	Sí	No aplicable
Contraseña requerida	enum	NO, OPT, YES[d)]	M	Sí	No aplicable
Servicio requerido	enum	NO, OPT, YES [d)]	M	Sí	No aplicable
Usuario requerido	enum	NO, OPT, YES [d)]	M	Sí	No aplicable
Cadenas contadas soportadas	boolean	CMC_TRUE CMC_FALSE	M	Sí	No aplicable
Soporte de ausencia de marca como leer	booleano	CMC_TRUE CMC_FALSE	M	Sí	No aplicable
Interfaz de usuario disponible	booleano	CMC_TRUE CMC_FALSE	M	Sí	No aplicable
Usuarios	array_string	Nombres de recibientes	O	Sí	No aplicable
Versión de la implementación	uint16	100 ó 200	M	Sí	No aplicable
Versión de la especificación	uint16	100 ó 200	M	Sí	No aplicable

a) Prefijo de valor "CMC_CONF_".

b) Prefijo de valor "CMC_LINE_TERM_".

c) Prefijo de valor "CMC_OBJECT_TYPE_".

d) Prefijo de valor "CMC_REQUIRED_".

Cuadro 11/X.446 – Sumario de las propiedades del recibiente CMC

Recibiente					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Dirección	cadena	Cualquier cadena válida	M	No	Ninguno
Retorno de contenido solicitado	booleano	CMC_TRUE CMC_FALSE	O	No	Ninguno
Nombre	cadena	Cualquier cadena válida	O	No	Ninguno
Clase de objeto	enum	RECIPIENT[a)]	M	Sí	No aplicable
Recibo solicitado	enum	RN, NRN, BOTH, NONE[b)]	O	No	Ninguno
Informe solicitado	enum	DR, NDR, BOTH, NONE[c)]	O	No	Ninguno
Bandera de responsabilidad	booleano	CMC_TRUE CMC_FALSE	M	No	CMC_TRUE
Rol (o papel)	enum	TO, CC, BCC, ORIGINATOR, AUTHORIZING_USER, REPLY_TO, FORWARDED, ACTUAL, INTENDED[d)]	O	No	Ninguno
Tipo	enum	UNKNOW, INDIVIDUAL, GROUP[e)]	M	No	INDIVIDUAL

a) Prefijo de valor "CMC_OBJECT_TYPE_".

b) Prefijo de valor "CMC_RECEIPT_".

c) Prefijo de valor "CMC_REPORT_".

d) Prefijo de valor "CMC_RECIPIENT_ROLE_".

e) Prefijo de valor "CMC_RCT_".

Cuadro 12/X.446 – Sumario de las propiedades del informe CMC

Informe					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
ID de aplicación	cadena	Cualquier cadena válida	O	No	Ninguno
ID	guid	Cualquier cadena ISO 9070 válida	M	Sí	No aplicable
Cuenta de ítems	uint32	Cualquier número entero válido	M	Sí	No aplicable
ID de sistema de mensajería	cadena	Cualquier cadena válida	O	No	Ninguno
Clase de objeto	enum	REPORT[a)]	M	Sí	No aplicable
Leído	booleano	CMC_TRUE CMC_FALSE	O	No	Ninguno
Tamaño	uint32	Tamaño del informe en octetos	O	No	Ninguno
Asunto	cadena	Cualquier cadena válida	M	No	Ninguno
ID de mensaje de asunto	cadena	Cualquier cadena válida	O	No	Ninguno
Hora de recepción	iso_date_time	Cualquier fecha y hora ISO 8601	M	Sí	Ninguno
Hora de envío	iso_date_time	Cualquier fecha y hora ISO 8601	M	Sí	Ninguno
No enviado	booleano	CMC_TRUE CMC_FALSE	O	No	Ninguno

a) Prefijo de valor "CMC_OBJECT_TYPE_".

Cuadro 13/X.446 – Sumario de las propiedades del contenedor raíz CMC

Contenedor raíz					
Nombre de propiedad	**Tipo (CMC_pv_)**	**Valores posibles**	**Clasificación**	**Lectura solamente**	**Valor por defecto**
Vástago permitido	booleano	CMC_TRUE CMC_FALSE	O	No	CMC_FALSE
Comentario	cadena	Cualquier cadena válida	O	No	Ninguno
Ubicación	enum	LOCAL SERVER UNKNOWN[a)]	O	No	Ninguno
Nombre	cadena	Cualquier cadena válida	O	No	Ninguno
Clase de objeto	enum	ROOT CONTAINER[b)]	M	Sí	No aplicable
Compartido	booleano	CMC_TRUE CMC_FALSE	O	No	CMC_FALSE

a) Prefijo de valor "CMC_ROOT_CONTAINER_LOCATION_".

b) Prefijo de valor "CMC_OBJECT_TYPE_".

Seguidamente se presenta la información del manual sobre estas propiedades.

5.1 Propiedades del objeto de libro de direcciones

Un libro de direcciones es un objeto de contenedor que se compone de direcciones de entidades y puede contener otros libros de direcciones. El soporte de libros de direcciones no es obligatorio. Las subcláusulas siguientes definen, declaran y describen propiedades de libros de direcciones.

5.1.1 Vástago permitido

NOMBRE

Vástago de libro de direcciones permitido (*address book child allowed*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ADDRESS_BOOK_CHILD_ALLOWED      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Child Allowed//EN"
```

DESCRIPCIÓN

La propiedad que permite o prohíbe la existencia de un vástago del libro de direcciones.

El valor por defecto de esta propiedad es CMC_FALSE.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.1.2 Comentario

NOMBRE

Comentario de libro de direcciones (*address book comment*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ADDRESS_BOOK_COMMENT          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Comment//EN"
```

DESCRIPCIÓN

Un comentario descriptivo sobre el libro de direcciones.

Esta es una propiedad de tipo **CMC_pv_string**.

5.1.3 Ubicación

NOMBRE

Ubicación de libro de direcciones (*address book location*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ADDRESS_BOOK_LOCATION        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Location//EN"
```

DESCRIPCIÓN

Esta propiedad específica la ubicación del libro de direcciones.

Son valores válidos para esta propiedad:

CMC_ADDRESS_BOOK_LOCATION_LOCAL
CMC_ADDRESS_BOOK_LOCATION_SERVER
CMC_ADDRESS_BOOK_LOCATION_UNKNOWN

CMC_ADDRESS_BOOK_LOCATION_LOCAL – Específica que el libro de direcciones es local y no está en el servidor de mensajería.

CMC_ADDRESS_BOOK_LOCATION_SERVER – Específica que el libro de direcciones está en el servidor de mensajería.

CMC_ADDRESS_BOOK_LOCATION_UNKNOWN – Específica que la ubicación del libro de direcciones es desconocida. Este es el valor por defecto.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.1.4 Nombre

NOMBRE

Nombre de libro de direcciones (*address book name*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ADDRESS_BOOK_NAME        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Name//EN"
```

DESCRIPCIÓN

Esta propiedad específica el nombre del libro de direcciones.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.1.5 Clase de objeto

NOMBRE

Clase de objeto de libro de direcciones (*address book object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"
```

DESCRIPCIÓN

Esta propiedad define la clase del objeto como libro de direcciones.

Esta propiedad la crea la función **cmc_open_object_handle()**.

El único valor válido para esta propiedad es CMC_PT_OBJECT_CLASS_ADDRESS_BOOK, que específica que la clase del objeto es un libro de direcciones.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.1.6 Progenitor

NOMBRE

Progenitor del libro de direcciones (*address book parent*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ADDRESS_BOOK_PARENT          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Parent//EN"
```

DESCRIPCIÓN

Esta propiedad específica el progenitor del libro de direcciones. Si la implementación soporta el anidamiento de libros de direcciones, esta propiedad específica el libro de direcciones progenitor. Si el libro de direcciones es el nivel más alto, esta propiedad no está presente. En otro caso, es obligatoria.

Ésta es una propiedad de tipo **CMC_pv_object_handle**.

5.1.7 Nombre del servidor

NOMBRE

Nombre del servidor de libro de direcciones (*address book server name*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ADDRESS_BOOK_SERVER_NAME        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Server Name//EN"
```

DESCRIPCIÓN

Esta propiedad específica el nombre del servidor en el que está ubicado el libro de direcciones.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.1.8 Compartido

NOMBRE

Libro de direcciones compartido (*address book shared*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ADDRESS_BOOK_SHARED              \
    "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Shared//EN"
```

DESCRIPCIÓN

Esta propiedad indica si más de un usuario han ganado acceso a este libro de direcciones.

Si esta propiedad está soportada, su valor por defecto para es CMC_FALSE.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.1.9 Tipo

NOMBRE

Tipo de libro de direcciones (*address book type*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ADDRESS_BOOK_TYPE            \
    "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Type//EN"
```

DESCRIPCIÓN

Esta propiedad específica el tipo del libro de direcciones.

Son valores válidos para esta propiedad:

CMC_ADDRESS_BOOK_TYPE_GLOBAL
CMC_ADDRESS_BOOK_TYPE_PERSONAL

CMC_ADDRESS_BOOK_TYPE_GLOBAL – Específica que el libro de direcciones es de un tipo global, o que abarca la totalidad de la empresa. Un libro de direcciones global no es necesariamente un libro de direcciones compartido.

CMC_ADDRESS_BOOK_TYPE_PERSONAL – Específica que el libro de direcciones es de un tipo personal, que es originado y mantenido localmente.

El valor por defecto para esta propiedad es CMC_ADDRESS_BOOK_TYPE_PERSONAL.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.2 Propiedades del objeto de ítem de contenido

En esta Recomendación, un ítem de contenido es un objeto asociado con el contenido de un mensaje. Se utiliza para representar añadiduras y notas, aunque no se distingue entre ellas en la interfaz de programación. Las subcláusulas siguientes definen, declaran y describen propiedades de objeto de añadidura.

5.2.1 Juego de caracteres

NOMBRE

Juego de caracteres de ítem de contenido (*content item character set*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_CHARACTER_SET      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Character Set//EN"
```

DESCRIPCIÓN

Esta propiedad específica el juego de caracteres de la información de contenido incorporada en el ítem de contenido. En ausencia de esta propiedad, el juego de caracteres por defecto para información de contenido incorporada en el ítem de información es el mismo que el del contexto de la sesión.

El valor de esta propiedad es una cadena que representa el identificador público formal del juego de caracteres. El identificador público formal puede ser uno de los siguientes:

```
#define   CMC_CHARSET_437          "–//XAPIA/CHARSET//NONSGML IBM 437//EN"
#define   CMC_CHARSET_850          "–//XAPIA/CHARSET//NONSGML IBM 850//EN"
#define   CMC_CHARSET_1252         "–//XAPIA/CHARSET//NONSGML Microsoft 1252//EN"
#define   CMC_CHARSET_ISTRING      "–//XAPIA/CHARSET//NONSGML Apple ISTRING//EN"
#define   CMC_CHARSET_UNICODE      "–//XAPIA/CHARSET//NONSGML UNICODE//EN"
#define   CMC_CHARSET_T61          "–//XAPIA/CHARSET//NONSGML TSS T61//EN"
#define   CMC_CHARSET_IA5          "–//XAPIA/CHARSET//NONSGML TSS IA5//EN"
#define   CMC_CHARSET_ISO_10646 "–//XAPIA/CHARSET//NONSGML ISO 10646//EN"
#define   CMC_CHARSET_ISO_646      "–//XAPIA/CHARSET//NONSGML ISO 646//EN"
#define   CMC_CHARSET_ISO_8859_1   "–//XAPIA/CHARSET//NONSGML ISO 8859-1//EN"
```

Las implementaciones pueden prever otros juegos de caracteres.

Ésta es una propiedad de tipo **CMC_pv_guid**.

5.2.2 Información de contenido

NOMBRE

Información de contenido de ítem de contenido (*content item content information*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_CONTENT_INFORMATION   \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Content Information//EN"
```

DESCRIPCIÓN

El valor de esta propiedad es el contenido de un ítem de contenido.

Ésta es una propiedad de tipo **CMC_pv_opaque_data**.

5.2.3 Tipo de contenido

NOMBRE

Tipo de contenido de ítem de contenido (*content item content type*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_CONTENT_TYPE        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Content Type//EN"
```

DESCRIPCIÓN

Esta propiedad específica el tipo de contenido del ítem de contenido. Un valor NULL designa un tipo no definido de ítem de contenido.

Los siguientes valores de GUID son válidos para la propiedad Tipo de los objetos de ítem de contenido.

```
#define CMC_CT_PLAIN_TEXT                                   \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Plain Text//EN"
#define CMC_CT_GIF_IMAGE                                    \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML GIF Image//EN"
#define CMC_CT_JPEG_IMAGE                                   \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML JPEG Image//EN"
#define CMC_CT_BASIC_AUDIO                                  \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Basic Audio//EN"
#define CMC_CT_MPEG_VIDEO                                   \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML MPEG Video//EN"
#define CMC_CT_MESSAGE                                      \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Message//EN"
#define CMC_CT_PARTIAL_MESSAGE                              \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Partial Message//EN"
#define CMC_CT_EXTERNAL_MESSAGE                             \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML External Message//EN"
#define CMC_CT_APPLICATION_OCTET_STREAM                     \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Application Octet Stream//EN"
#define CMC_CT_APPLICATION_POSTSCRIPT                       \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Application PostScript//EN"
#define CMC_CT_ALTERNATIVE_MULTIPART                        \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Alternative Multipart//EN"
#define CMC_CT_DIGEST_MULTIPART                             \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Digest Multipart//EN"
#define CMC_CT_MIXED_MULTIPART                              \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Mixed Multipart//EN"
#define CMC_CT_OLE                                          \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML OLE//EN"
#define CMC_CT_MIXED_MULTIPART                              \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML Mixed Multipart//EN"
#define CMC_CT_X400_G3_FAX                                  \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 G3 Fax//EN"
#define CMC_CT_X400_G4_FAX                                  \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 G4 Fax//EN"
#define CMC_CT_X400_ENCRYPTED                               \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Encrypted//EN"
#define CMC_CT_X400_NATIONALLY_DEFINED                      \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Nationally Defined//EN"
#define CMC_CT_X400_FILE_TRANSFER                           \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 File Transfer//EN"
#define CMC_CT_X400_VOICE                                   \
    "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Voice//EN"
```

```
#define CMC_CT_X400_VIDEOTEX                                    \
        "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Videotex//EN"
#define CMC_CT_X400_MIXED_MODE                                  \
        "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Mixed Mode//EN"
#define CMC_CT_X400_PRIVATELY_DEFINED_6937                      \
        "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Privately Defined 6937//EN"
#define CMC_CT_X400_EXTERNAL_TRACE                              \
        "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 External Trace//EN"
#define CMC_CT_X400_INTERNAL_TRACE                              \
        "–//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Internal Trace//EN"
#define CMC_CT_SMTP_SESSION_TRANSCRIPT                          \
        "–//XAPIA/CMC/CONTENT TYPE//NONSGML SMTP Session Transcript//EN"
```

CMC_CT_PLAIN_TEXT – Específica contenido de texto ordinario o no formateado.

CMC_CT_GIF_IMAGE – Específica contenido de datos de imagen en el formato de imagen para gráficos Graphics Image Format utilizado por MIME y la World Wide Web.

CMC_CT_JPEG-IMAGE – Específica contenido de datos de imagen en la forma de la norma del grupo Joint Picture Encoding Group de la ISO, utilizada por MIME y la World Wide Web.

CMC_CT_BASIC_AUDIO – Específica contenido de datos de audio en la forma de audio codificado con la ley mu, 8 bits, para la RDSI, o con la MIC definida en la Recomendación G.711, con una velocidad de muestreo de 8000 Hz y un solo canal.

CMC_CT_MPEG_VIDEO – Específica contenido de vídeo en la forma del grupo Motion Picture Encoding Group de la ISO, ISO 11172 utilizada por MIME y la World Wide Web.

CMC_CT_MESSAGE – Específica que el contenido es un mensaje encapsulado.

CMC_CT_PARTIAL_MESSAGE – Específica que el contenido es una parte de otro mensaje. Este tipo de contenido permite entregar un mensaje grande como varias piezas distintas, para facilitar la recepción.

CMC_CT_EXTERNAL_MESSAGE – Específica que el contenido es externo al mensaje. La propiedad de información de contenido contiene una referencia textual a la información de contenido externa.

CMC_CT_APPLICATION_OCTET_STREAM – Específica que el contenido es un tren de octetos dependientes de la aplicación.

CMC_CT_APPLICATION_POSTSCRIPT – Específica que el contenido es un programa escrito en el lenguaje PostScript de la compañía Adobe Systems, Inc.

CMC_CT_ALTERNATIVE_MULTIPART – Específica que el contenido es, o bien una forma alternativa de contenido de otra nota, o de otro ítem de contenido, dentro del objeto de mensaje.

CMC_CT_DIGEST_MULTIPART – Específica que el contenido es un mensaje de un grupo de mensajes conexos dentro del objeto de mensaje. Los mensajes pueden utilizarse como una secuencia de intervenciones en la discusión de un tema, captadas en un hilo de mensajes, como suelen presentarse en los sistemas de tableros de boletines (*bulletin board systems*).

CMC_CT_MIXED_MULTIPART – Específica que el contenido es un mensaje de una secuencia ordenada de mensajes dentro del objeto de mensajes.

CMC_CT_PARALLEL_MULTIPART – Específica que el contenido es un mensaje de los que forman un grupo de mensajes en una secuencia arbitrariamente ordenada, dentro del objeto de mensaje.

CMC_CT_OLE – Específica que el tipo del ítem de contenido es ítem de contenido de objeto OLE (*object linking and embedding*, vinculación e incrustación de objetos).

CMC_CT_X400_G3_FAX – Específica que el contenido representa imágenes de facsímil de grupo 3, una secuencia de cadenas de bits. Cada componente de datos G3 codifica una página individual de datos, como se específica en las Recomendaciones T.4 y T.30.

CMC_CT_X400_G4_FAX – Específica que el contenido representa un documento en forma final del tipo que puede ser procesado por terminales facsímil del grupo 4 clase 1.

CMC_CT_X400_ENCRYPTED – Específica que el contenido está formado por cadenas de bits y está codificado de acuerdo con las reglas de la codificación básica de la Recomendación X.209.

CMC_CT_X400_NATIONALLY_DEFINED – Específica que el contenido es un objeto de información cuya semántica y sintaxis abstracta están definidas en el plano nacional por un país cuya identidad está convenida bilateralmente por el originador del mensaje y todos los recibientes potenciales del mensaje.

CMC_CT_X400_FILE_TRANSFER – Específica que la información de contenido consiste en cantidades relativamente grandes de datos. La propiedad información de contenido contiene la referencia textual a su semántica y sintaxis abstracta, las que se designan por un identificador de objeto.

CMC_CT_X400_VOICE – Específica que el contenido es la palabra digitalizada, una cadena de bits. Su codificación no está actualmente definida en la Recomendación X.420 de la versión 1988.

CMC_CT_X400_VIDEOTEX – Específica que el contenido representa datos videotex. Su sintaxis está definida en las Recomendaciones T.100 y T.101.

CMC_CT_X400_MIXED_MODE – Específica que el contenido representa un documento en forma final del tipo que puede ser procesado por terminales teletex en modo mixto y terminales facsímil del grupo 4 clases 2 y 3.

CMC_CT_X400_PRIVATELY_DEFINED_6937 – Específica que el contenido está definido privadamente. El contenido se codifica de acuerdo con los juegos de caracteres y las reglas de codificación especificadas en ISO 6937.

CMT_CT_X400_EXTERNAL_TRACE – Específica que el contenido es información de rastreo externo X.400 para fines de diagnóstico.

CMC_CT_X400_INTERNAL_TRACE – Específica que el contenido es información de rastreo interno para fines de diagnóstico.

CMC_CT_SMTP_SESSION_TRANSCRIPT – Específica que el contenido es información de transcripción de sesión SMTP para fines de diagnóstico.

Ésta es una propiedad de tipo **CMC_pv_guid**.

5.2.4 Hora de creación

NOMBRE

Hora (o tiempo) de creación de ítem de contenido (*content item create time*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_CREATE_TIME        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Create Time//EN"
```

DESCRIPCIÓN

Esta propiedad específica la fecha y hora en que se creó el ítem de contenido.

Ésta es una propiedad de tipo **CMC_pv_iso_date_time**.

5.2.5 Tipo de codificación

NOMBRE

Tipo de codificación de ítem de contenido (*content item encoding type*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT ITEM_ENCODING_TYPE      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Encoding Type//EN"
```

DESCRIPCIÓN

Esta propiedad específica el tipo de codificación del contenido del ítem de contenido.

El valor por defecto de esta propiedad es CMC_ET_7_BIT.

Los siguientes valores son válidos para la propiedad Tipo de codificación del objeto de ítem de contenido:

```
#define CMC_ET_7_BIT                                \
    "–//XAPIA/CMC/ENCODING TYPE//NONSGML 7 Bit//EN"
#define CMC_ET_BASE64                               \
    "–//XAPIA/CMC/ENCODING TYPE//NONSGML Base64//EN"
```

```
#define CMC_ET_BINARY                                   \
    "-//XAPIA/CMC/ENCODING TYPE//NONSGML Binary//EN"
#define CMC_ET_8_BIT                                    \
    "-//XAPIA/CMC/ENCODING TYPE//NONSGML 8 Bit//EN"
#define CMC_ET_QUOTED_PRINTABLE                         \
    "-//XAPIA/CMC/ENCODING TYPE//NONSGML Quoted Printable//EN"
```

CMC_ET_7_BIT – Específica que la información de contenido no ha sido codificada. Significa también que los octetos de la información de contenido tienen 7 bits de datos.

CMC_ET_BASE64 – Específica que la información de contenido se ha codificado en la forma de Base 64 de RFC 1521/MIME para una secuencia arbitraria de octetos.

CMC_ET_BINARY – Específica que la información de contenido no se ha codificado. Significa también que la información de contenido consta de cantidades relativamente grandes de datos y que los octetos pueden tener fijado el bit de orden superior.

CMC_ET_8_BIT – Específica que la información de contenido no se ha codificado. Significa también que la información de contenido consta de líneas relativamente cortas de octetos con el bit de orden superior fijado.

CMC_ET_QUOTED_PRINTABLE – Específica que la información de contenido se ha codificado en la forma de RFC 1521/MIME para caracteres imprimibles grandes del juego de caracteres ASCII, por lo que es poco probable que los octetos resultantes sean modificados por transportes de correos.

Ésta es una propiedad de tipo **CMC_pv_guid**.

5.2.6 Directorio de ficheros

NOMBRE

Directorio de ficheros de ítem de contenido (*content item file directory*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_FILE_DIRECTORY     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item File Directory//EN"
```

DESCRIPCIÓN

Esta propiedad específica el directorio de ficheros del ítem de contenido cuando se añadió al mensaje. En aquellos casos en que la propiedad Tipo de contenido es CMC_CT_EXTERNAL_MESSAGE, esta propiedad indica el nombre del servidor así como el nombre del directorio en un servidor distante. El formato de esta cadena está definido por la implementación.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.2.7 Nombre de fichero

NOMBRE

Nombre de fichero de ítem de contenido (*content item file name*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_FILE_NAME          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item File Name//EN"
```

DESCRIPCIÓN

Esta propiedad específica el nombre de fichero de ítem de contenido cuando fue añadido el mensaje. Cuando la propiedad Tipo de contenido es CMC_CT_EXTERNAL_MESSAGE, esta propiedad indica el nombre del fichero en un servidor distante. El formato de esta cadena está definido por la implementación.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.2.8 Número de ítem

NOMBRE

Número (de ítem) de ítem de contenido (*content item item number*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_ITEM_NUMBER        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Item Number/EN"
```

DESCRIPCIÓN

Esta propiedad específica el número secuencial del ítem dentro de su contenedor progenitor, un objeto de mensaje u otro ítem de contenido. Ésta es una propiedad obligatoria.

En un contenedor progenitor determinado no puede haber dos ítems cuyo número de ítem tengan el mismo valor.

Ésta es una propiedad de tipo **CMC_pv_uint32**.

5.2.9 Tipo de ítem

NOMBRE

Tipo (de ítem) de ítem de contenido (*content item item type*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_ITEM_TYPE          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Item Type//EN"
```

DESCRIPCIÓN

Esta propiedad específica el tipo del ítem de contenido. Son valores válidos para esta propiedad:

CMC_IT_NOTE
CMC_IT_ATTACHMENT
CMC_IT_ANNOTATION

CMC_IT_NOTE – Específica que es una nota.

CMC_IT_ATTACHMENT – Específica que es una añadidura.

CMC_IT_ANNOTATION – Específica una anotación en otro objeto de ítem de contenido.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.2.10 Último modificado

NOMBRE

Último ítem de contenido modificado (*content item last modified*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_LAST_MODIFIED      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Last Modified//EN"
```

DESCRIPCIÓN

Esta propiedad específica la fecha y hora en que fue modificado por última vez el fichero del que se obtuvo el ítem de contenido.

Ésta es una propiedad de tipo **CMC_pv_iso_date_time**.

5.2.11 Clase de objeto

NOMBRE

Clase de objeto de ítem de contenido (*content item object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class //EN"
```

DESCRIPCIÓN

Esta propiedad define la clase del objeto como un ítem de contenido. Esta propiedad se crea por la función **cmc_open_object_handle()**.

El único valor válido para esta propiedad es CMC_PT_OBJECT_CLASS_CONTENT_ITEM, que específica que la clase del objeto es un ítem de contenido.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.2.12 Posición de reproducción

NOMBRE

Posición de reproducción de ítem de contenido (*content item render position*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_RENDER_POSITION    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Render Position//EN"
```

DESCRIPCIÓN

Esta propiedad específica la posición del ítem de contenido dentro de su contenedor.

Ésta es una propiedad de tipo **CMC-pv_uint32**.

5.1.13 Tamaño

NOMBRE

Tamaño de ítem de contenido (*content item size*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_SIZE            \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Size//EN"
```

DESCRIPCIÓN

Esta propiedad específica el tamaño del ítem de contenido.

Ésta es una propiedad de tipo **CMC_pv_uint32**.

5.2.14 Título

NOMBRE

Título de ítem de contenido (*content item title*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_CONTENT_ITEM_TITLE           \
    "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Title//EN"
```

DESCRIPCIÓN

Esta propiedad específica la descripción íntegra del ítem de contenido. Por ejemplo, "Informe financiero trimestral" podría ser un título de ítem de contenido.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.3 Propiedades del objeto de lista de distribución

Las listas de distribución identifican grupos de usuarios. Una lista de distribución contiene objetos de recibientes. Las siguientes subcláusulas definen, declaran y describen propiedades de lista de distribución.

5.3.1 Dirección

NOMBRE

Dirección de lista de distribución (*distribution list address*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_DISTRIBUTION_LIST_ADDRESS              \
    "–//XAPIA/CMC/PROPERTY//NONSGML Distribution List Address//EN"
```

DESCRIPCIÓN

Esta propiedad proporciona la dirección que se va a utilizar para enviar correo a la lista de distribución. Esta dirección suele ser el mismo valor que el nombre de la lista de distribución. Es una propiedad facultativa de una lista de distribución.

Esta propiedad la genera el sistema de mensajería.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.3.2 Comentario

NOMBRE

Comentario de lista de distribución (*distribution list comment*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_DISTRIBUTION_LIST_COMMENT              \
    "–//XAPIA/CMC/PROPERTY//NONSGML Distribution List Comment//EN"
```

DESCRIPCIÓN

Un comentario descriptivo sobre la lista de distribución.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.3.3 Hora de la última modificación

NOMBRE

Hora (o tiempo) de la última modificación de la lista de distribución (*distribution list last modification time*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_DISTRIBUTION_LIST_LAST_MODIFICATION_TIME              \
    "–//XAPIA/CMC/PROPERTY//NONSGML Distribution List Last Modification Time//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la fecha y hora de la última modificación de la lista de distribución.

Ésta es una propiedad de tipo **CMC_pv_iso_date_time**.

5.3.4 Nombre

NOMBRE

Nombre de lista de distribución (*distribution list name*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_DISTRIBUTION_LIST_NAME       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Distribution List Name//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el nombre de la lista de distribución. Es una propiedad obligatoria para los objetos de lista de distribución.

La cadena puede ser generada por el sistema de mensajería a partir de un directorio, o por el usuario.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.3.5 Clase de objeto

NOMBRE

Clase de objeto de lista de distribución (*distribution list object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"
```

DESCRIPCIÓN

Esta propiedad define la clase del objeto como una lista de distribución. Esta propiedad se crea por la función **cmc_open_object_handle()**.

El único valor válido para esta propiedad es CMC_PT_OBJECT_CLASS_DISTRIBUTION_LIST, que especifica que la clase del objeto es una lista de distribución.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.3.6 Progenitor

NOMBRE

Progenitor de lista de distribución (*distribution list parent*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_DISTRIBUTION_LIST_PARENT        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Distribution List Parent//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el progenitor de la lista de distribución. Si la implementación soporta el anidamiento de listas de distribución, esta propiedad especifica la lista de distribución progenitora. Si la lista de distribución es el nivel superior, esta propiedad no está presente. De lo contrario, es obligatoria. Esta propiedad es nula en el caso del progenitor.

Ésta es una propiedad de tipo **cmc_pv_object_handle**.

5.3.7 Compartida

NOMBRE

Lista de distribución compartida (*distribution list shared*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_DISTRIBUTION_LIST_SHARED        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Distribution List Shared//EN"
```

DESCRIPCIÓN

Esta propiedad indica si más de un usuario tiene acceso a esta lista de distribución.

Si esta propiedad soportada, el valor por defecto es CMC_FALSE.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.4 Propiedades del objeto de mensaje

El objeto de mensaje es una colección de propiedades de objeto específicas de los mensajes. Las siguientes subcláusulas definen, declaran y describen propiedades del objeto de mensaje.

5.4.1 Identificador de aplicación

NOMBRE

Identificador de aplicación de mensaje (*message application id*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_APPLICATION_ID      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Application Id//EN"
```

DESCRIPCIÓN

Esta propiedad especifica un identificador globalmente único para el mensaje. Esta propiedad la determina la aplicación.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.4.2 Estado de mensaje de aplicación

NOMBRE

Estado de mensaje de aplicación (de mensajes) (*message application message status*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_APPLICATION_MSG_STATUS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Application Msg Status//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el estado determinado por el llamante para el mensaje. Esta propiedad puede utilizarla el llamante para determinar si un mensaje es un proyecto (o borrador) de mensaje, o si es un mensaje finalizado, y marcarlo como tal. No hay semánticas implicadas para el servicio de mensajería; sin embargo, el valor de la propiedad subsiste de una sesión a otra.

Un valor válido para esta propiedad es:

CMC_MESSAGE_STATUS_DRAFT

CMC_MESSAGE_STATUS_DRAFT – Especifica que el mensaje está en modo proyecto (o borrador).

Ésta es una propiedad de tipo **CMC_pv_flags**.

5.4.3 Acción automática

NOMBRE

Acción automática de mensaje (*message auto-action*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_AUTO_ACTION                \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Auto Action//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la acción automática o la disposición que habrá de tomarse sobre el mensaje después de enviado. El soporte de esta propiedad es facultativo para las implementaciones conformes con esta Recomendación. En el caso de mensajes de nueva creación, la propiedad se crea por una llamada a **cmc_add_properties()**. En el caso de mensajes de nueva creación, la propiedad puede también modificarse por una llamada a **cmc_add_properties()**, o suprimirse por una llamada a **cmc_delete_properties()**. Esta propiedad en el objeto de mensaje prevalecerá sobre la preferencia fijada en el objeto contenedor de perfiles.

El valor válido para esta propiedad es:

CMC_AA_DELETE

Fijado: El mensaje especificado habrá de ser suprimido por el sistema de mensajería subyacente después que lo haya depositado con éxito para transferencia.

Despejado: El mensaje especificado se coloca en la carpeta de mensajes enviados, si ésta existe. Si no existe, se suprime el mensaje.

Ésta es una propiedad de tipo **CMC_pv_flags**.

5.4.4 Hora de entrega diferida

NOMBRE

Hora de entrega diferida de mensaje (*message deferred delivery time*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_DEFERRED_DELIVERY_TIME                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Deferred Delivery Time//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la fecha y hora en UTC (tiempo universal coordinado, *Coordinate Universal Time*) antes de la cual el mensaje no debe entregarse a los recibientes.

Ésta es una propiedad de tipo **CMC_pv_iso_date_time**.

5.4.5 Identificador

NOMBRE

Identificador de mensaje (*message id*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_ID          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Id//EN"
```

DESCRIPCIÓN

Esta propiedad especifica un identificador globalmente único para el mensaje. Esta propiedad se fija por la función **cmc_send_message_object()**, se define por el servicio de mensajería (establecido en el momento del depósito), y es única dentro del dominio.

En aplicaciones de cabecera (*gateway applications*), el identificador de mensaje puede ser añadido o actualizado por el llamante.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.4.6 Estado de mensaje de entrada

NOMBRE

Estado de mensaje (de mensaje) de entrada (*message in message status*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_IN_MSG_STATUS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message In Msg Status//EN"
```

DESCRIPCIÓN

Esta propiedad indica el estado (especificado por el servicio de mensajería) de entrada o de recepción de un mensaje. Esta propiedad la utiliza el servicio de mensajería para registrar información relativa a las modalidades de la recepción y del procesamiento del mensaje. Por ejemplo, el hecho de que un mensaje ha sido simplemente recibido y colocado en un apartado de entrada, o ha sido leído, o ha sido modificado con respecto a su modo de recepción original, puede ser especificado por el servicio de mensajería.

El soporte de esta propiedad es facultativo para las implementaciones conformes con esta Recomendación. Esta propiedad es de lectura solamente; el servicio de mensajería subyacente la crea y puede modificarla. El usuario no puede suprimirla.

Son valores válidos para esta propiedad:

CMC_MESSAGE_STATUS_NEW
CMC_MESSAGE_STATUS_READ
CMC_MESSAGE_STATUS_CHANGED

CMC_MESSAGE_STATUS_NEW – Especifica que el mensaje acaba de ser recibido por el servicio de mensajería subyacente. Esta bandera se reinicializa cuando se cierra la sesión, se gana acceso al objeto de mensaje o se cierra el contenedor de mensajes.

CMC_MESSAGE_STATUS_READ – Especifica que el mensaje ha sido leído. Esta bandera de estado se fija cuando una propiedad de uno de los objetos de ítem de contenido subordinados al mensaje ha sido leído por una llamada a **cmc_read_properties()**.

CMC_MESSAGE_STATUS_CHANGED – Especifica si el contenido de un mensaje ha cambiado con respecto a la forma en que se recibió originalmente. Esta bandera de estado se fija cuando se añade o modifica una propiedad contenida en el objeto de mensaje por una llamada a la función **cmc_add_properties()**, o se suprime por una llamada a la función **cmc_delete_properties()**.

Ésta es una propiedad de tipo **CMC_pv_flags**.

5.4.7 En respuesta a

NOMBRE

Mensaje en respuesta a (*message in reply to*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_IN_REPLY_TO          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message In Reply To//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la anterior correspondencia a que se responde por medio de este mensaje.

El valor de la propiedad puede ser una referencia textual, o una aproximación textual al identificador de mensaje de la correspondencia anterior.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.4.8 Cuenta de ítems

NOMBRE

Cuenta de ítems de mensaje (*message item count*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_ITEM_COUNT    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Item Count//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el número de ítems de nivel superior contenidos en un mensaje. Esta cuenta no incluye los ítems de contenido anidados en otros ítems de contenido, mensajes o informes. Esta propiedad la fija la implementación.

Ésta es una propiedad de tipo **CMC_pv_uint32**.

5.4.9 Diagnóstico de la notificación de no recepción

NOMBRE

Diagnóstico NRN de mensaje (*message NRN diagnostic*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PROP_TYPE_MESSAGE_NRN_DIAGNOSTIC                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message NRN Diagnostic//EN"
```

DESCRIPCIÓN

Esta propiedad especifica los detalles de diagnóstico del motivo de la notificación de no recepción. Estos detalles son adicionales al motivo de no recepción. Esta propiedad sólo es pertinente en el caso de los mensajes de tipo CMC_MT_RECEIPT.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.4.10 Motivo de la notificación de no recepción

NOMBRE

Motivo de la notificación de no recepción del mensaje (*message NRN reason*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PROP_TYPE_MESSAGE_NRN_REASON                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message NRN Reason//EN"
```

DESCRIPCIÓN

Esta propiedad explica el motivo por el cual no se recibió un mensaje. Esta propiedad sólo es pertinente en mensajes de tipo CMC_MT_RECEIPT con la propiedad de tipo de recepción fijada al valor CMC_RECEIPT_NRN.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.4.11 Clase de objeto

NOMBRE

Clase de objeto de mensaje (*message object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"
```

DESCRIPCIÓN

Esta propiedad define la clase de objeto como un mensaje.

Esta propiedad se crea por la función **cmc_open_object_handle()**.

El único valor válido que puede tener esta propiedad es CMC_PT_OBJECT_CLASS_MESSAGE, que especifica que la clase del objeto es un mensaje.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.4.12 Estado de mensaje de salida

NOMBRE

Estado (de mensaje) de mensaje de salida (*message out message status*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_OUT_MSG_STATUS                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Out Msg Status//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el estado de salida del mensaje o las disposiciones que habrán de tomarse sobre el mismo, según lo determinado por el servicio de mensajería. Esta propiedad la utiliza el servicio de mensajería subyacente para registrar información en cuanto el estado del mensaje en lo que respecta a las disposiciones que se habrán de tomar sobre el mismo. Por ejemplo, el servicio de mensajería puede especificar el hecho de que un mensaje se ha marcado para supresión, o se ha depositado para transferencia, o se encuentra en curso de envío.

El soporte de esta propiedad es facultativo para las implementaciones conformes con esta Recomendación. La propiedad es de lectura solamente; el servicio de mensajería subyacente la crea y la modifica. El usuario no puede suprimirla.

Son valores válidos para esta propiedad:

CMC_MESSAGE_STATUS_DELETED
CMC_MESSAGE_STATUS_SUBMITTED
CMC_MESSAGE_STATUS_SENT

CMC_MESSAGE_STATUS_DELETED – Especifica que el mensaje se encuentra en curso de ser suprimido. En algunos entornos de implementación (por ejemplo, usuario desconectado), no es posible tratar inmediatamente una operación de supresión de un mensaje. Esta bandera indica que aunque el mensaje aparezca en el contenedor de mensajes, está marcado para supresión.

CMC_MESSAGE_STATUS_SUBMITTED – Especifica que el servicio de mensajería subyacente ha depositado el mensaje para su transferencia, sea por una llamada a la función **cmc_send_message_object()**, sea por una llamada a la función **cmc_commit_object()** para comprometer un objeto de mensaje a un contenedor de mensajes de tipo apartado de salida. En algunos entornos de implementación (por ejemplo, usuario desconectado) no es posible tratar inmediatamente una operación de envío de un mensaje. Esta bandera indica que aunque el mensaje aparece en un contenedor de mensajes, el servicio de mensajería subyacente lo ha marcado para depósito.

CMC_MESSAGE_STATUS_SENT – Especifica que el mensaje está en curso de envío por el servicio de mensajería subyacente. En algunos entornos de implementación, la transferencia de un mensaje por el servicio de mensajería subyacente puede no ser inmediata. En tales casos, el mensaje puede aparecer en el contenedor aunque haya sido enviado por la aplicación. Esta bandera indica ese estado.

Ésta es una propiedad de tipo **CMC_pv_flags**.

5.4.13 Prioridad

NOMBRE

Prioridad de mensaje (*message priority*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_PRIORITY      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Priority//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la prioridad del mensaje. La propiedad puede establecerse como valor por defecto. Se puede fijar cuando se crea el mensaje.

Son valores válidos para esta propiedad:

CMC_PRIORITY_URGENT
CMC_PRIORITY_NORMAL
CMC_PRIORITY_LOW

CMC_PRIORITY_URGENT – Especifica que el mensaje tiene un alto nivel de prioridad (urgente).

CMC_PRIORITY_NORMAL – Especifica que el mensaje tiene el nivel de prioridad nominal. Este es el valor por defecto.

CMC_PRIORITY_LOW – Especifica que el mensaje tiene un bajo nivel de prioridad.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.4.14 Recibo solicitado

NOMBRE

Recibo de mensaje solicitado (*message receipt requested*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_RECEIPT_REQUESTED              \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Receipt Requested//EN"
```

DESCRIPCIÓN

Esta propiedad indica si se ha solicita un recibo del mensaje enviado.

Son valores válidos para esta propiedad:

CMC_RECEIPT_RN
CMC_RECEIPT_NRN
CMC_RECEIPT_BOTH
CMC_RECEIPT_NONE

CMC_RECEIPT_RN – Se solicita que se retorne una notificación de recibo solamente cuando el recibiente haya recibido el mensaje en cuestión.

CMC_RECEIPT_NRN – Solicita que se retorne una notificación de no recepción solamente cuando el recibiente no haya recibido el mensaje en cuestión.

CMC_RECEIPT_BOTH – Solicita que se retorne siempre una notificación, sea de recepción, sea de no recepción, según que el recibiente haya o no recibido el mensaje en cuestión.

CMC_RECEIPT_NONE – Solicita que no se retorne una notificación relativa a la recepción independientemente de que el recibiente haya recibido o no el mensaje en cuestión.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.4.15 Tipo de recepción

NOMBRE

Tipo de notificación relativa a la recepción del mensaje (*message receipt type*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_RECEIPT_TYPE                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Receipt Type//EN"
```

DESCRIPCIÓN

Especifica el tipo de recepción retornada en lo que respecta a la recepción del mensaje en cuestión. Se utiliza para indicar si el mensaje ha sido o no recibido por el recibiente deseado.

Son valores válidos para esta propiedad:

CMC_RECEIPT_RN
CMC_RECEIPT_NRN

CMC_RECEIPT_RN – Especifica que ésta es una notificación de recepción.

CMC_RECEIPT_NRN – Especifica que ésta es una notificación de no recepción.

Esta propiedad sólo es pertinente si la propiedad de tipo de mensaje tiene el valor CMC_MESSAGE_TYPE_RECEIPT.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.4.16 Informe solicitado

NOMBRE

Informe de mensaje solicitado (*message report requested*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_REPORT_REQUESTED                                \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Report Requested//EN"
```

DESCRIPCIÓN

Especifica el tipo de informe que habrá de retornarse para el mensaje en cuestión. Se utiliza para indicar si el mensaje ha sido o no entregado por e sistema de transporte de mensajería subyacente.

Son valores válidos para esta propiedad:

CMC_REPORT_DR
CMC_REPORT_NDR
CMC_REPORT_BOTH
CMC_REPORT_NONE

CMC_REPORT_DR – Especifica que se ha solicitado un informe de entrega.

CMC_REPORT_NDN – Especifica que se ha solicitado un informe de no entrega.

CMC_REPORT_BOTH – Especifica que se ha solicitado un informe de entrega o de no entrega, de los dos el que sea aplicable.

CMC_REPORT_NONE – Especifica que se ha solicitado que no se retorne un informe de entrega, ni de no entrega.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.4.17 Rol (o papel)

NOMBRE

Rol de mensaje (*message role*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_ROLE                                            \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Role//EN"
```

DESCRIPCIÓN

El rol del mensaje en cuestión.

Son valores válidos para esta propiedad:

CMC_MESSAGE_ROLE_ORIGINAL
CMC_MESSAGE_ROLE_RETURNED
CMC_MESSAGE_ROLE_FORWARDED
CMC_MESSAGE_ROLE_REPLIED
CMC_MESSAGE_ROLE_OBSOLETED
CMC_MESSAGE_ROLE_RESENT

CMC_MESSAGE_ROLE_ORIGINAL – Especifica que éste es el mensaje original.

CMC_MESSAGE_ROLE_RETURNED – Especifica que éste es un mensaje retornado, que es el contenido de otro mensaje.

CMC_MESSAGE_ROLE_FORWARDED – Especifica que éste es un mensaje reenviado, que es el contenido de otro mensaje.

CMC_MESSAGE_ROLE_REPLIED – Especifica que éste es un mensaje de respuesta a otro mensaje.

CMC_MESSAGE_ROLE_OBSOLETED – Especifica que éste es un mensaje caducado.

CMC_MESSAGE_ROLE_RESENT – Especifica que ésta es una copia enviada de nuevo de otro mensaje, el original.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.4.18 Sensibilidad

NOMBRE

Sensibilidad del mensaje (*message sensitivity*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_SENSITIVITY          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Sensitivity//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la sensibilidad del mensaje.

Son valores válidos de esta propiedad:

CMC_MESSAGE_SENSITIVITY_PERSONAL
CMC_MESSAGE_SENSITIVITY_PRIVATE
CMC_MESSAGE_SENSITIVITY_CONFIDENTIAL
CMC_MESSAGE_SENSITIVITY_NONE

CMC_MESSAGE_SENSITIVITY_PERSONAL – Especifica que el mensaje es personal.

CMC_MESSAGE_SENSITIVITY_PRIVATE – Especifica que el mensaje es privado.

CMC_MESSAGE_SENSITIVITY_CONFIDENTIAL – Especifica que el mensaje es confidencial.

CMC_MESSAGE_SENSITIVITY_NONE – Especifica que el mensaje es insensible.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.4.19 Tamaño

NOMBRE

Tamaño del mensaje (*message size*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_SIZE          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Size//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el tamaño del mensaje.

Ésta es una propiedad de tipo **CMC_pv_uint32**.

5.4.20 Asunto

NOMBRE

Asunto del mensaje (*message subject*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_SUBJECT        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Subject//EN"
```

DESCRIPCIÓN

Esta propiedad indica el asunto del mensaje. El valor por defecto de esta propiedad es una cadena nula.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.4.21 Hora de recepción

NOMBRE

Hora (o tiempo) de recepción del mensaje (*message time received*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_TIME_RECEIVED        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Time Received//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la fecha y hora de recepción del mensaje.

Ésta es una propiedad de tipo **CMC_pv_iso_date_time**.

5.4.22 Hora de envío

NOMBRE

Hora (o tiempo) de envío del mensaje (*message time sent*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_TIME_SENT     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Time Sent//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la fecha y hora de envío del mensaje.

Esta propiedad la fija el servicio mediante la función **cmc_send_message_object()**.

Ésta es una propiedad de tipo **CMC_pv_iso_date_time**.

5.4.23 Tipo

NOMBRE

Tipo del mensaje (*message type*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_TYPE        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Type//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el tipo del mensaje. El soporte de esta propiedad es facultativo para las implementaciones conformes con esta Recomendación. En el caso de mensajes recibidos o existentes, la propiedad puede ser creada por el sistema de mensajería. En el caso de mensajes de nueva creación, la propiedad se crea por una llamada a la función **cmc_add_properties()**. En el caso de mensajes de nueva creación, la propiedad puede también ser modificada por una llamada a **cmc_add_properties()** o suprimida por una llamada a **cmc_delete_properties()**.

Son valores válidos para esta propiedad:

CMC_MT_IPM
CMC_MT_RECEIPT
CMC_MT_EDI

CMC_MT_DIRECTORY
CMC_MT_DOCMGMT
CMC_MT_WORKFLOW
CMC_MT_CALSCHED

CMC_MT_IPM – Mensaje de correo electrónico, o mensaje interpersonal según la terminología de la Recomendación X.400.

CMC_MT_RECEIPT – Un recibo de mensajería. Este tipo de mensaje se utiliza para la notificación de recepción y la notificación de no recepción. Puede ser también útil para otros recibos de mensajes.

Los siguientes tipos de mensajes están reservados para finalidades especificadas. Los valores representan el trabajo que se encuentra en curso en la Asociación XAPIA y otros grupos industriales. Estos tipos de mensajes pueden modificarse en futuras versiones de esta Recomendación, para reflejar la compleción de este trabajo.

CMC_MT_EDI – Mensaje de tipo intercambio electrónico de datos. La forma y el formato de los mensajes EDI no están definidos en esta Recomendación.

CMC_MT_DIRECTORY – Mensaje de tipo servicios de directorio. Este tipo de mensaje prevé la utilización de servicios de mensajería como un transporte para funciones de interrogación de directorios. La forma y el formato de los mensajes de directorio no están definidos en esta Recomendación.

CMC_MT_DOCMGMT – Mensaje de tipo gestión de documentos. Este tipo de mensaje prevé el acceso y la búsqueda de servicios de biblioteca utilizando el servicio de mensajería como un transporte para las funciones de interrogación de la gestión de documentos. La forma y el formato de los mensajes de gestión de documentos no están definidos en esta Recomendación.

CMC_MT_WORKFLOW – Mensaje de tipo de gestión de flujo de trabajo. Este tipo de mensaje facilita el tratamiento automatizado de procesos comerciales utilizando el servicio de mensajería como un transporte para las funciones de flujo de trabajo. La forma y el formato de los mensajes de gestión de flujos de trabajo no están definidos en esta Recomendación.

CMC_MT_CALSCHED – Mensaje de tipo calendarización (*calendaring*) e itinerarización (*scheduling*). Este tipo de mensaje permite utilizar el servicio de mensajería como un transporte para funciones de calendarización e itinerarización. La forma y el formato de los mensajes de calendarización e itinerarización no están definidos en esta Recomendación. Existen otras especificaciones de la Asociación XAPIA que prevén la definición de la interoperabilidad de la calendarización y la itinerarización.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.5 Propiedades del objeto de contenedor de mensajes

Un objeto de contenedor de mensajes es una colección de propiedades de contenedor, objetos de mensajes del contenedor y, posiblemente, de otros contenedores de mensajes. Las subcláusulas siguientes definen, declaran y describen las propiedades del objeto de contenedor de mensajes.

5.5.1 Vástago permitido

NOMBRE

Vástago de contenedor de mensajes permitido (*message container child allowed*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_CONTAINER_CHILD_ALLOWED                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Child Allowed//EN"
```

DESCRIPCIÓN

Esta propiedad permite o prohíbe la existencia de un vástago del contenedor de mensajes.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.5.2 Comentario

NOMBRE

Comentario de contenedor de mensajes (*message container comment*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_CONTAINER_COMMENT              \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Comment//EN"
```

DESCRIPCIÓN

Expresa un comentario descriptivo sobre el contenedor de mensajes.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.5.3 Ubicación

NOMBRE

Ubicación de contenedor de mensajes (*message container location*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_CONTAINER_LOCATION            \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Location//EN"
```

DESCRIPCIÓN

Indica la ubicación del contenedor de mensajes.

Son valores válidos para esta propiedad:

CMC_MESSAGE_CONTAINER_LOCATION_LOCAL
CMC_MESSAGE_CONTAINER_LOCATION_SERVER
CMC_MESSAGE_CONTAINER_LOCATION_UNKNOWN

CMC_MESSAGE_CONTAINER_LOCATION_LOCAL – Especifica que el contenedor de mensajes es local y no se encuentra en el servidor de mensajería.

CMC_MESSAGE_CONTAINER_LOCATION_SERVER – Especifica que el contenedor de mensajes se encuentra en el servidor de mensajería.

CMC_MESSAGE_CONTAINER_LOCATION_UNKNOWN – Especifica que la ubicación del contenedor de mensajes es desconocida.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.5.4 Nombre

NOMBRE

Nombre de contenedor de mensajes (*message container name*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_CONTAINER_NAME       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Name//EN"
```

DESCRIPCIÓN

Especifica el nombre del contenedor de mensajes.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.5.5 Clase de objeto

NOMBRE

Clase de objeto de contenedor de mensajes (*message container object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"
```

DESCRIPCIÓN

Esta propiedad define la clase del objeto como un contenedor de mensajes.

Esta propiedad se crea por **cmc_open_object_handle()**.

El único valor válido para esta propiedad es CMC_PT_OBJECT_CLASS_MESSAGE_CONTAINER, que especifica que la clase del objeto es un contenedor de mensajes.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.5.6 Progenitor

NOMBRE

Progenitor de contenedor de mensajes (*message container parent*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_CONTAINER_PARENT                \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Parent//EN"
```

DESCRIPCIÓN

Especifica el progenitor del contenedor de mensajes. Si la implementación soporta el anidamiento de contenedores de mensajes, esta propiedad especifica el contenedor de mensajes progenitor. Si el contenedor de mensajes es el nivel superior, esta propiedad no está presente. De lo contrario, es obligatoria.

Ésta es una propiedad de tipo **CMC_pv_object_handle**.

5.5.7 Nombre del servidor

NOMBRE

Nombre del servidor del contenedor de mensajes (*message container server name*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_CONTAINER_SERVER_NAME           \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Server Name//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el nombre del servidor en que se encuentra el contenedor de mensajes.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.5.8 Compartido

NOMBRE

Contendor de mensajes compartido (*message container shared*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_CONTAINER_SHARED     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Shared//EN"
```

DESCRIPCIÓN

Esta propiedad especifica si más de un usuario tienen acceso al contenedor de mensajes.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.5.9 Tipo

NOMBRE

Tipo de contenedor de mensajes (*message container type*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_MESSAGE_CONTAINER_TYPE       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Type//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el tipo del contenedor del mensaje.

Son valores válidos para esta propiedad:

CMC_MCT_DELETED
CMC_MCT_DRAFTS
CMC_MCT_FILED
CMC_MCT_INBOX
CMC_MCT_OUTBOX
CMC_MCT_SENT

CMC_MCT_DELETED – Especifica que el contenedor de mensajes es para mensajes suprimidos.

CMC_MCT_DRAFTS – Especifica que el contenedor de mensajes es para proyectos (o borradores) de mensajes.

CMC_MCT_FILED – Especifica que el contenedor de mensajes es para mensajes archivados.

CMC_MCT_INBOX – Especifica que el contenedor de mensajes es un apartado de entrada. Una implementación puede tener más de un apartado de entrada.

CMC_MCT_OUTBOX – Especifica que el contenedor de mensajes es para mensajes de salida. Una implementación debe tener por lo menos un apartado de salida, que es obligatorio. Los objetos comprometidos al apartado de salida no pueden ser modificados. Los objetos comprometidos sólo pueden ser suprimidos o copiados.

CMC_MCT_SENT – Especifica que el contenedor de mensajes es para mensajes que han sido enviados. El apartado de mensajes enviados es facultativo. Una implementación no puede tener más de un apartado de mensajes enviados (0-1 apartados de mensajes enviados).

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.6 Propiedades del objeto información por cada recibiente

Los objetos de información por cada recibiente son componentes de un informe que se genera para indicar el estado de entrega o no entrega de un mensaje. Las siguientes subcláusulas definen, declaran y describen las propiedades del objeto de información por cada recibiente.

5.6.1 Comentario

NOMBRE

Comentario de información por cada recibiente (*per recipient information comment*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PRI_COMMENT                                  \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Comment//EN"
```

DESCRIPCIÓN

Esta propiedad proporciona información suplementaria sobre el estado del mensaje.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.6.2 Hora de entrega

NOMBRE

Hora de entrega de información por cada recibiente (*per recipient information delivery time*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PRI_DELIVERY_TIME                            \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Delivery Time//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la fecha y hora en que se entrego el mensaje en cuestión.

Esta propiedad la fija el servicio mediante la función **cmc_send_message_object()**.

Esta propiedad es obligatoria si el tipo de información por cada recibiente es CMC_PRI_DR.

Ésta es una propiedad de tipo **CMC_pv_iso_date_time**.

5.6.3 Diagnóstico

NOMBRE

Diagnóstico de información por cada recibiente (*per recipient information diagnostic*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PRI DIAGNOSTIC                               \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Diagnostic//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la información de diagnóstico detallada, indicando el motivo por el cual no se entregó el mensaje en cuestión.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.6.4 Clase de objeto

NOMBRE

Clase de objeto de información por cada recibiente (*per recipient information object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"
```

DESCRIPCIÓN

Esta propiedad define la clase del objeto como una información por cada recibiente.

Esta propiedad se crea por la función **cmc_open_object_handle()**.

El único valor válido para esta propiedad es CMC_PT_OC_PER_RECIPIENT_INFORMATION, que especifica que la clase del objeto es una información por cada recibiente.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.6.5 Motivo (o razón)

NOMBRE

Motivo (o razón) de la información por cada recibiente (*per recipient information reason*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PRI_REASON                                      \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Reason//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el motivo por el cual se generó la información por cada recibiente.

Esta propiedad es obligatoria si el tipo de la información por cada recibiente es CMC_PRI_NDR.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.6.6 Dirección de recibiente

NOMBRE

Dirección de recibiente para la información por cada recibiente (*per recipient information recipient address*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PRI_RECIPIENT_ADDRESS                           \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Recipient Address//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la dirección del recibiente deseado para el mensaje en cuestión, el cual, o bien recibió el mensaje, o no pudo recibirlo, según se indica por el tipo de información por cada recibiente. El recibiente de este informe no sería usualmente el originador del mensaje en cuestión. El recibiente de un informe no puede responder al informe.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.6.7 Nombre de recibiente

NOMBRE

Nombre de recibiente para la información por cada recibiente (*per recipient information recipient name*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PRI_RECIPIENT_NAME                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Recipient Name//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el nombre del recibiente deseado para el mensaje en cuestión, el cual, o bien recibió el mensaje, o no pudo recibirlo, como se indica por el tipo de información por cada recibiente. El recibiente de este informe no sería usualmente el originador del mensaje en cuestión. El recibiente de un informe no puede responder al informe.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.6.8 Tipo

NOMBRE

Tipo de información por cada recibiente (*per recipient information type*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PRI_TYPE                                              \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Type//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el tipo de la información por cada recibiente.

Son valores válidos para esta propiedad:

CMC_PRI_DR
CMC_PRI_NDR
CMC_PRI_UNKNOWN

CMC_PRI_DR – Especifica que la información por cada recibiente es del tipo notificación de entrega.

CMC_PRI_NDR – Especifica que la información por cada recibiente es del tipo notificación de no entrega.

CMC_PRI_UNKNOWN – Especifica que el tipo de la información por cada recibiente no se especificó o no es aplicable.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.7 Propiedades del objeto de contenedor de perfiles

El objeto de contenedor de perfiles identifica la información específica del contexto y de la configuración de la sesión. Las siguientes subcláusulas definen, declaran y describen propiedades del contenedor de perfiles.

5.7.1 Acción automática

NOMBRE

Acción automática de contenedor de perfiles (*profile container auto-action*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_CONTAINER_AUTO_ACTION      \
    "–//NONSGML Profile Container Auto Action//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la acción automática o la disposición que habrá de tomarse sobre el mensaje después de enviado. El soporte de esta propiedad es facultativa para las implementaciones conformes con esta Recomendación. En el caso de mensajes de nueva creación, la propiedad se crea por la función **cmc_add_properties()**. En el caso de mensajes de nueva creación, la propiedad puede también ser modificada mediante una llamada a **cmc_add_properties()**, o suprimida mediante una llamada **cmc_delete_properties()**. Sobre este valor puede prevalecer la propiedad CMC_PT_MESSAGE_AUTO_ACTION del objeto de mensaje, en cada mensaje.

El valor válido para esta propiedad es:

CMC_AA_DELETE

Fijado: El sistema de mensajería subyacente suprime el mensaje especificado después de depositarlo con éxito para transferencia.

Despejado: El mensaje se coloca en la carpeta de mensajes enviados, si ésta existe. De lo contrario, se suprime el mensaje.

Ésta es una propiedad de tipo **CMC_pv_flags**.

5.7.2 Juego de caracteres

NOMBRE

Juego de caracteres del contenedor de perfiles (*profile container character set*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_CHARACTER_SET            \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Character Set//EN"
```

DESCRIPCIÓN

Especifica el juego de caracteres que se utilizará para transportar datos en forma de cadenas de caracteres entre el usuario y la CMC. El valor de esta propiedad es una matriz de identificadores de objetos de juego de caracteres, asociados con la implementación. La matriz es una matriz contada. El primer identificador de juego de caracteres de la matriz es el juego de caracteres por defecto, que se utiliza si el usuario no especifica uno explícitamente en la función **cmc_logon()**. En lo referente a los identificadores de objetos para los juegos de caracteres comunes, véase la cláusula sobre la información específica de la plataforma en el B.2.4.

Ésta es una propiedad de tipo **CMC_pv_array_of_guid**.

5.7.3 Conformidad

NOMBRE

Conformidad del contenedor de perfiles (*profile container conformance*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_CONF            \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Conf//EN"
```

DESCRIPCIÓN

Especifica el nivel de conformidad de la implementación. El valor de la propiedad será o bien CMC_CONF_SIMPLE_CMC si la implementación soporta la CMC simple, o CMC_CONF_FULL_CMC si la implementación soporta una versión autónoma de la CMC completa. Un valor de CMC_CONF_FULL_CMC implica que la implementación soporta también la interfaz CMC simple, como se requiere en la cláusula relativa a la conformidad.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.7.4 Servicio por defecto

NOMBRE

Servicio por defecto para el contenedor de perfiles (*profile container default service*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_DEFAULT_SERVICE          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Default Service//EN"
```

DESCRIPCIÓN

El nombre del servicio por defecto. Se asignará el valor de puntero NULL si no hay disponible ningún nombre de servicio por defecto. Esta propiedad, junto con CMC_PT_PROFILE_DEFAULT_USER, puede utilizarse como valor por defecto para el nombre de servicio y el nombre de usuario para la función **cmc_logon()**. Se retornará en el juego de caracteres por defecto de la implementación.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.7.5 Usuario por defecto

NOMBRE

Usuario por defecto del contenedor de perfiles (*profile container default user*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_DEFAULT_USER          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Default User//EN"
```

DESCRIPCIÓN

Especifica el nombre del usuario CMC por defecto. Se asignará un valor de puntero NULL si no está disponible ningún nombre de usuario por defecto. Esta propiedad, junto con la de CMC_PROFILE_DEFAULT_SERVICE, puede utilizarse como valor por defecto para el nombre del proveedor y el nombre del usuario para la función **cmc_logon()**. Se retornará en el juego de caracteres por defecto de la implementación.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.7.6 Terminador de línea

NOMBRE

Terminador de línea de contenedor de perfiles (*profile container line term*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_LINE_TERM     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Line Term//EN"
```

DESCRIPCIÓN

Especifica los caracteres de terminación de línea que habrán de utilizarse para terminar líneas en las cadenas de caracteres. Esta propiedad puede tener los valores: CMC_LINE_TERM_CRLF si el delimitador de línea es un carácter de retorno del carro seguido de un carácter de cambio de renglón, CMC_LINE_TERM_LF si el delimitador de línea es un carácter cambio de renglón, o CMC_LINE_TERM_CR si el delimitador de línea es un carácter de retorno del carro.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.7.7 Clase de objeto

NOMBRE

Clase de objeto de contenedor de perfiles (*profile container object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS                          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"
```

DESCRIPCIÓN

Esta propiedad define la clase del objeto como un libro de direcciones.

El único valor válido de esta propiedad es CMC_PT_OBJECT_CLASS_PROFILE, que especifica que la clase del objeto es un objeto contenedor de perfiles.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.7.8 Extensiones de objetos soportadas

NOMBRE

Extensiones de objetos de contenedores de perfil soportadas (*profile container object extensions supported*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_OBJECT_EXT    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Object Ext//EN"
```

DESCRIPCIÓN

Especifica las extensiones de clase de objetos soportadas por la implementación. Los valores de la propiedad son una matriz de identificadores globales de clases de objetos para las extensiones de clases de objetos soportadas por la implementación. No hay un orden implícito para los GUID de los objetos en la matriz.

Ésta es una propiedad de tipo **CMC_pv_array_guid**.

5.7.9 Objetos soportados

NOMBRE

Objetos de contenedores de perfiles soportados (*profile container objects supported*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_OBJECT_SUP    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Object Sup//EN"
```

DESCRIPCIÓN

Especifica las clases de objetos soportadas por la implementación. Los valores de esta propiedad son una matriz de los identificadores globales de las clases de objetos para las clases de objetos soportadas por la implementación. No hay un orden implícito para los GUID de los objetos en la matriz.

Ésta es una propiedad de tipo **CMC_pv_array_guid**.

5.7.10 Propiedades soportadas

NOMBRE

Propiedades de contenedores de perfiles soportadas (*profile container properties supported*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_PROP_SUP      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Prop Sup//EN"
```

DESCRIPCIÓN

Especifica las propiedades soportadas por la implementación. Los valores de esta propiedad son una matriz de los identificadores globales de propiedades para las propiedades de objetos soportadas por la implementación. No hay un orden implícito para los GUID de propiedades en la matriz.

Ésta es una propiedad de tipo **CMC_pv_array_guid**.

