UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T X.446

(08/97)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE X: RÉSEAUX POUR DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

Systèmes de messagerie

Interface de programme d'application d'appel commun de messagerie

Recommandation UIT-T X.446

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX POUR DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

RÉSEAUX PUBLICS POUR DONNÉES	X.1–X.199
Services et fonctionnalités	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalisation et commutation	X.50–X.89
Aspects réseau	X.90–X.149
Maintenance	X.150–X.179
Dispositions administratives	X.180–X.199
INTERCONNEXION DES SYSTÈMES OUVERTS	X.200–X.299
Modèle et notation	X.200–X.209
Définitions des services	X.210–X.219
Spécifications des protocoles en mode connexion	X.220–X.229
Spécifications des protocoles en mode sans connexion	X.230–X.239
Formulaires PICS	X.240–X.259
Identification des protocoles	X.260–X.269
Protocoles de sécurité	X.270–X.279
Objets gérés de couche	X.280–X.289
Tests de conformité	X.290–X.299
INTERFONCTIONNEMENT DES RÉSEAUX	X.300–X.399
Généralités	X.300–X.349
Systèmes de transmission de données par satellite	X.350–X.399
SYSTÈMES DE MESSAGERIE	**X.400–X.499**
ANNUAIRE	X.500–X.599
RÉSEAUTAGE OSI ET ASPECTS SYSTÈMES	X.600–X.699
Réseautage	X.600–X.629
Efficacité	X.630–X.649
Dénomination, adressage et enregistrement	X.650–X.679
Notation de syntaxe abstraite numéro un (ASN.1)	X.680–X.699
GESTION OSI	X.700–X.799
Cadre général et architecture de la gestion-systèmes	X.700–X.709
Service et protocole de communication de gestion	X.710–X.719
Structure de l'information de gestion	X.720–X.729
Fonctions de gestion	X.730–X.799
SÉCURITÉ	X.800–X.849
APPLICATIONS OSI	X.850–X.899
Engagement, concomitance et rétablissement	X.850–X.859
Traitement transactionnel	X.860–X.879
Opérations distantes	X.880–X.899
TRAITEMENT OUVERT RÉPARTI	X.900–X.999

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

RECOMMANDATION UIT-T X.446

INTERFACE DE PROGRAMME D'APPLICATION D'APPEL COMMUN DE MESSAGERIE

Résumé

La présente Recommandation spécifie une interface d'appel simple, à travers laquelle des applications tributaires d'une messagerie peuvent invoquer les services du système MHS en passant par une interface de programmation normalisée. Cette Recommandation, élaborée de concert avec l'association XAPI, définit l'interface de programmation d'application mise en œuvre pour le système MHS par les principaux vendeurs et fournisseurs de services du monde.

Source

La Recommandation UIT-T X.446, élaborée par la Commission d'études 7 (1997-2000) de l'UIT-T, a été approuvée le 9 août 1997 selon la procédure définie dans la Résolution n° 1 de la CMNT.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

La Conférence mondiale de normalisation des télécommunications (CMNT), qui se réunit tous les quatre ans, détermine les thèmes d'études à traiter par les Commissions d'études de l'UIT-T lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution n° 1 de la CMNT.

Dans certains secteurs de la technologie de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT avait/n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Page

1 Introduction 1
1.1 But 1
1.2 Aperçu général 1
1.3 Terminologie 2
1.3.1 Définitions 2
1.3.2 Abréviations 2
1.4 Références normatives 3
1.4.1 Recommandations | Normes internationales identiques 3
1.4.2 Paires de Recommandations | Normes internationales équivalentes par leur contenu technique 3
1.4.3 Références supplémentaires 3
1.5 Niveaux 4
1.6 Conventions de nom en langage C 4
2 Architecture CMC 5
2.1 Modèle fonctionnel 5
2.2 Modèle informatique 6
2.2.1 Interfaces 6
2.2.2 Session 7
2.2.3 Prise en charge de caractères étendus 7
2.2.4 Notification d'événement 7
2.2.5 Extensions 8
2.3 Modèle de configuration 8
2.3.1 Gestionnaire CMC 9
2.3.2 Directives pour les liaisons de plates-formes 10
2.3.3 Interrogation de l'information concernant la configuration 10
2.4 Modèle objet 10
2.4.1 Composants du modèle 10
3 Classes d'objets CMC 13
3.1 Classes d'objets de l'API CMC 13
3.1.1 Carnet d'adresses (*Address book*) 14
3.1.2 Article de contenu (*Content item*) 14
3.1.3 Liste de distribution (*Distribution list*) 14
3.1.4 Message (*Message*) 14
3.1.5 Conteneur de messages (*Message Container*) 15
3.1.6 Information par destinataire (*Per Recipient Information*) 15
3.1.7 Conteneur de profil (*Profile Container*) 16
3.1.8 Destinataire (*Recipient*) 16
3.1.9 Compte rendu (*Report*) 16
3.1.10 Conteneur racine (*Root Container*) 17
4 Structures de données 17
4.1 Types de données de base 18
4.2 Types de données tableau 18
4.3 Attachement (*Attachment*) 20
4.4 Booléen (*Boolean*) 21
4.5 Tampon (*Buffer*) 21
4.6 Structures de données de rappel automatique (*Callback Data Structures*) 21
4.7 Chaîne avec comptage (*Counted String*) 23
4.8 Descripteur opaque de curseur (*Cursor Handle*) 24
4.9 Contrainte de curseur (*Cursor Restriction*) 24
4.10 Clé de tri de curseur (*Cursor sort key*) 26

Page

4.11 Table de distribution (*Dispatch Table*) ... 27
4.12 Enuméré (*Enumerated*) ... 33
4.13 Evénements (*Events*) ... 33
4.14 Extension (*Extension*) ... 33
4.15 Fanions (*Flags*) ... 34
4.16 Identificateur Guid (*Guid*) ... 35
4.17 Identificateur (*Identifier*) ... 35
4.18 Date et heure ISO (*date and time*) ... 35
4.19 Message (*Message*) ... 36
4.20 Référence de message (*Message Référence*) ... 38
4.21 Résumé de message (*Message summary*) ... 39
4.22 Nom (*Name*) ... 39
4.23 Descripteur opaque d'objet (*Object handle*) ... 40
4.24 Identificateur d'objet (*Object identifier*) ... 40
4.25 Données opaques (*Opaque data*) ... 40
4.26 Propriété (*Property*) ... 41
4.27 Destinataire (*Recipient*) ... 42
4.28 Compte rendu (*Report*) ... 43
4.29 Code retour (*Return code*) ... 44
4.30 Identificateur de session (*Session Id*) ... 44
4.31 Descripteur opaque de flux (*Stream handle*) ... 44
4.32 Chaîne (*String*) ... 45
4.33 Temps (*Time*) ... 45
4.34 Identificateur d'interface utilisateur (*User interface Id*) ... 46

5 Propriétés d'objet ... 46
5.1 Propriétés de l'objet carnet d'adresses ... 56
5.1.1 Enfant autorisé (*Child allowed*) ... 56
5.1.2 Commentaire (*Comment*) ... 56
5.1.3 Emplacement (*Location*) ... 57
5.1.4 Nom (*Name*) ... 57
5.1.5 Classe d'objets (*Object class*) ... 57
5.1.6 Parent (*Parent*) ... 58
5.1.7 Nom de serveur (*Server name*) ... 58
5.1.8 Partagé (*Shared*) ... 58
5.1.9 Type (*Type*) ... 58
5.2 Propriétés de l'objet article de contenu ... 59
5.2.1 Jeu de caractères (*Character set*) ... 59
5.2.2 Information de contenu (*Content information*) ... 59
5.2.3 Type de contenu (*Content type*) ... 60
5.2.4 Instant de création (*Create time*) ... 62
5.2.5 Type de codage (*Encoding type*) ... 62
5.2.6 Répertoire de fichier (*File directory*) ... 63
5.2.7 Nom de fichier (*File name*) ... 63
5.2.8 Numéro d'article (*Item number*) ... 64
5.2.9 Type d'article (*Item type*) ... 64
5.2.10 Dernière modification (*Last modified*) ... 64
5.2.11 Classe d'objets (*Object class*) ... 64
5.2.12 Position de rendu (*Render position*) ... 65
5.2.13 Taille (*Size*) ... 65
5.2.14 Titre (*Title*) ... 65
5.3 Propriétés de l'objet liste de distribution ... 66
5.3.1 Adresse (*Address*) ... 66
5.3.2 Commentaire (*Comment*) ... 66
5.3.3 Instant de dernière modification (*Last modification time*) ... 66

Page

5.3.4 Nom (*Name*) ... 66
5.3.5 Classe d'objets (*Object class*) ... 67
5.3.6 Parent (*Parent*) ... 67
5.3.7 Partagé (*Shared*) ... 67
5.4 Propriétés de l'objet message ... 67
5.4.1 Identificateur d'application (*Application Id*) ... 68
5.4.2 Statut de message de l'application (*Application message status*) ... 68
5.4.3 Action automatique (*Auto-action*) ... 68
5.4.4 Instant de remise différée (*Deferred delivery time*) ... 69
5.4.5 Identificateur (*Id*) ... 69
5.4.6 Statut de message en entrée (*In message status*) ... 69
5.4.7 En réponse à (*In reply to*) ... 70
5.4.8 Comptage d'articles (*Item count*) ... 70
5.4.9 Diagnostic de non-acquittement (*NRN diagnostic*) ... 70
5.4.10 Motif de non-acquittement (*NRN reason*) ... 70
5.4.11 Classe d'objets (*Object class*) ... 71
5.4.12 Statut de message en sortie (*Out message status*) ... 71
5.4.13 Priorité (*Priority*) ... 72
5.4.14 Acquittement demandé (*Receipt requested*) ... 72
5.4.15 Type d'acquittement (*Receipt type*) ... 73
5.4.16 Compte rendu demandé (*Report requested*) ... 73
5.4.17 Rôle (*Role*) ... 73
5.4.18 Sensibilité (*Sensitivity*) ... 74
5.4.19 Taille (*Size*) ... 74
5.4.20 Sujet (*Subject*) ... 75
5.4.21 Instant de réception (*Time received*) ... 75
5.4.22 Instant d'émission (*Time sent*) ... 75
5.4.23 Type (*Type*) ... 75
5.5 Propriétés de l'objet conteneur de messages ... 76
5.5.1 Enfant autorisé (*Child allowed*) ... 76
5.5.2 Commentaire (*Comment*) ... 76
5.5.3 Emplacement (*Location*) ... 77
5.5.4 Nom (*Name*) ... 77
5.5.5 Classe d'objets (*Object class*) ... 77
5.5.6 Parent (*Parent*) ... 78
5.5.7 Nom de serveur (*Server name*) ... 78
5.5.8 Partagé (*Shared*) ... 78
5.5.9 Type (*Type*) ... 78
5.6 Propriétés de l'objet information par destinataire ... 79
5.6.1 Commentaire (*Comment*) ... 79
5.6.2 Instant de remise (*Delivery time*) ... 79
5.6.3 Diagnostic (*Diagnostic*) ... 79
5.6.4 Classe d'objets (*Object class*) ... 80
5.6.5 Motif (*Reason*) ... 80
5.6.6 Adresse du destinataire (*Recipient address*) ... 80
5.6.7 Nom du destinataire (*Recipient name*) ... 80
5.6.8 Type (*Type*) ... 81
5.7 Propriétés de l'objet conteneur de profil ... 81
5.7.1 Action automatique (*Auto-Action*) ... 81
5.7.2 Jeu de caractères (*Character Set*) ... 82
5.7.3 Conformité (*Conformance*) ... 82
5.7.4 Service par défaut (*Default Service*) ... 82
5.7.5 Utilisateur par défaut (*Default User*) ... 83
5.7.6 Fin de ligne (*Line Terminator*) ... 83
5.7.7 Classe d'objets (*Object Class*) ... 83
5.7.8 Extensions d'objet prises en charge (*Object Extensions Supported*) ... 83
5.7.9 Objets pris en charge (*Objects Supported*) ... 84
5.7.10 Propriétés prises en charge (*Properties Supported*) ... 84
5.7.11 Extensions de propriété prises en charge (*Property Extensions Supported*) ... 84
5.7.12 Mot de passe exigé (*Required Password*) ... 84
5.7.13 Service exigé (*Required Service*) ... 85

Page

5.7.14 Utilisateur exigé (*Required User*) ... 85
5.7.15 Prise en charge des chaînes avec comptage (*Support Counted Strings*) ... 85
5.7.16 Prise en charge d'absence de marquage en lecture (*Support No Mark As Read*) ... 85
5.7.17 Interface utilisateur disponible (*User Interface Available*) ... 86
5.7.18 Utilisateurs (*Users*) ... 86
5.7.19 Version de la mise en œuvre (*Version of the Implementation*) ... 86
5.7.20 Version de la spécification (*Version of the Specification*) ... 86
5.8 Propriétés d'objets destinataire ... 87
5.8.1 Adresse (*Address*) ... 87
5.8.2 Renvoi de contenu exigé (*Content Return Requested*) ... 87
5.8.3 Nom (*Name*) ... 87
5.8.4 Classe d'objets (*Object Class*) ... 87
5.8.5 Acquittement demandé (*Receipt Requested*) ... 88
5.8.6 Compte rendu demandé (*Report Requested*) ... 88
5.8.7 Fanion de responsabilité (*Responsibility Flag*) ... 89
5.8.8 Rôle (*Role*) ... 89
5.8.9 Type (*Type*) ... 90
5.9 Propriétés de l'objet compte rendu ... 90
5.9.1 Identificateur d'application (*Application Id*) ... 90
5.9.2 Identificateur (*Id*) ... 90
5.9.3 Comptage d'article (*Item Count*) ... 91
5.9.4 Identificateur du système de messagerie (*Messaging System Id*) ... 91
5.9.5 Classe d'objets (*Object Class*) ... 91
5.9.6 Lu (*Read*) ... 91
5.9.7 Taille (*Size*) ... 92
5.9.8 Sujet (*Subject*) ... 92
5.9.9 Identificateur du message sujet (*Subject Message Id*) ... 92
5.9.10 Instant de réception (*Time Received*) ... 92
5.9.11 Instant d'émission (*Time Sent*) ... 93
5.9.12 Non émis (*Unsent*) ... 93
5.10 Propriétés de l'objet conteneur racine ... 93
5.10.1 Enfant autorisé (*Child Allowed*) ... 93
5.10.2 Commentaire (*Comment*) ... 93
5.10.3 Emplacement (*Location*) ... 94
5.10.4 Nom (*Name*) ... 94
5.10.5 Classe d'objets (*Object Class*) ... 94
5.10.6 Partagé (*Shared*) ... 95

6 Interface fonctionnelle ... 95
6.1 Fonctions de l'interface CMC simple ... 95
6.1.1 Emission de message ... 96
6.1.2 Réception de messages ... 100
6.1.3 Consultation de noms ... 106
6.1.4 Administration ... 109
6.2 Fonctions CMC complètes ... 116
6.2.1 Fonctions de liaison ... 117
6.2.2 Fonctions de composition ... 119
6.2.3 Fonctions d'énumération ... 130
6.2.4 Fonctions de notification d'événements ... 144
6.2.5 Fonctions de messagerie ... 149
6.2.6 Fonctions de manipulation de nom ... 152
6.2.7 Fonctions de flux ... 154

7 Codes retour ... 160

8 Conformité ... 175

Annexe A – Résumé des déclarations en langage C ... 177

A.1 Résumé des déclarations en langage C ... 177

Page

Annexe B – Extensions de fournisseurs CMC 220
B.1 Extensions CMC faites par les fournisseurs 220
B.1.1 Extensions de fonctions 221
B.1.2 Extensions de données 227
B.2 Résumé des déclarations en langage C de l'ensemble d'extensions 229
B.2.1 Ensemble d'extensions X.400 230
B.2.2 Extensions supplémentaires pour un mappage CMC simple vers X400 231
B.2.3 Autres ensembles d'extension 234
B.2.4 Informations propres aux plates-formes, y compris les associations d'exécution 234
B.2.5 Utilisation des services de l'ossature X.400 par l'interface CMC simple 236

Annexe C – Exemples de programmation 255
C.1 Exemples de programmation 255
C.1.1 Demande de configuration, ouverture et fermeture de session 255
C.1.2 Fonctions d'émission et d'émission de documents 255
C.1.3 Lister, lire et supprimer le premier message non lu 257
C.1.4 Recherche un destinataire donné et obtenir ses détails 258
C.1.5 Utilisation d'extensions 258
C.1.6 Mise en œuvre de la liaison CMC 259
C.2 Exemple de mise en œuvre de liaison de mise en œuvre CMC 261
C.3 Composition d'un message 262
C.4 Recherche de nouveaux messages 265
C.5 Remplissage d'un message 267
C.6 Suppression d'un message 271
C.7 Extraction d'un message 273

INTERFACE DE PROGRAMME D'APPLICATION D'APPEL COMMUN DE MESSAGERIE

(Genève, 1997)

1 Introduction

Le présent paragraphe présente une introduction de l'interface de programme d'application (API, *application program interface*) de l'appel commun de messagerie (CMC, *common messaging call*) et de ses spécifications. Il indique le but de cette interface, en donne un aperçu général, en détaille les abréviations, fournit des références de documents, explique le niveau d'abstraction de l'interface, définit les conventions de dénomination en langage C et spécifie les prescriptions de conformité.

La présente Recommandation est une extension de la première version de l'interface API d'appel CMC publiée en juin 1993 par l'association API X.400. La présente Recommandation étend la prise en charge d'applications connaissant la messagerie, décrite dans le document original à la prise en charge d'applications tributaires de la messagerie.

1.1 But

L'objectif de la présente Recommandation est de spécifier une interface de programme d'application à haut niveau pouvant être prise en charge par la majorité des services de messagerie installés à l'heure actuelle. L'interface API est conçue afin de permettre à des programmeurs d'application d'intégrer facilement une messagerie dans leurs applications, ainsi que les communications qui en résultent, ce qui aboutit à la création d'un important ensemble *d'applications connaissant la messagerie*.

La présente Recommandation est destinée à des créateurs de services de messagerie qui pourraient souhaiter prendre en charge une telle interface de programme d'application. La présente Recommandation peut également servir de guide aux créateurs d'applications pour la compréhension des caractéristiques indépendantes de la mise en œuvre de l'interface API de l'appel commun de messagerie. En ce qui concerne la prise en charge de la messagerie, les concepteurs d'application doivent se conformer aux manuels fournis pour le système qu'ils utilisent.

1.2 Aperçu général

L'interface de programme d'application d'appel commun de messagerie fournit un ensemble de fonctions de haut niveau destinées à des applications utilisant la messagerie afin de leur permettre d'émettre et de recevoir des messages électroniques.

Il existe, dans l'ensemble des applications utilisant la messagerie, des applications connaissant la messagerie et des applications tributaires de la messagerie.

Les applications connaissant la messagerie peuvent fonctionner d'une manière tout à fait satisfaisante comme applications isolées, mais elles peuvent également se connecter à un service de messagerie en vue de fournir des fonctions plus étendues. Un exemple possible est une application de traitement de texte ou de tableur qui est en mesure d'envoyer le document ou un fichier en utilisant une option ENVOI DE FICHIER figurant dans son menu.

Les applications tributaires de la messagerie dépendent d'une manière inhérente de l'existence du service de messagerie pour fournir leurs fonctions. Des exemples possibles sont l'échange de données informatisé (EDI, *electronic data interchange*), les applications de distribution d'informations, les applications de conférence ou en collaboration et, éventuellement, certaines bases de données réparties.

L'interface décrite est conçue de manière à être indépendante du protocole de messagerie effectivement utilisé entre l'expéditeur et le destinataire. L'interface prendra en charge la création et la réception de formats de message normalisés correspondant aussi bien à des normes telles que X.400 ou SMTP/MIME (RFC 822/RFC 1521) qu'à des formats de messages propres à un fournisseur. Cet objectif est atteint au moyen de la définition générique de capacités communes à la plupart des protocoles de messagerie, associée à un mécanisme de définition d'extensions pouvant être utilisées pour invoquer des services spécifiques d'un protocole.

L'interface est également conçue de manière à être indépendante du système d'exploitation et du matériel sous-jacents utilisés par le service de messagerie.

Un autre facteur important dans la conception de cette interface API est de permettre aux applications de mettre en œuvre des actions simples pouvant être réalisées au moyen d'un nombre minimal d'appels de fonction, tout en fournissant également la possibilité d'effectuer des actions plus complexes. Pour atteindre ces objectifs souvent contradictoires, l'interface API d'appel CMC se présente en deux versions: une interface CMC simple et une interface CMC complète. L'interface CMC simple fournit un nombre minimal d'appels de fonction nécessaires pour l'émission et la réception d'un message par des applications en rapport avec la messagerie. L'interface CMC complète fournit un ensemble plus complet de fonctions disponibles pour des applications utilisant la messagerie d'une manière plus robuste.

L'interface API d'appel CMC est conçue de manière à compléter des interfaces API existantes définies par les groupes XAPIA-X/OPEN, telles que les interfaces API XMHS et XMS.

L'interface CMC est conçue en vue de fournir une interface commune pour la quasi-totalité des services de messagerie électronique. Toute mise en œuvre de l'appel CMC devra mapper les vues et capacités définies pour l'appel CMC avec celles du système de messagerie sous-jacent.

Il est essentiel qu'un mappage commun soit défini par le segment de l'industrie représentatif du protocole de la messagerie ou de l'interface considérée, afin de maximiser l'interopérabilité entre des applications CMC utilisant des services sous-jacents de messagerie similaires.

Dans une telle perspective:

- la présente Recommandation définit le mappage commun entre l'appel CMC simple et le protocole X.400 de mémoire de message;
- les organismes de normalisation, fournisseurs et groupes de fournisseurs, représentant un protocole de messagerie ou une interface spécifique, sont encouragés à définir un mappage commun entre l'appel CMC et le protocole de messagerie ou l'interface considérée.

Pour maximiser l'interopérabilité entre des applications CMC utilisant des services sous-jacents de messagerie différents, il est essentiel que les définitions d'un mappage commun soient conçues en gardant une telle interopérabilité à l'esprit.

Les directives suivantes sont proposées à cet effet:

- mapper les chaînes de caractères des messages vers des jeux de caractères internationaux, chaque fois que cela est adéquat ou possible;
- mapper les types d'attachement des messages vers des types d'attachement communément acceptés, chaque fois que cela est adéquat ou possible.

Cette liste n'est pas exhaustive et des directives supplémentaires peuvent être proposées à l'avenir, une fois que des mises en œuvre auront été installées.

1.3 Terminologie

1.3.1 Définitions

La présente Recommandation définit les termes suivants:

1.3.1.1 appel CMC complet: interface API fournissant les fonctions de prise en charge d'applications tributaires de la messagerie.

1.3.1.2 appel CMC simple: interface API fournissant les fonctions de prise en charge d'applications connaissant la messagerie.

1.3.1.3 T.611: interface API du CCITT pour une utilisation avec les services de télécopie, de télex et de télétex.

1.3.2 Abréviations

La présente Recommandation utilise les abréviations suivantes:

API interface de programme d'application (*application program interface*)

CMC appel commun de messagerie (*common messaging call*)

XAPIA association pour l'interface de programme d'application X.400 (*X.400 application program interface association*)

API XMHS interface de programme d'application X/OPEN vers la messagerie électronique (X.400) [*X/OPEN application program interface to electronic mail (X.400)*]

API XMS interface de programme d'application X/OPEN vers le stockage de message (*X/OPEN message store application program interface*)

API XOM interface API X/OPEN de manipulation de données abstraites OSI (*X/OPEN OSI-abstract-data manipulation API*)

UI interface utilisateur (*user interface*)

1.4 Références normatives

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui de ce fait en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée.