5.7.11 Extensiones de propiedades soportadas

NOMBRE

Propiedades de contenedores de perfiles soportadas (*profile container properties supported*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_PROP_EXT      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Prop Ext//EN"
```

DESCRIPCIÓN

Especifica las extensiones de propiedades soportadas por la implementación. Los valores de esta propiedad son una matriz de los identificadores globales de propiedades para las extensiones de propiedades de objetos soportadas por la implementación. No hay un orden implícito para los GUID de propiedades en la matriz.

Ésta es una propiedad de tipo **CMC_pv_array_guid**.

5.7.12 Contraseña requerida

NOMBRE

Contraseña del contenedor de perfiles requerida (*profile container required password*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_REQ_PASSWORD          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Req Password//EN"
```

DESCRIPCIÓN

Indica si se requiere o no una contraseña para establecer una sesión con el servicio. Los valores de la propiedad son: CMC_REQUIRED_NO si no se requiere contraseña para establecer la sesión, CMC_REQUIRED_OPT si la contraseña es facultativa para establecer la sesión, o CMC_REQUIRED_YES si se requiere la contraseña para establecer la sesión.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.7.13 Servicio requerido

NOMBRE

(Nombre de) Servicio requerido para el contenedor de perfiles (*profile container required service*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_REQ_SERVICE          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Req Service//EN"
```

DESCRIPCIÓN

Indica si se requiere o no un nombre de servicio para establecer una sesión con el servicio. Los valores de la propiedad son: CMC_REQUIRED_NO si no se requiere un nombre de servicio para establecer la sesión, CMC_REQUIRED_OPT si un nombre de servicio es facultativo para establecer la sesión, o CMC_REQUIRED_YES si se requiere un nombre de servicio para establecer la sesión.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.7.14 Usuario requerido

NOMBRE

(Nombre de) Usuario requerido para el contenedor de perfiles (*profile container required user*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_REQ_USER      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Req User//EN"
```

DESCRIPCIÓN

Indica si se requiere o no un nombre de usuario para establecer una sesión con el servicio. Los valores de la propiedad son: CMC_REQUIRED_NO si no se requiere un nombre de usuario para establecer la sesión, CMC_REQUIRED_OPT si un nombre de usuario es facultativo para establecer la sesión, o CMC_REQUIRED_YES si se requiere un nombre de usuario para establecer la sesión.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.7.15 Soporte de cadenas contadas

NOMBRE

Soporte de cadenas contadas por el contenedor de perfiles (*profile container support counted strings*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_SUP_COUNTED_STR        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Sup Counted Str//EN"
```

DESCRIPCIÓN

Indica si el servicio soporta cadenas contadas. El valor de la propiedad se fijará a verdadero si la bandera CMC_COUNTED_STRING_TYPE está soportada durante el establecimiento de la sesión.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.7.16 Soporte de ausencia de marca como leer

NOMBRE

Soporte de ausencia de marca como leer por el contenedor de perfiles (*profile container support no mark as read*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_SUP_NOMKMSGREAD        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Sup NoMkMsgRead//EN"
```

DESCRIPCIÓN

Indica si el servicio soporta la operación CMC_DO_NOT_MARK_AS_READ de la función **cmc_read()**. El valor de la propiedad se fijará a verdadero si la función **cmc_read()** soporta la bandera CMC_DO_NOT_MARK_AS_READ.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.7.17 Interfaz de usuario disponible

NOMBRE

Interfaz de usuario disponible en el contenedor de perfiles (*profile container user interface available*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_UI_AVAIL      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile UI Avail//EN"
```

DESCRIPCIÓN

Indica si una interfaz de usuario está disponible para la introducción y resolución de parámetros. El valor de la propiedad se fijará a verdadero si la implementación CMC proporciona la interfaz UI.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.7.18 Usuarios

NOMBRE

Usuarios del contenedor de perfiles (*profile container users*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_USERS         \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Users//EN"
```

DESCRIPCIÓN

Indica los usuarios que actualmente tienen establecida una sesión con el contenedor raíz. Los valores de esta propiedad son los nombres de recibiente de los usuarios que tienen establecida una sesión con el contenedor raíz. El soporte de esta propiedad es facultativo para las implementaciones conformes con esta Recomendación.

No hay un orden implícito para los nombres de recibientes subsiguientes en la matriz.

Ésta es una propiedad de tipo **CMC_pv_array_string**.

5.7.19 Versión de la implementación

NOMBRE

Versión de la implementación del contenedor de perfiles (*profile container version of the implementation*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_VER_IMPLEM    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Ver Implem//EN"
```

DESCRIPCIÓN

Indica la versión de la implementación. El valor de esta propiedad se fijará al número de la versión de la implementación, multiplicado por 100. Por ejemplo, la versión 1.01 se retornará como 101.

Ésta es una propiedad de tipo **CMC_pv_uint16**.

5.7.20 Versión de la especificación

NOMBRE

Versión de la especificación del contenedor de perfiles (*profile container version of the specification*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_PROFILE_VER_SPEC      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Ver Spec//EN"
```

DESCRIPCIÓN

La versión de la especificación CMC soportada por la implementación. El valor de la propiedad se fijará al número de la versión de la especificación CMC soportada por la implementación, multiplicado por 100. Por ejemplo, la versión 1.00 se retornará como 100.

Ésta es una propiedad de tipo **CMC_pv_uint16**.

5.8 Propiedades de objetos de recibientes

Los objetos de recibientes identifican a usuarios individuales en un servicio de mensajería. Un objeto de recibiente no es un objeto de contenedor. La siguientes subcláusulas definen, declaran, y describen propiedades de objetos de recibientes.

5.8.1 Dirección

NOMBRE

Dirección de recibientes (*recipient address*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_RECIPIENT_ADDRESS     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Address//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la dirección del recibiente. El formato de la cadena depende de la implementación.

En aplicaciones de cabeceras, la cabecera puede añadir la dirección del objeto de recibiente cuyo rol es originador.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.8.2 Retorno de contenido solicitado

NOMBRE

Retorno de contenido del recibiente solicitado (*recipient content return requested*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_RECIPIENT_CONTENT_RETURN_REQUESTED                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Content Return Requested//EN"
```

DESCRIPCIÓN

Esta propiedad especifica si el mensaje en cuestión debe retornarse con un informe de no entrega en el caso de fracaso de la entrega. Si no se solicita informe, el mensaje no se retornará cualquiera que sea la indicación de esta propiedad.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.8.3 Nombre

NOMBRE

Nombre de recibiente (*recipient name*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_RECIPIENT_NAME        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Name//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el nombre visualizable del recibiente. El formato de la cadena depende de la implementación.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.8.4 Clase de objeto

NOMBRE

Clase de objeto de recibiente (*recipient object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"
```

DESCRIPCIÓN

Esta propiedad define la clase del objeto como un recibiente.

Esta propiedad se crea por **cmc_open_object_handle()**.

El único valor de esta propiedad es CMC_PT_OBJECT_CLASS_RECIPIENT, que especifica que la clase del objeto es un recibiente.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.8.5 Recibo solicitado

NOMBRE

Recibo del recibiente solicitado (*recipient receipt requested*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_RECIPIENT_RECEIPT_REQUESTED                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Receipt Requested//EN"
```

DESCRIPCIÓN

Indica el tipo del recibo que habrá de retornarse para el mensaje.

Si tanto el objeto de mensaje como el objeto de recibiente especifican la propiedad de recibo solicitado, el valor especificado en el objeto de recibiente prevalecerá sobre el valor especificado en el objeto de mensaje. La implementación no está obligada a soportar está propiedad en el nivel de objeto de recibiente.

Son valores válidos para esta propiedad:

CMC_RECEIPT_RN
CMC_RECEIPT_NRN
CMC_RECEIPT_BOTH
CMC_RECEIPT_NONE

CMC_RECEIPT_RN – Solicita que se retorne una notificación de recepción solamente si el recibiente ha recibido el mensaje.

CMC_RECEIPT_NRN – Solicita que se retorne una notificación de recepción solamente si el recibiente no ha recibido el mensaje.

CMC_RECEIPT_BOTH – Solicita que se retorne o bien una notificación de recepción cuando el recibiente haya recibido el mensaje en cuestión, o una notificación de no recepción cuando no lo haya recibido.

CMC_RECEIPT_NONE – Solicita que no se retorne en ningún caso una notificación relativa a la recepción o la no recepción del mensaje.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.8.6 Informe solicitado

NOMBRE

Informe de recibiente solicitado (*recipient report requested*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_RECIPIENT_REPORT_REQUESTED                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Report Requested//EN"
```

DESCRIPCIÓN

Especifica el tipo del informe que habrá de retornarse para el mensaje en cuestión. Se utiliza para indicar si el mensaje ha sido entregado o no entregado por el sistema de transporte de mensajería subyacente.

Son valores de esta propiedad:

CMC_REPORT_DR
CMC_REPORT_NDR
CMC_REPORT_BOTH
CMC_REPORT_NONE

CMC_REPORT_DR – Especifica que se ha solicitado un informe de entrega.

CMC_REPORT_NDR – Especifica que se ha solicitado un informe de no entrega.

CMC_REPORT_BOTH – Especifica que se ha solicitado un informe de entrega o un informe de no entrega, de los dos el que sea aplicable.

CMC_REPORT_NONE – Especifica que no se ha solicitado ninguno de los dos informes ni el de entrega ni el de no entrega.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.8.7 Bandera de responsabilidad

NOMBRE

Bandera de responsabilidad de recibiente (*recipient responsibility flag*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_RECIPIENT_RESPONSIBILITY_FLAG          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Responsibility Flag//EN"
```

DESCRIPCIÓN

Esta propiedad especifica un indicador que determina si el recibiente en cuestión debe recibir una copia del mensaje. Es útil en aplicaciones de cabeceras y en aquellas situaciones en las que una aplicación puede ganar acceso a múltiples versiones de la CMC.

El valor por defecto de esta propiedad es CMC_TRUE.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.8.8 Rol (o papel)

NOMBRE

Rol de recibiente (*recipient role*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_RECIPIENT_ROLE        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Role//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el rol del recibiente.

Son valores válidos para esta propiedad:

CMC_RECIPIENT_ROLE_TO
CMC_RECIPIENT_ROLE_CC
CMC_RECIPIENT_ROLE_BCC
CMC_RECIPIENT_ROLE_ORIGINATOR
CMC_RECIPIENT_ROLE_AUTHORIZING_USER
CMC_RECIPIENT_ROLE_REPLY_TO
CMC_RECIPIENT_ROLE_FORWARDED
CMC_RECIPIENT_ROLE_ACTUAL
CMC_RECIPIENT_ROLE_INTENDED

CMC_RECIPIENT_ROLE_TO – Especifica el recibiente primario.

CMC_RECIPIENT_ROLE_CC – Especifica el recibiente de una copia (o copia de papel carbón, *carbon copy*).

CMC_RECIPIENT_ROLE_BCC – Especifica el recibiente de una copia ciega (*blind carbon copy*).

CMC_RECIPIENT_ROLE_ORIGINATOR – Especifica el originador.

CMC_RECIPIENT_ROLE_AUTHORIZING_USER – Especifica el usuario autorizante.

CMC_RECIPIENT_ROLE_REPLY_TO – Especifica el recibiente al que debe dirigirse la respuesta.

CMC_RECIPIENT_ROLE_FORWARDED – Especifica el recibiente a que se reenvía el mensaje.

CMC_RECIPIENT_ROLE_ACTUAL – Especifica el recibiente real.

CMC_RECIPIENT_ROLE_INTENDED – Especifica el recibiente deseado.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.8.9 Tipo

NOMBRE

Tipo de recibiente (*recipient type*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_RECIPIENT_TYPE        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Type//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el tipo del recibiente.

Son valores válidos para esta propiedad:

CMC_RCT_UNKNOWN (=0)
CMC_RCT_INDIVIDUAL
CMC_RCT_GROUP
CMC_RCT_REPORT_RECIPIENT

CMC_RCT_UNKNOWN – Especifica un tipo de recibiente desconocido.

CMC_RCT_INDIVIDUAL – Especifica que el recibiente es un individuo.

CMC_RCT_GROUP – Especifica que el recibiente es una lista de distribución.

CMC_RCT_REPORT_RECIPIENT – Especifica que el recibiente es recibiente de un mensaje de informe.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.9 Propiedades del objeto de informe

El objeto de informe es una colección de propiedades de objetos específicos del informe. Las subcláusulas siguientes definen, declaran y describen las propiedades del objeto de informe.

5.9.1 Identificador de aplicación

NOMBRE

Identificador de aplicación de informe (*report application id*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_APPLICATION_ID \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Application Id//EN"
```

DESCRIPCIÓN

Esta propiedad especifica un identificador globalmente único para el informe. Esta propiedad la fija la aplicación.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.9.2 Identificador

NOMBRE

Identificador de informe (*report id*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_ID     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Id//EN"
```

DESCRIPCIÓN

Esta propiedad especifica un identificador globalmente único para el informe. La fija la función **cmc_send_message_object()**, la define el servicio de mensajería (se establece en el momento del depósito) y es único en el dominio.

En aplicaciones de cabeceras, el identificador de mensaje puede ser añadido o actualizado por el llamante.

Ésta es una propiedad de tipo **CMC_pv_guid**.

5.9.3 Cuenta de ítems

NOMBRE

Cuenta de ítems de informe (*report item count*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_ITEM_COUNT                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Item Count//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el número de ítems de contenido de nivel superior contenidos en un informe. Esta cuenta no incluye los ítems de contenido anidados en otros ítems de contenido, o mensajes. Esta propiedad la fija la implementación.

Ésta es una propiedad de tipo **CMC_pv_uint32**.

5.9.4 Identificador del sistema de mensajería

NOMBRE

Identificador del sistema de mensajería de informes (*report messaging system id*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_MESSAGING_SYSTEM_ID                           \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Messaging System Id//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el identificador del sistema subyacente de transporte de mensajes o el identificador de la cabecera que creó este informe.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.9.5 Clase de objeto

NOMBRE

Clase de objeto de informe (*report object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"
```

DESCRIPCIÓN

Esta propiedad define la clase de objeto como un informe.

Esta propiedad se crea por la función **cmc_open_object_handle()**.

El único valor para esta propiedad es CMC_PT_OBJECT_CLASS_REPORT, que específica que la clase del objeto es un informe.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.9.6 Leído

NOMBRE

Informe leído (*report read*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_READ                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Read//EN"
```

DESCRIPCIÓN

Esta propiedad especifica si el informe ha sido leído.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.9.7 Tamaño

NOMBRE

Tamaño de informe (*report size*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_SIZE                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Size//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el tamaño del informe.

Ésta es una propiedad de tipo **CMC_pv_uint32**.

5.9.8 Asunto

NOMBRE

Asunto del informe (*report subject*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_SUBJECT                                 \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Subject//EN"
```

DESCRIPCIÓN

Esta propiedad indica el asunto del informe. Esta propiedad puede tener como valor por defecto una cadena nula.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.9.9 Identificador de mensaje de asunto

NOMBRE

Identificador de mensaje de asunto del informe (*report subject message id*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_SUBJECT_MESSAGE_ID                      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Subject Message Id//EN"
```

DESCRIPCIÓN

Esta propiedad identifica el usuario del mensaje que provocó la generación del informe en cuestión.

El valor de la propiedad puede ser una referencia textual o una aproximación textual del identificador de mensaje de la correspondencia anterior.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.9.10 Hora de recepción

NOMBRE

Hora de recepción del informe (*report time received*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_TIME_RECEIVED                           \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Time Received//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la fecha y hora en que se recibió el mensaje.

Ésta es una propiedad de tipo **CMC_pv_iso_date_time**.

5.9.11 Hora de envío

NOMBRE

Hora de envío del informe (*report time sent*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_TIME_SENT                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Time Sent//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la fecha y hora en que se envió el mensaje.

Esta propiedad la fija el servicio en **cmc_send_message_object**.

Ésta es una propiedad de tipo **CMC_pv_iso_date_time**.

5.9.12 No enviado

NOMBRE

Informe no enviado (*report unsent*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_REPORT_UNSENT                                       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Unsent//EN"
```

DESCRIPCIÓN

Esta propiedad especifica que el informe no ha sido enviado.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.10 Propiedades del objeto de contenedor raíz

La raíz es el objeto de contenedor medular esencial y está compuesto de diversas propiedades y otros objetos de contenedores. El contenedor raíz está compuesta de libros de direcciones (que contienen recibientes y otros libros de direcciones), un contenedor de perfiles, y contenedores de mensajes. El objeto de recibiente en el contenedor raíz no puede ser modificado.

Las siguientes subcláusulas definen, declaran y describen propiedades del objeto de contenedor raíz.

5.10.1 Vástago permitido

NOMBRE

Vástago de contenedor raíz permitido (*root container child allowed*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ROOT_CONTAINER_CHILD_ALLOWED          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Child Allowed//EN"
```

DESCRIPCIÓN

Esta propiedad permite o prohíbe la existencia de un vástago del contenedor raíz.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

5.10.2 Comentario

NOMBRE

Comentario de contenedor raíz (*root container comment*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ROOT_CONTAINER_COMMENT        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Comment//EN"
```

DESCRIPCIÓN

Esta propiedad proporciona un comentario descriptivo sobre el contenedor raíz.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.10.3 Ubicación

NOMBRE

Ubicación del contenedor raíz (*root container location*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ROOT_CONTAINER_LOCATION       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Location//EN"
```

DESCRIPCIÓN

Esta propiedad especifica la ubicación del contenedor raíz.

Son valores válidos para esta propiedad:

CMC_ROOT_CONTAINER_LOCATION_LOCAL
CMC_ROOT_CONTAINER_LOCATION_SERVER
CMC_ROOT_CONTAINER_LOCATION_UNKNOWN

CMC_ROOT_CONTAINER_LOCATION_LOCAL – Especifica que el contenedor raíz es local y no se encuentra en el servidor de mensajería.

CMC_ROOT_CONTAINER_LOCATION_SERVER – Especifica que el contenedor raíz se encuentra en el servidor de mensajería.

CMC_ROOT_CONTAINER_LOCATION_UNKNOWN – Especifica que la ubicación del contenedor raíz es desconocida.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.10.4 Nombre

NOMBRE

Nombre del contenedor raíz (*root container name*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ROOT_CONTAINER_NAME           \
    "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Name//EN"
```

DESCRIPCIÓN

Esta propiedad especifica el nombre del contenedor raíz.

Ésta es una propiedad de tipo **CMC_pv_string**.

5.10.5 Clase de objeto

NOMBRE

Clase de objeto de contenedor raíz (*root container object class*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_OBJECT_CLASS          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"
```

DESCRIPCIÓN

Esta propiedad define la clase del objeto como la raíz.

Esta propiedad se crea por la función **cmc_open_object_handle()**.

El único valor válido para esta propiedad es CMC_PT_OBJECT_CLASS_ROOT, que especifica que la clase del objeto es la raíz.

Ésta es una propiedad de tipo **CMC_pv_enum**.

5.10.6 Compartido

NOMBRE

Contenedor raíz compartido (*root container shared*)

DECLARACIÓN EN LENGUAJE C

```
#define CMC_PT_ROOT_CONTAINER_SHARED       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Shared//EN"
```

DESCRIPCIÓN

Esta propiedad especifica si el contenedor raíz está compartido con otra entidad.

Ésta es una propiedad de tipo **CMC_pv_boolean**.

6 Funciones de la interfaz

Esta cláusula define las funciones de la interfaz de llamada de mensajería común. Se especifican las funciones de la interfaz genérica y de la interfaz en lenguaje C. Las funciones de la interfaz en lenguaje C se repiten en el anexo A, "Sumario de declaraciones en el lenguaje de programación C".

6.1 Funciones de la CMC simple

La CMC simple ofrece un conjunto básico de funciones que están destinadas a proporcionar capacidades conscientes de la mensajería (*messaging-aware capabilities*) a aplicaciones habilitadas para mensajería (*messaging-enabled applications*). Estas funciones fueron publicadas anteriormente como CMC versión 1.0. El cuadro 14 recapitula las funciones de la interfaz CMC simple.

Cuadro 14/X.446 – Funciones de la interfaz CMC simple

Función	Descripción
Envío de mensajes	
Enviar (*send*)	Envía un mensaje de correos
Enviar documentos (*send documents*)	Función basada en cadenas de caracteres para el envío de correspondencia
Recepción de mensajes	
Tratar (*act on*)	Ejecuta una acción sobre un mensaje especificado
Listar (*list*)	Lista información de sumario sobre los mensajes que cumplen los criterios especificados
Leer (*read*)	Lee y retorna un mensaje especificado
Consulta de nombres	
Consultar (*look up*)	Consulta información de direccionamiento
Administración	
Liberar (*free*)	Libera memoria asignada por el servicio de mensajería
Terminar sesión (*log off*)	Termina una sesión con el servicio de mensajería
Establecer sesión (*log on*)	Establece una sesión con el servicio de mensajería
Indagar configuración (*query configuration*)	Proporciona información sobre el servicio CMC instalado

A continuación se presenta la información contenida en las páginas del manual para estas funciones.

6.1.1 Envío de mensajes

6.1.1.1 Enviar

NOMBRE

Enviar (*send*) – Envía un mensaje de correos.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_send(
   CMC_session_id        session,
   CMC_message           *message,
   CMC_flags             send_flags,
   CMC_ui_id             ui_id,
   CMC_extension         *send_extensions
);
```

DESCRIPCIÓN

Esta función envía un mensaje de correos.

El llamante puede proporcionar facultativamente una lista de recibientes, texto de asunto, añadiduras y/o texto de notas. Si se proporciona uno o más recibientes, la función puede enviar el mensaje.

El retorno exitoso de esta función no implica necesariamente la validación de los recibientes.

ARGUMENTOS

Sesión (identificador de sesión)

Identificador de sesión opaca que representa una sesión con el servicio de mensajería.

Los identificadores de sesión se crean por una llamada a la función de establecimiento de función (logon) y se invalidan por una llamada a la función de terminación de sesión (logoff).

Si el identificador de sesión no es válido, y si no se ha creado una sesión válida a través de la interfaz UI, se retorna el código de error CMC_E_INVALID_SESSION_ID.

Mensaje (mensaje)

Estructura de mensaje que comprende el contenido del mensaje que habrá de enviarse. Si la bandera del argumento de la extensión CMC_X_SEND_UI_REQUESTED no está soportada, tiene que haber por lo menos un recibiente de tipo TO, CC, o BCC.

Todos los demás campos son facultativos. Los campos de hora de envío y de referencia de mensaje no se tienen en cuenta.

Se aplican las siguientes condiciones a los campos de la estructura de mensaje:

Recibientes (*recipients*) – El número de recibientes por cada mensaje puede estar limitado en algunos servicios. Si se excede el límite, se retorna el código de error CMC_E_TOO_MANY_RECIPIENTS. Si no se especifica ningún recibiente, se debe asignar a *recibientes* un puntero de valor NULL.

El descriptor de recibiente puede incluir, o bien el nombre del recibiente, una dirección, o un par de nombre/dirección. Si sólo se especifica un nombre, el nombre se resuelve en una dirección utilizando reglas de resolución de nombre definidas por la implementación. Si sólo se especifica una dirección, esta dirección se utiliza para la entrega y para el nombre visualizable del recibiente. Si se especifica una dirección y un nombre, no debe efectuarse la resolución de nombre. Si una implementación no puede soportar la presencia de ambos, es decir, de un nombre y una dirección, no se tiene en cuenta el nombre. La dirección se expresa en un formato definido por la implementación y se supone que se ha obtenido de la implementación mediante algún otro medio. No se requiere que un recibiente de tipo originador envíe; si está presente, su acción la define la implementación CMC.

Añadiduras (*attachments*) – El número de añadiduras para cada mensaje puede estar limitado en algunos servicios. Si se excede el límite, se retorna el código de error CMC_E_TOO_MANY_FILES. Un puntero de valor NULL indica que no hay añadiduras. Los ficheros de añadiduras se leen antes de que retorne la función **cmc_send()** de modo que los ficheros puedan ser libremente modificados o suprimidos sin afectar al mensaje.

Asunto (*subject*) – Un valor de puntero NULL indica que no hay texto de asunto. Algunas implementaciones pueden truncar las líneas de asunto que sean demasiado largos o contienen caracteres de retorno del carro/cambio de renglón/nueva página.

Texto de nota (*note text*) – Un valor de puntero NULL indica que no hay texto. Las implementaciones pueden imponer límites al tamaño del texto. Si el texto de nota excede el límite del servicio, podría ocurrir que el texto del cuerpo principal perdiera su categoría y se convirtiera en una añadidura, o que se generara el código de error CMC_E_TEXT_TOO_LARGE.

Tipo de mensaje (*message type*) – Puntero a una cadena que indica el tipo de mensaje. El tipo especifica el tipo de mensaje que se está enviando (para más detalles, véase la descripción de la estructura de datos de mensaje). Para especificar un mensaje interpersonal se utiliza la cadena "CMC: IPM". Si un puntero tiene el valor NULL o apunta a una cadena vacía, se supone el valor "CMC: IPM".

Banderas (*flags*) – Se puede utilizar la siguiente bandera cuando se envía un mensaje

CMC_MSG_TEXT_NOTE_AS_FILE

Se hará caso omiso de todas las demás banderas. Para más información sobre estas banderas véase la descripción de la estructura de mensaje.

Banderas de Enviar (banderas)

Máscara de bits de banderas. Las banderas no especificadas deben siempre pasarse como cero. Las banderas no documentadas están reservadas.

CMC_LOGON_UI_ALLOWED

CMC_SEND_UI_REQUESTED

CMC_ERROR_UI_ALLOWED

CMC_COUNTED_STRING_TYPE

CMC_LOGON_UI_ALLOWED – Se fija si la función debe presentar un cuadro de diálogo para invitar al establecimiento de la sesión, si es necesario. Si no se fija, la función no visualizará un cuadro de diálogo y retornará el error CMC_E_INVALID_SESSION_ID si el usuario no ha establecido la sesión.

CMC_SEND_UI_REQUESTED – Se fija si la función debe visualizar un cuadro de diálogo para invitar a recibientes, campos del mensaje, y otras opciones de envío. Si no está fijada, la función no visualizará un cuadro de diálogo, pero al menos tiene que especificarse un recibiente.

CMC_ERROR_UI_ALLOWED – Se fija si la función puede visualizar un cuadro de diálogo cuando encuentra errores recuperables. Si no está fijada, la función no puede visualizar un cuadro de diálogo y retornará simplemente un código de error.

CMC_COUNTED_STRING_TYPE – Se fija si la cadena utilizada en el mensaje es del tipo de cadena contada. Si no se fija, se supone que las cadenas están terminadas por el carácter nulo. Si el parámetro sesión es válido, no se tiene en cuenta esta bandera.

Identificador de UI (identificador de UI)

Asa de la interfaz de usuario (por ejemplo una ventana de diálogo) que se utiliza para resolver cualquier cuestión que se plantee cuando el servicio ejecuta la función, para invitar al usuario a que aporte información adicional, o para verificar o acusar recibo de información que se ha proporcionado.

No se tiene en cuenta si la interfaz UI no está soportada por la implementación CMC.

Extensiones de Enviar (extensiones)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Enviar (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_ATTACHMENT_NOT_FOUND
CMC_E_ATTACHMENT_OPEN_FAILURE
CMC_E_ATTACHMENT_READ_FAILURE
CMC_E_ATTACHMENT_WRITE_FAILURE
CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_FLAG
CMC_E_INVALID_MESSAGE_PARAMETER
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_SESSION_ID
CMC_E_INVALID_UI_ID
CMC_E_LOGON_FAILURE
CMC_E_RECIPIENT_NOT_FOUND
CMC_E_TEXT_TOO_LARGE
CMC_E_TOO_MANY_FILES
CMC_E_TOO_MANY_RECIPIENTS
CMC_E_UNSUPPORTED_DATA_EXT
CMC_E_UNSUPPORTED_FLAG
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_USER_CANCEL
CMC_E_USER_NOT_LOGGED_ON

6.1.1.2 Enviar documentos

NOMBRE

Enviar documentos (*send documents*) – Función basada en cadena que se utiliza para enviar correspondencia.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_send_documents(
    CMC_string          recipient_addresses,
    CMC_string          subject,
    CMC_string          text_note,
    CMC_flags           send_doc_flags,
    CMC_string          file_paths,
    CMC_string          attach_titles,
    CMC_string          delimiter,
    CMC_ui_id           ui_id
);
```

DESCRIPCIÓN

Esta función envía un mensaje de correos. Tiene por finalidad principal efectuar llamadas a partir de un lenguaje "scripting" (por ejemplo, un macro de hoja de cálculo) que no puede tratar estructuras de datos.

Esta función tratará de establecer una sesión sin utilizar para ello la interfaz UI. Si esto no es posible, invitará a proporcionar información de establecimiento de sesión, para establecer una sesión. La sesión se cierra siempre, una vez concluida.

ARGUMENTOS

Direcciones de recibientes (cadena)

Puntero a una cadena que contiene las direcciones de los recibientes del mensaje. Cuando se especifican múltiples recibientes, éstos deben separarse mediante el carácter delimitador. Se supone que los recibientes son recibientes primarios, a menos que vayan precedidos de los prefijos "cc:" o "bcc:", en cuyo caso son recibientes de copia o recibientes de copia ciega, respectivamente. Se puede también utilizar el prefijo "to:", para los recibientes primarios, por razones de consistencia. Un puntero de valor NULL indica que se debe invitar a los recibientes a aportar información en un diálogo.

Asunto (cadena)

Puntero a una cadena que contiene asunto de un mensaje. Un puntero de valor NULL indica que no hay texto de asunto.

Nota textual (cadena)

Un puntero a una cadena contiene el texto de la nota que se transportará con el mensaje. Un puntero de valor NULL indica que no hay texto de nota.

Banderas de Enviar documentos (banderas)

Máscara de bits de banderas. Las banderas no especificadas deben siempre pasarse como cero. Las banderas no documentadas están reservadas.

CMC_COUNTED_STRING_TYPE
CMC_FIRST_ATTACH_AS_TEXT_NOTE

CMC_COUNTED_STRING_TYPE – Se fija si la cadena utilizada en el mensaje es del tipo de cadena contada. Si no se fija, se supone que las cadenas están terminadas por el carácter nulo.

CMC_FIRST_ATTACH_AS_TEXT_NOTE – Se fija si la primera añadidura debe enviarse como la nota textual del mensaje. Si no se fija, la nota textual va contenida en el campo de nota textual.

Trayectos de ficheros (cadena)

Puntero a una cadena que contiene los nombres de trayecto (*path names*) para los ficheros de añadiduras. Cuando se especifican múltiples nombres de trayecto, deben ir separados por el carácter delimitador.

Títulos de añadiduras (cadena)

Puntero a una cadena que contiene los títulos de las añadiduras que van a ser vistos por el recibiente. Cuando se especifican múltiples nombres, deben ir separados por el carácter delimitador.

Delimitador (cadena)

Puntero a un carácter que se utiliza para delimitar los nombres en los trayectos de ficheros, nombres de ficheros, y cadenas de direcciones de recibientes. Como carácter delimitador deberá elegirse uno que no se utilice en los nombres de ficheros del sistema operativo ni en los nombres de recibientes. Este parámetro no puede ser NULL.

Identificador de UI (identificador de UI)

Un puntero a un identificador para una interfaz de usuario (por ejemplo una ventana de diálogo) utilizada para resolver cualquier cuestión que, de no ser resuelta, podría producir un error, e interrogaciones al usuario para que suministre información adicional.

No se tiene en cuenta si la interfaz UI no está soportada por la implementación CMC.

RESULTADOS

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_ATTACHMENT_NOT_FOUND
CMC_E_ATTACHMENT_OPEN_FAILURE
CMC_E_ATTACHMENT_READ_FAILURE
CMC_E_ATTACHMENT_WRITE_FAILURE
CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_FLAG
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_UI_ID
CMC_E_LOGON_FAILURE
CMC_E_RECIPIENT_NOT_FOUND
CMC_E_TEXT_TOO_LARGE
CMC_E_TOO_MANY_FILES
CMC_E_TOO_MANY_RECIPIENTS
CMC_E_UNSUPPORTED_FLAG
CMC_E_USER_CANCEL
CMC_E_USER_NOT_LOGGED_ON

6.1.2 Recepción de mensajes

6.1.2.1 Tratar

NOMBRE

Tratar (*act on*) – Ejecuta una acción sobre un mensaje especificado.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_act_on(
    CMC_session_id              session,
    CMC_message_reference       *message_reference,
    CMC_enum                    operation,
    CMC_flags                   act_on_flags,
    CMC_ui_id                   ui_id,
    CMC_extension               *act_on_extensions
);
```

DESCRIPCIÓN

Esta función ejecuta la acción especificada sobre el mensaje indicado por la referencia de mensaje.

ARGUMENTOS

Sesión (identificador de sesión)

Identificador de sesión opaca que representa una sesión con el servicio de mensajería.

Los identificadores de sesión se crean por una llamada a la función de establecimiento de sesión (logon) y se invalidan por una llamada a la función de terminación de sesión (logoff).

Si el identificador de sesión no es válido, se retorna el error CMC_E_INVALID_SESSION_ID.

Referencia de mensaje (referencia de mensaje)

Especifica la referencia (de mensaje) del mensaje que se va a tratar.

Si la referencia de mensaje no es válida (o deja de ser válida, por ejemplo, por haberse suprimido), se retorna el error CMC_E_INVALID_MESSAGE_REFERENCE. Los punteros de valor NULL a referencias de mensajes y las referencias de mensaje de longitud cero no se consideran válidas para operaciones que requieren este parámetro.

Operación (enum)

Especifica la operación que se va a ejecutar sobre el mensaje. Son operaciones válidas:

CMC_ACT_ON_EXTENDED (= 0)
CMC_ACT_ON_DELETE

CMC_ACT_ON_EXTENDED – Consulta la lista de extensiones para determinar la acción que se va a ejecutar.

CMC_ACT_ON_DELETE – La acción solicitada es suprimir el mensaje especificado retirándolo del apartado de correos. Esta operación requiere un parámetro de referencia de mensaje válido.

Banderas de Tratar (banderas)

Máscara de bits de banderas. Las banderas no especificadas deben siempre pasarse con 0. Las banderas no documentadas están reservadas. Un valor fijado a la bandera es el siguiente:

CMC_ERROR_UI_ALLOWED

CMC_ERROR_UI_ALLOWED – Se fija si la función puede visualizar un cuadro de diálogo al encontrarse errores recuperables. Si no se fija, la función no puede visualizar un cuadro de diálogo y se limitará a retornar un código de error.

Identificador de UI (identificador de UI)

El asa de interfaz de usuario (por ejemplo, ventana de diálogo) utilizada para resolver cualquier cuestión que, de no ser resuelta, podría producir un error.

No se tiene en cuenta si la UI no está soportada por la implementación CMC.

Extensiones de Tratar (extensión)

Un puntero a una matriz de estructuras de extensión CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Tratar (extensión)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_ENUM
CMC_E_INVALID_FLAG
CMC_E_INVALID_MESSAGE_REFERENCE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_SESSION_ID
CMC_E_INVALID_UI_ID
CMC_E_MESSAGE_IN_USE
CMC_E_UNSUPPORTED_ACTION
CMC_E_UNSUPPORTED_FLAG
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.1.2.2 Listar

NOMBRE

Listar (*list*) – Lista la información de sumario sobre los mensajes que cumplen un criterio especificado.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_list(
    CMC_session_id              session,
    CMC_string                  message_type,
    CMC_flags                   list_flags,
    CMC_message_reference       *seed,
    CMC_uint32                  *count,
    CMC_ui_id                   ui_id,
    CMC_message_summary         **result,
    CMC_extension               *list_extensions
);
```

DESCRIPCIÓN

Esta función lista la información de sumario, incluida una referencia de mensaje, sobre los mensajes que cumplen el criterio especificado. La referencia o referencias de mensaje retornadas permiten procesar ulteriormente el mensaje o mensajes mediante las funciones **cmc_read()** y **cmc_act_on()**.

Son criterios facultativos:

– el mensaje es de un tipo de mensaje especificado, y

– el mensaje no está leído todavía.

La búsqueda comienza después de una referencia de mensaje de "semilla" especificada, o al comienzo del apartado de correo. Se puede especificar un número máximo de los mensajes que habrán de listarse. La función indica el número de mensajes que fueron realmente retornados.

Facultativamente, cada sumario de mensajes retornado en "resultado" sólo puede incluir la referencia de mensaje.

ARGUMENTOS

Sesión (identificador de sesión)

Identificador de sesión opaca que representa una sesión con el servicio de mensajería.

Los identificadores de sesión se crean mediante una llamada a la función de establecimiento de sesión (logon) y se invalidan mediante una llamada a la función de terminación de sesión (logoff).

Si el identificador de sesión no es válido, se retorna el error CMC_E_INVALID_SESSION_ID.

Tipo de mensaje (cadena)

Se retorna información solamente para mensajes del tipo especificado. Si no se reconoce el tipo de mensaje, se retorna el error CMC_E_UNRECOGNIZED_MESSAGE_TYPE. El formato de la cadena de tipo de mensaje se trata en 5.4.23.

Un valor NULL indica que debe retornarse información para todos los mensajes disponibles.

Banderas de listar (banderas)

Máscara de bits de banderas. Las banderas no especificadas deben siempre pasarse como cero. Las banderas no documentadas están reservadas. Son valores de banderas:

CMC_ERROR_UI_ALLOWED
CMC_LIST_UNREAD_ONLY
CMC_LIST_MSG_REFS_ONLY
CMC_LIST_COUNT_ONLY

CMC_ERROR_UI_ALLOWED – Se fija si la función puede visualizar un cuadro de diálogo al encontrar errores subsanables. Si no está fijada, la función no puede visualizar un cuadro de diálogo y retornará simplemente un código de error.

CMC_LIST_UNREAD_ONLY – Si está fijada, la lista sólo debe incluir los mensajes no leídos. Si no está fijada, la lista puede incluir los mensajes leídos y los no leídos.

CMC_LIST_MSG_REFS_ONLY – Si está fijada, sólo se llena y se retorna la referencia de mensaje en la estructura de resultado. Los valores de otros campos no están definidos y no deben tenerse en cuenta. Si no está fijada, se retorna toda la información en la estructura de resultado.

CMC_LIST_COUNT_ONLY – Si esta fijada, la función no debe retornar ninguna estructura de sumario, sino solamente la cuenta de los mensajes que satisfacen el criterio especificado. Si no está fijada, se retornarán las estructuras de sumario.

Semilla (referencia de mensaje)

Especifica la referencia (de mensaje) del mensaje tras la cual debe comenzar la búsqueda. Si la referencia de mensaje no es válida (o ha dejado de ser válida, por ejemplo por haber sido suprimida), se retorna el error CMC_E_INVALID_MESSAGE_REFERENCE.

Un puntero de semilla de referencia de mensaje con el valor NULL indica que la búsqueda debe comenzar por el primer mensaje en el apartado de correos.

Cuenta (Uint32)

Especifica el número máximo de mensajes que habrán de retornarse. Un valor de cero especifica que no hay valor máximo.

Identificador de UI (identificador de UI)

Asa de interfaz de usuario (por ejemplo, de una ventana de diálogo) utilizada para resolver cualquier cuestión que se plantee y que, de no ser resuelta, produciría un error.

No se tiene en cuenta si la interfaz UI no está soportada por la implementación CMC.

Extensiones de Listar (extensiones)

Un puntero a una matriz de estructuras de extensión CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Cuenta (Uint32)

Especifica el número de mensajes efectivamente retornados. Si ningún mensaje satisface el criterio, o si el apartado de correos está vacío, se retorna un valor de cero.

Resultado (sumario de mensaje)

El campo "resultado" es la dirección a la que se retorna una matriz de estructuras de sumario de mensaje CMC. Las estructuras que forman esta matriz las asigna el servicio y la matriz completa debe liberarse con una sola llamada a la función **cmc_free()**.

El campo de referencia de mensaje contenido en cada sumario de mensaje CMC puede utilizarse para identificar mensajes en llamadas subsiguientes a **cmc_read()** y **cmc_act_on()**.

NOTA – Puede ser necesario copiar el campo de referencia de mensaje antes de invocar la función **cmc_free()** para esta estructura.

Si se ha fijado la bandera CMC_LIST_MSG_REFS_ONLY, las estructuras de sumario de mensaje CMC retornarán solamente referencias de mensajes. Los valores de todos los demás campos no están definidos y no deben tenerse en cuenta.

Extensiones de Listar (extensión)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_FLAG
CMC_E_INVALID_MESSAGE_REFERENCE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_SESSION_ID
CMC_E_INVALID_UI_ID
CMC_E_UNRECOGNIZED_MESSAGE_TYPE
CMC_E_UNSUPPORTED_FLAG
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.1.2.3 Leer

NOMBRE

Leer (*read*) – Lee y retorna un mensaje especificado.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_read(
    CMC_session_id              session,
    CMC_message_reference       *message_reference,
    CMC_flags                   read_flags,
    CMC_message                 **message,
    CMC_ui_id                   ui_id,
    CMC_extension               *read_extensions
);
```

DESCRIPCIÓN

Esta función retorna una estructura de mensaje que contiene los datos del mensaje indicado por la referencia de mensaje especificada. Facultativamente, la estructura de mensaje retornada puede incluir solamente el mensaje y los encabezamientos de añadiduras.

Si la bandera CMC_MSG_TEXT_NOTE_AS_FILE está fijada en la estructura de mensaje retornada, el campo de nota textual está contenido en el fichero a que hace referencia la primera añadidura.

En el caso de sistemas que pueden marcar mensajes como leídos, un mensaje tendrá el estado "leído" ("READ") después de ejecutada con éxito esta función, a menos que esté fijada la bandera CMC_DO_NOT_MARK_AS_READ.

ARGUMENTOS

Sesión (identificador de sesión)

Identificador de sesión opaca que representa una sesión con el servicio de mensajería.

Los identificadores de sesión se crean por una llamada a la función de establecimiento de función (logon) y se invalidan por una llamada la función de terminación de función (logoff). Si el indicador de sesión no es válido, se retorna el error CMC_E_INVALID_SESSION_ID.

Referencia de mensaje (referencia de mensaje)

Especifica la referencia del mensaje que habrá de leerse y retornarse. Si la referencia de mensaje no es válida (o ha dejado de ser válida, por ejemplo, por haberse suprimido), se retorna el error CMC_E_INVALID_MESSAGE_REFERENCE.

Un puntero de referencia de mensaje de valor NULL indica que el primer mensaje en el apartado de correos debe ser leído y retornado.

Banderas de Leer (banderas)

Máscara de bits de bandera. Las banderas no especificadas deben pasarse siempre como cero. Las banderas no documentadas están reservadas. Son valores de banderas:

CMC_ERROR_UI_ALLOWED
CMC_MSG_AND_ATT_HDRS_ONLY
CMC_DO_NOT_MARK_AS_READ
CMC_READ_FIRST_UNREAD_MESSAGE

CMC_ERROR_UI_ALLOWED – Si está fijada, la función puede visualizar un cuadro de diálogo al encontrar errores subsanables. Si no está fijada, la función no puede visualizar un cuadro de diálogo y retornará simplemente un código de error.

CMC_MSG_AND_ATT_HDRS_ONLY – Si está fijada, los campos attachments[n].attach_filename no estarán definidos cuando retorna la función **cmc_read()**, y no deben tenerse en cuenta. Esto puede ser útil para reducir la cantidad de datos transferidos. Si no está fijada, los campos de nombres de ficheros de añadidura se retornarán normalmente.

NOTA – Si CMC_MSG_TEXT_NOTE_AS_FILE está fijada en el mensaje para indicar que la nota textual está almacenada en la primera añadidura, el campo de nombre de fichero de añadidura se retornará para esa añadidura cualquiera que sea el valor fijado a la bandera.

CMC_DO_NOT_MARK_AS_READ – Si está fijada, el estado del mensaje no se cambia a leído cuando se retorna la función. Con esto se suprime también el envío de un informe de recibo. Se puede interrogar a la implementación mediante CMC_CONFIG_SUP_NOMKMSGREAD en la función **cmc_query_config()**, para determinar si soporta esta característica.

CMC_READ_FIRST_UNREAD_MESSAGE – Sólo está disponible cuando se pasa una referencia de mensaje de valor NULL para recibir el primer mensaje en el apartado de correos. Si está fijada, debe retornarse el primer mensaje no marcado como leído. Si no está fijada, el primer mensaje en el apartado de correos debe retornarse esté o no marcado como leído.

Identificador de UI (identificador de UI)

Asa de interfaz de usuario (por ejemplo, de una ventana de diálogo) utilizada para resolver cualquier cuestión que surja cuando el servicio realiza la función.

No se tiene en cuenta si la implementación CMC no soporta la interfaz UI.

Extensiones de Leer (extensión)

Un puntero a una matriz de estructuras de extensión CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Mensaje (mensaje)

El campo "mensaje" es la dirección a la que habrá de retornarse un puntero a la estructura de mensaje CMC. Esta estructura la asigna el servicio, y debe liberarse mediante **cmc_free()**.

Los datos de añadiduras se retornarán en ficheros, y la estructura de mensaje CMC indicará los nombres de esos ficheros.

Si se ha fijado la bandera CMC_MSG_AND_ATT_HDRS_ONLY (véase "banderas"), la estructura de mensaje CMC no retornará los ficheros de añadiduras como se ha dicho anteriormente.

Extensiones de Leer (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_ATTACHMENT_OPEN_FAILURE
CMC_E_ATTACHMENT_READ_FAILURE
CMC_E_ATTACHMENT_WRITE_FAILURE
CMC_E_DISK_FULL
CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_FLAG
CMC_E_INVALID_MESSAGE_REFERENCE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_SESSION_ID
CMC_E_INVALID_UI_ID
CMC_E_TOO_MANY_FILES
CMC_E_UNABLE_TO_NOT_MARK_READ
CMC_E_UNSUPPORTED_FLAG
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.1.3 Consulta de nombres

6.1.3.1 Consulta

NOMBRE

Consultar (*look up*) – Consulta información de direccionamiento en el directorio.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_look_up(
    CMC_session_id          session,
    CMC_recipient           *recipient_in,
    CMC_flags               look_up_flags,
    CMC_ui_id               ui_id,
    CMC_uint32              *count,
    CMC_recipient           **recipient_out,
    CMC_extension           *look_up_extensions
);
```

DESCRIPCIÓN

Esta función consulta información de direccionamiento en el directorio proporcionado por el servicio de mensajería CMC. Se utiliza para obtener una dirección resolviendo un nombre adecuado.

Pueden retornarse múltiples direcciones. Se asigna y retorna una matriz de descriptores de recibientes que contiene información totalmente resuelta sobre cada entrada.

ARGUMENTOS

Sesión (identificador de sesión)

Identificador de sesión opaca que representa una sesión con el servicio de mensajería.

Los identificadores de sesión son creados por una llamada a la función de establecimiento de sesión (logon) e invalidados por una llamada a la función de terminación de sesión (logoff).

Si el indicador de sesión no es válido y no se ha creado una sesión válida a través de la interfaz UI, se retorna el error CMC_E_INVALID_SESSION_ID.

Recibiente en entrada (Recipient in) (recibiente)

Para la resolución del nombre, el campo de nombre en la estructura contiene el nombre que habrá de resolverse. El tipo de nombre se puede fijar para que proporcione información sobre la resolución de nombre deseada. Para los tipos posibles, véase la documentación sobre la estructura de recibiente.

Para visualizar los detalles del recibiente, la estructura de recibiente debe contener una inscripción que resuelva a un sólo recibiente. De no ser así de uno, se retorna el error CMC_E_AMBIGUOUS_RECIPIENT.

Para visualizar la UI con el fin de crear listas de direccionamiento, este argumento apuntará a una matriz de recibientes que será terminada por la bandera CMC_RECIP_LAST_ELEMENT. La lista de recibientes se utilizará como valor por defecto para la visualización en la UI de lista de direcciones.

Para la resolución de nombres y para la visualización de los detalles de los recibientes, se hará caso omiso de todas las estructuras de recibientes, excepto la primera.

Banderas de Consultar (banderas)

Máscara de bits de banderas. Las banderas no especificadas deben pasarse siempre como cero. Las banderas no documentadas están reservadas. Son valores de banderas:

CMC_LOGON_UI_ALLOWED

CMC_ERROR_UI_ALLOWED

CMC_COUNTED_STRING_TYPE

CMC_LOOKUP_RESOLVE_PREFIX_SEARCH

CMC_LOOKUP_RESOLVE_IDENTITY

CMC_LOOKUP_RESOLVE_UI

CMC_LOOKUP_DETAILS_UI

CMC_LOOKUP_ADDRESSING_UI

CMC_LOGON_UI_ALLOWED – Se fija si la función debe visualizar un cuadro de diálogo para invitar a establecer una sesión, si se requiere. Si no está fijada, la función no visualizará un cuadro de diálogo y retornará el error CMC_E_INVALID_SESSION_ID si el usuario no ha establecido una sesión.

CMC_ERROR_UI_ALLOWED – Se fija si la función puede visualizar un cuadro de diálogo al encontrar errores subsanables. Si no está fijada, la función no puede visualizar un cuadro de diálogo y retornará simplemente un código de error.

CMC_COUNTED_STRING_TYPE – Se fija si la cadena utilizada en los parámetros de la llamada es de tipo de cadena contada. Si no está fijada, se supone que las cadenas terminan en el carácter nulo. Si el parámetro de sesión es válido, se ignora esta bandera.

CMC_LOOKUP_RESOLVE_PREFIX_SEARCH – Si está fijada, el método de búsqueda debe ser la búsqueda por prefijo. En este método, todos los nombres que concuerdan con la cadena de prefijo, empezando por el primer carácter del nombre, se consideran como concordantes. Si no está fijada, el método de búsqueda será el de la concordancia exacta. Las implementaciones CMC deben soportar la búsqueda simple por prefijo. La aptitud para efectuar búsquedas basadas en comodines o subcadenas es facultativa.

CMC_LOOKUP_RESOLVE_IDENTITY – Si está fijada, la función retornará un registro de recibiente para la identidad del usuario en el sistema de correos. Si ésta no puede determinarse de una manera única, se efectuará una resolución del nombre ambigua. Esto permite a la aplicación averiguar la dirección del usuario actual.

CMC_LOOKUP_RESOLVE_UI – Se fija si la implementación CMC debe tratar de eliminar la ambigüedad de los nombres presentando al usuario un dialogo de resolución de nombre. Si esta bandera no está fijada, las resoluciones que no dan por resultado un nombre único retornarán el error CMC_E_AMBIGUOUS_RECIPIENT en aquellos servicios en que la resolución de nombre tiene que dar un nombre único. Los servicios que pueden retornar múltiples nombres retornarán una lista, indicada por otros parámetros de la función. El soporte de esta bandera es facultativa para las implementaciones.

CMC_LOOKUP_DETAILS_UI – Si está fijada, la función visualizará los detalles de la UI para el recibiente a que se apunta en recipient_in. Con esto sólo se tratará el primer recibiente de la lista. Si la resolución del nombre produce más de una dirección, no se lleva a cabo y se retorna el error CMC-E_AMBIGUOUS_RECIPIENT.

CMC_LOOKUP_ADDRESSING_UI – Si está fijada, la función visualizará la UI para permitir la creación de una lista de recibientes con el fin de determinar la dirección de un mensaje y hojear el directorio general. La lista de recibientes pasada a la función será la lista original de recibientes para la UI. La función retornará la lista de recibientes seleccionada por el usuario. El soporte de esta bandera es facultativo para las implementaciones.

Identificador de UI (identificador de UI)

Asa de interfaz de usuario (por ejemplo, de una ventana de diálogo) utilizada para resolver cualquier cuestión que se presente cuando el servicio realiza la función.

No se tiene en cuenta si la implementación CMC no soporta la UI.

Cuenta (Uint32)

Especifica el número máximo de nombres que habrán de retornarse. Un valor de cero especifica que no hay máximo. El valor se retornará en la ubicación señalada por este parámetro. La información de cuenta retornada debe ser un puntero válido a una ubicación.

Extensiones de Consultar (extensión)

Un puntero a una matriz de estructuras de extensión CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Recibiente en salida (Recipient Out) (recibiente)

Puntero a una matriz de una o más estructuras de recibiente asignadas por la función **cmc_look_up()**. Las estructuras pueden utilizarse entonces en llamadas a **cmc_send()**. El puntero retornado se pasa a **cmc_free()** para liberar todas las estructuras de recibientes.

Cuenta (Uint32)

Especifica el número de nombres realmente retornados. Si ningún nombre satisface el criterio de concordancia, se retorna un valor de cero, así como el error CMC_E_RECIPIENT_NOT_FOUND.

Si el número de nombres retornados es menor que el número de nombres que se sabe están disponibles, se fijará la bandera CMC_RECIP_LIST_TRUNCATED en la estructura del último recibiente, de la matriz, junto con la bandera CMC_RECIP_LAST_ELEMENT.

Extensiones de Consultar (extensión)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_AMBIGUOUS_RECIPIENT
CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_FLAG
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_SESSION_ID
CMC_E_INVALID_UI_ID
CMC_E_LOGON_FAILURE
CMC_E_NOT_SUPPORTED
CMC_E_RECIPIENT_NOT_FOUND
CMC_E_UNSUPPORTED_DATA_EXT
CMC_E_UNSUPPORTED_FLAG
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_USER_CANCEL
CMC_E_USER_NOT_LOGGED_ON

6.1.4 Administración

Las funciones de administración definidas en esta Recomendación son: liberar, terminar sesión (logoff), establecer sesión (logon) e indagar configuración.

6.1.4.1 Liberar

NOMBRE

Liberar (*free*) – Libera la memoria asignada por el sistema de mensajería.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_free(
    CMC_buffer       memory
);
```

DESCRIPCIÓN

Esta función libera la memoria asignada por el servicio de mensajería. Después de la llamada, el puntero *memory* queda invalidado y no volverá a ser referenciado. Cuando cualquier función CMC asigna y retorna una memoria tampón a la aplicación, la aplicación liberará esa memoria mediante una llamada a esta función, después de que haya terminado de utilizarla.

Cuando una función CMC retorna un puntero de base a una estructura compleja que contiene varios niveles de punteros, todo lo que hará la aplicación para liberar la estructura completa o de la matriz de estructuras es llamar a esta función con el puntero de base retornado por la función CMC. Las funciones CMC que retornan estructuras de esta forma son:

cmc_copy_object()
cmc_commit_object()
cmc_copy_object_handle()
cmc_identifier_to_name()
cmc_list()
cmc_list_objects()
cmc_list_properties()
cmc_look_up()
cmc_name_to_identifier()
cmc_open_cursor()
cmc_open_stream()
cmc_query_configuration()
cmc_read()
cmc_read_stream()
cmc_read_property_costs()
cmc_read_properties()
cmc_read_cursor()

El comportamiento de **cmc_free()** no está definido cuando se invoca esta función con un puntero a un bloque de memoria que no ha sido asignado por el servicio de mensajería, un puntero a un bloque de memoria que ya ha sido liberado, o un puntero contenido en una estructura retornada por la implementación CMC.

En algunas situaciones, las extensiones especificadas para una función pueden ser una combinación de extensiones de entrada y de salida. En este caso, las extensiones de salida deben liberarse, una a una, utilizando **cmc_free()**. Un ejemplo de esto se muestra en el anexo C, Ejemplos de programación.

ARGUMENTOS

Memoria (memoria tampón)

Un puntero a una memoria asignada por el servicio de mensajería. No se tiene en cuenta un valor de NULL.

RESULTADOS

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES más adelante.

ERRORES

CMC_E_FAILURE
CMC_E_INVALID_MEMORY

6.1.4.2 Terminar sesión (logoff)

NOMBRE

Terminar sesión (logoff) (*logoff*) – Termina la sesión con el servicio CMC.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_logoff(
    CMC_session_id          session,
    CMC_ui_id               ui_id,
    CMC_flags               logoff_flags,
    CMC_extension           *logoff_extensions
);
```

DESCRIPCIÓN

Esta función permite a la aplicación llamante terminar la sesión con el servicio CMC. Los usuarios del servicio CMC deben aplicar la función **cmc_free()** a todos los punteros asignados por el servicio durante esta sesión, antes de llamar a la función **cmc_logoff()**. Si no se hace esto se pueden producir pérdidas en la memoria o un comportamiento no definido cuando se gane ulteriormente acceso a estos punteros, después de terminada la sesión.

NOTA – Algunas implementaciones del servicio CMC pueden optar por liberar todos los punteros creados para sesión cuando se invoca la función **cmc_logoff()**. Sin embargo, el soporte de la liberación general al terminarse una sesión es facultativo para el servicio CMC.

ARGUMENTOS

Sesión (identificador de sesión)

Identificador de sesión opaca que representa una sesión con el servicio de mensajería. Queda invalidado como resultado de esta llamada.

Si el identificador de sesión no es válido, se retorna el error CMC_E_INVALID_SESSION_ID.

Identificador de UI (identificador de UI)

Un identificador para una interfaz de usuario (por ejemplo, el asa de ventana progenitora para la aplicación llamante) utilizada para resolver cualquier cuestión que de no ser resuelta podría producir un resultado de error.

No se tiene en cuenta si la UI no está soportada por la implementación CMC.

Banderas de Terminar sesión (banderas)

Máscara de bits de banderas. Las banderas no especificadas deben siempre pasarse como cero. Las banderas no documentadas están reservadas.

CMC_ERROR_UI_ALLOWED
CMC_LOGOFF_UI_ALLOWED

CMC_ERROR_UI_ALLOWED – Se fija si la función puede visualizar un cuadro de diálogo al encontrar errores subsanables. Si no se fija, la función no puede visualizar un cuadro de diálogo y se limitará a retornar un código de error.

CMC_LOGOFF_UI_ALLOWED – Se fija si la función puede visualizar una interfaz UI diferente de la empleada para errores cuando el usuario termina la sesión.

Extensiones de Terminar sesión (extensión)

Un puntero a una matriz de estructuras de extensión CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Terminar sesión (extensión)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_FLAG
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_SESSION_ID
CMC_E_INVALID_UI_ID
CMC_E_UNSUPPORTED_FLAG
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_USER_NOT_LOGGED_ON

6.1.4.3 Establecer sesión (logon)

NOMBRE

Establecer sesión (logon) (*logon*) – Establece una sesión con el servicio CMC.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_logon(
    CMC_string                  service,
    CMC_string                  user,
    CMC_string                  password,
    CMC_object_identifier       character_set,
    CMC_ui_id                   ui_id,
    CMC_uint16                  caller_cmc_version,
    CMC_flags                   logon_flags,
    CMC_session_id              *session,
    CMC_extension               *logon_extensions
);
```

DESCRIPCIÓN

Esta función permite a la aplicación llamante establecer una sesión con el servicio CMC.

La función retorna un identificador de sesión que la aplicación puede utilizar en subsiguientes llamadas CMC.

ARGUMENTOS

Servicio (cadena)

Una cadena que indica la ubicación del servicio de mensajería subyacente, por ejemplo el trayecto a una memoria de mensajes o un nombre de nodo servidor distante. Este valor puede ser NULL si el servicio de mensajería subyacente no requiere un nombre de servicio. Este argumento puede ser necesario en algunas implementaciones e ignorado en otras.

El servicio de mensajería subyacente a una implementación CMC, o a una instalación de una implementación, puede, facultativamente, soportar múltiples protocolos de mensajería simultáneamente. Si una implementación soporta múltiples protocolos, el servicio elige el protocolo particular en base a criterios como los siguientes:

– configuración del soporte de protocolo;
– disponibilidad dinámica del soporte de protocolo;

– capacidades de recibiente (si se conocen);
– análisis del formato de direcciones/notación utilizados;
– otros criterios específicos del sistema.

Estos criterios pueden aplicarse con una granularidad de mensaje por mensaje o de recibiente por recibiente.

Usuario (cadena)

Una cadena que identifica el usuario CMC, por ejemplo un nombre de usuario del servicio de mensajería. Este valor puede ser NULL si el servicio de mensajería subyacente no requiere un nombre de usuario o si se permite la UI.

Contraseña (cadena)

Una cadena que contiene la contraseña requerida para el acceso al servicio CMC. Este valor puede ser NULL si el servicio de mensajería subyacente no requiere una contraseña o si se permite la UI.

Juego de caracteres (identificador de objeto)

Un identificador de objeto que identifica el juego de caracteres de las cadenas utilizadas por el llamante CMC. Los posibles valores disponibles por el cliente son retornados por la implementación CMC como respuesta a la función **cmc_query_configuration()**. El cliente puede pasar el valor NULL, en cuyo caso el juego de caracteres que habrá de utilizarse lo determina el servicio CMC.

Identificador de UI (identificador de UI)

Un identificador para una interfaz de usuario (por ejemplo el asa de la ventana progenitora para la aplicación llamante) que se utilizará para resolver cualquier cuestión que, de no ser resuelta, podría producir un error, o para invitar a establecer una sesión si se permite y se ha pedido.

Este argumento no se tiene en cuenta si la implementación CMC no soporta la UI.

Versión de la CMC llamante (Uint16)

El número de la versión CMC de la aplicación llamante, multiplicado por 100. Por ejemplo, la versión 1.01 se representa por el número entero 101. La versión de esta Recomendación es 2.00 y se representa por el valor 200.

Banderas de Establecer sesión (banderas)

Máscara de bits de banderas. Las banderas no especificadas deben siempre pasarse como cero. Las banderas no documentadas están reservadas.

CMC_LOGON_UI_ALLOWED
CMC_ERROR_UI_ALLOWED
CMC_COUNTED_STRING_TYPE
CMC_FULL_CMC

CMC_LOGON_UI_ALLOWED – Se fija si la función debe visualizar un cuadro de diálogo para invitar al establecimiento de una sesión, si se requiere. Si no se fija, la función no visualizará un cuadro de diálogo y retornará un error si no se ha suministrado suficiente información.

CMC_ERROR_UI_ALLOWED – Se fija si la función puede visualizar un cuadro de diálogo al encontrar errores subsanables. Si no está fijada, la función no puede visualizar un cuadro de diálogo y se limitará a retornar un código de error.

CMC_COUNTED_STRING_TYPE – El llamante CMC fija esta bandera si la cadena utilizada por el llamante para interacciones CMC es del tipo de longitud primero. Si no está fijada, se supondrá que las cadenas terminan con el carácter nulo.

CMC_FULL_CMC – Se fija si la aplicación solicita la funcionalidad de la CMC completa. Si esta bandera no está fijada, la aplicación gana acceso a la CMC simple. La CMC completa sólo está disponible si el llamante especifica una versión CMC llamante superior o igual a 200.

Extensiones de Establecer sesión (extensiones)

Un puntero a una matriz de estructuras de extensión CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

Mediante extensiones, la aplicación puede averiguar qué extensiones están disponibles y qué extensiones de datos estarán activas para la sesión. La extensión que se emplea para esto es CMC_X_COM_SUPPORT_EXT. Toda implementación CMC que soporte extensiones tiene que soportar esta extensión. Para más información sobre esta extensión, véase B.2 sobre extensiones comunes.

RESULTADOS

Sesión (identificador de sesión)

Identificador de sesión opaca que representa una sesión con el servicio CMC.