1.4.1 Recommandations | Normes internationales identiques

– Recommandation UIT-T X.402 (1995) | ISO/CEI 10021-2:1996, *Technologies de l'information – Systèmes de messagerie: architecture globale*.

– Recommandation UIT-T X.411 (1995) | ISO/CEI 10021-4:1997, *Technologies de l'information – Systèmes de messagerie: système de transfert de messages: définition et procédures de services abstraits*.

– Recommandation UIT-T X.413 (1995) | ISO/CEI 10021-5:1996, *Technologies de l'information – Systèmes de messagerie: mémoire de messages: définition du service abstrait*.

– Recommandation UIT-T X.419 (1995) | ISO/CEI 10021-6:1996, *Technologies de l'information – Systèmes de messagerie: spécifications de protocole*.

– Recommandation UIT-T X.420 (1996) | ISO/CEI 10021-7:1997, *Technologies de l'information – Systèmes de messagerie: système de messagerie de personne à personne*.

1.4.2 Paires de Recommandations | Normes internationales équivalentes par leur contenu technique

– Recommandation X.208 du CCITT (1988), *Spécification de la syntaxe abstraite numéro un (ASN.1)*.

ISO/CEI 8824:1990, *Technologies de l'information – Interconnexion de systèmes ouverts – Spécification de notation de syntaxe abstraite numéro un (ASN.1)*.

– Recommandation X.209 du CCITT (1988), *Spécification des règles de codage de base pour la notation de syntaxe abstraite numéro un (ASN.1)*.

ISO/CEI 8825:1990, *Technologies de l'information – Interconnexion de systèmes ouverts – Spécification de règles de base pour coder la notation de syntaxe abstraite numéro UNE (ASN.1)*.

– Recommandation UIT-T X.400/F.400 (1996), *Aperçu général du service et du service de messagerie*.

ISO/CEI 10021-1:1997, *Technologies de l'information – Communication de texte – Systèmes d'échange de texte en mode message (MOTIS) – Partie 1: Présentation générale du système et des services*.

1.4.3 Références supplémentaires

– ISO/CEI 8601:1988, *Eléments de données et formats d'échange – Echange d'information – Représentation de la date et de l'heure*.
– ISO/CEI 9070:1991, *Technologies de l'information – Facilités de support SGML – Procédures d'enregistrement pour identificateurs de propriétaire de texte public*.
– ISO/CEI 10021-3:1990, *Technologies de l'information – Communication de texte – Systèmes d'échange de texte en mode message (MOTIS) – Partie 3: Conventions relatives à la définition de texte abstrait*.
– IMAP – "Internet Message Access Protocol", Version 4, RFC 1730, décembre 1994 (*protocole d'accès de message Internet*).
– MIME – "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, septembre 1993 (*mécanismes de spécification et de description du format de corps de message Internet*).
– RFC 822 – "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, août 1982 (*norme pour le format de messages de texte Internet ARPA*).
– SMTP – "Simple Mail Transfer Protocol", RFC 821, août 1982 (*protocole simple de transfert de courrier*).
– XMHS API – API to Electronic Mail (X.400), CAE Specification, X/Open Company Limited and X.400 API Association, 1991 (*interface API avec la messagerie électronique*).

– XMS API – Message Store API, Preliminary Specification, X/Open Company Limited and X.400 API Association, 1991 (*interface API de mémoire de message, spécification préliminaire*).

– XOM API – OSI-Abstract-Data Manipulation API, CAE Specification, X/Open Company Limited and X.400 API Association, 1991 (*interface API de manipulation de données abstraites OSI*).

– ANSI C – American National Standard for Information Systems – Programming Language C, X3.159-1989 (*le langage de programmation C*).

1.5 Niveaux

La présente Recommandation décrit l'interface API d'appel CMC à deux niveaux d'abstraction. Elle décrit une interface "générique" indépendante de tout langage de programmation et une interface en langage C basée sur la norme ANSI du langage de programmation C. L'interface "générique" est donnée afin de fournir un guide pour l'élaboration de spécifications particulières pour d'autres langages de programmation, tels que Pascal.

Les spécifications des interfaces génériques et en langage C sont combinées afin de faciliter la lecture. Le paragraphe 4 donne les structures de données CMC en les décrivant d'une manière générale, mais contient également une déclaration en langage C. Les fonctions CMC sont spécifiées d'une manière générique dans le paragraphe 6, qui contient cependant également un résumé en langage C. Dans un but de clarté, les constantes et codes erreur sont décrits dans l'ensemble de la présente Recommandation en utilisant la syntaxe de langage C définie ci-dessous. L'Annexe A donne un résumé des déclarations et constantes en langage C utilisées dans l'ensemble de la Recommandation.

1.6 Conventions de nom en langage C

Le Tableau 1 ci-dessous indique comment l'identificateur d'un élément de l'interface en langage C est déterminé à partir du nom de l'élément de l'interface générique, compte tenu du type de cet élément. Le nom générique est préfixé par la chaîne de caractères figurant dans la deuxième colonne, les caractères alphabétiques étant convertis dans la casse indiquée par la troisième colonne.

Tableau 1/X.446 – Détermination des conventions de nom en langage C

Type d'élément	Préfixe	Casse
type de données	CMC_	minuscule
valeur de données	CMC_	majuscule
fonction	cmc_	minuscule
argument de fonction	aucun	minuscule
résultat de fonction	aucun	minuscule
constante	CMC_	majuscule
erreur	CMC_E_	majuscule
macro	CMC_	majuscule
classe d'objets	CMC_OC_	majuscule
type contenu	CMC_CT_	majuscule
propriété	CMC_PT_	majuscule
étiquette de structure	CMC_TAG_	majuscule
réservé pour des ensembles d'extensions	CMC_XS_	les deux
réservé pour des extensions	CMC_X_	les deux
réservé pour une utilisation par les réalisateurs	CMCP	les deux

Les éléments préfixés par "CMCP" (majuscules et minuscules) sont réservés pour une utilisation interne propre aux réalisateurs du service CMC. Ils ne sont pas destinés à une utilisation directe dans des programmes écrits en utilisant l'interface CMC.

Les préfixes "CMC_XS_" et "CMC_X_" (majuscules et minuscules) sont réservés pour des extensions de l'interface faites par les fournisseurs.

Pour les valeurs de données constantes, une chaîne de caractères est en général ajoutée au préfixe "CMC_" afin d'indiquer à quelle structure de données ou à quelle fonction s'applique la constante.

2 Architecture CMC

Le présent paragraphe décrit l'architecture fonctionnelle sous-jacente à l'interface API d'appel CMC. Il définit le modèle fonctionnel CMC, le modèle de configuration CMC, le modèle informatique de l'interface API d'appel CMC et le modèle objet CMC. Le modèle fonctionnel définit les fonctions de messagerie normalisées par la présente Recommandation. Le modèle de configuration définit de quelle manière plusieurs mises en œuvre de l'appel CMC peuvent coexister sur une plate-forme donnée. Le modèle informatique définit des caractéristiques communes de l'interface de programmation CMC. Le modèle objet décrit les caractéristiques des objets définis dans la présente Recommandation.

2.1 Modèle fonctionnel

L'interface CMC est définie entre une application utilisant la messagerie et un service de messagerie. Le service de messagerie peut, à son tour, prendre en charge de multiples services de protocole de messagerie, utilisant chacun des formats de messagerie et des protocoles différents qui correspondent, par exemple, aux normes X.400, RFC 822 et RFC 1521. Toutes les fonctions de cette interface sont conçues pour être indépendantes des services de protocole de messagerie. L'interface API permet toutefois l'invocation de fonctions propres à un protocole au moyen de la définition de propriétés propres à la mise en œuvre et de l'utilisation d'extensions (voir 2.2.5, Extensions).

La Figure 1 ci-dessous décrit l'interface CMC.

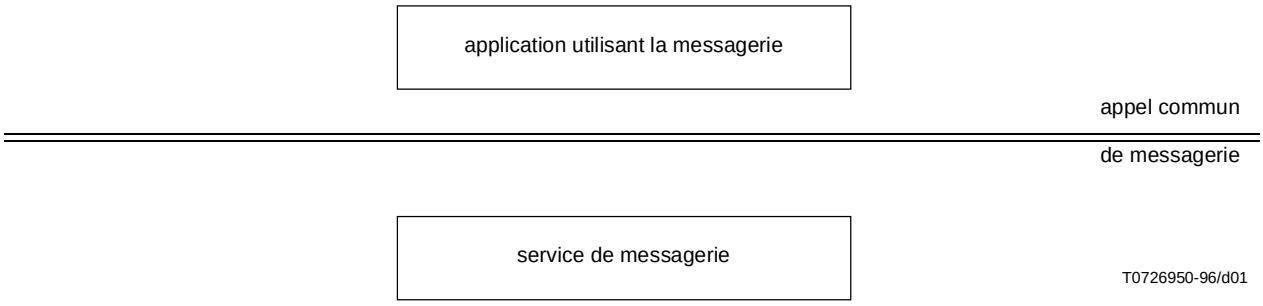


Figure 1/X.446 – Positionnement de l'interface API d'appel commun de messagerie

La Figure 2 présente les composants fonctionnels situés au-dessous de l'interface API d'appel CMC.

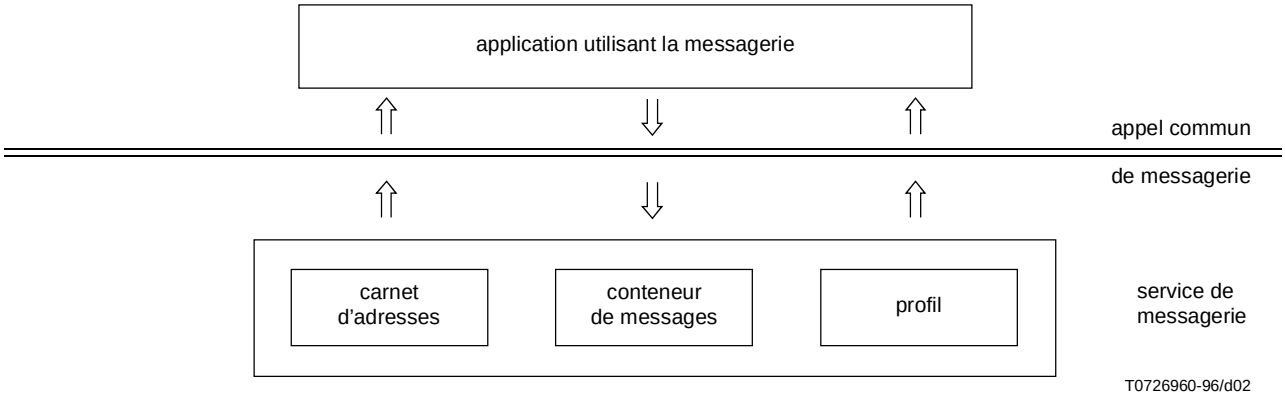


Figure 2/X.446 – Modèle d'interface API d'appel commun de messagerie

L'interface API d'appel CMC se constitue de trois composants fonctionnels: le carnet d'adresses, le conteneur de messages et le profil. Le carnet d'adresses contient les informations de destinataires et de listes de distribution utilisées pour l'adressage. Le conteneur de messages contient les messages. Les conteneurs de messages usuels sont les boîtes aux lettres en entrée, en sortie et en émission. Le profil contient les informations relatives à la mise en œuvre de l'appel CMC et à l'information utilisateur.

Le modèle informatique d'appel CMC spécifie les interactions entre une application utilisant la messagerie et ces composants fonctionnels.

2.2 Modèle informatique

Le modèle informatique d'appel CMC décrit les interfaces définies par la spécification et les caractéristiques communes de ces interfaces. Ces caractéristiques communes incluent les concepts de session CMC, de prise en charge du jeu de caractères, un mécanisme d'extension et la notification d'événements.

2.2.1 Interfaces

L'interface API d'appel CMC définit deux interfaces: l'interface CMC simple et l'interface CMC complète. L'interface CMC simple est destinée à fournir la fonction de messagerie de base à des applications connaissant la messagerie. L'interface CMC complète est conçue pour offrir une fonction de messagerie étendue à des applications utilisant la messagerie.

2.2.1.1 Interface CMC simple

L'interface CMC simple présente une compatibilité amont avec la mise en œuvre de la version CMC 1.0. L'interface CMC simple ajoute, par rapport à la version CMC 1.0, un nouveau type de message CMC REPORT (*compte rendu CMC*) permettant de consolider dans un message de compte rendu unique des comptes rendus de remise ou de non-remise concernant un message original, ce qui est cohérent avec plusieurs mises en œuvre de la messagerie X.400.

L'interface CMC simple prend en charge trois tâches principales: l'émission de messages, la lecture de messages et la consultation de l'information d'adressage. Les fonctions de cette interface sont destinées à fournir une prise en charge de la messagerie pour des applications utilisant la messagerie. Ces dernières sont des applications qui ne dépendent pas des services de messagerie pour réaliser leurs fonctions de base (par exemple des applications de traitement de texte, de tableur, de gestion d'image ou de documents). L'accès à des services de messagerie permet d'améliorer l'utilisation de ces applications dans l'environnement informatique d'une entreprise.

Une telle application souhaitant émettre un message doit en premier lieu établir une session avec le service de messagerie au moyen de la fonction **cmc_logon()**, ou d'une manière interactive en positionnant le fanion LOGON_UI_ALLOWED dans l'argument extensions de la fonction **cmc_send()**. Une application dépose un message auprès du service de messagerie au moyen d'une fonction **cmc_send()**. L'application utilisant la messagerie est responsable du remplissage de la structure du message CMC utilisée par la fonction **cmc_send()**. L'application peut également utiliser, pour émettre un message, une fonction **cmc_send_documents()** plus limitée. Cette fonction est prévue principalement pour un appel à partir d'un langage de macros. La fermeture d'une session est réalisée au moyen de la fonction **cmc_logoff()**.

Une application utilisant la messagerie qui souhaite obtenir un message établit une session au moyen de la fonction **cmc_logon()**. L'application peut ensuite extraire un résumé de l'information de la boîte aux lettres au moyen de la fonction **cmc_list()**. Les messages individuels peuvent être extraits au moyen de la fonction **cmc_read()**. La fonction **cmc_act_on()** permet à l'utilisateur d'agir sur un message se trouvant dans la boîte aux lettres (par exemple de le supprimer). La mémoire allouée par le système pour des structures de données est libérée en passant le pointeur obtenu en retour à la fonction **cmc_free()**. La fermeture d'une session est effectuée au moyen de la fonction **cmc_logoff()**. L'appel CMC simple normalise uniquement l'accès au conteneur de messages de la boîte aux lettres en réception. L'accès à d'autres conteneurs de messages au moyen de l'interface CMC simple peut être fourni par des extensions spécifiques au fournisseur.

L'application utilisant la messagerie qui souhaite faire une consultation de nom établit une session avec le service de messagerie au moyen de la fonction **cmc_logon()**, ou d'une manière interactive en positionnant le fanion LOGON_UI_ALLOWED dans l'argument extensions de la fonction **cmc_look_up()**. L'application utilise ensuite la fonction **cmc_look_up()** pour traduire un nom convivial en une adresse de messagerie. La mémoire allouée par le système pour des structures de données est libérée en passant à la fonction **cmc_free()** le pointeur obtenu en retour. La fermeture d'une session est effectuée au moyen de la fonction **cmc_logoff()**. La recherche dans le carnet d'adresses au moyen de la fonction **cmc_look_up()** dépend de la mise en œuvre. La recherche dans des carnets d'adresses particuliers au moyen de l'interface CMC simple peut être fournie par des extensions propres au fournisseur.

2.2.1.2 Interface CMC complète

L'interface CMC complète ajoute aux fonctions connaissant la messagerie, fournies par l'interface CMC simple, des fonctions supplémentaires utilisant la messagerie. Les principales fonctions fournies sont la composition de message, l'accès aux dossiers de messages et leur modification, l'accès au moyen de flux à des contenus d'information de taille importante et la modification de carnet d'adresses. Les applications tributaires de la messagerie ont besoin du service de messagerie pour accomplir leurs fonctions de base (par exemple des applications de système frontal de messagerie, d'agent de messagerie ou de gestion de flux de travaux). L'accès à des services de messagerie constitue un préalable à l'exploitation de telles applications.

La mise en œuvre des fonctions étendues de l'interface CMC complète est facilitée par l'utilisation d'un certain nombre de structures de données supplémentaires. Les capacités fournies par ces structures de données comprennent un modèle de données par objets, une définition de propriétés pour les objets permettant une définition extensible des objets du service de message, la dénomination de contenu facilitant la prise en charge de contenus multimédias dans les messages ainsi qu'un ensemble robuste de types de messages (par exemple pour la fixation de dates et de l'ordonnancement, pour les flux de travaux, pour l'EDI ou pour des messages actifs) et des objets conteneur imbriqués permettant la prise en charge de dossiers pour le stockage des messages et des carnets d'adresses.

2.2.2 Session

Les appels de fonction CMC se font dans le contexte d'une session aussi bien pour l'interface CMC simple que pour l'interface CMC complète. Une session est établie au moyen de la fonction **cmc_logon()** et clôturée au moyen de la fonction **cmc_logoff()**. La fonction **cmc_logon()** effectue également l'authentification de l'utilisateur vis-à-vis du système de messagerie et positionne les attributs de session. Le contexte d'une session est identifié par un identificateur opaque de session qui est renvoyé par la fonction **cmc_logon()**. Les attributs de contexte de session contiennent le jeu de caractères et le numéro de version. Il n'existe pas, à l'heure actuelle, de prise en charge du partage de sessions entre applications.

Pour des applications de passerelle, un utilisateur unique représentant la passerelle peut établir des sessions pour le compte d'usagers individuels multiples et disposer, pour ce faire, d'autorisations qui vont au-delà de celles d'un usager individuel.

2.2.3 Prise en charge de caractères étendus

L'interface CMC complète prend en charge des chaînes de caractères à octets doubles (par exemple des caractères UNICODE). Ceci est réalisé en définissant la constante CMC_WCHAR dans un environnement de développement d'application, avant l'inclusion du fichier xcmc.h (c'est-à-dire en positionnant CMC_WCHAR=1). La taille de caractère est d'un octet si la constante CMC_WCHAR n'est pas définie. La définition de la constante CMC_WCHAR force toutes les définitions de chaînes de caractères de l'interface CMC complète à utiliser des caractères à deux octets. Les chaînes de caractères à octets doubles ne sont prises en charge que par l'interface CMC complète. Le fichier xcmc.h génère des prototypes de définition de fonctions utilisant des octets doubles ou simples pour tous les appels de l'interface API, selon que la constante CMC_WCHAR est définie ou non.

Ceci assure une compatibilité en amont avec la version CMC 1.0 et permet la prise en charge de chaînes de caractères à octets doubles pour l'interface CMC simple et l'interface CMC complète. Les mises en œuvre exportent les fonctions à double octet dans une librairie DLL distincte. Les applications n'ont pas le droit de mélanger les deux représentations dans une même instance de l'application. La prise en charge des chaînes de caractères à octets doubles n'est pas obligatoire pour une conformité minimale.

2.2.4 Notification d'événement

L'interface CMC simple ne prend pas en charge la notification d'événements survenant dans le service sous-jacent, telle que la notification de nouveaux messages. L'interface CMC complète fournit quatre fonctions pour la prise en charge de cette activité. Deux modes de notification sont pris en charge: l'interrogation et le rappel automatique.

Dans le mode de notification avec interrogation, l'application enregistre un intérêt pour l'interrogation d'un événement au moyen de la fonction **cmc_register_event()**. L'application interroge ensuite la mise en œuvre, avec une temporisation optionnelle, au moyen de la fonction **cmc_check_event(),** afin de superviser l'apparition d'un événement. La fonction renvoie un résultat indiquant une réussite si l'événement s'est produit et peut renvoyer en outre des données propres à l'événement. L'application appelle la fonction **cmc_unregister_event()** lorsqu'elle n'est plus intéressée par la supervision d'un événement.

Le deuxième mode d'interaction utilise le rappel automatique vers des fonctions définies par l'application. Dans ce mode, l'application enregistre auprès de la mise en œuvre une fonction de rappel au moyen de la fonction **cmc_register_event()**. La fonction à rappeler dans l'application est alors appelée automatiquement lorsque l'événement se produit. L'application peut également souhaiter forcer un rappel au moyen de la fonction **cmc_call_callbacks()**. Cette fonction est utile dans des environnements où une mise en œuvre ne peut effectuer l'appel du programme à rappeler lorsque le code de la mise en œuvre est en cours d'exécution. L'application appelle la fonction **cmc_unregister_event()** lorsqu'elle n'est plus intéressée par la supervision d'un événement.

La présente Recommandation définit un seul événement (CMC_EVENT_NEW_MESSAGES) signalant l'arrivée d'un nouveau message dans un conteneur. Les structures de données associées à cet événement sont spécifiées au 4.6 sous le titre "Structures de données de rappel automatique". L'application indique les conteneurs à superviser pour l'arrivée de nouveaux messages lorsqu'elle s'enregistre pour le nouvel événement de message. Il est possible de superviser l'arrivée de nouveaux messages dans des conteneurs multiples. Si l'application n'enregistre pas de rappel automatique pour l'événement, elle peut effectuer une interrogation pour de nouveaux événements de messages sur l'ensemble de conteneurs spécifiés dans la fonction **cmc_check_event()**. Si l'application enregistre une fonction de rappel automatique, cette dernière est appelée lorsqu'un nouveau message arrive dans l'un des conteneurs spécifiés dans la fonction **cmc_register_event()**.

Cette architecture de notification d'événement permet d'ajouter de nouveaux événements dans de futures extensions de l'appel CMC ou au moyen d'extensions faites par les fournisseurs.

2.2.5 Extensions

Les structures de données et les fonctions définies dans la présente Recommandation pour l'interface CMC simple et l'interface CMC complète peuvent être étendues d'une manière méthodique par l'utilisation d'extensions. Les extensions sont utilisées pour ajouter des champs supplémentaires à des structures de données et des paramètres supplémentaires à des appels de fonctions. Une structure de données générique normalisée a été définie pour de telles extensions. Elle se constitue d'un code article identifiant l'extension, de données d'article contenant la longueur des données, de l'extension ou les données elles-mêmes, d'une référence d'article pointant vers l'endroit où est stockée la valeur de l'extension, ou NULL lorsqu'il n'existe pas de zone mémoire liée à l'article, et de fanions d'extension.

Les extensions constituant des paramètres supplémentaires d'un appel de fonction peuvent être utilisées en entrée ou en sortie, ce qui signifie que l'extension peut être transmise comme paramètre en entrée de l'application vers le service CMC ou transmise comme paramètre en retour du service CMC vers l'application. L'application alloue la mémoire pour la structure d'extension et toute autre structure liée à l'extension dans le cas d'une extension constituant un paramètre en entrée. Le service CMC alloue, si nécessaire, la mémoire pour le résultat de l'extension constituant un paramètre en sortie. Dans ce cas, l'application est responsable de la libération de cette mémoire au moyen de la fonction **cmc_free()**.

Les extensions jouent un double rôle dans la présente Recommandation. Elles peuvent fournir un mécanisme permettant d'accueillir des fonctions qui ne sont pas communes à l'ensemble des services de messagerie. Elles permettent également de fournir un mécanisme pour de futures extensions de la Recommandation en minimisant tout problème de compatibilité amont.

L'utilisation d'extensions dans le premier cas, tout en étant très importante, doit être faite avec précaution. La dépendance vis-à-vis de fonctionnalités propres à des services de messagerie limite la portabilité d'une application entre services de messagerie particuliers. Par ailleurs, de telles fonctionnalités peuvent ne pas survivre à la traversée de passerelles multiples dans un réseau de messageries différentes.

Les réalisateurs sont encouragés à spécifier les extensions d'une manière aussi générale que possible et de proposer ces extensions comme contributions à l'ensemble des extensions de l'interface CMC, en vue de minimiser les problèmes de portabilité. Il sera, de cette manière, possible de faire évoluer l'interface API d'appel CMC dans une direction favorable tout en continuant à maximiser la portabilité.

Prière de se référer aux annexes pour plus de détails concernant l'enregistrement des extensions et les extensions définies dans la présente Recommandation.

2.3 Modèle de configuration

Le modèle de configuration CMC permet à des mises en œuvre CMC multiples de coexister dans un même environnement en spécifiant un gestionnaire CMC qui joue le rôle de courtier entre plusieurs mises en œuvre CMC. La Figure 3 indique les relations entre le gestionnaire CMC et les mises en œuvre CMC.

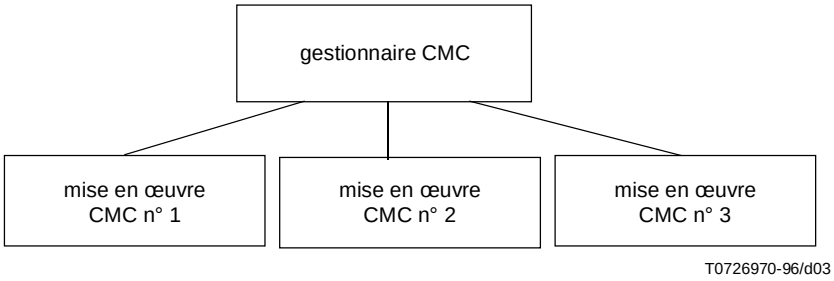


Figure 3/X.446 – Gestionnaire CMC et mises en œuvre CMC

2.3.1 Gestionnaire CMC

Les courtiers du gestionnaire CMC renvoient des tables de distribution vers les applications au moyen de la fonction **cmc_bind_implementation()**. Les tables de distribution représentent un tableau de pointeurs de fonctions CMC dont la position ordinale doit correspondre à l'ordre spécifié dans le fichier d'en-tête CMC (*CMC header file*). L'application appelle la fonction appartenant à la mise en œuvre CMC adéquate en utilisant la table de distribution correspondant à la mise en œuvre. Ceci implique que l'application doit conserver des copies des pointeurs vers chaque mise en œuvre CMC avec laquelle elle souhaite établir des liaisons. La fonction **cmc_free()** est utilisée pour libérer la table de distribution créée par un appel à la fonction **cmc_bind_implementation()**. La fonction **cmc_unbind_implementation()** est utilisée pour supprimer toutes les informations créées par le gestionnaire CMC ou la mise en œuvre CMC. Les mises en œuvre se voient attribuer des noms constituant des identificateurs globalement non ambigus (identificateurs GUID) (GUIDS, *globally unique identifier*). Les applications obtiennent les noms des mises en œuvre et les identificateurs GUID à partir des fichiers d'en-tête des fournisseurs, de la documentation des fournisseurs ou par convention. Le gestionnaire CMC est responsable du mappage de la table de distribution de la mise en œuvre vers l'espace d'adresses de l'application dans le cas de plates-formes dont les applications peuvent résider dans des espaces d'adresses distincts. Le gestionnaire CMC peut souhaiter créer et gérer des copies locales de toute table de distribution qu'il reçoit et transmet. Les applications doivent créer et gérer leurs propres copies dans le cas d'espaces d'adresses distincts mentionné ci-dessus. Le flux d'exécution est alors défini de la manière suivante:

1) l'application appelle la fonction **cmc_bind_implementation()** afin d'obtenir un pointeur vers une table de distribution pour une mise en œuvre CMC souhaitée;

2) le gestionnaire CMC reçoit l'appel et appelle à son tour la fonction **cmc_bind_implementation()** de la mise en œuvre CMC appropriée. Le gestionnaire CMC doit fournir un moyen propre à la plate-forme permettant de déterminer quelles sont les mises en œuvre CMC existantes et l'endroit où elles résident;

3) la mise en œuvre CMC reçoit l'appel pour la fonction **cmc_bind_implementation()** reçue et procède à la création et au remplissage d'une table de distribution qui est transmise en retour vers le gestionnaire CMC. Le gestionnaire CMC peut alors souhaiter créer une copie locale de la table de distribution;

4) le gestionnaire CMC termine l'exécution de la fonction **cmc_bind_implementation()** et renvoie à l'application le pointeur vers la table de distribution, à moins qu'un nouveau mappage ne soit nécessaire au préalable;

5) l'application commence à traiter des appels vers toute mise en œuvre CMC liée qui existe désormais d'une manière concurrente;

6) l'application peut à tout instant appeler la fonction **cmc_free()** afin de libérer la mémoire occupée par les tables de distribution créées par le gestionnaire CMC, la mise en œuvre CMC ou les deux. La fonction **cmc_unbind_implementation()** est appelée par l'application pour demander au gestionnaire CMC le nettoyage des données associées aux liaisons d'une mise en œuvre CMC particulière. Le gestionnaire CMC doit à son tour faire un appel vers la mise en œuvre CMC en question pour lui demander d'en faire de même;

7) lorsque toutes les mises en œuvre CMC sont déliées, l'application peut se terminer ou faire un autre appel de fonction **cmc_bind_implementation()**. Il existe un risque de perte de contrôle de la mémoire et d'un comportement imprévisible qui en résulte, si les liaisons ne sont pas détachées d'une manière symétrique par la fonction **cmc_bind_implementation()**.

Les applications utilisant la version CMC 1.0 doivent appeler la mise en œuvre CMC directement sans passer par le gestionnaire CMC. La version CMC 1.0 ne prend pas en charge de mises en œuvre multiples.

2.3.2 Directives pour les liaisons de plates-formes

La version CMC 2.0 prend en charge un gestionnaire CMC et des mises en œuvre CMC multiples sur une même plate-forme. Les directives suivantes doivent être respectées lors de la prise en charge du gestionnaire CMC et de mises en œuvre CMC multiples:

- toute liaison de plate-forme doit spécifier un mécanisme permettant aux mises en œuvre CMC de se faire enregistrer et résilier auprès d'un gestionnaire CMC;
- le gestionnaire CMC doit prendre au moins en charge les fonctions **cmc_bind_implementation()** et **cmc_unbind_implementation()** de l'interface CMC 2.0;
- le gestionnaire CMC peut également prendre en charge les fonctions suivantes:
  - interfonctionnement avec des mises en œuvre CMC dans un autre espace d'adresses;
  - interfonctionnement avec des mises en œuvre CMC sur une autre machine;
  - recherche dans une liste de mises en œuvre CMC enregistrées;
- certaines plates-formes peuvent nécessiter que les mises en œuvre CMC modifient les noms donnés aux fonctions CMC afin de prendre en charge des mises en œuvre multiples. Le gestionnaire CMC devra servir de courtier pour les mappages vers ces noms de fonction modifiés.

2.3.3 Interrogation de l'information concernant la configuration

La configuration persistante du service peut être interrogée par l'application utilisant la messagerie. L'application peut demander au service de déterminer de quelle manière sont prises en charge diverses versions de l'interface API d'appel CMC, des extensions et des paramètres d'environnement faisant partie de la configuration. L'interface API ne définit pas de fonction pour la modification de cette information de configuration. La présente Recommandation ne définit pas sous quelle forme est stockée cette information (par exemple, un format de fichier).

Deux mécanismes sont fournis pour l'interrogation des informations de configuration. L'interface CMC simple fournit une fonction **cmc_query_configuration()**. L'interface CMC complète utilise les fonctions d'énumération pour extraire l'information de configuration à partir d'un conteneur de profil.

2.4 Modèle objet

La spécification de l'interface CMC est basée sur un modèle robuste de données par objets. Une méthode d'accès très générale est définie en outre par un groupe de fonctions par objets qui permettent de gérer ces objets au sein du service de messagerie. Ces fonctions génériques fournissent une méthode simple mais très robuste pour la création et la gestion d'un objet et des propriétés d'objet définies par la spécification de l'interface CMC.

Le modèle objet de la spécification de l'interface CMC est relativement transparent pour l'utilisateur de l'interface CMC simple. Cet ensemble de fonctions utilisant la messagerie a été conçu afin de simplifier l'accès aux fonctions de service de messagerie. L'interface CMC complète fournit par ailleurs un ensemble étendu de fonctions utilisant la messagerie fournissant l'accès aux caractéristiques robustes d'un service de messagerie et de son modèle objet.

Le présent sous-paragraphe fournit un aperçu général concernant les objets CMC et les classes d'objets ainsi qu'une illustration d'un certain nombre de propriétés de chaque objet.

2.4.1 Composants du modèle

Le modèle objet contient, dans le cadre de la spécification de l'interface CMC, des objets, des classes d'objets et des propriétés. La Figure 4 donne une illustration des composants du modèle objet CMC. Un objet est une collection de propriétés. Les objets sont classés selon leur type ou classe d'objets. Une propriété est un attribut d'un objet.

2.4.1.1 Objets

Les objets sont identifiés par leur descripteur opaque propre à la session. Le descripteur opaque encapsule l'identificateur de session. La fonction **cmc_open_object_handle()** renvoie un descripteur opaque pour un nouvel objet. L'information de contenu définissant un objet de service de messagerie donné peut être ajoutée au moyen de la fonction **cmc_add_properties()**. Une propriété donnée ne peut figurer qu'une seule fois dans un objet. Cette fonction peut donc également être utilisée pour mettre à jour ou modifier l'information de contenu associée à une propriété donnée. La fonction **cmc_delete_properties()** peut être utilisée pour supprimer une ou plusieurs propriétés individuelles d'un objet. La liste des propriétés contenues dans un objet peut être fournie par la fonction **cmc_list_properties()**. L'information de contenu d'une ou de plusieurs propriétés peut être lue au moyen de la fonction **cmc_read_properties()**.

La présente Recommandation autorise l'utilisation de propriétés à valeurs multiples, utilisées en relation avec certains objets décrits dans la présente Recommandation.

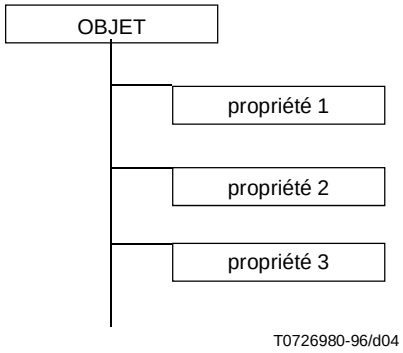


Figure 4/X.446 – Modèle objet

Les conteneurs sont une variété d'objets particulière. Ils se composent d'une collection de propriétés, mais également d'autres objets.

Une fois défini, un objet doit être ajouté à un conteneur et confié au stockage persistant de ce conteneur, respectivement au moyen des fonctions **cmc_copy_object()** et **cmc_commit_object()**. Un objet peut être supprimé dans un objet conteneur au moyen de la fonction **cmc_delete_object()**.

L'énumération des objets d'un conteneur est facilitée par le curseur de conteneur. Un curseur est une fonction propre à la mise en œuvre utilisée pour trier et filtrer les éléments d'un objet conteneur. Le curseur peut également être utilisé afin de faciliter l'affichage d'un "ascenseur" sur une barre de défilement, représentant une position relative au sein d'un conteneur. La fonction **cmc_open_cursor()** est utilisée pour définir un curseur. Le contexte du curseur est géré au moyen de la référence à son descripteur opaque. Les descripteurs opaques de curseur sont attribués par le service de messagerie, tout comme le sont les identificateurs d'objet et de session. Ils doivent être supprimés en utilisant la fonction **cmc_free()** lorsqu'ils ne sont plus utilisés. La position relative du curseur peut être lue au moyen de la fonction **cmc_read_cursor()** et mise à jour au moyen de la fonction **cmc_update_cursor_position()**. Le curseur peut également être mis à jour au moyen de la fonction **cmc_update_cursor_position_with_seed()** pour venir occuper une position relative par rapport à un objet dans un conteneur. La liste des objets d'un conteneur correspondant à une condition de filtrage sur un curseur peut être fournie par la fonction **cmc_list_number_matched()**.

La liste des objets du conteneur associé à un curseur donné peut être fournie par la fonction **cmc_list_objects()**. Les objets sont référencés par les descripteurs opaques renvoyés par cette fonction. La fonction **cmc_copy_object_handle()** peut être utilisée pour faire une copie de la référence à l'un de ces objets.

Il est possible qu'un conteneur ne contienne aucun objet, par exemple dans le cas d'un conteneur de messages vide ou d'un carnet d'adresses vide.

La prise en charge de l'imbrication de conteneurs n'est pas obligatoire pour l'interface CMC complète. La prise en charge de l'imbrication de conteneurs de messages ou de carnets d'adresses n'est pas obligatoire. Une mise en œuvre doit renvoyer dans un tel cas le code erreur CMC_E_UNSUPPORTED_ACTION. La mise en œuvre ne peut pas garantir qu'un message imbriqué dans un autre message sera retransmis. Les mises en œuvre doivent accepter des messages imbriqués fournis par l'application. Il est possible que l'imbrication ne soit pas préservée, en tant que telle, une fois que le descripteur opaque de l'objet imbriqué aura été libéré. Les objets message créés par la mise en œuvre peuvent ne pas utiliser l'imbrication. Les mises en œuvre ont la latitude de générer ou non des objets imbriqués.

2.4.1.2 Classes d'objets

Les classes d'objets sont les types d'objets définis par la présente Recommandation. Les classes d'objets de l'interface CMC contiennent des propriétés et éventuellement d'autres objets. Le paragraphe 3 décrit les classes d'objets et le paragraphe 5 les propriétés de chaque classe d'objets. Les objets pouvant être contenus dans une autre classe sont indiqués par une hiérarchie de confinement. La Figure 5 présente la hiérarchie de confinement. Cette figure n'indique toutefois pas toutes les possibilités de récursion sur les objets CMC.

Il n'existe pas dans la présente Recommandation de hiérarchie de classes explicitement définie. Toutefois, certaines classes d'objets ont les mêmes propriétés.

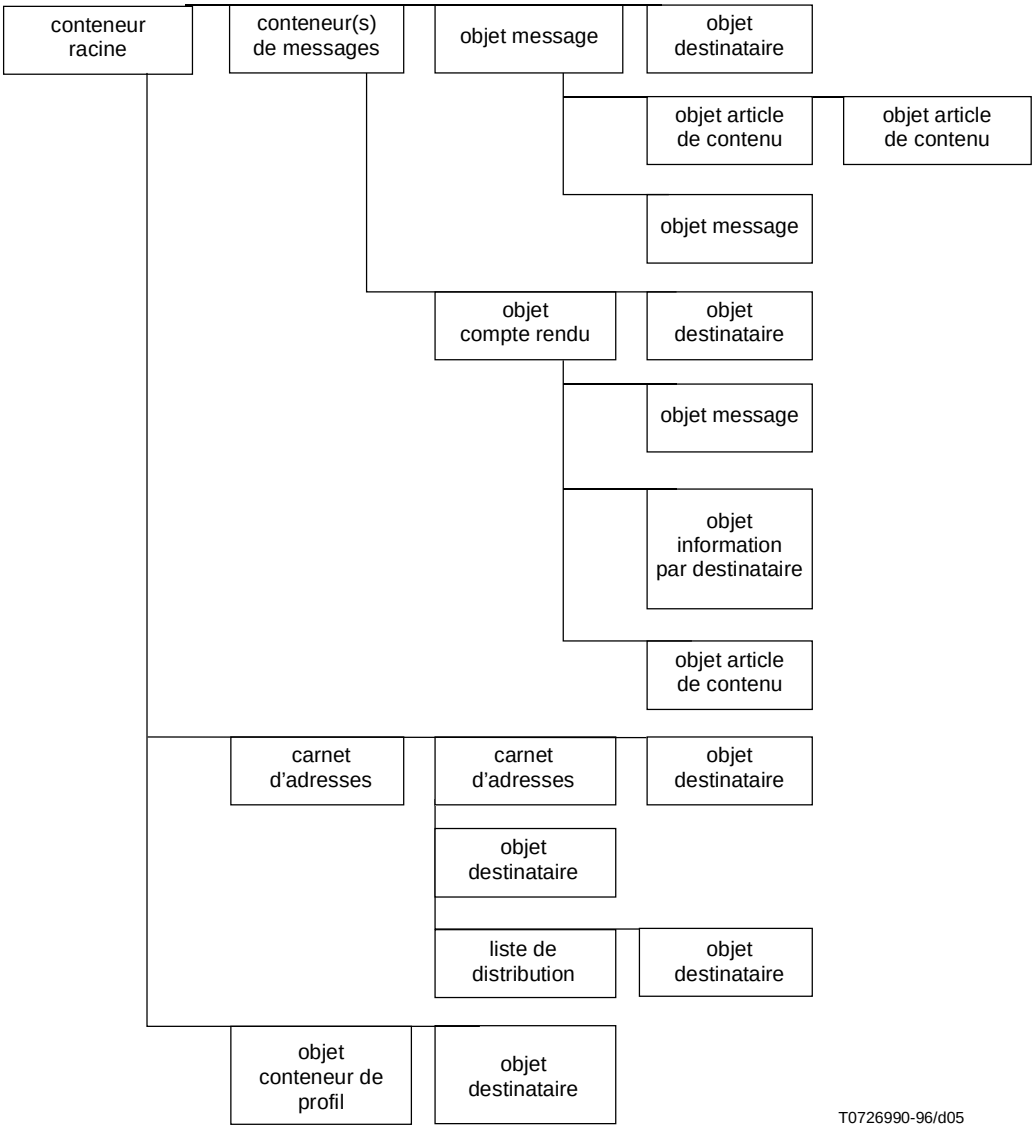


Figure 5/X.446 – Hiérarchie de confinement de l'interface CMC 2.0

2.4.1.3 Propriétés d'objets

Les propriétés sont des attributs d'un objet donné. Les propriétés définissent l'objet. Elles sont représentées par un nom non ambigu (ou, en variante, par un identificateur propre à la mise en œuvre), par un type de valeur et par des données (ou par l'information de contenu de la valeur). Une propriété est identifiée d'une manière non ambiguë par un nombre entier et une chaîne de caractères de nom basée sur l'identificateur public formel défini par l'ISO 9070.

Certaines mises en œuvre peuvent autoriser l'utilisateur à définir des propriétés. Cette possibilité permet une personnalisation du service sous-jacent. Les propriétés définies par l'utilisateur sont distinguées par leur nom. Un identificateur de propriété non ambigu est généré pour une propriété définie par l'utilisateur au moyen d'un mécanisme propre à la plate-forme. La fonction **cmc_identifier_to_name()** fournit le mappage entre un identificateur de propriété et le nom de propriété associé. La fonction **cmc_name_to_identifier()** fournit le mappage entre un nom de propriété et l'identificateur de propriété correspondant. Les identificateurs de propriété et les noms de propriété sont fournis par le service afin de permettre d'accéder aux diverses propriétés au moyen de la méthode la plus adéquate. L'espace de noms du numéro identifiant les propriétés est divisé en numéros définis par l'association XAPIA, par la mise en œuvre et par l'utilisateur. Il existe un risque de duplication entre mises en œuvre ou versions pour les numéros définis par l'utilisateur.

Certains objets contiennent des volumes d'information importants. Un message multimédia peut, par exemple, avoir un contenu audio ou vidéo de plusieurs méga-octets. Les fonctions de flux ont été ajoutées à l'interface CMC complète afin de faciliter la lecture et l'écriture de contenus d'information importants. L'accès au contenu d'information se fait d'une manière comparable à un accès fichier en langage C. Une propriété est ouverte pour des opérations de flux de lecture ou d'écriture au moyen de la fonction **cmc_open_stream()**. Cette fonction renvoie un descripteur opaque de flux qui gère le contexte de ce flux durant la session. Le descripteur opaque est alloué par le service et doit être libéré en utilisant la fonction **cmc_free()** lorsqu'il n'est plus utilisé. Les fonctions **cmc_read_stream()** et **cmc_write_stream()** sont utilisées respectivement pour lire et écrire des informations de contenu de flux. La fonction **cmc_seek_stream()** est utilisée pour accéder à une position d'octet donnée au sein du flux. Le flux est fermé comme résultat secondaire d'un appel à la fonction **cmc_free()**.

Différentes mises en œuvre peuvent conduire à des coûts de traitement différents pour la lecture de différentes classes de propriétés. Les propriétés correspondant à des volumes d'information de contenu importants peuvent avoir des coûts élevés. Les propriétés correspondant à des volumes d'information de contenu faibles ou à une information déjà disponible pour le service peuvent avoir un coût de traitement pour la lecture faible ou nul. Le coût relatif de traitement en lecture d'une propriété, propre à la mise en œuvre, peut être déterminé par la fonction **cmc_read_property_costs()**.

3 Classes d'objets CMC

3.1 Classes d'objets de l'API CMC

Les sous-paragraphes qui suivent définissent les classes d'objets de l'interface API d'appel CMC, donnent les noms des classes, détaillent les prescriptions concernant les classes d'objets et indiquent de quelle manière sont créés, ajoutés et modifiés les objets de chaque classe.

Le Tableau 2 présente un résumé des classes d'objets. La première colonne contient le nom de la classe d'objets. La deuxième colonne indique si la classe d'objets est obligatoire ou facultative. La troisième colonne indique si l'objet est accessible uniquement en lecture. Une mention "non" dans cette colonne signifie que les objets appartenant à cette classe d'objets peuvent être ajoutés, supprimés ou confiés respectivement par les fonctions **cmc_copy_object()**, **cmc_delete_object()**, ou **cmc_commit_object()**, sauf indication contraire faite par un caractère "*". La quatrième colonne indique le nombre d'instances autorisées pour un objet. La dernière colonne donne l'identité du créateur de la classe d'objets: mise en œuvre (I), appelant (A) ou l'un ou l'autre (E).

Le paragraphe 5 de la présente Recommandation décrit les propriétés de chaque classe d'objets.

Tableau 2/X.446 – Résumé des classes d'objets

Classe d'objets	M/O	En lecture seulement	Nombre d'instances	Créateur
carnet d'adresses	O	non	quelconque	E
article de contenu	O	non	quelconque	E
liste de distribution	O	non	quelconque	E
message	M	non	quelconque	E
conteneur de messages – projets	O	non	quelconque	C
conteneur de messages – mis en fichier	O	non	quelconque	C
conteneur de messages – boîte aux lettres en entrée	O	non*	zéro ou plus	I
conteneur de messages – boîte aux lettres en sortie	O	non*	une	I
conteneur de messages – émis, supprimés	O	non	une	I
information par destinataire	O	non	une ou plus	E
conteneur de profil	M	oui	une	I
destinataire	M	non	quelconque	E
compte rendu	O	non	quelconque	E
conteneur racine	O	non*	une	I
M obligatoire (*mandatory*) O optionnel (*optional*)				

3.1.1 **Carnet d'adresses** (*Address book*)

NOM

Carnet d'adresses

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

La classe conteneur de carnet d'adresses est constituée des conteneurs destinés à renfermer des objets destinataire et liste de distribution. Les conteneurs de carnets d'adresses peuvent être imbriqués, mais les mises en œuvre n'ont pas l'obligation de prendre cette imbrication en charge. Une mise en œuvre CMC n'a pas l'obligation de prendre en charge les conteneurs de carnets d'adresses. Les sous-types de carnet d'adresses sont "global" et "personnel". Les carnets d'adresses renferment des objets destinataire, des objets liste de distribution et, d'une manière optionnelle, d'autres carnets d'adresses.