Extensiones de Establecer sesión (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_COUNTED_STRING_UNSUPPORTED
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_CONFIGURATION
CMC_E_INVALID_ENUM
CMC_E_INVALID_FLAG
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_UI_ID
CMC_E_LOGON_FAILURE
CMC_E_PASSWORD_REQUIRED
CMC_E_SERVICE_UNAVAILABLE
CMC_E_UNSUPPORTED_CHARACTER_SET
CMC_E_UNSUPPORTED_FLAG
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_UNSUPPORTED_VERSION

6.1.4.4 Indagación de la configuración

NOMBRE

Indagar configuración (*query configuration*) – Obtiene información sobre la configuración CMC instalada.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_query_configuration(
    CMC_session_id        session,
    CMC_enum              item,
    CMC_buffer            reference,
    CMC_extension         *config_extensions
);
```

DESCRIPCIÓN

Esta función indaga (o averigua) la configuración de la implementación subyacente, y retorna la información solicitada sobre la misma asignando memoria si es necesario.

NOTA – La configuración no puede modificarse en la totalidad de la CMC; el formato del fichero de la configuración subyacente depende de la implementación.

ARGUMENTOS

Sesión (identificador de sesión)

Identificador de sesión opaca que representa una sesión con un servicio de mensajería.

Los identificadores de sesión se crean por una llamada a la función de establecer sesión (logon) y se invalidan por una llamada a la función de terminar sesión (logoff).

Se puede dar el valor NULL al argumento sesión para indicar que no hay sesión y que debe retornarse información independiente de la sesión. Con esto se proporcionará, por defecto, información de establecimiento de sesión.

Si este valor se fija a un identificador de sesión válido, se retornará información de configuración dependiente de la sesión.

Si el identificador de sesión no es válido, se retorna el error CMC_E_INVALID_SESSION_ID.

Item (enum)

Este argumento indica qué información de configuración debe retornarse. Son posibles valores:

CMC_CONFIG_CHARACTER_SET
CMC_CONFIG_LINE_TERM
CMC_CONFIG_DEFAULT_SERVICE
CMC_CONFIG_DEFAULT_USER
CMC_CONFIG_REQ_PASSWORD
CMC_CONFIG_REQ_SERVICE
CMC_CONFIG_REQ_USER
CMC_CONFIG_UI_AVAIL
CMC_CONFIG_SUP_NOMKMSGREAD
CMC_CONFIG_SUP_COUNTED_STR
CMC_CONFIG_VER_IMPLEM
CMC_CONFIG_VER_SPEC

CMC_CONFIG_CHARACTER_SET – El argumento de referencia debe ser un puntero a una matriz de identificadores de objetos de CMC. Aquí se retornará un puntero a la matriz de cadenas de identificadores de objetos de juegos de caracteres para la implementación. La matriz terminará con un identificador de objeto CMC de valor NULL. El primer identificador de objeto de juego de caracteres en la matriz es el juego de caracteres por defecto que se utiliza si el llamante no especifica uno explícitamente. La cláusula sobre la información específica de la plataforma B.2.4, contiene los valores de identificador de objeto definidos para los juegos de caracteres comunes. Este puntero a la matriz debe liberarse invocando la función **cmc_free()**. Este identificador de objeto lo utiliza el llamante al establecer una sesión, para ordenar a la implementación que utilice un juego de caracteres diferente del juego de caracteres por defecto.

CMC_CONFIG_LINE_TERM – El argumento de referencia debe ser un puntero a una variable CMC_enum que se fijará a un valor de CMC_LINE_TERM_CRLF si el delimitador de línea es un carácter de retorno del carro seguido por un carácter cambio de renglón, CMC_LINE_TERM_LF si el delimitador de línea es un carácter de cambio de renglón, o CMC_LINE_TERM_CR si el delimitador de línea es un carácter de retorno del carro.

CMC_CONFIG_DEFAULT_SERVICE – El argumento de referencia debe ser un puntero a una cadena CMC_String, en la que se escribirá un puntero al nombre de servicio por defecto, si está disponible, seguido por el carácter nulo. Se escribirá un valor de puntero NULL si no hay disponible ningún nombre de servicio por defecto. Este puntero debe liberarse por una llamada a la función **cmc_free()**. Esta cadena, junto con la retornada por CMC_CONFIG_DEFAULT_USER, puede utilizarse como valor por defecto cuando se pida al usuario que indique el nombre de servicio, el nombre de usuario y la contraseña. Se retornará en el juego de caracteres por defecto de la implementación.

CMC_CONFIG_DEFAULT_USER – El argumento de referencia debe ser un puntero a una cadena CMC_String, en la que se escribirá un puntero al nombre de usuario por defecto, si está disponible, seguido por un carácter nulo. Se escribirá un valor de puntero NULL si no está disponible ningún valor de usuario por defecto. Este puntero debe liberarse por una llamada a la función **cmc_free()**. Esta cadena, junto con la retornada por CMC_CONFIG_DEFAULT_SERVICE, puede utilizarse como valor por defecto cuando se pida al usuario que indique el nombre del proveedor, el nombre de usuario y la contraseña. Se retornará en el juego de caracteres por defecto de la implementación.

CMC_CONFIG_REQ_PASSWORD – El argumento de referencia deberá ser un puntero a una variable CMC_enum, que se fijará a un valor CMC_REQUIRED_NO si no se requiere contraseña para establecer la sesión, CMC_REQUIRED_OPT si la contraseña es facultativa para establecer la sesión, o CMC_REQUIRED_YES si se requiere la contraseña para establecer la sesión.

CMC_CONFIG_REQ_SERVICE – El argumento de referencia debe ser un puntero a una variable CMC_enum, que se fijará a un valor de CMC_REQUIRED_NO si no se requiere el nombre de servicio para establecer la sesión, CMC_REQUIRED_OPT si el nombre de servicio es facultativo para establecer la sesión, o CMC_REQUIRED_YES si se requiere el nombre de servicio para establecer la sesión.

CMC_CONFIG_REQ_USER – El argumento de referencia debe ser un puntero a una variable CMC_enum, que se fijará a un valor de CMC_REQUIRED_NO si no se requiere el nombre de usuario para establecer la sesión, CMC_REQUIRED_OPT si el nombre de usuario es facultativo para establecer la sesión, o CMC_REQUIRED_YES si se requiere el nombre de usuario para establecer la sesión.

CMC_CONFIG_UI_AVAIL – El argumento de referencia debe ser un puntero a una variable CMC_boolean, que se fijará al valor verdadero si hay una interfaz UI proporcionada por la implementación CMC.

CMC_CONFIG_SUP_NOMKMSGREAD – El argumento de referencia debe ser un puntero a una variable CMC_boolean, que se fijará al valor verdadero si la bandera CMC_DO_NOT_MARK_AS_READ está soportada por la función **cmc_read()**.

CMC_CONFIG_SUP_COUNTED_STR – El argumento de referencia debe ser un puntero a una variable CMC_boolean, que se fijará al valor verdadero si la bandera CMC_COUNTED_STRING_TYPE está soportada durante el establecimiento de la sesión.

CMC_CONFIG_VER_IMPLEM – El argumento de referencia debe ser un puntero a una variable CMC_uint16, que se fija al número de la versión de la implementación, multiplicado por 100. Por ejemplo, versión 1.01 retornará 101.

CMC_CONFIG_VER_SPEC – El argumento de referencia deberá ser un puntero a una variable CMC_uint16, que se fijará al número de la versión de la especificación CMC para la implementación, multiplicado por 100. Por ejemplo, versión 1.00 retornará 100.

Extensiones de Indagar configuración (extensión)

Un puntero a una matriz de estructuras de extensión CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de extensiones.

Mediante las extensiones, la aplicación puede averiguar qué extensiones están disponibles. La extensión para efectuar esto es CMC_X_COM_SUPPORT_EXT. Toda implementación CMC que soporta extensiones tiene que soportar esta extensión. Para más información sobre esta extensión, véase B.2 sobre extensiones comunes.

RESULTADOS

Referencia (memoria tampón)

Este argumento apunta a la memoria tampón que recibirá la información de configuración. El número de octetos implicados por el valor del parámetro de ítem tienen que ser propiedad del llamante y podrán modificarse. El tipo de la variable o de la memoria tampón depende del argumento del ítem.

Extensiones de Indagar configuración (extensión)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_ENUM
CMC_E_INVALID_PARAMETER
CMC_E_NOT_SUPPORTED
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2 Funciones de la CMC completa

La CMC completa es un conjunto potenciado de funciones destinadas a proporcionar capacidades basadas en mensajes a aplicaciones habilitadas para mensajería. El cuadro 15 recapitula las funciones de la interfaz CMC completa.

Cuadro 15/X.446 – Funciones de la interfaz CMC completa

Función	Descripción
Funciones de administración	
Liberar (*free*)	Para la descripción, véase la CMC simple
Terminar sesión (o logoff) (*logoff*)	Para la descripción, véase la CMC simple
Establecer sesión (o logon) (*logon*)	Para la descripción, véase la CMC simple
Funciones de vinculación	
Vincular implementación (*bind implementation*)	Crea y retorna una tabla de despacho
Desvincular implementación (*unbind implementation*)	Libera todo dato asociado con una llamada a **cmc_bind_implementation()** en una implementación CMC específica
Funciones de composición	
Copiar objeto (*copy object*)	Copia un objeto de fuente a un objeto de contenedor
Añadir propiedades (*add properties*)	Añade o modifica un conjunto de propiedades en un objeto
Comprometer objeto (*commit object*)	Compromete un objeto al dispositivo de almacenamiento persistente en un objeto de contenedor
Copiar asa de objeto (*copy object handle*)	Copia un asa de objeto
Suprimir objetos (*delete objects*)	Suprime los objetos especificados en un contenedor
Suprimir propiedades (*delete properties*)	Suprime las propiedades especificadas de un objeto
Abrir asa de objeto (*open object handle*)	Abre un asa de objeto
Restaurar objeto (*restore object*)	Restaura los datos de un objeto tomándolos de un fichero
Guardar (salvaguardar) objeto (*save object*)	Guarda (salva) los datos de un objeto en un fichero
Funciones de enumeración	
Obtener último error (*get last error*)	Retorna un mensaje de error en forma de texto localizado relativo al último error que se produjo en el objeto
Obtener asa de la raíz (*get root handle*)	Retorna un asa del contenedor que es la raíz de la jerarquía de modelo de objetos para la sesión
Listar propiedades contenidas (*list contained properties*)	Lista las propiedades en un objeto de contenedor

Cuadro 15/X.446 – Funciones de la interfaz CMC completa *(fin)*

Función	Descripción
Funciones de enumeración *(cont.)*	
Listar número de concordancias (*list number matched*)	Lista el número de objetos en un contenedor que satisfacen un criterio especificado
Listar objetos (*list objects*)	Lista los objetos en un objeto de contenedor
Listar propiedades (*list properties*)	Lista las propiedades en un objeto
Abrir cursor (*open cursor*)	Abre un cursor para un objeto de contenedor
Leer cursor (*read cursor*)	Lee la posición actual de un cursor expresada por un número quebrado
Leer propiedades (*read properties*)	Lee la información de contenido de un conjunto de propiedades
Leer costo de propiedades (*read property costs*)	Lee el costo relativo asociado con la lectura de un conjunto de propiedades
Actualizar posición de cursor (*update cursor position*)	Actualiza la posición actual de un cursor expresada por un número quebrado
Actualizar posición de cursor con semilla (*update cursor position with seed*)	Actualiza la posición actual de un cursor con respecto a un objeto en el contenedor
Funciones de notificación de eventos	
Comprobar evento (*check event*)	Comprueba si ha ocurrido un evento de mensajería
Registrar evento (*register event*)	Registra los eventos cuya comprobación interesa al llamante
Anular registro de evento (*unregister event*)	Anula el registro de eventos que ya no interesan al llamante
Invocar llamadas de retorno (*call callbacks*)	Invoca una o más funciones de llamada de retorno que están registradas si ha ocurrido el evento
Funciones de mensajería	
Crear objeto de mensaje derivado (*create derived message object*)	Crea un mensaje para reenviar un mensaje dado, o para responder a un mensaje dado
Enviar objeto de mensaje (*send message object*)	Deposita un objeto de mensaje en el agente de transferencia de mensajes para que sea enviado
Funciones de tratamiento de mensajes	
Identificador a nombre (*identifier to name*)	Convierte un identificador de propiedad en un nombre de propiedad
Nombre a identificador (*name to identifier*)	Convierte un nombre de propiedad en un identificador de propiedad
Funciones de trenes	
Exportar tren (*export stream*)	Exporta los datos de un tren a un fichero
Importar fichero a tren (*import file to stream*)	Importa los datos de un fichero a un tren
Abrir tren (*open stream*)	Abre una propiedad para las operaciones de leer o escribir tren
Leer tren (*read stream*)	Lee un tren de información de contenido
Buscar tren (*seek stream*)	Va a una determinada posición en un tren de información de contenido
Escribir tren (*write stream*)	Escribe un tren de información de contenido

A continuación se presenta la información contenida en las páginas del manual para estas funciones.

6.2.1 Funciones de vinculación

Las funciones de vinculación permiten a una implementación crear y retornar una tabla de despacho, y seguidamente liberar todo dato asociado con la función vincular implementación.

6.2.1.1 Vincular implementación

NOMBRE

Vincular implementación (*bind implementation*) – Crea y retorna una tabla de despacho.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_bind_implementation(
    CMC_guid                    implementation_name,
    CMC_dispatch_table          **dispatch_table,
    CMC_extension               *cmc_bind_implementation_extensions
);
```

DESCRIPCIÓN

Esta función crea y llena una tabla de despacho con direcciones de funciones CMC para el llamante. Esta función tiene que estar soportada por el gestor CMC y la implementación CMC. Las tareas administrativas locales se pueden efectuar con el gestor CMC y/o la implementación CMC en este momento.

ARGUMENTOS

Nombre de implementación (GUID)

Un identificador globalmente único que representa una determinada implementación CMC. Las diferentes versiones de la misma especificación CMC deben distinguirse dentro de este GUID, de modo que las diferentes versiones puedan coexistir.

Extensiones de Vincular implementación (extensión)

Un puntero a una matriz de estructuras de extensión CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Tabla de despacho (tabla de despacho)

La dirección de la tabla de despacho de la implementación CMC. Esta tabla de despacho la asigna la implementación CMC y debe liberarse con una llamada a **cmc_free()**.

Extensiones de Vincular implementación (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNRECOGNIZED_IDENTIFIER
CMC_E_BIND_FAILURE
CMC_E_ID_NOT_FOUND

6.2.1.2 Desvincular implementación

NOMBRE

Desvincular implementación (*unbind implementation*) – Libera todo dato asociado a una llamada a **cmc_bind_implementation()** en una determinada implementación CMC.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_unbind_implementation(
    CMC_guid                    implementation_name,
    CMC_extension               *cmc_unbind_implementation_extensions
);
```

DESCRIPCIÓN

Esta función libera y disocia todo dato asociado con una vinculación a una determinada implementación CMC. Las tareas administrativas locales se pueden efectuar con el gestor CMC y/o la implementación CMC en este momento.

ARGUMENTOS

Nombre de implementación (GUID)

Un identificador globalmente único que representa una determinada implementación CMC que se está desvinculando de la aplicación o del gestor CMC.

Extensiones de Desvincular implementación (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Desvincular implementación (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INSUFFICIENT_MEMORY

CMC_E_FAILURE

CMC_E_INVALID_PARAMETER

CMC_E_UNRECOGNIZED_IDENTIFIER

CMC_E_UNBIND_FAILURE

CMC_E_ID_NOT_FOUND

6.2.2 Funciones de composición

Las funciones de composición proporcionan la aptitud para crear y manipular los objetos y las propiedades de los objetos CMC.

6.2.2.1 Añadir propiedades

NOMBRE

Añadir propiedades (*add properties*) – Añade un conjunto de propiedades a un objeto.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_add_properties(
    CMC_object_handle          object,
    CMC_uint32                 number_properties,
    CMC_property               *properties,
    CMC_extension              *add_properties_extensions
);
```

DESCRIPCIÓN

Esta función añadirá un conjunto de propiedades a un objeto.

Si la propiedad ya existe en el objeto, será remplazada. Si la propiedad no existe, será añadida. No hay un orden definido por la CMC para las propiedades de un objeto. El orden en que una nueva propiedad se añadirá a un objeto es específico de la implementación (por ejemplo, puede que no sea agregada al final del objeto).

Las propiedades añadidas a un objeto no pueden ser comprometidas hasta que se haya invocado la función **cmc_commit_object()**.

ARGUMENTOS

Objeto (asa de objeto)

Un asa opaca a un objeto.

Si el asa de objeto no es válida, se retorna el error CMC_E_INVALID_OBJECT_HANDLE.

Numero de propiedades (Uint32)

El número de propiedades en el argumento **propiedades**.

Propiedades (propiedad)

Un puntero a una matriz de estructuras de propiedades que se añaden al objeto.

Extensiones de Añadir propiedades (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Añadir propiedades (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.2.2 Comprometer objeto

NOMBRE

Comprometer objeto (*commit object*) – Compromete un objeto al dispositivo de almacenamiento persistente en un objeto de contenedor.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_commit_object(
    CMC_object_handle          source_object,
    CMC_extension              *commit_object_extensions
);
```

DESCRIPCIÓN

Esta función comprometerá un objeto al dispositivo de almacenamiento persistente en un objeto de contenedor.

Si el objeto se compromete al contenedor de mensajes apartado de salida, la acción tiene por consecuencia que el objeto no pueda ser modificado. El objeto sólo podrá ser suprimido o copiado.

Cuando un mensaje se compromete al apartado de salida, se convierte en un candidato para depósito en cualquier momento. La implementación puede enviar el mensaje cuando mejor le convenga. Se puede utilizar **cmc_send_message_object()** para facilitar el envío inmediato del mensaje.

Todas las propiedades actuales del objeto de fuente serán comprometidas al dispositivo de almacenamiento persistente en un objeto de contenedor. El objeto tiene que haber sido añadido a un contenedor mediante una previa llamada a **cmc_copy_object()**.

Todo cursor para el contenedor sigue siendo válido después que el objeto ha sido comprometido al contenedor. Todo objeto comprometido al contenedor, después de abierto el cursor, no podrá ser listado en una llamada a **cmc_list_objects()** para el contenedor. Si el contenedor asociado con el objeto no soporta el compromiso de objetos, se retorna el código de error CMC_E_UNSUPPORTED_ACTION.

ARGUMENTOS

Objeto de fuente (asa de objeto)

Un asa opaca para el objeto de fuente que se comprometerá al dispositivo de almacenamiento persistente del contenedor.

Si el asa de objeto no es válida, se retorna el error CMC_E_INVALID_OBJECT_HANDLE.

Extensiones de Comprometer objeto (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Comprometer objeto (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_UNSUPPORTED_ACTION
CMC_E_INVALID_OBJECT_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_DISK_FULL
CMC_E_ACCESS_DENIED
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.2.3 Copiar objeto

NOMBRE

Copiar objeto (*copy object*) – Copia un objeto de fuente a un objeto de contenedor.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_copy_object(
   CMC_object_handle         container,
   CMC_object_handle         source_object,
   CMC_object_handle         *new_object,
   CMC_extension             *copy_object_extensions
);
```

DESCRIPCIÓN

Esta función copia un objeto de fuente al objeto de contenedor especificado. Si el objeto de fuente es un objeto de contenedor, la función copiar objeto realizará una copia profunda, en la que todas las propiedades y el objeto contenido se copian recursivamente.

Todas las propiedades actuales del objeto serán guardadas, con el objeto, en el objeto de contenedor especificado. La función añade, al objeto contenedor especificado, un nuevo objeto que contiene todas las propiedades del objeto de fuente. Se retorna un asa del nuevo objeto en el contenedor. El objeto de fuente y su contenido quedan inalterados. El nuevo objeto será comprometido al objeto de contenedor con una llamada a **cmc_commit_object()** para que sea persistente dentro del objeto de contenedor.

Los cursores de contenedor siguen siendo válidos después de que se añaden objetos al contenedor asociado con el cursor. Ningún objeto añadido después de abierto el cursor puede ser listado en una llamada a **cmc_list_object()** para el contenedor. Si el contenedor especificado sólo es accesible en lectura, se retorna el código de error CMC_E_UNSUPPORTED_ACTION.

ARGUMENTOS

Contenedor (asa de objeto)

Un asa opaca para el objeto de contenedor.

Si el asa de objeto no es válida, se retorna el error CMC_E_INVALID_OBJECT_HANDLE.

Objeto de fuente (asa de objeto)

Un asa opaca para el objeto de fuente.

Si el asa de objeto no es válida, se retorna el error CMC_E_INVALID_OBJECT_HANDLE.

Extensiones de Copiar objeto (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Nuevo objeto (asa de objeto)

Un asa opaca para el nuevo objeto.

Extensiones de Copiar objeto (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_UNSUPPORTED_ACTION
CMC_E_INVALID_SOURCE_OBJECT
CMC_E_INVALID_CONTAINER_OBJECT
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER

6.2.2.4 Copiar asa de objeto

NOMBRE

Copiar asa de objeto (*copy object handle*) – Copia un asa de un objeto.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_copy_object_handle(
    CMC_object_handle       source_handle,
    CMC_object_handle       *new_handle,
    CMC_extension           *copy_object_handle_extensions
);
```

DESCRIPCIÓN

Esta función copia un asa de un objeto. No crea una copia de un objeto. En lugar de esto, la nueva asa de objeto señalará de manera eficaz la misma información de contenido señalada por el asa del objeto de fuente. Las asas de cursor no pueden ser copiadas.

Esta función proporciona un método directo de copiar un asa de objeto a partir de una matriz de asas de objetos retornadas por otra llamada CMC. Una llamada a **cmc_free()** con el asa del objeto de fuente no liberará la información de contenido señalada por la nueva asa de objeto. La implementación sólo liberará la memoria correspondiente cuando se haya suprimido la última referencia a la misma mediante una llamada a **cmc_free()**, con la última asa de objeto señalando la información de contenido.

ARGUMENTOS

Asa de fuente (asa de objeto)

Un asa opaca para el objeto de fuente que se va a copiar.

Si el asa de objeto no es válida, se retorna el error CMC_E_INVALID_OBJECT_HANDLE.

Extensiones de Copiar asa de objeto (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Nueva asa (asa de objeto)

Un nueva asa opaca para el objeto.

Extensiones de Copiar asa de objeto (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.2.5 Suprimir objetos

NOMBRE

Suprimir objetos (*delete objects*) – Suprime los objetos especificados.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_delete_objects(
    CMC_uint32              number_objects,
    CMC_object_handle       *object,
    CMC_extension           *delete_objects_extensions
);
```

DESCRIPCIÓN

Esta función suprime los objetos especificados. La función Suprimir objetos realiza una supresión profunda, en la que se suprimen todos los objetos especificados y todos los objetos contenidos. Una vez retornada esta función, las asas de los objetos están invalidadas.

ARGUMENTOS

Número de objetos (asa de objeto)

El número de objetos en el parámetro **objetos**.

Objetos (asa de objeto)

Un puntero a una matriz de asas opacas a los objetos que habrán de suprimirse.

Si cualquiera de las asas de objetos no es válida, se retorna el error CMC_E_INVALID_OBJECT_HANDLE.

Extensiones de Suprimir objetos (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Suprimir objetos (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_ACCESS_DENIED
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.2.6 Suprimir propiedades

NOMBRE

Suprimir propiedades (*delete properties*) – Suprime el conjunto de propiedades especificadas del objeto.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_delete_properties(
    CMC_object_handle       object,
    CMC_uint32              number_properties,
    CMC_id                  *property_ids,
    CMC_extension           *delete_properties_extensions
);
```

DESCRIPCIÓN

Esta función suprime las propiedades especificadas del objeto.

ARGUMENTOS

Objeto (asa de objeto)

El asa opaca del objeto.

Si el asa del objeto no es válida, se retorna el error CMC_E_INVALID_OBJECT_HANDLE.

Número de propiedades (Uint32)

El número de propiedades en el argumento **propiedades**.

Identificadores de propiedades (identificador de propiedad)

Un puntero a una matriz de identificadores únicos de las propiedades de objeto que habrán de suprimirse.

Extensiones de Suprimir propiedades (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Suprimir propiedades (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE

CMC_E_INSUFFICIENT_MEMORY

CMC_E_FAILURE

CMC_E_INVALID_PARAMETER

CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.2.7 Abrir asa de objeto

NOMBRE

Abrir asa de objeto (*open object handle*) – Crea una nueva asa de objeto.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_open_object_handle(
    CMC_session_id          session,
    CMC_id                  object_class,
    CMC_object_handle       *new_object,
    CMC_extension           *open_object_handle_extensions
);
```

DESCRIPCIÓN

Esta función creará una nueva asa de objeto. El servicio asigna los recursos necesarios para un nuevo objeto y retorna el asa asociada con este objeto. Este objeto no existe en ningún contenedor de objetos, hasta que se añade mediante una llamada a **cmc_copy_object()**. La información de contenido para este objeto no existe, hasta que se añaden propiedades al objeto mediante una llamada a **cmc_add_properties()**.

ARGUMENTOS

Sesión (identificador de sesión)

El asa opaca que representa una sesión con el servicio de mensajería.

Si el identificador de sesión no es válido, se retorna el error CMC_E_INVALID_SESSION_ID.

Clase de objeto (identificador)

Identifica la clase del objeto.

Extensiones de Abrir asa de objeto (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Nuevo objeto (asa de objeto)

Un asa opaca para el nuevo objeto. El identificador de sesión está encapsulado en el asa de objeto. El asa de objeto es suficiente para señalar el propio objeto en una sesión dada.

Extensiones de Abrir asa de objeto (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_SESSION_ID
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_UNRECOGNIZED_IDENTIFIER

6.2.2.8 Restaurar objeto

NOMBRE

Restaurar objeto (*restore object*) – Restaura los datos de un objeto a partir del sistema de ficheros.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_restore_object(
   CMC_object_handle   container,
   CMC_string          file_specification,
   CMC_flags           restore_flags,
   CMC_object_handle   *restored_object,
   CMC_extension       *restore_object_extensions
);
```

DESCRIPCIÓN

Esta función restaura un objeto a partir de un fichero. Por ejemplo, esta función proporciona un método sencillo para insertar el contenido de un fichero como una añadidura en un mensaje. En este caso *restored object* (objeto restaurado) representa un objeto de ítem de contenido de nueva creación que más adelante se asociará con el objeto de mensaje que está en composición. Por otro lado, esta función proporciona un método para extraer un mensaje almacenado en el sistema de ficheros por una anterior llamada a la función **cmc_save_object()**. La representación, en el dispositivo de almacenamiento permanente, de los objetos almacenados en el sistema de ficheros no está definida, ya que puede variar de un sistema de mensajería a otro. Por esta razón, en general, las aplicaciones no deben confiar en su aptitud para importar objetos que han sido exportados utilizando otros sistemas de mensajería. Esta restricción, sin embargo, no se aplica en el caso de un objeto de añadidura de mensaje.

Cuando el objeto restaurado es un ítem de contenido, esta función tiene también por efecto dar valores iniciales a las siguientes propiedades:

- Nombre de fichero
- Fecha de creación
- Fecha de la última modificación.

Para dar valores a las demás propiedades hay que invocar la función **cmc_add_properties()**.

NOTA – Para finalizar el proceso de ítem de contenido, el ítem de contenido de fichero tiene todavía que ser asociado con un mensaje en composición.

ARGUMENTOS

Contenedor (asa de objeto)

Un asa del objeto de contenedor que contendrá el objeto restaurado.

Especificación de fichero (cadena)

Una especificación completa de sistema de ficheros para el fichero que contiene los datos del objeto.

Banderas de Restaurar objeto (banderas)

Máscara de bits de las banderas. Las banderas no especificadas se pasan siempre como cero. Las banderas no documentadas están reservadas.

CMC_RESTORE_OBJECT_OVERWRITE

CMC_RESTORE_OBJECT_OVERWRITE – Esta bandera se fija si la función debe aplastar un objeto existente.

Extensiones de Restaurar objeto (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Objeto restaurado (asa de objeto)

Un asa del objeto restaurado.

Extensiones de Restaurar objeto (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_UNSUPPORTED_ACTION
CMC_E_INVALID_OBJECT_HANDLE
CMC_E_INVALID_CONTAINER_OBJECT
CMC_E_ACCESS_DENIED
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_FLAG
CMC_E_INVALID_FILE_SPECIFICATION
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.2.9 Guardar (o salvaguardar) objeto

NOMBRE

Guardar (o salvaguardar) objeto (*save object*) – Guarda los datos de un objeto en el sistema de ficheros.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_save_object(
    CMC_object_handle   object,
    CMC_string   file_specification,
    CMC_flags    save_flags,
    CMC_extension*save_object_extensions
);
```

DESCRIPCIÓN

Esta función guarda los datos de un objeto en un fichero. Por ejemplo, esta función proporciona un método sencillo para datos de una añadidura de mensaje al sistema de ficheros. En este caso *object* (objeto) representa un objeto de añadidura que más adelante se asociará con un mensaje. Por otro lado, esta función proporciona un método para almacenar un mensaje en el sistema de ficheros. Los datos del mensaje pueden ser restaurados a partir del fichero mediante una ulterior llamada a la función **cmc_restore_object()**. La representación, en el soporte de registro permanente, de los objetos almacenados en el sistema de ficheros no está definida, pues puede variar de un sistema de mensajería a otro. Por esta razón, en general, las aplicaciones no deben confiar en su aptitud para importar objetos que se han exportado utilizando otros sistemas de mensajería.

ARGUMENTOS

Objeto (asa de objeto)

Un asa al objeto (por ejemplo, objeto de mensaje o de añadidura) cuyos datos serán exportados.

Especificación de fichero (cadena)

Una especificación completa de sistema de ficheros para el fichero que contendrá los datos del objeto.

Banderas de Guardar objeto (banderas)

Máscara de bits de las banderas. Las banderas no especificadas deben pasarse siempre como cero. Las banderas no documentadas están reservadas.

CMC_SAVE_OBJECT_OVERWRITE
CMC_SAVE_OBJECT_NOCREATE

CMC_SAVE_OBJECT_OVERWRITE – Esta bandera se fija si la función, cuando encuentra un fichero existente que concuerda con la especificación de fichero, debe aplastarlo.

CMC_SAVE_OBJECT_NOCREATE – Esta bandera se fija si la función, cuando no encuentra un fichero existente que concuerda con la especificación de fichero, no debe crearlo.

Extensiones de Guardar objeto (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Guardar objeto (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_ACCESS_DENIED
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_FLAG
CMC_E_INVALID_FILE_SPECIFICATION
CMC_E_DISK_FULL
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3 Funciones de enumeración

Las funciones de enumeración proporcionan la aptitud para listar, leer, y actualizar los objetos y las propiedades de los objetos CMC.

6.2.3.1 Obtener el último error

NOMBRE

Obtener último error (*get last error*) – Retorna un mensaje de error textual, localizado, relativo al último error que se produjo en el objeto.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_get_last_error(
    CMC_session_id   session,
    CMC_object_handle   object,
    CMC_string   *error_buffer,
    CMC_extension*get_last_error_extensions
);
```

DESCRIPCIÓN

La función cmc_get_last_error la utilizan las aplicaciones de clientes para extraer y presentar visualmente al usuario una cadena localizada que corresponde al último error retornado por una función invocada con respecto al objeto en cuestión. La implementación asigna un espacio para el almacenamiento de la memoria tampón retornada e incumbirá al cliente liberarlo. Si la función retorna un error (valor diferente de cero), la aplicación llamante no debe volver a invocar la función cmc_get_last_error para que le proporcione diagnósticos adicionales. Incluso si la función retorna el valor cero, todavía es posible que no haya una cadena disponible. El código de retorno tiene que ser cero para que la aplicación utilice la cadena descriptiva. Las implementaciones de cmc_get_last_error deben localizar los mensajes de error en el lenguaje del sistema, para lo cual es necesario que el usuario determine el juego de caracteres apropiado en la llamada a cmc_logon.

Si tanto el parámetro sesión como el parámetro objeto tienen el valor NULL, ello indica que la función cmc_logon ha pedido que se obtenga el último error, para lo cual la cadena de error retornada estaría escrita de acuerdo con la página de código por defecto para el sistema. Si el identificador de sesión es válido y el valor de objeto no es válido, se retorna el error CMC_E_INVALID_OBJECT_HANDLE. Si el parámetro identificador de sesión no es válido y el parámetro objeto es válido, se retorna el error CMC_E_INVALID_SESSION_ID.

ARGUMENTOS

Sesión (identificador de sesión)

Identificador de sesión que representa la sesión con el servicio CMC en la que se produjo el error.

Objeto (asa de objeto)

Un asa del objeto (por ejemplo, objeto de mensaje o de añadidura) desde el cual se retornarán datos.

Extensiones de Obtener último error (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Memoria tampón de error (cadena)

La dirección de la memoria tampón en la que la implementación almacena la cadena que describe el error. Esta memoria tampón la asigna el servicio y deberá liberarse con una llamada a **cmc_free()**.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_INVALID_SESSION_ID
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_PARAMETER
CMC_E_FAILURE
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.2 Obtener asa de la raíz

NOMBRE

Obtener asa de la raíz (*get root handle*) – Obtiene un asa del contenedor que es la raíz de la jerarquía del modelo de objetos.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_get_root_handle(
    CMC_session  session,
    CMC_object_handle   *root_object_handle,
    CMC_extensions   *get_root_handle_extensions
);
```

DESCRIPCIÓN

Esta función retorna un asa del contenedor que es la raíz de la jerarquía de modelos de objetos para la sesión. Todas las llamadas que se hagan a esta función durante la existencia de esta sesión retornarán la misma asa de objeto.

ARGUMENTOS

Sesión (identificador de sesión)

Asa de sesión opaca que representa una sesión con el servicio de mensajería.

Si el asa de sesión no es válida, se retorna el error CMC_E_INVALID_SESSION_ID.

Extensiones de Obtener asa de la raíz (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Asa de objeto de raíz (asa de objeto)

Un asa del contenedor que es la raíz de la jerarquía del modelo de objetos.

Extensiones de Obtener asa de la raíz (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_SESSION_ID
CMC_E_ACCESS_DENIED
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.3 Listar propiedades contenidas

NOMBRE

Listar propiedades contenidas (*list contained properties*) – Lista las propiedades de objetos en un contenedor.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_list_contained_properties(
    CMC_cursor_handle   cursor,
    CMC_sint32   *number_objects,
    CMC_uint32   *number_properties,
    CMC_id       *property_ids,
    CMC_property ***properties,
    CMC_extension*list_contained_properties_extensions
);
```

DESCRIPCIÓN

Esta función lista las propiedades de objetos en un contenedor. Una de las finalidades de esta función es extraer información de sumario sobre los objetos en el contenedor (por ejemplo para componer un sumario de los mensajes en el apartado de entrada).

ARGUMENTOS

Cursor (asa de cursor)

El asa opaca para el cursor del objeto de contenedor especificado.

Número de objetos (Sint32)

Un puntero al número máximo de asas de objetos que habrá de retornarse. El valor cero indica que no hay un máximo. Un valor negativo indica que las asas del número especificado de objetos que preceden a la posición actual del cursor deben retornarse en el mismo orden de clasificación especificado por el cursor. Por ejemplo, si la posición actual del cursor corresponde al octavo objeto en el contenedor, el valor –3 listará las asas de los objetos quinto, sexto y séptimo, y el cursor se reposicionará en el quinto objeto. El valor 4 lista las asas de los objetos octavo, noveno, décimo y undécimo, y el cursor se reposicionará en el duodécimo objeto.

Número de propiedades (Uint32)

Un puntero al número de propiedades en el argumento **identificadores de propiedades**.

Identificadores de propiedades (identificador de propiedad)

Un puntero a una matriz de identificadores de propiedades que corresponden a las propiedades que habrán de listarse.

Extensiones de Listar propiedades contenidas (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Número de objetos (Sint32)

El número de objetos cuyas propiedades se retornan efectivamente.

Número de propiedades (Uint32)

El número de propiedades que habrá de retornarse efectivamente para cada objeto.

Propiedades (propiedad)

La dirección de una matriz de matrices de estructuras de propiedades en un objeto de contenedor que son listadas. Cada matriz es el conjunto de propiedades asociadas con un objeto individual. El número de elementos en la matriz se da por el resultado **Número de propiedades**. El número de matrices se da por el resultado **Número de objetos**. Esta matriz de matrices la asigna el servicio y deberá liberarse por una llamada a **cmc_free()**.

Extensiones de Listar propiedades contenidas (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_CURSOR_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_PROPERTY_ID
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.4 Listar número de concordancias

NOMBRE

Listar número de elementos concordantes (*list number matched*) – Lista el número de elementos en un objeto de contenedor que satisfacen las restricciones especificadas por un cursor.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_list_number_matched(
   CMC_cursor_handle   cursor,
   CMC_uint32   *number_matches,
   CMC_extension*list_number_matched_extensions
);
```

DESCRIPCIÓN

Esta función retorna el número de elementos en un contenedor que satisfacen las restricciones especificadas por un cursor. Este valor puede utilizarse, junto con la posición actual del cursor representada por un número quebrado, para visualizar una "corredera" ("*thumb*") en una barra de desfile.

ARGUMENTOS

Cursor (asa de cursor)

El asa opaca de un cursor.

Si el asa del cursor no es válida, se retorna el error CMC_E_INVALID_CURSOR_HANDLE.

Extensiones de Listar número de concordancias (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Número de concordancias (Uint32)

El número de elementos en el contenedor que satisfacen las restricciones especificadas por el cursor. Si es cero, ningún elemento satisface las restricciones especificadas por el cursor.

Extensiones de Listar número de concordancias (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_CURSOR_HANDLE

CMC_E_INSUFFICIENT_MEMORY

CMC_E_FAILURE

CMC_E_INVALID_PARAMETER

CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.5 Listar objetos

NOMBRE

Listar objetos (*list objects*) – Lista los elementos en un objeto de contenedor.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_list_objects(
    CMC_cursor_handle   cursor,
    CMC_sint32   *number_objects,
    CMC_object_handle   **objects,
    CMC_extension*list_objects_extensions
);
```

DESCRIPCIÓN

Esta función retorna un puntero a una matriz de asas de objetos que corresponden a los elementos en un objeto de contenedor. El objeto de contenedor es señalado por un cursor que ha sido abierto por una llamada a la función **cmc_open_cursor()**. El servicio actualiza la posición del cursor, de manera que ulteriores llamadas a esta función retornarán asas de objetos que señalarán elementos adicionales del contenedor, según la posición actualizada del cursor.

ARGUMENTOS

Cursor (asa de cursor)

El asa opaca de un cursor.

Si el asa del cursor no es válida, se retorna el error CMC_E_INVALID_CURSOR_HANDLE.

Número de objetos (Sint32)

Un puntero al número máximo de asas de objetos que habrá de retornarse. El valor cero indica que no hay un máximo. Un valor negativo indica que las asas del número especificado de objetos que preceden a la posición actual del cursor deben retornarse en el mismo orden de clasificación especificado por el cursor. Por ejemplo, si la posición actual del cursor corresponde al octavo objeto en el contenedor, el valor –3 listará las asas de los objetos quinto, sexto y séptimo, y el cursor se reposicionará en el quinto objeto. El valor 4 lista las asas de los objetos octavo, noveno, décimo y undécimo, y el cursor se reposicionará en el duodécimo objeto.

Extensiones de Listar objetos (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Cursor (asa de cursor)

El asa opaca del cursor para el objeto de contenedor especificado. Su posición puede haber sido actualizada por el servicio.

Número de objetos (Sint32)

El número de asas de objetos que efectivamente se retornan. Si ningún elemento satisface las restricciones del cursor, o si el objeto de contenedor estaba vacío, se retorna un valor de cero.

Objetos (asa de objeto)

La dirección de una matriz de asas de objetos que corresponden a los elementos en el objeto contenedor. Esta matriz la asigna el servicio y deberá liberarse por una llamada a **cmc_free()**.

NOTA – Las asas de objeto individuales en esta matriz se invalidan cuando se libera la matriz. La aplicación puede retener asas a los objetos antes de aplicar la función **cmc_free()** a la matriz. La utilización de un asa que ha sido liberada provoca un comportamiento no definido.

Extensiones de Listar objetos (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_CURSOR_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.6 Listar propiedades

NOMBRE

Listar propiedades (*list properties*) – Lista las propiedades de un objeto.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_list_properties(
   CMC_object_handle   object,
   CMC_uint32   *number_properties,
   CMC_id       **property_ids,
   CMC_extension*list_properties_extensions
);
```

DESCRIPCIÓN

Esta función retorna los identificadores únicos de las propiedades de un objeto. Una llamada subsiguiente de la función **cmc_read_properties()** la información de contenido de propiedad para el objeto.

ARGUMENTOS

Objeto (asa de cursor)

El asa opaca del objeto que ha de listarse.

Si el asa del cursor no es válida, se retorna el error CMC_E_INVALID_OBJECT_HANDLE.

Número de propiedades (Uint32)

Un puntero al número máximo de identificadores de propiedades que habrán de retornarse. El valor cero indica que deberán listarse todas las propiedades.

Extensiones de Listar propiedades (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Número de propiedades (Uint32)

El número de identificadores de propiedades que efectivamente se retornan.

Identificadores de propiedades (identificador)

La dirección de una matriz de identificadores de propiedades únicos que corresponden a las propiedades del objeto. Esta matriz la asigna el servicio y deberá liberarse por una llamada a **cmc_free()**.

Extensiones de Listar propiedades (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE

CMC_E_INSUFFICIENT_MEMORY

CMC_E_FAILURE

CMC_E_INVALID_PARAMETER

CMC_E_INVALID_PROPERTY_NAME

CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.7 Abrir cursor

NOMBRE

Abrir cursor (*open cursor*) – Abre un cursor para un objeto de contenedor.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_open_cursor(
    CMC_object_handle   object,
    CMC_cursor_restriction *restriction,
    CMC_uint32   number_sort_keys,
    CMC_cursor_sort_key *sort_keys,
    CMC_cursor_handle   *cursor,
    CMC_extension*open_cursor_extensions
);
```

DESCRIPCIÓN

Esta función retorna un asa opaca de un cursor para el objeto de contenedor especificado. El cursor puede definirse para que opere sobre el contenedor con reglas de clasificación especificadas.

ARGUMENTOS

Objeto (asa de objeto)

El asa opaca de un objeto de contenedor.

Si el asa de objeto no es válida, se retorna el error CMC_E_INVALID_OBJECT_HANDLE.

Restricción (restricción de cursor)

Un puntero a una estructura de restricción de cursor que habrá de utilizarse en la enumeración de los elementos en el contenedor. Las implementaciones pueden no soportar todos los tipos de restricciones.

Número de claves de clasificación (Uint32)

El número de elementos en el argumento claves de clasificación (*sort-keys*). Si el valor cero, las reglas de clasificación para el contenedor no están definidas.

Claves de clasificación (clave de clasificación de cursor)

Un puntero a una matriz de claves de clasificación del cursor para la clasificación del contenedor. El primer elemento es la primera clave de clasificación, el segundo elemento es la segunda clave de clasificación, y así sucesivamente. Es posible que no todas las implementaciones soporten más de una clave de clasificación. Los objetos que no tienen la propiedad indicada por la clave de clasificación se colocan al final.

Extensiones de Abrir cursor (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Cursor (asa de cursor)

Un asa opaca para el cursor del objeto de contenedor especificado. Esta asa la asigna el servicio y deberá liberarse por una llamada a **cmc_free()**.

Extensiones de Abrir cursor (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_RESTRICTION
CMC_E_UNSUPPORTED_KEYS
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.8 Leer cursor

NOMBRE

Leer cursor (*read cursor*) – Lee la posición actual del cursor especificado en un objeto de contenedor, representada por un número quebrado.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_read_cursor(
    CMC_cursor_handle   cursor,
    CMC_uint32   *position_numerator,
    CMC_uint32   *position_denominator,
    CMC_extension*read_cursor_extensions
);
```

DESCRIPCIÓN

Esta función retorna la posición actual del cursor especificado, representada por un número quebrado. Los valores retornados en el numerador de la posición y en el denominador de la posición permiten determinar y dibujar una "corredera" ("*thumb*") en una barra de desfile. El valor máximo de la barra de desfile podría ser determinado por una propiedad específica de un objeto de contenedor.

ARGUMENTOS

Cursor (asa de cursor)

Un asa opaca de un cursor.

Si el asa del cursor no es válida, se retorna el código de error CMC_E_INVALID_CURSOR_HANDLE.

Extensiones de Leer cursor (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Numerador de la posición (Uint32)

Es el numerador del número quebrado que determina la posición actual del cursor. La razón del numerador de la posición al denominador de la posición da una posición fraccionaria aproximada del cursor a través de los elementos del objeto de contenedor. El espacio para el almacenamiento de este argumento lo asigna el llamante.

Denominador de la posición (Uint32)

Es el denominador del número quebrado que determina la posición actual del cursor. La razón del numerador de la posición al denominador de la posición da una posición fraccionaria aproximada del cursor a través de los elementos del objeto de contenedor. El espacio para el almacenamiento de este argumento lo asigna el llamante.

Extensiones de Leer cursor (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_CURSOR_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_PROPERTY_NAME
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.9 Leer propiedades

NOMBRE

Leer propiedades (*read properties*) – Lee la información de contenido asociada con un conjunto de propiedades de un objeto.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_read_properties(
    CMC_object_handle   object,
    CMC_uint32   *number_properties,
    CMC_id       *property_ids,
    CMC_property **properties,
    CMC_extension*read_properties_extensions
);
```

DESCRIPCIÓN

Esta función retorna la información de contenido de las propiedades especificadas en un objeto.

Si una propiedad especificada no está en el objeto, el tipo de propiedad CMC_pv_return_code se retornará en la posición de esa **propiedad**, en el argumento propiedades, con el valor de propiedad del código de retorno CMC_E_PROPERTY_ID_NOT_FOUND. El identificador de propiedad para esta propiedad no está definido por esta Recomendación.

ARGUMENTOS

Objeto (asa de objeto)

El asa opaca del objeto que ha de listarse.

Si el asa del objeto no es válida, se retorna el código de error CMC_E_INVALID_OBJECT_HANDLE.

Número de propiedades (Uint32)

Un puntero al número de identificadores de propiedades en el argumento **identificadores de propiedades**.

Identificadores de propiedades (identificador)

Un puntero a una matriz de identificadores de propiedades únicos que corresponden a las propiedades que han de leerse.

Extensiones de Leer propiedades (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Número de propiedades (Uint32)

El número de propiedades efectivamente retornado. Si ninguna de las propiedades especificadas estaba en el objeto, se retorna un valor de cero.

Propiedades (propiedad)

Un puntero a una matriz de estructuras de propiedades que contienen la información de contenido para las propiedades que se leyeron. Esta matriz la asigna el servicio y debe liberarse por una llamada a la función **cmc_free()**.

Extensiones de Leer propiedades (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE

CMC_E_INSUFFICIENT_MEMORY

CMC_E_FAILURE

CMC_E_INVALID_PARAMETER

CMC_E_INVALID_PROPERTY_NAME

CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.10 Leer costos de propiedades

NOMBRE

Leer costos de propiedades (*read property costs*) – Lee el costo relativo asociado con la lectura de propiedades individuales de un objeto.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_read_property_costs(
   CMC_object_handle   object,
   CMC_uint32   *number_properties,
   CMC_id       *property_ids,
   CMC_enum     *costs,
   CMC_extension*read_property_costs_extensions
);
```

DESCRIPCIÓN

Esta función retorna el costo relativo asociado con la lectura de propiedades individuales de un objeto.

El soporte de esta función no es obligatorio para la conformidad con esta Recomendación. Las implementaciones que no soportan esta función retornarán el error CMC_E_NOT_SUPPORTED.

La base para la determinación del costo de la lectura de la propiedad es específica de la implementación.

ARGUMENTOS

Objeto (asa de objeto)

El asa opaca del objeto que ha de listarse.

Si el asa del objeto no es válida, se retorna el código de error CMC_E_INVALID_OBJECT_HANDLE.

Número de propiedades (Uint32)

Un puntero al número de identificadores de propiedades en el argumento **identificadores de propiedades**.

Identificadores de propiedades (identificador)

Un puntero a una matriz de identificadores de propiedades únicos que corresponden a las propiedades que han de leerse.

Extensiones de Leer costos de propiedades (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Número de propiedades (Uint32)

El número de costos de propiedades efectivamente retornados. Si no se leyó ningún costo de propiedad específico, se retorna un valor de cero.

Costos (enum)

Un puntero a una matriz de costos relativos de propiedades. Los costos individuales están en correspondencia biunívoca con los nombres de propiedades especificados. Son valores de costos relativos válidos:

CMC_COST_UNDETERMINED
CMC_COST_NONE
CMC_COST_MINOR
CMC_COST_MAJOR

CMC_COST_UNDETERMINED – El costo de la lectura de la propiedad no puede determinarse.

CMC_COST_NONE – No hay costo relativo asociado con la lectura de la propiedad.

CMC_COST_MINOR – El costo asociado con la lectura de la propiedad es relativamente bajo.

CMC_COST_MAJOR – El costo asociado con la lectura de la propiedad es relativamente alto.

Esta matriz la asigna el servicio y debe liberarse por una llamada a la función **cmc_free()**.

Extensiones de Leer costos de propiedades (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_PROPERTY_NAME
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_NOT_SUPPORTED

6.2.3.11 Actualizar la posición de cursor

NOMBRE

Actualizar posición de cursor (*update cursor position*) – Actualiza la posición del cursor especificado en un contenedor de objetos, representada por un número quebrado.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_update_cursor_position(
    CMC_cursor_handle   cursor,
    CMC_uint32   position_numerator,
    CMC_uint32   position_denominator,
    CMC_extension*update_cursor_position_extensions
);
```

DESCRIPCIÓN

Esta función actualiza el cursor llevándolo a una posición especificada a través de los elementos de un objeto de contenedor. La posición se determina por la razón del numerador de la posición al denominador de la posición.

ARGUMENTOS

Cursor (asa de cursor)

El asa opaca de un cursor. Si el asa del cursor no es válida, se retorna el código de error CMC_E_INVALID_CURSOR_HANDLE.

Numerador de la posición (Uint32)

El numerador del quebrado que determina la posición del cursor. La razón del numerador de la posición al denominador de la posición da la posición fraccionaria del cursor a través de los elementos del objeto de contenedor.

Denominador de la posición (Uint32)

El denominador del quebrado que determina la posición del cursor. La razón del numerador de la posición al denominador de la posición da la posición fraccionaria del cursor a través de los elementos del objeto de contenedor.

Extensiones de Actualizar la posición de cursor (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Actualizar la posición de cursor (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_CURSOR_HANDLE

CMC_E_FAILURE

CMC_E_INVALID_PARAMETER

CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.12 Actualizar la posición de cursor con una semilla

NOMBRE

Actualizar posición de cursor con semilla (*update cursor position with seed*) – Actualiza la posición, representada por un número quebrado, del cursor especificado, con relación a un determinado objeto de semilla en el objeto de contenedor.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_update_cursor_position_with_seed(
    CMC_cursor_handle   cursor,
    CMC_object_handle   seed,
    CMC_extension*update_cursor_position_with_seed_extensions
);
```

DESCRIPCIÓN

Esta función actualiza el cursor llevándolo a una posición especificada a través de los elementos de un objeto de contenedor. La posición se determina por la posición relativa del objeto de semilla en el contenedor.

ARGUMENTOS

Cursor (asa de cursor)

El asa opaca de un cursor. Si el asa del cursor no es válida, se retorna el código de error CMC_E_INVALID_CURSOR_HANDLE.

Semilla (asa de objeto)

El asa opaca del objeto, en el contenedor, con relación al cual se debe actualizar la posición del cursor.

Extensiones de Actualizar la posición de cursor con semilla (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Actualizar la posición de cursor con semilla (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_CURSOR_HANDLE
CMC_E_INVALID_CURSOR_HANDLE
CMC_E_INVALID_OBJECT_HANDLE
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.4 Funciones de notificación de eventos

Las funciones de notificación de eventos permiten a una implementación efectuar comprobaciones para determinar la presencia de eventos, registrar y anular el registro de eventos, e invocar llamadas de retorno.

6.2.4.1 Comprobar evento

NOMBRE

Comprobar evento (*check event*) – Efectúa una comprobación para determinar la presencia de un evento del servicio de mensajería.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_check_event(
   CMC_session_id   session,
   CMC_event    event_type,
   CMC_uint32   minimum_timeout,
   CMC_buffer   check_event_data,
   CMC_buffer   *callback_data,
   CMC_extension*check_event_extensions
);
```

DESCRIPCIÓN

Esta función efectúa una comprobación para determinar la presencia de un evento asociado con el sistema de mensajería. Proporciona una alternativa a registrar llamadas de retorno ("*callbacks*") en la implementación CMC para aquellas aplicaciones que prefieren una interrogación síncrona para determinar la presencia de eventos o hacer que las implementaciones que no soportan las llamadas de retorno notifiquen la presencia de esos eventos.

Cada evento tiene asociada una bandera. Puede también tener asociados parámetros de entrada o de salida. Estas estructuras de datos de eventos se dan en el tipo de datos llamada de retorno (*callback*).

Si no se ha producido un evento y el periodo mínimo de temporización es diferente de cero, la implementación espera la aparición del evento durante el tiempo especificado, antes de retornar al programa llamante. Si ocurre el evento antes de la expiración del periodo de temporización, la función retorna inmediatamente. Si expira el periodo de temporización sin que haya ocurrido el evento, la función retorna el código CMC_E_NO_EVENT.

En determinadas circunstancias definidas por la implementación, que no se consideran errores reales, esta función puede terminar prematuramente, es decir, sin que se detecte ningún evento y antes de la expiración del periodo de temporización. En este caso la función retorna el código CMC_E_FUNCTION_INTERRUPTED.

NOTA – Pueden producirse otros errores que provoquen el retorno prematuro de esta función. En tales casos no se devuelve el código CMC_E_FUNCTION_INTERRUPTED, sino el correspondiente código de error CMC.

ARGUMENTOS

Sesión (identificador de sesión)

El asa opaca que representa una sesión con el servicio de mensajería.

Si el identificador de sesión no es válido, se retorna el código de error CMC_E_INVALID_SESSION_ID.

Tipo evento (evento)

Una máscara de bits de los eventos cuya presencia interesa al llamante comprobar. Los eventos no especificados deben pasarse siempre como cero. Los eventos no documentados están reservados. La definición de los eventos CMC se da en la descripción del tipo de datos evento.

Periodo mínimo de temporización (Uint32)

Periodo de tiempo, expresado en segundos, transcurrido el cual la función retorna, aunque el evento no haya ocurrido.

Si el valor es cero, la función se limita a verificar la presencia del evento, tras lo cual retorna inmediatamente.

Si el valor es CMC_NO_TIMEOUT, la función debe esperar la aparición del evento sin límite de tiempo.

Si se utiliza un valor diferente de CMC_NO_TIMEOUT, el periodo mínimo de tiempo que la función deberá efectivamente esperar depende de la implementación.

Datos de Comprobar evento (memoria tampón)

Un puntero a una estructura de datos de comprobación asociada a este evento. Para esta estructura específica de datos de comprobación, véase el tipo de datos llamada de retorno. La determinación de si es la implementación o la aplicación la que asigna la memoria tampón es específica del evento y se detalla en la descripción del tipo de datos.

Extensiones de Comprobar evento (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Datos de llamada de retorno (memoria tampón)

La dirección de la estructura de datos llamada de retorno asociada con este evento. Para esta llamada, la estructura se retorna directamente a la aplicación, y no se dirige a una función de llamada de retorno. Para la descripción del evento y la estructura específica de los datos de la llamada de retorno, véase el tipo de datos llamada de retorno (*callback*). La determinación de si es la implementación o la aplicación la que asigna la memoria tampón es específica del evento y se detalla en la descripción del tipo de datos.

Extensiones de Comprobar evento (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_EVENT
CMC_E_INVALID_FUNCTION_EXT
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_FUNCTION_INTERRUPTED
CMC_E_INVALID_SESSION_ID
CMC_E_NO_EVENT

6.2.4.2 Registrar evento

NOMBRE

Registrar evento (*register event*) – Registra eventos que interesan al llamante.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_register_event(
   CMC_session_id  session,
   CMC_event    event_type,
   CMC_callback callback,
   CMC_buffer   register_data,
   CMC_extension*register_event_extensions
);
```

DESCRIPCIÓN

Esta función especifica aquellos eventos, dentro de un sistema de mensajería, de cuya presencia interesa al llamante que se le avise.

Para notificar al llamante la presencia de un evento se pueden emplear dos funciones: o bien invocar llamada de retorno, o bien comprobar evento, para que esta función efectúe una interrogación síncrona sobre los eventos que el llamante ha registrado para comprobación. No se requiere que las implementaciones CMC soporten llamadas de retorno.

Un evento puede también tener asociados parámetros de entrada y de salida. Estos parámetros están contenidos en los datos de clientes. La estructura de datos de clientes para eventos se da en el tipo de datos llamada de retorno.

ARGUMENTOS

Sesión (identificador de sesión)

El asa opaca que representa una sesión con el servicio de mensajería.

Si el identificador de sesión no es válido, se retorna el código de error CMC_E_INVALID_SESSION_ID.

Tipo evento (evento)

Una máscara de bits de los eventos sobre cuya presencia interesa al llamante hacer comprobaciones. Los eventos no especificados deben pasarse siempre como cero. Los eventos no documentados están reservados. La definición de los eventos CMC se da en la descripción del tipo de datos evento.

Llamada de retorno (llamada de retorno)

El procedimiento de cliente que debe ser invocado por el servicio para tratar la actividad de llamada de retorno. Un valor NULL indica que no se puede invocar la función de llamada de retorno y que el evento deberá señalarse por la función comprobar evento. Si una implementación no soporta llamadas de retorno, se retorna el código de error CMC_E_CALLBACK_NOT_SUPPORTED.

Datos de Registrar evento (memoria tampón)

Un puntero a una estructura de datos de registrar (evento) asociada a este evento. Para la estructura específica de datos registrar, véase el tipo de datos llamada de retorno. La determinación de si es la implementación o la aplicación la que asigna la memoria tampón es específica del evento y se detalla en la descripción del evento.

Extensiones de Registrar evento (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Registrar evento (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_EVENT
CMC_E_INVALID_FUNCTION_EXT
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_CALLBACK_NOT_SUPPORTED
CMC_E_INVALID_SESSION_ID

6.2.4.3 Anular registro de evento

NOMBRE

Anular registro de evento (*unregister event*) – Anula el registro de eventos que ya no interesan al llamante.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_unregister_event(
    CMC_session_id   session,
    CMC_flags    event_type,
    CMC_callback callback,
    CMC_buffer   unregister_data,
    CMC_extension*unregister_event_extensions
);
```

DESCRIPCIÓN

Esta función especifica aquellos eventos, dentro de un sistema de mensajería, de cuya presencia ya no interesa al llamante que se le avise.

ARGUMENTOS

Sesión (identificador de sesión)

El asa opaca que representa una sesión con el servicio de mensajería.

Si el identificador de sesión no es válido, se retorna el código de error CMC_E_INVALID_SESSION_ID.

Tipo evento (banderas)

Una máscara de bits de los eventos sobre cuya presencia ya no interesa al llamante hacer comprobaciones. Los eventos no especificados deben pasarse siempre como cero. Los eventos no documentados están reservados. La definición de los eventos CMC se da en la descripción del tipo de datos evento.

Llamada de retorno (llamada de retorno)

El procedimiento de cliente que fue registrado para tratar la actividad de llamada de retorno. Un valor NULL indica que no se designó ninguna función de llamada de retorno. Si una implementación no soporta llamadas de retorno, se devuelve el código de error CMC_E_CALLBACK_NOT_SUPPORTED.

Datos de Anular registro de evento (memoria tampón)

Un puntero a una estructura de datos de anular registro de evento que puede utilizarse para pasar datos de eventos que necesitará la función de llamada de retorno con el fin de proporcionar un contexto para discontinuar el registro de evento. La estructura de datos de anular registro se indica en la descripción del tipo de datos llamada de retorno.

Extensiones de Anular registro de evento (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Anular registro de evento (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_FAILURE

CMC_E_INSUFFICIENT_MEMORY

CMC_E_INVALID_EVENT

CMC_E_INVALID_FUNCTION_EXT

CMC_E_INVALID_PARAMETER

CMC_E_UNSUPPORTED_FUNCTION_EXT

CMC_E_NOT_SUPPORTED

CMC_E_INVALID_SESSION_ID

6.2.4.4 Invocar llamadas de retorno

NOMBRE

Invocar llamadas de retorno (*call callbacks*) – Invoca una o más funciones de llamada de retorno que se registran si ha ocurrido el evento.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_call_callbacks(
    CMC_session_id   session,
    CMC_event    event_type,
    CMC_extension*call_callbacks_extensions
);
```

DESCRIPCIÓN

Esta función hace que el servicio de mensajería invoque las funciones de llamada de retorno registradas, asociadas con el evento o eventos de llamada de retorno especificados. El servicio de mensajería procesará cada evento de llamada de retorno especificado e invocará las funciones de llamada de retorno registradas si se han producido cambios que hubieran ocasionado las llamadas de retorno de ese evento. El orden en que se invocan las llamadas de retorno es específico de la implementación.

Esta función es útil en aquellos entornos en que una implementación sólo puede invocar llamadas de retorno cuando se está ejecutando código de la implementación. Es decir, esta función es útil para las implementaciones en las que las llamadas de retorno sólo pueden invocarse como un subproducto de la invocación de cualquier función CMC en esa implementación.

El soporte de esta función es facultativo para la conformidad con la especificación de la interfaz CMC. Si la función no está soportada, se retorna el código de error CMC_E_NOT_SUPPORTED.

ARGUMENTOS

Sesión (identificador de sesión)

El asa opaca que representa una sesión con el servicio de mensajería. Si se especifica un asa de sesión válida, se invocan las funciones de llamada de retorno registradas en esa sesión. Si el asa de sesión no es válida, se retorna el código de error CMC_E_INVALID_SESSION_ID.

Tipo evento (evento)

Una máscara bits de eventos. Los eventos no especificados deben pasarse siempre como cero. Los eventos no documentados están reservados. La definición de los eventos CMC se da en la descripción del tipo de datos evento.

Extensiones de Invocar llamadas de retorno (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Invocar llamadas de retorno (extensión)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_EVENT
CMC_E_INVALID_FUNCTION_EXT
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_SESSION_ID
CMC_E_NOT_SUPPORTED
CMC_E_SERVICE_UNAVAILABLE
CMC_E_UNSUPPORTED_FLAG
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.5 Funciones de mensajería

Las funciones de mensajería proporcionan la aptitud para crear mensajes derivados, enviar mensajes, y esperar nuevos mensajes.

6.2.5.1 Crear objeto de mensaje derivado

NOMBRE

Crear objeto de mensaje derivado (*create derived message object*) – Crea un objeto de mensaje que es adecuado para reenviar un mensaje o responder a un mensaje.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_create_derived_message_object(
   CMC_object_handle   original_message,
   CMC_enum     derived_action,
   CMC_boolean  inherit_contents,
   CMC_boolean  modified_message,
   CMC_object_handle   *derived_message,
   CMC_extensions   *create_derived_message_object_extensions
);
```

DESCRIPCIÓN

Esta función se utiliza para crear un mensaje que es adecuado para reenviar, o responder a, un mensaje dado. El parámetro "objeto" debe ser un objeto de mensaje con al menos un recibiente.

El mensaje derivado puede tener las propiedades del mensaje original y otras propiedades adicionales. Además, las propiedades del objeto derivado pueden no tener los mismos valores que las propiedades correspondientes del mensaje original. Por ejemplo, algunas implementaciones pueden modificar el asunto del mensaje al formar el mensaje de respuesta, cambiando, digamos, "Resultados financieros trimestrales" por "Re: Resultados financieros trimestrales". La implementación tiene que definir las reglas que se aplicarán a esta función, y determinar qué atributos serán modificados y qué atributos suplementarios serán generados en el mensaje derivado.

Para responder a este mensaje se necesita un objeto de recibiente originador.

ARGUMENTOS

Mensaje original (asa de objeto)

Un asa del objeto de mensaje que habrá de ser reenviado o respondido.

Acción derivada (enum)

Indica si el mensaje derivado va a ser reenviado o respondido. Puede tener uno de los siguientes valores:

CMC_DERIVED_ACTION_FORWARD
CMC_DERIVED_ACTION_REPLY_ORIGINATOR
CMC_DERIVED_ACTION_REPLY_ALL

CMC_DERIVED_ACTION_FORWARD – El mensaje derivado será reenviado.

CMC_DERIVED_ACTION_REPLY_ORIGINATOR – El mensaje derivado se enviará en respuesta al originador.

CMC_DERIVED_ACTION_REPLY_ALL – El mensaje derivado se enviará en respuesta a todos los recibientes del mensaje original.

Heredar contenido (booleano)

El contenido completo del mensaje original, o bien se copia y se incluye en el mensaje derivado, o no se tiene en cuenta. Si el valor de este argumento es verdadero, el nuevo objeto hereda el contenido completo.

Mensaje modificado (booleano)

Especifica si el mensaje original debe modificarse.

Extensiones de Crear objeto de mensaje derivado (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Mensaje derivado (asa de objeto)

Una nueva asa a un objeto de mensaje que puede ser reenviado o enviado como respuesta.

Extensiones de Crear objeto de mensaje derivado (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_UNSUPPORTED_ACTION
CMC_E_REQUIRED_PROPS_MISSING

6.2.5.2 Enviar objeto de mensaje

NOMBRE

Enviar objeto de mensaje (*send message object*) – Envía un objeto de mensaje desde un apartado de salida.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_send_message_object(
   CMC_object_handle   object,
   CMC_extensions   *send_message_object_extensions
);
```

DESCRIPCIÓN

Esta función se utiliza para enviar un mensaje desde el apartado de salida si el contenedor apartado de salida está soportado. La función tratará también de transferir todos los demás mensajes comprometidos en el apartado de salida. El parámetro "objeto" debe ser un objeto de mensaje con al menos un recibiente.

ARGUMENTOS

Objeto (asa de objeto)

Un asa del objeto de mensaje que habrá de ser depositado en el servicio de mensajería.

Extensiones de Enviar objeto de mensaje (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Enviar objeto (extensión)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT
CMC_E_REQUIRED_PROPS_MISSING

6.2.6 Funciones de tratamiento de nombres

Las funciones de tratamiento de nombres proporcionan la aptitud para convertir un identificador de propiedad en un nombre de propiedad y para convertir un nombre de propiedad en un identificador de propiedad.

6.2.6.1 Identificador a nombre

NOMBRE

Identificador a nombre (*identifier to name*) – Convierte un identificador en el nombre único asociado a dicho identificador.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_identifier_to_name(
    CMC_id       identifier,
    CMC_name     *name,
    CMC_extension*identifier_to_name_extensions
);
```

DESCRIPCIÓN

Esta función convierte un identificador en el nombre único correspondiente a dicho identificador. Puede utilizarse para la conversión de identificadores de clases de objetos y de identificadores de propiedades.

El nombre es un identificador público formal, definido por ISO 9070. El identificador es un identificador único, específico de la implementación. El identificador se utiliza para identificar unívocamente la propiedad o la clase de objeto en la estructura de propiedad CMC.

ARGUMENTOS

Identificador (identificador)

El identificador que va a ser convertido en un nombre.

Extensiones de Identificador a nombre (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Nombre (nombre)

El nombre de la cadena de identificador. Esta cadena la asigna el servicio y debe liberarse por una llamada a **cmc_free()**.

Extensiones de Identificador a nombre (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_PROPERTY_ID
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_PROPERTY_NAME_NOT_FOUND
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.6.2 Nombre a identificador

NOMBRE

Nombre a identificador (*name to identifier*) – Convierte un nombre único en su correspondiente identificador.

SINOPSIS

```
#include <xcmc.h>
CMC_return_code
cmc_name_to_identifier(
    CMC_name     name,
    CMC_id       *identifier,
    CMC_extension*name_to_identifier_extensions
);
```

DESCRIPCIÓN

Esta función convierte un nombre único en su correspondiente identificador. Puede utilizarse para la conversión en identificadores de clases de objetos y en identificadores de propiedades.

El nombre es un identificador público formal, definido por ISO 9070. El identificador es un identificador único, específico de la implementación. El identificador se utiliza para identificar unívocamente una propiedad o una clase de objeto.

ARGUMENTOS

Nombre (nombre)

El nombre que va a ser convertido en un identificador.

Extensiones de Nombre a identificador (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Identificador (identificador)

El identificador que corresponde al nombre.

Extensiones de Identificador a nombre (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_PROPERTY_NAME
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_PROPERTY_ID_NOT_FOUND
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.7 Funciones de trenes

Algunas propiedades de la CMC pueden definirse en términos de grandes cantidades de información de contenido. Estas propiedades necesitan un grupo de funciones que permitan el acceso a la información de contenido en forma trenes (de información) de entrada o de salida.

6.2.7.1 Exportar tren

NOMBRE

Exportar tren (*export stream*) – Exporta los datos de un tren a un sistema de ficheros.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_export_stream(
    CMC_stream_handle   stream,
    CMC_string   file_specification,
    CMC_uint32   count,
    CMC_flags    export_flags,
    CMC_extension*export_stream_extensions
);
```

DESCRIPCIÓN

Esta función exporta datos de un tren a un sistema de ficheros.