3.1.2 **Article de contenu** (*Content item*)

NOM

Article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette classe d'objets identifie des objets associés au contenu d'un message. Elle est utilisée pour représenter des attachements et des notes, bien que l'interface de programmation ne fasse pas de distinction entre les deux. Les types d'objets article de contenu sont des identificateurs globalement non ambigus. Les objets article de contenu peuvent être imbriqués, la prise en charge de l'imbrication par une mise en œuvre est toutefois optionnelle.

Les mises en œuvre peuvent imposer une limite au nombre d'articles de contenu par message ou à la taille d'un article de contenu. Un appel pour l'ajout d'un article de contenu peut générer le code erreur CMC_E_TOO_MANY_CONTENT_ITEMS si le nombre d'articles de contenu autorisé est dépassé. Le code erreur CMC_E_TEXT_TOO_LARGE peut être généré si la taille de l'article de contenu dépasse la taille maximale fixée par la mise en œuvre.

3.1.3 **Liste de distribution** (*Distribution list*)

NOM

Liste de distribution

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

La classe d'objets liste de distribution identifie des objets représentant des groupes d'objets destinataire. Les listes de distribution contiennent des objets destinataire et, d'une manière optionnelle, d'autres listes de distribution. L'imbrication de listes de distribution peut ne pas être préservée une fois que le descripteur opaque de la liste de distribution a été libéré. Les mises en œuvre n'ont pas l'obligation de prendre en charge les listes de distribution ou l'imbrication de listes de distribution. Ces listes de distribution sont identifiées par des objets destinataire dont la propriété de type est "groupe".

L'utilisation de l'interface CMC pour la construction d'une liste de distribution n'implique pas que le système de messagerie prenne nécessairement en charge l'accès à des listes de distribution dont les membres sont gérés par un service de carnet d'adresses ou d'annuaire autre que celui pris en charge par la mise en œuvre CMC.

3.1.4 **Message** (*Message*)

NOM

Message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette classe d'objets identifie des objets message véhiculant des contenus d'information à travers un service de messagerie. Ces objets message peuvent être du courrier ou des accusés de réception. Les objets message peuvent être imbriqués par les applications et les mises en œuvre doivent accepter de tels messages. L'imbrication peut ne pas être préservée une fois que le descripteur opaque de l'objet imbriqué a été libéré. Les mises en œuvre n'ont pas l'obligation de prendre en charge l'imbrication de messages. Les objets message peuvent contenir des objets destinataire, des objets article de contenu et des objets message imbriqués.

3.1.5 **Conteneur de messages** (*Message Container*)

NOM

Conteneur de messages

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Les conteneurs de messages sont des collections de propriétés de conteneur de messages, d'objets message, et éventuellement d'autres conteneurs de messages. Cet objet conteneur fournit également les améliorations utilisées pour des collections spécialisées telles que les boîtes aux lettres en entrée, les boîtes aux lettres en sortie, les dossiers de suppression ou les dossiers de message définis par l'utilisateur.

La classe d'objets conteneur de messages fournit une capacité de dossier destinée à contenir des objets message, et éventuellement des objets compte rendu et d'autres objets conteneur de messages. Les conteneurs de messages peuvent être imbriqués mais les mises en œuvre n'ont pas l'obligation de prendre en charge l'imbrication d'objets conteneur de messages. Les sous-types de conteneurs de messages définis par l'interface CMC sont les suivants: projets, supprimés, envoyés, mis en fichier, boîte aux lettres en entrée et boîte aux lettres en sortie.

3.1.5.1 **Classe de conteneur de messages: projets** (*Draft*)

Le conteneur de messages en projet renferme des messages qui ont été créés mais non émis. La prise en charge du conteneur de messages en projet est optionnelle.

3.1.5.2 **Classe de conteneur de messages: supprimés, émis** (*Deleted, Sent*)

Le conteneur de messages supprimés contient les messages supprimés. Le conteneur de messages émis contient les messages émis. La prise en charge des conteneurs de messages supprimés et émis est optionnelle.

3.1.5.3 **Classe de conteneur de messages: mis en fichier** (*Filed*)

Le conteneur de messages mis en fichier contient les messages mis en fichier. La prise en charge des conteneurs de messages mis en fichier est optionnelle.

3.1.5.4 **Classe de conteneur de messages: boîte aux lettres en entrée** (*Inbox*)

La boîte aux lettres en entrée contient les messages entrants. La prise en charge d'une boîte aux lettres en entrée est optionnelle. Il peut exister plus d'une boîte aux lettres en entrée.

3.1.5.5 **Classe de conteneur de messages: boîte aux lettres en sortie** (*Outbox*)

La boîte aux lettres en sortie contient les messages à émettre. La prise en charge d'une boîte aux lettres en sortie est optionnelle. Il ne peut exister qu'une seule boîte aux lettres en sortie.

3.1.6 **Information par destinataire** (*Per Recipient Information*)

NOM

Information par destinataire

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette classe d'objets identifie des objets qui rendent compte de la remise ou de la non-remise d'un message à destinataire unique. Les objets appartenant à cette classe sont contenus dans des objets compte rendu. Un au moins de ces objets doit figurer dans l'objet compte rendu. La prise en charge est en général optionnelle, mais elle est obligatoire pour des mises en œuvre prenant en charge des objets compte rendu. Les objets information par destinataire ne peuvent pas être imbriqués.

3.1.7 **Conteneur de profil** (*Profile Container*)

NOM

Conteneur de profil

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

La classe conteneur de profil renferme une information de contexte de session et de configuration. Il n'existe qu'un seul conteneur de profil, situé immédiatement sous l'objet conteneur racine. L'objet conteneur est créé par le service de messagerie sous-jacent. Il est du type en lecture seulement et ne peut pas être modifié par l'utilisateur. Le contenu de l'objet conteneur de profil est également créé par le service de messagerie sous-jacent, en lecture seulement et ne peut pas être modifié par l'utilisateur. La prise en charge du conteneur de profil est obligatoire pour les mises en œuvre se conformant à la présente Recommandation.

Le conteneur de profil contient un objet destinataire correspondant à l'utilisateur ayant ouvert la session. Il peut également contenir d'autres objets destinataire correspondant à d'autres utilisateurs de la session si la mise en œuvre prend en charge des ouvertures de session partagées. Ceci permet de prendre en charge des capacités de bulletin d'information et de forum de discussion. L'objet conteneur de profil contient en outre des propriétés d'attribut de conteneur de profil correspondant à des attributs particuliers de contexte de session ou de configuration. Il existe des propriétés définies pour les attributs de configuration d'interface CMC simple et d'interface CMC complète. Les propriétés du conteneur de profil sont accessibles en lecture seulement.

3.1.8 **Destinataire** (*Recipient*)

NOM

Destinataire

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

La classe d'objets destinataire identifie des utilisateurs du service de messagerie. Les objets destinataire peuvent être des individus ou des groupes. Le type de destinataire peut être un individu, un groupe de destinataires (par exemple une liste de distribution) ou un type inconnu. Une mise en œuvre donnée peut fournir des propriétés propres à la mise en œuvre pour un objet destinataire.

3.1.9 **Compte rendu** (*Report*)

NOM

Compte rendu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

La classe d'objets compte rendu identifie des objets rendant compte du statut de remise d'un message. Les objets de cette classe contiennent des avis de remise et de non-remise. Certains systèmes de transfert de messages (tels que le protocole SMTP) peuvent ne pas prendre en charge la génération de comptes rendus. Les objets compte rendu peuvent contenir des objets destinataire, des objets information par destinataire, des objets article de contenu et des objets message.

3.1.10 **Conteneur racine** (*Root Container*)

NOM

Conteneur racine

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Le conteneur racine renferme les conteneurs de niveau sommital pour les objets messagerie d'un utilisateur. Il n'existe qu'un seul type de conteneur racine et un seul conteneur racine par utilisateur. Sa prise en charge par une mise en œuvre de l'interface CMC est obligatoire. Le conteneur racine contient des conteneurs de messages, un conteneur de profil et, d'une manière optionnelle, des conteneurs de carnets d'adresses.

4 Structures de données

Le présent paragraphe définit les structures de données utilisées dans l'interface API d'appel CMC et dont la liste est donnée par le Tableau 3.

Tableau 3/X.446 – Structures de données CMC

Nom du type de données	Description
attachement	structure d'attachement de message
booléen	valeur logique vraie ou fausse
tampon	pointeur vers un article de données
structures de rappel automatique	définitions de type pour des valeurs de données d'une fonction de rappel automatique
chaîne avec comptage	chaîne de caractères avec indication explicite de longueur
descripteur opaque de curseur	descripteur opaque pour un curseur de conteneur
contrainte de curseur	contrainte de filtrage de l'énumération des objets dans un conteneur
clé de tri de curseur	définit l'ordre de tri des éléments énumérés par un curseur dans un conteneur
table de distribution	structure contenant des pointeurs vers les fonctions d'une mise en œuvre CMC
énuméré	type de données contenant une valeur appartenant à une énumération
événements	type de données pour les événements d'un service de messagerie
extension	structure d'extension
fanions	conteneur pour des bits de fanion
identificateur Guid	identificateur global non ambigu
date et heure ISO	valeur de chaîne de caractères contenant une date et une heure formatées conformément à l'ISO 8601
message	structure de message

Tableau 3/X.446 – Structures de données CMC *(fin)*

Nom du type de données	Description
référence de message	structure de référence de message
résumé de message	structure de résumé de message
descripteur opaque d'objet	descripteur opaque d'un objet CMC
identificateur d'objet	structure d'identificateur d'objet
données opaques	chaîne d'octets avec comptage contenant des données propres à l'application
propriété	partie d'information de contenu d'un objet
identificateur	identificateur non ambigu propre à l'application
nom	nom non ambigu
destinataire	structure émetteur/destinataire
compte rendu	message de statut concernant les notifications de remise, de non-remise, de réception, etc.
code retour	valeur envoyée en retour indiquant la réussite d'une fonction ou la raison de son échec
identificateur de session	descripteur opaque pour une session
descripteur opaque de flux	descripteur opaque d'un flux de propriété
chaîne	pointeur de chaîne de caractères
temps	structure de date et d'heure
identificateur d'interface utilisateur	descripteur opaque d'interface utilisateur

4.1 Types de données de base

Certains types de données sont définis au moyen des "types de données intermédiaires" suivants dont les définitions précises en langage C sont définies par le système:

float32 représentation à 32 bits d'un nombre en virgule flottante.

float64 représentation à 64 bits d'un nombre en virgule flottante.

sint16 représentation à 16 bits d'un entier avec signe.

sint32 représentation à 32 bits d'un entier avec signe.

uint8 représentation à 8 bits d'un entier sans signe.

uint16 représentation à 16 bits d'un entier sans signe.

uint32 représentation à 32 bits d'un entier sans signe.

DÉCLARATION EN LANGAGE 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 Types de données tableau

La présente Recommandation prend en charge des propriétés à valeurs multiples au moyen de tableaux de types de données de base et d'autres types de données. Les types de données tableau sont les suivants:

array_boolean	tableau de données booléennes
array_buffer	tableau de pointeurs vers des emplacements de stockage en mémoire
array_counted_string	tableau de chaînes de caractères avec indication explicite de longueur
array_enum	tableau de types de données énumérés

array_extension	tableau de types de données d'extension
array_float32	tableau de nombres en virgule flottante à 32 bits
array_float64	tableau de nombres en virgule flottante à 64 bits
array_guid	tableau d'identificateurs globalement non ambigus
array_iso_date_time	tableau de structures de date et d'heure ISO
array_object_handle	tableau de descripteurs opaques d'objets
array_opaque_data	tableau de chaînes de caractères avec comptage de données propres à l'application
array_return_code	tableau de codes retour
array_sint16	tableau d'entiers à 16 bits avec signe
array_sint32	tableau d'entiers à 32 bits avec signe
array_string	tableau de chaînes de caractères
array_time	tableau de structures de temps
array_uint16	tableau d'entiers à 16 bits sans signe
array_uint32	tableau d'entiers à 32 bits sans signe

DÉCLARATION EN LANGAGE 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;
```

DESCRIPTION

Une valeur de données d'un de ces types contient un identificateur de longueur donnant la taille du tableau.

La prise en charge de propriétés à valeurs multiples par une mise en œuvre est optionnelle.

4.3 **Attachement** (*Attachment*)

NOM

Attachement – Définition de type pour une structure CMC d'attachement de message

DÉCLARATION EN LANGAGE 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;
```

DESCRIPTION

Une valeur de données de ce type constitue un attachement. Cette structure de données est définie en vue de la prise en charge de mises en œuvre de l'interface CMC 1.0 et de l'interface CMC simple. Un attachement se compose des parties suivantes:

1) **titre d'attachement** (*attach_title*): titre optionnel d'un attachement, par exemple le nom de fichier original de l'attachement;

2) **type d'attachement** (*attach_type*): identificateur d'objet spécifiant le type d'attachement. Le format de l'identificateur d'objet CMC est défini au 4.24. Une valeur NULL désigne un type d'attachement non défini;

   Deux identificateurs d'objet prédéfinis sont à la disposition des applications et des mises en œuvre CMC.

CMC_ATT_OID_BINARY	Les données contenues dans un fichier doivent être traitées comme des données binaires, ceci étant l'option par défaut.

CMC_ATT_OID_TEXT — Les données contenues dans un fichier doivent être traitées comme des chaînes de caractères de texte. On doit faire l'hypothèse que le jeu de caractères est celui utilisé en entrée pour la session et qu'un mappage sera fait, si possible, vers le jeu de caractères utilisé en sortie pour la session.

3) **nom de fichier d'attachement** (*attach_filename*): nom du fichier dans lequel se trouve le contenu de l'attachement. L'emplacement du fichier dépend de la mise en œuvre, mais doit être accessible à l'application appelante;

4) **fanions d'attachement** (*attach_flags*): bits pour des attributs booléens. Les bits inutilisés doivent être non positionnés;

   a) CMC_ATT_APP_OWNS_FILE

      positionné — indique en sortie que l'application est désormais propriétaire du fichier et qu'elle est responsable de sa suppression. Cet attribut est ignoré en entrée;

      non positionné — indique en sortie que la mise en œuvre CMC est propriétaire du fichier et que l'application ne peut y accéder qu'en écriture.

   b) CMC_ATT_LAST_ELEMENT

      positionné — identifie le dernier élément d'un tableau de ce type de structures;

      non positionné — il ne s'agit pas du dernier élément du tableau.

5) **extensions d'attachement** (*attach_extensions*): pointeur vers le premier élément d'un tableau d'extensions par attachement. Une valeur NULL indique qu'il n'existe pas d'extension.

4.4 Booléen (*Boolean*)

NOM

Boolean – Définition de type pour une valeur de données booléenne

DÉCLARATION EN LANGAGE C

```
typedef   CMC_uint16   CMC_boolean;
```

DESCRIPTION

Une valeur de données de ce type est booléenne, c'est-à-dire qu'elle peut avoir une valeur vraie ou fausse.

Dans l'interface C, "Faux" est représenté par un zéro {CMC_FALSE}, et "Vrai" est représenté par toute autre valeur entière, toutefois la constante symbolique {CMC_TRUE} représente la valeur entière un.

4.5 Tampon (*Buffer*)

NOM

Tampon – Définition de type pour un espace de stockage en mémoire d'un type non défini

DÉCLARATION EN LANGAGE C

```
typedef       void * CMC_buffer;
```

DESCRIPTION

Une valeur de données de ce type constitue un pointeur vers un emplacement de stockage en mémoire dont le type n'est pas défini. La taille d'un pointeur void * dépend de la plate-forme.

4.6 Structures de données de rappel automatique (*Callback Data Structures*)

NOM

Structures de données de rappel automatique – Définitions de type utilisées pour les valeurs de données d'une fonction de rappel automatique

DÉCLARATION EN LANGAGE 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
);
```

DESCRIPTION

Les procédures de rappel automatique permettent au service d'informer les applications au sujet de l'apparition d'un événement. Les procédures de rappel automatique sont du type **cmc_callback**.

Les programmeurs qui écrivent des procédures de rappel automatique doivent tenir compte de la méthode spécifique utilisée par la plate-forme sur laquelle s'exécute le rappel automatique et de l'impact des fonctions de rappel automatique sur les performances. Les rappels automatiques sont invoqués dans une séquence spécifique du service, soit lorsque l'activité de rappel automatique se manifeste, soit lorsque la fonction **cmc_call_callbacks()** est appelée. En pratique, l'application en cours de traitement au moment de l'invocation sera suspendue jusqu'à ce que le rappel automatique se soit effectué. Les temps de réponse de l'application CMC seront affectés si la fonction de rappel automatique ne s'exécute pas rapidement.

Les composants du prototype de fonction de rappel automatique sont les suivants:

– **session** (*session*) – Descripteur opaque représentant une session avec le service de messagerie.

– **événement** (*event*) – Masque de bits d'événements. Un bit et un seul sera positionné pour indiquer l'événement survenu et la manière d'interpréter l'argument **callback_data**. Les fanions suivants sont définis:

  CMC_EVENT_NEW_MESSAGES

  prière de se référer au type de données événement pour la définition de ce fanion.

– **données de rappel automatique** (*callback_data*) – Pointeur vers la structure de données de rappel automatique propre à l'événement.

– **données d'enregistrement** (*register_data*) – Pointeur vers la structure de données transmise lors de l'enregistrement du rappel automatique dans la fonction **cmc_register()** propre à l'événement.

– **extensions de rappel automatique** (*callback_extensions*) – Pointeur vers un tableau de structures d'extension CMC utilisé pour cette fonction de rappel automatique.

  Toute fonction de rappel automatique renvoie dans son argument de données de retour automatique un pointeur vers l'une des structures de données de rappel automatique. La structure de données de rappel automatique dépend du contexte du rappel automatique et elle est déterminée par la valeur de l'argument d'événement tel qu'il est décrit ci-dessous.

  La structure de données de rappel automatique est le mécanisme utilisé par le service de messagerie pour fournir à l'application une mise à jour de l'information propre à l'opération. Les applications peuvent faire transmettre un contexte supplémentaire à leurs fonctions de rappel automatique en utilisant la structure de données d'enregistrement. La structure de données de rappel automatique est allouée par la mise en œuvre CMC, la structure de données d'enregistrement est allouée par l'application.

  Lorsqu'un rappel automatique est résilié, il est également possible de spécifier des données non enregistrées associées à la fonction **cmc_unregister_event()**, afin de fournir un contexte pour la suppression de son enregistrement (par exemple, afin d'utiliser un ensemble plus réduit de conteneurs pour attendre de nouveaux messages). La structure de données de résiliation est allouée par l'application. Les types d'argument valides pour chaque événement sont indiqués ici.

Dans le contexte de la présente Recommandation, l'application ne peut également procéder à une interrogation d'événement au moyen de la fonction **cmc_check_event()**. Un contexte d'événement peut être défini au moyen d'une structure de données de vérification utilisée pour l'appel de la fonction **cmc_check_event()**. La structure de données de vérification est allouée par l'application. Les types d'argument valides pour chaque événement sont indiqués ici.

La présente Recommandation spécifie un seul événement CMC_EVENT_NEW_MESSAGES. Une application peut faire une interrogation concernant de nouveaux messages au moyen de la fonction **cmc_check_event()** ou enregistrer des rappels automatiques afin d'être rappelée au moment de l'arrivée de nouveaux messages. Si l'interrogation est utilisée, elle peut être limitée au moyen de la fonction **cmc_check_event()** à des conteneurs spécifiques définis dans l'argument de vérification de données avec la structure CMC_TAG_new_message_check_data. Les éléments de données de cette structure sont les suivants:

– **nombre de conteneurs** (*number_containers*) – Nombre de descripteurs opaques de l'argument **conteneurs**. Cet argument doit être égal à zéro si l'événement ne dépend pas d'un conteneur.

– **conteneurs** (*containers*) – Tableau de descripteurs opaques devant être supervisés pour l'apparition d'événements. Cet argument doit avoir la valeur NULL si l'événement ne dépend pas d'un conteneur.

La fonction **cmc_check_event()** renvoie la structure CMC_TAG_new_message_callback_data. Les éléments de données de cette structure sont les suivants:

– **disponible** (*available*) – Descripteur opaque d'un conteneur de messages (parmi ceux spécifiés par l'argument **conteneurs**) auquel correspond l'événement. La valeur est positionnée sur CMC_NULL_HANDLE si aucun événement ne s'est manifesté.

La structure CMC_TAG_new_message_register_data est passée par référence dans l'argument données d'enregistrement de la fonction **cmc_register_event()** lorsqu'un rappel automatique est enregistré. Les éléments de données de cette structure sont identiques à ceux de la structure de données CMC_TAG_new_message_check_data.

Si un rappel automatique est enregistré, la structure CMC_TAG_new_message_callback_data est passée à la fonction de rappel automatique lorsqu'un événement se manifeste. La structure de données CMC_TAG_new_message_register_data est également passée à la fonction de rappel automatique.

La structure CMC_TAG_new_message_unregister_data est passée par référence dans l'argument données d'enregistrement de la fonction cmc_unregister lorsqu'un rappel automatique est résilié. Les éléments de données de cette structure de données sont les suivants:

– **nombre de conteneurs** (*number_containers*) – Nombre de descripteurs opaques de l'argument **conteneurs**. Cet argument doit être égal à zéro si l'événement ne dépend pas d'un conteneur.

– **conteneurs** (*containers*) – Tableau de descripteurs opaques indiquant les conteneurs pour lesquels l'application n'est plus intéressée par la notification de nouveaux messages. Ce tableau doit être un sous-ensemble des descripteurs opaques spécifiés dans l'argument conteneurs de l'argument données d'enregistrement dans la fonction **cmc_register()**. Cet argument doit avoir la valeur NULL si l'événement ne dépend pas d'un conteneur.

L'ordre dans lequel les fonctions de rappel automatique sont invoquées par le service est dans tous les cas propre à la mise en œuvre.

4.7 Chaîne avec comptage (*Counted String*)

NOM

Chaîne avec comptage – Définition de type pour une structure CMC de chaîne avec comptage

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Une valeur de données de ce type représente une chaîne avec comptage, pour laquelle la longueur de la chaîne est spécifiée d'une manière explicite en tête de la chaîne de caractères. La chaîne ne se termine pas nécessairement par un caractère nul.

La prise en charge du type chaîne avec comptage est optionnelle. Elle est destinée à fournir une prise en charge pour des jeux de caractères contenant des valeurs nulles.

Prière de se référer au type chaîne de caractères CMC en ce qui concerne la détermination du jeu de caractères.

Les composants d'une chaîne avec comptage sont les suivants:

1) **longueur** (*length*) – Octet de longueur de la chaîne qui suit.

2) **chaîne de caractères** (*string*) – Caractères constituant la chaîne.

4.8 Descripteur opaque de curseur (*Cursor Handle*)

NOM

Descripteur opaque de curseur – Définition de type pour une structure de descripteur opaque de curseur CMC

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Une valeur de données de ce type constitue un descripteur opaque de curseur. Les descripteurs opaques de curseur CMC sont définis d'une manière propre à la mise en œuvre. Le descripteur gère un contexte propre à la session au moyen d'un curseur de conteneur. Le curseur facilite l'énumération des objets dans un conteneur. Il est également utilisé pour afficher un "ascenseur" sur une barre de déroulement indiquant dans une fenêtre une position relative dans une collection d'objets. Les descripteurs opaques de curseur ne peuvent pas être copiés.

4.9 Contrainte de curseur (*Cursor Restriction*)

NOM

Contrainte de curseur – Définition de type pour un type de données de contrainte de curseur CMC

DÉCLARATION EN LANGAGE 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;
```

DESCRIPTION

Une valeur de données de ce type représente une contrainte de curseur CMC. Une contrainte de curseur est la définition d'un filtre s'appliquant à l'énumération du contenu d'un objet conteneur. Une contrainte de curseur contient les composants suivants:

1) **type** (*type*): type de contrainte de curseur. Les types de contrainte suivants sont pris en charge:

   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 – Filtres contenant des contraintes subordonnées toutes vraies.

CMC_RESTRICTION_OR – Filtres contenant au moins une contrainte subordonnée vraie.

CMC_RESTRICTION_NOT – Filtres contenant des contraintes subordonnées toutes fausses.

CMC_RESTRICTION_STRING – Filtres d'exactitude de la comparaison d'une chaîne de caractères avec une valeur de propriété.

CMC_RESTRICTION_CONTENT – Filtres de comparaison logique d'une constante avec une valeur de propriété.

CMC_RESTRICTION_COMPARISON – Filtres de comparaison logique de deux valeurs de propriété.

CMC_RESTRICTION_BITTEST – Filtres de comparaison d'une valeur de propriété avec le masque binaire de test spécifié.

CMC_RESTRICTION_EXIST – Filtres de test d'existence d'une propriété dans un objet.

2) **contrainte** (*restriction*): spécifie la valeur de la contrainte de curseur.

3) **extensions de propriété** (*property_extensions*): pointeur vers le premier élément dans un tableau d'extensions de propriétés.

L'élément de structure exactitude possède les valeurs suivantes d'énumération d'exactitude de chaîne:

   CMC_EXACTNESS_PRECISE
   CMC_EXACTNESS_STARTS_WITH
   CMC_EXACTNESS_MIXED_CASE

CMC_EXACTNESS_PRECISE – La valeur de la propriété correspond exactement à la constante chaîne de caractères.
CMC_EXACTNESS_STARTS_WITH – La valeur de la propriété débute par la constante chaîne de caractères.
CMC_EXACTNESS_MIXED_CASE – La valeur de la propriété correspond indépendamment de la casse.

L'élément de structure logique peut prendre l'une des valeurs énumérées suivantes d'opérateur logique:

CMC_LOGICAL_LT
CMC_LOGICAL_LE
CMC_LOGICAL_EQ
CMC_LOGICAL_NE
CMC_LOGICAL_GT
CMC_LOGICAL_GE

CMC_LOGICAL_LT – Inférieur à
CMC_LOGICAL_LE – Inférieur ou égal à
CMC_LOGICAL_EQ – Egal à
CMC_LOGICAL_NE – Non égal à
CMC_LOGICAL_GT – Supérieur à
CMC_LOGICAL_GE – Supérieur ou égal à

L'élément de structure de comparaison peut prendre l'une des valeurs énumérées suivantes de comparaison avec un masque de bits:

CMC_COMPARISON_OR
CMC_COMPARISON_AND

CMC_COMPARISON_OR – La valeur de la propriété est égale à l'opération OU logique du masque binaire.
CMC_COMPARISON_AND – La valeur de la propriété est égale à l'opération ET logique du masque binaire.

4.10 **Clé de tri de curseur** (*Cursor sort key*)

NOM

Clé de tri de curseur – Définition de type pour un type de données de clé de tri de curseur CMC

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Une valeur de données de ce type représente une clé de tri de curseur CMC. Une clé de tri de curseur définit dans quel ordre sont triés les éléments d'un conteneur lorsqu'ils sont énumérés par un curseur. Une mise en œuvre peut avoir un tableau de clés de tri de curseur. Une clé de tri de curseur se compose des éléments suivants:

1) **propriété** (*property*): spécifie la propriété utilisée pour le tri des éléments énumérés.

2) **ordre** (*order*): spécifie l'ordre de tri des éléments énumérés. Les ordres de tri valides sont les suivants:

CMC_SORT_DEFAULT
CMC_SORT_ASCEND
CMC_SORT_DESCEND

CMC_SORT_DEFAULT – Les éléments du conteneur ne seront pas nécessairement triés mais restent dans l'ordre par défaut. Le résultat de cet ordre est propre à la mise en œuvre.

CMC_SORT_ASCEND – Les éléments du conteneur sont triés dans l'ordre ascendant. Les objets ne possédant pas la propriété indiquée par la clé de tri sont placés à la fin.

CMC_SORT_DESCEND – Les éléments du conteneur sont triés dans l'ordre descendant. Les objets ne possédant pas la propriété indiquée par la clé de tri sont placés à la fin.

4.11 **Table de distribution** (*Dispatch Table*)

NOM

Table de distribution – Définition de type pour une structure contenant des pointeurs vers les fonctions d'une mise en œuvre CMC

DÉCLARATION EN LANGAGE C

```
typedef struct {
CMC_extension                *dispatch_table_extensions;

/* ÉMISSION */
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
);

/* ÉMISSION DE 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
);

/* AGIR SUR */
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
);

/* LISTE */
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
);

/* LECTURE */
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
);
```

```
/* RECHERCHE */
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
);

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

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

/* OUVERTURE DE SESSION */
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
);

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

/* CMC COMPLET */

/* COPIE D'OBJET */
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
);

/* AJOUT DE PROPRIÉTÉS */
CMC_return_code
(*cmc_add_properties)(
   CMC_object_handle            object,
   CMC_uint32                   number_properties,
   CMC_property                 *properties,
   CMC_extension                *add_properties_extensions
);

/* CONFIER UN OBJET */
CMC_return_code
(*cmc_commit_object)(
   CMC_object_handle            source_object,
   CMC_extension                *commit_object_extensions
);
```

```
/* COPIE DE DESCRIPTEUR OPAQUE D'OBJET */
CMC_return_code
(*cmc_copy_object_handle)(
   CMC_object_handle             source_object,
   CMC_object_handle             *new_object,
   CMC_extension                 *copy_object_handle_extensions
);

/* CRÉATION D'UN OBJET MESSAGE DÉRIVÉ */
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
);

/* SUPPRESSION D'OBJETS */
CMC_return_code
(*cmc_delete_objects)(
   CMC_uint32                    number_objects,
   CMC_object_handle             *object,
   CMC_extension                 *delete_objects_extensions
);

/* SUPPRESSION DE PROPRIÉTÉS */
CMC_return_code
(*cmc_delete_properties)(
   CMC_object_handle             object,
   CMC_uint32                    number_properties,
   CMC_id                        *property_ids,
   CMC_extension                 *delete_properties_extensions
);

/* OBTENTION DU DESCRIPTEUR OPAQUE RACINE */
CMC_return_code
(*cmc_get_root_handle)(
   CMC_session_id                session,
   CMC_object_handle             *root_object_handle,
   CMC_extension                 *get_root_handle_extensions
);

/* IDENTIFICATEUR VERS NOM */
CMC_return_code
(*cmc_identifier_to_name)(
   CMC_id                        identifier,
   CMC_name                      *name,
   CMC_extension                 *identifier_to_name_extensions
);

/* LISTE DES PROPRIÉTÉS CONTENUES */
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
);

/* LISTE DU NOMBRE DE CONCORDANCES */
CMC_return_code
(*cmc_list_number_matched)(
   CMC_cursor_handle             *cursor,
   CMC_uint32                    *number_matches,
   CMC_extension                 *list_number_matched_extensions
);
```

```
/* LISTE D'OBJETS */
CMC_return_code
(*cmc_list_objects)(
    CMC_cursor_handle             *cursor,
    CMC_sint32                    *number_objects,
    CMC_object_handle             *objects,
    CMC_extension                 *list_objects_extensions
);

/* LISTE DE PROPRIÉTÉS */
CMC_return_code
(*cmc_list_properties)(
    CMC_object_handle             *object,
    CMC_uint32                    *number_properties,
    CMC_id                        *property_ids,
    CMC_extension                 *list_properties_extensions
);

/* NOM VERS IDENTIFICATEUR */
CMC_return_code
(*cmc_name_to_identifier)(
    CMC_name                      name,
    CMC_id                        *identifier,
    CMC_extension                 *name_to_identifier_extensions
);

/* OUVERTURE DE CURSEUR */
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
);

/* OUVERTURE DE DESCRIPTEUR OPAQUE D'OBJET */
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
);

/* LECTURE DE CURSEUR */
CMC_return_code
(*cmc_read_cursor)(
    CMC_cursor_handle             *cursor,
    CMC_uint32                    *position_numerator,
    CMC_uint32                    *position_denominator,
    CMC_extension                 *read_cursor_extensions
);

/* LECTURE DE PROPRIÉTÉS */
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
);

/* LECTURE DE COÛT DE PROPRIÉTÉ */
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
);
```

```
/* RESTAURATION D'UN OBJET */
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
);

/* SAUVEGARDE D'UN OBJET */
CMC_return_code
(*cmc_save_object)(
    CMC_object_handle             object,
    CMC_string                    file_specification,
    CMC_flags                     save_flags,
    CMC_extension                 *save_object_extensions
);

/* ÉMISSION D'UN OBJET MESSAGE */
CMC_return_code
(*cmc_send_message_object)(
    CMC_object_handle             message_to_send,
    CMC_extension                 *send_message_object_extensions
);

/* MISE À JOUR DE LA POSITION DU CURSEUR */
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
);

/* MISE À JOUR DE LA POSITION DU CURSEUR AVEC PLACEMENT */
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
);

/* SUPERVISION D'ÉVÉNEMENT */
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
);

/* ENREGISTREMENT D'ÉVÉNEMENT */
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
);

/* RÉSILIATION D'ÉVÉNEMENT */
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
);
```

```
/* APPEL DE RAPPELS AUTOMATIQUES */
CMC_return_code
(*cmc_call_callbacks)(
   CMC_session_id               session,
   CMC_event                    event_type,
   CMC_extension                *call_callbacks_extensions
);

/* EXPORTATION DE FLUX */
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
);

/* IMPORTER UN FICHIER DANS UN FLUX */
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
);

/* OUVERTURE DE FLUX */
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
);

/* LECTURE DE FLUX */
CMC_return_code
(*cmc_read_stream)(
   CMC_stream_handle            stream,
   CMC_uint32                   *count,
   CMC_buffer                   content_information,
   CMC_extension                *read_stream_extensions
);

/* RECHERCHE DANS UN FLUX */
CMC_return_code
(*cmc_seek_stream)(
   CMC_stream_handle            stream,
   CMC_enum                     operation,
   CMC_uint32                   *location,
   CMC_extension                *seek_stream_extensions
);

/* ÉCRITURE DE FLUX */
CMC_return_code
(*cmc_write_stream)(
   CMC_stream_handle            *stream,
   CMC_uint32                   *count,
   CMC_buffer                   *content_information,
   CMC_extension                *write_stream_extensions
);

/* OBTENIR LA DERNIÈRE ERREUR */
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;
```

```
/* LIER LA MISE EN ŒUVRE   */
CMC_return_code
cmc_bind_implementation (
    CMC_guid                        implementation_name,
    CMC_dispatch_table              **dispatch_table,
    CMC_extension                   *cmc_bind_extensions
);

/* DÉLIER LA MISE EN ŒUVRE */
CMC_return_code
cmc_unbind_implementation (
    CMC_guid                        implementation_name,
    CMC_extension                   *cmc_unbind_implementation_extensions
);
```

DESCRIPTION

Une valeur de données de ce type représente une table de distribution pour une mise en œuvre CMC. La table de distribution contient un élément pour chaque fonction d'une mise en œuvre CMC. Prière de se référer aux exemples donnés au C.2 (bind.c et cmc_bind.c) pour ce qui est de l'utilisation de la table de distribution.

4.12 **Enuméré** (*Enumerated*)

NOM

Enuméré – Définition de type pour une valeur de données énumérée

DÉCLARATION EN LANGAGE C

```
typedef  CMC_sint32  CMC_enum;
```

DESCRIPTION

Une valeur de données de ce type de données contient une valeur prise dans une liste d'énumérations.

4.13 **Evénements** (*Events*)

NOM

Evénements – Définition de type pour un événement CMC

DÉCLARATION EN LANGAGE C

```
typedef   CMC_uint32      CMC_event;
```

DESCRIPTION

Une valeur de données de ce type contient 32 bits d'événement. Les événements non documentés sont réservés. Les bits d'événement mis à zéro sont dits "non positionnés", les bits d'événement mis à un sont dits "positionnés". Les bits d'événement non spécifiés doivent être non positionnés.

positionné: de nouveaux messages sont arrivés dans un conteneur de messages.

non positionné: aucun nouveau message n'est arrivé dans un conteneur de messages.

Le seul type d'événement valide défini par la présente Recommandation est:

CMC_EVENT_NEW_MESSAGES

4.14 **Extension** (*Extension*)

NOM

Extension – Définition de type pour une structure d'extension CMC

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Une valeur de données de ce type représente une extension. La même structure d'extension est utilisée pour spécifier et recevoir des informations se rapportant à des appels de fonction CMC *et* des structures de données CMC.

Les appels de fonction et les structures de données permettent d'une manière générale de faire des extensions en entrée *et* en sortie, la direction étant donnée d'une manière implicite par le code article de l'extension. Les extensions en entrée peuvent faire référence à des zones de mémoire allouées par l'application et les extensions en sortie peuvent faire référence des zones de mémoire allouées par le service CMC. Certaines mises en œuvre de la fonction **cmc_act_on()** peuvent, par exemple, permettre de sauvegarder, en vue d'une lecture et d'une émission ultérieure, des messages partiellement remplis en utilisant l'extension CMC_X_COM_SAVE_MESSAGE pour transférer de l'information dans la structure de message et recevoir en retour la référence de message correspondante. Prière de consulter 4.11 et 4.14 pour une liste complète des extensions de messages communes spécifiées dans la présente Recommandation.

Les utilisateurs de tableaux d'extensions CMC pouvant contenir de la mémoire d'extension en sortie allouée par le service CMC doivent utiliser la fonction **cmc_free()** pour libérer le pointeur renvoyé dans le champ référence de l'entité. Ces structures sont identifiées par le positionnement du fanion de sortie CMC_EXT_OUTPUT et par une référence d'article non égale à NULL. Les appelants demandent explicitement une extension de fonction en sortie lors d'un appel de fonction en positionnant le code article de l'extension adéquat. Toutes les structures subordonnées contenues dans la mémoire allouée seront libérées lorsque le pointeur de la structure de base est libéré.

Les extensions de données n'ont pas besoin d'être libérées d'une manière explicite, parce qu'elles sont libérées en même temps que la structure qui les contient. Le tableau d'extensions de message résultant d'un appel **cmc_read()** sera, par exemple, libéré d'une manière implicite lorsque la fonction **cmc_free()** est appelée pour la structure de message qui le contient.

Une extension se compose des éléments suivants:

1) **code article** (*item_code*) – Code identifiant sans ambiguïté cette extension.

2) **données de l'article** (*item_data*) – En fonction de la valeur du champ code article, le champ données de l'article peut contenir la longueur de la valeur de l'article, la valeur de l'article lui-même ou une autre information au sujet de l'article. La spécification de l'extension décrit de quelle manière interpréter ce champ.

3) **référence de l'article** (*item_reference*) – En fonction du champ code article, le champ référence de l'article peut contenir un pointeur vers l'emplacement mémoire où est stocké l'article ou NULL s'il n'existe pas de mémoire associée à l'article. La spécification de l'extension décrit de quelle manière interpréter ce champ.

4) **fanions d'extension** (*extension flags*) – Bits pour des attributs booléens. Les 16 bits d'ordre supérieur sont réservés pour des extensions faites par la spécification de l'interface CMC. Tout bit non utilisé devra être non positionné. Les 16 bits de fanion d'ordre inférieur sont réservés pour une définition faite par l'extension.

   a) CMC_EXT_REQUIRED

      positionné: renvoi d'une erreur si cette extension ne peut pas être prise en charge.

      non positionné: permettre une prise en charge de cette extension avec le "meilleur effort", y compris l'absence de prise en charge.

   b) CMC_EXT_OUTPUT

      positionné: indique, pour des extensions en sortie, que cette extension contient un pointeur vers une zone mémoire allouée par la mise en œuvre CMC qui devra être libérée en utilisant la fonction **cmc_free()**.

      non positionné: la mise en œuvre n'a pas alloué de mémoire devant être libérée par l'application. Ce fanion est toujours non positionné, comme décrit plus haut.

   c) CMC_EXT_LAST_ELEMENT

      positionné: identifie la dernière structure d'un tableau de telles structures. Celle-ci doit se trouver à la fin du tableau d'extensions.

      non positionné: il ne s'agit pas du dernier élément du tableau.

4.15 Fanions (*Flags*)

NOM

Fanion – Définition de type pour un fanion CMC.

DÉCLARATION EN LANGAGE C

```
typedef   CMC_uint32      CMC_flags;
```

DESCRIPTION

Une valeur de données de ce type contient 32 bits de fanion. La signification des bits dépend du contexte dans lequel les valeurs de données de fanion sont utilisées. Les fanions non documentés sont réservés. Un fanion positionné sur zéro est dit "non positionné". Les fanions positionnés sur une valeur non nulle sont dits "positionnés". Les fanions non utilisés doivent toujours être non positionnés.

4.16 Identificateur Guid (*Guid*)

NOM

Identificateur Guid – Définition de type pour une structure d'identificateur CMC globalement non ambigu (GUID)

DÉCLARATION EN LANGAGE C

```
typedef CMC_string      CMC_guid
```

DESCRIPTION

Une valeur de données de ce type représente un identificateur global non ambigu. La chaîne de caractères est formatée conformément à la définition du texte d'identificateur public formel de l'ISO 9070, de manière à en garantir l'unicité. Les identificateurs GUID utilisés par l'interface CMC ont le format suivant:

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

dans lequel *name type* est le type de nom et *name* est le nom de l'objet auquel est assigné l'identificateur GUID. Comme exemple, la classe d'objets CONTENT ITEM possède l'identificateur suivant:

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

Certaines des valeurs d'identificateur GUID de l'interface CMC peuvent être définies sous la forme d'un identificateur d'objet ISO/OSI (OID). L'identificateur OID peut être encapsulé dans un identificateur public formel ISO 9070. Cette encapsulation est réalisée de la manière suivante:

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

dans laquelle **<oid>** est la forme numérique d'identificateur OID OSI telle qu'elle est définie par le type de données identificateur d'objet dans le paragraphe 4.24.

4.17 Identificateur (*Identifier*)

NOM

Identificateur – Définition de type pour un identificateur non ambigu propre à la mise en œuvre

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Une valeur de données de ce type représente un identificateur non ambigu propre à la mise en œuvre. Ce type de données est utilisé pour des identificateurs localement non ambigus tels que des identificateurs de propriété ou de classes d'objets.

4.18 Date et heure ISO (*date and time*)

NOM

Date et heure ISO – Définition de type pour une valeur de date et d'heure formatée conformément à l'ISO 8601

DÉCLARATION EN LANGAGE C

```
typedef CMC_string      CMC_date_time;
```

DESCRIPTION

Une valeur de données de ce type de données est une valeur de date et d'heure conforme à la représentation combinée de date et d'heure de l'ISO 8601. Le format de ce type de données prend en charge l'heure du jour, représentée sous l'une des formes suivantes: heure locale, heure d'horloge utilisée localement d'une manière publique, temps universel coordonné (UTC, *coordinated universal time*) – référence de temps maintenue par le Bureau International de l'Heure formant la base d'une diffusion coordonnée de fréquences étalon et de signaux horaires (de temps normalisés) – ou différence de l'heure locale avec le temps UTC.

La valeur de la donnée est la concaténation des représentations de la **date** et de l'**heure**. Le caractère [T] est utilisé dans la désignation du temps pour indiquer le début de la représentation de l'heure du jour dans l'expression combinée de la date et de l'heure. Le caractère [Z] est utilisé pour indiquer la désignation de fuseau horaire si le temps est exprimé en temps UTC. Par exemple, dans la chaîne de caractères ccyymmddThhmmssZ, la chaîne [cc] indique le siècle, [yy] indique l'année, [mm] indique le mois, [dd] indique le jour du mois, [hh] indique l'heure dans un format de 24 heures, [mm] indique les minutes dans l'heure, et [ss] indique les secondes dans la minute.

Dans le cas de l'heure locale fournie comme différence avec le temps UTC, la date et l'heure sont représentées par des chaînes de caractères de la forme ccyymmddThhmmss+hhmm, ccyymmddThhmmss+hh, ccyymmddThhmmss–hhmm, ou ccyymmddThhmmss–hh, [cc] étant le siècle, [yy] l'année, [mm] le mois, [dd] le jour du mois, [hh] l'heure dans un format de 24 heures, [mm] les minutes dans l'heure et [ss] les secondes dans la minute. Le descripteur de fuseau horaire n'est pas présent et la chaîne de date et d'heure est concaténée avec l'heure et la minute ou le décalage horaire par rapport au temps UTC. La différence entre l'heure locale et le temps UTC est exprimée en heures et minutes ou en heures seulement, indépendamment de la précision de l'expression de l'heure locale qui lui est associée. Elle est sous une forme positive (c'est-à-dire avec un signe plus [+] en tête) si l'heure locale est en avance, et sous une forme négative (c'est-à-dire avec un signe moins [–] en tête) si l'heure locale est en retard par rapport au temps UTC. Par exemple, la chaîne 19850414T152746+0100 représente la date du 14 avril 1985 et 27 minutes et 46 secondes après 15 heures, heure locale, à un endroit qui est normalement en avance d'une heure par rapport au temps UTC. La chaîne 19850414T152746–05 représente la date du 14 avril 1985 et 27 minutes et 46 secondes après 15 heures, heure locale, à un endroit qui est normalement en retard de 5 heures par rapport au temps UTC.

1) **date** (*date*) – Date calendaire exprimée sous la forme de la représentation complète du format de base, tel qu'il est défini dans le paragraphe 5.2.1.1 de l'ISO 8601. Le 14 avril 1985 est représenté par la chaîne 19850414.

2) **time** (*temps*) – L'heure du jour, exprimée soit en heure locale, en temps universel coordonné (UTC) local ou en différence de l'heure locale avec le temps UTC. Le format de temps est la représentation complète du format de base tel qu'il est défini dans les paragraphes 5.3.3 et 5.3.3.1 de l'ISO 8601. Par exemple, le temps UTC de 20 minutes et 30 secondes après 23 heures est représenté par la chaîne 232030Z. L'heure locale de 10 minutes et 15 secondes après 12 heures est représentée par la chaîne 121510. La même heure locale est représentée comme différence avec le temps UTC par la chaîne 121510–06 ou 121510–0600 si l'heure locale est en retard de six heures par rapport au temps UTC.

4.19 **Message** (*Message*)

NOM

Message – Définition de type pour une structure de message CMC

DÉCLARATION EN LANGAGE 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;
```

DESCRIPTION

Une valeur de données de ce type représente un message. Cette structure de données est destinée à permettre la prise en charge de mises en œuvre des interfaces CMC 1.0 et CMC simple. Un message se compose des éléments suivants:

1) **référence de message** (*message_reference*): identifie le message. La référence de message est non ambiguë au sein d'une boîte aux lettres.

2) **type de message** (*message_type*): chaîne de caractères identifiant le type de message. Les trois identificateurs de chaîne de caractères suivants peuvent être utilisés:

   a) identificateurs d'objet – Utilisés pour des types identifiés par des identificateurs d'objet tels qu'ils sont définis dans la Recommandation X.208;

   b) valeurs CMC enregistrées – Utilisées pour des types définis dans la présente Recommandation;

   c) valeurs définies d'une manière bilatérale – Utilisées pour des types qui ne sont pas enregistrés.

NOTE – Les valeurs enregistrées d'une manière bilatérale ne garantissent pas la non-ambiguïté.

Le format de chaque type est donné ci-dessous. Les espaces de séparation peuvent être constitués d'une combinaison quelconque de tabulations et d'espaces. Le caractère "*" indique que l'élément syntactique en question peut apparaître une fois ou plus, séparé par des blancs. Il n'est pas fait de distinction entre majuscules et minuscules pour les chaînes de caractères délimitées par des apostrophes doubles.