ARGUMENTOS

Tren (asa de tren)

Un asa del tren desde el cual se pasarán los datos que van a ser exportados.

Especificación de fichero (cadena)

Una especificación completa del sistema de ficheros, para el fichero que contendrá los datos del tren.

Cuenta (Uint32)

Especifica el número de octetos que van a ser exportados.

Banderas de Exportar tren (banderas)

Máscara de bits de banderas. Las banderas no especificadas se pasan siempre como cero. Las banderas no documentadas están reservadas.

CMC_EXPORT_STREAM_OVERWRITE

CMC_EXPORT_STREAM_NOCREATE

CMC_EXPORT_STREAM_APPEND

CMC_EXPORT_STREAM_OVERWRITE – Se fija si la función debe aplastar un fichero existente que concuerda con la especificación de fichero.

CMC_EXPORT_STREAM_NOCREATE – Se fija si la función no debe crear un fichero que concuerde con la especificación de fichero si el fichero no existe ya.

CMC_EXPORT_STREAM_APPEND – Se fija si la función debe agregar los datos del tren al final de un fichero existente que concuerda con la especificación de fichero.

Extensiones de Exportar tren (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Exportar tren (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_STREAM_HANDLE
CMC_E_ACCESS_DENIED
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_FLAG
CMC_E_INVALID_FILE_SPECIFICATION
CMC_E_DISK_FULL
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.7.2 Importar fichero a tren

NOMBRE

Importar fichero a tren (*import file to stream*) – Importa datos del sistema de ficheros a un tren.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_import_file_to_stream(
    CMC_stream_handle   stream,
    CMC_string   file_specification,
    CMC_uint32   file_offset,
    CMC_extension*import_file_to_stream_extensions
);
```

DESCRIPCIÓN

Esta función importa datos de un fichero a un tren.

ARGUMENTOS

Tren (asa de tren)

Un asa del tren al que se pasarán los datos que van a ser importados.

Especificación de fichero (cadena)

Una especificación completa del sistema de ficheros, para el fichero desde el que se importan los datos.

Desplazamiento de fichero (Uint32)

Especifica el desplazamiento, en octetos, con respecto al principio del fichero, de la posición a partir de la cual se comienza a leer los datos.

Extensiones de Importar fichero a tren (extensión)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Importar fichero a tren (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_STREAM_HANDLE
CMC_E_ACCESS_DENIED
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_INVALID_FLAG
CMC_E_INVALID_FILE_SPECIFICATION
CMC_E_INVALID_FILE_OFFSET
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.7.3 Abrir tren

NOMBRE

Abrir tren (*open stream*) – Abre un tren para operaciones de lectura o escritura relativas a una propiedad.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_open_stream(
    CMC_object_handle   object,
    CMC_property_id  property_id,
    CMC_enum     operation,
    CMC_stream_handle   *stream,
    CMC_extension*open_stream_extensions
);
```

DESCRIPCIÓN

Esta función abre un tren para la lectura o escritura de una gran cantidad de información de contenido relativa a una propiedad.

ARGUMENTOS

Objeto (asa de objeto)

Asa opaca de objeto. Encapsula el identificador de sesión.

Identificador de propiedad (identificador de propiedad)

Propiedad para la operación de lectura o escritura mediante el tren.

Operación (enum)

La operación para la que se utiliza el tren. Son operaciones válidas:

CMC_OPEN_READ
CMC_OPEN_WRITE

CMC_OPEN_READ – Abre el tren para operaciones de lectura.

CMC_OPEN_WRITE – Abre el tren para operaciones de escritura.

Extensiones de Abrir tren (extensiones)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Tren (asa de tren)

El asa de tren asignada para ganar acceso a la propiedad especificada. El valor retornado se pasa a **cmc_free()** para liberar el asa y toda información específica del servicio, sobre el tren, cuando éste deje de utilizarse.

Extensiones de Abrir tren (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_OBJECT_HANDLE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PROPERTY_ID
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.7.4 Leer tren

NOMBRE

Leer tren (*read stream*) – Lee un tren de información de contenido de la propiedad especificada.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_read_stream(
    CMC_stream_handle   stream,
    CMC_uint32   *count,
    CMC_buffer   content_information,
    CMC_extension*read_stream_extensions
);
```

DESCRIPCIÓN

Esta función lee la información de contenido de la propiedad especificada pasándola a una memoria tampón gestionada por el usuario.

ARGUMENTOS

Tren (asa de tren)

Asa opaca de tren. Encapsula el asa de sesión y el asa de objeto.

Cuenta (Uint32)

Especifica el número máximo de octetos que habrán de leerse. El valor cero especifica que no hay máximo.

Extensiones de Leer tren (extensiones)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Cuenta (Uint32)

Especifica el número de octetos de información de contenido efectivamente leídos. Si no se lee nada, se retorna un valor de cero.

Información de contenido (memoria tampón)

Una memoria tampón que contiene la información de contenido leída. Esta memoria tampón la asigna el servicio y deberá liberarse por una llamada a **cmc_free()**. La gestiona el usuario de la interfaz API, no el servicio.

Extensiones de Leer tren (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_STREAM_HANDLE
CMC_E_ACCESS_DENIED
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.7.5 Buscar en tren

NOMBRE

Buscar en tren (*seek stream*) – Desplaza hacia la posición especificada en la información de contenido del tren de la propiedad especificada.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_seek_stream(
   CMC_stream_handle   stream,
   CMC_enum     operation,
   CMC_uint32   *location,
   CMC_extension*seek_stream_extensions
);
```

DESCRIPCIÓN

Esta función desplaza hacia la posición especificada en un tren de una propiedad. La posición se especifica como un desplazamiento, en octetos, con respecto al principio, al fin, o a la posición actual dentro de la información de contenido.

ARGUMENTOS

Tren (asa de tren)

Asa opaca de tren. Encapsula el asa de sesión y el asa de objeto.

Operación (enum)

Indica el punto de partida de la búsqueda. Especifica que la búsqueda se efectúa desde el principio de la información de contenido, o desde el final de la información de contenido, o desde la posición actual en la información de contenido. Son operaciones válidas:

CMC_SEEK_BEGINNING
CMC_SEEK_END
CMC_SEEK_CURRENT_POSITION

CMC_SEEK_BEGINNING – Busca en la posición que corresponde al desplazamiento especificado desde el principio de la información de contenido.

CMC_SEEK_END – Busca en la posición que corresponde al desplazamiento especificado desde el final de la información de contenido.

CMC_SEEK_CURRENT_POSITION – Busca en la posición que corresponde al desplazamiento especificado desde la posición actual en la información de contenido.

Posición (Uint32)

Puntero al desplazamiento en octetos, o posición en el tren.

Extensiones de Buscar en tren (extensiones)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Posición (Uint32)

Posición que corresponde al desplazamiento en octetos que se ha especificado.

Extensiones de Buscar en tren (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_STREAM_HANDLE
CMC_E_ACCESS_DENIED
CMC_E_INSUFFICIENT_MEMORY
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.7.6 Escribir tren

NOMBRE

Escribir tren (*write stream*) – Escribe un tren de información de contenido en la propiedad especificada.

SINOPSIS

```
#include <xcmc.h>

CMC_return_code
cmc_write_stream(
    CMC_stream_handle   stream,
    CMC_uint32   count,
    CMC_buffer   content_information,
    CMC_extension*write_stream_extensions
);
```

DESCRIPCIÓN

Esta función escribe información de contenido en la propiedad especificada.

ARGUMENTOS

Tren (asa de tren)

Asa opaca de tren. Encapsula el asa de sesión y el asa de objeto.

Cuenta (Uint32)

Especifica el número de octetos de información de contenido que se escriben en la propiedad.

Información de contenido (memoria tampón)

Puntero a una memoria tampón que contiene la información de contenido que habrá de escribirse.

Extensiones de Escribir tren (extensiones)

Un puntero a una matriz de estructuras de extensiones CMC para esta función. La matriz puede contener extensiones de entrada para proporcionar información adicional a la función y extensiones de salida para recibir información de la función. Un valor de NULL indica que el llamante no está utilizando ninguna extensión. Para más información, véase la estructura de las extensiones.

RESULTADOS

Extensiones de Escribir tren (extensiones)

Si se pasaron extensiones de salida a la función en la lista de extensiones, los resultados del servicio estarán disponibles en la extensión. Para más información, véase la estructura de las extensiones.

Código de retorno (código de retorno)

Indica si la función tuvo o no éxito, y, si no lo tuvo, por qué. Puede ser éxito o uno de los valores indicados a continuación en ERRORES.

ERRORES

CMC_E_INVALID_STREAM_HANDLE
CMC_E_ACCESS_DENIED
CMC_E_INSUFFICIENT_MEMORY
CMC_E_NO_MORE_BYTES_TO_WRITE
CMC_E_FAILURE
CMC_E_INVALID_PARAMETER
CMC_E_UNSUPPORTED_FUNCTION_EXT

7 Códigos de retorno

En esta cláusula se definen los códigos de retorno para la interfaz CMC. Se especifican los códigos de retorno de la interfaz CMC simple y de la interfaz CMC completa; los códigos de retorno de la interfaz en lenguaje C se especifican en el anexo A, "Sumario de declaraciones en el lenguaje de programación C". Los cuadros 16-21 recapitulan los códigos de retorno genéricos y las funciones a las que pertenecen los códigos de retorno. A continuación de los cuadros se define cada código de retorno.

La implementación en lenguaje C debe retornar, si es posible, solamente los valores que pertenecen a una función específica. Si es necesario, la implementación puede retornar otros códigos de error de los que figuran en la lista, aunque no estén asignados específicamente a una función. Se desaconseja retornar códigos de error que no estén indicados en la recapitulación que sigue.

CUADRO 16/X.446 – CÓDIGOS DE RETORNO PARA LA INTERFAZ CMC SIMPLE

Código de retorno	Tra-tar	Libe-rar	Lis-tar	Termi-nar sesión	Esta-blecer sesión	Inda-gar	Leer	Con-sultar	En-viar	Enviar Docum.
CMC_E_ACCESS_DENIED	–	–	–	–	–	–	–	–	–	–
CMC_E_AMBIGUOUS_RECIPIENT	–	–	–	–	–	–	–	X	–	–
CMC_E_ATTACHMENT_NOT_FOUND	–	–	–	–	–	–	–	–	X	X
CMC_E_ATTACHMENT_OPEN_FAILURE	–	–	–	–	–	–	X	–	X	X
CMC_E_ATTACHMENT_READ_FAILURE	–	–	–	–	–	–	X	–	X	X
CMC_E_ATTACHMENT_WRITE_FAILURE	–	–	–	–	–	–	X	–	X	X
CMC_E_BIND_FAILURE	–	–	–	–	–	–	–	–	–	–
CMC_E_CALLBACK_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–	–
CMC_E_COUNTED_STRING_UNSUPPORTED	–	–	–	–	X	–	–	–	–	–
CMC_E_DISK_FULL	–	–	–	–	–	–	X	–	–	–
CMC_E_FAILURE	X	X	X	X	X	X	X	X	X	X
CMC_E_FUNCTION_INTERRUPTED	–	–	–	–	–	–	–	–	–	–
CMC_E_ID_NOT_FOUND	–	–	–	–	–	–	–	–	–	–
CMC_E_INSUFFICIENT_MEMORY	X	–	X	X	X	X	X	X	X	X
CMC_E_INVALID_CONFIGURATION	–	–	–	–	X	–	–	–	–	–
CMC_E_INVALID_CONTAINER_OBJECT	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_CURSOR_HANDLE	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_ENUM	X	–	–	–	X	X	–	–	–	–
CMC_E_INVALID_EVENT	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_FILE_OFFSET	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_FILE_SPECIFICATION	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_FLAG	X	–	X	X	X	–	X	X	X	–
CMC_E_INVALID_FUNCTION_EXT	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_MEMORY	–	X	–	–	–	–	–	–	–	–
CMC_E_INVALID_MESSAGE_PARAMETER	–	–	–	–	–	–	–	–	X	–
CMC_E_INVALID_MESSAGE_REFERENCE	X	–	X	–	–	–	X	–	–	–
CMC_E_INVALID_OBJECT_HANDLE	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_PARAMETER	X	X	X	X	X	X	X	X	X	X
CMC_E_INVALID_PROPERTY_ID	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_PROPERTY_NAME	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_RESTRICTION	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_SESSION_ID	X	–	X	X	–	–	X	–	–	–
CMC_E_INVALID_SOURCE_OBJECT	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_STREAM_HANDLE	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_UI_ID	X	–	X	X	X	–	X	X	X	X

CUADRO 16/X.446 – CÓDIGOS DE RETORNO PARA LA INTERFAZ CMC SIMPLE *(FIN)*

Código de retorno	Tratar	Liberar	Listar	Terminar sesión	Establecer sesión	Indagar	Leer	Consultar	Enviar	Enviar Docum.
CMC_E_INVALID_VALUE	–	–	–	–	–	–	–	–	–	–
CMC_E_LOGON_FAILURE	–	–	–	–	X	–	–	X	X	X
CMC_E_MESSAGE_IN_USE	X	–	–	–	–	–	–	–	–	–
CMC_E_NAME_NOT_FOUND	–	–	–	–	–	–	–	–	–	–
CMC_E_NO_EVENT	–	–	–	–	–	–	–	–	–	–
CMC_E_NO_MORE_BYTES_TO_WRITE	–	–	–	–	–	–	–	–	–	–
CMC_E_NOT_SUPPORTED	–	–	–	–	–	X	–	X	–	–
CMC_E_PASSWORD_REQUIRED	–	–	–	–	X	–	–	–	–	–
CMC_E_PROPERTY_DATA_TYPE_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_ID_NOT_FOUND	–	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_NAME_NOT_FOUND	–	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_PROBLEMS	–	–	–	–	–	–	–	–	–	–
CMC_E_RECIPIENT_NOT_FOUND	–	–	–	–	–	–	–	X	X	X
CMC_E_REQUIRED_PROPS_MISSING	–	–	–	–	–	–	–	–	–	–
CMC_E_RESTRICTION_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–	–
CMC_E_SERVICE_UNAVAILABLE	–	–	–	–	X	–	–	–	–	–
CMC_E_TEXT_TOO_LARGE	–	–	–	–	–	–	–	–	X	X
CMC_E_TOO_MANY_CONTENT_ITEMS	–	–	–	–	–	–	–	–	–	–
CMC_E_TOO_MANY_FILES	–	–	–	–	–	–	X	–	X	X
CMC_E_TOO_MANY_RECIPIENTS	–	–	–	–	–	–	–	–	X	X
CMC_E_UNABLE_TO_NOT_MARK_READ	–	–	–	–	–	–	X	–	–	–
CMC_E_UNBIND_FAILURE	–	–	–	–	–	–	–	–	–	–
CMC_E_UNRECOGNIZED_IDENTIFIER	–	–	–	–	–	–	–	–	–	–
CMC_E_UNRECOGNIZED_MESSAGE_TYPE	–	–	X	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_ACTION	X	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_CHARACTER_SET	–	–	–	–	X	–	–	–	–	–
CMC_E_UNSUPPORTED_DATA_EXT	–	–	–	–	–	–	–	X	X	–
CMC_E_UNSUPPORTED_FLAG	X	–	X	X	X	–	X	X	X	–
CMC_E_UNSUPPORTED_FUNCTION_EXT	X	–	X	X	X	X	X	X	X	–
CMC_E_UNSUPPORTED_KEYS	–	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_VALUE	–	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_VERSION	–	–	–	–	X	–	–	–	–	–
CMC_E_USER_CANCEL	–	–	–	–	–	–	–	X	X	X
CMC_E_USER_NOT_LOGGED_ON	–	–	–	X	–	–	–	X	X	X

CUADRO 17/X.446 – CÓDIGOS DE RETORNO DE LAS FUNCIONES DE ADMINISTRACIÓN Y DE VINCULACIÓN PARA LA INTERFAZ CMC COMPLETA

Código de retorno	Liberar	Terminar sesión	Establecer sesión		Vincular	Des-vincular
CMC_E_ACCESS_DENIED	–	–	–		–	–
CMC_E_AMBIGUOUS_RECIPIENT	–	–	–		–	–
CMC_E_ATTACHMENT_NOT_FOUND	–	–	–		–	–
CMC_E_ATTACHMENT_OPEN_FAILURE	–	–	–		–	–
CMC_E_ATTACHMENT_READ_FAILURE	–	–	–		–	–
CMC_E_ATTACHMENT_WRITE_FAILURE	–	–	–		–	–
CMC_E_BIND_FAILURE	–	–	–		X	–
CMC_E_CALLBACK_NOT_SUPPORTED	–	–	–		–	–
CMC_E_COUNTED_STRING_UNSUPPORTED	–	–	X		–	–
CMC_E_DISK_FULL	–	–	–		–	–
CMC_E_FAILURE	X	X	X		X	X
CMC_E_FUNCTION_INTERRUPTED	–	–	–		–	–
CMC_E_ID_NOT_FOUND	–	–	–		X	X
CMC_E_INSUFFICIENT_MEMORY	–	X	X		X	X
CMC_E_INVALID_CONFIGURATION	–	–	X		–	–
CMC_E_INVALID_CONTAINER_OBJECT	–	–	–		–	–
CMC_E_INVALID_CURSOR_HANDLE	–	–	–		–	–
CMC_E_INVALID_ENUM	–	–	X		–	–
CMC_E_INVALID_EVENT	–	–	–		–	–
CMC_E_INVALID_FILE_OFFSET	–	–	–		–	–
CMC_E_INVALID_FILE_SPECIFICATION	–	–	–		–	–
CMC_E_INVALID_FLAG	–	X	X		–	–
CMC_E_INVALID_FUNCTION_EXT	–	–	–		–	–
CMC_E_INVALID_MEMORY	X	–	–		–	–
CMC_E_INVALID_MESSAGE_PARAMETER	–	–	–		–	–
CMC_E_INVALID_MESSAGE_REFERENCE	–	–	–		–	–
CMC_E_INVALID_OBJECT_HANDLE	–	–	–		–	–
CMC_E_INVALID_PARAMETER	X	X	X		X	X
CMC_E_INVALID_PROPERTY_ID	–	–	–		–	–
CMC_E_INVALID_PROPERTY_NAME	–	–	–		–	–
CMC_E_INVALID_RESTRICTION	–	–	–		–	–
CMC_E_INVALID_SESSION_ID	–	X	–		–	–
CMC_E_INVALID_SOURCE_OBJECT	–	–	–		–	–
CMC_E_INVALID_STREAM_HANDLE	–	–	–		–	–
CMC_E_INVALID_UI_ID	–	X	X		–	–

CUADRO 17/X.446 – CÓDIGOS DE RETORNO DE LAS FUNCIONES DE ADMINISTRACIÓN Y DE VINCULACIÓN PARA LA INTERFAZ CMC COMPLETA *(FIN)*

Código de retorno	Liberar	Terminar sesión	Establecer sesión		Vincular	Des-vincular
CMC_E_INVALID_VALUE	–	–	–		–	–
CMC_E_LOGON_FAILURE	–	–	X		–	–
CMC_E_MESSAGE_IN_USE	–	–	–		–	–
CMC_E_NAME_NOT_FOUND	–	–	–		–	–
CMC_E_NO_EVENT	–	–	–		–	–
CMC_E_NO_MORE_BYTES_TO_WRITE	–	–	–		–	–
CMC_E_NOT_SUPPORTED	–	–	–		–	–
CMC_E_PASSWORD_REQUIRED	–	–	X		–	–
CMC_E_PROPERTY_DATA_TYPE_NOT_SUPPORTED	–	–	–		–	–
CMC_E_PROPERTY_ID_NOT_FOUND	–	–	–		–	–
CMC_E_PROPERTY_NAME_NOT_FOUND	–	–	–		–	–
CMC_E_PROPERTY_PROBLEMS	–	–	–		–	–
CMC_E_RECIPIENT_NOT_FOUND	–	–	–		–	–
CMC_E_REQUIRED_PROPS_MISSING	–	–	–		–	–
CMC_E_RESTRICTION_NOT_SUPPORTED	–	–	–		–	–
CMC_E_SERVICE_UNAVAILABLE	–	–	X		–	–
CMC_E_TEXT_TOO_LARGE	–	–	–		–	–
CMC_E_TOO_MANY_CONTENT_ITEMS	–	–	–		–	–
CMC_E_TOO_MANY_FILES	–	–	–		–	–
CMC_E_TOO_MANY_RECIPIENTS	–	–	–		–	–
CMC_E_UNABLE_TO_NOT_MARK_READ	–	–	–		–	–
CMC_E_UNBIND_FAILURE	–	–	–		–	X
CMC_E_UNRECOGNIZED_IDENTIFIER	–	–	–		X	X
CMC_E_UNRECOGNIZED_MESSAGE_TYPE	–	–	–		–	–
CMC_E_UNSUPPORTED_ACTION	–	–	–		–	–
CMC_E_UNSUPPORTED_CHARACTER_SET	–	–	X		–	–
CMC_E_UNSUPPORTED_DATA_EXT	–	–	–		–	–
CMC_E_UNSUPPORTED_FLAG	–	X	X		–	–
CMC_E_UNSUPPORTED_FUNCTION_EXT	–	X	X		–	–
CMC_E_UNSUPPORTED_KEYS	–	–	–		–	–
CMC_E_UNSUPPORTED_VALUE	–	–	–		–	–
CMC_E_UNSUPPORTED_VERSION	–	–	X		–	–
CMC_E_USER_CANCEL	–	–	–		–	–
CMC_E_USER_NOT_LOGGED_ON	–	X	–		–	–

CUADRO 18/X.446 – CÓDIGOS DE RETORNO DE LAS FUNCIONES DE COMPOSICIÓN PARA LA INTERFAZ CMC COMPLETA

Código de retorno	Añadir props.	Com-prom. objeto	Copiar objeto	Copiar asa de objeto	Supri-mir objetos	Supri-mir props.	Abrir asa de objeto	Restaurar objeto	Guar-dar objeto
CMC_E_ACCESS_DENIED	–	X	–	–	X	–	–	X	X
CMC_E_AMBIGUOUS_RECIPIENT	–	–	–	–	–	–	–	–	–
CMC_E_ATTACHMENT_NOT_FOUND	–	–	–	–	–	–	–	–	–
CMC_E_ATTACHMENT_OPEN_FAILURE	–	–	–	–	–	–	–	–	–
CMC_E_ATTACHMENT_READ_FAILURE	–	–	–	–	–	–	–	–	–
CMC_E_ATTACHMENT_WRITE_FAILURE	–	–	–	–	–	–	–	–	–
CMC_E_BIND_FAILURE	–	–	–	–	–	–	–	–	–
CMC_E_CALLBACK_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–
CMC_E_COUNTED_STRING_UNSUPPORTED	–	–	–	–	–	–	–	–	–
CMC_E_DISK_FULL	–	X	–	–	–	–	–	–	X
CMC_E_FAILURE	X	X	X	X	X	X	X	X	X
CMC_E_FUNCTION_INTERRUPTED	–	–	–	–	–	–	–	–	–
CMC_E_ID_NOT_FOUND	–	–	–	–	–	–	–	–	–
CMC_E_INSUFFICIENT_MEMORY	X	X	X	X	X	X	X	X	X
CMC_E_INVALID_CONFIGURATION	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_CONTAINER_OBJECT	–	–	X	–	–	–	–	X	–
CMC_E_INVALID_CURSOR_HANDLE	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_ENUM	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_EVENT	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_FILE_OFFSET	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_FILE_SPECIFICATION	–	–	–	–	–	–	–	X	X
CMC_E_INVALID_FLAG	–	–	–	–	–	–	–	X	X
CMC_E_INVALID_FUNCTION_EXT	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_MEMORY	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_MESSAGE_PARAMETER	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_MESSAGE_REFERENCE	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_OBJECT_HANDLE	X	X	–	X	X	X	–	X	X
CMC_E_INVALID_PARAMETER	X	X	X	X	X	X	X	X	X
CMC_E_INVALID_PROPERTY_ID	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_PROPERTY_NAME	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_RESTRICTION	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_SESSION_ID	–	–	–	–	–	–	X	–	–
CMC_E_INVALID_SOURCE_OBJECT	–	–	X	–	–	–	–	–	–
CMC_E_INVALID_STREAM_HANDLE	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_UI_ID	–	–	–	–	–	–	–	–	–

CUADRO 18/X.446 – CÓDIGOS DE RETORNO DE LAS FUNCIONES DE COMPOSICIÓN PARA LA INTERFAZ CMC COMPLETA *(FIN)*

Código de retorno	Añadir props.	Com-prom. objeto	Copiar objeto	Copiar asa de objeto	Supri-mir objetos	Supri-mir props.	Abrir asa de objeto	Restaurar objeto	Guar-dar objeto
CMC_E_INVALID_VALUE	–	–	–	–	–	–	–	–	–
CMC_E_LOGON_FAILURE	–	–	–	–	–	–	–	–	–
CMC_E_MESSAGE_IN_USE	–	–	–	–	–	–	–	–	–
CMC_E_NAME_NOT_FOUND	–	–	–	–	–	–	–	–	–
CMC_E_NO_EVENT	–	–	–	–	–	–	–	–	–
CMC_E_NO_MORE_BYTES_TO_WRITE	–	–	–	–	–	–	–	–	–
CMC_E_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–
CMC_E_PASSWORD_REQUIRED	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_DATA_TYPE_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_ID_NOT_FOUND	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_NAME_NOT_FOUND	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_PROBLEMS	–	–	–	–	–	–	–	–	–
CMC_E_RECIPIENT_NOT_FOUND	–	–	–	–	–	–	–	–	–
CMC_E_REQUIRED_PROPS_MISSING	–	–	–	–	–	–	–	–	–
CMC_E_RESTRICTION_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–
CMC_E_SERVICE_UNAVAILABLE	–	–	–	–	–	–	–	–	–
CMC_E_TEXT_TOO_LARGE	–	–	–	–	–	–	–	–	–
CMC_E_TOO_MANY_CONTENT_ITEMS	–	–	–	–	–	–	–	–	–
CMC_E_TOO_MANY_FILES	–	–	–	–	–	–	–	–	–
CMC_E_TOO_MANY_RECIPIENTS	–	–	–	–	–	–	–	–	–
CMC_E_UNABLE_TO_NOT_MARK_READ	–	–	–	–	–	–	–	–	–
CMC_E_UNBIND_FAILURE	–	–	–	–	–	–	–	–	–
CMC_E_UNRECOGNIZED_IDENTIFIER	–	–	–	–	–	–	X	–	–
CMC_E_UNRECOGNIZED_MESSAGE_TYPE	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_ACTION	–	X	X	–	–	–	–	X	–
CMC_E_UNSUPPORTED_CHARACTER_SET	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_DATA_EXT	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_FLAG	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_FUNCTION_EXT	X	X	–	X	X	X	X	X	X
CMC_E_UNSUPPORTED_KEYS	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_VALUE	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_VERSION	–	–	–	–	–	–	–	–	–
CMC_E_USER_CANCEL	–	–	–	–	–	–	–	–	–
CMC_E_USER_NOT_LOGGED_ON	–	–	–	–	–	–	–	–	–

CUADRO 19/X.446 – CÓDIGOS DE RETORNO DE LAS FUNCIONES DE ENUMERACIÓN PARA LA INTERFAZ CMC COMPLETA

Código de retorno	Obte-ner último error	Obte-ner asa de la raíz	Listar pros. Contds.	Listar número de concords.	Listar objetos	Listar props.	Abrir cursor	Leer cursor	Leer pros.	Leer costos de props.	Actua lizar posi-ción cursor	Act. Pos. cursor con semilla
CMC_E_ACCESS_DENIED	–	X	–	–	–	–	–	–	–	–	–	–
CMC_E_AMBIGUOUS_RECIPIENT	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_ATTACHMENT_NOT_FOUND	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_ATTACHMENT_OPEN_FAILURE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_ATTACHMENT_READ_FAILURE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_ATTACHMENT_WRITE_FAILURE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_BIND_FAILURE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_CALLBACK_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_COUNTED_STRING_ UNSUPPORTED	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_DISK_FULL	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_FAILURE	X	X	X	X	X	X	X	X	X	X	X	X
CMC_E_FUNCTION_INTERRUPTED	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_ID_NOT_FOUND	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INSUFFICIENT_MEMORY	X	X	X	X	X	X	X	X	X	X	–	–
CMC_E_INVALID_CONFIGURATION	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_CONTAINER_OBJECT	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_CURSOR_HANDLE	–	–	X	X	X	–	–	–	–	–	X	X
CMC_E_INVALID_ENUM	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_EVENT	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_FILE_OFFSET	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_FILE_SPECIFICATION	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_FLAG	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_FUNCTION_EXT	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_MEMORY	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_MESSAGE_PARAMETER	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_MESSAGE_REFERENCE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_OBJECT_HANDLE	X	–	–	–	–	X	X	X	X	X	–	X
CMC_E_INVALID_PARAMETER	X	X	X	X	X	X	X	X	X	X	X	X
CMC_E_INVALID_PROPERTY_ID	–	–	X	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_PROPERTY_NAME	–	–	–	–	–	X	–	X	X	X	–	–
CMC_E_INVALID_RESTRICTION	–	–	–	–	–	–	X	–	–	–	–	–
CMC_E_INVALID_SESSION_ID	X	X	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_SOURCE_OBJECT	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_STREAM_HANDLE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_INVALID_UI_ID	–	–	–	–	–	–	–	–	–	–	–	–

CUADRO 19/X.446 – CÓDIGOS DE RETORNO DE LAS FUNCIONES DE ENUMERACIÓN PARA LA INTERFAZ CMC COMPLETA *(FIN)*

Código de retorno	Obte-ner último error	Obte-ner asa de la raíz	Listar pros. Contds.	Listar número de concords.	Listar obje-tos	Listar props.	Abrir cursor	Leer cursor	Leer pros.	Leer costos de props.	Actua lizar posi-ción cursor	Act. Pos. cursor con semilla
CMC_E_INVALID_VALUE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_LOGON_FAILURE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_MESSAGE_IN_USE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_NAME_NOT_FOUND	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_NO_EVENT	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_NO_MORE_BYTES_TO_WRITE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_PASSWORD_REQUIRED	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_DATA_TYPE_NOT_ SUPPORTED	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_ID_NOT_FOUND	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_NAME_NOT_FOUND	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_PROPERTY_PROBLEMS	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_RECIPIENT_NOT_FOUND	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_REQUIRED_PROPS_MISSING	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_RESTRICTION_NOT_SUPPORTED	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_SERVICE_UNAVAILABLE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_TEXT_TOO_LARGE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_TOO_MANY_CONTENT_ITEMS	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_TOO_MANY_FILES	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_TOO_MANY_RECIPIENTS	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNABLE_TO_NOT_MARK_READ	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNBIND_FAILURE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNRECOGNIZED_IDENTIFIER	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNRECOGNIZED_MESSAGE_TYPE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_ACTION	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_CHARACTER_SET	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_DATA_EXT	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_FLAG	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_FUNCTION_EXT	X	X	X	X	X	X	X	X	X	X	X	X
CMC_E_UNSUPPORTED_KEYS	–	–	–	–	–	–	X	–	–	–	–	–
CMC_E_UNSUPPORTED_VALUE	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_UNSUPPORTED_VERSION	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_USER_CANCEL	–	–	–	–	–	–	–	–	–	–	–	–
CMC_E_USER_NOT_LOGGED_ON	–	–	–	–	–	–	–	–	–	–	–	–

CUADRO 20/X.446 – CÓDIGO DE RETORNO DE LAS FUNCIONES DE NOTIFICACIÓN DE EVENTOS Y DE MENSAJERÍA PARA LA INTERFAZ CMC COMPLETA

Código de retorno	Com-probar evento	Regis-trar evento	Anular registro de evento	Llama-das de retorno		Crear mensaje derivado	Enviar objeto mensaje
CMC_E_ACCESS_DENIED	–	–	–	–		–	–
CMC_E_AMBIGUOUS_RECIPIENT	–	–	–	–		–	–
CMC_E_ATTACHMENT_NOT_FOUND	–	–	–	–		–	–
CMC_E_ATTACHMENT_OPEN_FAILURE	–	–	–	–		–	–
CMC_E_ATTACHMENT_READ_FAILURE	–	–	–	–		–	–
CMC_E_ATTACHMENT_WRITE_FAILURE	–	–	–	–		–	–
CMC_E_BIND_FAILURE	–	–	–	–		–	–
CMC_E_CALLBACK_NOT_SUPPORTED	–	X	–	–		–	–
CMC_E_COUNTED_STRING_UNSUPPORTED	–	–	–	–		–	–
CMC_E_DISK_FULL	–	–	–	–		–	–
CMC_E_FAILURE	X	X	X	X		X	X
CMC_E_FUNCTION_INTERRUPTED	X	–	–	–		–	–
CMC_E_ID_NOT_FOUND	–	–	–	–		–	–
CMC_E_INSUFFICIENT_MEMORY	X	X	X	X		–	–
CMC_E_INVALID_CONFIGURATION	–	–	–	–		–	–
CMC_E_INVALID_CONTAINER_OBJECT	–	–	–	–		–	–
CMC_E_INVALID_CURSOR_HANDLE	–	–	–	–		–	–
CMC_E_INVALID_ENUM	–	–	–	–		–	–
CMC_E_INVALID_EVENT	X	X	X	X			
CMC_E_INVALID_FILE_OFFSET	–	–	–	–		–	–
CMC_E_INVALID_FILE_SPECIFICATION	–	–	–	–		–	–
CMC_E_INVALID_FLAG	–	–	–	–		–	–
CMC_E_INVALID_FUNCTION_EXT	X	X	X	X		–	–
CMC_E_INVALID_MEMORY	–	–	–	–		–	–
CMC_E_INVALID_MESSAGE_PARAMETER	–	–	–	–		–	–
CMC_E_INVALID_MESSAGE_REFERENCE	–	–	–	–		–	–
CMC_E_INVALID_OBJECT_HANDLE	–	–	–	–		X	X
CMC_E_INVALID_PARAMETER	X	X	X	X		X	X
CMC_E_INVALID_PROPERTY_ID	–	–	–	–		–	–
CMC_E_INVALID_PROPERTY_NAME	–	–	–	–		–	–
CMC_E_INVALID_RESTRICTION	–	–	–	–		–	–
CMC_E_INVALID_SESSION_ID	X	X	X	X		–	–
CMC_E_INVALID_SOURCE_OBJECT	–	–	–	–		–	–
CMC_E_INVALID_STREAM_HANDLE	–	–	–	–		–	–
CMC_E_INVALID_UI_ID	–	–	–	–		–	–

CUADRO 20/X.446 – CÓDIGO DE RETORNO DE LAS FUNCIONES DE NOTIFICACIÓN DE EVENTOS Y DE MENSAJERÍA PARA LA INTERFAZ CMC COMPLETA *(FIN)*

Código de retorno	Com-probar evento	Regis-trar evento	Anular registro de evento	Llama-das de retorno		Crear mensaje derivado	Enviar objeto mensaje
CMC_E_INVALID_VALUE	–	–	–	–		–	–
CMC_E_LOGON_FAILURE	–	–	–	–		–	–
CMC_E_MESSAGE_IN_USE	–	–	–	–		–	–
CMC_E_NAME_NOT_FOUND	–	–	–	–		–	–
CMC_E_NO_EVENT	X	–	–	–		–	–
CMC_E_NO_MORE_BYTES_TO_WRITE	–	–	–	–		–	–
CMC_E_NOT_SUPPORTED	–	–	X	X		–	–
CMC_E_PASSWORD_REQUIRED	–	–	–	–		–	–
CMC_E_PROPERTY_DATA_TYPE_NOT_ SUPPORTED	–	–	–	–		–	–
CMC_E_PROPERTY_ID_NOT_FOUND	–	–	–	–		–	–
CMC_E_PROPERTY_NAME_NOT_FOUND	–	–	–	–		–	–
CMC_E_PROPERTY_PROBLEMS	–	–	–	–		–	–
CMC_E_RECIPIENT_NOT_FOUND	–	–	–	–		–	–
CMC_E_REQUIRED_PROPS_MISSING	–	–	–	–		X	X
CMC_E_RESTRICTION_NOT_SUPPORTED	–	–	–	–		–	–
CMC_E_SERVICE_UNAVAILABLE	–	–	–	X		–	–
CMC_E_TEXT_TOO_LARGE							
CMC_E_TOO_MANY_CONTENT_ITEMS	–	–	–	–		–	–
CMC_E_TOO_MANY_FILES	–	–	–	–		–	–
CMC_E_TOO_MANY_RECIPIENTS	–	–	–	–		–	–
CMC_E_UNABLE_TO_NOT_MARK_READ	–	–	–	–		–	–
CMC_E_UNBIND_FAILURE	–	–	–	–		–	–
CMC_E_UNRECOGNIZED_IDENTIFIER	–	–	–	–		–	–
CMC_E_UNRECOGNIZED_MESSAGE_TYPE	–	–	–	–		–	–
CMC_E_UNSUPPORTED_ACTION	–	–	–	–		X	–
CMC_E_UNSUPPORTED_CHARACTER_SET	–	–	–	–		–	–
CMC_E_UNSUPPORTED_DATA_EXT	–	–	–	–		–	–
CMC_E_UNSUPPORTED_FLAG	–	–	–	X		–	–
CMC_E_UNSUPPORTED_FUNCTION_EXT	X	X	X	X		X	X
CMC_E_UNSUPPORTED_KEYS	–	–	–	–		–	–
CMC_E_UNSUPPORTED_VALUE	–	–	–	–		–	–
CMC_E_UNSUPPORTED_VERSION	–	–	–	–		–	–
CMC_E_USER_CANCEL	–	–	–	–		–	–
CMC_E_USER_NOT_LOGGED_ON	–	–	–	–		–	–

CUADRO 21/X.446 – CÓDIGOS DE RETORNO DE LAS FUNCIONES DE TRATAMIENTO DE NOMBRES Y DE TRENES PARA LA INTERFAZ CMC COMPLETA

Código de retorno	Id. a nombre	Nombre a Id.		Exportar tren	Importar tren	Abrir tren	Leer tren	Buscar en tren	Escribir en tren
CMC_E_ACCESS_DENIED	–	–		X	X	–	X	X	X
CMC_E_AMBIGUOUS_RECIPIENT	–	–		–	–	–	–	–	–
CMC_E_ATTACHMENT_NOT_FOUND	–	–		–	–	–	–	–	–
CMC_E_ATTACHMENT_OPEN_FAILURE	–	–		–	–	–	–	–	–
CMC_E_ATTACHMENT_READ_FAILURE	–	–		–	–	–	–	–	–
CMC_E_ATTACHMENT_WRITE_FAILURE	–	–		–	–	–	–	–	–
CMC_E_BIND_FAILURE	–	–		–	–	–	–	–	–
CMC_E_CALLBACK_NOT_SUPPORTED	–	–		–	–	–	–	–	–
CMC_E_COUNTED_STRING_UNSUPPORTED	–	–		–	–	–	–	–	–
CMC_E_DISK_FULL	–	–		X	–	–	–	–	–
CMC_E_FAILURE	X	X		X	X	X	X	X	X
CMC_E_FUNCTION_INTERRUPTED	–	–		–	–	–	–	–	–
CMC_E_ID_NOT_FOUND	–	–		–	–	–	–	–	–
CMC_E_INSUFFICIENT_MEMORY	X	X		X	X	X	X	X	X
CMC_E_INVALID_CONFIGURATION	–	–		–	–	–	–	–	–
CMC_E_INVALID_CONTAINER_OBJECT	–	–		–	–	–	–	–	–
CMC_E_INVALID_CURSOR_HANDLE	–	–		–	–	–	–	–	–
CMC_E_INVALID_ENUM	–	–		–	–	–	–	–	–
CMC_E_INVALID_EVENT	–	–		–	–	–	–	–	–
CMC_E_INVALID_FILE_OFFSET	–	–		–	X	–	–	–	–
CMC_E_INVALID_FILE_SPECIFICATION	–	–		X	X	–	–	–	–
CMC_E_INVALID_FLAG	–	–		X	X	–	–	–	–
CMC_E_INVALID_FUNCTION_EXT	–	–		–	–	–	–	–	–
CMC_E_INVALID_MEMORY	–	–		–	–	–	–	–	–
CMC_E_INVALID_MESSAGE_PARAMETER	–	–		–	–	–	–	–	–
CMC_E_INVALID_MESSAGE_REFERENCE	–	–		–	–	–	–	–	–
CMC_E_INVALID_OBJECT_HANDLE	–	–		–	–	X	–	–	–
CMC_E_INVALID_PARAMETER	X	X		X	X	–	X	X	X
CMC_E_INVALID_PROPERTY_ID	X	–	–	–	–	X	–	–	–
CMC_E_INVALID_PROPERTY_NAME	–	X		–	–	–	–	–	–
CMC_E_INVALID_RESTRICTION	–	–		–	–	–	–	–	–
CMC_E_INVALID_SESSION_ID	–	–		–	–	–	–	–	–
CMC_E_INVALID_SOURCE_OBJECT	–	–		–	–	–	–	–	–
CMC_E_INVALID_STREAM_HANDLE	–	–		X	X	–	X	X	X
CMC_E_INVALID_UI_ID	–	–		–	–	–	–	–	–

CUADRO 21/X.446 – CÓDIGOS DE RETORNO DE LAS FUNCIONES DE TRATAMIENTO DE NOMBRES Y DE TRENES PARA LA INTERFAZ CMC COMPLETA *(FIN)*

Código de retorno	Id. a nombre	Nombre a Id.		Exportar tren	Importar tren	Abrir tren	Leer tren	Buscar en tren	Escribir en tren
CMC_E_INVALID_VALUE	–	–		–	–	–	–	–	–
CMC_E_LOGON_FAILURE	–	–		–	–	–	–	–	–
CMC_E_MESSAGE_IN_USE	–	–		–	–	–	–	–	–
CMC_E_NAME_NOT_FOUND	–	–		–	–	–	–	–	–
CMC_E_NO_EVENT	–	–		–	–	–	–	–	–
CMC_E_NO_MORE_BYTES_TO_WRITE	–	–		–	–	–	–	–	X
CMC_E_NOT_SUPPORTED	–	–		–	–	–	–	–	–
CMC_E_PASSWORD_REQUIRED	–	–		–	–	–	–	–	–
CMC_E_PROPERTY_DATA_TYPE_NOT_ SUPPORTED	–	–		–	–	–	–	–	–
CMC_E_PROPERTY_ID_NOT_FOUND	–	X		–	–	–	–	–	–
CMC_E_PROPERTY_NAME_NOT_FOUND	X	–		–	–	–	–	–	–
CMC_E_PROPERTY_PROBLEMS	–	–		–	–	–	–	–	–
CMC_E_RECIPIENT_NOT_FOUND	–	–		–	–	–	–	–	–
CMC_E_REQUIRED_PROPS_MISSING	–	–		–	–	–	–	–	–
CMC_E_RESTRICTION_NOT_SUPPORTED	–	–		–	–	–	–	–	–
CMC_E_SERVICE_UNAVAILABLE	–	–		–	–	–	–	–	–
CMC_E_TEXT_TOO_LARGE	–	–		–	–	–	–	–	–
CMC_E_TOO_MANY_CONTENT_ITEMS	–	–		–	–	–	–	–	–
CMC_E_TOO_MANY_FILES	–	–		–	–	–	–	–	–
CMC_E_TOO_MANY_RECIPIENTS	–	–		–	–	–	–	–	–
CMC_E_UNABLE_TO_NOT_MARK_READ	–	–		–	–	–	–	–	–
CMC_E_UNBIND_FAILURE	–	–		–	–	–	–	–	–
CMC_E_UNRECOGNIZED_IDENTIFIER	–	–		–	–	–	–	–	–
CMC_E_UNRECOGNIZED_MESSAGE_TYPE	–	–		–	–	–	–	–	–
CMC_E_UNSUPPORTED_ACTION	–	–		–	–	–	–	–	–
CMC_E_UNSUPPORTED_CHARACTER_SET	–	–		–	–	–	–	–	–
CMC_E_UNSUPPORTED_DATA_EXT	–	–		–	–	–	–	–	–
CMC_E_UNSUPPORTED_FLAG	–	–		–	–	–	–	–	–
CMC_E_UNSUPPORTED_FUNCTION_EXT	X	X		X	X	X	X	X	X
CMC_E_UNSUPPORTED_KEYS	–	–		–	–	–	–	–	–
CMC_E_UNSUPPORTED_VALUE	–	–		–	–	–	–	–	–
CMC_E_UNSUPPORTED_VERSION	–	–		–	–	–	–	–	–
CMC_E_USER_CANCEL	–	–		–	–	–	–	–	–
CMC_E_USER_NOT_LOGGED_ON	–	–		–	–	–	–	–	–

Los códigos de retorno se definen como sigue:

CMC_E_ACCESS_DENIED (CMC E acceso denegado)	Se ha denegado el acceso.
CMC_E_AMBIGUOUS RECIPIENT (CMC E recibiente ambiguo)	El nombre del recibiente es ambiguo. Se han encontrado múltiples concordancias.
CMC_E_ATTACHMENT NOT FOUND (CMC E añadidura no encontrada)	La añadidura especificada no se encontró como fue especificada.
CMC_E_ATTACHMENT_OPEN_FAILURE (CMC E fallo de apertura de añadidura)	La añadidura especificada se encontró, pero no se pudo abrir, o no se pudo crear el fichero de añadidura.
CMC_E_ATTACHMENT_READ_FAILURE (CMC E fallo de lectura de añadidura)	La añadidura especificada se encontró y se abrió, pero se produjo un error al leerla.
CMC_E_ATTACHMENT_WRITE_FAILURE (CMC E fallo de escritura de añadidura)	El fichero de añadidura se creó correctamente, pero se produjo un error al escribirlo.
CMC_E_BIND_FAILURE (CMC E error de vinculación)	No se pudo vincular la aplicación a la implementación.
CMC_E_CALLBACK_NOT_SUPPORTED (CMC E llamada de retorno no soportada)	La llamada de retorno especificada no está soportada por la implementación.
CMC_E_COUNTED_STRING_NOT_SUPPORTED (CMC E cadena contada no soportada)	Esta implementación no soporta el tipo de cadena contada.
CMC_E_DISK_FULL (CMC E disco lleno)	El espacio de disco es insuficiente para efectuar la operación solicitada (el espacio de disco puede ser local o compartido)
CMC_E_FAILURE (CMC E fallo)	Hubo un fallo de tipo general cuya descripción no corresponde a la de ningún otro código de error.
CMC_E_FUNCTION_INTERRUPTED (CMC E función interrumpida)	La función ha sido interrumpida.
CMC_E_ID_NOT_FOUND (CMC E identificador no encontrado)	No se encontró el identificador especificado.
CMC_E_INSUFFICIENT_MEMORY (CMC E memoria insuficiente)	La memoria disponible es insuficiente para efectuar la operación solicitada.
CMC_E_INVALID_CONFIGURATION (CMC E configuración no válida)	La configuración del servicio de mensajería subyacente no es válida, por lo que no se puede establecer una sesión con este servicio.
CMC_E_INVALID_CONTAINER_OBJECT (CMC E objeto contenedor no válido)	Se ha especificado un objeto de contenedor que no es válido.
CMC_E_INVALID_CURSOR_HANDLE (CMC E asa de cursor no válida)	Se ha especificado un asa de cursor que no es válida.
CMC_E_INVALID_ENUM (CMC E enum no válido)	Un valor CMC_enum no es válido.
CMC_E_INVALID_EVENT (CMC E evento no válido)	Un evento especificado no es válido.
CMC_E_INVALID_FILE_OFFSET (CMC E desplazamiento de fichero no válido)	Se ha especificado, un desplazamiento de fichero que no es válido.
CMC_E_INVALID_FILE_ SPECIFICATION (CMC E especificación de fichero no válida)	Se ha especificado un fichero que no es válido.
CMC_E_INVALID_FLAG (CMC E bandera no válida)	Un valor de bandera en el parámetro banderas no es válido.
CMC_E_INVALID_FUNCTION_EXT (CMC E extensión de función no válida)	La extensión de la función no es válida.
CMC_E_INVALID_MEMORY (CMC E memoria no válida)	El puntero a una posición en la memoria, que se ha pasado, no es válido.
CMC_E_INVALID_MESSAGE_ PARAMETER (CMC E parámetro de mensaje no válido)	Uno de los parámetros del mensaje no es válido.

CMC_E_INVALID_MESSAGE_ REFERENCE (CMC E referencia de mensaje no válida)	La referencia de mensaje especificada no es válida o ha dejado de ser válida (por ejemplo, se ha suprimido)
CMC_E_INVALID_OBJECT_HANDLE (CMC E asa de objeto no válida)	Se ha especificado un asa de objeto que no es válida.
CMC_E_INVALID_PARAMETER (CMC E parámetro no válido)	Un parámetro de una función no es válido.
CMC_E_INVALID_PROPERTY_ID (CMC E id de propiedad no válido)	Se ha especificado un identificador de propiedad que no es válido.
CMC_E_INVALID_PROPERTY_NAME (CMC E nombre de propiedad no válido)	Se ha especificado un nombre de propiedad que no es válido.
CMC_E_INVALID_RESTRICTION (CMC E restricción no válida)	Se ha especificado una restricción que no es válida.
CMC_E_INVALID_SESSION_ID (CMC E identificador de sesión no válido)	El identificador de sesión especificado no es válido o ha dejado de ser válido (por ejemplo, después de terminada la sesión)
CMC_E_INVALID_SOURCE_OBJECT (CMC E objeto de fuente no válido)	Se ha especificado un objeto de fuente que no es válido.
CMC_E_INVALID_STREAM_HANDLE (CMC E asa de tren no válida)	Se ha especificado un asa de tren que no es válida.
CMC_E_INVALID_UI_ID (CMC E identificador de interfaz de usuario no válido)	Se ha especificado un identificador de interfaz de usuario que no es válido o que ha dejado de ser válido.
CMC_E_INVALID_VALUE (CMC E valor no válido)	El valor no es válido.
CMC_E_LOGON_FAILURE (CMC E fallo en establecimiento de sesión)	La sesión no pudo establecerse porque el nombre de servicio, nombre de usuario, y/o contraseña especificados no son válidos.
CMC_E_MESSAGE_IN_USE (CMC E mensaje en uso)	La acción solicitada no puede ejecutarse en este momento porque el mensaje se está utilizando.
CMC_E_NAME_NOT_FOUND (CMC E nombre no encontrado)	No se encontró el nombre especificado.
CMC_E_NO_EVENT (CMC E ausencia de evento)	El evento especificado no existe.
CMC_E_NO_MORE_BYTES_TO_WRITE (CMC E no hay más octetos por escribir)	No quedan más octetos por escribir en el tren.
CMC_E_NOT_SUPPORTED (CMC E no soportado)	La operación solicitada por está soportada por esta implementación.
CMC_E_PASSWORD_REQUIRED (CMC E contraseña requerida)	Se requiere una contraseña para este servicio de mensajería.
CMC_E_PROPERTY_DATA_TYPE_NOT_ SUPPORTED (CMC E tipo de datos de la propiedad no soportado)	El tipo de datos de la propiedad no está soportado por esta implementación.
CMC_E_PROPERTY_ID_NOT_FOUND (CMC E identificador de propiedad no encontrado)	No se encontró el identificador de propiedad especificado.
CMC_E_PROPERTY_NAME_NOT_FOUND (CMC E nombre de propiedad no encontrado)	No se encontró el nombre de propiedad especificado.
CMC_E_PROPERTY_PROBLEMS (CMC E problemas de propiedad)	Existen problemas relacionados con las propiedades.
CMC_E_RECIPIENT_NOT_FOUND (CMC E recibiente no encontrado)	No se encontró uno o varios recibientes especificados.
CMC_E_REQUIRED_PROPS_MISSING (CMC E ausencia de propiedades requeridas)	Falta una o varias propiedades especificadas.
CMC_E_RESTRICTION_NOT_ SUPPORTED (CMC E restricción no soportada)	La restricción especificada es demasiado compleja y no está soportada por la implementación.

CMC_E_SERVICE_UNAVAILABLE (CMC E servicio indisponible)	El servicio solicitado no está disponible.
CMC_E_TEXT_TOO_LARGE (CMC E texto demasiado largo)	La cadena de texto pasada a la implementación es demasiado larga.
CMC_E_TOO_MANY_CONTENT_ITEMS (CMC E demasiados ítems de contenido)	Se ha excedido el número máximo aceptable de ítems de contenido.
CMC_E_TOO_MANY_FILES (CMC E demasiados ficheros)	La implementación no puede soportar el número especificado de ficheros.
CMC_E_TOO_MANY_RECIPIENTS (CMC E demasiados recibientes)	La especificación no puede soportar el número especificado de recibientes.
CMC_E_UNABLE_TO_NOT_MARK_READ (CMC E incapaz para no marcar como leído)	La bandera CMC_E_UNABLE_TO_NOT_MARK_ READ no puede ser soportada.
CMC_E_UNBIND_FAILURE (CMC E fallo de desvinculación)	Se ha producido un fallo cuando se ha intentado desvincular la aplicación de la implementación.
CMC_E_UNRECOGNIZED_IDENTIFIER (CMC E identificador no reconocido)	No se reconoció el identificador especificado.
CMC_E_UNRECOGNIZED_MESSAGE_ TYPE (CMC E tipo de mensaje no reconocido)	El tipo de mensaje especificado no está soportado por esta implementación.
CMC_E_UNSUPPORTED_ACTION (CMC E acción no soportada)	La acción solicitada no está soportada por esta implementación.
CMC_E_UNSUPPORTED_CHARACTER_ SET (CMC E juego de caracteres no soportado)	El juego de caracteres solicitado no está soportado.
CMC_E_UNSUPPORTED_DATA_EXT (CMC E extensión de datos no soportada)	La extensión de datos no está soportada.
CMC_E_UNSUPPORTED_FLAG (CMC E bandera no soportada)	La bandera no está soportada
CMC_E_UNSUPPORTED_FUNCTION_EXT (CMC E extensión de función no soportada)	La extensión de la función solicitada no está soportada.
CMC_E_UNSUPPORTED_KEYS (CMC E claves no soportadas)	Las claves de clasificación especificadas no están soportadas.
CMC_E_UNSUPPORTED_VALUE (CMC E valor no soportado)	El valor no está soportado.
CMC_E_UNSUPPORTED_VERSION (CMC E versión no soportada)	La versión especificada en la llamada no puede ser soportada por esta implementación CMC.
CMC_E_USER_CANCEL (CMC E cancelación por usuario)	La operación fue cancelada por el usuario.
CMC_E_USER_NOT_LOGGED_ON (CMC E usuario no ha establecido sesión)	El usuario no ha establecido la sesión y la bandera CMC_E_USER_NOT_LOGGED_ON no está fijada.

8 Conformidad

Para que una implementación de la interfaz de programas de aplicación (API) para llamadas de mensajería común (CMC) sea conforme con esta Recomendación tiene que cumplir los siguientes criterios:

- Todas las funciones y estructuras de datos tienen que ser implementadas tal como están definidas. Los enunciados (*statements*) en otras partes de la Recomendación que describen características como facultativas, o con excepciones, tienen precedencia sobre este criterio.
- La implementación tiene que poder transportar por lo menos el tipo de mensaje IPM de la CMC.
- Se recomienda que las implementaciones de la CMC simple y de la CMC completa soporten las aplicaciones CMC 1.0 de la Asociación XAPIA.
- Las implementaciones de la CMC completa tienen que soportar la CMC simple y la CMC completa.

- Todas las clases de objetos indicadas en la cláusula 3 tienen que ser implementadas tal como están definidas. Los enunciados (*statements*) en otras partes de la Recomendación que describen características como facultativas, o con excepciones, tienen precedencia sobre este criterio.
- Las propiedades de objetos designadas como obligatorias en los cuadros de características de las propiedades deberán ser soportadas.
- El soporte del juego de caracteres incumbe a la implementación subyacente. Se requiere el soporte de un juego de caracteres por defecto definido por la implementación. Facultativamente, otros juegos de caracteres pueden estar soportados. No se requiere el soporte de cadenas contadas.
- Todas las extensiones son facultativas. Se insta a los vendedores a dar soporte al conjunto de extensiones normalizadas definidas por la CMC, especificado en esta Recomendación. Asimismo, se recomienda que se desarrollen conjuntos de extensiones normalizadas para todo servicio de mensajería, privado o no privado, al que se le proporcione una interfaz CMC, a fin de tener en cuenta las características específicas de ese servicio de mensajería, y que el conjunto de extensiones pueda registrarse externamente.
- El creador del conjunto de extensiones definirá una conformidad mínima con un conjunto de extensiones.
- El gestor CMC y la implementación CMC tienen que proporcionar una implementación de las llamadas a las funciones CMC_Bind_Implementation() y CMC_Unbind_Implementation(), y deben retornar un puntero a la tabla de despacho en la llamada a CMC_Bind_Implementation(). Si se soportan múltiples implementaciones, el Gestor CMC podría proporcionar un medio para enumerar las implementaciones CMC conocidas en una plataforma dada (capacidad de hojeado (*browsing*) facultativa) y un medio de registrar las implementaciones CMC.
- Las implementaciones CMC tienen que permitir que sus funciones sean invocadas directamente y también indirectamente a través de la tabla de despacho.

Anexo A

Sumario de declaraciones en el lenguaje de programación C

A.1 Sumario de declaración en lenguaje C

En esta subcláusula se presentan las declaraciones que definen la interfaz CMC en el lenguaje de programación C. Todas las declaraciones, salvo las relativas a constantes simbólicas, aparecen también en la cláusula 4, Estructuras de datos o en la cláusula 6, Funciones de la interfaz.

Las declaraciones aquí reunidas forman el contenido de un fichero de encabezamiento ("*header file*") que se hará accesible a los programadores de aplicaciones. El fichero de encabezamiento es **<xcmc.h>**. Los símbolos definidos por las declaraciones son los únicos símbolos que el servicio hace visible para la aplicación.

```
/*BEGIN CMC 2.0 INTERFACE*/

#ifndef_XCMC_H
#define_XCMC_H

#ifdef_cplusplus
extern "C" {
#endif

/*BASIC DATA TYPES*/
#ifndef DIFFERENT_PLATFORM
typedef char                    CMC_sint8;
typedef short                   CMC_sint16;
typedef long int                CMC_sint32;
typedef unsigned short int      CMC_uint16;
typedef unsigned long int       CMC_uint32;
typedef void *                  CMC_buffer;
typedef unsigned char           CMC_byte;
typedef long int                CMC_size;
typedef float                   CMC_float32;
typedef double                  CMC_float64;

/*CHARACTER SIZE DEFINITION*/
#ifndef CMC_WCHAR
#define CMC_CHAR                char
#else
#define CMC_CHAR                CMC_sint16
#endif
typedef CMC_CHAR *              CMC_string;
#else
typedef CMC_CHAR                char
typedef CMC_CHAR *              CMC_string;
#endif

typedef CMC_uint16              CMC_boolean;
typedef CMC_sint32              CMC_enum;
typedef CMC_uint32              CMC_return_code;
typedef CMC_uint32              CMC_flags;
typedef CMC_string              CMC_object_identifier;
typedef CMC_string              CMC_guid;
typedef CMC_string              CMC_date_time;

#define CMC_FALSE               ((CMC_boolean) 0)
#define CMC_TRUE                ((CMC_boolean) 1)

/*DATA STRUCTURES*/

/*COUNTED STRING*/
typedef struct {
       CMC_uint32               length;
       CMC_CHAR                 string[1];
} CMC_counted_string;

/*SESSION ID*/
typedef CMC_uint32              CMC_session_id;
```

```
#ifndef DIFFERENT_PLATFORM
/*CURSOR HANDLE*/
typedef CMC_uint32              CMC_cursor_handle;

/*OBJECT HANDLE*/
typedef CMC_uint32              CMC_object_handle;

/*STREAM HANDLE*/
typedef CMC_uint32              CMC_stream_handle;

/*NULLHANDLE*/
#define CMC_NULL_OBJECT_HANDLE ((CMC_object_handle) 0)
#endif

/*OPAQUE DATA*/
typedef struct CMC_TAG_OPAQUE_DATA {
       CMC_size                 size;
       CMC_byte                 *data;
} CMC_opaque_data;

/*TIME*/
/* unusedX fields needed to align struct on 4-byte boundary */
typedef struct {
       CMC_sint8                second;
       CMC_sint8                minute;
       CMC_sint8                hour;
       CMC_sint8                day;
       CMC_sint8                month;
       CMC_sint8                year;
       CMC_sint8                isdst;
       CMC_sint8                unused1;
       CMC_sint16               tmzone;
       CMC_sint16               unused2;
} CMC_time, CMC_iso_date_time;

#define CMC_NO_TIMEZONE         ((CMC_sint16) 0x8000)

/*UI ID*/
typedef CMC_uint32              CMC_ui_id;

/*EXTENSION*/
typedef struct {
       CMC_uint32               item_code;
       CMC_uint32               item_data;
       CMC_buffer               item_reference;
       CMC_flags                extension_flags;
} CMC_extension;

/*PROPERTY ID*/
typedef CMC_uint32              CMC_id;

/*PROPERTY NAME*/
typedef CMC_string              CMC_name;

/*MULTIVALUED PROPERTY DEFINITIONS*/

typedef struct CMC_TAG_ARRAY_BOOLEAN {
       CMC_uint32               count;
       CMC_boolean              *bits;
} CMC_array_boolean;

typedef struct CMC_TAG_ARRAY_BUFFER {
       CMC_uint32               count;
       CMC_buffer               *buffer;
} CMC_array_buffer;

typedef struct CMC_TAG_ARRAY_COUNTED_STRING {
       CMC_uint32               count;
       CMC_counted_string       *string;
} CMC_array_counted_string;

typedef struct CMC_TAG_ARRAY_ENUM {
       CMC_uint32               count;
       CMC_enum                 *set;
} CMC_array_enum;
```

```
typedef struct CMC_TAG_ARRAY_EXTENSION {
       CMC_uint32                count;
       CMC_extension             *extension;
} CMC_array_extension;

typedef struct CMC_TAG_ARRAY_FLOAT32 {
       CMC_uint32                count;
       CMC_float32               *number;
} CMC_array_float32;

typedef struct CMC_TAG_ARRAY_FLOAT64 {
       CMC_uint32                count;
       CMC_float64               *number;
} CMC_array_float64;

typedef struct CMC_TAG_ARRAY_GUID {
       CMC_uint32                count;
       CMC_guid                  *guid;
} CMC_array_guid;

typedef struct CMC_TAG_ARRAY_ISO_DATE_TIME {
       CMC_uint32                count;
       CMC_date_time             *time;
} CMC_array_iso_date_time;

typedef struct CMC_TAG_ARRAY_OBJECT_HANDLE {
       CMC_uint32                count;
       CMC_object_handle         *ohandles;
} CMC_array_object_handle;

typedef struct CMC_TAG_ARRAY_OPAQUE_DATA {
       CMC_uint32                count;
       CMC_opaque_data           *data;
} CMC_array_opaque_data;

typedef struct CMC_TAG_ARRAY_RETURN_CODE {
       CMC_uint32                count;
       CMC_return_code           *code;
} CMC_array_return_code;

typedef struct CMC_TAG_ARRAY_SINT16 {
       CMC_uint32                count;
       CMC_sint16                *number;
} CMC_array_sint16;

typedef struct CMC_TAG_ARRAY_SINT32 {
       CMC_uint32                count;
       CMC_sint32                *number;
} CMC_array_sint32;

typedef struct CMC_TAG_ARRAY_STRING {
       CMC_uint32                count;
       CMC_string                *string;
} CMC_array_string;

typedef struct CMC_TAG_ARRAY_TIME {
       CMC_uint32                count;
       CMC_time                  *time;
} CMC_array_time;

typedef struct CMC_TAG_ARRAY_UINT16 {
       CMC_uint32                count;
       CMC_uint16                *number;
} CMC_array_uint16;

typedef struct CMC_TAG_ARRAY_UINT32 {
       CMC_uint32                count;
       CMC_uint32                *number;
} CMC_array_uint32;
```

```
/*PROPERTY*/
typedef struct CMC_TAG_PROPERTY {
        CMC_id                      property_id;
        CMC_enum                    type;
        union {
                    CMC_boolean                 CMC_pv_boolean;
                    CMC_byte                    CMC_pv_byte;
                    CMC_buffer                  CMC_pv_buffer;
                    CMC_counted_string          CMC_pv_counted_string;
                    CMC_enum                    CMC_pv_enumerated;
                    CMC_extension               CMC_pv_extension;
                    CMC_float32                 CMC_pv_float32;
                    CMC_float64                 CMC_pv_float64;
                    CMC_flags                   CMC_pv_flags;
                    CMC_guid                    CMC_pv_guid;
                    CMC_iso_date_time           CMC_pv_iso_date_time;
                    CMC_object_handle           CMC_pv_object_handle;
                    CMC_opaque_data             CMC_pv_opaque_data;
                    CMC_return_code             CMC_pv_return_code;
                    CMC_sint16                  CMC_pv_sint16;
                    CMC_sint32                  CMC_pv_sint32;
                    CMC_string                  CMC_pv_string;
                    CMC_time                    CMC_pv_time;
                    CMC_uint16                  CMC_pv_uint16;
                    CMC_uint32                  CMC_pv_uint32;
                    CMC_array_boolean           CMC_pv_array_boolean;
                    CMC_array_buffer            CMC_pv_array_buffer;
                    CMC_array_counted_string    CMC_pv_array_counted_string;
                    CMC_array_enum              CMC_pv_array_enum;
                    CMC_array_extension         CMC_pv_array_extension;
                    CMC_array_float32           CMC_pv_array_float32;
                    CMC_array_float64           CMC_pv_array_float64;
                    CMC_array_guid              CMC_pv_array_guid;
                    CMC_array_iso_date_time     CMC_pv_array_iso_date_time;
                    CMC_array_object_handle     CMC_pv_array_object_handle;
                    CMC_array_opaque_data       CMC_pv_array_opaque_data;
                    CMC_array_return_code       CMC_pv_array_return_code;
                    CMC_array_sint16            CMC_pv_array_sint16;
                    CMC_array_sint32            CMC_pv_array_sint32;
                    CMC_array_string            CMC_pv_array_string;
                    CMC_array_time              CMC_pv_array_time;
                    CMC_array_uint16            CMC_pv_array_uint16;
                    CMC_array_uint32            CMC_pv_array_uint32;
        } value;
} CMC_property;

/*EVENT*/
typedef CMC_uint32              CMC_event;

/* EVENT TYPES */
#define CMC_EVENT_NEW_MESSAGES            ((CMC_enum) 0)

/*CALLBACK*/

typedef struct CMC_TAG_NEW_MESSAGE_CB_DATA {
        CMC_object_handle             *available;
} CMC_new_message_callback_data;

typedef struct CMC_TAG_NEW_MESSAGE_CHECK_DATA {
        CMC_uint32                    number_containers;
        CMC_object_handle             *containers;
} CMC_new_message_check_data;

typedef CMC_new_message_check_data  CMC_new_message_register_data;

typedef CMC_new_message_check_data  CMC_new_message_unregister_data;

typedef void (*CMC_callback) (
        CMC_session_id            session,
        CMC_event                 event,
        CMC_buffer                callback_data,
        CMC_buffer                register_data,
        CMC_extension             *callback_extensions
);
/*CURSOR RESTRICTION*/
```

```
typedef struct CMC_TAG_RESTRICTION_AND {
       CMC_uint32                                count;
       struct CMC_TAG_RESTRICTION_CURSOR . . .  *restriction;
} CMC_restriction_and;

typedef struct CMC_TAG_RESTRICTION_OR {
       CMC_uint32                                count;
       struct CMC_TAG_RESTRICTION_CURSOR . . .  *restriction;
} CMC_restriction_or;

typedef struct CMC_TAG_RESTRICTION_NOT {
       CMC_uint32                                count;
       struct CMC_TAG_RESTRICTION_CURSOR . . .  *restriction;
} CMC_restriction_not;

typedef struct CMC_TAG_RESTRICTION_STRING {
       CMC_enum                                  exactness;
       CMC_id                                    property;
       CMC_string                                string_constant;
} CMC_restriction_string;

typedef struct CMC_TAG_RESTRICTION_CONTENT {
       CMC_enum                                  logical;
       CMC_id                                    property;
       CMC_buffer                                property_value;
} CMC_restriction_content;

typedef struct CMC_TAG_RESTRICTION_COMPARISON {
       CMC_enum                                  logical;
       CMC_id                                    property1;
       CMC_id                                    property2;
} CMC_restriction_comparison;

typedef struct CMC_TAG_RESTRICTION_BITTEST {
       CMC_uint32                                comparison;
       CMC_id                                    property;
       CMC_uint32                                bittest;
} CMC_restriction_bittest;

typedef struct CMC_TAG_RESTRICTION_SIZE {
       CMC_enum                                  logical;
       CMC_id                                    property;
       CMC_uint32                                byte_size;
} CMC_restriction_size;

typedef struct CMC_TAG_RESTRICTION_EXIST {
       CMC_id                                    property;
} CMC_restriction_exist;

typedef struct CMC_TAG_RESTRICTION_CURSOR {
       CMC_enum                                  type;
       union {
                    CMC_restriction_and             restriction_and;
                    CMC_restriction_or              restriction_or;
                    CMC_restriction_not             restriction_not;
                    CMC_restriction_string          restriction_string;
                    CMC_restriction_content         restriction_content;
                    CMC_restriction_comparison      restriction_comparison;
                    CMC_restriction_bittest         restriction_bittest;
                    CMC_restriction_size            restriction_size;
                    CMC_restriction_exist           restriction_exist;
       } cr;
       CMC_extension                  *property_extensions;
} CMC_cursor_restriction;

/* RESTRICTION TYPES AND CONSTANTS */
#define CMC_RESTRICTION_AND                      ((CMC_enum) 0)
#define CMC_RESTRICTION_OR                       ((CMC_enum) 1)
#define CMC_RESTRICTION_NOT                      ((CMC_enum) 2)
#define CMC_RESTRICTION_STRING                   ((CMC_enum) 3)
#define CMC_RESTRICTION_CONTENT                  ((CMC_enum) 4)
#define CMC_RESTRICTION_COMPARISON               ((CMC_enum) 5)
#define CMC_RESTRICTION_BITTEST                  ((CMC_enum) 6)
#define CMC_RESTRICTION_SIZE                     ((CMC_enum) 7)
#define CMC_RESTRICTION_EXIST                    ((CMC_enum) 8)
```

```
#define CMC_EXACTNESS_PRECISE                  ((CMC_enum) 0)
#define CMC_EXACTNESS_STARTS_WITH              ((CMC_enum) 1)
#define CMC_EXACTNESS_MIXED_CASE               ((CMC_enum) 2)

#define CMC_LOGICAL_LT                         ((CMC_enum) 0)
#define CMC_LOGICAL_LE                         ((CMC_enum) 1)
#define CMC_LOGICAL_EQ                         ((CMC_enum) 2)
#define CMC_LOGICAL_NE                         ((CMC_enum) 3)
#define CMC_LOGICAL_GT                         ((CMC_enum) 4)
#define CMC_LOGICAL_GE                         ((CMC_enum) 5)

#define CMC_COMPARISON_OR                      ((CMC_enum) 0)
#define CMC_COMPARISON_AND                     ((CMC_enum) 1)

/*CURSOR SORT KEY*/
typedef struct TAG_CURSOR_SORT_KEY{
       CMC_id                        property;
       CMC_enum                      order;
} CMC_cursor_sort_key;

/* CURSOR SORT KEY CONSTANTS */
#define CMC_SORT_DEFAULT                       ((CMC_enum) 0)
#define CMC_SORT_ASCEND                        ((CMC_enum) 1)
#define CMC_SORT_DESCEND                       ((CMC_enum) 2)

/*ATTACHMENT*/

typedef struct {
       CMC_string                    attach_title;
       CMC_object_identifier         attach_type;
       CMC_string                    attach_filename;
       CMC_flags                     attach_flags;
       CMC_extension                 *attach_extensions;
} CMC_attachment;

/* ATTACHMENT FLAGS */
#define CMC_ATT_APP_OWNS_FILE              ((CMC_flags) 1)
#define CMC_ATT_LAST_ELEMENT               ((CMC_flags) 0x80000000)

#define CMC_ATT_OID_BINARY                 "1 2 840 113658 1 1"
#define CMC_ATT_OID_TEXT                   "1 2 840 113658 1 1 0"

/*MESSAGE REFERENCE*/
typedef CMC_counted_string      CMC_message_reference;

/*RECIPIENT*/
typedef struct {
       CMC_string                name;
       CMC_enum                  name_type;
       CMC_string                address;
       CMC_enum                  role;
       CMC_flags                 recip_flags;
       CMC_extension             *recip_extensions;
} CMC_recipient;

/* NAME TYPES */
#define CMC_TYPE_UNKNOWN                       ((CMC_enum) 0)
#define CMC_TYPE_INDIVIDUAL                    ((CMC_enum) 1)
#define CMC_TYPE_GROUP                         ((CMC_enum) 2)

/* ROLES */
#define CMC_ROLE_TO                            ((CMC_enum) 0)
#define CMC_ROLE_CC                            ((CMC_enum) 1)
#define CMC_ROLE_BCC                           ((CMC_enum) 2)
#define CMC_ROLE_ORIGINATOR                    ((CMC_enum) 3)
#define CMC_ROLE_AUTHORIZING_USER              ((CMC_enum) 4)
#define CMC_ROLE_REPLY_TO                      ((CMC_enum) 5)

/* RECIPIENT FLAGS */
#define CMC_RECIP_IGNORE                       ((CMC_flags) 1)
#define CMC_RECIP_LIST_TRUNCATED               ((CMC_flags) 2)
#define CMC_RECIP_LAST_ELEMENT                 ((CMC_flags) 0x80000000)
```

```
/*MESSAGE*/
typedef struct {
       CMC_message_reference         *message_reference;
       CMC_string                    message_type;
       CMC_string                    subject;
       CMC_time                      time_sent;
       CMC_string                    text_note;
       CMC_recipient                 *recipients;
       CMC_attachment                *attachments;
       CMC_flags                     message_flags;
       CMC_extension                 *message_extensions;
} CMC_message;

/* MESSAGE FLAGS */
#define CMC_MSG_READ                      ((CMC_flags) 1)
#define CMC_MSG_TEXT_NOTE_AS_FILE         ((CMC_flags) 2)
#define CMC_MSG_UNSENT                    ((CMC_flags) 4)
#define CMC_MSG_DELETE_AFTER_SEND         ((CMC_flags) 8)
#define CMC_MSG_LAST_ELEMENT              ((CMC_flags) 0x80000000)

/* MESSAGE TYPES */
#define CMC_MESSAGE_TYPE_IPM                  "CMC:IPM"
#define CMC_MESSAGE_TYPE_IP_RN                "CMC:IP RN"
#define CMC_MESSAGE_TYPE_IP_NRN               "CMC:IP NRN"
#define CMC_MESSAGE_TYPE_DR                   "CMC:DR"
#define CMC_MESSAGE_TYPE_NDR                  "CMC:NDR"
#define CMC_MESSAGE_TYPE_REPORT               "CMC:REPORT"

/*MESSAGE SUMMARY*/
typedef struct {
       CMC_message_reference         *message_reference;
       CMC_string                    message_type;
       CMC_string                    subject;
       CMC_time                      time_sent;
       CMC_uint32                    byte_length;
       CMC_recipient                 *originator;
       CMC_flags                     summary_flags;
       CMC_extension                 *message_summary_extensions;
} CMC_message_summary;

/* MESSAGE SUMMARY FLAGS */
#define CMC_SUM_READ                      ((CMC_flags) 1)
#define CMC_SUM_UNSENT                    ((CMC_flags) 2)
#define CMC_SUM_HAS_ATTACHMENTS           ((CMC_flags) 4)
#define CMC_SUM_LAST_ELEMENT              ((CMC_flags) 0x80000000)

/*REPORT*/
typedef struct {
       CMC_recipient                 *msg_recipient;
       CMC_enum                      report_type;
       CMC_time                      delivered_time;
       CMC_uint32                    reason_code;
       CMC_flags                     report_flags;
} CMC_report;

/* REPORT FLAGS */
#define CMC_REPORT_LAST_ELEMENT           ((CMC_flags) 0x00000001)

/* REPORT TYPES */
#define CMC_X400_DR                       ((CMC_enum) 0)
#define CMC_X400_NDR                      ((CMC_enum) 1)

/*CMC FUNCTIONS*/
/*CROSS FUNCTION FLAGS*/
#define CMC_ERROR_UI_ALLOWED              ((CMC_flags) 0x01000000)
#define CMC_LOGON_UI_ALLOWED              ((CMC_flags) 0x02000000)
#define CMC_COUNTED_STRING_TYPE           ((CMC_flags) 0x04000000)

/*OBJECT CLASSES*/
#define CMC_TYPE_OC_ADDRESS_BOOK              ((CMC_enum) 1)
#define CMC_TYPE_OC_CONTENT_ITEM              ((CMC_enum) 2)
#define CMC_TYPE_OC_MESSAGE                   ((CMC_enum) 3)
#define CMC_TYPE_OC_MESSAGE_CONTAINER         ((CMC_enum) 4)
```

```
#define CMC_TYPE_OC_DISTRIBUTION_LIST          ((CMC_enum) 5)
#define CMC_TYPE_OC_RECIPIENT                  ((CMC_enum) 6)
#define CMC_TYPE_OC_REPORT                     ((CMC_enum) 7)
#define CMC_TYPE_OC_ROOT_CONTAINER             ((CMC_enum) 8)
#define CMC_TYPE_OC_PER_RECIPIENT_INFORMATION  ((CMC_enum) 9)
#define CMC_TYPE_OC_PROFILE_CONTAINER          ((CMC_enum) 10)

#define CMC_OC_MESSAGE  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Message//EN"

#define CMC_OC_CONTENT_ITEM  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Content Item//EN"

#define CMC_OC_RECIPIENT  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Recipient//EN"

#define CMC_OC_REPORT  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Report//EN"

#define CMC_OC_MESSAGE_CONTAINER  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Message Container//EN"

#define CMC_OC_ADDRESS_BOOK  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Address Book//EN"

#define CMC_OC_DISTRIBUTION_LIST  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Distribution List//EN"

#define CMC_OC_ROOT_CONTAINER  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Root Container//EN"

#define CMC_OC_PER_RECIPIENT_INFORMATION  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Per Recipient Information//EN"

#define CMC_OC_PROFILE_CONTAINER  \
       "–//XAPIA/CMC/OBJECT CLASS//NONSGML Profile Container//EN"

/*OBJECT PROPERTIES*/

/* Object Class. Applies to all objects. */

#define CMC_PT_OBJECT_CLASS \
       "–//XAPIA/CMC/PROPERTY//NONSGML Object Class//EN"

/* Address Book */

#define CMC_PT_ADDRESS_BOOK_CHILD_ALLOWED \
       "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Child Allowed//EN"

#define CMC_PT_ADDRESS_BOOK_COMMENT \
       "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Comment//EN"

#define CMC_PT_ADDRESS_BOOK_LOCATION  \
       "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Location//EN"

#define CMC_PT_ADDRESS_BOOK_NAME \
       "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Name//EN"

#define CMC_PT_ADDRESS_BOOK_PARENT \
       "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Parent//EN"

#define CMC_PT_ADDRESS_BOOK_SERVER_NAME \
       "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Server Name//EN"

#define CMC_PT_ADDRESS_BOOK_SHARED \
       "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Shared//EN"

#define CMC_PT_ADDRESS_BOOK_TYPE \
       "–//XAPIA/CMC/PROPERTY//NONSGML Address Book Type//EN"

/* Content Type */

#define CMC_PT_CONTENT_ITEM_CHARACTER_SET \
       "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Character Set//EN"

#define CMC_PT_CONTENT_ITEM_CONTENT_INFORMATION \
       "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Content Information//EN"

#define CMC_PT_CONTENT_ITEM_CREATE_TIME \
       "–//XAPIA/CMC/PROPERTY//NONSGML Content Item Create Time//EN"
```

```
#define CMC_PT_CONTENT_ITEM_ENCODING_TYPE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item Encoding Type//EN"

#define CMC_PT_CONTENT_ITEM_FILE_DIRECTORY \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item File Directory//EN"

#define CMC_PT_CONTENT_ITEM_FILE_NAME \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item File Name//EN"

#define CMC_PT_CONTENT_ITEM_LAST_MODIFIED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item Last Modified//EN"

#define CMC_PT_CONTENT_ITEM_RENDER_POSITION \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item Render Position//EN"

#define CMC_PT_CONTENT_ITEM_SIZE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item Size//EN"

#define CMC_PT_CONTENT_ITEM_TITLE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item Title//EN"

#define CMC_PT_CONTENT_ITEM_CONTENT_TYPE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item Content Type//EN"

#define CMC_PT_CONTENT_ITEM_ITEM_NUMBER \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item Number//EN"

#define CMC_PT_CONTENT_ITEM_ITEM_TYPE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Content Item Item Type//EN"

/* Distribution List */

#define CMC_PT_DISTRIBUTION_LIST_NAME \
       "-//XAPIA/CMC/PROPERTY//NONSGML Distribution List Name//EN"

#define CMC_PT_DISTRIBUTION_LIST_ADDRESS \
       "-//XAPIA/CMC/PROPERTY//NONSGML Distribution List Address//EN"

#define CMC_PT_DISTRIBUTION_LIST_COMMENT \
       "-//XAPIA/CMC/PROPERTY//NONSGML Distribution List Comment//EN"

#define CMC_PT_DISTRIBUTION_LIST_LAST_MODIFICATION_TIME \
       "-//XAPIA/CMC/PROPERTY//NONSGML Distribution List Last Modification
        Time//EN"

#define CMC_PT_DISTRIBUTION_LIST_PARENT \
       "-//XAPIA/CMC/PROPERTY//NONSGML Distribution List Parent//EN"

#define CMC_PT_DISTRIBUTION_LIST_SHARED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Distribution List Shared//EN"

/* Message */

#define CMC_PT_MESSAGE_TYPE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Type//EN"

#define CMC_PT_MESSAGE_PRIORITY \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Priority//EN"

#define CMC_PT_MESSAGE_SIZE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Size//EN"

#define CMC_PT_MESSAGE_SUBJECT \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Subject//EN"

#define CMC_PT_MESSAGE_APPLICATION_ID \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Application Id//EN"

#define CMC_PT_MESSAGE_TIME_RECEIVED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Time Received//EN"

#define CMC_PT_MESSAGE_TIME_SENT \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Time Sent//EN"

#define CMC_PT_MESSAGE_DEFERRED_DELIVERY_TIME \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Deferred Delivery Time//EN"

#define CMC_PT_MESSAGE_IN_REPLY_TO \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message In Reply To//EN"
```

```
#define CMC_PT_MESSAGE_ID \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Id//EN"

#define CMC_PT_MESSAGE_RECEIPT_REQUESTED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Receipt Requested//EN"

#define CMC_PT_MESSAGE_SENSITIVITY \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Sensitivity//EN"

#define CMC_PT_MESSAGE_ITEM_COUNT \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Item Count//EN"

#define CMC_PT_MESSAGE_NRN_DIAGNOSTIC   \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message NRN Diagnostic//EN"

#define CMC_PT_MESSAGE_NRN_REASON \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message NRN Reason//EN"

#define CMC_PT_MESSAGE_RECEIPT_TYPE  \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Receipt Type//EN"

#define CMC_PT_MESSAGE_REPORT_REQUESTED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Report Requested//EN"

#define CMC_PT_MESSAGE_ROLE    \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Role//EN"

#define CMC_PT_MESSAGE_AUTO_ACTION  \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Auto Action//EN"

#define CMC_PT_CLIENT_MSG_STATUS                  \
       "-//XAPIA/CMC/PROPERTY//NONSGML Client Msg Status//EN"

#define CMC_PT_OUT_MSG_STATUS              \
       "-//XAPIA/CMC/PROPERTY//NONSGML Out Msg Status//EN"

#define CMC_PT_APPLICATION_MSG_STATUS                \
       "-//XAPIA/CMC/PROPERTY//NONSGML Application Msg Status//EN"

/* Message Container */

#define CMC_PT_MESSAGE_CONTAINER_CHILD_ALLOWED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Container Child Allowed//EN"

#define CMC_PT_MESSAGE_CONTAINER_COMMENT \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Container Comment//EN"

#define CMC_PT_MESSAGE_CONTAINER_LOCATION \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Container Location//EN"

#define CMC_PT_MESSAGE_CONTAINER_NAME \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Container Name//EN"

#define CMC_PT_MESSAGE_CONTAINER_PARENT \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Container Parent//EN"

#define CMC_PT_MESSAGE_CONTAINER_SERVER_NAME \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Container Server Name//EN"

#define CMC_PT_MESSAGE_CONTAINER_SHARED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Container Shared//EN"

#define CMC_PT_MESSAGE_CONTAINER_TYPE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Message Container Type//EN"

/* Recipient */

#define CMC_PT_RECIPIENT_ADDRESS \
       "-//XAPIA/CMC/PROPERTY//NONSGML Recipient Address//EN"

#define CMC_PROP_TYPE_RECIPIENT_CONTENT_RETURN_REQUESTED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Recipient Content Return Requested//EN"

#define CMC_PT_RECIPIENT_NAME \
       "-//XAPIA/CMC/PROPERTY//NONSGML Recipient Name//EN"

#define CMC_PT_RECIPIENT_RECEIPT_REQUESTED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Recipient Receipt Requested//EN"

#define CMC_PT_RECIPIENT_REPORT_REQUESTED   \
       "-//XAPIA/CMC/PROPERTY//NONSGML Recipient Report Requested//EN"
```

```
#define CMC_PT_RECIPIENT_ROLE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Recipient Role//EN"

#define CMC_PT_RECIPIENT_TYPE \
       "-//XAPIA/CMC/PROPERTY//NONSGML Recipient Type//EN"

#define CMC_PT_RECIPIENT_RESPONSIBILITY_FLAG \
       "-//XAPIA/CMC/PROPERTY//NONSGML Recipient Responsibility Flag//EN"

/* Report */

#define CMC_PT_REPORT_READ\
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Read//EN"

#define CMC_PT_REPORT_UNSENT  \
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Unsent//EN"

#define CMC_PT_REPORT_SIZE\
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Size//EN"

#define CMC_PT_REPORT_SUBJECT  \
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Subject//EN"

#define CMC_PT_REPORT_TIME_RECEIVED  \
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Time Received//EN"

#define CMC_PT_REPORT_TIME_SENT \
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Time Sent//EN"

#define CMC_PT_REPORT_APPLICATION_ID\
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Application Id//EN"

#define CMC_PT_REPORT_SUBJECT_MESSAGE_ID     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Subject Message Id//EN"

#define CMC_PT_REPORT_ITEM_COUNT   \
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Item Count//EN"

#define CMC_PT_REPORT_ID    \
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Id//EN"

#define CMC_PT_REPORT_MESSAGING_SYSTEM_ID   \
       "-//XAPIA/CMC/PROPERTY//NONSGML Report Messaging System Id//EN"

/* Root Container */

#define CMC_PT_ROOT_CONTAINER_CHILD_ALLOWED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Root Container Child Allowed//EN"

#define CMC_PT_ROOT_CONTAINER_COMMENT \
       "-//XAPIA/CMC/PROPERTY//NONSGML Root Container Comment//EN"

#define CMC_PT_ROOT_CONTAINER_LOCATION \
       "-//XAPIA/CMC/PROPERTY//NONSGML Root Container Location//EN"

#define CMC_PT_ROOT_CONTAINER_NAME \
       "-//XAPIA/CMC/PROPERTY//NONSGML Root Container Name//EN"

#define CMC_PT_ROOT_CONTAINER_SHARED \
       "-//XAPIA/CMC/PROPERTY//NONSGML Root Container Shared//EN"

/* Per Recipient Information */

#define CMC_PT_PRI_TYPE \
       "-//XAPIA/CMC/PROPERTY//NONSGML PRI Type//EN"

#define CMC_PT_PRI_DELIVERY_TIME   \
       "-//XAPIA/CMC/PROPERTY//NONSGML PRI Delivery Time//EN"

#define CMC_PT_PRI_REASON  \
       "-//XAPIA/CMC/PROPERTY//NONSGML PRI Reason//EN"

#define CMC_PT_PRI_DIAGNOSTIC   \
       "-//XAPIA/CMC/PROPERTY//NONSGML PRI Diagnostic//EN"

#define CMC_PT_PRI_RECIPIENT_NAME \
       "-//XAPIA/CMC/PROPERTY//NONSGML PRI Recipient Name//EN"

#define CMC_PT_PRI_RECIPIENT_ADDRESS\
       "-//XAPIA/CMC/PROPERTY//NONSGML PRI Recipient Address//EN"
```

```
#define CMC_PT_PRI_COMMENT\
       "-//XAPIA/CMC/PROPERTY//NONSGML PRI Comment//EN"

/* Profile Container */

#define CMC_PT_PROFILE_CHARACTER_SET    \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Character Set//EN"

#define CMC_PT_PROFILE_LINE_TERM        \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Line Term//EN"

#define CMC_PT_PROFILE_DEFAULT_SERVICE     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Default Service//EN"

#define CMC_PT_PROFILE_DEFAULT_USER      \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Default User//EN"

#define CMC_PT_PROFILE_REQ_PASSWORD      \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Req Password//EN"

#define CMC_PT_PROFILE_REQ_SERVICE        \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Req Service//EN"

#define CMC_PT_PROFILE_REQ_USER     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Req User//EN"

#define CMC_PT_PROFILE_UI_AVAIL     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile UI Avail//EN"

#define CMC_PT_PROFILE_SUP_NOMKMSGREAD     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Sup NoMkMsgRead//EN"

#define CMC_PT_PROFILE_SUP_COUNTED_STR     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Sup Counted Str//EN"

#define CMC_PT_PROFILE_VER_IMPLEM       \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Ver Implem//EN"

#define CMC_PT_PROFILE_VER_SPEC     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Ver Spec//EN"

#define CMC_PT_PROFILE_USERS     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Users//EN"

#define CMC_PT_PROFILE_OBJECT_SUP       \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Object Sup//EN"

#define CMC_PT_PROFILE_PROP_SUP   \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Prop Sup//EN"

#define CMC_PT_PROFILE_CONF       \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Conf//EN"

#define CMC_PT_PROFILE_OBJECT_EXT       \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Object Ext//EN"

#define CMC_PT_PROFILE_PROP_EXT     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Prop Ext//EN"

#define CMC_PT_PROFILE_AUTO_ACTION     \
       "-//XAPIA/CMC/PROPERTY//NONSGML Profile Auto Action//EN"

/* Property Value Constants. CMC_id values. */

/* Object Class. Applies to all objects. */

#define CMC_PV_OBJECT_CLASS                              0

/* Address Book */
#define CMC_PV_ADDRESS_BOOK_CHILD_ALLOWED                1
#define CMC_PV_ADDRESS_BOOK_COMMENT                      2
#define CMC_PV_ADDRESS_BOOK_LOCATION                     3
#define CMC_PV_ADDRESS_BOOK_NAME                         4
#define CMC_PV_ADDRESS_BOOK_PARENT                       5
#define CMC_PV_ADDRESS_BOOK_SERVER_NAME                  6
#define CMC_PV_ADDRESS_BOOK_SHARED                       7
#define CMC_PV_ADDRESS_BOOK_TYPE                         8
```

```
/* Content Item */
#define CMC_PV_CONTENT_ITEM_CHARACTER_SET                9
#define CMC_PV_CONTENT_ITEM_CONTENT_INFORMATION          10
#define CMC_PV_CONTENT_ITEM_CREATE_TIME                  11
#define CMC_PV_CONTENT_ITEM_ENCODING_TYPE                12
#define CMC_PV_CONTENT_ITEM_FILE_DIRECTORY               13
#define CMC_PV_CONTENT_ITEM_FILE_NAME                    14
#define CMC_PV_CONTENT_ITEM_LAST_MODIFIED                15
#define CMC_PV_CONTENT_ITEM_RENDER_POSITION              16
#define CMC_PV_CONTENT_ITEM_SIZE                         17
#define CMC_PV_CONTENT_ITEM_TITLE                        18
#define CMC_PV_CONTENT_ITEM_CONTENT_TYPE                 19
#define CMC_PV_CONTENT_ITEM_ITEM_NUMBER                  20
#define CMC_PV_CONTENT_ITEM_ITEM_TYPE                    21
/* Distribution List */
#define CMC_PV_DISTRIBUTION_LIST_NAME                    22
#define CMC_PV_DISTRIBUTION_LIST_ADDRESS                 23
#define CMC_PV_DISTRIBUTION_LIST_COMMENT                 24
#define CMC_PV_DISTRIBUTION_LIST_LAST_MODIFICATION_TIME  25
#define CMC_PV_DISTRIBUTION_LIST_PARENT                  26
#define CMC_PV_DISTRIBUTION_LIST_SHARED                  27
/* Message */
#define CMC_PV_MESSAGE_TYPE                              28
#define CMC_PV_MESSAGE_PRIORITY                          29
#define CMC_PV_MESSAGE_SIZE                              30
#define CMC_PV_MESSAGE_SUBJECT                           31
#define CMC_PV_MESSAGE_APPLICATION_ID                    32
#define CMC_PV_MESSAGE_TIME_RECEIVED                     33
#define CMC_PV_MESSAGE_TIME_SENT                         34
#define CMC_PV_MESSAGE_DEFERRED_DELIVERY_TIME            35
#define CMC_PV_MESSAGE_IN_REPLY_TO                       36
#define CMC_PV_MESSAGE_ID                                37
#define CMC_PV_MESSAGE_RECEIPT_REQUESTED                 38
#define CMC_PV_MESSAGE_SENSITIVITY                       39
#define CMC_PV_MESSAGE_ITEM_COUNT                        40
#define CMC_PV_MESSAGE_NRN_DIAGNOSTIC                    41
#define CMC_PV_MESSAGE_NRN_REASON                        42
#define CMC_PV_MESSAGE_REPORT_REQUESTED                  43
#define CMC_PV_MESSAGE_ROLE                              44
#define CMC_PV_MESSAGE_AUTO_ACTION                       45
#define CMC_PV_MESSAGE_CLIENT_MSG_STATUS                 46
#define CMC_PV_MESSAGE_OUT_MSG_STATUS                    47
#define CMC_PV_MESSAGE_APPLICATION_MSG_STATUS            48
/* Message Container */
#define CMC_PV_MESSAGE_CONTAINER_CHILD_ALLOWED           49
#define CMC_PV_MESSAGE_CONTAINER_COMMENT                 50
#define CMC_PV_MESSAGE_CONTAINER_LOCATION                51
#define CMC_PV_MESSAGE_CONTAINER_NAME                    52
#define CMC_PV_MESSAGE_CONTAINER_PARENT                  53
#define CMC_PV_MESSAGE_CONTAINER_SERVER_NAME             54
#define CMC_PV_MESSAGE_CONTAINER_SHARED                  55
#define CMC_PV_MESSAGE_CONTAINER_TYPE                    56
/* Recipient */
#define CMC_PV_RECIPIENT_ADDRESS                         57
#define CMC_PV_RECIPIENT_CONTENT_RETURN_REQUESTED        58
#define CMC_PV_RECIPIENT_NAME                            59
#define CMC_PV_RECIPIENT_RECEIPT_REQUESTED               60
#define CMC_PV_RECIPIENT_REPORT_REQUESTED                61
#define CMC_PV_RECIPIENT_ROLE                            62
#define CMC_PV_RECIPIENT_TYPE                            63
#define CMC_PV_RECIPIENT_RESPONSIBILITY_FLAG             64
/* Report */
#define CMC_PV_REPORT_READ                               65
#define CMC_PV_REPORT_UNSENT                             66
#define CMC_PV_REPORT_SIZE                               67
#define CMC_PV_REPORT_SUBJECT                            68
#define CMC_PV_REPORT_TIME_RECEIVED                      69
#define CMC_PV_REPORT_TIME_SENT                          70
#define CMC_PV_REPORT_APPLICATION_ID                     71
#define CMC_PV_REPORT_SUBJECT_MESSAGE_ID                 72
#define CMC_PV_REPORT_ITEM_COUNT                         73
#define CMC_PV_REPORT_ID                                 74
#define CMC_PV_REPORT_MESSAGING_SYSTEM_ID                75
```

```
/* Root Container */
#define CMC_PV_ROOT_CONTAINER_CHILD_ALLOWED             76
#define CMC_PV_ROOT_CONTAINER_COMMENT                   77
#define CMC_PV_ROOT_CONTAINER_LOCATION                  78
#define CMC_PV_ROOT_CONTAINER_NAME                      79
#define CMC_PV_ROOT_CONTAINER_SHARED                    80
/* Per Recipient Information */
#define CMC_PV_PRI_TYPE                                 81
#define CMC_PV_PRI_DELIVERY_TIME                        82
#define CMC_PV_PRI_REASON                               83
#define CMC_PV_PRI_DIAGNOSTIC                           84
#define CMC_PV_PRI_RECIPIENT_NAME                       85
#define CMC_PV_PRI_RECIPIENT_ADDRESS                    86
#define CMC_PV_PRI_COMMENT                              87
/* Profile */
#define CMC_PV_PROFILE_CHARACTER_SET                    88
#define CMC_PV_PROFILE_LINE_TERM                        89
#define CMC_PV_PROFILE_DEFAULT_SERVICE                  90
#define CMC_PV_PROFILE_DEFAULT_USER                     91
#define CMC_PV_PROFILE_REQ_PASSWORD                     92
#define CMC_PV_PROFILE_REQ_SERVICE                      93
#define CMC_PV_PROFILE_REQ_USER                         94
#define CMC_PV_PROFILE_UI_AVAIL                         95
#define CMC_PV_PROFILE_SUP_NOMKMSGREAD                  96
#define CMC_PV_PROFILE_SUP_COUNTED_STR                  97
#define CMC_PV_PROFILE_VER_IMPLEM                       98
#define CMC_PV_PROFILE_VER_SPEC                         99
#define CMC_PV_PROFILE_USERS                            100
#define CMC_PV_PROFILE_OBJECT_SUP                       101
#define CMC_PV_PROFILE_PROP_SUP                         102
#define CMC_PV_PROFILE_CONF                             103
#define CMC_PV_PROFILE_OBJECT_EXT                       104
#define CMC_PV_PROFILE_PROP_EXT                         105
#define CMC_PV_PROFILE_AUTO_ACTION                      106

/*OBJECT PROPERTY CONSTANT VALUES*/

/* Address Book */
#define CMC_ADDRESS_BOOK_LOCATION_LOCAL        ((CMC_enum) 0)
#define CMC_ADDRESS_BOOK_LOCATION_SERVER       ((CMC_enum) 1)
#define CMC_ADDRESS_BOOK_LOCATION_UNKNOWN      ((CMC_enum) 2)

#define CMC_ADDRESS_BOOK_TYPE_GLOBAL           ((CMC_enum) 0)
#define CMC_ADDRESS_BOOK_TYPE_PERSONAL         ((CMC_enum) 1)

/* Content Information */
#define CMC_CHARSET_437          "-//XAPIA/CHARSET//NONSGML IBM 437//EN"
#define CMC_CHARSET_850          "-//XAPIA/CHARSET//NONSGML IBM 850//EN"
#define CMC_CHARSET_1252         "-//XAPIA/CHARSET//NONSGML Microsoft 1252//EN"
#define CMC_CHARSET_ISTRING      "-//XAPIA/CHARSET//NONSGML Apple ISTRING//EN"
#define CMC_CHARSET_UNICODE      "-//XAPIA/CHARSET//NONSGML UNICODE//EN"
#define CMC_CHARSET_T61          "-//XAPIA/CHARSET//NONSGML TSS T61//EN"
#define CMC_CHARSET_IA5          "-//XAPIA/CHARSET//NONSGML TSS IA5//EN"
#define CMC_CHARSET_ISO_10646    "-//XAPIA/CHARSET//NONSGML ISO 10646//EN"
#define CMC_CHARSET_ISO_646      "-//XAPIA/CHARSET//NONSGML ISO 646//EN"
#define CMC_CHARSET_ISO_8859_1   "-//XAPIA/CHARSET//NONSGML ISO 8859-1//EN"

/* Encoding Type */
#define CMC_ET_7_BIT                                            \
       "-//XAPIA/CMC/ENCODING TYPE//NONSGML 7 Bit//EN"
#define CMC_ET_BASE64                                           \
       "-//XAPIA/CMC/ENCODING TYPE//NONSGML Base64//EN"
#define CMC_ET_BINARY                                           \
       "-//XAPIA/CMC/ENCODING TYPE//NONSGML Binary//EN"
#define CMC_ET_8_BIT                                            \
       "-//XAPIA/CMC/ENCODING TYPE//NONSGML 8 Bit//EN"
#define CMC_ET_QUOTED_PRINTABLE                                 \
       "-//XAPIA/CMC/ENCODING TYPE//NONSGML Quoted Printable//EN"

/* Content Type */
#define CMC_CT_PLAIN_TEXT                                       \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Plain Text//EN"
#define CMC_CT_GIF_IMAGE                                        \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML GIF Image//EN"
```

```
#define CMC_CT_JPEG_IMAGE                                          \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML JPEG Image//EN"
#define CMC_CT_BASIC_AUDIO                                         \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Basic Audio//EN"
#define CMC_CT_MPEG_VIDEO                                          \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML MPEG Video//EN"
#define CMC_CT_MESSAGE                                             \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Message//EN"
#define CMC_CT_PARTIAL_MESSAGE                                     \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Partial Message//EN"
#define CMC_CT_EXTERNAL_MESSAGE                                    \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML External Message//EN"
#define CMC_CT_APPLICATION_OCTET_STREAM            \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Application Octet Stream//EN"
#define CMC_CT_APPLICATION_POSTSCRIPT              \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Application PostScript//EN"
#define CMC_CT_ALTERNATIVE_MULTIPART               \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Alternative Multipart//EN"
#define CMC_CT_DIGEST_MULTIPART                                    \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Digest Multipart//EN"
#define CMC_CT_MIXED_MULTIPART                                     \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Mixed Multipart//EN"
#define CMC_CT_OLE                          \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML OLE//EN"
#define CMC_CT_MIXED_MULTIPART                                     \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML Mixed Multipart//EN"
#define CMC_CT_X400_G3_FAX \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 G3 Fax//EN"
#define CMC_CT_X400_G4_FAX \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 G4 Fax//EN"
#define CMC_CT_X400_ENCRYPTED \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Encrypted//EN"
#define CMC_CT_X400_NATIONALLY_DEFINED  \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Nationally Defined//EN"
#define CMC_CT_X400_FILE_TRANSFER   \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 File Transfer//EN"
#define CMC_CT_X400_VOICE        \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Voice//EN"
#define CMC_CT_X400_VIDEOTEX     \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Videotex//EN"
#define CMC_CT_X400_MIXED_MODE      \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Mixed Mode//EN"
#define CMC_CT_X400_PRIVATELY_DEFINED_6937       \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Privately Defined 6937//EN"
#define CMC_CT_X400_EXTERNAL_TRACE      \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 External Trace//EN"
#define CMC_CT_X400_INTERNAL_TRACE      \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML X400 Internal Trace//EN"
#define CMC_CT_SMTP_SESSION_TRANSCRIPT  \
       "-//XAPIA/CMC/CONTENT TYPE//NONSGML SMTP Session Transcript//EN"

/* Content Item Type */
#define CMC_IT_NOTE                             ((CMC_enum) 0)
#define CMC_IT_ATTACHMENT                       ((CMC_enum) 1)
#define CMC_IT_ANNOTATION                       ((CMC_enum) 2)

/* Message Types and Message Constants */
#define CMC_MT_IPM                              ((CMC_enum) 0)
#define CMC_MT_RECEIPT                          ((CMC_enum) 1)
#define CMC_MT_EDI                              ((CMC_enum) 2)
#define CMC_MT_DIRECTOR                         ((CMC_enum) 3)
#define CMC_MT_DOCMGMT                          ((CMC_enum) 4)
#define CMC_MT_WORKFLOW                         ((CMC_enum) 5)
#define CMC_MT_CALSCHED                         ((CMC_enum) 6)

#define CMC_PRIORITY_NORMAL                     ((CMC_enum) 0)
#define CMC_PRIORITY_URGENT                     ((CMC_enum) 1)
#define CMC_PRIORITY_LOW                        ((CMC_enum) 2)

#define CMC_MESSAGE_SENSITIVITY_PERSONAL        ((CMC_enum) 0)
#define CMC_MESSAGE_SENSITIVITY_PRIVATE         ((CMC_enum) 1)
#define CMC_MESSAGE_SENSITIVITY_CONFIDENTIAL    ((CMC_enum) 2)
#define CMC_MESSAGE_SENSITIVITY_NONE            ((CMC_enum) 3)
```

```
#define CMC_RECEIPT_RN                          ((CMC_enum) 0)
#define CMC_RECEIPT_NRN                         ((CMC_enum) 1)
#define CMC_RECEIPT_BOTH                        ((CMC_enum) 2)
#define CMC_RECEIPT_NONE                        ((CMC_enum) 3)

#define CMC_REPORT_DR                           ((CMC_enum) 0)
#define CMC_REPORT_NDR                          ((CMC_enum) 1)
#define CMC_REPORT_BOTH                         ((CMC_enum) 2)
#define CMC_REPORT_NONE                         ((CMC_enum) 3)

#define CMC_MESSAGE_ROLE_ORIGINAL               ((CMC_enum) 0)
#define CMC_MESSAGE_ROLE_RETURNED               ((CMC_enum) 1)
#define CMC_MESSAGE_ROLE_FORWARDED              ((CMC_enum) 2)
#define CMC_MESSAGE_ROLE_REPLIED                ((CMC_enum) 3)
#define CMC_MESSAGE_ROLE_OBSOLETED              ((CMC_enum) 4)
#define CMC_MESSAGE_ROLE_RESENT                 ((CMC_enum) 5)

#define CMC_AA_DELETE                           ((CMC_flags) 1)

/*Client Message Status*/
#define CMC_MESSAGE_STATUS_DRAFT                ((CMC_enum) 0)

/*Out Message Status*/
#define CMC_MESSAGE_STATUS_DELETED              ((CMC_enum) 0)
#define CMC_MESSAGE_STATUS_SUBMITTED            ((CMC_enum) 1)
#define CMC_MESSAGE_STATUS_SENT                 ((CMC_enum) 2)

/*In Message Status*/
#define CMC_MESSAGE_STATUS_NEW                  ((CMC_enum) 0)
#define CMC_MESSAGE_STATUS_READ                 ((CMC_enum) 1)
#define CMC_MESSAGE_STATUS_CHANGED              ((CMC_enum) 2)

/* Message Container Types and Constants */

#define CMC_MESSAGE_CONTAINER_LOCATION_LOCAL    ((CMC_enum) 0)
#define CMC_MESSAGE_CONTAINER_LOCATION_SERVER   ((CMC_enum) 1)
#define CMC_MESSAGE_CONTAINER_LOCATION_UNKNOWN  ((CMC_enum) 2)

#define CMC_MCT_INBOX                           ((CMC_enum) 0)
#define CMC_MCT_OUTBOX                          ((CMC_enum) 1)
#define CMC_MCT_DRAFTS                          ((CMC_enum) 2)
#define CMC_MCT_DELETED                         ((CMC_enum) 3)
#define CMC_MCT_FILED                           ((CMC_enum) 4)
#define CMC_MCT_SENT                            ((CMC_enum) 5)

/* Recipient */
#define CMC_RECIPIENT_ROLE_TO                   ((CMC_enum) 0)
#define CMC_RECIPIENT_ROLE_CC                   ((CMC_enum) 1)
#define CMC_RECIPIENT_ROLE_BCC                  ((CMC_enum) 2)
#define CMC_RECIPIENT_ROLE_ORIGINATOR           ((CMC_enum) 3)
#define CMC_RECIPIENT_ROLE_AUTHORIZING_USER     ((CMC_enum) 4)
#define CMC_RECIPIENT_ROLE_IN_REPLY_TO          ((CMC_enum) 5)

#define CMC_RCT_UNKNOWN                         ((CMC_enum) 0)
#define CMC_RCT_INDIVIDUAL                      ((CMC_enum) 1)
#define CMC_RCT_GROUP                           ((CMC_enum) 2)
#define CMC_RCT_REPORT_RECIPIENT                ((CMC_enum) 3)

/* Report */
#define CMC_RPT_RECEIPT_NOTICE                  ((CMC_enum) 0)
#define CMC_RPT_NONRECEIPT_NOTICE               ((CMC_enum) 1)
#define CMC_RPT_DELIVERY_NOTICE                 ((CMC_enum) 2)
#define CMC_RPT_NONDELIVERY_NOTICE              ((CMC_enum) 3)

/* Root Container */
#define CMC_ROOT_CONTAINER_LOCATION_LOCAL       ((CMC_enum) 0)
#define CMC_ROOT_CONTAINER_LOCATION_SERVER      ((CMC_enum) 1)
#define CMC_ROOT_CONTAINER_LOCATION_UNKNOWN     ((CMC_enum) 2)

/* Per Recipient Information */
#define CMC_PRI_DR                              ((CMC_enum) 0)
#define CMC_PRI_NDR                             ((CMC_enum) 1)
#define CMC_PRI_UNKNOWN                         ((CMC_enum) 2)

/* Profile */
#define CMC_CONF_SIMPLE_CMC                     ((CMC_enum) 0)
#define CMC_CONF_FULL_CMC                       ((CMC_enum) 1)
```

```
/* EXTENSION FLAGS */
#define CMC_EXT_REQUIRED                        ((CMC_flags) 0x00010000)
#define CMC_EXT_OUTPUT                          ((CMC_flags) 0x00020000)
#define CMC_EXT_LAST_ELEMENT                    ((CMC_flags) 0x80000000)
#define CMC_EXT_RSV_FLAG_MASK                   ((CMC_flags) 0xFFFF0000)
#define CMC_EXT_ITEM_FLAG_MASK                  ((CMC_flags) 0x0000FFFF)

#ifndef CMC_WCHAR

/* SEND */
CMC_return_code
cmc_send(
       CMC_session_id            session,
       CMC_message               *message,
       CMC_flags                 send_flags,
       CMC_ui_id                 ui_id,
       CMC_extension             *send_extensions
);

/* SEND DOCUMENT */
CMC_return_code
cmc_send_documents(
       CMC_string                recipient_addresses,
       CMC_string                subject,
       CMC_string                text_note,
       CMC_flags                 send_doc_flags,
       CMC_string                file_paths,
       CMC_string                file_names,
       CMC_string                delimiter,
       CMC_ui_id                 ui_id
);

/* ACT ON */
CMC_return_code
cmc_act_on(
       CMC_session_id            session,
       CMC_message_reference     *message_reference,
       CMC_enum                  operation,
       CMC_flags                 act_on_flags,
       CMC_ui_id                 ui_id,
       CMC_extension             *act_on_extensions
);

/* LIST */
CMC_return_code
cmc_list(
       CMC_session_id            session,
       CMC_string                message_type,
       CMC_flags                 list_flags,
       CMC_message_reference     *seed,
       CMC_uint32                *count,
       CMC_ui_id                 ui_id,
       CMC_message_summary       **result,
       CMC_extension             *list_extensions
);

/* READ */
CMC_return_code
cmc_read(
       CMC_session_id            session,
       CMC_message_reference     *message_reference,
       CMC_flags                 read_flags,
       CMC_message               **message,
       CMC_ui_id                 ui_id,
       CMC_extension             *read_extensions
);
```

```
/* LOOK UP */
CMC_return_code
cmc_look_up(
       CMC_session_id            session,
       CMC_recipient             *recipient_in,
       CMC_flags                 look_up_flags,
       CMC_ui_id                 ui_id,
       CMC_uint32                *count,
       CMC_recipient             **recipient_out,
       CMC_extension             *look_up_extensions
);

/* FREE */
CMC_return_code
cmc_free(
       CMC_buffer                memory
);

/* LOGOFF */
CMC_return_code
cmc_logoff(
       CMC_session_id            session,
       CMC_ui_id                 ui_id,
       CMC_flags                 logoff_flags,
       CMC_extension             *logoff_extensions
);

/* LOGON */
CMC_return_code
cmc_logon(
       CMC_string                service,
       CMC_string                user,
       CMC_string                password,
       CMC_object_identifier     character_set,
       CMC_ui_id                 ui_id,
       CMC_uint16                caller_cmc_version,
       CMC_flags                 logon_flags,
       CMC_session_id            *session,
       CMC_extension             *logon_extensions
);

/* QUERY CONFIGURATION */
CMC_return_code
cmc_query_configuration(
       CMC_session_id            session,
       CMC_enum                  item,
       CMC_buffer                reference,
       CMC_extension             *config_extensions
);

/* FULL CMC */

/* COPY OBJECT */
CMC_return_code
cmc_copy_object(
       CMC_object_handle         container,
       CMC_object_handle         source_object,
       CMC_object_handle         *new_object,
       CMC_extension             *copy_object_extensions
);

/* ADD PROPERTIES */
CMC_return_code
cmc_add_properties(
       CMC_object_handle         object,
       CMC_uint32                number_properties,
       CMC_property              *properties,
       CMC_extension             *add_properties_extensions
);
```

```
/* COMMIT OBJECT */
CMC_return_code
cmc_commit_object(
       CMC_object_handle         source_object,
       CMC_extension             *commit_object_extensions
);

/* COPY OBJECT HANDLE */
CMC_return_code
cmc_copy_object_handle(
       CMC_object_handle         source_handle,
       CMC_object_handle         *new_handle,
       CMC_extension             *copy_object_handle_extensions
);

/* CREATE DERIVED MESSAGE OBJECT */
CMC_return_code
cmc_create_derived_message_object(
       CMC_object_handle         original_message,
       CMC_enum                  derived_action,
       CMC_boolean               inherit_contents,
       CMC_object_handle         *derived_message,
       CMC_boolean               modified_message,
       CMC_extension             *create_derived_object_extensions
);

/* DELETE OBJECTS */
CMC_return_code
cmc_delete_objects(
       CMC_uint32                number_objects,
       CMC_object_handle         *object,
       CMC_extension             *delete_objects_extensions
);

/* DELETE PROPERTIES */
CMC_return_code
cmc_delete_properties(
       CMC_object_handle         object,
       CMC_uint32                number_properties,
       CMC_id                    *property_ids,
       CMC_extension             *delete_properties_extensions
);

/* GET ROOT HANDLE */
CMC_return_code
cmc_get_root_handle(
       CMC_session_id            session,
       CMC_object_handle         *root_object_handle,
       CMC_extension             *get_root_handle_extensions
);

/* IDENTIFIER TO NAME */
CMC_return_code
cmc_identifier_to_name(
       CMC_id                    identifier,
       CMC_name                  *name,
       CMC_extension             *identifier_to_name_extensions
);

/* LIST CONTAINED PROPERTIES */
CMC_return_code
cmc_list_contained_properties(
       CMC_cursor_handle         cursor,
       CMC_sint32                *number_object,
       CMC_sint32                *number_properties,
       CMC_id                    *property_ids,
       CMC_property              ***properties,
       CMC_extension             *list_contained_properties_extensions
);
```

```
/* LIST NUMBER MATCHED */
CMC_return_code
cmc_list_number_matched(
       CMC_cursor_handle         *cursor,
       CMC_uint32                *number_matches,
       CMC_extension             *list_number_matched_extensions
);

/* LIST OBJECTS */
CMC_return_code
cmc_list_objects(
       CMC_cursor_handle         *cursor,
       CMC_sint32                *number_objects,
       CMC_object_handle         **objects,
       CMC_extension             *list_objects_extensions
);

/* LIST PROPERTIES */
CMC_return_code
cmc_list_properties(
       CMC_object_handle         *object,
       CMC_uint32                *number_properties,
       CMC_id                    **property_ids,
       CMC_extension             *list_properties_extensions
);

/* NAME TO IDENTIFIER */
CMC_return_code
cmc_name_to_identifier(
       CMC_name                  name,
       CMC_id                    *identifier,
       CMC_extension             *name_to_identifier_extensions
);

/* OPEN CURSOR */
CMC_return_code
cmc_open_cursor(
       CMC_object_handle         object,
       CMC_cursor_restriction    *restrictions,
       CMC_uint32                number_sort_orders,
       CMC_cursor_sort_key       *sort_keys,
       CMC_cursor_handle         *cursor,
       CMC_extension             *open_cursor_extensions
);

/* OPEN OBJECT HANDLE */
CMC_return_code
cmc_open_object_handle(
       CMC_session_id            session,
       CMC_id                    object_class,
       CMC_object_handle         *new_object,
       CMC_extension             *open_object_handle_extensions
);

/* READ CURSOR */
CMC_return_code
cmc_read_cursor(
       CMC_cursor_handle         *cursor,
       CMC_uint32                *position_numerator,
       CMC_uint32                *position_denominator,
       CMC_extension             *read_cursor_extensions
);

/* READ PROPERTIES */
CMC_return_code
cmc_read_properties(
       CMC_object_handle         object,
       CMC_uint32                *number_properties,
       CMC_id                    *property_ids,
       CMC_property              **properties,
       CMC_extension             *read_properties_extensions
);
```

```
/* READ PROPERTY COSTS */
CMC_return_code
cmc_read_property_costs(
       CMC_object_handle         object,
       CMC_uint32                *number_properties,
       CMC_id                    *property_ids,
       CMC_enum                  **costs,
       CMC_extension             *read_property_costs_extensions
);

/* RESTORE OBJECT */
CMC_return_code
cmc_restore_object(
       CMC_object_handle         container,
       CMC_string                file_specification,
       CMC_object_handle         *restored_object,
       CMC_flags                 restore_flags,
       CMC_extension             *restore_object_extensions
);

/* SAVE OBJECT */
CMC_return_code
cmc_save_object(
       CMC_object_handle         object,
       CMC_string                file_specification,
       CMC_flags                 save_flags,
       CMC_extension             *save_object_extensions
);

/* SEND MESSAGE OBJECT */
CMC_return_code
cmc_send_message_object(
       CMC_object_handle         message_to_send,
       CMC_extension             *send_message_object_extensions
);

/* UPDATE CURSOR POSITION */
CMC_return_code
cmc_update_cursor_position(
       CMC_cursor_handle         *cursor,
       CMC_uint32                position_numerator,
       CMC_uint32                position_denominator,
       CMC_extension             *update_cursor_position_extensions
);

/* UPDATE CURSOR POSITION WITH SEED */
CMC_return_code
cmc_update_cursor_position_with_seed(
       CMC_cursor_handle         cursor,
       CMC_object_handle         seed,
       CMC_extension             *update_cursor_position_with_seed_extensions
);

/* CHECK EVENT */
CMC_return_code
cmc_check_event(
       CMC_session_id            session,
       CMC_event                 event_type,
       CMC_uint32                minimum_timeout,
       CMC_buffer                check_event_data,
       CMC_buffer                *callback_data,
       CMC_extension             *check_event_extensions
);

/* REGISTER EVENT */
CMC_return_code
cmc_register_event(
       CMC_session_id            session,
       CMC_event                 event_type,
       CMC_callback              callback,
       CMC_buffer                register_data,
       CMC_extension             *register_event_extensions
);
```

```
/* UNREGISTER EVENT */
CMC_return_code
cmc_unregister_event(
       CMC_session_id            session,
       CMC_flags                 event_type,
       CMC_callback              callback,
       CMC_buffer                unregister_data,
       CMC_extension             *unregister_event_extensions
);

/* CALL CALLBACKS */
CMC_return_code
cmc_call_callbacks(
       CMC_session_id            session,
       CMC_event                 event_type,
       CMC_extension             *call_callbacks_extensions
);

/* EXPORT STREAM */
CMC_return_code
cmc_export_stream(
       CMC_stream_handle         stream,
       CMC_string                file_specification,
       CMC_uint32                count,
       CMC_flags                 export_flags,
       CMC_extension             *export_stream_extensions
);

/* IMPORT FILE TO STREAM */
CMC_return_code
cmc_import_file_to_stream(
       CMC_stream_handle         stream,
       CMC_string                file_specification,
       CMC_uint32                file_offset,
       CMC_extension             *import_file_to_stream_extensions
);

/* OPEN STREAM */
CMC_return_code
cmc_open_stream(
       CMC_object_handle         object,
       CMC_property              *property,
       CMC_enum                  operation,
       CMC_stream_handle         *stream,
       CMC_extension             *open_stream_extensions
);

/* READ STREAM */
CMC_return_code
cmc_read_stream(
       CMC_stream_handle         stream,
       CMC_uint32                *count,
       CMC_buffer                content_information,
       CMC_extension             *read_stream_extensions
);

/* SEEK STREAM */
CMC_return_code
cmc_seek_stream(
       CMC_stream_handle         stream,
       CMC_enum                  operation,
       CMC_uint32                *location,
       CMC_extension             *seek_stream_extensions
);

/* WRITE STREAM */
CMC_return_code
cmc_write_stream(
       CMC_stream_handle         *stream,
       CMC_uint32                *count,
       CMC_buffer                *content_information,
       CMC_extension             *write_stream_extensions
);
```

```
/* GET LAST ERROR */
CMC_return_code
cmc_get_last_error(
       CMC_session_id            session,
       CMC_object_handle         objRef,
       CMC_string                *error_buffer,
       CMC_extension             *get_last_error_extensions
);

/* MULTIPLE IMPLEMENTATIONS DISPATCH TABLE */

typedef struct {
       CMC_extension                 *dispatch_table_extensions;

       /* SEND */
       CMC_return_code
       (*cmc_send)(
          CMC_session_id             session,
          CMC_message                *message,
          CMC_flags                  send_flags,
          CMC_ui_id                  ui_id,
          CMC_extension              *send_extensions
       );

       /* SEND DOCUMENT */
       CMC_return_code
       (*cmc_send_documents)(
          CMC_string                 recipient_addresses,
          CMC_string                 subject,
          CMC_string                 text_note,
          CMC_flags                  send_doc_flags,
          CMC_string                 file_paths,
          CMC_string                 file_names,
          CMC_string                 delimiter,
          CMC_ui_id                  ui_id
       );

       /* ACT ON */
       CMC_return_code
       (*cmc_act_on)(
          CMC_session_id             session,
          CMC_message_reference      *message_reference,
          CMC_enum                   operation,
          CMC_flags                  act_on_flags,
          CMC_ui_id                  ui_id,
          CMC_extension              *act_on_extensions
       );

       /* LIST */
       CMC_return_code
       (*cmc_list)(
          CMC_session_id             session,
          CMC_string                 message_type,
          CMC_flags                  list_flags,
          CMC_message_reference      *seed,
          CMC_uint32                 *count,
          CMC_ui_id                  ui_id,
          CMC_message_summary        **result,
          CMC_extension              *list_extensions
       );

       /* READ */
       CMC_return_code
       (*cmc_read)(
          CMC_session_id             session,
          CMC_message_reference      *message_reference,
          CMC_flags                  read_flags,
          CMC_message                **message,
          CMC_ui_id                  ui_id,
          CMC_extension              *read_extensions
       );
```

```
/* LOOK UP */
CMC_return_code
(*cmc_look_up)(
   CMC_session_id             session,
   CMC_recipient              *recipient_in,
   CMC_flags                  look_up_flags,
   CMC_ui_id                  ui_id,
   CMC_uint32                 *count,
   CMC_recipient              **recipient_out,
   CMC_extension              *look_up_extensions
);

/* FREE */
CMC_return_code
(*cmc_free)(
   CMC_buffer                 memory
);

/* LOGOFF */
CMC_return_code
(*cmc_logoff)(
   CMC_session_id             session,
   CMC_ui_id                  ui_id,
   CMC_flags                  logoff_flags,
   CMC_extension              *logoff_extensions
);

/* LOGON */
CMC_return_code
(*cmc_logon)(
   CMC_string                 service,
   CMC_string                 user,
   CMC_string                 password,
   CMC_object_identifier      character_set,
   CMC_ui_id                  ui_id,
   CMC_uint16                 caller_cmc_version,
   CMC_flags                  logon_flags,
   CMC_session_id             *session,
   CMC_extension              *logon_extensions
);

/* QUERY CONFIGURATION */
CMC_return_code
(*cmc_query_configuration)(
   CMC_session_id             session,
   CMC_enum                   item,
   CMC_buffer                 reference,
   CMC_extension              *config_extensions
);

/* FULL CMC */

/* COPY OBJECT */
CMC_return_code
(*cmc_copy_object)(
   CMC_object_handle          container,
   CMC_object_handle          source_object,
   CMC_object_handle          *new_object,
   CMC_extension              *copy_object_extensions
);

/* ADD PROPERTIES */
CMC_return_code
(*cmc_add_properties)(
   CMC_object_handle          object,
   CMC_uint32                 number_properties,
   CMC_property               *properties,
   CMC_extension              *add_properties_extensions
);
```

```
/* COMMIT OBJECT */
CMC_return_code
(*cmc_commit_object)(
   CMC_object_handle          source_object,
   CMC_extension              *commit_object_extensions
);

/* COPY OBJECT HANDLE */
CMC_return_code
(*cmc_copy_object_handle)(
   CMC_object_handle          source_handle,
   CMC_object_handle          *new_handle,
   CMC_extension              *copy_object_handle_extensions
);

/* CREATE DERIVED MESSAGE OBJECT */
CMC_return_code
(*cmc_create_derived_message_object)(
   CMC_object_handle          original_message,
   CMC_enum                   derived_action,
   CMC_boolean                inherit_contents,
   CMC_object_handle          *derived_message,
   CMC_boolean                modified_message,
   CMC_extension              *create_derived_object_extensions
);

/* DELETE OBJECTS */
CMC_return_code
(*cmc_delete_objects)(
   CMC_uint32                 number_objects,
   CMC_object_handle          *object,
   CMC_extension              *delete_objects_extensions
);

/* DELETE PROPERTIES */
CMC_return_code
(*cmc_delete_properties)(
   CMC_object_handle          object,
   CMC_uint32                 number_properties,
   CMC_id                     *property_ids,
   CMC_extension              *delete_properties_extensions
);

/* GET ROOT HANDLE */
CMC_return_code
(*cmc_get_root_handle)(
   CMC_session_id             session,
   CMC_object_handle          *root_object_handle,
   CMC_extension              *get_root_handle_extensions
);

/* IDENTIFIER TO NAME */
CMC_return_code
(*cmc_identifier_to_name)(
   CMC_id                     identifier,
   CMC_name                   *name,
   CMC_extension              *identifier_to_name_extensions
);

/* LIST CONTAINED PROPERTIES */
CMC_return_code
(*cmc_list_contained_properties)(
   CMC_cursor_handle          cursor,
   CMC_sint32                 *number_object,
   CMC_sint32                 *number_properties,
   CMC_id                     *property_ids,
   CMC_property               ***properties,
   CMC_extension              *list_contained_properties_extensions
);
```

```
/* LIST NUMBER MATCHED */
CMC_return_code
(*cmc_list_number_matched)(
   CMC_cursor_handle          *cursor,
   CMC_uint32                 *number_matches,
   CMC_extension              *list_number_matched_extensions
);

/* LIST OBJECTS */
CMC_return_code
(*cmc_list_objects)(
   CMC_cursor_handle          *cursor,
   CMC_sint32                 *number_objects,
   CMC_object_handle          **objects,
   CMC_extension              *list_objects_extensions
);

/* LIST PROPERTIES */
CMC_return_code
(*cmc_list_properties)(
   CMC_object_handle          *object,
   CMC_uint32                 *number_properties,
   CMC_id                     **property_ids,
   CMC_extension              *list_properties_extensions
);

/* NAME TO IDENTIFIER */
CMC_return_code
(*cmc_name_to_identifier)(
   CMC_name                   name,
   CMC_id                     *identifier,
   CMC_extension              *name_to_identifier_extensions
);

/* OPEN CURSOR */
CMC_return_code
(*cmc_open_cursor)(
   CMC_object_handle          object,
   CMC_cursor_restriction     *restrictions,
   CMC_uint32                 number_sort_orders,
   CMC_cursor_sort_key        *sort_keys,
   CMC_cursor_handle          *cursor,
   CMC_extension              *open_cursor_extensions
);

/* OPEN OBJECT HANDLE */
CMC_return_code
(*cmc_open_object_handle)(
   CMC_session_id             session,
   CMC_id                     object_class,
   CMC_object_handle          *new_object,
   CMC_extension              *open_object_handle_extensions
);

/* READ CURSOR */
CMC_return_code
(*cmc_read_cursor)(
   CMC_cursor_handle          *cursor,
   CMC_uint32                 *position_numerator,
   CMC_uint32                 *position_denominator,
   CMC_extension              *read_cursor_extensions
);

/* READ PROPERTIES */
CMC_return_code
(*cmc_read_properties)(
   CMC_object_handle          object,
   CMC_uint32                 *number_properties,
   CMC_id                     *property_ids,
   CMC_property               **properties,
   CMC_extension              *read_properties_extensions
);
```

```
/* READ PROPERTY COSTS */
CMC_return_code
(*cmc_read_property_costs)(
   CMC_object_handle          object,
   CMC_uint32                 *number_properties,
   CMC_id                     *property_ids,
   CMC_enum                   **costs,
   CMC_extension              *read_property_costs_extensions
);

/* RESTORE OBJECT */
CMC_return_code
(*cmc_restore_object)(
   CMC_object_handle          container,
   CMC_string                 file_specification,
   CMC_object_handle          *restored_object,
   CMC_flags                  restore_flags,
   CMC_extension              *restore_object_extensions
);

/* SAVE OBJECT */
CMC_return_code
(*cmc_save_object)(
   CMC_object_handle          object,
   CMC_string                 file_specification,
   CMC_flags                  save_flags,
   CMC_extension              *save_object_extensions
);

/* SEND MESSAGE OBJECT */
CMC_return_code
(*cmc_send_message_object)(
   CMC_object_handle          message_to_send,
   CMC_extension              *send_message_object_extensions
);

/* UPDATE CURSOR POSITION */
CMC_return_code
(*cmc_update_cursor_position)(
   CMC_cursor_handle          *cursor,
   CMC_uint32                 position_numerator,
   CMC_uint32                 position_denominator,
   CMC_extension              *update_cursor_position_extensions
);

/* UPDATE CURSOR POSITION WITH SEED */
CMC_return_code
(*cmc_update_cursor_position_with_seed)(
   CMC_cursor_handle          cursor,
   CMC_object_handle          seed,
   CMC_extension              *update_cursor_position_with_seed_extensions
);

/* CHECK EVENT */
CMC_return_code
(*cmc_check_event)(
   CMC_session_id             session,
   CMC_event                  event_type,
   CMC_uint32                 minimum_timeout,
   CMC_buffer                 check_event_data,
   CMC_buffer                 *callback_data,
   CMC_extension              *check_event_extensions
);

/* REGISTER EVENT */
CMC_return_code
(*cmc_register_event)(
   CMC_session_id             session,
   CMC_event                  event_type,
   CMC_callback               callback,
   CMC_buffer                 register_data,
   CMC_extension              *register_event_extensions
);
```

```
/* UNREGISTER EVENT */
CMC_return_code
(*cmc_unregister_event)(
   CMC_session_id             session,
   CMC_flags                  event_type,
   CMC_callback               callback,
   CMC_buffer                 unregister_data,
   CMC_extension              *unregister_event_extensions
);

/* CALL CALLBACKS */
CMC_return_code
(*cmc_call_callbacks)(
   CMC_session_id             session,
   CMC_event                  event_type,
   CMC_extension              *call_callbacks_extensions
);

/* EXPORT STREAM */
CMC_return_code
(*cmc_export_stream)(
   CMC_stream_handle          stream,
   CMC_string                 file_specification,
   CMC_uint32                 count,
   CMC_flags                  export_flags,
   CMC_extension              *export_stream_extensions
);

/* IMPORT FILE TO STREAM */
CMC_return_code
(*cmc_import_file_to_stream)(
   CMC_stream_handle          stream,
   CMC_string                 file_specification,
   CMC_uint32                 file_offset,
   CMC_extension              *import_file_to_stream_extensions
);

/* OPEN STREAM */
CMC_return_code
(*cmc_open_stream)(
   CMC_object_handle          object,
   CMC_property               *property,
   CMC_enum                   operation,
   CMC_stream_handle          *stream,
   CMC_extension              *open_stream_extensions
);

/* READ STREAM */
CMC_return_code
(*cmc_read_stream)(
   CMC_stream_handle          stream,
   CMC_uint32                 *count,
   CMC_buffer                 content_information,
   CMC_extension              *read_stream_extensions
);

/* SEEK STREAM */
CMC_return_code
(*cmc_seek_stream)(
   CMC_stream_handle          stream,
   CMC_enum                   operation,
   CMC_uint32                 *location,
   CMC_extension              *seek_stream_extensions
);

/* WRITE STREAM */
CMC_return_code
(*cmc_write_stream)(
   CMC_stream_handle          *stream,
   CMC_uint32                 *count,
   CMC_buffer                 *content_information,
   CMC_extension              *write_stream_extensions
);
```

```
        /* GET LAST ERROR */
        CMC_return_code
        (*cmc_get_last_error)(
            CMC_session_id              session,
            CMC_object_handle           objRef,
            CMC_string                  *error_buffer,
            CMC_extension               *get_last_error_extensions
        );

} CMC_dispatch_table;

/* BIND IMPLEMENTATION */
CMC_return_code
cmc_bind_implementation(
        CMC_guid                    implementation_name,
        CMC_dispatch_table          **dispatch_table,
        CMC_extension               *cmc_bind_extensions
);

/* UNBIND IMPLEMENTATION */
CMC_return_code
cmc_unbind_implementation(
        CMC_guid                    implementation_name,
        CMC_extension               *cmc_unbind_implementation_extensions
);

#else   /* WCHAR / UNICODE Function Counterparts */

/* SEND */
CMC_return_code
cmc_send_W(
        CMC_session_id              session,
        CMC_message                 *message,
        CMC_flags                   send_flags,
        CMC_ui_id                   ui_id,
        CMC_extension               *send_extensions
);

/* SEND DOCUMENT */
CMC_return_code
cmc_send_documents_W(
        CMC_string                  recipient_addresses,
        CMC_string                  subject,
        CMC_string                  text_note,
        CMC_flags                   send_doc_flags,
        CMC_string                  file_paths,
        CMC_string                  file_names,
        CMC_string                  delimiter,
        CMC_ui_id                   ui_id
);

/* ACT ON */
CMC_return_code
cmc_act_on_W(
        CMC_session_id              session,
        CMC_message_reference       *message_reference,
        CMC_enum                    operation,
        CMC_flags                   act_on_flags,
        CMC_ui_id                   ui_id,
        CMC_extension               *act_on_extensions
);

/* LIST */
CMC_return_code
cmc_list_W(
        CMC_session_id              session,
        CMC_string                  message_type,
        CMC_flags                   list_flags,
        CMC_message_reference       *seed,
        CMC_uint32                  *count,
        CMC_ui_id                   ui_id,
        CMC_message_summary         **result,
        CMC_extension               *list_extensions
);
```

```
/* READ */
CMC_return_code
cmc_read_W(
       CMC_session_id            session,
       CMC_message_reference     *message_reference,
       CMC_flags                 read_flags,
       CMC_message               **message,
       CMC_ui_id                 ui_id,
       CMC_extension             *read_extensions
);

/* LOOK UP */
CMC_return_code
cmc_look_up_W(
       CMC_session_id            session,
       CMC_recipient             *recipient_in,
       CMC_flags                 look_up_flags,
       CMC_ui_id                 ui_id,
       CMC_uint32                *count,
       CMC_recipient             **recipient_out,
       CMC_extension             *look_up_extensions
);

/* FREE */
CMC_return_code
cmc_free_W(
       CMC_buffer                memory
);

/* LOGOFF */
CMC_return_code
cmc_logoff_W(
       CMC_session_id            session,
       CMC_ui_id                 ui_id,
       CMC_flags                 logoff_flags,
       CMC_extension             *logoff_extensions
);

/* LOGON */
CMC_return_code
cmc_logon_W(
       CMC_string                service,
       CMC_string                user,
       CMC_string                password,
       CMC_object_identifier     character_set,
       CMC_ui_id                 ui_id,
       CMC_uint16                caller_cmc_version,
       CMC_flags                 logon_flags,
       CMC_session_id            *session,
       CMC_extension             *logon_extensions
);

/* QUERY CONFIGURATION */
CMC_return_code
cmc_query_configuration_W(
       CMC_session_id            session,
       CMC_enum                  item,
       CMC_buffer                reference,
       CMC_extension             *config_extensions
);

/* FULL CMC */

/* COPY OBJECT */
CMC_return_code
cmc_copy_object_W(
       CMC_object_handle         container,
       CMC_object_handle         source_object,
       CMC_object_handle         *new_object,
       CMC_extension             *copy_object_extensions
);
```

```
/* ADD PROPERTIES */
CMC_return_code
cmc_add_properties_W(
       CMC_object_handle         object,
       CMC_uint32                number_properties,
       CMC_property              *properties,
       CMC_extension             *add_properties_extensions
);

/* COMMIT OBJECT */
CMC_return_code
cmc_commit_object_W(
       CMC_object_handle         source_object,
       CMC_extension             *commit_object_extensions
);

/* COPY OBJECT HANDLE */
CMC_return_code
cmc_copy_object_handle_W(
       CMC_object_handle         source_handle,
       CMC_object_handle         *new_handle,
       CMC_extension             *copy_object_handle_extensions
);

/* CREATE DERIVED MESSAGE OBJECT */
CMC_return_code
cmc_create_derived_message_object_W(
       CMC_object_handle         original_message,
       CMC_enum                  derived_action,
       CMC_boolean               inherit_contents,
       CMC_object_handle         *derived_message,
       CMC_boolean               modified_message,
       CMC_extension             *create_derived_object_extensions
);

/* DELETE OBJECTS */
CMC_return_code
cmc_delete_objects_W(
       CMC_uint32                number_objects,
       CMC_object_handle         *object,
       CMC_extension             *delete_objects_extensions
);

/* DELETE PROPERTIES */
CMC_return_code
cmc_delete_properties_W(
       CMC_object_handle         object,
       CMC_uint32                number_properties,
       CMC_id                    *property_ids,
       CMC_extension             *delete_properties_extensions
);

/* GET ROOT HANDLE */
CMC_return_code
cmc_get_root_handle_W(
       CMC_session_id            session,
       CMC_object_handle         *root_object_handle,
       CMC_extension             *get_root_handle_extensions
);

/* IDENTIFIER TO NAME */
CMC_return_code
cmc_identifier_to_name_W(
       CMC_id                    identifier,
       CMC_name                  *name,
       CMC_extension             *identifier_to_name_extensions
);
```

```
/* LIST CONTAINED PROPERTIES */
CMC_return_code
cmc_list_contained_properties_W(
       CMC_cursor_handle         cursor,
       CMC_sint32                *number_object,
       CMC_sint32                *number_properties,
       CMC_id                    *property_ids,
       CMC_property              ***properties,
       CMC_extension             *list_contained_properties_extensions
);

/* LIST NUMBER MATCHED */
CMC_return_code
cmc_list_number_matched_W(
       CMC_cursor_handle         *cursor,
       CMC_uint32                *number_matches,
       CMC_extension             *list_number_matched_extensions
);

/* LIST OBJECTS */
CMC_return_code
cmc_list_objects_W(
       CMC_cursor_handle         *cursor,
       CMC_sint32                *number_objects,
       CMC_object_handle         **objects,
       CMC_extension             *list_objects_extensions
);

/* LIST PROPERTIES */
CMC_return_code
cmc_list_properties_W(
       CMC_object_handle         *object,
       CMC_uint32                *number_properties,
       CMC_id                    **property_ids,
       CMC_extension             *list_properties_extensions
);

/* NAME TO IDENTIFIER */
CMC_return_code
cmc_name_to_identifier_W(
       CMC_name                  property_name,
       CMC_id                    *property_id,
       CMC_extension             *name_to_identifier_extensions
);

/* OPEN CURSOR */
CMC_return_code
cmc_open_cursor_W(
       CMC_object_handle         object,
       CMC_cursor_restriction    *restrictions,
       CMC_uint32                number_sort_orders,
       CMC_cursor_sort_key       *sort_keys,
       CMC_cursor_handle         *cursor,
       CMC_extension             *open_cursor_extensions
);

/* OPEN OBJECT HANDLE */
CMC_return_code
cmc_open_object_handle_W(
       CMC_session_id            session,
       CMC_id                    object_class,
       CMC_object_handle         *new_object,
       CMC_extension             *open_object_handle_extensions
);

/* READ CURSOR */
CMC_return_code
cmc_read_cursor_W(
       CMC_cursor_handle         *cursor,
       CMC_uint32                *position_numerator,
       CMC_uint32                *position_denominator,
       CMC_extension             *read_cursor_extensions
);
```

```
/* READ PROPERTIES */
CMC_return_code
cmc_read_properties_W(
       CMC_object_handle         object,
       CMC_uint32                *number_properties,
       CMC_id                    *property_ids,
       CMC_property              **properties,
       CMC_extension             *read_properties_extensions
);

/* READ PROPERTY COSTS */
CMC_return_code
cmc_read_property_costs_W(
       CMC_object_handle         object,
       CMC_uint32                *number_properties,
       CMC_id                    *property_ids,
       CMC_enum                  **costs,
       CMC_extension             *read_property_costs_extensions
);

/* RESTORE OBJECT */
CMC_return_code
cmc_restore_object_W(
       CMC_object_handle         container,
       CMC_string                file_specification,
       CMC_object_handle         *restored_object,
       CMC_flags                 restore_flags,
       CMC_extension             *restore_object_extensions
);

/* SAVE OBJECT */
CMC_return_code
cmc_save_object_W(
       CMC_object_handle         object,
       CMC_string                file_specification,
       CMC_flags                 save_flags,
       CMC_extension             *save_object_extensions
);

/* SEND MESSAGE OBJECT */
CMC_return_code
cmc_send_message_object_W(
       CMC_object_handle         message_to_send,
       CMC_extension             *send_message_object_extensions
);

/* UPDATE CURSOR POSITION */
CMC_return_code
cmc_update_cursor_position_W(
       CMC_cursor_handle         *cursor,
       CMC_uint32                position_numerator,
       CMC_uint32                position_denominator,
       CMC_extension             *update_cursor_position_extensions
);

/* UPDATE CURSOR POSITION WITH SEED */
CMC_return_code
cmc_update_cursor_position_with_seed_W(
       CMC_cursor_handle         cursor,
       CMC_object_handle         seed,
       CMC_extension             *update_cursor_position_with_seed_extensions
);

/* CHECK EVENT */
CMC_return_code
cmc_check_event_W(
       CMC_session_id            session,
       CMC_event                 event_type,
       CMC_uint32                minimum_timeout,
       CMC_buffer                check_event_data,
       CMC_buffer                *callback_data,
       CMC_extension             *check_event_extensions
);
```

```
/* REGISTER EVENT */
CMC_return_code
cmc_register_event_W(
       CMC_session_id            session,
       CMC_event                 event_type,
       CMC_callback              callback,
       CMC_buffer                register_data,
       CMC_extension             *register_event_extensions
);

/* UNREGISTER EVENT */
CMC_return_code
cmc_unregister_event_W(
       CMC_session_id            session,
       CMC_flags                 event_type,
       CMC_callback              callback,
       CMC_buffer                unregister_data,
       CMC_extension             *unregister_event_extensions
);

/* CALL CALLBACKS */
CMC_return_code
cmc_call_callbacks_W(
       CMC_session_id            session,
       CMC_event                 event_type,
       CMC_extension             *call_callbacks_extensions
);

/* EXPORT STREAM */
CMC_return_code
cmc_export_stream_W(
       CMC_stream_handle         stream,
       CMC_string                file_specification,
       CMC_uint32                count,
       CMC_flags                 export_flags,
       CMC_extension             *export_stream_extensions
);

/* IMPORT FILE TO STREAM */
CMC_return_code
cmc_import_file_to_stream_W(
       CMC_stream_handle         stream,
       CMC_string                file_specification,
       CMC_uint32                file_offset,
       CMC_extension             *import_file_to_stream_extensions
);

/* OPEN STREAM */
CMC_return_code
cmc_open_stream_W(
       CMC_object_handle         object,
       CMC_property              *property,
       CMC_enum                  operation,
       CMC_stream_handle         *stream,
       CMC_extension             *open_stream_extensions
);

/* READ STREAM */
CMC_return_code
cmc_read_stream_W(
       CMC_stream_handle         stream,
       CMC_uint32                *count,
       CMC_buffer                content_information,
       CMC_extension             *read_stream_extensions
);

/* SEEK STREAM */
CMC_return_code
cmc_seek_stream_W(
       CMC_stream_handle         stream,
       CMC_enum                  operation,
       CMC_uint32                *location,
       CMC_extension             *seek_stream_extensions
);
```

```
/* WRITE STREAM */
CMC_return_code
cmc_write_stream_W(
       CMC_stream_handle         *stream,
       CMC_uint32                *count,
       CMC_buffer                *content_information,
       CMC_extension             *write_stream_extensions
);

/* GET LAST ERROR */
CMC_return_code
cmc_get_last_error_W(
       CMC_session_id            session,
       CMC_object_handle         objRef,
       CMC_string                *error_buffer,
       CMC_extension             *get_last_error_extensions
);

/* MULTIPLE IMPLEMENTATIONS DISPATCH TABLE UNICODE */

typedef struct {
       CMC_extension                 *dispatch_table_extensions;
       /* SEND */
       CMC_return_code
       (*cmc_send_W)(
              CMC_session_id         session,
              CMC_message            *message,
              CMC_flags              send_flags,
              CMC_ui_id              ui_id,
              CMC_extension          *send_extensions
       );

       /* SEND DOCUMENT */
       CMC_return_code
       (*cmc_send_documents_W)(
              CMC_string             recipient_addresses,
              CMC_string             subject,
              CMC_string             text_note,
              CMC_flags              send_doc_flags,
              CMC_string             file_paths,
              CMC_string             file_names,
              CMC_string             delimiter,
              CMC_ui_id              ui_id
       );

       /* ACT ON */
       CMC_return_code
       (*cmc_act_on_W)(
              CMC_session_id         session,
              CMC_message_reference  *message_reference,
              CMC_enum               operation,
              CMC_flags              act_on_flags,
              CMC_ui_id              ui_id,
              CMC_extension          *act_on_extensions
       );

       /* LIST */
       CMC_return_code
       (*cmc_list_W)(
              CMC_session_id         session,
              CMC_string             message_type,
              CMC_flags              list_flags,
              CMC_message_reference  *seed,
              CMC_uint32             *count,
              CMC_ui_id              ui_id,
              CMC_message_summary    **result,
              CMC_extension          *list_extensions
       );
```

```
/* READ */
CMC_return_code
(*cmc_read_W)(
       CMC_session_id          session,
       CMC_message_reference   *message_reference,
       CMC_flags               read_flags,
       CMC_message             **message,
       CMC_ui_id               ui_id,
       CMC_extension           *read_extensions
);

/* LOOK UP */
CMC_return_code
(*cmc_look_up_W)(
       CMC_session_id          session,
       CMC_recipient           *recipient_in,
       CMC_flags               look_up_flags,
       CMC_ui_id               ui_id,
       CMC_uint32              *count,
       CMC_recipient           **recipient_out,
       CMC_extension           *look_up_extensions
);

/* FREE */
CMC_return_code
(*cmc_free_W)(
       CMC_buffer              memory
);

/* LOGOFF */
CMC_return_code
(*cmc_logoff_W)(
       CMC_session_id          session,
       CMC_ui_id               ui_id,
       CMC_flags               logoff_flags,
       CMC_extension           *logoff_extensions
);

/* LOGON */
CMC_return_code
(*cmc_logon_W)(
       CMC_string              service,
       CMC_string              user,
       CMC_string              password,
       CMC_object_identifier   character_set,
       CMC_ui_id               ui_id,
       CMC_uint16              caller_cmc_version,
       CMC_flags               logon_flags,
       CMC_session_id          *session,
       CMC_extension           *logon_extensions
);

/* QUERY CONFIGURATION */
CMC_return_code
(*cmc_query_configuration_W)(
       CMC_session_id          session,
       CMC_enum                item,
       CMC_buffer              reference,
       CMC_extension           *config_extensions
);

/* FULL CMC */

/* COPY OBJECT */
CMC_return_code
(*cmc_copy_object_W)(
       CMC_object_handle       container,
       CMC_object_handle       source_object,
       CMC_object_handle       *new_object,
       CMC_extension           *copy_object_extensions
);
```

```
/* ADD PROPERTIES */
CMC_return_code
(*cmc_add_properties_W)(
       CMC_object_handle       object,
       CMC_uint32              number_properties,
       CMC_property            *properties,
       CMC_extension           *add_properties_extensions
);

/* COMMIT OBJECT */
CMC_return_code
(*cmc_commit_object_W)(
       CMC_object_handle       source_object,
       CMC_extension           *commit_object_extensions
);

/* COPY OBJECT HANDLE */
CMC_return_code
(*cmc_copy_object_handle_W)(
       CMC_object_handle       source_handle,
       CMC_object_handle       *new_handle,
       CMC_extension           *copy_object_handle_extensions
);

/* CREATE DERIVED MESSAGE OBJECT */
CMC_return_code
(*cmc_create_derived_message_object_W)(
       CMC_object_handle       original_message,
       CMC_enum                derived_action,
       CMC_boolean             inherit_contents,
       CMC_object_handle       *derived_message,
       CMC_boolean             modified_message,
       CMC_extension           *create_derived_object_extensions
);

/* DELETE OBJECTS */
CMC_return_code
(*cmc_delete_objects_W)(
       CMC_uint32              number_objects,
       CMC_object_handle       *object,
       CMC_extension           *delete_objects_extensions
);

/* DELETE PROPERTIES */
CMC_return_code
(*cmc_delete_properties_W)(
       CMC_object_handle       object,
       CMC_uint32              number_properties,
       CMC_id                  *property_ids,
       CMC_extension           *delete_properties_extensions
);

/* GET ROOT HANDLE */
CMC_return_code
(*cmc_get_root_handle_W)(
       CMC_session_id          session,
       CMC_object_handle       *root_object_handle,
       CMC_extension           *get_root_handle_extensions
);

/* IDENTIFIER TO NAME */
CMC_return_code
(*cmc_identifier_to_name_W)(
       CMC_id                  identifier,
       CMC_name                *name,
       CMC_extension           *identifier_to_name_extensions
);
```

```
/* LIST CONTAINED PROPERTIES */
CMC_return_code
(*cmc_list_contained_properties_W)(
       CMC_cursor_handle       cursor,
       CMC_sint32              *number_object,
       CMC_sint32              *number_properties,
       CMC_id                  *property_ids,
       CMC_property            ***properties,
       CMC_extension           *list_contained_properties_extensions
);

/* LIST NUMBER MATCHED */
CMC_return_code
(*cmc_list_number_matched_W)(
       CMC_cursor_handle       *cursor,
       CMC_uint32              *number_matches,
       CMC_extension           *list_number_matched_extensions
);

/* LIST OBJECTS */
CMC_return_code
(*cmc_list_objects_W)(
       CMC_cursor_handle       *cursor,
       CMC_sint32              *number_objects,
       CMC_object_handle       **objects,
       CMC_extension           *list_objects_extensions
);

/* LIST PROPERTIES */
CMC_return_code
(*cmc_list_properties_W)(
       CMC_object_handle       *object,
       CMC_uint32              *number_properties,
       CMC_id                  **property_ids,
       CMC_extension           *list_properties_extensions
);

/* NAME TO IDENTIFIER */
CMC_return_code
(*cmc_name_to_identifier_W)(
       CMC_name                name,
       CMC_id                  *identifier,
       CMC_extension           *name_to_identifier_extensions
);

/* OPEN CURSOR */
CMC_return_code
(*cmc_open_cursor_W)(
       CMC_object_handle       object,
       CMC_cursor_restriction  *restrictions,
       CMC_uint32              number_sort_orders,
       CMC_cursor_sort_key     *sort_keys,
       CMC_cursor_handle       *cursor,
       CMC_extension           *open_cursor_extensions
);

/* OPEN OBJECT HANDLE */
CMC_return_code
(*cmc_open_object_handle_W)(
       CMC_session_id          session,
       CMC_id                  object_class,
       CMC_object_handle       *new_object,
       CMC_extension           *open_object_handle_extensions
);

/* READ CURSOR */
CMC_return_code
(*cmc_read_cursor_W)(
       CMC_cursor_handle       *cursor,
       CMC_uint32              *position_numerator,
       CMC_uint32              *position_denominator,
       CMC_extension           *read_cursor_extensions
);
```

```
/* READ PROPERTIES */
CMC_return_code
(*cmc_read_properties_W)(
       CMC_object_handle       object,
       CMC_uint32              *number_properties,
       CMC_id                  *property_ids,
       CMC_property            **properties,
       CMC_extension           *read_properties_extensions
);

/* READ PROPERTY COSTS */
CMC_return_code
(*cmc_read_property_costs_W)(
       CMC_object_handle       object,
       CMC_uint32              *number_properties,
       CMC_id                  *property_ids,
       CMC_enum                **costs,
       CMC_extension           *read_property_costs_extensions
);

/* RESTORE OBJECT */
CMC_return_code
(*cmc_restore_object_W)(
       CMC_object_handle       container,
       CMC_string              file_specification,
       CMC_object_handle       *restored_object,
       CMC_flags               restore_flags,
       CMC_extension           *restore_object_extensions
);

/* SAVE OBJECT */
CMC_return_code
(*cmc_save_object_W)(
       CMC_object_handle       object,
       CMC_string              file_specification,
       CMC_flags               save_flags,
       CMC_extension           *save_object_extensions
);

/* SEND MESSAGE OBJECT */
CMC_return_code
(*cmc_send_message_object_W)(
       CMC_object_handle       message_to_send,
       CMC_extension           *send_message_object_extensions
);

/* UPDATE CURSOR POSITION */
CMC_return_code
(*cmc_update_cursor_position_W)(
       CMC_cursor_handle       *cursor,
       CMC_uint32              position_numerator,
       CMC_uint32              position_denominator,
       CMC_extension           *update_cursor_position_extensions
);

/* UPDATE CURSOR POSITION WITH SEED */
CMC_return_code
(*cmc_update_cursor_position_with_seed_W)(
       CMC_cursor_handle       cursor,
       CMC_object_handle       seed,
       CMC_extension           *update_cursor_position_with_seed_extensions
);

/* CHECK EVENT */
CMC_return_code
(*cmc_check_event_W)(
       CMC_session_id          session,
       CMC_event               event_type,
       CMC_uint32              minimum_timeout,
       CMC_buffer              check_event_data,
       CMC_buffer              *callback_data,
       CMC_extension           *check_event_extensions
);
```

```
/* REGISTER EVENT */
CMC_return_code
(*cmc_register_event_W)(
       CMC_session_id          session,
       CMC_event               event_type,
       CMC_callback            callback,
       CMC_buffer              register_data,
       CMC_extension           *register_event_extensions
);

/* UNREGISTER EVENT */
CMC_return_code
(*cmc_unregister_event_W)(
       CMC_session_id          session,
       CMC_flags               event_type,
       CMC_callback            callback,
       CMC_buffer              unregister_data,
       CMC_extension           *unregister_event_extensions
);

/* CALL CALLBACKS */
CMC_return_code
(*cmc_call_callbacks_W)(
       CMC_session_id          session,
       CMC_event               event_type,
       CMC_extension           *call_callbacks_extensions
);

/* EXPORT STREAM */
CMC_return_code
(*cmc_export_stream_W)(
       CMC_stream_handle       stream,
       CMC_string              file_specification,
       CMC_uint32              count,
       CMC_flags               export_flags,
       CMC_extension           *export_stream_extensions
);

/* IMPORT FILE TO STREAM */
CMC_return_code
(*cmc_import_file_to_stream_W)(
       CMC_stream_handle       stream,
       CMC_string              file_specification,
       CMC_uint32              file_offset,
       CMC_extension           *import_file_to_stream_extensions
);

/* OPEN STREAM */
CMC_return_code
(*cmc_open_stream_W)(
       CMC_object_handle       object,
       CMC_property            *property,
       CMC_enum                operation,
       CMC_stream_handle       *stream,
       CMC_extension           *open_stream_extensions
);

/* READ STREAM */
CMC_return_code
(*cmc_read_stream_W)(
       CMC_stream_handle       stream,
       CMC_uint32              *count,
       CMC_buffer              content_information,
       CMC_extension           *read_stream_extensions
);

/* SEEK STREAM */
CMC_return_code
(*cmc_seek_stream_W)(
       CMC_stream_handle       stream,
       CMC_enum                operation,
       CMC_uint32              *location,
       CMC_extension           *seek_stream_extensions
);
```

```
        /* WRITE STREAM */
        CMC_return_code
        (*cmc_write_stream_W)(
                CMC_stream_handle       *stream,
                CMC_uint32              *count,
                CMC_buffer              *content_information,
                CMC_extension           *write_stream_extensions
        );

        /* GET LAST ERROR */
        CMC_return_code
        (*cmc_get_last_error_W)(
                CMC_session_id          session,
                CMC_object_handle       objRef,
                CMC_string              *error_buffer,
                CMC_extension           *get_last_error_extensions
        );

} CMC_dispatch_table;

/* BIND IMPLEMENTATION */
CMC_return_code
cmc_bind_implementation_W(
        CMC_guid                  implementation_name,
        CMC_dispatch_table        **dispatch_table,
        CMC_extension             *cmc_bind_extensions
);

/* UNBIND IMPLEMENTATION */
CMC_return_code
cmc_unbind_implementation_W(
        CMC_guid                  implementation_name,
        CMC_extension             *cmc_unbind_implementation_extensions
);

#endif
typedef CMC_return_code (*CMC_P_BIND_IMPLEMENTATION)(
        CMC_guid                  implementation_name,
        CMC_dispatch_table        **dispatch_table,
        CMC_extension             *cmc_bind_extensions
);
typedef CMC_return_code (*CMC_P_UNBIND_IMPLEMENTATION)(
        CMC_guid                  implementation_name,
        CMC_extension             *cmc_unbind_extensions
);

/* Function Constants */

/* SEND */
#define CMC_SEND_UI_REQUESTED                   ((CMC_flags) 1)

/* SEND DOCUMENT */
#define CMC_FIRST_ATTACH_AS_TEXT_NOTE           ((CMC_flags) 2)

/* ACT ON */
#define CMC_ACT_ON_EXTENDED                     ((CMC_enum)  0)
#define CMC_ACT_ON_DELETE                       ((CMC_enum)  1)

/* LIST */
#define CMC_LIST_UNREAD_ONLY                    ((CMC_flags) 1)
#define CMC_LIST_MSG_REFS_ONLY                  ((CMC_flags) 2)
#define CMC_LIST_COUNT_ONLY                     ((CMC_flags) 4)

#define CMC_LENGTH_UNKNOWN                      0xFFFFFFFF

/* READ */
#define CMC_DO_NOT_MARK_AS_READ                 ((CMC_flags) 1)
#define CMC_MSG_AND_ATT_HDRS_ONLY               ((CMC_flags) 2)
#define CMC_READ_FIRST_UNREAD_MESSAGE           ((CMC_flags) 4)

/* LOOKUP */
#define CMC_LOOKUP_RESOLVE_PREFIX_SEARCH        ((CMC_flags) 1)
#define CMC_LOOKUP_RESOLVE_IDENTITY             ((CMC_flags) 2)
#define CMC_LOOKUP_RESOLVE_UI                   ((CMC_flags) 4)
#define CMC_LOOKUP_DETAILS_UI                   ((CMC_flags) 8)
#define CMC_LOOKUP_ADDRESSING_UI                ((CMC_flags) 16)
```

```
/* LOGOFF */
#define CMC_LOGOFF_UI_ALLOWED                  ((CMC_flags) 1)

/* LOGON */
#define CMC_VERSION                            ((CMC_uint16) 100)

/* QUERY CONFIGURATION ENUMS */
#define CMC_CONFIG_CHARACTER_SET               ((CMC_enum) 1)
#define CMC_CONFIG_LINE_TERM                   ((CMC_enum) 2)
#define CMC_CONFIG_DEFAULT_SERVICE             ((CMC_enum) 3)
#define CMC_CONFIG_DEFAULT_USER                ((CMC_enum) 4)
#define CMC_CONFIG_REQ_PASSWORD                ((CMC_enum) 5)
#define CMC_CONFIG_REQ_SERVICE                 ((CMC_enum) 6)
#define CMC_CONFIG_REQ_USER                    ((CMC_enum) 7)
#define CMC_CONFIG_UI_AVAIL                    ((CMC_enum) 8)
#define CMC_CONFIG_SUP_NOMKMSGREAD             ((CMC_enum) 9)
#define CMC_CONFIG_SUP_COUNTED_STR             ((CMC_enum) 10)
#define CMC_CONFIG_VER_IMPLEM                  ((CMC_enum) 11)
#define CMC_CONFIG_VER_SPEC                    ((CMC_enum) 12)

/* CONFIG LINE TERM ENUM */
#define CMC_LINE_TERM_CRLF                     ((CMC_enum) 0)
#define CMC_LINE_TERM_CR                       ((CMC_enum) 1)
#define CMC_LINE_TERM_LF                       ((CMC_enum) 2)

/* CONFIG REQUIRED LOGON PARAMETER ENUM */
#define CMC_REQUIRED_NO                        ((CMC_enum) 0)
#define CMC_REQUIRED_YES                       ((CMC_enum) 1)
#define CMC_REQUIRED_OPT                       ((CMC_enum) 2)

/* CREATE DERIVED MESSAGE OBJECT */
#define CMC_DERIVED_ACTION_FORWARD             ((CMC_enum) 0)
#define CMC_DERIVED_ACTION_REPLY_ORIGINATOR    ((CMC_enum) 1)
#define CMC_DERIVED_ACTION_REPLY_ALL           ((CMC_enum) 2)

/* READ PROPERTY COSTS */
#define CMC_COST_UNDETERMINED                  ((CMC_enum) 0)
#define CMC_COST_NONE                          ((CMC_enum) 1)
#define CMC_COST_MINOR                         ((CMC_enum) 2)
#define CMC_COST_MAJOR                         ((CMC_enum) 3)

/* RESTORE OBJECT FLAGS */
#define CMC_RESTORE_OBJECT_OVERWRITE           ((CMC_flags) 1)

/* SAVE OBJECT FLAGS */
#define CMC_SAVE_OBJECT_OVERWRITE              ((CMC_flags) 1)
#define CMC_SAVE_OBJECT_NOCREATE               ((CMC_flags) 2)

/* EXPORT STREAM */
#define CMC_EXPORT_STREAM_OVERWRITE            ((CMC_flags) 1)
#define CMC_EXPORT_STREAM_NOCREATE             ((CMC_flags) 2)
#define CMC_EXPORT_STREAM_APPEND               ((CMC_flags) 3)

/* OPEN STREAM */
#define CMC_OPEN_READ                          ((CMC_enum) 0)
#define CMC_OPEN_WRITE                         ((CMC_enum) 1)

/* SEEK STREAM */
#define CMC_SEEK_BEGINNING                     ((CMC_enum) 0)
#define CMC_SEEK_END                           ((CMC_enum) 1)
#define CMC_SEEK_CURRENT_POSITION              ((CMC_enum) 2)

/* DEFINED OBJECT ID'S FOR CHARACTER SETS */
#define CMC_CHAR_CP437                         "1 2 840 113556 3 2 437"
#define CMC_CHAR_CP850                         "1 2 840 113556 3 2 850"
#define CMC_CHAR_CP1252                        "1 2 840 113556 3 2 1252"
#define CMC_CHAR_ISTRING                       "1 2 840 113556 3 2 0"
#define CMC_CHAR_UNICODE                       "1 2 840 113556 3 2 1"

/* RETURN CODE FLAGS */
#define CMC_ERROR_DISPLAYED                    ((CMC_return_code) 0x00008000)
#define CMC_ERROR_RSV_MASK                     ((CMC_return_code) 0x0000FFFF)
#define CMC_ERROR_IMPL_MASK                    ((CMC_return_code) 0xFFFF0000)
```

```
/* RETURN CODES */
#define CMC_SUCCESS                          ((CMC_return_code) 0)
#define CMC_E_AMBIGUOUS_RECIPIENT            ((CMC_return_code) 1)
#define CMC_E_ATTACHMENT_NOT_FOUND           ((CMC_return_code) 2)
#define CMC_E_ATTACHMENT_OPEN_FAILURE        ((CMC_return_code) 3)
#define CMC_E_ATTACHMENT_READ_FAILURE        ((CMC_return_code) 4)
#define CMC_E_ATTACHMENT_WRITE_FAILURE       ((CMC_return_code) 5)
#define CMC_E_COUNTED_STRING_UNSUPPORTED     ((CMC_return_code) 6)
#define CMC_E_DISK_FULL                      ((CMC_return_code) 7)
#define CMC_E_FAILURE                        ((CMC_return_code) 8)
#define CMC_E_INSUFFICIENT_MEMORY            ((CMC_return_code) 9)
#define CMC_E_INVALID_CONFIGURATION          ((CMC_return_code) 10)
#define CMC_E_INVALID_ENUM                   ((CMC_return_code) 11)
#define CMC_E_INVALID_FLAG                   ((CMC_return_code) 12)
#define CMC_E_INVALID_MEMORY                 ((CMC_return_code) 13)
#define CMC_E_INVALID_MESSAGE_PARAMETER      ((CMC_return_code) 14)
#define CMC_E_INVALID_MESSAGE_REFERENCE      ((CMC_return_code) 15)
#define CMC_E_INVALID_PARAMETER              ((CMC_return_code) 16)
#define CMC_E_INVALID_SESSION_ID             ((CMC_return_code) 17)
#define CMC_E_INVALID_UI_ID                  ((CMC_return_code) 18)
#define CMC_E_LOGON_FAILURE                  ((CMC_return_code) 19)
#define CMC_E_MESSAGE_IN_USE                 ((CMC_return_code) 20)
#define CMC_E_NOT_SUPPORTED                  ((CMC_return_code) 21)
#define CMC_E_PASSWORD_REQUIRED              ((CMC_return_code) 22)
#define CMC_E_RECIPIENT_NOT_FOUND            ((CMC_return_code) 23)
#define CMC_E_SERVICE_UNAVAILABLE            ((CMC_return_code) 24)
#define CMC_E_TEXT_TOO_LARGE                 ((CMC_return_code) 25)
#define CMC_E_TOO_MANY_FILES                 ((CMC_return_code) 26)
#define CMC_E_TOO_MANY_RECIPIENTS            ((CMC_return_code) 27)
#define CMC_E_UNABLE_TO_NOT_MARK_AS_READ     ((CMC_return_code) 28)
#define CMC_E_UNRECOGNIZED_MESSAGE_TYPE      ((CMC_return_code) 29)
#define CMC_E_UNSUPPORTED_ACTION             ((CMC_return_code) 30)
#define CMC_E_UNSUPPORTED_CHARACTER_SET      ((CMC_return_code) 31)
#define CMC_E_UNSUPPORTED_DATA_EXT           ((CMC_return_code) 32)
#define CMC_E_UNSUPPORTED_FLAG               ((CMC_return_code) 33)
#define CMC_E_UNSUPPORTED_FUNCTION_EXT       ((CMC_return_code) 34)
#define CMC_E_UNSUPPORTED_VERSION            ((CMC_return_code) 35)
#define CMC_E_USER_CANCEL                    ((CMC_return_code) 36)
#define CMC_E_USER_NOT_LOGGED_ON             ((CMC_return_code) 37)
#define CMC_E_INVALID_OBJECT_HANDLE          ((CMC_return_code) 38)
#define CMC_E_PROPERTY_ID_NOT_FOUND          ((CMC_return_code) 39)
#define CMC_E_INVALID_CURSOR_HANDLE          ((CMC_return_code) 40)
#define CMC_E_REQUIRED_PROPS_MISSING         ((CMC_return_code) 41)
#define CMC_E_INVALID_SOURCE_OBJECT          ((CMC_return_code) 42)
#define CMC_E_INVALID_CONTAINER_OBJECT       ((CMC_return_code) 43)
#define CMC_E_UNRECOGNIZED_IDENTIFIER        ((CMC_return_code) 44)
#define CMC_E_INVALID_PROPERTY_NAME          ((CMC_return_code) 45)
#define CMC_E_INVALID_RESTRICTION            ((CMC_return_code) 46)
#define CMC_E_UNSUPPORTED_KEYS               ((CMC_return_code) 47)
#define CMC_E_INVALID_STREAM_HANDLE          ((CMC_return_code) 48)
#define CMC_E_INVALID_FILE_OFFSET            ((CMC_return_code) 49)
#define CMC_E_INVALID_PROPERTY_ID            ((CMC_return_code) 50)
#define CMC_E_NO_MORE_BYTES_TO_WRITE         ((CMC_return_code) 51)
#define CMC_E_NAME_NOT_FOUND                 ((CMC_return_code) 52)
#define CMC_E_ID_NOT_FOUND                   ((CMC_return_code) 53)
#define CMC_E_TOO_MANY_CONTENT_ITEMS         ((CMC_return_code) 54)
#define CMC_E_BIND_FAILURE                   ((CMC_return_code) 55)
#define CMC_E_UNBIND_FAILURE                 ((CMC_return_code) 56)
#define CMC_E_INVALID_EVENT                  ((CMC_return_code) 57)
#define CMC_E_CALLBACK_NOT_SUPPORTED         ((CMC_return_code) 58)
#define CMC_E_ACCESS_DENIED                  ((CMC_return_code) 59)
#define CMC_E_INVALID_FILE_SPECIFICATION     ((CMC_return_code) 60)
#define CMC_E_PROPERTY_NAME_NOT_FOUND        ((CMC_return_code) 61)
#define CMC_E_INVALID_FUNCTION_EXT           ((CMC_return_code) 62)
#define CMC_E_FUNCTION_INTERRUPTED           ((CMC_return_code) 63)

#ifdef __cplusplus
}       /* extern "C" */
#endif

#endif  /* _XCMC_H */
```