```
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"
```

Ces valeurs enregistrées sont définies comme suit:

"CMC: IPM" message interpersonnel: un message interpersonnel est un message comparable à une note contenant une liste de destinataires, un sujet optionnel, un texte de note optionnel et zéro ou plusieurs attachements. La structure "message" est optimisée pour l'accueil de messages du type IPM;

"CMC: IP RN" avis d'acquittement d'un message interpersonnel: un avis d'acquittement indique qu'un message a été lu par le destinataire;

"CMC: IP NRN" avis de non-acquittement d'un message interpersonnel: un avis de non-acquittement indique qu'un message a été enlevé de la boîte aux lettres du destinataire sans avoir été lu (le message a, par exemple, été rejeté par l'utilisateur ou le service, ou a été relayé automatiquement vers un autre destinataire);

"CMC: DR" compte rendu de remise: un compte rendu de remise indique que le service a été en mesure de livrer le message au destinataire;

"CMC: NDR" compte rendu de non-remise: un compte rendu de non-remise indique que le service n'a pas été en mesure de livrer le message au destinataire;

"CMC: REPORT" comptes rendus de remise et de non-remise lorsque le message original a de multiples destinataires: cette valeur indique que le service sous-jacent de messagerie a été en mesure de livrer le message à certains destinataires, mais pas à d'autres.

Le format dans lequel se présentent ces messages au sein des structures définies dépend des protocoles de messagerie utilisés par le service de messagerie. Les messages qui ne sont pas des messages interpersonnels sont souvent des messages générés par un programme, qui respectent un format analogue à celui d'une note (comparable à celui d'un message interpersonnel), mais ils sont émis dans le but de véhiculer une information concernant un message émis précédemment.

NOTE – Ces types de messages correspondent à des messages X.400, mais peuvent être utilisés avec des messageries n'utilisant pas le protocole X.400. Il en résulte que ces types de messages CMC sont conçus pour une utilisation plus générale que pour le protocole spécifique X.400.

Les identificateurs suivants sont des exemples valides:

OID: 1 2 840 113556 3 2 850

CMC: IPM

BLT: mon type particulier de message

Une forme canonique a également été définie pour ces types de messages afin de faciliter la comparaison de chaînes de caractères. La mise en œuvre CMC procède dans tous les cas à une réduction à la forme canonique, dans laquelle:

1) tous les blancs de séparation sont convertis en un espace unique, et tous les éléments syntactiques sont séparés par un espace;

2) les identificateurs de type (c'est-à-dire OID, CMC, BLT) sont convertis en majuscules.

Certaines mises en œuvre CMC ne prendront en charge que le type de message interpersonnel (CMC: IPM). Ces mises en œuvre peuvent traiter d'autres types de messages comme des messages interpersonnels ou générer une erreur.

Le traitement fait par une mise en œuvre n'est pas défini pour des chaînes de caractères qui ne sont pas dans l'un de ces formats.

3) **sujet** (*subject*): chaîne de caractères indiquant le sujet du message.

4) **instant d'émission** (*time_sent*): date et heure d'émission du message (ou de dépôt).

5) **note de texte** (*text_note*): chaîne de caractères du texte de la note du message. L'absence de note de texte est indiquée par une valeur NULL. La note de texte se trouve dans le premier attachement si le fanion CMC_TEXT_NOTE_AS_FILE est positionné.

   Le format de la note de texte est constitué d'une succession de paragraphes, indépendamment du fait qu'elle est transmise en mémoire ou par fichier. Les paragraphes se terminent par une fin de ligne appropriée à la plate-forme (retour CR pour le Macintosh, nouvelle ligne LF pour Unix et combinaison CR/LF pour DOS et Windows, etc.). La mise en œuvre CMC peut couper des lignes entre mots et procéder à un saut de ligne entre mots pour des lignes longues (paragraphes).

NOTE – Il n'y a pas de garantie de fidélité (les fonctions de lecture de la mise en œuvre CMC peuvent renvoyer, par exemple, un paragraphe long sous la forme d'une série de paragraphes plus courts).

6) **destinataires** (*recipients*): pointeur vers le premier élément d'un tableau de destinataires d'un message.

7) **attachements** (*attachments*): pointeur vers le premier élément d'un tableau d'attachements d'un message.

8) **fanions de message** (*message_flags*): bits utilisés pour des attributs booléens. Les bits inutilisés doivent être non positionnés.

   a) CMC_MSG_READ

      positionné: le message a été lu.

      non positionné: le message n'a pas été lu.

   b) CMC_MSG_TEXT_NOTE_AS_FILE

      positionné: le champ note de texte est ignoré et le texte de la note de texte se trouve dans le fichier référencé par le premier attachement.

      non positionné: le texte de la note de texte se trouve dans le champ note de texte.

   c) CMC_MSG_UNSENT

      positionné: le message n'a pas été émis (c'est-à-dire qu'il s'agit d'un projet). Ce type de message peut être créé au moyen de l'extension CMC_X_COM_SAVE_MESSAGE.

      non positionné: le message a été émis.

   d) CMC_MSG_LAST_ELEMENT

      positionné: identifie la dernière structure d'un tableau de telles structures. Celle-ci doit se trouver à la fin du tableau d'extensions.

      non positionné: il ne s'agit pas du dernier élément du tableau.

9) **extensions de message** (*message_extensions*): pointeur vers le premier élément d'un tableau d'extensions d'un message.

4.20 **Référence de message** (*Message Référence*)

NOM

Référence de message – Définition de type pour une structure de référence de message CMC

DÉCLARATION EN LANGAGE C

```
typedef   CMC_counted_string  CMC_message_reference;
```

DESCRIPTION

Une valeur de données de ce type représente une chaîne de caractères avec comptage contenant le descripteur opaque de message utilisé par la boîte aux lettres. Cette structure de données est fournie pour la prise en charge de mises en œuvre de l'interface CMC 1.0 et de l'interface CMC simple. La validité d'une référence de message n'est garantie que pendant la durée de vie de la session et il n'existe pas de garantie qu'elle corresponde à un identificateur de message utilisé par le système de messagerie sous-jacent. Le programme d'application peut copier la référence de message durant la session.

4.21 **Résumé de message** (*Message summary*)

NOM

Résumé de message – Définition de type d'une structure de résumé de message CMC

DÉCLARATION EN LANGAGE 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;
```

DESCRIPTION

Une valeur de données de ce type représente un résumé de message. Cette structure de données est fournie pour la prise en charge de mises en œuvre de l'interface CMC 1.0 et de l'interface CMC simple. Un résumé de message se compose des éléments suivants:

1) **référence de message** (*message_reference*): voir la définition donnée pour la structure de message.
2) **type de message** (*message_type*): voir la définition donnée pour la structure de message.
3) **sujet** (*subject*): voir la définition donnée pour la structure de message.
4) **instant d'émission** (*time_sent*): voir la définition donnée pour la structure de message.
5) **longueur en octets** (*byte_length*): taille du message. Cette valeur doit tenir compte de toutes les informations associées au message: attachements, champs d'enveloppe et d'en-tête, etc. Les mises en œuvre peuvent renvoyer une valeur approchée ou la constante CMC_LENGTH_UNKNOWN si l'information de taille du message est inconnue ou indisponible.
6) **expéditeur** (*originator*): expéditeur du message.
7) **fanions de résumé de message** (*summary_flags*): bits utilisés pour des attributs booléens. Les bits non utilisés doivent être non positionnés.
   a) CMC_SUM_READ

      positionné: le message a été lu.

      non positionné: le message n'a pas été lu.

   b) CMC_SUM_UNSENT

      positionné: le message n'a pas été émis (il s'agit d'un projet).

      non positionné: le message a été émis.

   c) CMC_SUM_LAST_ELEMENT

      positionné: identifie la dernière structure d'un tableau de telles structures.

      non positionné: il ne s'agit pas du dernier élément du tableau.

   d) CMC_SUM_HAS_ATTACHMENTS

      positionné: le message possède des attachements.

      non positionné: le message ne possède pas d'attachement.
8) **extensions de résumé de message** (*message_summary_extensions*): pointeur vers le premier élément d'un tableau d'extensions de résumé par message.

4.22 **Nom** (*Name*)

NOM

Nom – Définition de type pour un nom CMC 2.0 non ambigu

DÉCLARATION EN LANGAGE C

```
typedef CMC_string     CMC_name;
```

DESCRIPTION

Une valeur de données de ce type représente un nom non ambigu. La chaîne de caractères est formatée conformément à l'identificateur public formel de texte défini par l'ISO 9070, afin d'en garantir la non-ambiguïté. Les noms CMC ont le format suivant:

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

dans lequel *name type* est le type de nom, *name* le nom de l'objet. Par exemple, le nom de la classe d'objets CONTENT ITEM est:

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

4.23 Descripteur opaque d'objet (*Object handle*)

NOM

Descripteur opaque d'objet – Définition de type pour une structure de descripteur opaque d'objet CMC

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Une valeur de données de ce type représente un descripteur opaque d'objet. Les descripteurs opaques d'objet sont non ambigus pour le service de messagerie. Les descripteurs opaques sont persistants pour la durée de la session ou jusqu'au moment de leur suppression. Le descripteur opaque fournit le contexte pour un objet CMC. Le descripteur opaque encapsule l'identificateur de session. La fonction **cmc_copy_object_handle()** est utilisée pour copier un descripteur opaque d'objet.

La non-existence d'un descripteur opaque doit pouvoir être mémorisée dans un descripteur opaque d'objet. La constante CMC_NULL_HANDLE définie par le système est utilisée à cet effet.

4.24 Identificateur d'objet (*Object identifier*)

NOM

Identificateur d'objet – Définition de type pour une structure d'identificateur d'objet CMC

DÉCLARATION EN LANGAGE C

```
typedef   CMC_string      CMC_object_identifier;
```

DESCRIPTION

Une valeur de données de ce type représente un identificateur d'objet tel qu'il est défini dans la Recommandation X.208. Cette structure de données est fournie pour la prise en charge de mises en œuvre de l'interface CMC 1.0 et de l'interface CMC simple. L'identificateur est globalement non ambigu. La syntaxe utilisée dans la présente Recommandation correspondra à la forme numéro de la Recommandation X.208, avec le format suivant:

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

Un exemple d'identificateur d'objet est le suivant:

1 2 840 113556 3 2 850

NOTE – Le format utilisé pour la chaîne de caractères d'identificateur d'objet est le même que pour le type de message OID.

4.25 Données opaques (*Opaque data*)

NOM

Données opaques – Définition de type pour une valeur de données opaques

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Une valeur de données de ce type de données est une valeur de données opaques. Une structure de données opaques se constitue des composants suivants:

1) **taille** (*size*) – Spécifie le nombre d'octets des données opaques sur lesquelles pointe le composant **données**.

2) **données** (*data*) – Pointeur vers un tableau de valeurs de 8 bits. Les données n'ont pas de sémantique explicite.

4.26 Propriété (*Property*)

NOM

Propriété – Définition de type pour un type de données de propriété CMC

DÉCLARATION EN LANGAGE 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;
```

DESCRIPTION

Une valeur de données de ce type représente un tableau de propriétés CMC. Une propriété est la méthode de spécification de l'information de contenu propre à un tableau CMC. Une propriété se compose des éléments suivants:

1) **identificateur** (*id*): identifie la propriété d'une manière non ambiguë.

2) **type** (*type*): spécifie le type de données de la propriété.

3) **valeur** (*value*): définit la valeur de la propriété.

4) **extensions de propriété** (*property_extensions*): pointeur vers le premier élément d'un tableau d'extensions de propriété.

4.27 Destinataire (*Recipient*)

NOM

Destinataire – Définition de type pour une structure expéditeur/destinataire

DÉCLARATION EN LANGAGE 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;
```

DESCRIPTION

Une valeur de données de ce type représente un expéditeur ou un destinataire. Cette structure de données est fournie pour la prise en charge de mises en œuvre de l'interface CMC 1.0 et de l'interface CMC simple. Elle se compose des éléments suivants:

1) **nom** (*name*): nom affichable du destinataire. L'ordre dans lequel est interprété ce nom comme nom d'individu, puis comme nom de groupe, ou l'inverse, est une affaire de mise en œuvre lors de la résolution du nom pour obtenir une adresse.

2) **type de nom** (*name_type*): données énumérées indiquant le type de destinataire:

CMC_TYPE_UNKNOWN (= 0)	type de destinataire non connu.
CMC_TYPE_INDIVIDUAL	le destinataire est un individu.
CMC_TYPE_GROUP	le nom représente un groupe de destinataires.

NOTE – Ceci n'a de sens que si un nom est présent. Le type de nom est positionné en sortie par la mise en œuvre. Il peut être utilisé en entrée comme une indication pour l'optimisation de la résolution de nom.

3) **adresse** (*address*): adresse du destinataire compréhensible pour le système de messagerie sous-jacent. La présente Recommandation ne définit pas le format de cette chaîne de caractères d'adresse qui est prévue pour recevoir toute notation de chaîne de caractères prise en charge par une mise en œuvre donnée et configurée pour une installation donnée. Les utilisateurs finaux sont invités à consulter l'administrateur de leur service local afin de savoir quelle notation de chaîne de caractères est prise en charge par leur installation.

4) **rôle** (*role*): données énumérées définissant le rôle du destinataire:

CMC_ROLE_TO	destinataire primaire (TO).
CMC_ROLE_CC	destinataire en copie (CC).
CMC_ROLE_BCC	destinataire en copie muette (BCC).
CMC_ROLE_ORIGINATOR	expéditeur du message.
CMC_ROLE_AUTHORIZING_USER	utilisateur autorisant un message.
CMC_ROLE_REPLY_TO	destinataire de la réponse.

Un destinataire CC peut être converti (d'une manière silencieuse) en un destinataire principal TO si le service de messagerie sous-jacent ne prend pas en charge des destinataires CC. Les services qui ne peuvent pas prendre en charge de destinataires BCC doivent rejeter les messages qui en contiennent. Un même destinataire ne peut être présent plusieurs fois que s'il joue des rôles différents.

La mise en œuvre CMC doit renvoyer le tableau des destinataires dans l'ordre de sortie suivant. L'expéditeur doit être le premier élément du tableau, suivi des destinataires regroupés dans l'ordre REPLY TO, TO, CC, et BCC. L'utilisateur donnant l'autorisation doit, s'il existe, être le dernier destinataire figurant dans le tableau. Il n'est requis aucun ordre en entrée.

5) **fanions de destinataire** (*recip_flags*): bits pour des attributs booléens. Les bits inutilisés doivent être non positionnés.

   a) CMC_RECIP_IGNORE

      positionné: ignorer ce destinataire (cet attribut est utile en cas de réutilisation, dans une réponse, de la liste des destinataires du message entrant).

      non positionné: ne pas ignorer ce destinataire.

   b) CMC_RECIP_LIST_TRUNCATED

      positionné: indique que toutes les structures de destinataire demandées n'ont pas été renvoyées par le système. Ce bit n'est utilisé par la fonction **cmc_look_up()** que si la liste complète des destinataires correspondant au nom de recherche n'a pas pu être renvoyée. Ce fanion n'est positionné que dans la dernière structure du tableau.

      non positionné: le tableau de destinataires a été renvoyé dans sa totalité.

   c) CMC_RECIP_LAST_ELEMENT

      positionné: identifie la dernière structure d'un tableau de telles structures.

      non positionné: il ne s'agit pas du dernier élément du tableau.

6) **extensions de destinataire** (*recip_extensions*): pointeur vers le premier élément d'un tableau d'extensions pour un destinataire.

4.28 Compte rendu (*Report*)

NOM

Compte rendu – Définition de type pour une structure de compte rendu et de compte rendu de non-remise

DÉCLARATION EN LANGAGE 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;
```

DESCRIPTION

Une valeur de données de ce type représente un compte rendu, un compte rendu de non-remise ou les deux. Cette structure de données est fournie pour la prise en charge de mises en œuvre de l'interface CMC 1.0 et de l'interface CMC simple. Un compte rendu se compose des éléments suivants:

1) **type de compte rendu** (*report_type*): valeur énumérée identifiant le type de compte rendu. Ce type peut être l'un des suivants:

   CMC_X400_DR ((CMC_enum) 0)

   CMC_X400_NDR ((CMC_enum) 1)

2) **instant de remise** (*delivered_time*): date et heure de remise du message original au destinataire. La valeur est NULL dans le cas du type CMC_X400_NDR, ou donne la date et l'heure de remise dans le cas du type CMC_X400_DR.

3) **code motif** (*reason_code*): motif de non-remise d'un message. Cette valeur est égale à ZÉRO pour un type CMC_X400_DR, ou l'une des valeurs suivantes pour un type CMC_X400_NDR:

   code_motif.<16 bits d'ordre élevé> = X.411.NonDeliveryReasonCode (*code motif de non-remise X.411*).

   code_motif.<16 bits d'ordre inférieur> = X.411.NonDeliveryDiagnosticCode (*code diagnostic de non-remise X.411*).

4) **fanions de compte rendu** (*report_flags*): bits pour des attributs booléens. Les bits inutilisés doivent être non positionnés.

– CMC_REPORT_LAST_ELEMENT

positionné: identifie la dernière structure d'un tableau de telles structures.

non positionné: il ne s'agit pas du dernier élément du tableau.

NOTE – L'interface CMC définit des types spécifiques de message pour les comptes rendus de remise ("CMC:DR") et les comptes rendus de non-remise ("CMC:NDR") qui peuvent être traités d'une manière indépendante car ils sont vus comme des messages distincts. L'information de remise et de non-remise est véhiculée, pour la Recommandation X.400, dans une base d'information générique de compte rendu. Il est possible qu'un compte rendu X.400 contienne des comptes rendus de remise pour certains destinataires et des comptes rendus de non-remise pour d'autres destinataires lorsqu'un message est envoyé à des destinataires multiples d'un même agent MTA. Ceci ne correspond pas bien avec les fanions CMC:DR ou CMC:NDR en sortie (X.400 vers CMC) parce que la Recommandation X.400 ne peut ni les voir, ni les stocker sous forme d'une base d'information distincte et ne peut donc les traiter d'une manière indépendante. En conséquence, un nouveau type de message "CMC: REPORT" a été ajouté afin de traiter les prescriptions de compte rendu X.400.

4.29 Code retour (*Return code*)

NOM

Code retour – Définition de type pour une valeur renvoyée par toutes les fonctions CMC

DÉCLARATION EN LANGAGE C

```
typedef   CMC_uint32      CMC_return_code;
```

DESCRIPTION

Un code retour est défini comme valeur à 32 bits. Une valeur non nulle indique une erreur, le code erreur correspondant à la valeur renvoyée. Une valeur nulle indique une réussite. Les valeurs contenues dans les 16 bits d'ordre inférieur sont réservées à des codes erreur définis dans la présente Recommandation. Les valeurs contenues dans les 16 bits d'ordre supérieur sont réservées à des codes erreur définis par la mise en œuvre avec les 16 bits d'ordre inférieur positionnés sur une valeur d'erreur CMC appropriée.

Des erreurs peuvent être résolues dans le domaine d'application d'un appel CMC en utilisant, par exemple, un dialogue réalisé au moyen de l'interface utilisateur. Si un dialogue est invoqué pour résoudre l'erreur, mais que celle-ci reste non résolue une fois le dialogue terminé, le bit de fanion défini dans CMC_ERROR_UI_DISPLAYED est positionné dans l'erreur afin d'indiquer qu'elle a déjà été présentée à l'utilisateur.

4.30 Identificateur de session (*Session Id*)

NOM

Identificateur de session – Définition de type pour un identificateur de session CMC

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Identificateur opaque de session. Le contexte indiqué par l'identificateur de session contient des informations propres à la session, telles que le jeu de caractères utilisé et les descripteurs opaques de toutes les sessions ouvertes avec les services de messagerie sous-jacents. L'identificateur de session CMC est créé par la fonction CMC d'ouverture de la session CMC et détruit par la fonction de fin de session CMC.

Prière de se référer au B.2.4 pour ce qui est de la définition concernant une plate-forme spécifique.

4.31 Descripteur opaque de flux (*Stream handle*)

NOM

Descripteur opaque de flux – Définition de type pour une structure de descripteur opaque de flux CMC

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Une valeur de données de ce type représente un descripteur opaque de flux. Les descripteurs opaques de flux sont non ambigus pour le service de message. Les descripteurs opaques sont persistants pendant la durée de la session ou jusqu'au moment de leur suppression. Le descripteur opaque fournit le contexte pour un flux de contenu d'information. Le flux encapsule l'identificateur de session et les descripteurs opaques d'objet. Les descripteurs opaques de flux ne peuvent pas être copiés.

4.32 **Chaîne** (*String*)

NOM

Chaîne – Définition de type pour une chaîne de caractères CMC

DÉCLARATION EN LANGAGE C

```
typedef   cmc_string*      CMC_string;
```

DESCRIPTION

Une valeur de données de ce type représente une chaîne de caractères. Le tableau de caractères désigné est interprété par défaut comme un tableau de caractères terminé par un caractère nul. Toutes les mises en œuvre doivent prendre en charge des chaînes de caractères terminées par un caractère nul. La taille de la représentation des caractères et le caractère nul correspondant sont déterminés par le jeu de caractères choisi.

Si une application souhaite utiliser des chaînes de caractères avec comptage au lieu de la terminaison par un caractère nul, et que la mise en œuvre CMC les prend en charge, l'application positionnera le fanion CMC_COUNTED_STRING_TYPE au moment de l'ouverture de la session. Les données désignées par le pointeur de chaîne de caractères CMC seront alors supposées être dans le format de données de chaîne de caractères avec comptage. Ce fanion doit être positionné dans le paramètre fanions si une ouverture de session implicite est faite au moyen d'une fonction.

La mise en œuvre CMC examine le contexte de la session pour déterminer quel est le jeu de caractères utilisé par la chaîne. La chaîne sera interprétée en utilisant le jeu de caractères par défaut si un contexte de session n'a pas été créé avant l'appel. La mise en œuvre doit toujours tenter d'utiliser le mappage du jeu de caractères par défaut de la session pour toutes les chaînes de caractères transmises à l'application cliente.

4.33 **Temps** (*Time*)

NOM

Temps – Définition de type pour une structure de temps CMC

DÉCLARATION EN LANGAGE 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;
```

DESCRIPTION

Une valeur de données de ce type représente une valeur de temps. Cette structure de données est fournie pour la prise en charge de mises en œuvre de l'interface CMC 1.0 et de l'interface CMC simple. Une valeur de temps contient les composants suivants:

1) *second*: secondes; domaine 0..59.

2) *minute*: minutes; domaine 0..59.

3) *hour*: heures à partir de minuit; domaine 0..23.

4) *day*: jours du mois; domaine 1..31.

5) *month*: mois à partir de janvier; domaine 0..11.

6) *year*: années à partir de 1900.

7) *isdst*: indicateur de passage à l'heure d'été; pas d'heure d'été si l'indicateur n'est pas nul.

8) *tmzone*: fuseau horaire, en minutes par rapport au temps moyen de Greenwich. La valeur prédéfinie CMC_NO_TIMEZONE indique que l'information de fuseau horaire n'est pas disponible.