Anexo B

Extensiones de vendedores de CMC

B.1 Extensiones de vendedores de CMC

Esta Recomendación permite extensiones de vendedores en muchas áreas. Los vendedores pueden añadir extensiones a ciertas estructuras de datos CMC y cada función contiene un parámetro para el uso de extensiones funcionales. Los vendedores pueden definir nuevas clases de objetos CMC, ampliar el conjunto de propiedades asociadas con una clase de objeto, añadir valores enumerados adicionales y asociar un identificador de implementación CMC a una implementación. Además, una parte de la funcionalidad descrita en esta Recomendación se ha definido utilizando extensiones comunes definidas en esta Recomendación, para asegurar la retrocompatibilidad con la CMC 1.0 de la Asociación XAPIA. Futuras versiones de esta Recomendación podrán también definir ulteriores conjuntos de extensiones. Por esta razón, es importante tener un conjunto de directrices para la denominación y definición de extensiones. Estas directrices se indican a continuación:

1) Se facilitarán a los vendedores o grupos de vendedores gamas de códigos de ítem (item_code) de extensiones en bloques de 256, para la creación de conjuntos de extensiones. Un vendedor o grupo de vendedores puede obtener más de una gama de códigos de ítem si son necesarias para el conjunto de extensiones. El identificador de conjunto de extensiones para todas las gamas de códigos de ítem de los conjuntos será el que ocupe la primera posición en el primer bloque facilitado. Este identificador de conjunto de extensiones se utiliza para interrogar al servicio sobre el soporte de un determinado conjunto de extensiones.

   Por ejemplo, los bloques de extensiones para el grupo de vendedores X pueden ser 0x00000400, 0x00000900 y 0x00004300 y el identificador del conjunto de extensiones sería 0x00000400 si ese fue el primer bloque asignado al vendedor. Las aplicaciones interrogarían a un servicio para averiguar si soporta el conjunto de extensiones 0x00000400 para las extensiones de este grupo de vendedores.

2) Un conjunto de extensiones tendrá también asignado un prefijo específico para uso en los nombres de todas las extensiones que forman el conjunto de extensiones. El formato de este prefijo será:

CMC_XS_[id de vendedor]	para el identificador del conjunto de extensiones
CMC_X_[id de vendedor]_[nombre de extensión]	para los códigos de ítem de las extensiones que forman el conjunto de extensiones

   En el mencionado ejemplo del grupo de vendedores X, si su identificador de vendedor fuera CX, definiría sus extensiones en la forma siguiente:

```
#define CMC_XS_CX        0x00000400
#define CMC_X_CX_EXT1    0x00000401
#define CMC_X_CX_EXT2    0x00000402
```

3) A los conjuntos de extensiones definidos por esta Recomendación se asignará un número y un prefijo de conjunto de extensiones tomados de la Asociación X.400 API. Los implementadores pueden también obtener un prefijo de conjunto de extensiones y un bloque de códigos de extensión, de la Asociación X.400 API, solicitándolos por escrito. En el anexo B se indican números predefinidos de conjuntos de extensiones. El soporte de diferentes conjuntos de extensiones se indica por la configuración de la implementación CMC y puede averiguarse por la función **cmc_query_configuration()** utilizando la extensión CMC_X_COM_SUPPORT_EXT.

4) Se ha asignado también un valor de conjunto de extensiones de BILATERAL. Cualquier implementación puede definir extensiones dentro del conjunto BILATERAL. No es necesario registrar el número de un conjunto de extensiones. Este conjunto se proporciona para que los implementadores puedan definir extensiones sin necesidad de un registro formal. Por el hecho de existir esta libertad, las extensiones de diferentes vendedores pueden discordar unas de otras y afectar la portabilidad de las aplicaciones y las posibilidades de ubicar en un mismo lugar diferentes implementaciones CMC. El prefijo para estas extensiones será CMC_X_BLT_ y el correspondiente identificador de conjunto es CMC_XS_BLT.

5) Muchos objetos se denominan mediante identificadores globalmente únicos [brevemente GUID (GUID, *globally unique identifiers*)]. Los vendedores pueden asignar GUIDS con nombres específicos del vendedor. Cuando un vendedor registra un conjunto de extensiones, se le asigna también una rama en el espacio de nombres de GUID:

   –//XAPIA/CMC20/OBJECT CLASS/VENDOR [id de vendedor]//NONSGML [nombre de ext.]//EN para clases de objetos,

   –//XAPIA/CMC20/PROPERTY/VENDOR [id de vendedor]//NONSGML [nombre de ext.]//EN para clases de propiedades,

–//XAPIA/CMC20/CONTENT TYPE/VENDOR [id de vendedor]//NONSGML [nombre de ext.]//EN para tipos de contenido,

–//XAPIA/CMC20/CHARSET/VENDOR [id de vendedor]//NONSGML [nombre de ext.]//EN para juegos de caracteres, y

–//XAPIA/CMC20/ENCODING TYPE/VENDOR [id de vendedor]//NONSGML [nombre de ext.]//EN para tipos de codificación.

NOTA – La especificación de extensiones de vendedor no implica que las extensiones se transportarán inalteradas a través de los protocolos de mensajería y las cabeceras. Los detalles de las limitaciones, debidas a los protocolos y las cabeceras, relacionadas con estas extensiones, deben especificarse en los manuales de los vendedores.

6) Los vendedores pueden también extender los valores enumerados especificados en esta Recomendación. Los valores enumerados de 0 a 512 están reservados para esta Recomendación. El vendedor puede reutilizar valores de código de ítem para valores enumerados. Para la definición de constantes asociadas con estos valores, el vendedor debe utilizar el prefijo CMC_X_[id de vendedor]_[enum], asegurándose de que las constantes no son discordantes con los nombres de las extensiones.

Para minimizar los problemas de portatibilidad, se insta a los implementadores a especificar las extensiones de la manera más general posible, y presentar estas extensiones como propuestas de adiciones al conjunto de extensiones definidas por CMC. Mediante este proceso, el conjunto de interfaces API de CMC evolucionará positivamente de tal manera que la portabilidad será cada vez mayor.

B.1.1 Extensiones de funciones

B.1.1.1 CMC_X_COM_SUPPORT_EXT

Esta extensión la utilizan las aplicaciones de clientes para interrogar a la implementación CMC sobre las extensiones que soporta. Puede utilizarse antes de establecer una sesión para obtener una información previa sobre el soporte antes del establecimiento de la sesión. Cuando esta extensión se utiliza con **cmc_logon()**, indicará también las extensiones de datos que el cliente desea añadir a las estructuras de datos para la sesión.

NOTA – Algunas implementaciones pueden soportar diferentes extensiones según el servicio utilizado por la aplicación de cliente para crear la sesión, por lo que se recomienda utilizar esta extensión en el momento del establecimiento de la sesión, para verificar el soporte de la extensión.

Si una implementación CMC soporta alguna extensión, tiene que soportar también ésta.

USADA POR

```
cmc_query_config()
cmc_logon()
```

ENTRADA

extension_flags (banderas de extensión)

Todas las banderas CMC son válidas. No se definen más banderas.

item_data (datos de ítem)

Cuenta de los ítems en una matriz señalada por la referencia de ítem.

item_reference (referencia de ítem)

Puntero al primer elemento de una matriz de estructuras que indica las extensiones que la aplicación solicita sean soportadas por la implementación. La declaración en lenguaje C para esta estructura es la siguiente:

```
typedef struct {
   CMC_uint32              item_code;
   CMC_flags               flags;
} CMC_X_COM_support;
```

El item_code en la estructura se fija al código de ítem de la extensión sobre la cual la aplicación está interrogando al servicio. Las extensiones pueden pertenecer a conjuntos de extensiones o ser extensiones individuales. Un código de ítem de valor nulo no se tendrá en cuenta. Las banderas para las estructuras que se utilizan en entrada son:

CMC_X_COM_SUP_EXCLUDE – Excluye este ítem cuando se decide si la implementación soporta un conjunto de extensiones. Al establecerse la sesión, no debe añadirse este ítem a estructuras para esta sesión, incluso si en otras inscripciones se solicita que se añada. Esta bandera sólo se utiliza con conjuntos de extensiones.

SALIDA

extension_flags

inalteradas

item_data

inalterados

item_reference

La implementación fija las banderas en las estructuras de modo que indiquen soporte de la extensión. Estas banderas no se fijarán si CMC_X_COM_SUP_EXCLUDE se fijó en entrada. Los valores posibles se indican a continuación.

CMC_X_COM_SUPPORTED – La extensión para este ítem de código está soportada. Si es una extensión de datos y se pasa al establecer la sesión, se incluirá en las estructuras utilizadas para esta sesión. En el caso de conjuntos de extensiones, las extensiones requeridas de funciones y de datos en el conjunto están soportadas.

CMC_X_COM_NOT_SUPPORTED – El ítem de código no está soportado. En el caso de conjuntos de extensiones, no todas las extensiones requeridas de funciones y de datos están soportadas. Si se trata de una extensión de datos o de un conjunto de extensiones que contiene extensiones de datos, los datos no serán añadidos a estructuras para esta sesión.

CMC_X_COM_DATA_EXT_SUPPORTED – Para conjuntos de extensiones solamente. Puede ser retornada por la implementación para indicar que todas las extensiones de datos requeridas para el conjunto están soportadas, pero no todas las extensiones de funciones requeridas. Al igual que CMC_X_COM_SUPPORTED, si ésta se retorna en la llamada a **cmc_logon()**, las extensiones de datos se incluirán con las estructuras de datos para esta sesión.

CMC_X_COM_FUNC_EXT_SUPPORTED – Para conjuntos de extensiones solamente. Puede ser retornada por la implementación para indicar que todas las extensiones de funciones requeridas para el conjunto están soportadas, pero no todas las extensiones de datos requeridas. A diferencia de CMC_X_COM_SUPPORTED, si ésta se retorna en la llamada de **cmc_logon()**, las extensiones de datos disponibles NO se incluirán con las estructuras de datos para esta sesión y no será necesario solicitarlas explícitamente.

B.1.1.2 CMC_X_UI_ID_EXT

Esta extensión la utilizan aplicaciones de clientes para especificar información de interfaz de usuario propia de la plataforma, para las funciones CMC. La implementación CMC puede utilizar la información de interfaz de usuario para presentar diálogos de usuario con el fin de resolver argumentos adicionales para la llamada CMC o cualquier otra cuestión que surja cuando el servicio realice la función. Por ejemplo, en un entorno basado en ventanas, ésta podría ser el asa de la ventana progenitora para la aplicación llamante.

NOTA – Las implementaciones CMC no están obligadas a proporcionar la interfaz de usuario, y la provisión de una interfaz de usuario para una característica no implica necesariamente que habrá una interfaz de usuario disponible para todas las características de la CMC.

Los códigos de error generados como resultado de la utilización de esta extensión de función se retornarán como códigos de error según el proceso usual para los códigos de retorno.

USADA POR

Todas las funciones de la CMC completa.

ENTRADA

extension_flags

Todas las banderas CMC son válidas. Las banderas no especificadas deben pasarse siempre como cero. Esta función utiliza la siguiente bandera adicional:

CMC_X_ERROR_UI_ALLOWED

CMC_X_ERROR_UI_ALLOWED – Se fija si la función puede visualizar una UI al encontrar errores subsanables. Si no se fija, la función no puede visualizar una UI y retornará un código de error. Esta bandera es válida para todas las funciones CMC que soportan esta extensión.

item data

cero

item_reference

Un puntero a un identificador para una interfaz de usuario (por ejemplo una ventana de diálogo) utilizada para resolver cualquier cuestión que, de no ser resuelta, podría generar un error, e interrogaciones al usuario para que suministre la información adicional requerida.

SALIDA

extension_flags

inalteradas

item_data

inalterados

item_reference

inalterada

B.1.1.3 CMC_X_COM_CONFIG_DATA

Obtiene todos los valores disponibles con **cmc_query_configuration()**, en una estructura.

USADA POR

cmc_query_configuration()

ENTRADA

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

item data

cero

item_reference

NULL

SALIDA

extension_flags

Se fijará CMC_EXT_OUTPUT si se retorna con éxito una estructura.

item_data

inalterados

item_reference

Puntero a una estructura que contiene toda la información obtenida por la llamada a la función de indagación de la configuración. La declaración en lenguaje C para esta estructura es la siguiente:

```
typedef struct {
   CMC_uint16              ver_spec;
   CMC_uint16              ver_implem;
   CMC_object_identifier   *character_set;
   CMC_enum                line_term;
   CMC_string              default_service;
   CMC_string              default_user;
```

```
    CMC_enum                req_password;
    CMC_enum                req_service;
    CMC_enum                req_user;
    CMC_boolean             ui_avail;
    CMC_boolean             sup_nomkmsgread;
    CMC_boolean             sup_counted_str;
} CMC_X_COM_configuration;
```

La definición de cada uno de los miembros de la estructura corresponde a los datos retornados a través del argumento referencia por la función **cmc_query_configuration()** para el valor de nombre similar del argumento ítem. Esta estructura debe liberarse con una llamada a **cmc_free()**.

B.1.1.4 CMC_X_COM_PROPERTY_HINTS

Esta extensión de función avisa anticipadamente a **cmc_list_objects()** sobre las propiedades que el llamante necesitará próximamente. Este aviso previo permitirá a las implementaciones optimizar la extracción de propiedades obteniendo de una sola vez todas las propiedades sobre las que fueron avisadas.

USADA POR

cmc_list_objects()

ENTRADA

extension_flags

Todas las banderas CMC son válidas. Las banderas no especificadas deben pasarse siempre como cero. No se definen banderas adicionales.

item data

El número de nombres de propiedades CMC en la matriz de estructuras señalada por item_reference.

item_reference

Un puntero a una matriz de nombres de propiedades CMC. Estos identificadores especifican las propiedades sobre las que se avisa.

SALIDA

extension_flags

inalteradas

item_data

inalterados

item_reference

inalterada

B.1.1.5 CMC_X_COM_CAN_SEND_RECIP

Comprueba si el servicio de mensajes está listo para enviar el mensaje al recibiente especificado.

USADA POR

cmc_look_up()

ENTRADA

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

item data

cero

item_reference

NULL

En entrada, el parámetro recipient_in de **cmc_look_up()** contiene el recibiente sobre el cual se interrogará al servicio. La extensión sólo mirará al primer recibiente, si se han pasado más de uno.

SALIDA

extension_flags

inalteradas

item_data

Se fija a CMC_X_COM_NOT_READY si no hay un transporte disponible para este tipo de recibiente, a CMC_X_COM_READY si se puede enviar el mensaje inmediatamente al recibiente, y a CMC_X_COM_DEFER si el mensaje se aceptará, pero se aplazará hasta que haya un transporte disponible.

item_reference

inalterada

B.1.1.6 CMC_X_COM_SAVE_MESSAGE

Guarda una estructura de mensaje en el apartado de entrada

USADA POR

cmc_act_on()

ENTRADA

extension_flags

Tienen que contener CMC_EXT_REQUIRED para indicar que se debe ejecutar la acción de guardar y no la acción de suprimir. Todas las banderas CMC son válidas. No se definen más banderas.

item_data

cero

item_reference

Puntero a la estructura de mensaje que se va a guardar en el apartado de entrada. La implementación CMC fijará la bandera CMC_MSG_UNSENT de este mensaje para indicar que no se ha enviado.

En entrada, el parámetro operación de **cmc_act_on()** debe fijarse a CMC_ACT_ON_EXTENDED para indicar que la operación está contenida en las extensiones.

SALIDA

extension_flags

CMC_EXT_OUTPUT se fijará si un mensaje se guarda correctamente y se retorna la referencia de mensaje.

item_data

inalterados

item_reference

Puntero a la referencia de mensaje que señala el mensaje guardado en el apartado de entrada. Este puntero debe ser liberado por **cmc_free()**.

B.1.1.7 CMC_X_COM_SENT_MESSAGE

Retorna una estructura de mensaje que contiene toda la información para el mensaje que se acaba de enviar. Es útil para obtener información contenida en el conjunto de estructuras de mensaje con UI, más bien que por la aplicación llamante.

USADA POR

cmc_send()

ENTRADA

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

item data

cero

item_reference

NULL

SALIDA

extension_flags

CMC_EXT_OUTPUT se fijará si la referencia de ítem contiene un puntero a un mensaje.

item_data

inalterados

item_reference

Puntero a una estructura de mensaje que contiene toda la información para el mensaje que acaba de enviarse. Este puntero debe liberarse por **cmc_free()**.

B.1.1.8 CMC_X_COM_PROP_STATUS

Esta extensión de función indica que la operación realizada tiene que retornar el estado de cada propiedad. Un error que se produce cuando se intenta modificar o suprimir una propiedad se designa por un error de propiedad. Si una operación que influye en múltiples propiedades encuentra problemas que impiden tratar algunas de esas propiedades, esta extensión permite al llamante recibir informes sobre los problemas de propiedad.

USADA POR

cmc_add_properties()

cmc_delete_properties()

ENTRADA

extension_flags

Todas las banderas CMC son válidas. Las banderas no especificadas deben pasarse siempre como cero. No se definen banderas adicionales.

item data

cero

item_reference

NULL

SALIDA

extension_flags

Se fija la bandera CMC_EXT_OUTPUT si se informa de algún problema de propiedad.

item_data

Cuenta de los ítems en la matriz a que apunta la referencia de ítem. Es cero si no se informan problemas de propiedad.

item_reference

Puntero a una matriz de estructuras que indica los problemas de propiedad informados. La declaración en lenguaje C para esta estructura es:

```
typedef struct {
   CMC_uint32              index;
   CMC_id                  id;
   CMC-return_code         error_code;
} CMC_X_COM_prop_problem;
```

donde:

- index especifica el índice de la propiedad que interviene en las *propiedades* o en la matriz de *identificadores de propiedades* de la función, en entrada;
- id identifica la propiedad que interviene;
- error_code especifica el error encontrado cuando se procesa la petición de esa propiedad.

La matriz la asigna el servicio y debe liberarse por una llamada a **cmc_free()**.

Cuando esta extensión informa de problemas de propiedad, la función retorna el código de error CMC_E_PROPERTY_PROBLEMS. En este caso se puede suponer que toda propiedad con relación a la cual no se haya informado de un problema ha sido procesada correctamente.

ERRORES

CMC_E_DISK_FULL
CMC_E_FAILURE
CMC_E_INSUFFICIENT_MEMORY
CMC_E_INVALID_ENUM
CMC_E_INVALID_MEMORY
CMC_E_REQUIRED_PROPS_MISSING
CMC_E_SERVICE_UNAVAILABLE
CMC_E_TEXT_TOO_LARGE
CMC_E_UNRECOGNIZED_MESSAGE_TYPE
CMC_E_UNSUPPORTED_ACTION
CMC_E_UNSUPPORTED_CHARACTER_SET
CMC_E_UNSUPPORTED_FLAG

B.1.2 Extensiones de datos

B.1.2.1 CMC_X_COM_TIME_RECEIVED

Extensión de datos para una estructura de tiempo (u hora) para el tiempo (u hora) de entrega del mensaje.

En el momento de establecimiento de la sesión, el código de ítem se pasa en la matriz CMC_X_COM_SUPPORT_EXT para indicar que este miembro de datos debe añadirse a la estructura de mensaje y a la estructura de sumario de mensaje durante la sesión.

USADA POR

CMC_message
CMC_message_summary

ENTRADA

Esta extensión se ignora en la entrada de una estructura de mensaje.

SALIDA

extension_flags

NULL

item_data

cero

item_reference

Puntero a una estructura de tiempo (u hora) que indica el tiempo (u hora) de recepción del mensaje. Para más información, véase la estructura CMC_time.

B.1.2.2 CMC_X_COM_RECIP_ID

Una extensión de datos para añadir un identificador de recibiente opaco único a la estructura de recibiente. La proporciona la implementación durante la resolución de nombre de recibiente y puede utilizarse para evitar una ulterior resolución de nombre durante el envío en algunos servicios. Es análoga a la referencia de mensaje.

En el momento del establecimiento de la sesión, el código de ítem se pasa en la matriz CMC_X_COM_SUPPORT_EXT para indicar que este miembro de datos debe añadirse a la estructura de recibiente durante la sesión.

USADA POR

CMC_recipient

ENTRADA

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

item_data

longitud del identificador de recibiente

item_reference

puntero al identificador de recibiente

SALIDA

extension_flags

inalteradas

item_data

longitud del identificador de recibiente

item_reference

puntero al identificador de recibiente

B.1.2.3 CMC_X_COM_ATTACH_CHARPOS

Extensión de datos para soportar la visualización de una representación gráfica de la añadidura en la nota textual del mensaje. La extensión contiene la posición de carácter para la representación.

En el momento del establecimiento de la sesión, el código de ítem se pasa en la matriz CMC_X_COM_SUPPORT_EXT para indicar que este miembro de datos debe añadirse a la estructura de añadidura durante la sesión.

USADA POR

CMC_attachment

ENTRADA

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

item_data

Desplazamiento medido en caracteres y basado en cero, de la añadidura, dentro de los datos de la nota de texto.

NOTA – Éste es un desplazamiento medido en caracteres y no un desplazamiento medido en octetos; esta distinción es importante cuando se están utilizando juegos de caracteres de múltiples octetos.

item_reference

NULL

SALIDA

extension_flags

inalteradas

item_data

Desplazamiento medido en caracteres y basado en cero, de la añadidura, dentro de los datos de la nota textual.

item_reference

inalterada

B.1.2.4 CMC_X_COM_PRIORITY

Extensión de datos para prioridad de mensaje.

En el momento del establecimiento de la sesión, el código de ítem se pasa en la matriz CMC_X_COM_SUPPORT_EXT para indicar que este miembro de datos debe añadirse a la estructura de mensaje durante la sesión.

USADA POR

CMC_message
CMC_message_summary

ENTRADA

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

item_data

Se fija a CMC_X_COM_URGENT, CMC_X_COM_NORMAL, o CMC_X_COM_LOW, según la urgencia del mensaje.

item_reference

NULL

SALIDA

extension_flags

inalteradas

item_data

Se fija a CMC_X_COM_URGENT, CMC_X_COM_NORMAL, o CMC_X_COM_LOW, según la urgencia del mensaje.

item_reference

inalterada

B.2 Sumario de las declaraciones en lenguaje C para el conjunto de extensiones

Esta subcláusula indica las declaraciones que definen la interfaz CMC para el conjunto de extensiones comunes en el lenguaje de programación C.

Las declaraciones aquí reunidas forman el contenido de un fichero de encabezamiento ("*header file*") al que tendrán acceso los programadores de aplicaciones. Se incluyen en el fichero de encabezamiento **<xcmcext.h>**. Los símbolos definidos por las declaraciones son los únicos símbolos que el servicio hace visible para la aplicación.

```
/* COMMON EXTENSIONS DECLARATIONS */

/* EXTENSION SET ID */

#define CMC_XS_COM                    ((CMC_uint32) 0)

/* FUNCTION EXTENSIONS */

/* Query for extension support in implementation */
```

```
#define CMC_X_COM_SUPPORT_EXT        ((CMC_uint32) 16)

typedef struct {
             CMC_uint32         item_code;
             CMC_flags          flags;
          } CMC_X_COM_support;

#define CMC_X_COM_SUPPORTED          ((CMC_flags) 1)
#define CMC_X_COM_NOT_SUPPORTED      ((CMC_flags) 2)
#define CMC_X_COM_DATA_EXT_SUPPORTED ((CMC_flags) 4)
#define CMC_X_COM_FUNC_EXT_SUPPORTED ((CMC_flags) 8)
#define CMC_X_COM_SUP_EXCLUDE        ((CMC_flags) 16)

/* Get back a structure with configuration data */

#define CMC_X_COM_CONFIG_DATA        ((CMC_uint32) 17)

typedef struct {
             CMC_uint16             ver_spec;
             CMC_uint16             ver_implem;
             CMC_object_identifier  character_set;
             CMC_enum               line_term;
             CMC_string             default_service;
             CMC_string             default_user;
             CMC_enum               req_password;
             CMC_enum               req_service;
             CMC_enum               req_user;
             CMC_boolean            ui_avail;
             CMC_boolean            sup_nomkmsgread;
             CMC_boolean            sup_counted_str;
       } CMC_X_COM_configuration;

/* Check to see if a recipient can be sent */

#define CMC_X_COM_CAN_SEND_RECIP     ((CMC_uint32) 18)

#define CMC_X_COM_READY              ((CMC_enum) 0)
#define CMC_X_COM_NOT_READY          ((CMC_enum) 1)
#define CMC_X_COM_DEFER              ((CMC_enum) 2)

/* Save a message to the inbox */

#define CMC_X_COM_SAVE_MESSAGE       ((CMC_uint32) 19)

/* Get back a message structure for the message just sent */

#define CMC_X_COM_SENT_MESSAGE       ((CMC_uint32) 20)

/* DATA EXTENSIONS */

/* attach received data to message and message summary structures */

#define CMC_X_COM_TIME_RECEIVED      ((CMC_uint32) 128)

/* attach a unique id to resolved recipient structures */

#define CMC_X_COM_RECIP_ID           ((CMC_uint32) 129)

/* set character position in the message text to display an icon
  associated with a particular attachment */

#define CMC_X_COM_ATTACH_CHARPOS     ((CMC_uint32) 130)

#define CMC_X_COM_PRIORITY           ((CMC_uint32) 131)

#define CMC_X_COM_NORMAL             ((CMC_enum) 0)
#define CMC_X_COM_LOW                ((CMC_enum) 1)
#define CMC_X_COM_URGENT             ((CMC_enum) 2)
```

B.2.1 Conjunto de extensiones X.400

Los siguientes identificadores de conjunto de extensiones se registran en la Asociación XAPIA para la utilización de X.400:

```
#define CMC_XS_X400                  ((CMC_uint32) 0x00000600)
#define CMC_X_X400_ERROR             ((CMC_uint32) 0x00000601)
#define CMC_X_X400_MSG_PARENT        ((CMC_uint32) 0x00000602)
```

```
#define CMC_X_X400_MSG_ID           ((CMC_uint32) 0x00000603)
#define CMC_X_X400_MSG_REPORT_ID    ((CMC_uint32) 0x00000604)
#define CMC_X_X400_REPORT           ((CMC_uint32) 0x00000605)
```

B.2.1.1 Estructura CMC_Report (informe CMC)

La siguiente estructura "C" se está utilizando en la extensión CMC_X_X400_REPORT:

```
typedef struct {
      CMC_recipient       *msg_recipient;
      CMC_enum            report_type;
      CMC_time            delivered_time;
      CMC_uint32          reason_code;
      CMC_flags           report_flags;
} CMC_report;

/* report_type */

      #define CMC_X400_DR       ((CMC_enum) 0)
      #define CMC_X400_NDR      ((CMC_enum) 1)

/* report_flags */

      #define CMC_REPORT_LAST_ELEMENT ((CMC_flags) 0x80000000)
```

B.2.1.2 Código de error: CMC_EX_X400_STD

Se define un nuevo código de error para precisar que el fallo de una función CMC se debe a una condición de excepción/aborto/error X.400. Este código se utilizará en los 16 bits de orden superior del código de retorno CMC para indicar que el error está asociado con los servicios de mensajería de las Recomendaciones X.400-X.420 (1988).