Toutes les valeurs de temps sont données dans l'heure locale adéquate. Le champ instant d'émission des structures de message CMC et de résumé de message CMC est donné dans l'heure locale de l'émetteur.

NOTE – Il est possible de convertir la valeur de temps en heure locale de l'appelant si le champ fuseau horaire contient une valeur autre que CMC_NO_TIMEZONE. La fonction effectivement utilisée pour la conversion est en dehors du domaine d'application de la présente Recommandation.

4.34 Identificateur d'interface utilisateur (*User interface Id*)

NOM

Identificateur d'interface utilisateur – Définition de type pour un descripteur opaque d'interface utilisateur CMC

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Valeur utilisée pour transmettre une information d'interface utilisateur aux fonctions CMC. Dans un environnement utilisant des fenêtres, par exemple, cette information sera le descripteur opaque de la fenêtre mère de l'application appelante.

Une valeur NULL est toujours valide et correspond au comportement par défaut défini par la mise en œuvre.

NOTE – Les mises en œuvre CMC n'ont pas l'obligation de fournir une interface utilisateur, et si cette interface est fournie pour une fonction, ceci n'implique pas qu'elle l'est pour toutes les fonctions de l'interface CMC.

Prière de se référer au B.2.4 en ce qui concerne une plate-forme donnée.

5 Propriétés d'objet

Le présent paragraphe définit les propriétés d'objet pour les classes d'objets de l'interface API de l'appel commun de messagerie. Un objet est une collection de propriétés. Les propriétés des objets sont définies ci-après en vue de normaliser leur représentation dans la présente Recommandation.

Les définitions de propriétés d'objet sont précédées par des tableaux résumant les propriétés de chaque classe d'objets. Les tableaux résumant les propriétés d'objet donnent, dans les deux premières colonnes, la liste des noms de propriété et des types des valeurs de toutes les propriétés définies. La troisième colonne fournit une description de la propriété. La quatrième colonne donne la liste de toutes les valeurs possibles pour chaque propriété. Les astérisques indiquent les valeurs par défaut. La propriété ne possède pas de valeur par défaut si aucun astérisque n'est présent. La cinquième colonne indique si la propriété est obligatoire (M) ou optionnelle (O). La sixième colonne indique si la propriété n'est accessible qu'en écriture. Une indication "non" dans cette colonne signifie que la propriété peut être modifiée, mise à jour et supprimée respectivement au moyen d'un appel à une fonction **cmc_update_properties(), cmc_add_properties(),** ou **cmc_delete_properties()**, sauf indication contraire. La dernière colonne spécifie le créateur de la propriété qui peut être la mise en œuvre (I), l'appelant (C) ou l'un ou l'autre (E).

Les propriétés peuvent posséder des valeurs par défaut. Une mise en œuvre doit, toutefois, remplir l'objet avec les valeurs explicites pour toutes les propriétés prises en charge qui possèdent des valeurs par défaut. Ceci simplifiera les énumérations de propriétés par l'application.

Tableau 4/X.446 – Résumé de la propriété carnet d'adresses CMC

Carnet d'adresses					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
enfant autorisé	boolean	CMC_TRUE, CMC_FALSE	O	non	CMC_FALSE
commentaire	string	toute chaîne de caractères valide	O	non	aucune
emplacement	enum	LOCAL SERVER UNKNOWN[a)]	O	non	UNKNOWN[a)]
nom	string	toute chaîne de caractères valide	O	non	chaîne de caractères vide
classe d'objets	enum	ADDRESS BOOK[b)]	M	oui	non disponible
parent	object_handle	tout descripteur opaque d'objet valide	M, en cas d'imbrication	non	néant
nom du serveur	string	toute chaîne de caractères valide	O	non	néant
partagé	boolean	CMC_TRUE, CMC_FALSE	O	non	CMC_FALSE
type	enum	GLOBAL, PERSONAL[c)]	O	non	PERSONAL[b)]

a) préfixé par la valeur "CMC_ADDRESS_BOOK_LOCATION_"

b) préfixé par la valeur "CMC_OBJECT_TYPE_"

c) préfixé par la valeur "CMC_ADDRESS_BOOK_TYPE_"

Tableau 5/X.446 – Résumé de la propriété carnet d'adresses CMC

Article de contenu					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
jeu de caractères	guid	GUID pour tout jeu de caractères	O	non	dépend de la plate-forme
information de contenu	opaque_data	toute donnée	O	non	néant
type de contenu	guid	GUID pour tout type contenu	O	non	néant
instant de création	iso_date_time	toute date et heure ISO 8601	O	oui	néant
type de codage	guid	GUID pour tout type de codage	O	non	7-BIT[a)]
répertoire du fichier	string	tout répertoire de fichier valide	O	non	néant
nom de fichier	string	tout nom de fichier valide	O	non	néant
numéro d'article	uint32	jusqu'à un maximum défini par la mise en œuvre	M, pour plus d'une entité de contenu	non	néant
type d'article	enum	NOTE ATTACHMENT ANNOTATION[b)]	O	non	NOTE
dernière modification	iso_date_time	toute date et heure ISO 8601	O	oui	néant
classe d'objets	enum	CONTENT ITEM[c)]	M	oui	non disponible
position de rendu	uint32	position de bit au sein du conteneur	O	non	néant
taille	uint32	taille de caractère	O	non	néant
titre	string	toute chaîne de caractères valide	O	non	néant

a) préfixé par la valeur "CMC_ADDRESS_BOOK_TYPE_"

b) préfixé par la valeur "CMC_IT_"

c) préfixé par la valeur "CMC_OBJECT_TYPE_"

Tableau 6/X.446 – Résumé de la propriété liste de distribution CMC

Liste de distribution					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
adresse	string	toute adresse valide	O	oui	néant
commentaire	string	toute chaîne de caractères valide	O	non	néant
instant de dernière modification	iso_date_time	toute date et heure ISO 8601	O	oui	néant
nom	string	toute chaîne de caractères valide	M	non	chaîne de caractères vide
classe d'objets	enum	DISTRIBUTION LIST[a)]	M	oui	non disponible
parent	object_handle	tout descripteur opaque d'objet valide	M, en cas d'imbrication	non	néant
partagé	boolean	CMC_TRUE, CMC_FALSE	O	non	CMC_FALSE
a) préfixé par la valeur "CMC_OBJECT_TYPE_"					

Tableau 7/X.446 – Résumé de la propriété message CMC

Message					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
identificateur d'application	string	toute chaîne de caractères valide	O	non	néant
statut de message de l'application	flags	projet	O	non	néant
action automatique	flags	CMC_AA_ DELETE	O	non	néant
instant de remise différée	iso_date_time	toute date et heure ISO 8601	O	non	néant
statut de message en entrée	flags	NEW, READ, CHANGED[a)]	O	oui	néant
identificateur	string	toute chaîne de caractères valide	M	oui	néant
en réponse à	string	toute chaîne de caractères valide	O	non	néant
comptage d'article	uint32	jusqu'à un maximum défini par la mise en œuvre	M	oui	néant
diagnostic de non-acquittement	string	toute chaîne de caractères valide	O	non	néant

Tableau 7/X.446 – Résumé de la propriété message CMC *(fin)*

Message					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
motif de non-acquittement	string	toute chaîne de caractères valide	M, si le message est du type avec acquittement	non	néant
classe d'objets	object_class	MESSAGE[b)]	M	oui	non disponible
statut de message en sortie	flags	DELETED, SUBMITTED, SENT[a)]	O	oui	néant
priorité	enum	URGENT NORMAL LOW[c)]	O	non	normal
acquittement demandé	boolean	CMC_TRUE CMC_FALSE	O	non	CMC_FALSE
type d'acquittement	enum	RN, NRN[d)]	O	non	néant
compte rendu demandé	enum	DR, NDR, BOTH, NONE[e)]	O	non	néant
rôle	enum	ORIGINAL RETURNED FORWARDED REPLIED OBSOLETED RESENT[f)]	O	non	néant
sensibilité	enum	PERSONAL PRIVATE CONFIDENTIAL NONE[g)]	O	non	aucun
taille	uint32	toute valeur d'octet valide	O	non	néant
sujet	string	toute chaîne de caractères valide	O	non	néant
instant de réception	iso_date_time	toute date et heure ISO 8601	M	oui	néant
instant d'émission	iso_date_time	toute date et heure ISO 8601	M	oui	néant
type	enum	IPM, REPORT[h)]	M	non	IPM

a) préfixé par la valeur "CMC_MESSAGE_STATUS_"

b) préfixé par la valeur "CMC_OBJECT_TYPE_"

c) préfixé par la valeur "CMC_PRIORITY_"

d) préfixé par la valeur "CMC_RECEIPT_"

e) préfixé par la valeur "CMC_REPORT_"

f) préfixé par la valeur "CMC_MESSAGE_ROLE_"

g) préfixé par la valeur "CMC_MESSAGE_SENSITIVITY_"

h) préfixé par la valeur "CMC_MCT_"

Tableau 8/X.446 – Résumé de la propriété conteneur de messages CMC

Conteneur de messages					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
action automatique	flags	CMC-AA_Delete	O	oui	non-positionné
enfant autorisé	boolean	CMC_TRUE CMC_FALSE	O	non	CMC_FALSE
commentaire	string	toute chaîne de caractères valide	O	non	néant
emplacement	enum	LOCAL, SERVER, UNKOWN[a)]	O	non	néant
nom	string	toute chaîne de caractères valide	O	non	néant
classe d'objets	enum	MESSAGE CONTAINER[b)]	M	oui	non disponible
parent	object_handle	tout descripteur opaque d'objet valide	M, en cas d'imbrication	non	néant
nom du serveur	string	toute chaîne de caractères valide	O	non	néant
partagé	boolean	CMC_TRUE CMC_FALSE	O	non	CMC_FALSE
type	enum	DELETED DRAFTS INBOX OUTBOX SENT[c)]	O	oui	néant

a) préfixé par la valeur "CMC_MESSAGE_CONTAINER_"

b) préfixé par la valeur "CMC_OBJECT_TYPE_"

c) préfixé par la valeur "CMC_MCT_"

Tableau 9/X.446 – Résumé de la propriété information par destinataire CMC

Information par destinataire					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
commentaire	string	toute chaîne de caractères valide	O	non	néant
instant de remise	iso_date_time	toute date et heure ISO 8601	O	non	néant
diagnostic	string	toute chaîne de caractères valide	O	non	néant
classe d'objets	enum	PER RECIPIENT INFORMATION[a)]	M	oui	non disponible
motif	string	toute chaîne de caractères valide	O	non	néant
adresse du destinataire	string	toute chaîne de caractères valide	M	oui	non disponible
nom du destinataire	string	toute chaîne de caractères valide	M	non	non disponible
type	enum	DR, NDR, UNKNOWN[b)]	M	oui	non disponible

a) préfixé par la valeur "CMC_OBJECT_TYPE_"

b) préfixé par la valeur "CMC_PRI_"

Tableau 10/X.446 – Résumé de la propriété conteneur de profil CMC

Conteneur de profil					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
action automatique	flags	CMC-AA_Delete	O	oui	non-positionné
jeu de caractères	array_of_guid	une interface GUID ou plus pour tout jeu de caractères	M	oui	non disponible
commentaire	string	toute chaîne de caractères valide	O	non	néant
conformité	enum	SIMPLE_CMC, FULL_CMC[a)]	M	oui	non disponible
service par défaut	string	toute chaîne de caractères valide	M	oui	non disponible
utilisateur par défaut	string	toute chaîne de caractères valide	M	oui	non disponible
fin de ligne	enum	CRLF, LF, CR[b)]	M	oui	non disponible
classe d'objets	enum	PROFILE_ CONTAINER[c)]	M	Oui	non disponible

Tableau 10/X.446 – Résumé de la propriété conteneur de profil CMC *(fin)*

Conteneur de profil					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
extensions d'objet prises en charge	array_of_guid	une interface GUID ou plus pour des objets CMC	M	oui	non disponible
objets pris en charge	array_of_guid	une interface GUID ou plus pour des objets CMC	M	oui	non disponible
propriétés prises en charge	array_of_guid	une interface GUID ou plus pour des propriétés CMC	M	oui	non disponible
extensions de propriétés prises en charge	array_of_guid	une interface GUID ou plus pour des propriétés CMC	M	oui	non disponible
mot de passe exigé	enum	NO, OPT, YES[d)]	M	oui	non disponible
service exigé	enum	NO, OPT, YES[d)]	M	oui	non disponible
utilisateur exigé	enum	NO, OPT, YES[d)]	M	oui	non disponible
prise en charge des chaînes de caractères avec comptage	boolean	CMC_TRUE CMC_FALSE	M	oui	non disponible
prise en charge de l'absence de marquage en lecture	boolean	CMC_TRUE CMC_FALSE	M	oui	non disponible
interface utilisateur disponible	boolean	CMC_TRUE CMC_FALSE	M	oui	non disponible
utilisateurs	array_string	noms de destinataires	O	oui	non disponible
version de la mise en œuvre	uint16	100 ou 200	M	oui	non disponible
version de la spécification	uint16	100 ou 200	M	oui	non disponible

a) préfixé par la valeur "CMC_CONF_"

b) préfixé par la valeur "CMC_LINE_TERM_"

c) préfixé par la valeur "CMC_OBJECT_TYPE_"

d) préfixé par la valeur "CMC_REQUIRED_"

Tableau 11/X.446 – Résumé de la propriété destinataire CMC

Destinataire					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
adresse	string	toute chaîne de caractères valide	M	non	néant
renvoi de contenu exigé	boolean	CMC_TRUE CMC_FALSE	O	non	néant
nom	string	toute chaîne de caractères valide	O	non	néant
classe d'objets	enum	RECIPIENT[a)]	M	oui	non disponible
acquittement demandé	enum	RN, NRN, BOTH, NONE[b)]	O	non	néant
compte rendu demandé	enum	DR, NDR, BOTH, NONE[c)]	O	non	néant
fanion de responsabilité	boolean	CMC_TRUE CMC_FALSE	M	non	CMC_TRUE
rôle	enum	TO, CC, BCC, ORIGINATOR, AUTHORIZING_USER, REPLY_TO, FORWARDED, ACTUAL, INTENDED[d)]	O	non	néant
type	enum	UNKNOW, INDIVIDUAL, GROUP[e)]	M	non	INDIVIDUAL

a) préfixé par la valeur "CMC_OBJECT_TYPE_"

b) préfixé par la valeur "CMC_RECEIPT_"

c) préfixé par la valeur "CMC_REPORT_"

d) préfixé par la valeur "CMC_RECIPIENT_ROLE_"

e) préfixé par la valeur "CMC_RCT_"

Tableau 12/X.446 – Résumé de la propriété compte rendu CMC

Compte-rendu					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
identificateur d'application	string	toute chaîne de caractères valide	O	non	néant
identificateur	guid	toute chaîne de caractères ISO 9070 valide	M	oui	non disponible
comptage d'article	uint32	tout entier valide	M	oui	non disponible
identificateur du système de messagerie	string	toute chaîne de caractères valide	O	non	néant
classe d'objets	enum	REPORT[a)]	M	oui	non disponible
lecture	boolean	CMC_TRUE CMC_FALSE	O	non	néant
taille	uint32	taille du compte rendu en octets	O	non	néant
sujet	string	toute chaîne de caractères valide	M	non	néant
identificateur de message sujet	string	toute chaîne de caractères valide	O	non	néant
instant de réception	iso_date_time	toute date et heure ISO 8601	M	oui	néant
instant d'émission	iso_date_time	toute date et heure ISO 8601	M	oui	néant
non émis	boolean	CMC_TRUE CMC_FALSE	O	non	néant

a) préfixé par la valeur "CMC_OBJECT_TYPE_"

Tableau 13/X.446 – Résumé de la propriété conteneur racine

Conteneur racine					
Nom de la propriété	**Type (CMC_pv_)**	**Valeurs possibles**	**Classification**	**En lecture seulement**	**Valeur par défaut**
enfant autorisé	boolean	CMC_TRUE CMC_FALSE	O	non	CMC_FALSE
commentaire	string	toute chaîne de caractères valide	O	non	néant
emplacement	enum	LOCAL SERVER UNKNOWN[a)]	O	non	néant
nom	string	toute chaîne de caractères valide	O	non	néant
classe d'objets	enum	ROOT CONTAINER[b)]	M	oui	non disponible
partagé	boolean	CMC_TRUE CMC_FALSE	O	non	CMC_FALSE

a) préfixé par la valeur "CMC_ROOT_CONTAINER_LOCATION_"

b) préfixé par la valeur "CMC_OBJECT_TYPE_"

Les pages de manuel correspondant à ces propriétés sont données ci-après.

5.1 Propriétés de l'objet carnet d'adresses

Un carnet d'adresses est un objet conteneur constitué d'adresses d'articles et pouvant contenir d'autres carnets d'adresses. La prise en charge des carnets d'adresses n'est pas obligatoire. Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés de carnet d'adresses.

5.1.1 **Enfant autorisé** (*Child allowed*)

NOM

Carnet d'adresses enfant autorisé

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Propriété autorisant ou interdisant l'existence d'un enfant du carnet d'adresses.

La valeur par défaut de cette propriété est CMC_FALSE.

Cette propriété est du type **CMC_pv_boolean**.

5.1.2 **Commentaire** (*Comment*)

NOM

Commentaire de carnet d'adresses

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété fournit un commentaire décrivant le carnet d'adresses.

Cette propriété est du type **CMC_pv_string**.

5.1.3 **Emplacement** (*Location*)

NOM

Emplacement du carnet d'adresses

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique l'emplacement du carnet d'adresses.

Cette propriété peut prendre les valeurs suivantes:

CMC_ADDRESS_BOOK_LOCATION_LOCAL
CMC_ADDRESS_BOOK_LOCATION_SERVER
CMC_ADDRESS_BOOK_LOCATION_UNKNOWN

CMC_ADDRESS_BOOK_LOCATION_LOCAL – Spécifie que l'emplacement du carnet d'adresses est local et non sur le serveur de messagerie.

CMC_ADDRESS_BOOK_LOCATION_SERVER – Spécifie que l'emplacement du carnet d'adresses est sur le serveur de messagerie.

CMC_ADDRESS_BOOK_LOCATION_UNKNOWN – Spécifie que l'emplacement du carnet d'adresses est inconnu. Ceci est la valeur par défaut.

Cette propriété est du type **CMC_pv_enum**.

5.1.4 **Nom** (*Name*)

NOM

Nom de carnet d'adresses

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le nom du carnet d'adresses.

Cette propriété est du type **CMC_pv_string**.

5.1.5 **Classe d'objets** (*Object class*)

NOM

Classe d'objets carnet d'adresses

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique que l'objet fait partie de la classe carnet d'adresses.

Cette propriété est créée par la fonction **cmc_open_object_handle()**.

La seule valeur valide de cette propriété est CMC_PT_OBJECT_CLASS_ADDRESS_BOOK spécifiant que l'objet fait partie de la classe carnet d'adresses.

Cette propriété est du type **CMC_pv_enum**.

5.1.6 **Parent** (*Parent*)

NOM

Parent du carnet d'adresses

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le parent du carnet d'adresses. Cette propriété indique le carnet d'adresses parent si la mise en œuvre prend en charge l'imbrication de carnets d'adresses. Elle n'est pas présente si le carnet d'adresses se trouve au niveau sommital; elle est obligatoire dans le cas contraire.

Cette propriété est du type **CMC_pv_object_handle**.

5.1.7 **Nom de serveur** (*Server name*)

NOM

Nom du serveur de carnet d'adresses

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le nom du serveur sur lequel se trouve le carnet d'adresses.

Cette propriété est du type **CMC_pv_string**.

5.1.8 **Partagé** (*Shared*)

NOM

Carnet d'adresses partagé

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique si plus d'un utilisateur peut accéder au carnet d'adresses.

La valeur par défaut de cette propriété est CMC_FALSE, si elle est prise en charge.

Cette propriété est du type **CMC_pv_boolean**.

5.1.9 **Type** (*Type*)

NOM

Type de carnet d'adresses

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le type de carnet d'adresses.

Cette propriété peut prendre les valeurs suivantes:

CMC_ADDRESS_BOOK_TYPE_GLOBAL
CMC_ADDRESS_BOOK_TYPE_PERSONAL

CMC_ADDRESS_BOOK_TYPE_GLOBAL – Spécifie que le carnet d'adresses est d'un sous-type global ou valable pour l'entreprise. Un carnet d'adresses global n'est pas nécessairement partagé.

CMC_ADDRESS_BOOK_TYPE_PERSONAL – Spécifie que le carnet d'adresses est de type personnel, créé et géré localement.

La valeur par défaut de cette propriété est CMC_ADDRESS_BOOK_TYPE_PERSONAL.

Cette propriété est du type **CMC_pv_enum.**

5.2 Propriétés de l'objet article de contenu

Un article de contenu est, dans le contexte de la présente Recommandation, un objet associé au contenu d'un message. Il est utilisé pour représenter des attachements et des notes, aucune distinction n'étant faite entre les deux au niveau de l'interface de programmation. Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés de l'objet attachement.

5.2.1 **Jeu de caractères** (*Character set*)

NOM

Article de contenu jeu de caractères

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le jeu de caractères de l'information de contenu dans l'article de contenu. En l'absence de cette propriété, le jeu de caractères par défaut de l'information de contenu dans de l'article de contenu est le même que celui du contexte de la session.

La valeur de la propriété est une chaîne de caractères représentant l'identificateur public formel du jeu de caractères. L'identificateur public formel peut prendre l'une des valeurs suivantes:

```
#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"
```

Des mises en œuvre peuvent fournir d'autres jeux de caractères.

Cette propriété est du type **CMC_pv_guid**.

5.2.2 **Information de contenu** (*Content information*)

NOM

Information de contenu de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété renferme le contenu d'un article de contenu.

Cette propriété est du type **CMC_pv_opaque_data**.

5.2.3 **Type de contenu** (*Content type*)

NOM

Type de contenu de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le type de contenu de l'article de contenu. Une valeur NULL désigne un type d'article de contenu non défini.