La definición "C" de este error es:

```
#define CMC_EX_X400_STD   ((CMC_uint16) 400)
```

Por tanto, si la implementación CMC desea clasificar una condición de error causada por el servicio de mensajes X.400 subyacente y se retornarán extensiones CMC relacionadas con X.400, facultativas, el código de retorno CMC puede fijarse a lo siguiente:

CMC_return_code.<16 bits de orden inferior> = CMC_E_FAILURE, o el error más apropiado.

CMC_return_code.<16 bits de orden superior> = CMC_EX_X400_STD

B.2.2 Extensiones adicionales para la correspondencia CMC simple/X400

B.2.2.1 CMC_X_X400_ERROR

Si la función CMC falla porque la operación X.400 subyacente fracasó, el error resultante de las operaciones X.400 se retorna a la aplicación, de modo que ésta pueda determinar la verdadera causa del error. Esta extensión contiene errores específicos que están definidos por las Recomendaciones X.400-X.420 (1988). Para la explicación y el valor de este error, véase el documento pertinente.

NOTA – Si la aplicación CMC desea que la implementación CMC retorne esta extensión si se produce un error X.400, la aplicación debe suministrar el espacio para el almacenamiento de esta extensión cuando se invoque la función CMC; de lo contrario, la implementación CMC no puede retornar esta extensión porque el argumento extensión de cada función CMC es sólo una dirección a la que la aplicación ha asignado la memoria tampón. La alternativa es la siguiente: o bien se modifica la especificación CMC V1.0 para permitir que el argumento extensión de la función sea un argumento de entrada y de salida, o bien la implementación CMC suministra una nueva función ErrInfo para que la aplicación obtenga el error detallado en esta extensión después que una función CMC ha fallado como consecuencia de un error relacionado con X.400.

USADA POR

cmc_act_on(), cmc_list(), cmc_logon(), cmc_logoff(), cmc_read(), cmc_send()

SALIDA

item_code

CMC_X_X400_ERROR

item_data

item_data.<16 bits de orden superior> = número de la operación definido por X.400

item_data.<16 bits de orden inferior> = códigos de retorno de la operación definidos por X.400

item_reference

NULL

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

B.2.2.2 CMC_X_X400_MSG_PARENT

La memoria de mensajes X.400 soporta mensajes anidados, para lo cual utiliza los conceptos de mensaje progenitor (*parent message*) y mensaje vástago (*child message*). Por ejemplo, en el caso de una parte de cuerpo de un mensaje interpersonal (IPM, *interpersonal message*) que contiene un IPM reenviado, el IPM reenviado es un mensaje vástago y el IPM que reenvía es el mensaje progenitor, o, en el caso del contenido de un informe, el IPM retornado es el mensaje vástago y el informe propiamente dicho es el mensaje progenitor. Se utilizará una nueva extensión para permitir que la aplicación determine si un mensaje es progenitor o vástago.

Identificación utilizada para indicar si la referencia (de mensaje) del mensaje CMC o del sumario de mensaje CMC es un mensaje progenitor o un mensaje vástago X.413. Si el mensaje asociado es un mensaje progenitor, no se retornará esta extensión.

USADA POR

CMC_message and CMC_message_summary

SALIDA

item_code

CMC_X_X400_MSG_PARENT

item_data

Número secuencial de progenitor X.413 para mensaje vástago.

item_reference

NULL

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

B.2.2.3 CMC_X_X400_MSG_ID

Cuando se envía un mensaje, X.400 crea un identificador único para este mensaje, un identificador de sistema de transferencia de mensajes (MTS, *message transfer system*). Este identificador se utiliza para rastrear mensajes y para informar la entrega/no entrega de un mensaje. Se retorna a la aplicación CMC a través de la extensión ID de mensaje cuando se lee, lista o envía un mensaje. Así, la aplicación puede responder, o hacer referencia, a un determinado mensaje con la acción apropiada.

Es una identificación única de un mensaje, proporcionada por el servicio de mensajería subyacente cuando la aplicación CMC lee, lista, o envía un mensaje.

USADA POR

CMC_message, CMC_message_summary, y **cmc_send()**.

Lectura y listado de un mensaje:

Cuando se retorna al usuario una estructura de mensaje (o una estructura de sumario de mensaje) tras una llamada a **cmc_read()** [o **cmc_list()**], la extensión ID del mensaje se añade a la estructura. Los datos de ítem para la extensión ID del mensaje son insignificantes y por tanto se hacen cero. La referencia de ítem apunta a la estructura CMC_string asignada por el servicio y contiene un formato legible del identificador MTS único.

Envío de un mensaje:

Cuando el usuario envía un mensaje mediante una llamada a **cmc_send()**, el servicio CMC, el servicio CMC tiene la facultad de retornar al llamante el identificador MTS asignado a ese mensaje en la estructura de extensión de enviar, si el llamante asigna memoria para la plantilla de extensión con el código de ítem de CMC_X_X400_MSG_ID. Si esta extensión está ausente, el servicio CMC no retornará el identificador MTS. El servicio CMC retorna el identificador MTS asignando una cadena CMC (CMC_string) con los datos

requeridos y añadiendo, a la referencia de ítem, un puntero a estos datos. Las banderas de extensión (extension_flags) se fijan con el valor CMC_EXT_OUTPUT. Esto indica al llamante que, después de utilizar la referencia de ítem, deberá liberarla con **cmc_free()**.

SALIDA

item_code

CMC_X_X400_MSG_ID

item_data

NULL

item_reference

puntero a CMC_string de identificador MTS.

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

B.2.2.4 CMC_X_X400_MSG_REPORT_ID

Cuando lee un informe de entrega o no entrega, el servicio de mensajería subyacente retorna un identificador único para el informe (identificador MTS), que es diferente del que corresponde al mensaje original de que trata el informe.

Es un identificador único de un informe de entrega o no entrega proporcionado por el servicio de mensajería subyacente cuando la aplicación lee ese informe.

USADA POR

CMC_message

SALIDA

item_code

CMC_X_X400_MSG_REPORT_ID

item_data

NULL

item_reference

puntero a cmc_string de un formato legible del identificador MTS único.

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

B.2.2.5 CMC_X_X400_REPORT

Se utiliza para transportar a la aplicación CMC la información de entrega o no entrega X.400 específica cuando la base de la información que ha de retornarse es un informe X.400. Esta extensión se retorna como la extensión (de mensaje) del mensaje CMC.

Es el retorno de información específica de informe de entrega o no entrega, definida en la Recomendación X.411 (1988). Para la explicación y el valor de los códigos de motivo (o razón) y de los códigos de diagnóstico, véase el documento pertinente.

USADA POR

CMC_message

SALIDA

item_code

CMC_X_X400_REPORT

item_data

NULL

item_reference

puntero a la estructura CMC_report

extension_flags

Todas las banderas CMC son válidas. No se definen más banderas.

B.2.3 Otros conjuntos de extensiones

Otros conjuntos de extensiones serán definidos por la asociación XAPIA y grupos de vendedores para soportar diversos protocolos de mensajería. Actualmente se están definiendo conjuntos de extensiones para uso con servicios de facsímil G3, facsímil G3-64, facsímil G4, telex, y teletex mediante la Recomendación T.611. La XAPIA proporciona información sobre los conjuntos de extensiones que están disponibles.

B.2.4 Información específica de la plataforma, incluidas las vinculaciones durante la ejecución

Se insta a los implementadores de CMC a que proporcionen interfaces para vinculación durante la ejecución a sus implementaciones de servicio CMC. En general, estas interfaces dependen de la plataforma y/o sistema operativo. En esta subcláusula se indican varios requisitos generales, y específicos de la plataforma, para varias plataformas y sistemas operativos usuales.

A menos que se especifique otra cosa, las siguientes definiciones son aplicables a todas las plataformas:

```
byte                      CMC_sint8
16 bit int                CMC_sint16
32 bit long int           CMC_sint32
16 bit unsigned int       CMC_uint16
32 bit unsigned long int  CMC_uint32
32 bit pointer            CMC_buffer
32 bit char pointer       CMC_string
CMC_uint32                CMC_ui_id
CMC_uint32                CMC_session_id
```

B.2.4.1 Vinculaciones explícitas e implícitas

Todas las funciones de la interfaz API de la CMC deben poder ser vinculadas implícita y explícitamente. La vinculación implícita forma el enlace de la aplicación y la implementación de servicio CMC y lo incorpora en la aplicación. La vinculación explícita requiere que la aplicación contenga código para la fase de ejecución (*run-time code*) que enlace una implementación de servicio CMC.

También se recomienda que se carguen explícitamente todas las funciones de las extensiones, ya que, en algunas implementaciones CMC, el programa de aplicación no se carga si no se han cargado previamente dichas funciones.

A continuación se definen mecanismos estáticos y dinámicos de enlace de módulos de programación (*linking*).

B.2.4.2 Vinculación Apple Macintosh

Para enlaces estáticos de módulos de programación (brevemente, enlaces estáticos), las aplicaciones deben utilizar el convenio de llamadas del lenguaje Pascal y punteros planos de 32 bits para llamar una implementación CMC basada en Apple Macintosh.

Para enlaces dinámicos se debe tomar contacto con Apple Computer Inc.

La implementación CMC debe siempre tratar de proporcionar cadenas internacionales Apple (ISTRING).

B.2.4.3 Vinculación MS-DOS

Para enlaces estáticos, las aplicaciones deben utilizar llamadas "far", el convenio de llamadas del lenguaje C, y punteros "far" segmentados de 32 bits para llamar una implementación CMC basada en MS-DOS. Esto es compatible con el modelo de memoria "large" del lenguaje de programación C de Microsoft. Cualquier futura modificación de este mecanismo será publicada por Microsoft.

La implementación CMC debe siempre tratar de proporcionar la página de código 437 o la 850.

B.2.4.4 Vinculación MS-Windows 3.x

Para enlaces dinámicos, las implementaciones CMC basadas en MS-Windows 3.x deben utilizar bibliotecas enlazadas dinámicas (DLL, *dynamic linked libraries*) y enlazar por nombre a las funciones CMC.

Durante la ejecución, para determinar si un servicio CMC está disponible, las aplicaciones deben llamar la función GetProfileInt() para que busque la variable CMC en la cláusula [MAIL] de WIN.INI. Si esta variable está presente y su valor es diferente de cero, indica que está disponible una biblioteca CMC.DLL. Si dicha variable está ausente o tiene el valor cero, las funciones no pueden llamarse. Toda futura modificación de este mecanismo será publicada por Microsoft.

Las funciones CMC deben llamarse "far", utilizando el convenio de llamada del lenguaje de programación Pascal, y punteros "far" segmentados de 32 bits.

Las estructuras CMC serán alineadas con fronteras cada 4 octetos (32 bits). Esto no se aplicará a los campos de un octeto de la estructura de tiempo, ni de la estructura de cadena contada.

La implementación CMC debe siempre tratar de proporcionar la página de código 1252.

B.2.4.5 Vinculación MS-Windows NT

Para enlaces dinámicos, las implementaciones CMC basadas en MS-Windows NT deben utilizar bibliotecas enlazadas dinámicas y enlazar por nombre a las funciones CMC.

Durante la ejecución, para determinar si un servicio CMC está disponible, las aplicaciones deben interrogar el registro para determinar si CMC está disponible. El mecanismo exacto para esto será publicado por Microsoft.

Las funciones CMC deben llamarse utilizando el convenio de llamada STDCALL.

B.2.4.6 Vinculación OS/2 1.x y 2.x, DLL de 16 bits

Para enlaces dinámicos, las implementaciones CMC de 16 bits basadas en OS/2 1.x y x.2 deben utilizar bibliotecas enlazadas dinámicas y enlazar por nombre a las funciones CMC.

Durante la ejecución, para determinar si un servicio CMC está disponible, las aplicaciones deben llamar la función WinQueryProfileInt() para que busque la variable CMC en la cláusula [MAIL] de OS2.INI. La variable indicará si la DLL es de 16 o de 32 bits. Si esta variable está presente y su valor es diferente de cero, indica que está disponible una biblioteca CMC.DLL. Si dicha variable no se encuentra o tiene el valor cero, las funciones no pueden llamarse. Toda futura modificación de este mecanismo será publicada por IBM.

Las funciones CMC deben llamarse como "far", utilizando el convenio de llamada de sistema y punteros "far" segmentados de 32 bits.

La implementación CMC debe siempre tratar de proporcionar la página de código 850.

B.2.4.7 Vinculación OS/2 2.0, DLL de 32 bits

Para enlaces dinámicos, las implementaciones CMC de 32 bits basadas en OS/2 2.0 deben utilizar bibliotecas enlazadas dinámicas y enlazar por nombre a las funciones CMC.

Durante la ejecución, para determinar si un servicio CMC está disponible, las aplicaciones deben llamar la función WinQueryProfileInt() para que busque la variable CMC en la cláusula [MAIL] de OS2.INI. La variable indicará si la DLL es de 16 o de 32 bits. Si esta variable está presente y su valor es diferente de cero, indica que está disponible una biblioteca CMC.DLL. Si dicha variable no se encuentra o tiene el valor cero, las funciones no pueden llamarse. Toda futura modificación de este mecanismo será publicada por IBM.

Las funciones CMC deben llamarse "far", utilizando el convenio de llamada de sistema (System) y punteros "far" planos de 32 bits.

La implementación CMC debe siempre tratar de proporcionar la página de código 850.

B.2.4.8 Vinculación UNIX SVR4

Para enlaces dinámicos, las aplicaciones deben cumplir con la especificación de la interfaz binaria de aplicación (ABI, *application binary interface*) de UNIX System V Release 4.0 System V y enlazar por nombre a las funciones.

Durante la ejecución, para determinar si un servicio CMC está disponible, las aplicaciones deben buscar la implementación CMC en el trayecto absoluto /usr/lib/XAPI/libCMC.so. La implementación para el sistema se colocará en esta ubicación. Toda futura modificación de este mecanismo será publicada por el vendedor UNIX.

Las funciones CMC y estructuras CMC deben utilizar el convenio de llamada de sistema (System).

La implementación CMC debe siempre tratar de proporcionar la página de código 850.

B.2.5 Utilización de servicios X.400 fundamentales por la CMC simple

En esta subcláusula se describe cómo las funciones de la interfaz de programas de aplicación (API, *application program interface*) de llamadas de mensajería común (CMC, *common messaging call*) versión 1.0 se hacen corresponder con un sistema de tratamiento de mensajes X.400 subyacente en la frontera de la memoria de mensajes (MS, *message store*) y cómo los mensajes CMC se hacen corresponder con mensajes X.400. Esta Recomendación no trata lo siguiente:

La correspondencia (de las direcciones) del directorio X.500, al que puede ganarse acceso mediante cmc_look_up.

El diálogo de interfaz de usuario (UI, *user interface*), que es una opción en la función **cmc_send_documents()**, pues no es esencial para la interacción entre la (aplicación habilitada para mensajería) CMC y el sistema de mensajería X.400.

En esta Recomendación se supone que el lector está familiarizado con el elemento de servicio de operación a distancia (ROSE, *remote operation service element*), y los protocolos P1, P2, P22, P3, y P7, y los elementos de servicio, así como con las especificaciones y objetivos de la CMC API versión 1.0. Se hace referencia a las siguientes Recomendaciones:

– Recomendaciones X.200-X.219 (Modelo y notación de OSI, definición de servicio).

– Recomendaciones X.220-X.229 (Especificaciones de protocolos de OSI).

– Recomendaciones X.400-X.420 (MHS X.400 1984).

– Recomendaciones X.400-X.420 (MHS X.400 1988).

– Recomendación F.401 (1988), anexo B (Representación de direcciones O/R para uso por personas) o su equivalente.

– ISO/CEI 10021-2:1990/Amd.1 Annex F (Representation of O/R Addresses for Human usage).

– XAPIA CMC API versión 1.0.

– XAPIA CMC API versión 2.0.

La correspondencia entre CMC versión 1.0 y X.400 se describe en esta Recomendación con los dos objetivos siguientes:

- De acuerdo con los objetivos simples y de alto nivel de CMC 1.0, para la correspondencia no se utiliza el conjunto completo de las características X.400, por lo que se recomienda solamente un perfil básico.
- Responde a las principales preocupaciones en cuanto a la interoperabilidad de diferentes implementaciones de la CMC versión 1.0, utilizando uno cualquiera de los diversos sistemas de mensajería X.400 como un medio transporte de mensajes.

B.2.5.1 Introducción

La interfaz de programas de aplicación para llamadas de mensajería común (API para CMC, o brevemente CMC API) proporciona, a las aplicaciones habilitadas para mensajería, un conjunto de funciones de alto nivel que les permite enviar y recibir mensajes electrónicos. Esta interfaz tiene que estar soportada por servicios de mensajería. Un importante servicio de mensajería es el sistema de tratamiento de mensajes (MHS, *message handling system*) X.400 de OSI. Esta Recomendación está destinada a quienes desean integrar la CMC API con el MHS X.400.

Para cada implementación CMC, la visión y capacidades presentadas por la CMC tienen que hacerse corresponder con la visión y capacidades del servicio de mensajería subyacente. Para maximizar la interoperabilidad entre aplicaciones CMC que utilizan diferentes servicios de mensajería subyacentes, la XAPIA ofrece varias directrices. Las cadenas de caracteres de los mensajes deben hacerse corresponder con juegos de caracteres internacionales siempre que sea posible y los tipos de añadidura de mensaje deben hacerse corresponder con tipos de añadidura de mensaje comúnmente reconocidos, siempre que ello sea apropiado o posible.

Para lograr esta correspondencia, las características del servicio de mensajería subyacente (MHS) tienen que ser entendidas y utilizadas de la manera más adecuada. En el resto de esta Recomendación se examinan los temas siguientes:

- Una visión de conjunto de alto nivel de la Recomendación X.400.
- La utilización de la CMC API simple por encima del servicio de mensajería X.400, estudiándose puntos tales como opciones, detalles de interés, y posibles extensiones para soportar un conjunto más rico de funciones para aplicaciones habilitadas para mensajería y aplicaciones basadas en mensajería.
- Un perfil de correspondencia básica para la CMC API simple que proporciona una interoperabilidad simple para enviar y recibir mensajes a través de una diversidad de servicios de comunicaciones MHS X.400.

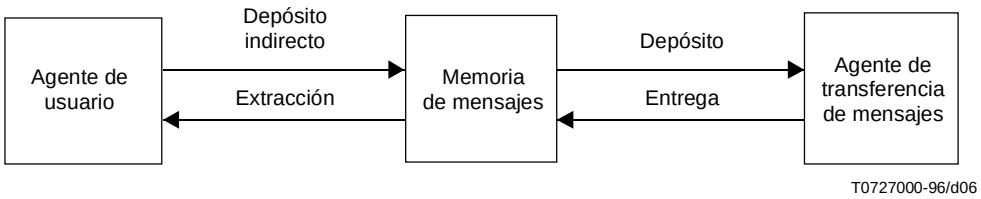
B.2.5.2 Visión de conjunto de alto nivel de X.400

Los servicios de tratamiento de mensajes proporcionados por el sistema de tratamiento de mensajes X.400 incluyen un servicio de mensajería interpersonal (IPM) y un servicio de transferencia de mensajes. Estos servicios permiten a los abonados intercambiar mensajes por el procedimiento de almacenamiento y reenvío (*store and forward*). El servicio de tratamiento de mensajes define un conjunto de tipos de mensajes y capacidades que un originador puede enviar a recibientes.

Un originador prepara un mensaje con la asistencia de un agente de usuario (UA, *user agent*). El agente de usuario es una aplicación que interactúa con el sistema de transferencia de mensajes (MTS, *message transfer system*). El sistema de transferencia de mensajes consta de un número de agentes de transferencia de mensajes (MTA, *message transfer agent*). Funcionando conjuntamente, estos MTA retransmiten los mensajes a los agentes de usuario de los recibientes deseados, y estos agentes de usuario ponen el mensaje a disposición de los recibientes deseados.

La versión 1988 de las Recomendaciones de la serie X.400 incluía una memoria de mensajes (MS). Un usuario puede depositar mensajes mediante la memoria de mensajes y recibir mensajes que han sido entregados a la memoria de mensajes. La memoria de mensajes funciona sólo a nombre de usuarios individuales.

Depósito y entrega mediante una memoria de mensajes



Las operaciones entre una MS y el MTS corresponden al protocolo P3. Las operaciones entre un UA y la MS corresponden al protocolo P7. Las operaciones del protocolo P7 son:

- Servicio de extracción (sumarizar, listar, capturar, suprimir, registrar MS con una posible señal asíncrona, y avisar).
- El depósito indirecto emplea los servicios de depósito de X.411 (depósito de mensaje, depósito de sonda, cancelación de entrega diferida, y control de depósito).
- Servicios de administración (registrar y cambiar credenciales).
- Para conectar y desconectar los servicios MS se utilizan las operaciones MS vincular (*MS-Bind*) y MS desvincular (*MS-Unbind*). La operación de vincular se utiliza para identificar, autenticar y establecer el contexto de seguridad para un usuario de servicio MS.

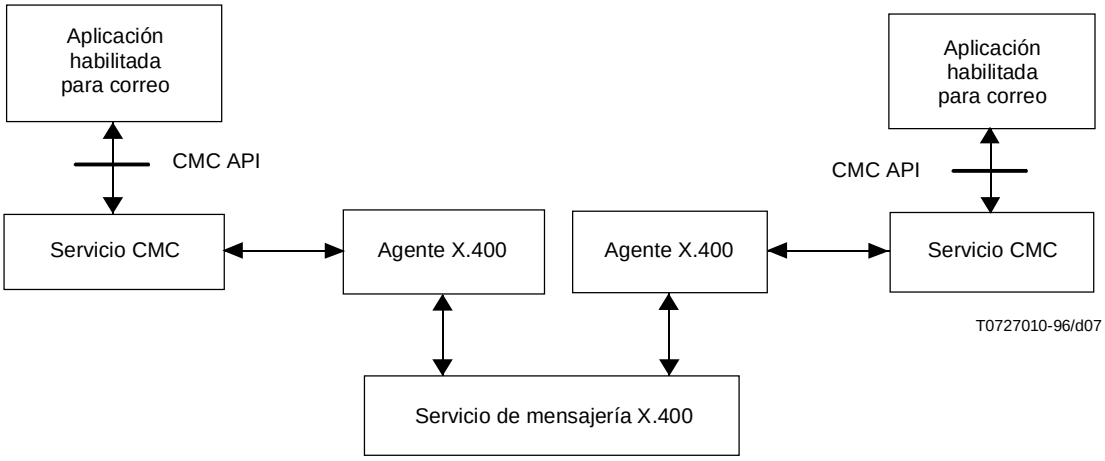
Las operaciones P7 las invoca el UA utilizando el elemento de servicio de operación a distancia (ROSE, definido en las Recomendaciones X.219 y X.229). El modelo ROSE consiste en una interacción de petición y respuesta. Permite al UA solicitar una operación P7 y obtener el resultado de esa operación P7.

B.2.5.3 Planteo y consideraciones generales

En esta subcláusula se presenta una visión genérica de la correspondencia de la CMC v1.0 (conocida también como CMC v2.0 simple) al MHS X.400, y algunas de las posibles consideraciones y opciones. El tema de la interfaz con las personas no se trata porque las aplicaciones habilitadas para mensajería se perciben como capaces de funcionar sin interacción con las personas, aunque pudieran funcionar en base a instrucciones de usuario (comandos) o de instrucciones expresadas en forma de lenguaje "script". Esto significa que las aplicaciones no requieren una interfaz gráfica de usuario y que pueden ejecutarse como procesos que funcionan "en el fondo" (*background processes*).

B.2.5.3.1 Funciones de la CMC y operaciones de la MS X.400

De acuerdo con los modelos funcionales de la CMC v1.0 y de la memoria de mensajes (MS) de X.400, las funciones CMC corresponden razonablemente bien a los servicios MS. Las funciones establecer sesión (logon) y terminar sesión (logoff) de la CMC corresponden a las operaciones vincular y desvincular de la MS. Enviar de la CMC corresponde al depósito indirecto de la MS. Leer de la CMC corresponde a capturar de la MS. Tratar y listar de la CMC están cubiertas por suprimir, sumarizar y listar de la MS.



Servicio de mesajería CMC de aplicación a aplicación

Las dos finalidades de la CMC API son: proporcionar un conjunto genérico de capacidades de mensajería que sean independientes de todo sistema operativo, y proporcionar un número mínimo de llamadas a funciones necesarias para enviar o recibir un mensaje. Un mínimo de llamadas a funciones e interoperabilidad son requisitos esenciales. Cuando se hacen corresponder las funciones de la CMC con las operaciones de la MS, y han de cumplirse estos requisitos, pueden seguirse dos métodos para conseguir una interfaz API simple de llamada a funciones. La elección de uno u otro de estos dos métodos determina la manera de implementar las llamadas CMC.

– Llamadas a funciones de la CMC simple y numerosas extensiones especiales para el entorno local.

– Llamadas a funciones de la CMC que ocultan la complejidad técnica y soportan extensiones genéricas.

B.2.5.3.2 Mensajes CMC y mensajes X.400

Gran parte del trabajo de traducir mensajes CMC a mensajes X.400 y viceversa implica conversiones entre estructuras de mensaje CMC y estructuras de mensaje X.400. Así, los nombres y direcciones de los usuarios del servicio de mensajería CMC tienen que ser convertidos a nombres y direcciones de usuarios (originador y recibiente) X.400, que se conocen por nombres O/R (e incluyen direcciones O/R).

Otras partes de una estructura de mensaje CMC que requieren conversión son el tipo de mensaje, la hora de envío, los recibientes y las añadiduras. Un asunto (*subject*) de mensaje CMC es simplemente un asunto del mensaje interpersonal (IPM) de X.400 (con algunas restricciones en cuanto a la longitud), mientras que la nota textual puede requerir un tratamiento especial por las diferencias que existen entre la Recomendación X.400 y CMC. Las banderas, extensiones, y otros parámetros de entrada originales, de la CMC, se utilizan fundamentalmente para ayudar a las conversiones. No se envían como parte del mensaje X.400, por lo que, en la mayoría de los casos, esa información se pierde cuando el mensaje X.400 entregado se convierte en sentido inverso en un mensaje CMC para la aplicación CMC de destino.

A continuación se indican algunas opciones relativas a conversiones, convenios y otros aspectos, necesarias para la utilización de la mensajería X.400 como los servicios subyacentes de mensajes.

Conversiones de texto

- Conversión de juegos de caracteres.
- Conversión de nombres y direcciones.

Convenios sobre el sistema operativo local y el sistema de mensajería

- Convenios sobre conexión/desconexión.
- Percepción de los errores del sistema de mensajería subyacente.
- Convenios sobre el dispositivo de almacenamiento de mensajes y convenios sobre requisitos especiales.

Convenios sobre mensajes salientes

- Convenios locales o genéricos tales como opciones de conversión de texto.
- Convenios sobre conversiones especiales en el extremo de destino.

Convenios sobre mensajes entrantes

- Tratamiento de mensajes CMC que no pueden tratarse localmente.
- Tratamiento de mensajes CMC que tienen partes que no pueden tratarse localmente.
- Tratamiento de mensajes CMC defectuosos.
- Tratamiento de mensajes que no son de la CMC.

Extensiones que son genéricas, pero específicas en el plano local

- Extensiones para tratar aspectos del sistema subyacente que no son genéricos para la CMC.
- Extensiones que añaden características X.400 especiales, como prioridad y notificación de entrega.
- Extensiones que utilizan partes de cuerpo X.400 específicas como añadiduras.
- Extensiones que se utilizan para asegurar la concordancia entre los sistemas locales emisor y de destino.

B.2.5.3.3 Añadiduras de mensaje CMC y partes de cuerpo X.400

Las añadiduras (textuales y binarias) de mensaje CMC son equivalentes a las partes de cuerpo X.400. La parte de cuerpo más apropiada para una añadidura de mensaje CMC es diferente en cada "versión" de la Recomendación X.400 (es decir, 1984, 1988 ó 1992). A continuación se indican los equivalentes recomendados.

ASCII en CMC	<=>	Texto IA5	
Nota textual CMC como un fichero	<=>	Parte de cuerpo texto IA5	
Añadidura textual CMC	<=>	X.400 1984	Parte de cuerpo texto IA5
Añadidura textual CMC	<=>	X.400 1988	Parte de cuerpo definida externamente
Añadidura textual CMC	<=>	X.400 1992	Parte de cuerpo Transferencia de ficheros
Añadidura binaria CMC	<=>	X.400 1984	Parte de cuerpo definida bilateralmente
Añadidura binaria CMC	<=>	X.400 1988	Parte de cuerpo definida externamente
Añadidura binaria CMC	<=>	X.400 1992	Parte de cuerpo Transferencia de ficheros

B.2.5.3.4 Convenios y requisitos de entrada/salida

El juego de caracteres común, casi universal, utilizado en X.400 es el Alfabeto Internacional N.° 5 (texto IA5) que es similar al ASCII, pero no exactamente igual. Caracteres ASCII tales como "@", "%" y "_" no están en el IA5 básico sino en la Versión Internacional de Referencia (IRV, *international reference version*). Para fines de visualización y entrada, las otras versiones (nacionales) del IA5 requieren un convenio de conversión. OIW del NIST (*National Institute of Standards and Technology*, Instituto nacional de normas y tecnología) tiene un algoritmo de conversión para el intercambio de texto IA5 y texto ASCII y se recomienda la utilización de dicho algoritmo si el texto tratado por la aplicación habilitada para mensajería CMC se basa en una versión del IA5 distinta de la Versión Internacional de Referencia. Se necesita un convenio similar para otros juegos de caracteres, tales como EBCDIC, ISO 10646, UNICODE, etc., así como sobre la manera de tratar los nombres de ficheros que tienen incorporados espacios en blanco, que han pasado a través de cadenas contadas CMC.

Los nombres y direcciones X.400 están organizados internamente en la Recomendación X.400 como un conjunto estructurado de objetos de datos similar a la estructura de recibiente CMC, pero no más complejo que esta estructura. Existe un convenio para visualizar el nombre y la dirección X.400 de una persona en tarjetas personales, etc. Este convenio, junto con los otros convenios de visualización normalizados que figuran en el anexo B/F.401 (que hace referencia al anexo F de ISO/CEI 10021-2) debe utilizarse para la representación de direcciones de recibientes X.400 mediante cadenas de texto CMC. La utilización de la consulta CMC (*CMC look up*) podría proporcionar una manera sencilla de ir de un nombre conocido como "eowens", o bien a una cadena de texto de dirección (nombre) como la ilustrada más adelante, o a una estructura de recibiente CMC.

> Los recibientes CMC (originador en una estructura de sumario de mensaje CMC o recibiente en una estructura de mensaje CMC) toman cadenas de direcciones que concuerdan con las especificadas en la Recomendación F.401. Por ejemplo, la cadena de dirección de recibiente CMC con "S=Owens; G=Edward; P=ccmail; A=telemail; C=US"

Se necesitarán múltiples convenios para la representación de nombres y direcciones si se soporta una combinación de sistemas de mensajería subyacentes. En la ilustración precedente se ha supuesto que hay un solo sistema de mensajería subyacente (X.400). También con relación a los nombres y direcciones pasados a través del sistema de mensajería, se requieren conversiones cuando los juegos de caracteres utilizados en el sistema de mensajería no soportan los convenios de nombres y direcciones originales. Si (en un texto inglés) name@address no puede pasarse, salvo si existe un convenio para representar esto por ejemplo como name(a)address en un texto para visualización en IA5, entonces, el convenio inverso debe aplicarse en el otro extremo, o el convenio tiene que ser comprendido por el recibiente. Por ejemplo, cuando una dirección X.400 incluye atributos definidos por el dominio como DDA para una dirección Internet, el nombre y la dirección podrían ser:

DDA:RFC-882=fred(a)widget.co.uk;O=gateway;P=abc;C=gb

Los convenios relativos al tamaño de las palabras y/o al orden de los octetos pueden también crear problemas, con la consecuencia de que algunas añadiduras serían inutilizables y los números resultarían adulterados. Normalmente, la solución a la mayor parte de estos problemas consiste en normalizar el contenido y formato de lo que se transmite por el hilo. Sin embargo, si lo que se transmite por el hilo depende del sistema de mensajería subyacente, y se puede estar utilizando un número desconocido de estos sistemas (aunque no todos ellos fueran utilizados por unas aplicaciones habilitadas para mensajería), sería necesaria una solución diferente. Una solución simple es hacer que las aplicaciones (emisoras) habilitadas para mensajería tengan conocimiento de las capacidades de las aplicaciones (receptoras) habilitadas para mensajería, y adapten el mensaje consecuentemente. Esto simplifica las transferencias entre sistemas con capacidades similares. Otra posible solución es enviar, con el mensaje, información adicional que ponga en conocimiento del sistema receptor las capacidades del emisor y/o los detalles particulares del formato del mensaje, y otros detalles especiales.

Si se puede establecer convenios especiales, la utilización de extensiones CMC proporciona un medio de informar a las implementaciones CMC emisoras cómo pasar esas informaciones a través del sistema de mensajería subyacente a la implementación CMC lectora, de modo que la aplicación habilitada para mensajería pueda elegir la manera de leer el mensaje. Por el momento se recomienda que la CMC 1.0 deje esto para que sea resuelto por la implementación local, ya que las cuestiones se resuelven mejor cuando se han comprendido todos los requisitos locales.

B.2.5.3.5 Conexión/desconexión de X.400 y requisitos de la MS

La conexión a un servicio subyacente X.400 y la desconexión de este servicio están sujetas a variaciones locales y también dependen del servicio o interfaz que intervenga. Se parte del supuesto de que la conexión se efectúa con la memoria de mensajes de X.413. Este servicio abarca los requisitos CMC y proporciona una funcionalidad adicional. Las diversas características y funcionalidades de la interfaz X.413 requieren interpretaciones, para que los servicios CMC esperados se proporcionen de una manera comprensible.

Para que la operación vincular X.413 actúe como una función establecer sesión CMC con un sistema de mensajería X.400, los argumentos "usuario" y "contraseña" tienen que ser cadenas que contengan el nombre del usuario del servicio de mensajería y la contraseña que da al usuario acceso al servicio subyacente. En términos de X.413, éstos son el ORAddressAndOrDirectoryName y las InitiatorCredentials. Suponiendo que el servicio X.413 sólo requiera la autenticación simple, se recomienda que "usuario" sea la versión de texto de visualización F.401 del nombre y la dirección del usuario X.400.

Cuando el usuario pudiera requerir una autenticación distinta de la simple y otros argumentos de entrada para la operación vincular MS X.413, las extensiones CMC especiales proporcionan una manera local de añadir peticiones de contexto de seguridad, de restricciones de captura, y de configuración del dispositivo de almacenamiento de correspondencia. Pueden ser también necesarias extensiones CMC suplementarias para tratar localmente el resultado retornado por la llamada a vincular MS o los errores de vincular retornados. Una solución simple consiste en ignorar el resultado de la llamada a vincular MS o los errores de vincular retornados, y hacer que la función establecer sesión CMC retorne CMC_E_FAILURE si la operación vincular MS retorna un error. Una solución diferente es la de utilizar una extensión CMC facultativa para informar del error X.400 asociado.

Los mensajes X.400 entrantes se almacenan en la memoria de mensajes y se leen, sea por una petición de un determinado mensaje de correo (un número conocido), sea pidiendo que se lea el siguiente ítem de correo "no leído". Las añadiduras se retornarán, o bien en un directorio temporal con sus nombres de ficheros de añadidura como nombres de ficheros, o bien en ficheros con nombres temporales en un directorio con alguna indicación del título del emisor. Las partes de los mensajes de correo entrantes que no puedan hacerse corresponder con la estructura de mensaje CMC serán descartadas.

La memoria de mensajes tendrá una interfaz P7 e implementará las características X.413 para hacer indagaciones sobre los correspondientes mensajes de correo de cliente contenidos y para entregar ítems solicitados, y también ítems "X.420", como el encabezamiento y las diversas partes de cuerpo que forman el contenido de un mensaje interpersonal X.420.

La operación desvincular MS cierra la asociación del usuario (o aplicación habilitada para mensajería) y no tiene argumento, ni resultado, ni error. Por tanto, una función de terminación de sesión CMC no tiene otras complicaciones debidas al servicio de mensajería X.400 subyacente.

Las aplicaciones (o usuarios) habilitadas para correo deben tener conocimiento de una característica de la memoria de mensajes. La memoria de mensajes puede contener inscripciones vástagos, además de las inscripciones principales para los mensajes almacenados. Estas inscripciones vástagos son listadas junto con sus inscripciones progenitores, pero la aplicación (o usuario) habilitada para mensajería sólo puede suprimir inscripciones vástagos suprimiendo su inscripción progenitora. La supresión de una inscripción vástago no funciona de la manera prevista.

B.2.5.4 Conversión de mensajes

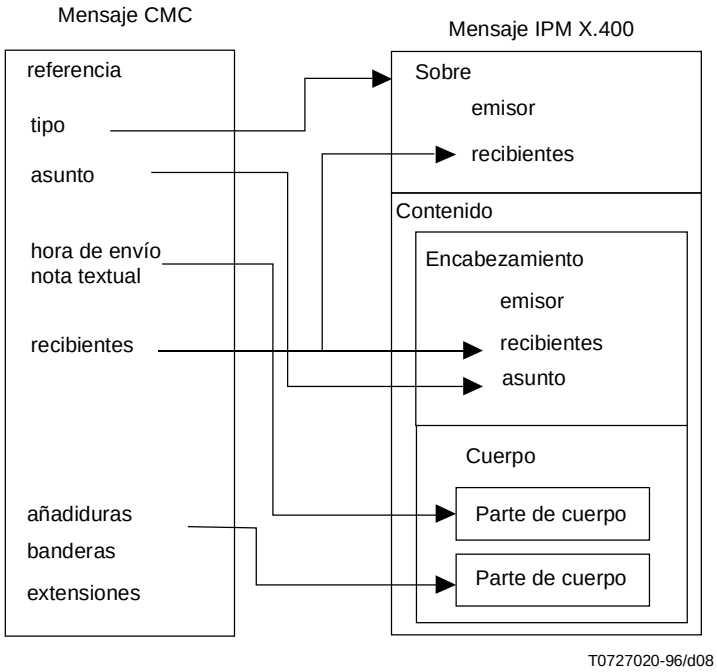
En esta subcláusula se examina la conversión de mensajes CMC a mensajes X.400, la conversión de mensajes X.400 a mensajes CMC y la conversión de mensajes que no son ni CMC ni X.400.

B.2.5.4.1 Conversión de mensajes CMC a mensajes X.400

Una estructura de mensaje CMC incluye un tipo de mensaje, un asunto, uno o más recibientes y posiblemente una nota y/o un conjunto de añadiduras. El equivalente X.400 de este mensaje CMC es una unidad de datos de protocolo de mensaje (MPDU, *message protocol data unit*) de usuario. Una MPDU de la versión 1984 tiene una estructura básica de sobre y contenido. Se proporcionan servicios adicionales a individuos que desean comunicar con otros por medio de agentes de usuario (UA). Este servicio es un sistema de mensajería interpersonal (IPMS, *interpersonal messaging system*) con un tipo de contenido asignado para esos mensajes. Los UA de IPM comunican con otros UA de IPM. El contenido del IPM se divide en encabezamiento y cuerpo. Por eso, para fines de conversión, el mensaje CMC se descompone en sobre, encabezamiento, y cuerpo.

El sobre X.400 contiene la identidad del emisor (nombre y dirección), las identidades de los recibientes, y detalles de sobre de mensaje X.400 específicos. El encabezamiento del IPM contiene la identidad del emisor, un conjunto de usuarios autorizantes, un conjunto de identidades de recibientes (primarios, de copia, y de copia ciega), un asunto y otros detalles específicos de X.400. El cuerpo está constituido por una secuencia de partes de cuerpo. Cada parte de cuerpo es de un tipo de un conjunto de tipos diferentes. Los tipos de mayor interés son IA5Text y parte de cuerpo definida bilateralmente (tipo 14). Se han definido tipos de mayor interés para versiones ulteriores (1988 y 1992) de la Recomendación X.400.

Las últimas adiciones de tipos de parte de cuerpo incluyen la parte de cuerpo definida externamente (1988) y la parte de cuerpo de transferencia de ficheros (1992). Estos dos tipos son mucho mejores para uso en CMC, pues permiten transportar (sin que se pierda) más que simplemente datos. Por tanto, hay una gama de opciones para la nota y las añadiduras.



A continuación se indican algunas de las opciones:

- Convertir todos los mensajes CMC a X.400 (1984) con el texto (nota y añadidura) contenido en partes de cuerpo IA5Text y las añadiduras "binarias" contenidas en parte de cuerpo definida bilateralmente (BDBP, *bilaterally defined body part*)/no identificado (tipo 14).

- Convertir todos los mensajes CMC a X.400 (1988) con nota textual en IA5Text y añadiduras contenidas en parte de cuerpo definida externamente (EDBP, *externally defined body part*) (tipo 15).

- Convertir todas las añadiduras según el año (la versión) de la Recomendación X.400.

- Efectuar las conversiones según una extensión CMC especial que indique la parte de cuerpo X.400 que habrá de utilizarse en cada caso, incluso la utilización de otras partes de cuerpo.

Las ulteriores versiones de la Recomendación X.400 tienen una mayor gama de opciones de parte de cuerpo. Estas últimas partes de cuerpo responden más estrechamente a los requisitos de añadiduras CMC. La serie de Recomendaciones X.400 de 1984 soportaba una parte de cuerpo estilo de texto y una parte de cuerpo estilo de cadena de octetos. La parte de cuerpo IA5Text y la parte de cuerpo definida bilateralmente sólo pueden transportar texto o cadenas "binarias", pero no la información de añadidura suplementaria de attach_title (título de añadidura) ni la de attach_filename (nombre de fichero de añadidura). Un convenio, como el de usar una parte de cuerpo suplementaria para la información suplementaria es posible, pero no se recomienda.

La serie X.400 de 1988 desaconseja la utilización de la parte de cuerpo definida bilateralmente y recomienda la utilización de la parte de cuerpo definida externamente. Las capacidades suplementarias de la parte de cuerpo definida externamente permiten transportar la información de título de añadidura junto con la parte de cuerpo. La serie X.400 de 1992 ha definido una parte de cuerpo transferencia de ficheros para transferir el contenido de un fichero almacenado y, facultativamente, sus atributos. La porción de contenido es como la parte de cuerpo definida externamente, mientras que la porción del parámetro facultativo transporta atributos como fichero almacenado conexo, tipo de contenido, relaciones, y atributos de fichero. Por tanto la parte de cuerpo transferencia de ficheros es ideal para transportar añadiduras CMC.

La serie X.400 de 1984 es un importante sistema de mensajería X.400 soportado, por lo que es necesaria una solución de compromiso. En un perfil básico sugerido se utilizan las partes de cuerpo X.400 de 1984 y se requiere que si se utilizan las ulteriores partes de cuerpo, puedan ser reducidas en grado para que sean equivalentes a las partes de cuerpo 1984. Asimismo, las aplicaciones X.400 habilitadas para mensajería que utilizan las partes de cuerpo más avanzadas deben poder aceptar el conjunto de perfiles básicos en lugar de aquéllas. Esto significa que los títulos de añadidura requieren una sustitución automatizada en el extremo receptor, porque no se transfieren.

Los sistemas de tratamiento de mensajes X.400 requieren la presencia de un conjunto de atributos obligatorios como parte de un depósito de mensaje. Entre estas entradas obligatorias está la de originador, nombre OR del emisor. Esto se requiere también para el establecimiento de sesión MS, de modo que si entre los recibientes del mensaje CMC no hay uno con el rol de CMC_ROL_ORIGINATOR, se utilizará por defecto el "usuario" de los parámetros del establecimiento de sesión MS. Se requiere también un nombre de recibiente y, si no se proporciona, el depósito de mensaje debe rechazarse.

El mensaje X.400 que se envía como un mensaje interpersonal (IPM) debe tener su tipo de contenido fijado a mensajería interpersonal 198(4 u 8). Otros atributos X.400 obligatorios deben suministrarse, sea como un valor por defecto (prioridad fijada como normal), o fijando los bits de PerMessageIndicators de modo que indiquen que no se permite la revelación de los recibientes, conversión implícita prohibida, ausencia de recibiente alterno, y ausencia de retorno de contenido. Además, otro atributo de entrada obligatorio debe solicitar que los bits de OriginatorReportRequest se fijen de modo que indiquen un informe de no entrega.

Para soportar la utilización de la parte de cuerpo más apropiada o para añadir otros atributos X.400 a un mensaje, se recomienda la utilización de extensiones CMC facultativas. Éstas se examinan más adelante. Sin embargo, para una utilización inicial y básica del servicio de mensajería X.400, se recomienda el perfil básico, que no requiere extensiones suplementarias, ya que esto, si bien contribuye a la interoperabilidad, no proporciona el mejor intercambio entre dos sistemas habilitados para mensajería cuando cada uno de ellos soporta las extensiones del otro.

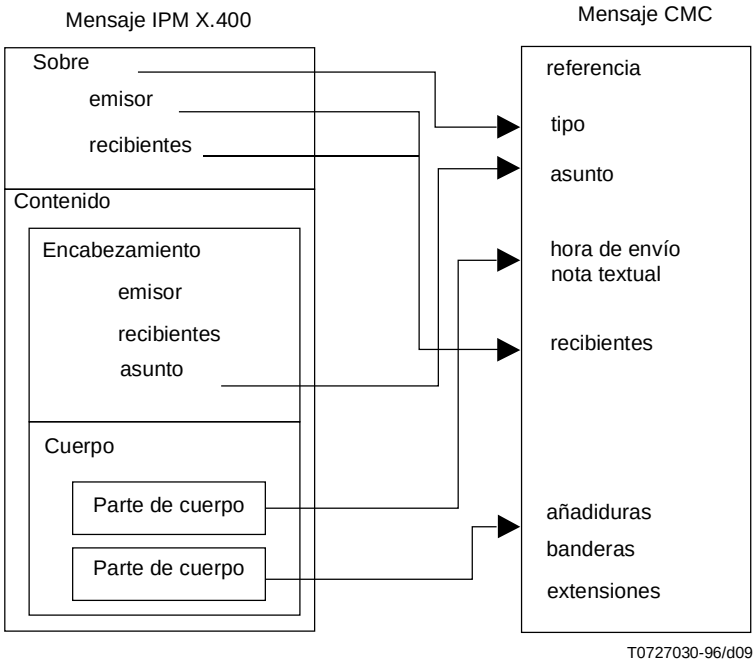
B.2.5.4.2 Conversión de mensajes CMC a mensajes X.400

Los mensajes X.400 existen en una diversidad de tipos y versiones. La principal combinación de tipo y tamaño que es apropiada para CMC es el mensaje interpersonal (IPM), si bien una memoria de mensajes X.400 puede contener cualquier género de mensaje X.400, incluidos mensajes dañados. Las principales modalidades son mensajes, sondas e informes. Entre los mensajes están los mensajes "P1", los mensajes interpersonales, y otros mensajes "P2" como los mensajes de intercambio electrónico de datos (EDI, *electronic data interchange*) X.435. Los mensajes interpersonales incluyen notificaciones interpersonales y versiones basadas en las Recomendaciones de 1984, 1988, y 1992.

Se ha facilitado la decisión sobre varias opciones:

- El usuario CMC ve los mensajes "CMC" y otros mensajes X.400 contenidos en la memoria de mensajes.
- El usuario CMC puede también leer parcialmente los otros mensajes X.400.
- El usuario CMC puede suprimir esos otros mensajes (salvo cuando sean vástagos de otros mensajes).

La conversión de mensajes CMC a mensajes X.400 se ha descrito en la subcláusula anterior. Por eso, la conversión en sentido inverso a mensajes CMC es bastante sencilla, aunque en ese proceso podría perderse alguna información. Por ejemplo, las partes de cuerpo X.400 de 1984 utilizadas para transportar añadiduras no pueden transportar también el nombre de añadidura, por lo que éste se pierde.



Los informes generados por la entrega o no entrega de un mensaje X.400 "CMC" aparecen como informes en la MS del emisor. A petición del originador, el mensaje contendrá el IPM retornado que es no entregable. También es posible que un informe X.400 contenga un informe de entrega para algunos recibientes y un informe de no entrega para otros recibientes, lo que dependerá del punto en que se produjo el fallo. Desde el punto de vista de la MS, los informes de entrega y los informes de no entrega forman parte del informe, lo que es muy diferente del punto de vista de la CMC. Estos informes tienen que ser convertidos de una manera apropiada de modo que el usuario CMC pueda correlacionar este informe con el mensaje que se envió.

Otros mensajes que están en la memoria de mensajes podrían ser mensajes "CMC" enviados por otros usuarios CMC que emplean un método diferente para convertir sus mensajes CMC a mensajes X.400. Por tanto es necesario poder tratar toda una gama de posibles conversiones. Cualquiera que sea el método que se adopte, deberá permitir al usuario receptor elegir entre no tener en cuenta la conversión, o aceptarla en todo o en parte, ya que hay disponibles otros medios de conversión.

La cuestión de la reducción de grado (*downgrading*) y de la elevación de grado (*upgrading*) o de otras conversiones X.400 de mensajes X.400 suscita problemas adicionales. Todo esto significa que, o bien hay pocas importaciones directas de un mensaje X.400 "no CMC" a un mensaje CMC que permitirían a un usuario CMC "ver" ese mensaje, o bien hay un conjunto complejo de reglas. En el caso de conversiones simples, las partes de cuerpo IA5Text son notas o añadiduras textuales. Todas las otras partes de cuerpo podrían clasificarse como "binarias", y se dejaría al usuario CMC con el "tipo", y quizás con el "título" como indicios para averiguar el contenido de la añadidura. Todos los mensajes que no son convertibles podrían indicarse mediante un texto en la línea de asunto.

A menudo, la versión (1984, 1988 ó 1992) del X.400 local será el factor de mayor peso a la hora de determinar si un mensaje X.400 o una parte de cuerpo del IPM es o no convertible. El tipo de un mensaje o de una parte de cuerpo podría utilizarse para descartar ese elemento y sustituirlo por un indicador del usuario CMC.

Se recomienda un convenio simple para el perfil básico (mínimo). El perfil del lado emisor del mensaje tiene fijada la petición de informe de no entrega, de modo que los informes puedan aparecer en la memoria de mensajes. Puesto que la información principal contenida en el informe es la que precisa si el informe fue o no entregado, sólo esto deberá retornarse a la aplicación habilitada para mensajería. Por tanto, una conversión requerida es la sustitución de un informe por un mensaje que contiene la nota textual de "este mensaje (no) fue entregado", así como los recibientes, etc.

Se necesitarán otras conversiones si el tipo de contexto es para una versión (año de X.400) diferente y las partes de cuerpo no son las previstas: por ejemplo, una parte de cuerpo definida bilateralmente en lugar de una parte de cuerpo definida externamente, o una parte de cuerpo transferencia de mensajes en lugar de una parte de cuerpo IA5Text. Aunque la conversión de una parte de cuerpo a otra en estos casos requiere código suplementario, las reglas de conversión son simples.

B.2.5.4.3 Conversión de mensajes que no son CMC/X.400

Cuando una aplicación habilitada para mensajería o su usuario tiene una dirección X.400, la memoria de mensajes X.400 almacenará todo mensaje X.400, como un mensaje EDI, IPM, o informe que se envía a esa dirección. Así, el usuario puede enviar muchos mensajes que no han sido originados por la CMC, o que, si bien son mensajes CMC, utilizan convenios diferentes, que el sistema de correos CMC local no puede tratar. Para abordar estas cuestiones tiene que haber un conjunto de convenios sobre lo que se descarta, lo que se convierte parcialmente para que el usuario conozca algunos de los detalles del mensaje, y sobre las partes que pueden sustituirse u omitirse.

Para una conversión simple existen las siguientes posibilidades:

- Todos los mensajes "no CMC" se descartan después de que hayan sido "leídos" por una aplicación CMC.
- Todos los mensajes CMC que tengan una forma diferente (por ejemplo, un mensaje PEM como una parte de cuerpo P22 X.400) que no pueda ser tratada por el sistema local (por ejemplo, basado en X.400 de 1984) se descartan. Las partes de cuerpo que la CMC local y la implementación X.400 puedan tratar se sustituyen por una nota textual simple, un fichero de cadena de caracteres, o un fichero "binario".
- Todas las partes de cuerpo desconocidas se convierten en una forma de fichero parcialmente utilizable mediante el empleo de una conversión de visualización simple de todos los caracteres imprimibles que aparezcan, y todos los caracteres no imprimibles se convierten en una forma de visualización "\'hex"hex'" de modo que un "usuario" pueda determinar si ese fichero es recuperable.

Como se ha expresado antes, pueden utilizarse extensiones CMC facultativas especiales para transportar información suplementaria en retorno a la aplicación habilitada para mensajería, o para hacer sugerencias y peticiones a la función leer de CMC sobre la manera de convertir o interpretar diversos atributos y partes de cuerpo en el mensaje recibido. Además de poder tratar mensajes extraños, la interfaz CMC local debe tener convenios sobre los retornos de errores del sistema de mensajería subyacente. La cuestión es determinar si esos errores del sistema de mensajería se pasan o no a la aplicación habilitada para mensajería CMC, y, si la respuesta es afirmativa, en qué forma: ¿se convierten a un error CMC de acuerdo con un convenio común, o se dejan en su forma de base?

En caso de error del sistema de mensajería subyacente se puede proceder como sigue:

- El error se sustituye por CMC_E_FAILURE en las llamadas, y no se tiene en cuenta en los demás casos.
- Si está presente, el error bruto se coloca en una extensión CMC de error X.400 especial.
- El error se convierte a una extensión CMC normalizada para errores de comunicación.

B.2.5.4.4 Extensiones CMC suplementarias

Se podrían emplear extensiones para muchas cosas. A continuación se presentan algunos ejemplos.

- Extensiones para seleccionar cuál de varios sistemas de mensajería subyacentes se va a utilizar.
- Extensiones para configurar y establecer el entorno que la CMC y el sistema de mensajería subyacente van a utilizar.
- Extensiones para añadir atributos X.400 suplementarios que no forman parte del conjunto básico utilizado por CMC.
- Extensiones para determinar, en base a ciertos factores, qué partes de cuerpo X.400 se van a utilizar.
- Extensiones para indicar a la función leer de CMC cómo tratar los mensajes entrantes no básicos.
- Extensiones que se pasan con el sistema de mensajería para indicar al receptor cómo tratarlo.

La utilización de extensiones plantea una serie de cuestiones. La primera es la de determinar si son realmente necesarias. Un establecimiento básico podría o debería no tener que recurrir a ninguna extensión, sino tratar solamente los requisitos mínimos de las llamadas CMC simples de una manera aceptable. Las diversas implementaciones pueden tratar específicamente requisitos locales y, de ese modo, adaptar sus acciones y comportamiento.

Hay muchas extensiones CMC posibles y también muchas maneras de caracterizar esas extensiones. Hay extensiones que son necesarias para adaptar la CMC versión 1.0 local a su sistema de mensajería subyacente local o incluso para seleccionar cuál de sus sistemas de mensajería subyacentes habrá de utilizar.

Estas extensiones se caracterizan como locales. Otras extensiones se pueden caracterizar como especiales, pues conciernen a la utilización de características que no son de la CMC (por lo menos en el caso de la CMC versión 1.0). Estas extensiones se necesitan para asegurar la interoperabilidad.

En el caso de la Recomendación X.400, hay extensiones que son básicas y por tanto genéricas, pues se necesitan para el tratamiento de la mensajería X.400. Las extensiones X.400 genéricas tratarían los retornos de resultados y errores X.400. Estas extensiones genéricas pudieran incluso generalizarse aún más para que comprendieran los retornos de resultados y errores de otros servicios de mensajería diferentes de X.400.

Las indicaciones de errores de llamadas X.400 pueden retornarse a la aplicación habilitada para mensajería que utiliza las llamadas CMC, mediante el empleo de una extensión CMC específica. Esto es esencial para los clientes de llamadas CMC que necesitan una recuperación tras el error y para ello tienen que averiguar donde falló la llamada y no quedarse sin saber esta circunstancia.

Otras extensiones CMC X.400 genéricas tratarían la prioridad de mensajes y la petición de informes de entrega y/o de no entrega. Otras extensiones podrían transportar las restricciones relativas al certificado del usuario y a la memoria de mensajes X.400. El tipo (de mensaje) del mensaje CMC transporta identificadores, tales como identificadores de objetos, que permiten al emisor del mensaje CMC especificar cualquier forma de mensaje X.400. Si las funciones CMC o el sistema X.400 subyacente no soportan ese tipo particular de mensaje X.400 (o requieren específicamente que no se use ese tipo), se necesitaría otra extensión "genérica" para transportar en retorno, al usuario de la llamada CMC, esa información de error. Cuestiones similares plantea la adición de una extensión especial para seleccionar la parte de cuerpo X.400 apropiada para el tipo (de añadidura) de la añadidura CMC.

Pueden utilizarse otras posibles extensiones para ayudar al lado receptor a tratar los mensajes entrantes, resolviendo ciertas cuestiones, por ejemplo si los informes deben descartarse, o no leerse pero comunicarse de una manera especial. La transferencia de extremo a extremo de extensiones (o de su información) para asistir al receptor en el tratamiento del mensaje es un tema demasiado lejano para ser objeto de consideración inmediata.

B.2.5.5 Perfil de correspondencia básico para la CMC simple (CMC 1.0)

Cuando se utiliza el sistema de tratamiento de mensajes X.400 como el sistema de mensajería subyacente, todas las implementaciones CMC versión 1.0 tienen que soportar el envío y la recepción de mensajes básicos. Un perfil básico proporciona esta funcionalidad.

Cada implementación CMC que utiliza el método del "perfil básico" tiene que poder enviar (CMC enviar) cualquier mensaje que requiera solamente el formato IPM básico y utilice las dos partes de cuerpo X.400 recomendadas para añadiduras (y nota textual) cuando se requiera. Asimismo, las aplicaciones habilitadas para mensajería X.400 que utilizan partes de cuerpo más avanzadas tienen que poder aceptar, en su lugar, el conjunto de perfiles básicos. Las implementaciones CMC que reciben (CMC leer) mensajes entregados por la Recomendación X.400 tienen que poder leer y tratar correctamente esos mensajes de "perfil básico". Todos los demás mensajes X.400 pueden ser rechazados.

Las siguientes reglas deben aplicarse al perfil básico. Si el mensaje entrante o la estructura del mensaje saliente no se ajustan a un patrón normalizado, el mensaje debe descartarse y, facultativamente, se podrá retornar un aviso de advertencia o un error a la aplicación habilitada para mensajería. De manera similar, cuando los parámetros de entrada no concuerdan con el conjunto admisible o no pueden ser tratados por el servicio de mensajería subyacente, deberá retornarse un error (o, si esto no fuera posible, un aviso de advertencia) a la aplicación habilitada para mensajería.

El resto de esta subcláusula describe la correspondencia básica mínima requerida para la utilización simple de los servicios de mensajería X.400. La utilización de la totalidad de los servicios de mensajes X.400 que están disponibles puede proporcionarse mediante más extensiones adicionales o facultativas que no han sido incluidas en este perfil básico. Por tanto, debe considerarse la interoperabilidad (si es factible) entre el perfil básico y el perfil ampliado las extensiones.

NOTA – Se utiliza un convenio especial de denominación cuando se hace referencia a los diversos campos definidos por X.400. Cada campo se designa por el nombre de la norma seguido del nombre del campo tal como aparece en la norma. Si el campo en cuestión está definido dentro de un campo, se utiliza el carácter "." para indicar esto. Por ejemplo: X.420.heading.authorizing-users.

B.2.5.5.1 Correspondencia de CMC_recipient

CMC_recipient (recibiente CMC) se hace corresponder con un X.411.ORName utilizando la representación textual de dirección O/R para uso por las personas definida en ISO/CEI 10021-2:1990/Amd.1, Annex F. Pueden utilizarse todas las formas legales de ORName definidas en X.411.

Son correspondencias específicas:

CMC_recipient.name

corresponde a un X.411.ORName.directory-name si la implementación soporta el nombre de directorio; de lo contrario, no se tiene en cuenta.

CMC_recipient.name_type

se supone que es INDIVIDUAL en salida de X.400 a CMC.

CMC_recipient.address

corresponde a X.411.ORName.ORAddress.

CMC_recipient.role

Si su valor es CMC_ROLE_ORIGINATOR, corresponde a X.411.OriginatorName.ORAddress. Corresponde también a X.420.Heading.originator cuando el mensaje es un IPM o IPN.

Si su valor es CMC_ROLE_TO, corresponde a X.411.RecipientName.ORAddress. Corresponde también a X.420.Heading.primary-recipients cuando el mensaje es un IPM o IPN.

Si su valor es CMC_ROLE_CC, corresponde a X.411.RecipientName.ORAddress. Corresponde también a X.420.Heading.copy-recipients cuando el mensaje es un IPM o IPN.

Si su valor es CMC_ROLE_BCC, corresponde a X.411.RecipientName.ORAddress. Corresponde también a X.420.Heading.blind-copy-recipients cuando el mensaje es un IPM o IPN.

Si su valor es CMC_ROLE_AUTHORIZING USER, corresponde a X.420.Heading.authorizing-users cuando el mensaje es un IPM o IPN.

CMC_recipient.recip_flags

se examinan para determinar cuál es la última.

CMC_recipient.recip_extensions

no se tienen en cuenta.

B.2.5.5.2 Correspondencia CMC_message

CMC_message (mensaje CMC) se hace corresponder con la base de información asociada con una operación X.413 de depósito o de captura.

Son correspondencias específicas:

CMC_message.message_reference

corresponde a X.413.entry-sequence-number

CMC_message.message_type

corresponde a X.413.entryType; donde "CMC:IPM" = delivered-message, "CMC:REPORT" = delivered-report, y "CMC:IPM" = returned-content con la nueva extensión CMC_X_X400_MSG_PARENT para identificar que éste es un mensaje anidado.

CMC_message.subject

corresponde a X.413.Content de X.420.heading.subject.

CMC_message.time_sent

es NULL en **cmc_send()**, o corresponde a X.413.Message-submission-time en **cmc_read()**.

CMC_message.text_note

es NULL, o en **cmc_send()** y si CMC_TEXT_NOTE_AS_FILE está fijada, corresponde a la primera parte de cuerpo de X.413.Content de X.420.body.ia5text.data, o en **cmc_read()** corresponde a la primera parte cuerpo ia5text disponible.

El X.420.body.ia5text.repertoire no se tiene en cuenta.

CMC_message.recipient

corresponde a X.411.RecipientName AND X.420.heading.originator, usuarios autorizantes, recibientes primarios, recibientes de copia, y recibientes de copia ciega de acuerdo con el valor fijado a CMC_recipient.role.

CMC_message.attachment

corresponde a X.420.body.BodyPart. Cada añadidura se hace corresponder a la parte de cuerpo correspondiente. Véase B.2.5.5.4 "Correspondencia de CMC_attachment".

CMC_message.message_flags

se fija de acuerdo con X.413.EntryStatus para CMC_SUM_READ y CMC_SUM_UNSENT, comprobando para determinar el último elemento para CMC_SUM_LAST_ELEMENT, y CMC_MSG_TEXT_NOTE_AS_FILE para el tratamiento de la primera añadidura y de la primera parte de cuerpo ia5text.

CMC_message.message_extensions

es NULL o facultativamente retorna la extensión CMC_X_X400_MSG_PARENT.

B.2.5.5.3 Correspondencia de CMC_message_summary

CMC_message_summary (sumario de mensaje CMC) se hace corresponder con los parámetros asociados con una operación X.413 de listar de sumarizar.

Son correspondencias específicas:

CMC_message_summary.message_reference

corresponde a X.413.entry-sequence-number

CMC_message_summary.message_type

corresponde a X.413.entryType; donde "CMC:IPM" = delivered-message, "CMC:REPORT" = delivered-report, y "CMC:IPM" = returned-content.

CMC_message_summary.subject

corresponde a X.413.Content de X.420.heading.subject.

CMC_message_summary.time_sent

corresponde a X.413.Message-submission-time.

CMC_message_summary.byte_length

corresponde a X.413.Content-length.

CMC_message_summary.originator

corresponde a X.413.Originator-name.

CMC_message_summary_flags

se fija de acuerdo con X.413.EntryStatus y/o el último elemento.

CMC_message_summary.message_summary_extensions

es NULL u opcionalmente retorna la extensión CMC_X_X400_MSG_REPORT.

B.2.5.5.4 Correspondencia de CMC_attachment

CMC_attachment (añadidura CMC) corresponde a partes de cuerpo del mensaje asociadas con una operación X.413 de depósito o de captura. Si el mensaje CMC tiene una nota textual, ésta se utiliza como la primera parte de cuerpo ia5text. Cada añadidura se hace corresponder con una parte de cuerpo X.420.

Son correspondencias específicas:

CMC_attachment.attach_title

es NULL o corresponde a X.420.body.ExternallyDefinedBodyPart.data.dataValueDescriptor.

CMC_attachment.attach_type

es NULL o corresponde a OID (*object identifier*, identificador de objeto) de una X.420.body.ExternallyDefinedBodyPart.data.directReference.

Un OID de tipo CMC_ATT_TEXT se hace corresponder al tipo incorporado u OID de una X.420.body.IA5TextBodyPart. El campo IA5TextBodyPart.data.repertoire no se utiliza.

Un tipo CMC_ATT_BINARY se hace corresponder al tipo incorporado u OID de una X.420.body.BilaterallyDefinedBodyPart.

Otro attach_type corresponde a X.420.body.ExternallyDefinedBodyPart. Los campos del tipo EXTERNAL se hacen corresponder como sigue:

- parámetro (facultativo) no se utiliza;
- datos de referencia directa corresponde al OID especificado;
- datos de referencia indirecta (facultativo) no se utiliza;
- descriptor de valor de datos (facultativo) no se utiliza;
- codificación se fija a arbitraria.

CMC_attachment.attach_filename

corresponde al nombre de fichero externo de la implementación y no se pasa a X.400. El contenido del fichero se almacena como los datos de las partes de cuerpo X.400 correspondientes.

CMC_attachment.attach_flags

su correspondencia se establece de acuerdo con el propietario del nombre de fichero de añadidura y con el último elemento de una añadidura.

CMC_attachment.attach_extensions.

NULL.

B.2.5.6 Correspondencias de funciones CMC

La mayor parte de las funciones CMC se hacen corresponder con una operación ROSE que contiene una operación X.413. Los identificadores de invocación utilizados en una operación ROSE tienen que ser un número único. Pueden ser generados por la implementación. No se utilizan identificadores enlazados. Si algunas extensiones o tipos de mensajes no están soportados en este perfil básico, se retorna un error a las funciones CMC, y se descartarán o no se tendrán en cuenta los correspondientes mensajes.

B.2.5.6.1 CMC Tratar (act on)

cmc_act_on() corresponde a un sobre ROSE que contiene una operación X.413 de suprimir. La operación de Suprimir tiene la siguiente estructura:

[information-base-type, items (choice of selector or sequence-number)]

El tipo de base de información es X.413.store-message (por defecto). Se supone que los ítems son números X.413.EntrySequenceNumber suministrados por la referencia de mensaje.

Correspondencia de parámetros

```
CMC_return_code
cmc_act_on(
CMC_session_id              session,              local session id
CMC_message_reference       *message_reference,   X.413.EntrySequenceNumber
CMC_enum                    operation,            supports CMC_ACT_ON_DELETE
                                                  only.(due to underlying X.400
                                                  operations)
CMC_flags                   act_on_flags,         NULL or ignored
CMC_ui_id                   ui_id,                NULL or ignored
CMC_extension               *act_on_extensions    NULL or CMC_X_X400_ERROR on
                                                  output
);
```

Comentarios adicionales

Ninguno.

B.2.5.6.2 CMC Liberar (free)

cmc_free() no necesita correspondencia con llamadas X.400.

Correspondencia de parámetros

```
CMC_return_code
cmc_free(
CMC_buffer                  memory
);
```

Comentarios adicionales

Ninguno.

B.2.5.6.3 CMC Listar (list)

cmc_list() corresponde a un sobre ROSE que contiene una operación X.413 de listar o sumarizar. Las operaciones X.413 de listar y sumarizar tienen la siguiente estructura:

[information-base-type, selector, (requested-attributes or summary-request)]

El tipo de base de información es X.413.store-message (por defecto). Se supone que los ítems son números X.413.EntrySequenceNumber suministrados por referencia de mensaje.

Correspondencia de parámetros

```
CMC_return_code
cmc_list(
   CMC_session_id         session,               local session id
   CMC_string             message_type,          list filter on entryType
   CMC_flags              list_flags,            UNREAD, REF_ONLY, COUNT_ONLY
   CMC_message_reference  *seed,                 X.413.EntrySequenceNumber
   CMC_uint32             *count,                Counter
   CMC_ui_id              ui_id,                 NULL
   CMC_message_summary    **result,              CMC:IPM or CMC: REPORT or OID
   CMC_extension          *list_extensions       Null or CMC_X_X400_ERROR
);
```

Comentarios adicionales

1) No se utilizan inscripciones vástagos (facultativas).

2) Si se proporciona una semilla (*seed*) en la llamada, puede utilizarse para especificar una gama: se selecciona una gama FROM de números secuenciales y se utiliza la semilla proporcionada. La gama TO (facultativa) de números secuenciales no se utiliza.

3) Si el parámetro tipo de mensaje CMC está especificado o la bandera CMC_LIST_UNREAD_ONLY está fijada, se utiliza un filtro. Cuando se satisfacen las dos condiciones, se utiliza el operador AND. Si la bandera CMC_LIST_UNREAD_ONLY está fijada, un ítem de filtro se fija a "Not Item Equality EntryStatus Value (processed)". Si se ha especificado el tipo de mensaje, un elemento de filtro se fija a "Item Equality EntryType Value (message OR report)".

4) Se especifica un límite si el parámetro cuenta de lista CMC no es cero. Si es cero, la cuenta de lista corresponde directamente al valor entero del límite.

5) No se utiliza contraordenación (*override*) (facultativa).

6) En la operación de listar se retornan los siguientes atributos:

   id-att-parent-sequence-number

   id-att-entry-type

   id-att-originator-name

   id-att-content-length

   id-att-message-submission-time

   id-att-subject

   id-att-content-type

B.2.5.6.4 CMC Terminar sesión (logoff)

cmc_logoff() corresponde a un sobre ROSE que contiene una operación X.413 de desvincular. La operación X.413 de desvincular no tiene argumentos, resultado, ni error.

Correspondencia de parámetros

```
CMC_return_code
cmc_logoff(
   CMC_session_id         session,               local session id
   CMC_ui_id              ui_id,                 NULL
   CMC_flags              logoff_flags,          NULL
   CMC_extension          *logoff_extensions     NULL
);
```

Comentarios adicionales

Ninguno.

B.2.5.6.5 CMC Establecer sesión (logon)

cmc_logon() corresponde a un sobre ROSE que contiene una operación X.413 de vincular. La operación X.413 de vincular tiene la siguiente estructura:

[initiator-name, initiator-credentials, security-context, fetch-restrictions, ms-configuration-request]

Correspondencia de parámetros

```
CMC_return_code
cmc_logon(
   CMC_string             service,               NULL, or local path service
   CMC_string             user,                  that access the MS textual
                                                 form of initiator-name,
                                                 see B.2.4.4.1 "Mapping of
                                                 CMC_recipient"
   CMC_string             password,              X.411.initiator-
                                                 credentials.simple
   CMC_object_identifier  character_set,         Password NULL or local return
                                                 from Query
   CMC_ui_id              ui_id,                 Configuration NULL
   CMC_uint16             caller_cmc_version,    local version number v1.0
   CMC_flags              logon_flags,           NULL or is not used
   CMC_session_id         *session,              local session id
   CMC_extension          *logon_extensions      NULL, or CMC_X_X400_ERROR
);
```

Comentarios adicionales

Los siguientes elementos facultativos no se utilizan:

a) Contexto de seguridad MS.

b) Restricción de capturar.

c) Petición de configuración de MS.

B.2.5.6.6 CMC Consultar (look up)

cmc_look_up() no requiere correspondencias ni llamadas X.400. Esta función no es obligatoria para el soporte del servicio de mensajería X.400.

Correspondencia de parámetros

```
CMC_return_code
cmc_look_up(
   CMC_session_id         session                local session id
   CMC_recipient          *recipient_in          see mapping of CMC_recipient
   CMC_flags              look_up_flags          all zero, or local
                                                 implementations
```

```
    CMC_ui_id              ui_id                  NULL
    CMC_uint32             *count                 output from local
                                                  implementation
    CMC_recipient          **recipient_out        see mapping of CMC_recipient
    CMC_extension          *look_up_extensions    NULL
);
```

Comentarios adicionales

Ninguno.

B.2.5.6.7 CMC Leer (read)

cmc_read() corresponde a un sobre ROSE que contiene una operación X.413 de capturar. La operación X.413 de capturar tiene la siguiente estructura:

[information-base-type, item choice of search (set of seq#) or precise (sequence#), requested_attributes]

El tipo de base de información es X.413.store-message (por defecto). Se supone que los ítems son números X.413.EntrySequenceNumber suministrados por la referencia de mensaje.

Correspondencia de parámetros

```
CMC_return_code
cmc_read(
    CMC_session_id         session,               local session id
    CMC_message_reference  *message_reference,    NULL for first UNREAD, or
                                                  X.413.EntrySequenceNumber
    CMC_flags              read_flags,            cannot support
                                                  CMC_DO_NOT_MARK_AS_READ
    CMC_message            **message              pointer to stored-message,
                                                  see mapping of CMC_message
    CMC_ui_id              ui_id                  NULL
    CMC_extension          *read_extensions       NULL, or CMC_X_X400_ERROR
);
```

Comentarios adicionales

Si está especificado el parámetro referencia de mensaje CMC, se selecciona una captura precisa (*precise fetch*). De lo contrario, se selecciona una captura con búsqueda (*search fetch*)

Caso de la captura precisa:

La referencia de mensaje suministrada puede utilizarse como el número secuencial MS.

Caso de la captura con búsqueda:

No se utilizan inscripciones vástagos (facultativas).

Se selecciona una gama FROM de números secuenciales y se utiliza el número secuencial "0".

La gama TO (facultativa) de números secuenciales no se utiliza.

Se especifica un filtro si la bandera CMC_READ_FIRST_UNREAD_ONLY está fijada. El ítem de filtro se fija a "Not Item Equality EntryStatus Value (processed)".

Límite (facultativo) no se utiliza.

Contraorden (*override*) (facultativa) no se utiliza.

Se retornarán los siguientes atributos:

id-att-parent-sequence-number

id-att-entry-type

id-att-message-submission-time

id-att-content-type (IPM, IPN, definido externamente, etc.)

B.2.5.6.8 CMC Enviar (send)

cmc_send() corresponde a un sobre ROSE que contiene una operación X.413 de depósito. La operación X.413 de depósito tiene la siguiente estructura:

[envelope(MessageSubmissionEnvelope) with an IPM content(Content)]

Correspondencia de parámetros

```
CMC_return_code
cmc_send(
    CMC_session_id        session,              local session id
    CMC_message           *message,             see mapping of CMC_message
    CMC_flags             send_flags,           always zero
    CMC_ui_id             ui_id,                NULL
    CMC_extension         *send_extensions      NULL, or CMC_X_COM_PRIORITY,
                                                or CMC_X_X400_ERROR
);
```

Comentarios adicionales

X.411 MessageSubmissionEnvelope (sobre P3):

El originador se llena con el ORName del originador (véase la correspondencia de ORName). Si no se especifica el originador, se utilizará el ORName empleado en **cmc_logon()**.

Tipo de información codificada original (facultativa) no se utiliza.

Tipo de contenido se fija a Built-in y el tipo es IPM-84.

Identificador de contenido (facultativo) no se utiliza.

Prioridad (facultativa) es normal (por defecto), o se toma la que corresponde de CMC_X_COM_PRIORITY si está presente.

Indicador por cada mensaje (facultativo) utiliza los valores por defecto.

disclosure-of-recipient está prohibida (por defecto);

implicit-conversion-prohibited está permitida (por defecto);

alternate-recipient allowed está prohibido (por defecto);

content-return-requested no está solicitado (por defecto).

Hora de entrega diferida (facultativa) no se utiliza.

Extensiones (facultativas) no se utilizan.

Las direcciones ORName de todos los recibientes han sido llenadas (véase la correspondencia de CMC_recipient). Después de llenarse cada ORName de recibiente, se fijan los campos siguientes:

Petición de informe del originador es "informe de no entrega" (por defecto).

Conversión explícita (facultativa) está prohibida (por defecto).

Extensiones (facultativas) no se utilizan.

El contenido del sobre P3 es un mensaje IPM P2 formado por un sobre P2 y una o más partes de cuerpo.

X.420.IPM.heading (sobre P2):

El usuario se llena con el ORName del originador (véase la correspondencia de CMC_recipient).

El identificador relativo de usuario se forma agregando al final de la cadena "CMC:IPM" el mismo ID de invocación generado para el sobre ROSE.

Para recibientes primarios, recibientes de copia y recibientes de copia ciega no se utilizan peticiones de notificación (facultativas), ni respuesta solicitada (facultativa).

IPM a que se responde (facultativo) no se utiliza.

IPM obsoletos (facultativos) no se utiliza.

IPM relacionados (facultativos) no se utiliza.

El asunto de mensaje corresponde directamente al campo asunto de mensaje CMC.

Hora de expiración (facultativa) no se utiliza.

Hora de respuesta (facultativa) no se utiliza.

Recibientes de respuesta (facultativos) no se utiliza.

Importancia (facultativa) es normal (por defecto).

Sensibilidad (facultativa) no se utiliza.

Reenvío automático (facultativo) no se utiliza.

Extensiones (facultativas) no se utiliza.

X.420.IPM.body:

Si el mensaje CMC tiene una nota textual, se utiliza como la primera parte de cuerpo IA5Text. Cuando se crea una parte de cuerpo IA5Text, el campo de repertorio (facultativo) no se tiene en cuenta.

Para todas las otras añadiduras CMC se crea unas parte de cuerpo X.400 correspondiente. Si el tipo CMC_attachment es CMC_ATT_TEXT, se hace corresponder con una parte de cuerpo IA5Text. Si el tipo es CMC_ATT_BINARY, se hace corresponder con una parte de cuerpo definida bilateralmente. Para otros tipos CMC, se utiliza el valor OID suministrado y se crea una parte de cuerpo definida externamente (EDBP).

B.2.5.6.9 CMC Enviar documentos (send documents)

cmc_send_documents() corresponde a sobres ROSE cada uno de los cuales contiene una operación X.413 de MS vincular, una operación X.413 de MS depósito, y una operación X.413 de MS desvincular. Esta es una secuencia combinada de las funciones **cmc_logon()**, **cmc_send()** y **cmc_logoff()**.

Correspondencia de parámetros

```
CMC_return_code
cmc_send_documents(
    CMC_string          recipient_addresses,   X.411.envelope.recipients and
                                               X.420 authorizing users,
                                               primary, copy, and blind copy
                                               recipients; see mapping of
                                               CMC_recipient
    CMC_string          subject,               X.420.heading.subject
    CMC_string          text_note,             first X.420.body.IA5TextBo-
                                               dyPart.data
    CMC_flags           send_doc_flags,        caller-supplied flags
    CMC_string          file_paths,            local filenames, not passed
                                               in X.400 message
    CMC_string          attach_titles,         X.420.EDBP.data.dataVa-
                                               lueDescriptor
    CMC_string          delimiter,             delimiter character
    CMC_ui_id           ui_id,                 NULL
);
```

Comentarios adicionales

Ninguno.

B.2.5.6.10 CMC Indagar configuración (query configuration)

La llamada a esta función no requiere ninguna correspondencia ni llamadas X.400. Se limita a retornar la configuración del ítem especificado.

Correspondencia de parámetros

```
CMC_return_code
cmc_query_configuration(
   CMC_session_id        session,               local session id
   CMC_enum              item,                  local implementation
   CMC_buffer            reference,             local implementation
   CMC_extension         *config_extensions     NULL
);
```

Comentarios adicionales

Ninguno.

Anexo C

Ejemplos de programación

C.1 Ejemplos de programación

Esta Recomendación ofrece los siguientes ejemplos de programación.

C.1.1 Funciones de indagar configuración (query configuration), establecer sesión (logon) y terminar sesión (logoff)

```
/* local variables used */

CMC_return_code      Status;
CMC_boolean          UI_available;
CMC_session_id       Session;

/* find out if UI is available with this implementation before starting */

Status = cmc_query_configuration(
          NULL,                              /* No session id.              */
          CMC_CONFIG_UI_AVAIL,               /* See if UI is available.     */
          &UI_available,                     /* Return value.               */
          NULL);                             /* No extensions.              */
       /* error handling */

/* Log on to system using UI */

Status = cmc_logon(
          NULL,                              /* Default service.            */
          NULL,                              /* Prompt for username.        */
          NULL,                              /* Prompt for password.        */
          NULL,                              /* Default Character set.      */
          (CMC_ui_id)NULL,                   /* Default UI ID.              */
          CMC_VERSION,                       /* Version 1 CMC calls.        */
          CMC_LOGON_UI_ALLOWED |             /* Full logon UI.              */
          CMC_ERROR_UI_ALLOWED,              /* Use UI to display errors.   */
          &Session,                          /* Returned session id.        */
          NULL);                             /* No extensions.              */
       /* error handling */

/* Do various CMC calls */

/* Log off from the implementation */

Status = cmc_logoff(

          Session,                           /* Session id.                 */
          (CMC_ui_id)NULL,                   /* No UI will be used.         */
          0,                                 /* No flags.                   */
          NULL);                             /* No extensions.              */
       /* error handling */
```

C.1.2 Funciones de enviar (send) y enviar documentos (send documents)

```
/* local variables used */

CMC_attachment       Attach;
CMC_session_id       Session;
CMC_message          Message;
CMC_recipient        Recip[2];
CMC_return_code      Status;

/* Build recipient list with two recipients. Add one "To" recipient. */

Recip[0].name           = "Bob Weaver";          /* Send to Bob Weaver.      */
Recip[0].name_type      = CMC_TYPE_INDIVIDUAL;   /* Bob’s a person.          */
Recip[0].address        = NULL;                  /* Look_up Bob’s address.   */
```

```
Recip[0].role           = CMC_ROLE_TO;           /* He's a "To" recipient.   */
Recip[0].flags          = 0;                     /* Not the last element.    */
Recip[0].extensions     = NULL;                  /* No recipient extensions. */

/* Add one "Cc" recipient. */

Recip[1].name           = "Mary Yu";             /* Send to Mary Yu.         */
Recip[1].name_type      = CMC_TYPE_INDIVIDUAL;   /* Mary's a person.         */
Recip[1].address        = NULL;                  /* Look_up Mary's address.  */
Recip[1].role           = CMC_ROLE_CC;           /* She's a "Cc" recipient.  */
Recip[1].flags          = CMC_RECIP_LAST_ELEMENT;/* Last recipient element.  */
Recip[1].extensions     = NULL;                  /* No recipient extensions. */

/* Attach a file. */

Attach.attach_title     = "stock.wks";           /* Original file name.      */
Attach.attach_typ       = NULL;                  /* No specific type.        */
Attach.attach_filename  = "tmp22.tmp";           /* File to attach.          */
Attach.attach_flags     = CMC_ATT_LAST_ELEMENT;  /* Last attachment.         */
Attach.attach_extensions= NULL;                  /* No attach. extensions.   */

/* Put it together in the message structure. */

Message.message_reference  = NULL;            /* Ignored on cmc_send calls.  */
Message.message_type       = NULL;            /* Interpersonal message type. */
Message.subject            = "Stock";         /* Message subject.            */
Message.time_sent          = NULL;            /* Ignored on cmc_send calls.  */
Message.text_note          = "Time to buy";   /* Message note.               */
Message.recipients         = Recip;           /* Message recipients.         */
Message.attachments        = &Attach;         /* Message attachments.        */
Message.message_flags      = 0;               /* No flags.                   */
Message.message_extensions = NULL;            /* No message extensions.      */

/* Send the message! */

Status = cmc_send(
          Session,                 /* Session id. - set with logon call.      */
          &Message,                /* Message structure.                      */
          0,                       /* No flags.                               */
          (CMC_ui_id)NULL,         /* No UI will be used.                     */
          NULL);                   /* No extensions.                          */
       /* error handling */

/* Now do the same thing with the send documents call and UI */

Status = cmc_send_documents(
          "to:Bob Weaver,cc:Mary Yu",        /* Message recipients.           */
          "Stock",                           /* Message subject.              */
          "Time to buy",                     /* Message note.                 */
          CMC_LOGON_UI_ALLOWED |
          CMC_SEND_UI_REQUESTED |
          CMC_ERROR_UI_ALLOWED,              /* Flags (allow various UI's).   */
          "stock.wks",                       /* File to attach.               */
          "tmp22.tmp",                       /* File name to carry on attach. */
          ",",                               /* Multi-value delimiter.        */
          NULL);                             /* Default UI ID.                */
       /* error handling */
```

C.1.3 Funciones de listar (list), leer (read), y suprimir (delete) el primer mensaje no leído

```
/* local variables used */

CMC_message_summary  *pMsgSummary;
CMC_message          *pMessage;
CMC_uint32           iCount;

/* read the first unread message and delete it */

iCount = 5;

Status = cmc_list(
          Session,                 /* Session id.                          */
          NULL,                    /* List ALL message types.              */
          CMC_LIST_UNREAD_ONLY,    /* Get only unread messages.            */
          NULL,                    /* Starting at the top.                 */
          &iCount,                 /* Input/Output message count.          */
          (CMC_ui_id)NULL,         /* No UI will be used.                  */
          &pMsgSummary,            /* Return message summary list.         */
          NULL);                   /* No extensions.                       */
       /* error handling */

Status = cmc_read(
          Session,                            /* Session id.                */
          pMsgSummary[0]->message_reference,  /* Message to read.           */
          CMC_MSG_AND_ATT_HDRS_ONLY,          /* don't get attach files.    */
          &pMessage,                          /* Returned message.          */
          (CMC_ui_id)NULL,                    /* No UI.                     */
          NULL);                              /* No extensions.             */
       /* error handling */

Status = cmc_act_on(
          Session,                            /* Session id.                */
          pMsgSummary[0]->message_reference,  /* Message to delete.         */
          CMC_ACT_ON_DELETE,                  /* Message to read.           */
          0,                                  /* no flags.                  */
          (CMC_ui_id)NULL,                    /* No UI.                     */
          NULL);                              /* No extensions.             */
       /* error handling */

/* free the memory returned by the implementation */

Status = cmc_free(pMsgSummary);
Status = cmc_free(pMessage);

/* do the same thing without the list call, since the read call can get the first
unread mail message */

Status = cmc_read(
          Session,                            /* Session id.                */
          NULL,                               /* Read the first message.    */
          CMC_READ_FIRST_UNREAD_MESSAGE |     /* get first unread msg.      */
          CMC_MSG_AND_ATT_HDRS_ONLY,          /* don't get attach files.    */
          &pMessage,                          /* Returned message.          */
          (CMC_ui_id)NULL,                    /* No UI.                     */
          NULL);                              /* No extensions.             */
       /* error handling */

Status = cmc_act_on(
          Session,                            /* Session id.                */
          pMessage->message_reference,        /* message to delete.         */
          CMC_ACT_ON_DELETE,                  /* Message to read.           */
          0,                                  /* no flags.                  */
          (CMC_ui_id)NULL,                    /* No UI.                     */
          NULL);                              /* No extensions.             */
       /* error handling */

/* free the memory returned by the implementation */

Status = cmc_free(pMessage);
```

C.1.4 Consultar sobre un recibiente específico y obtener sus detalles

```
/* local variables used */

CMC_session_id       Session;
CMC_recipient        *pRecipient;
CMC_recipient        Recip;
CMC_return_code      Status;

/* look up a name to pick correct recipient */

Recip.name               = "Bob Stack";           /* Send to Bob Weaver.      */
Recip.name_type          = CMC_TYPE_INDIVIDUAL;   /* Bob's a person.          */
Recip.address            = NULL;                  /* Look_up Bob's address.   */
Recip.role               = NULL;                  /* Role not used.           */
Recip.recip_flags        = 0;                     /* No flags.                */
Recip.recip_extensions   = NULL;                  /* No recipient extensions. */

Status = cmc_look_up(
          Session,                          /* Session id.                  */
          &Recip,                           /* Name to look up.             */
          CMC_LOOKUP_RESOLVE_UI |           /* Disambiguate using UI.       */
          CMC_ERROR_UI_ALLOWED,             /* Display errors using UI.     */
          (CMC_ui_id)NULL,                  /* Default UI ID.               */
          1,                                /* Only want 1 back.            */
          pRecipient,                       /* Returned recipient ptr.      */
          NULL);                            /* No extensions.               */

/* Display details stored for this recipient */

Status = cmc_look_up(
          Session,                          /* Session id.                  */
          pRecipient,                       /* Name to get details on.      */
          CMC_LOOKUP_DETAILS_UI |           /* Show details UI.             */
          CMC_ERROR_UI_ALLOWED,             /* Display errors using UI.     */
          (CMC_ui_id)NULL,                  /* Default UI ID.               */
          0,                                /* No limit on return count.    */
          NULL,                             /* No records returned.         */
          NULL);                            /* No extensions.               */

/* free the memory returned by the implementation */

cmc_free(pRecipient);
```

C.1.5 Utilización de extensiones

```
/* local variables used */

CMC_return_code      Status;
CMC_session_id       Session;
CMC_extension        Extension;
CMC_X_COM_support    Supported[2];
CMC_uint16           index;

/* find out if the common extension set is supported, but I don't need
   COM_X_CONFIG_DATA support */

Supported[0].item_code =        CMC_XS_COM;
Supported[0].flags =            0;

Supported[1].item_code =        CMC_X_COM_CONFIG_DATA;
Supported[1].flags =            CMC_X_COM_SUP_EXCLUDE;

Extension.item_code =           CMC_X_COM_SUPPORT_EXT;
Extension.item_data =           2;
Extension.item_reference =      Supported;
Extension.extension_flags =     CMC_EXT_LAST_ELEMENT;
```

```
Status = cmc_query_configuration(
          NULL,                              /* No session id.              */
          CMC_CONFIG_UI_AVAIL,               /* See if UI is available.     */
          &UI_available,                     /* Return value.               */
          &Extension);                       /* Pass in extensions.         */
       /* error handling */
if (Supported[0].flags & CMC_X_COM_NOT_SUPPORTED)
       return FALSE;  /* common extensions I need are not available */

/* Log on to system and get the data extensions for this session */

Supported[0].item_code =        CMC_XS_COM;
Supported[0].flags =            0;

Supported[1].item_code =        CMC_X_COM_CONFIG_DATA;
Supported[1].flags =            CMC_X_COM_SUP_EXCLUDE;

Extension.item_code =           CMC_X_COM_SUPPORT_EXT;
Extension.item_data =           2;
Extension.item_reference =      Supported;
Extension.extension_flags =     CMC_EXT_REQUIRED | CMC_EXT_LAST_ELEMENT;

Status = cmc_logon(
          NULL,                              /* Default service.            */
          NULL,                              /* Prompt for username.        */
          NULL,                              /* Prompt for password.        */
          NULL,                              /* Default Character set.      */
          (CMC_ui_id)NULL,                   /* Default UI ID.              */
          CMC_VERSION,                       /* Version 1 CMC calls.        */
          CMC_LOGON_UI_ALLOWED |             /* Full logon UI.              */
          CMC_ERROR_UI_ALLOWED,              /* Use UI to display errors.   */
          &Session,                          /* Returned session id.        */
          &Extension);                       /* Logon extensions.           */
       /* error handling */
if (Supported[0].flags & CMC_X_COM_NOT_SUPPORTED)
       return FALSE;  /* common extensions I need are not available */
       /* the common data extensions will be used for this session */

/*  example of how to free data returned from the CMC implementation in
    function output extensions. */

for (index = 0; ; index++)  {
       if (Extensions[index].extension_flags & CMC_EXT_OUTPUT) {
          if (cmc_free(Extensions[index].item_reference) != CMC_success)      {
             /* Handle unexpected error here */
             }
             }
          (Extensions[index].extension_flags & CMC_EXT_LAST_ELEMENT)
             break;
       }

/* Do various CMC calls */

/* Log off from the implementation */

Status = cmc_logoff(
          Session,                 /* Session id.                            */
          (CMC_ui_id)NULL,         /* No UI will be used.                    */
          0,                       /* No flags.                              */
          NULL);                   /* No extensions.                         */
       /* error handling */
```

C.1.6 cmc_bind_implementation

```
CMC_return_code   Status = CMC_SUCCESS;
CMC_boolean       UI_available;
CMC_session_id    Session;
HINSTANCE         hlibCMC = (HINSTANCE)NULL;
```

```
CMC_P_BIND_IMPLEMENTATION lpfnCMCBindImplementation = NULL;
CMC_dispatch_table           *pDispatchTable = NULL;
CMC_object_handle            root_object_handle = CMC_NULL_HANDLE;
extern CMC_guid              selected_implementation_name;

if (!(hlibCMC = LoadLibrary ("CMC.DLL")))
       {
       /* error handling */
       }

if (!(lpfnCMCBindImplementation =
   (CMC_P_BIND_IMPLEMENTATION)GetProcAddress (hlibCMC, "cmc_bind_implement-
    ation")))
       {
       /* error handling */
       }

/* Call into a selected CMC Manager to bind to specific CMC implementation.*/

Status = lpfnCMCBindImplementation(selected_implementation_name,
                             &pDispatchTable,
                             NULL);

if (pDispatchTable == NULL)
       {
       /* error handling */
       }

/* find out if UI is available with this implementation before starting.
   Mixing calls, be careful. */

Status = pDispatchTable->cmc_query_configuration(
       NULL,                            /* No session id.              */
       CMC_CONFIG_UI_AVAIL,             /* See if UI is available.     */
       &UI_available,                   /* Return value.               */
       NULL);                           /* No extensions.              */
   /* error handling */

/* Log on to system using UI */

Status = pDispatchTable->cmc_logon(
       NULL,                            /* Default service.            */
       NULL,                            /* Prompt for username.        */
       NULL,                            /* Prompt for password.        */
       NULL,                            /* Default Character set.      */
       (CMC_ui_id)NULL,                 /* Default UI ID.              */
       CMC_VERSION,                     /* Version 1 CMC calls.        */
       CMC_LOGON_UI_ALLOWED |           /* Full logon UI.              */
       CMC_ERROR_UI_ALLOWED,            /* Use UI to display errors.   */
       &Session,                        /* Returned session id.        */
       NULL);                           /* No extensions.              */
   /* error handling */

/* Make calls into specific CMC implementation. */

Status = pDispatchTable->cmc_get_root_handle(Session,
                             &root_object_handle,
                             NULL);
   /* error handling */

/* Log off from the implementation */

Status = pDispatchTable->cmc_logoff(
       Session,              /* Session id.            */
       (CMC_ui_id)NULL,      /* No UI will be used.    */
       0,                    /* No flags.              */
       NULL);                /* No extensions.         */
   /* error handling */
```

```
   /* Let implementation free the dispatch table it gave us. */

   pDispatchTable->cmc_free(pDispatchTable);

   /* Unbind from the CMC implementation. Cleans up storage
      created in CMC Manager and/or CMC implementation other
      than the CMC_dispatch_table. */

   cmc_unbind_implementation(selected_implementation_name,
                              NULL);

      /* error handling */

   /* Free the DLL instance. */

   FreeLibrary (hlibCMC);
```

C.2 Ejemplo de cmc_bind_implementation

```
CMC_return_code
cmc_bind_implementation(
   CMC_guid                 implementation_name,
   CMC_dispatch_table       **dispatch_table,
   CMC_extension            *cmc_bind_extensions
)
{
SCODE sc = SUCCESS_SUCCESS;
CMC_dispatch_table *pDispatchTable=NULL;

   pDispatchTable = (CMC_dispatch_table *)(calloc(1, sizeof(CMC_dispatch_table)));
   if (pDispatchTable == NULL)
      return CMC_E_INSUFFICIENT_MEMORY;

   else

      /* Store local for later cmc_free processing. */

   /* Populate the Dispatch Table. */

   pDispatchTable->cmc_send = cmc_send;
   pDispatchTable->cmc_send_documents = cmc_send_documents;
   pDispatchTable->cmc_act_on = cmc_act_on;
   pDispatchTable->cmc_list = cmc_list;
   pDispatchTable->cmc_read = cmc_read;
   pDispatchTable->cmc_look_up = cmc_look_up;
   pDispatchTable->cmc_free = cmc_free;
   pDispatchTable->cmc_logoff = cmc_logoff;
   pDispatchTable->cmc_logon = cmc_logon;
   pDispatchTable->cmc_query_configuration = cmc_query_configuration;
   pDispatchTable->cmc_add_object = cmc_add_object;
   pDispatchTable->cmc_add_properties = cmc_add_properties;
   pDispatchTable->cmc_commit_object = cmc_commit_object;
   pDispatchTable->cmc_copy_object_handle = cmc_copy_object_handle;
   pDispatchTable->cmc_create_derived_message_object =
                     cmc_create_derived_message_object;
   pDispatchTable->cmc_delete_objects = cmc_delete_objects;
   pDispatchTable->cmc_delete_properties = cmc_delete_properties;
   pDispatchTable->cmc_get_root_handle = cmc_get_root_handle;
   pDispatchTable->cmc_identifier_to_name = cmc_identifier_to_name;
   pDispatchTable->cmc_list_contained_properties = cmc_list_contained_properties;
   pDispatchTable->cmc_list_number_matched = cmc_list_number_matched;
   pDispatchTable->cmc_list_objects = cmc_list_objects;
   pDispatchTable->cmc_list_properties = cmc_list_properties;
   pDispatchTable->cmc_name_to_identifier = cmc_name_to_identifier;
   pDispatchTable->cmc_open_cursor = cmc_open_cursor;
   pDispatchTable->cmc_open_object_handle = cmc_open_object_handle;
   pDispatchTable->cmc_read_cursor = cmc_read_cursor;
   pDispatchTable->cmc_read_properties = cmc_read_properties;
   pDispatchTable->cmc_read_property_costs = cmc_read_property_costs;
   pDispatchTable->cmc_restore_object = cmc_restore_object;
   pDispatchTable->cmc_save_object = cmc_save_object;
   pDispatchTable->cmc_send_message_object = cmc_send_message_object;
   pDispatchTable->cmc_update_cursor_position = cmc_update_cursor_position;
   pDispatchTable->cmc_update_cursor_position_with_seed =
                     cmc_update_cursor_position_with_seed;
```

```
    pDispatchTable->cmc_check_event = cmc_check_event;
    pDispatchTable->cmc_register_event = cmc_register_event;
    pDispatchTable->cmc_unregister_event = cmc_unregister_event;
    pDispatchTable->cmc_call_callbacks = cmc_call_callbacks;
    pDispatchTable->cmc_export_stream = cmc_export_stream;
    pDispatchTable->cmc_import_file_to_stream = cmc_import_file_to_stream;
    pDispatchTable->cmc_open_stream = cmc_open_stream;
    pDispatchTable->cmc_read_stream = cmc_read_stream;
    pDispatchTable->cmc_seek_stream = cmc_seek_stream;
    pDispatchTable->cmc_write_stream = cmc_write_stream;
    pDispatchTable->cmc_get_last_error = cmc_get_last_error;

    /* Load the output variable. */

    *dispatch_table = pDispatchTable;

    return CMC_SUCCESS;
}
```

C.3 Composición de un mensaje

```
    #define NUM_RECIP_PROPS              4
    #define NUM_MESSAGE_PROPS            4
    #define NUM_CONTENT_PROPS            6

    #define RECIP_NAME_INDEX             0
    #define RECIP_ADDRESS_INDEX          1
    #define RECIP_ROLE_INDEX             2
    #define RECIP_TYPE_INDEX             3

    #define MSG_TYPE_INDEX               0
    #define MSG_PRIORITY_INDEX           1
    #define MSG_SUBJECT_INDEX            2
    #define MSG_ROLE_INDEX               3

    #define CONTENT_CHARSET_INDEX        0
    #define CONTENT_INFORMATION_INDEX    1
    #define CONTENT_SIZE_INDEX           2
    #define CONTENT_TITLE_INDEX          3
    #define CONTENT_ITEMNUM_INDEX        4
    #define CONTENT_ITEMTYPE_INDEX       5

    CMC_return_code          Status = CMC_SUCCESS;
    extern                   CMC_session_id Session;
    extern                   CMC_dispatch_table *pDispatchTable;
    CMC_object_handle        Message = CMC_NULL_HANDLE;
    CMC_object_handle        Recipient = CMC_NULL_HANDLE;
    CMC_object_handle        ContentItem = CMC_NULL_HANDLE;
    CMC_string               error_buf = NULL;
    CMC_property             RecipientProps[NUM_RECIP_PROPS];
    CMC_property             MessageProps[NUM_MESSAGE_PROPS];
    CMC_property             ContentProps[NUM_CONTENT_PROPS];
    CMC_opaque_data          MessageBody;
    CMC_CHAR                 MsgBuffer[MAX_BODY_LEN];

    /* Load Recipient Property Structure. */

    RecipientProps[RECIP_NAME_INDEX].property_id = CMC_PV_RECIPIENT_NAME;
    RecipientProps[RECIP_NAME_INDEX].type = CMC_string;
    RecipientProps[RECIP_NAME_INDEX].value.CMC_pv_string =
                                  "Pierre Peret";

    RecipientProps[RECIP_ADDRESS_INDEX].property_id = CMC_PV_RECIPIENT_ADDRESS;
    RecipientProps[RECIP_ADDRESS_INDEX].type = CMC_string;
    RecipientProps[RECIP_ADDRESS_INDEX].value.CMC_pv_string =
                                  "uunet!p.peret@A205.bull.com!USENET";

    RecipientProps[RECIP_ROLE_INDEX].property_id = CMC_PV_RECIPIENT_ROLE;
    RecipientProps[RECIP_ROLE_INDEX].type = CMC_enum;
    RecipientProps[RECIP_ROLE_INDEX].value.CMC_pv_enumerated =
                                  CMC_RECIPIENT_ROLE_TO;

    RecipientProps[RECIP_TYPE_INDEX].property_id = CMC_PV_RECIPIENT_TYPE;
    RecipientProps[RECIP_TYPE_INDEX].type = CMC_enum;
    RecipientProps[RECIP_TYPE_INDEX].value.CMC_pv_enumerated =
                                  CMC_RCT_INDIVIDUAL;
```

```
/* Load Message Property Structure. */

MessageProps[MSG_TYPE_INDEX].property_id = CMC_PV_MESSAGE_TYPE;
MessageProps[MSG_TYPE_INDEX].type = CMC_enum;
MessageProps[MSG_TYPE_INDEX].value.CMC_pv_enumerated =
                                CMC_MT_IPM;

MessageProps[MSG_PRIORITY_INDEX].property_id = CMC_PV_MESSAGE_PRIORITY;
MessageProps[MSG_PRIORITY_INDEX].type = CMC_enum;
MessageProps[MSG_PRIORITY_INDEX].value.CMC_pv_enumerated =
                                CMC_PRIORITY_NORMAL;

MessageProps[MSG_SUBJECT_INDEX].property_id = CMC_PV_MESSAGE_SUBJECT;
MessageProps[MSG_SUBJECT_INDEX].type = CMC_string;
MessageProps[MSG_SUBJECT_INDEX].value.CMC_pv_string =
                                "Hey Pierre, don't forget";

MessageProps[MSG_ROLE_INDEX].property_id = CMC_PV_MESSAGE_ROLE;
MessageProps[MSG_ROLE_INDEX].type = CMC_enum;
MessageProps[MSG_ROLE_INDEX].value.CMC_pv_enumerated =
                                CMC_MESSAGE_ROLE_ORIGINAL;

/* Load Message Content Item Property Structure. */

ContentProps[CONTENT_CHARSET_INDEX].property_id =
                                CMC_PV_CONTENT_ITEM_CHARACTER_SET;
ContentProps[CONTENT_CHARSET_INDEX].type = CMC_guid;
ContentProps[CONTENT_CHARSET_INDEX].value.CMC_pv_guid =
                                CMC_CHARSET_1252;

ContentProps[CONTENT_INFORMATION_INDEX].property_id =
                                CMC_PV_CONTENT_ITEM_CONTENT_INFORMATION;
ContentProps[CONTENT_INFORMATION_INDEX].type = CMC_opaque_data;

strcpy(MsgBuffer, "to FOCUS on the API");

MessageBody.size = strlen(MsgBuffer) + 1;
MessageBody.data = (CMC_byte *)calloc(1, strlen(MsgBuffer) + 1);

ContentProps[CONTENT_INFORMATION_INDEX].value.CMC_pv_opaque_data.
                                data = MessageBody.data;
ContentProps[CONTENT_INFORMATION_INDEX].value.CMC_pv_opaque_data.
                                size = MessageBody.size;

ContentProps[CONTENT_SIZE_INDEX].property_id =
                                CMC_PV_CONTENT_ITEM_SIZE;
ContentProps[CONTENT_SIZE_INDEX].type = CMC_uint32;
ContentProps[CONTENT_SIZE_INDEX].value.CMC_pv_uint32 =
                                MessageBody.size;

ContentProps[CONTENT_TITLE_INDEX].property_id =
                                CMC_PV_CONTENT_ITEM_TITLE;
ContentProps[CONTENT_TITLE_INDEX].type = CMC_string;
ContentProps[CONTENT_TITLE_INDEX].value.CMC_pv_string =
                                "Message Body";

ContentProps[CONTENT_ITEMNUM_INDEX].property_id =
                                CMC_PV_CONTENT_ITEM_ITEM_NUMBER;
ContentProps[CONTENT_ITEMNUM_INDEX].type = CMC_uint32;
ContentProps[CONTENT_ITEMNUM_INDEX].value.CMC_pv_uint32 = 0;

ContentProps[CONTENT_ITEMTYPE_INDEX].property_id =
                                CMC_PV_CONTENT_ITEM_ITEM_TYPE;
ContentProps[CONTENT_ITEMTYPE_INDEX].type = CMC_enum;
ContentProps[CONTENT_ITEMTYPE_INDEX].value.CMC_pv_enumerated =
                                CMC_IT_NOTE;

/* Create a Recipient Object. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                     &Recipient,
                                     CMC_TYPE_OC_RECIPIENT,
                                     NULL);
   /* error handling */
```

```
/* Populate the Recipient Object with some properties. */

Status = pDispatchTable->cmc_add_properties(Recipient,
                                     NUM_RECIP_PROPS,
                                     &RecipientProps,
                                     NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                     Recipient,
                                     &error_buf,
                                     NULL);

   /* NOTE - The add properties extension parameter in
      the cmc_add_properties call above could have been
      used for obtaining per property error information. */
   /* error handling */
   }

/* Create a Message Object. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                     &Message,
                                     CMC_TYPE_OC_MESSAGE,
                                     NULL);
   /* error handling */

/* Populate the Message Object with some properties. */

Status = pDispatchTable->cmc_add_properties(Message,
                                     NUM_MESSAGE_PROPS,
                                     &MessageProps,
                                     NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                     Message,
                                     &error_buf,
                                     NULL);

   /* NOTE – The add properties extension parameter in
      the cmc_add_properties call above could have been
      used for obtaining per property error information. */
   /* error handling */
   }

/* Create a Content Item Object. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                     &ContentItem,
                                     CMC_TYPE_OC_CONTENT_ITEM,
                                     NULL);
   /* error handling */

/* Populate the Content Item Object with some properties. */

Status = pDispatchTable->cmc_add_properties(ContentItem,
                                     NUM_CONTENT_PROPS,
                                     &ContentProps,
                                     NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                     ContentItem,
                                     &error_buf,
                                     NULL);

   /* NOTE - The add properties extension parameter in
      the cmc_add_properties call above could have been
      used for obtaining per property error information. */
   /* error handling */
   }
```

```
/* Now move the Recipient and Content Item Objects into the
   Message Object. */

Status = pDispatchTable->cmc_copy_object(Message,
                                         Recipient,
                                         &Message,
                                         NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                         Message,
                                         &error_buf,
                                         NULL);
   /* error handling */
   }

Status = pDispatchTable->cmc_copy_object(Message,
                                         ContentItem,
                                         &Message,
                                         NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                         Message,
                                         &error_buf,
                                         NULL);
   /* error handling */
   }

/* Try sending the message. */

Status = pDispatchTable->cmc_send_message_object(Message,
                                         NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                         Message,
                                         &error_buf,
                                         NULL);
   /* error handling */
   }

/* Cleanup. */

cfree(MessageBody.data);

pDispatchTable->cmc_free(ContentItem);
pDispatchTable->cmc_free(Recipient);
pDispatchTable->cmc_free(Message);
pDispatchTable->cmc_free(error_buf);
```

C.4 Comprobación para determinar la presencia de nuevos mensajes

```
CMC_return_code              Status = CMC_SUCCESS;
extern                       CMC_session_id Session;
extern                       CMC_dispatch_table *pDispatchTable;
CMC_object_handle            root_object_handle = NULL;
CMC_cursor_handle            RootCursor = CMC_NULL_HANDLE;
CMC_cursor_handle            FolderCursor = CMC_NULL_HANDLE;
CMC_cursor_restriction       RootRestriction;
CMC_string                   error_buf = NULL;
CMC_enum                     InboxTypeFlag = CMC_MCT_INBOX;
CMC_uint32                   NewMessageFlag = CMC_EVENT_NEW_MESSAGES;
CMC_uint32                   MinTimeOut = 0;
CMC_new_message_check_data   CheckData;
CMC_new_message_callback_data*CallbackData = NULL;
CMC_callback                 NewMessageCallBack;
```

```
/* Get the Root Object Handle. */

Status = pDispatchTable->cmc_get_root_handle(
                                    Session,
                                    &root_object_handle,
                                    NULL);
   /* error handling */

/* Open a cursor on the Root Container. Start by
   setting up a restriction to find the Inbox Folder.
   Assuming the Existence of single Inbox Folder. */

RootRestriction.type = CMC_RESTRICTION_CONTENT;
RootRestriction.cr.restriction_content.logical = CMC_LOGICAL_EQ;
RootRestriction.cr.restriction_content.property =
                  CMC_PV_MESSAGE_CONTAINER_TYPE;

RootRestriction.cr.restriction_content.property_value =
                  (CMC_buffer)&InboxTypeFlag;
RootRestriction.Property_extensions = NULL;

Status = pDispatchTable->cmc_open_cursor(
                         root_object_handle,
                         &RootRestriction,
                         0,
                         NULL,
                         &RootCursor,
                         NULL);
   /* error handling */

/* Register to poll for new messages */

/* Sets Container for which new messages are checked to the Inbox */

CheckData.number_containers = 1;
CheckData.containers = &RootCursor;

Status = pDispatchTable->cmc_register(
                         Session,
                         NewMessageFlag,
                         NULL,
                         (CMC_buffer) &CheckData,
                         NULL);
   /* error handling */

/* Check for New Messages */

Status = pDispatchTable->cmc_check_event(
                         Session,
                         NewMessageFlag,
                         MinTimeOut,
                         (CMC_buffer) &CheckData,
                         CallBackData,
                         NULL);
   /* error handling */

if (CheckData != NULL) {
   printf("You have new mail!\n");
}

/* Set up callback for new messages */

Status = pDispatchTable->cmc_register_event(
                             Session,
                             NewMessageFlag,
                             NewMessageCallBack,
                             (CMC_buffer) &CheckData,
                             NULL);
   /* error handling */

/* Force callback function */

Status = pDispatchTable->cmc_call_callbacks(
                             Session,
                             NewMessageFlag,
                             NULL);
   /* error handling */
```

```
   /* Unregister for the new messages event */

   Status = pDispatchTable->cmc_unregister_event(
                                Session,
                                NewMessageFlag,
                                NewMessageCallBack,
                                (CMC_buffer) &CheckData,
                                NULL);
      /* error handling */

   /* Cleanup. */

   pDispatchTable->cmc_free(RootCursor);
   pDispatchTable->cmc_free(FolderCursor);
   pDispatchTable->cmc_free(hFolder);
   pDispatchTable->cmc_free(Inbox);

CMC_return code
(*NewMessageCallBack)(      CMC_session_id     session,
                            CMC_event          event,
                            CMC_buffer         callback_data,
                            CMC_buffer         register_data,
                            CMC_extension      *callback_extensions)
{

   printf("You have new mail!\n");

   pDispatchTable->cmc_free(callback_data);
   pDispatchTable->cmc_free(register_data);

   return(CMC_SUCCESS);
}
```

C.5 Archivo de un mensaje

```
   #define NUM_RECIP_PROPS                    4
   #define NUM_MESSAGE_PROPS                  5
   #define NUM_CONTENT_PROPS                  6

   #define RECIP_NAME_INDEX                   0
   #define RECIP_ADDRESS_INDEX                1
   #define RECIP_ROLE_INDEX                   2
   #define RECIP_TYPE_INDEX                   3

   #define MSG_TYPE_INDEX                     0
   #define MSG_PRIORITY_INDEX                 1
   #define MSG_SUBJECT_INDEX                  2
   #define MSG_ROLE_INDEX                     3
   #define MSG_CLIENT_MSG_STATUS_INDEX        4

   #define CONTENT_CHARSET_INDEX              0
   #define CONTENT_INFORMATION_INDEX          1
   #define CONTENT_SIZE_INDEX                 2
   #define CONTENT_TITLE_INDEX                3
   #define CONTENT_ITEMNUM_INDEX              4
   #define CONTENT_ITEMTYPE_INDEX             5

   CMC_return_code          Status = CMC_SUCCESS;
   extern                   CMC_session_id Session;
   extern                   CMC_dispatch_table *pDispatchTable;
   CMC_object_handle        root_object_handle = CMC_NULL_HANDLE;
   CMC_object_handle        hFolder = CMC_NULL_HANDLE;
   CMC_object_handle        Message = CMC_NULL_HANDLE;
   CMC_object_handle        Recipient = CMC_NULL_HANDLE;
   CMC_object_handle        ContentItem = CMC_NULL_HANDLE;
   CMC_cursor_handle        RootCursor = CMC_NULL_HANDLE;
   CMC_cursor_restriction   RootRestriction;
   CMC_sint32               FolderCount = 1; /* Assume 1 Drafts Folder. */
   CMC_string               error_buf = NULL;
   CMC_enum                 DraftsTypeFlag = CMC_MCT_INBOX;
   CMC_property             RecipientProps[NUM_RECIP_PROPS];
   CMC_property             MessageProps[NUM_MESSAGE_PROPS];
   CMC_property             ContentProps[NUM_CONTENT_PROPS];
   CMC_opaque_data          MessageBody;
   CMC_CHAR                 MsgBuffer[MAX_BODY_LEN];
```

```
/* Get the Root Object Handle. */

Status = pDispatchTable->cmc_get_root_handle(Session,
                                &root_object_handle,
                                NULL);
   /* error handling */

/* Open a cursor on the Root Container. Start by
   setting up a restriction to find the Drafts Folder.
   Assuming the Existence of the Drafts Folder. */

RootRestriction.type = CMC_RESTRICTION_CONTENT;
RootRestriction.cr.restriction_content.logical = CMC_LOGICAL_EQ;
RootRestriction.cr.restriction_content.property =
                  CMC_PV_MESSAGE_CONTAINER_TYPE;
RootRestriction.cr.restriction_content.property_value =
                  (CMC_buffer)&DraftsTypeFlag;
RootRestriction.Property_extensions = NULL;

Status = pDispatchTable->cmc_open_cursor(root_object_handle,
                         &RootRestriction,
                         0,
                         NULL,
                         &RootCursor,
                         NULL);
   /* error handling */

Status = pDispatchTable->cmc_list_objects(&RootCursor,
                         &FolderCount,
                         &hFolder,
                         NULL);
   /* error handling */

/* Create and populate a Message. */

/* Load Recipient Property Structure. */

RecipientProps[RECIP_NAME_INDEX].property_id = CMC_PV_RECIPIENT_NAME;
RecipientProps[RECIP_NAME_INDEX].type = CMC_string;
RecipientProps[RECIP_NAME_INDEX].value.CMC_pv_string =
                              "Pierre Peret";
RecipientProps[RECIP_ADDRESS_INDEX].property_id = CMC_PV_RECIPIENT_ADDRESS;
RecipientProps[RECIP_ADDRESS_INDEX].type = CMC_string;
RecipientProps[RECIP_ADDRESS_INDEX].value.CMC_pv_string =
                              "uunet!p.peret@A205.bull.com!USENET";
RecipientProps[RECIP_ROLE_INDEX].property_id = CMC_PV_RECIPIENT_ROLE;
RecipientProps[RECIP_ROLE_INDEX].type = CMC_enum;
RecipientProps[RECIP_ROLE_INDEX].value.CMC_pv_enumerated =
                              CMC_RECIPIENT_ROLE_TO;
RecipientProps[RECIP_TYPE_INDEX].property_id = CMC_PV_RECIPIENT_TYPE;
RecipientProps[RECIP_TYPE_INDEX].type = CMC_enum;
RecipientProps[RECIP_TYPE_INDEX].value.CMC_pv_enumerated =
                              CMC_RCT_INDIVIDUAL;

/* Load Message Property Structure. */

MessageProps[MSG_TYPE_INDEX].property_id = CMC_PV_MESSAGE_TYPE;
MessageProps[MSG_TYPE_INDEX].type = CMC_enum;
MessageProps[MSG_TYPE_INDEX].value.CMC_pv_enumerated =
                              CMC_MT_IPM;
MessageProps[MSG_PRIORITY_INDEX].property_id = CMC_PV_MESSAGE_PRIORITY;
MessageProps[MSG_PRIORITY_INDEX].type = CMC_enum;
MessageProps[MSG_PRIORITY_INDEX].value.CMC_pv_enumerated =
                              CMC_PRIORITY_NORMAL;
MessageProps[MSG_SUBJECT_INDEX].property_id = CMC_PV_MESSAGE_SUBJECT;
MessageProps[MSG_SUBJECT_INDEX].type = CMC_string;
MessageProps[MSG_SUBJECT_INDEX].value.CMC_pv_string =
                              "Lunch";
MessageProps[MSG_ROLE_INDEX].property_id = CMC_PV_MESSAGE_ROLE;
MessageProps[MSG_ROLE_INDEX].type = CMC_enum;
MessageProps[MSG_ROLE_INDEX].value.CMC_pv_enumerated =
                              CMC_MESSAGE_ROLE_ORIGINAL;
MessageProps[MSG_CLIENT_MSG_STATUS_INDEX].property_id =
                              CMC_PV_MESSAGE_CLIENT_MSG_STATUS;
MessageProps[MSG_CLIENT_MSG_STATUS_INDEX].type = CMC_enum;
MessageProps[MSG_CLIENT_MSG_STATUS_INDEX].value.CMC_pv_enumerated =
                              CMC_MESSAGE_STATUS_DRAFT;
```

```
/* Load Message Content Item Property Structure. */

ContentProps[CONTENT_CHARSET_INDEX].property_id =
                              CMC_PV_CONTENT_ITEM_CHARACTER_SET;
ContentProps[CONTENT_CHARSET_INDEX].type = CMC_guid;
ContentProps[CONTENT_CHARSET_INDEX].value.CMC_pv_guid =
                              CMC_CHARSET_1252;

ContentProps[CONTENT_INFORMATION_INDEX].property_id =
                              CMC_PV_CONTENT_ITEM_CONTENT_INFORMATION;
ContentProps[CONTENT_INFORMATION_INDEX].type = CMC_opaque_data;

strcpy(MsgBuffer, "What time are we leaving for lunch?");

MessageBody.size = strlen(MsgBuffer) + 1;
MessageBody.data = (CMC_byte *)calloc(1, strlen(MsgBuffer) + 1);

ContentProps[CONTENT_INFORMATION_INDEX].value.CMC_pv_opaque_data.
                              data = MessageBody.data;
ContentProps[CONTENT_INFORMATION_INDEX].value.CMC_pv_opaque_data.
                              size = MessageBody.size;

ContentProps[CONTENT_SIZE_INDEX].property_id =
                              CMC_PV_CONTENT_ITEM_SIZE;
ContentProps[CONTENT_SIZE_INDEX].type = CMC_uint32;
ContentProps[CONTENT_SIZE_INDEX].value.CMC_pv_uint32 =
                              MessageBody.size;

ContentProps[CONTENT_TITLE_INDEX].property_id =
                              CMC_PV_CONTENT_ITEM_TITLE;
ContentProps[CONTENT_TITLE_INDEX].type = CMC_string;
ContentProps[CONTENT_TITLE_INDEX].value.CMC_pv_string =
                              "Message Body";

ContentProps[CONTENT_ITEMNUM_INDEX].property_id =
                              CMC_PV_CONTENT_ITEM_ITEM_NUMBER;
ContentProps[CONTENT_ITEMNUM_INDEX].type = CMC_uint32;
ContentProps[CONTENT_ITEMNUM_INDEX].value.CMC_pv_uint32 = 0;

ContentProps[CONTENT_ITEMTYPE_INDEX].property_id =
                              CMC_PV_CONTENT_ITEM_ITEM_TYPE;
ContentProps[CONTENT_ITEMTYPE_INDEX].type = CMC_enum;
ContentProps[CONTENT_ITEMTYPE_INDEX].value.CMC_pv_enumerated =
                              CMC_IT_NOTE;

/* Create a Recipient Object. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                      &Recipient,
                                      CMC_TYPE_OC_RECIPIENT,
                                      NULL);

   /* error handling */

/* Populate the Recipient Object with some properties. */

Status = pDispatchTable->cmc_add_properties(Recipient,
                              NUM_RECIP_PROPS,
                              &RecipientProps,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              Recipient,
                              &error_buf,
                              NULL);

   /*   NOTE - The add properties extension parameter in
        the cmc_add_properties call above could have been
        used for obtaining per property error information. */
   /* error handling */
   }
```

```
/* Create a Message Object. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                    &Message,
                                    CMC_TYPE_OC_MESSAGE,
                                    NULL);
   /* error handling */

/* Populate the Message Object with some properties. */

Status = pDispatchTable->cmc_add_properties(Message,
                              NUM_MESSAGE_PROPS,
                              &MessageProps,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              Message,
                              &error_buf,
                              NULL);
   /*   NOTE - The add properties extension parameter in
        the cmc_add_properties call above could have been
        used for obtaining per property error information. */
   /*   error handling */
   }

/* Create a Content Item Object. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                    &ContentItem,
                                    CMC_TYPE_OC_CONTENT_ITEM,
                                    NULL);
   /* error handling */

/* Populate the Message Object with some properties. */

Status = pDispatchTable->cmc_add_properties(ContentItem,
                              NUM_CONTENT_PROPS,
                              &ContentProps,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              ContentItem,
                              &error_buf,
                              NULL);

   /*   NOTE - The add properties extension parameter in
        the cmc_add_properties call above could have been
        used for obtaining per property error information. */
   /*   error handling */
   }

/* Now move the Recipient and Content Item Objects into the
   Message Object. */

Status = pDispatchTable->cmc_copy_object(Message,
                              Recipient,
                              &Message,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              Message,
                              &error_buf,
                              NULL);
   /* error handling */
   }
```

```
Status = pDispatchTable->cmc_copy_object(Message,
                              ContentItem,
                              &Message,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              Message,
                              &error_buf,
                              NULL);
   /* error handling */
   }

/* Move message into the Drafts Folder. */

Status = pDispatchTable->cmc_copy_object(hFolder,
                              Message,
                              &Message,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              hFolder,
                              &error_buf,
                              NULL);
   /* error handling */
   }

Status = pDispatchTable->cmc_commit_object(Message,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              hFolder,
                              &error_buf,
                              NULL);
   /* error handling */
   }

/* Cleanup. */

cfree(MessageBody.data);
pDispatchTable->cmc_free(ContentItem);
pDispatchTable->cmc_free(Recipient);
pDispatchTable->cmc_free(Message);
pDispatchTable->cmc_free(hFolder);
pDispatchTable->cmc_free(RootCursor);
pDispatchTable->cmc_free(error_buf);
```

C.6 Supresión de un mensaje

```
CMC_return_code          Status = CMC_SUCCESS;
extern                   CMC_session_id Session;
extern                   CMC_object_handle root_object_handle;
extern                   CMC_dispatch_table *pDispatchTable;
extern                   CMC_object_handle hFolder;
extern                   CMC_cursor_handle FolderCursor;
CMC_object_handle        hDeletedFolder = CMC_NULL_HANDLE;
CMC_object_handle        Message = CMC_NULL_HANDLE;
CMC_object_handle        MessageInDeleted = CMC_NULL_HANDLE;
CMC_cursor_restriction   RootRestriction;
CMC_cursor_handle        RootCursor = CMC_NULL_HANDLE;
CMC_sint32               FolderCount = 1; /* Assume 1 Drafts Folder. */
CMC_sint32               MessageCount = 1; /* Assume deletion of 1 entry */
CMC_string               error_buf = NULL;
CMC_enum                 DeletedTypeFlag = CMC_MCT_DELETED;

/* Open a cursor on the Root Container. Start by
   setting up a restriction to find the Deleted Folder.
   Assuming the Existence of the Deleted Folder. */
```

```
RootRestriction.type = CMC_RESTRICTION_CONTENT;
RootRestriction.cr.restriction_content.logical = CMC_LOGICAL_EQ;
RootRestriction.cr.restriction_content.property =
                  CMC_PV_MESSAGE_CONTAINER_TYPE;
RootRestriction.cr.restriction_content.property_value =
                  (CMC_buffer)&DeletedTypeFlag;
RootRestriction.Property_extensions = NULL;

Status = pDispatchTable->cmc_open_cursor(root_object_handle,
                                    &RootRestriction,
                                    0,
                                    NULL,
                                    &RootCursor,
                                    NULL);
   /* error handling */

Status = pDispatchTable->cmc_list_objects(&RootCursor,
                                    &FolderCount,
                                    &hDeletedFolder,
                                    NULL);
   /* error handling */

/* NOTE - Assuming UI code has set a FolderCursor on a Message entry
   in a list box which is the message to delete. */

/* Get the message to be deleted. */

Status = pDispatchTable->cmc_list_objects(&FolderCursor,
                                    &MessageCount,
                                    &Message,
                                    NULL);
   /* error handling */

/* Let’s first move the message to the Deleted Folder. */

Status = pDispatchTable->cmc_copy_object(hDeletedFolder,
                                    Message,
                                    &MessageInDeleted,
                                    NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                    hFolder,
                                    &error_buf,
                                    NULL);
   /* error handling */
   }

Status = pDispatchTable->cmc_commit_object(MessageInDeleted,
                                    NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                    hDeletedFolder,
                                    &error_buf,
                                    NULL);
   /* error handling */
   }

/* Now let’s permanently delete the message from the source folder.
   Invalidates Message handle. */

Status = pDispatchTable->cmc_delete_objects(MessageCount,
                                    &Message,
                                    NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                    Message,
                                    &error_buf,
                                    NULL);
   /* error handling */
   }
```

```
/* Cleanup. */

pDispatchTable->cmc_free(hDeletedFolder);
pDispatchTable->cmc_free(MessageInDeleted);
pDispatchTable->cmc_free(RootCursor);
pDispatchTable->cmc_free(error_buf);
```

C.7 Extracción de un mensaje

```
#define MAX_CACHE              25

CMC_return_code          Status = CMC_SUCCESS;
extern                   CMC_session_id Session;
extern                   CMC_dispatch_table *pDispatchTable;
CMC_object_handle        root_object_handle = NULL;
CMC_object_handle        hFolder = NULL;
CMC_object_handle        Messages = NULL;
CMC_cursor_handle        RootCursor,
CMC_cursor_handle        FolderCursor,
CMC_cursor_restriction   RootRestriction;
CMC_sint32               FolderCount = 1;
CMC_sint32               MessageCount = MAX_CACHE;
CMC_enum                 InboxTypeFlag = CMC_MCT_INBOX;
CMC_enum                 MessageClassFlag = CMC_TYPE_OC_MESSAGE;

/* Get the Root Object Handle. */

Status = pDispatchTable->cmc_get_root_handle(Session,
                                    &root_object_handle,
                                    NULL);
   /* error handling */

/* Open a cursor on the Root Container. Start by
   setting up a restriction to find the Inbox Folder.
   Assuming the Existence of single Inbox Folder. */

RootRestriction.type = CMC_RESTRICTION_CONTENT;
RootRestriction.cr.restriction_content.logical = CMC_LOGICAL_EQ;
RootRestriction.cursor_restriction.restriction_content.property =
                  CMC_PV_MESSAGE_CONTAINER_TYPE;
RootRestriction.cursor_restriction.restriction_content.property_value =
                  (CMC_buffer)&InboxTypeFlag;
RootRestriction.Property_extensions = NULL;

Status = pDispatchTable->cmc_open_cursor(root_object_handle,
                              &RootRestriction,
                              0,
                              NULL,
                              &RootCursor,
                              NULL);
   /* error handling */

Status = pDispatchTable->cmc_list_objects(&RootCursor,
                              &FolderCount,
                              &hFolder,
                              NULL);
   /* error handling */

/* Build a restriction on the Folder. Fetch all messages. */

FolderRestriction.type = CMC_RESTRICTION_CONTENT;
FolderRestriction.cr.restriction_content.logical = CMC_LOGICAL_EQ;
FolderRestriction.cr.restriction_content.property =
                  CMC_PV_OBJECT_CLASS;
FolderRestriction.cr.restriction_content.property_value =
                  (CMC_buffer)&MessageClassFlag;

/* Open a cursor on the Folder. */

Status = pDispatchTable->cmc_open_cursor(hFolder,
                              &FolderRestriction,
                              0,
                              NULL,
                              &FolderCursor,
                              NULL);
   /* error handling */
```

```
/* Enumerate all the messages in the inbox in chunks of MAX_CACHE. */

while (MessageCount != 0)
   {
Status = pDispatchTable->cmc_list_objects(&FolderCursor,
                              &MessageCount,
                              &Messages,
                              NULL);
   /* error handling */

/* Build Property array of desired properties (see composing a
   message example), call cmc_read_properties and Display in
   Listbox for each Message Object returned.

   NOTE - The individual object handles need to be copied by a call to
   cmc_copy_object_handle() prior to invoking cmc_free() on this pointer. */

pDispatchTable->cmc_free(Messages);

}

/* Cleanup. */

pDispatchTable->cmc_free(RootCursor);
pDispatchTable->cmc_free(FolderCursor);
pDispatchTable->cmc_free(hFolder);
```

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie B	Medios de expresión: definiciones, símbolos, clasificación
Serie C	Estadísticas generales de telecomunicaciones
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Transmisiones de señales radiofónicas, de televisión y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	RGT y mantenimiento de redes: sistemas de transmisión, circuitos telefónicos, telegrafía, facsímil y circuitos arrendados internacionales
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	**Redes de datos y comunicación entre sistemas abiertos**
Serie Z	Lenguajes de programación