Les valeurs d'identificateur GUID sont valides pour la propriété type des objets article de contenu.

```
#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 – Spécifie un contenu de texte simple ou non formaté.

CMC_CT_GIF_IMAGE – Spécifie un contenu de données graphique dans le format d'image graphique (GIF, *graphics image format*) utilisé par la norme MIME et le protocole WWW.

CMC_CT_JPEG_IMAGE – Spécifie un contenu de données graphique dans le format défini par la norme ISO du groupe joint de codage d'image (JPEG, *joint picture encoding group*) utilisé par la norme MIME et le protocole WWW.

CMC_CT_BASIC_AUDIO – Spécifie un contenu de données audio codé sous forme de signal audio utilisant la loi µ à 8 bits du RNIS ou la modulation MIC définie par la Recommandation G.711 avec un taux d'échantillonnage de 8000 Hz et un canal unique.

CMC_CT_MPEG_VIDEO – Spécifie un contenu vidéo dans le format défini par l'ISO 11172 du groupe de codage d'images animées (MPEG, *motion picture encoding group*), utilisé par la norme MIME et le protocole WWW.

CMC_CT_MESSAGE – Spécifie que le contenu est un message encapsulé.

CMC_CT_PARTIAL_MESSAGE – Spécifie que le contenu est une partie d'un autre message. Ce type de contenu permet de livrer un message long sous la forme de plusieurs parties séparées de manière à en faciliter la réception.

CMC_CT_EXTERNAL_MESSAGE – Spécifie que le contenu est externe au message. La propriété information de contenu contient un texte faisant référence à l'information de contenu externe.

CMC_CT_APPLICATION_OCTET_STREAM – Spécifie que le contenu est un flux d'octets dépendant de l'application.

CMC_CT_APPLICATION_POSTSCRIPT – Spécifie que le contenu est un programme en langage PostScript, défini par Adobe Systems Inc.

CMC_CT_ALTERNATIVE_MULTIPART – Spécifie que le contenu est sous la forme d'une des variantes de contenu d'une autre note ou article de contenu figurant dans l'objet message.

CMC_CT_DIGEST_MULTIPART – Spécifie que le contenu est l'un des messages d'un groupe en rapport avec l'objet message. Les messages peuvent constituer une séquence de discussion organisée sous la forme d'une filière de messages, comme c'est le cas dans les systèmes de bulletins d'information (BBS, *bulletin board systems*).

CMC_CT_MIXED_MULTIPART – Spécifie que le contenu fait partie d'une séquence ordonnée de messages au sein de l'objet message.

CMC_CT_PARALLEL_MULTIPART – Spécifie que le contenu fait partie, dans un ordre quelconque, d'une séquence de messages au sein de l'objet message.

CMC_CT_OLE – Spécifie que le type de l'élément continu est un objet protocole OLE (OLE, *object linking and embedding*) (liaison et incorporation d'objets).

CMC_CT_X400_G3_FAX – Spécifie que le contenu représente des images de télécopie du groupe 3 sous forme d'une chaîne binaire. Chaque composante de données G3 code une seule page de données comme spécifié par les Recommandations T.4 et T.30.

CMC_CT_X400_G4_FAX – Spécifie que le contenu représente une forme finale de document sous une forme pouvant être traitée par des terminaux de télécopie de la classe 1 du groupe 4.

CMC_CT_X400_ENCRYPTED – Spécifie que le contenu est une chaîne binaire codée conformément aux règles de codage de base de la Recommandation X.209.

CMC_CT_X400_NATIONALLY_DEFINED – Spécifie que le contenu est un objet d'information dont la sémantique et la syntaxe abstraite sont définies d'une manière nationale par un pays dont l'identité a fait l'objet d'un accord bilatéral entre l'expéditeur du message et la totalité de ses destinataires potentiels.

CMC_CT_X400_FILE_TRANSFER – Spécifie que l'information de contenu se constitue d'un volume de données relativement important. La propriété contenu d'information renferme le texte faisant référence à la sémantique et à la syntaxe abstraite qui sont indiquées par un identificateur d'objet.

CMC_CT_X400_VOICE – Spécifie que le contenu est un message vocal numérisé, dont le codage n'est pas défini à l'heure actuelle dans la version 1988 des Recommandations X.420.

CMC_CT_X400_VIDEOTEX – Spécifie que le contenu représente des données vidéotex dont la syntaxe est définie par les Recommandations T.100 et T.101.

CMC_CT_X400_MIXED_MODE – Spécifie que le contenu représente un document de forme finale d'un type pouvant être traité par des terminaux en mode combiné télétex et par des terminaux de télécopie des classes 2 et 3 du groupe 4.

CMC_CT_X400_PRIVATELY_DEFINED_6937 – Spécifie que le contenu est défini d'une manière privée. Le contenu est codé conformément aux jeux de caractères et aux règles de codage de l'ISO 6937.

CMC_CT_X400_EXTERNAL_TRACE – Spécifie que le contenu est une information de trace externe X.400, utilisée dans un but de diagnostic.

CMC_CT_X400_INTERNAL_TRACE – Spécifie que le contenu est une information de trace interne X.400, utilisée dans un but de diagnostic.

CMC_CT_SMTP_SESSION_TRANSCRIPT – Spécifie que le contenu est une information de transcription d'une session protocole SMTP, utilisée dans un but de diagnostic.

Cette propriété est du type **CMC_pv_guid**.

5.2.4 **Instant de création** (*Create time*)

NOM

Instant de création de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique la date et l'heure de création de l'article de contenu.

Cette propriété est du type **CMC_pv_iso_date_time**.

5.2.5 **Type de codage** (*Encoding type*)

NOM

Type de codage de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le type de codage de l'article de contenu.

La valeur par défaut de cette propriété est CMC_ET_7_BIT.

Les valeurs suivantes sont valides pour la propriété type de codage de l'objet article de contenu:

```
#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 – Spécifie qu'aucun codage n'a été effectué sur l'information de contenu, et qu'en outre l'information de contenu est constituée d'octets de données à 7 bits.

CMC_ET_BASE64 – Spécifie que l'information de contenu a été codée en utilisant la forme Base 64 de la Norme RFC 1521/MIME pour une suite quelconque d'octets.

CMC_ET_BINARY – Spécifie qu'aucun codage n'a été effectué sur l'information de contenu, et qu'en outre l'information de contenu est constituée d'une quantité relativement importante de données dont les octets peuvent avoir le bit d'ordre le plus élevé positionné.

CMC_ET_8_BIT – Spécifie qu'aucun codage n'a été effectué sur l'information de contenu, et qu'en outre, l'information de contenu se trouve sous la forme de lignes relativement courtes constituées d'octets, dont le bit d'ordre le plus élevé est positionné.

CMC_ET_QUOTED_PRINTABLE – Spécifie que l'information de contenu a été codée en utilisant la forme de la Norme RFC 1521/MIME pour des caractères appartenant au jeu de caractères ASCII et pouvant en grande partie être imprimés, de sorte qu'il est peu probable que les octets résultants soient modifiés par le transport de messagerie.

Cette propriété est du type **CMC_pv_guid**.

5.2.6 **Répertoire de fichier** (*File directory*)

NOM

Répertoire de fichier de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le répertoire dans lequel se trouvait l'article de contenu au moment où il a été ajouté au message. Lorsque la propriété type de continu est égale à CMC_CT_EXTERNAL_MESSAGE, cette propriété indique le nom du serveur ainsi que le nom du répertoire sur un serveur distant. Le format de cette chaîne de caractères est défini par la mise en œuvre.

Cette propriété est du type **CMC_pv_string**.

5.2.7 **Nom de fichier** (*File name*)

NOM

Nom de fichier de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le nom de fichier de l'article de contenu au moment où il a été ajouté au message. Cette propriété indique le nom de fichier sur un serveur distant lorsque la propriété type de contenu est égale à CMC_CT_EXTERNAL_MESSAGE. Le format de cette chaîne de caractères est défini par la mise en œuvre.

Cette propriété est du type **CMC_pv_string**.

5.2.8 **Numéro d'article** (*Item number*)

NOM

Numéro d'article de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le numéro de séquence de l'élément au sein du conteneur parent, d'un objet message ou d'un autre article de contenu. Cette propriété est obligatoire.

Deux éléments ne peuvent avoir la même valeur de numéro d'élément dans un conteneur parent donné.

Cette propriété est du type **CMC_pv_uint32**.

5.2.9 **Type d'article** (*Item type*)

NOM

Type d'article de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le type de l'article de contenu. Cette propriété peut prendre les valeurs suivantes:

CMC_IT_NOTE
CMC_IT_ATTACHMENT
CMC_IT_ANNOTATION

CMC_IT_NOTE – Spécifie un type note.

CMC_IT_ATTACHMENT – Spécifie un type attachement.

CMC_IT_ANNOTATION – Spécifie une annotation faite sur un autre objet article de contenu.

Cette propriété est du type **CMC_pv_enum**.

5.2.10 **Dernière modification** (*Last modified*)

NOM

Dernière modification de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique la date et l'heure de la dernière modification du fichier à partir duquel a été créé l'article de contenu.

Cette propriété est du type **CMC_pv_iso_date_time**.

5.2.11 **Classe d'objets** (*Object class*)

NOM

Classe d'objets article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété définit la classe de l'objet sous la forme d'un article de contenu. Cette propriété est créée par la fonction **cmc_open_object_handle()**.

La seule valeur valide de cette propriété est CMC_PT_OBJECT_CLASS_CONTENT_ITEM spécifiant que l'objet fait partie de la classe article de contenu.

Cette propriété est du type **CMC_pv_enum**.

5.2.12 **Position de rendu** (*Render position*)

NOM

Article de contenu position de rendu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique la position de l'article de contenu au sein de son conteneur.

Cette propriété est du type **CMC_pv_uint32**.

5.2.13 **Taille** (*Size*)

NOM

Taille de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique la taille de l'article de contenu.

Cette propriété est du type **CMC_pv_uint32**.

5.2.14 **Titre** (*Title*)

NOM

Titre de l'article de contenu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété donne la description complète de l'article de contenu, telle que "Rapport financier trimestriel".

Cette propriété est du type **CMC_pv_string**.

5.3 Propriétés de l'objet liste de distribution

Les listes de distribution identifient des groupes d'utilisateurs. Une liste de distribution contient des objets destinataire. Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés d'une liste de distribution.

5.3.1 **Adresse** (*Address*)

NOM

Adresse de la liste de distribution

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété fournit l'adresse d'une liste de distribution à utiliser pour le courrier destiné à la liste. La valeur de l'adresse coïncide souvent avec le nom de la liste de distribution. La propriété liste de distribution est optionnelle.

Cette propriété est générée par le système de messagerie.

Cette propriété est du type **CMC_pv_string**.

5.3.2 **Commentaire** (*Comment*)

NOM

Commentaire de liste de distribution

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété fournit un commentaire décrivant la liste de distribution.

Cette propriété est du type **CMC_pv_string**.

5.3.3 **Instant de dernière modification** (*Last modification time*)

NOM

Instant de dernière modification de la liste de distribution

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique la date et l'heure de la dernière mise à jour de la liste de distribution.

Cette propriété est du type **CMC_pv_iso_date_time**.

5.3.4 **Nom** (*Name*)

NOM

Nom de la liste de distribution

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété donne le nom de la liste de distribution. Elle constitue une propriété obligatoire des objets liste de distribution.

La chaîne de caractères peut être générée par le système de messagerie à partir d'un annuaire ou par l'utilisateur.

Cette propriété est du type **CMC_pv_string**.

5.3.5 Classe d'objets (*Object class*)

NOM

Classe d'objets liste de distribution

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique que l'objet appartient à la classe liste de distribution. Elle est créée par la fonction **cmc_open_object_handle()**.

La seule valeur valide de cette propriété est CMC_PT_OBJECT_CLASS_DISTRIBUTION_LIST spécifiant que l'objet fait partie de la classe liste de distribution.

Cette propriété est du type **CMC_pv_enum**.

5.3.6 Parent (*Parent*)

NOM

Parent de liste de distribution

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le parent de la liste de distribution, si la mise en œuvre prend en charge l'imbrication des listes de distribution. Elle n'est pas présente si la liste de distribution constitue le niveau sommital, mais est obligatoire dans le cas contraire. Cette propriété est nulle pour le parent.

Cette propriété est du type **CMC_pv_object_handle**.

5.3.7 Partagé (*Shared*)

NOM

Liste de distribution partagée

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique si plus d'un utilisateur peut accéder à la liste de distribution.

La valeur par défaut de cette propriété, si elle est prise en charge, est CMC_FALSE.

Cette propriété est du type **CMC_pv_boolean**.

5.4 Propriétés de l'objet message

L'objet message est une collection de propriétés d'objet propres à un message. Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés de l'objet message.

5.4.1 **Identificateur d'application** (*Application Id*)

NOM

Identificateur d'application du message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété définit un identificateur global non-ambigu pour le message. Elle est positionnée par l'application.

Cette propriété est du type **CMC_pv_string**.

5.4.2 **Statut de message de l'application** (*Application message status*)

NOM

Statut de message de l'application de messagerie

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le statut du message spécifié par l'appelant. Elle peut être utilisée par ce dernier afin d'indiquer si le message est un projet ou un message terminé. Aucune sémantique n'est impliquée au sujet du système de messagerie, cette propriété persiste toutefois d'une session à la suivante.

Cette propriété peut prendre les valeurs suivantes:

CMC_MESSAGE_STATUS_DRAFT

CMC_MESSAGE_STATUS_DRAFT – Spécifie que le message est en mode projet.

Cette propriété est du type **CMC_pv_flags**.

5.4.3 **Action automatique** (*Auto-action*)

NOM

Action automatique de message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique une action automatique ou la suppression du message après son émission. La prise en charge de cette propriété est optionnelle pour des mises en œuvre se conformant à la présente Recommandation. La propriété est créée, dans le cas de création de nouveaux messages, par la fonction **cmc_add_properties()**. Pour ces mêmes messages, elle peut également être modifiée au moyen de la fonction **cmc_add_properties()** ou supprimée par la fonction **cmc_delete_properties()**. Cette propriété de l'objet message se substituera à la préférence positionnée dans l'objet conteneur de profil.

Cette propriété peut posséder la valeur suivante:

CMC_AA_DELETE

positionnée: le message spécifié peut être supprimé par le système de messagerie sous-jacent une fois qu'il a été déposé avec succès pour un transfert;

non-positionné: le message spécifié est placé dans le dossier de messages émis, si ce dernier existe. Il est supprimé dans le cas contraire.

Cette propriété est du type **CMC_pv_flags**.

5.4.4 **Instant de remise différée** (*Deferred delivery time*)

NOM

Instant de remise différée du message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique la date et l'heure UTC (UTC, *coordinate universal time*) (temps coordonné universel) avant lesquelles le message ne doit pas être remis aux destinataires.

Cette propriété est du type **CMC_pv_iso_date_time**.

5.4.5 **Identificateur** (*Id*)

NOM

Identificateur de message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété définit un identificateur global non-ambigu pour le message. Cette propriété est positionnée par la fonction **cmc_send_message_object()**, elle est définie par le service de messagerie (au moment de dépôt) et elle est non-ambiguë au sein du domaine.

L'identificateur peut être ajouté ou mis à jour par l'appelant dans des applications de passerelle.

Cette propriété est du type **CMC_pv_string**.

5.4.6 **Statut de message en entrée** (*In message status*)

NOM

Statut de message en entrée du message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique au service de messagerie spécifié le statut du message, en entrée ou en sortie. Elle est utilisée par le service de messagerie sous-jacent pour mémoriser une information de statut concernant le mode de réception et de traitement du message. Le service de messagerie peut, par exemple, indiquer qu'un message vient d'être reçu dans une boîte aux lettres en entrée, a été lu ou a fait l'objet d'un acquittement.

La prise en charge de cette propriété est optionnelle pour des mises en œuvre se conformant à la présente Recommandation. La propriété est accessible en lecture seulement. Elle est créée et modifiée par le service de messagerie sous-jacent. Elle ne peut pas être supprimée par l'utilisateur.

Cette propriété peut prendre les valeurs suivantes:

CMC_MESSAGE_STATUS_NEW
CMC_MESSAGE_STATUS_READ
CMC_MESSAGE_STATUS_CHANGED

CMC_MESSAGE_STATUS_NEW – Spécifie que le message vient d'être reçu par le service de messagerie sous-jacent. Ce fanion sera remis à zéro lors de la fermeture de la session, lors de l'accès à l'objet message ou lors de la fermeture du conteneur de messages.

CMC_MESSAGE_STATUS_READ – Spécifie que le message a été lu. Ce fanion de statut est positionné lorsque l'un des objets article de contenu subordonnés au message a été lu au moyen de la fonction **cmc_read_properties()**.

CMC_MESSAGE_STATUS_CHANGED – Spécifie que le contenu du message a été modifié par rapport à la forme qu'il avait au moment de la réception initiale. Ce fanion de statut est positionné lorsqu'une propriété appartenant au message est ajoutée ou modifiée au moyen de la fonction **cmc_add_properties()**, ou supprimée au moyen de la fonction **cmc_delete_properties()**.

Cette propriété est du type **CMC_pv_flags**.

5.4.7 **En réponse à** (*In reply to*)

NOM

Message en réponse à

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique la correspondance précédente à laquelle répond le message.

La valeur de la propriété peut être une référence textuelle ou une approximation textuelle de l'identificateur de la correspondance précédente.

Cette propriété est du type **CMC_pv_string**.

5.4.8 **Comptage d'articles** (*Item count*)

NOM

Comptage d'articles de message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le nombre d'articles de contenu de niveau sommital contenus dans un message. Le comptage n'inclut pas les articles de contenu imbriqués dans d'autres articles de contenu, messages ou comptes rendus. Cette propriété est positionnée par la mise en œuvre.

Cette propriété est du type **CMC_pv_uint32**.

5.4.9 **Diagnostic de non-acquittement** (*NRN diagnostic*)

NOM

Diagnostic de non-acquittement de message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété donne les détails du diagnostic concernant le motif de l'avis de non-acquittement. Il s'agit de détails supplémentaires concernant le motif de non-acquittement. Cette propriété ne concerne que les messages du type CMC_MT_RECEIPT

Cette propriété est du type **CMC_pv_string**.

5.4.10 **Motif de non-acquittement** (*NRN reason*)

NOM

Motif de non-acquittement de message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété explique pourquoi le message n'a pas été acquitté. Elle ne concerne que les messages de type CMC_MT_RECEIPT avec la propriété de type d'avis CMC_RECEIPT_NRN.

Cette propriété est du type **CMC_pv_string**.

5.4.11 Classe d'objets (*Object class*)

NOM

Classe d'objets message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique que l'objet fait partie de la classe message.

Cette propriété est créée par la fonction **cmc_open_object_handle()**.

La seule valeur valide de la propriété est CMC_PT_OBJECT_CLASS_MESSAGE spécifiant que l'objet fait partie de la classe message.

Cette propriété est du type **CMC_pv_enum**.

5.4.12 Statut de message en sortie (*Out message status*)

NOM

Statut de message du message en sortie

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique au service de messagerie spécifié le statut du message: en sortie ou suppression. Elle est utilisée par le service de messagerie sous-jacent pour mémoriser une information de statut concernant le mode de suppression du message. Le service de messagerie peut, par exemple, spécifier qu'un message a été marqué pour une suppression, a été déposé pour un transfert ou est en train d'être émis.

La prise en charge de cette propriété est optionnelle pour des mises en œuvre se conformant à la présente Recommandation. La propriété est accessible en lecture seulement. Elle est créée et modifiée par le service de messagerie sous-jacent. Elle ne peut pas être supprimée par l'utilisateur.

Cette propriété peut prendre les valeurs suivantes:

CMC_MESSAGE_STATUS_DELETED
CMC_MESSAGE_STATUS_SUBMITTED
CMC_MESSAGE_STATUS_SENT

CMC_MESSAGE_STATUS_DELETED – Spécifie que le message est en attente de suppression. Dans certains environnements de mise en œuvre (par exemple, lorsque l'utilisateur est déconnecté), une opération de suppression faite sur un message ne peut pas être effectuée immédiatement. Ce fanion indique que le message a été marqué pour une suppression, même s'il apparaît dans un conteneur.

CMC_MESSAGE_STATUS_SUBMITTED – Spécifie que le message a été déposé en vue d'un transfert au service de messagerie sous-jacent, soit par la fonction **cmc_send_message_object()**, soit par la fonction **cmc_commit_object()** qui confie un objet message à une boîte aux lettres en sortie située dans un conteneur de messages. Dans certains environnements de mise en œuvre, (par exemple, lorsque l'utilisateur est déconnecté), une opération d'émission ne peut pas être effectuée immédiatement pour un message. Ce fanion indique que le message a été marqué pour un dépôt au service de messagerie sous-jacent, même s'il apparaît dans un conteneur de messages.

CMC_MESSAGE_STATUS_SENT – Spécifie que le message est en attente d'émission par le service de messagerie sous-jacent. Il est possible, dans certains environnements de mise en œuvre, que le transfert d'un message par le service de messagerie sous-jacent ne soit pas fait immédiatement. Dans de tels cas, le message peut apparaître dans un conteneur de messages, même s'il a été émis par l'application. Cet état est indiqué par ce fanion.

Cette propriété est du type **CMC_pv_flags**.

5.4.13 **Priorité** (*Priority*)

NOM

Priorité de message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique la priorité du message. Cette priorité peut prendre une valeur par défaut. Elle peut être positionnée lorsque le message est créé.

Cette propriété peut prendre les valeurs suivantes:

CMC_PRIORITY_URGENT
CMC_PRIORITY_NORMAL
CMC_PRIORITY_LOW

CMC_PRIORITY_URGENT – Spécifie que le message possède une priorité urgente.

CMC_PRIORITY_NORMAL – Spécifie que le message possède une priorité normale. Ceci est la valeur par défaut.

CMC_PRIORITY_LOW – Spécifie que le message possède une priorité basse.

Cette propriété est du type **CMC_pv_enum**.

5.4.14 **Acquittement demandé** (*Receipt requested*)

NOM

Acquittement demandé pour le message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique si un acquittement a été demandé pour le message émis.

Cette propriété peut prendre les valeurs suivantes:

CMC_RECEIPT_RN
CMC_RECEIPT_NRN
CMC_RECEIPT_BOTH
CMC_RECEIPT_NONE

CMC_RECEIPT_RN – Demande le renvoi d'un avis d'acquittement si le destinataire a accepté le message en question.

CMC_RECEIPT_NRN – Demande le renvoi d'un avis de non-acquittement si le destinataire n'a pas accepté le message en question.

CMC_RECEIPT_BOTH – Demande le renvoi d'un avis d'acquittement ou de non-acquittement selon que le destinataire a accepté ou non le message en question.

CMC_RECEIPT_NONE – Demande qu'aucun avis ne soit renvoyé, indépendamment de l'acceptation du message en question par le destinataire.

Cette propriété est du type **CMC_pv_enum**.

5.4.15 **Type d'acquittement** (*Receipt type*)

NOM

Type d'acquittement de message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le type d'acquittement renvoyé pour le message en question. Elle est utilisée pour indiquer si le message en question a été accepté ou non par le destinataire concerné.

Cette propriété peut prendre les valeurs suivantes:

CMC_RECEIPT_RN
CMC_RECEIPT_NRN

CMC_RECEIPT_RN – Spécifie un avis d'acquittement.

CMC_RECEIPT_NRN – Spécifie un avis de non-acquittement.

Cette propriété n'est utilisée que si la propriété type de message possède la valeur CMC_MESSAGE_TYPE_RECEIPT.

Cette propriété est du type **CMC_pv_enum**.

5.4.16 **Compte rendu demandé** (*Report requested*)

NOM

Compte rendu demandé pour le message

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_REPORT_REQUESTED                                \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Report Requested//EN"
```

DESCRIPTION

Cette propriété indique le type de compte rendu renvoyé pour le message en question. Elle est utilisée pour indiquer si le message a été remis ou non par les systèmes de transport de messagerie sous-jacents.

Cette propriété peut prendre les valeurs suivantes:

CMC_REPORT_DR
CMC_REPORT_NDR
CMC_REPORT_BOTH
CMC_REPORT_NONE

CMC_REPORT_DR – Spécifie qu'un compte rendu de remise est demandé.

CMC_REPORT_NDR – Spécifie qu'un compte rendu de non-remise est demandé.

CMC_REPORT_BOTH – Spécifie qu'un compte rendu de remise ou de non-remise est demandé, selon le cas.

CMC_REPORT_NONE – Spécifie qu'aucun compte rendu n'est demandé.

Cette propriété est du type **CMC_pv_enum**.

5.4.17 **Rôle** (*Role*)

NOM

Rôle de message

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_ROLE                                            \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Role//EN"
```

DESCRIPTION

Rôle de ce message.

Cette propriété peut prendre les valeurs suivantes:

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 – Spécifie qu'il s'agit du message original.

CMC_MESSAGE_ROLE_RETURNED – Spécifie qu'il s'agit d'un message renvoyé, correspondant au contenu d'un autre message.

CMC_MESSAGE_ROLE_FORWARDED – Spécifie qu'il s'agit d'un message réexpédié, correspondant au contenu d'un autre message

CMC_MESSAGE_ROLE_REPLIED – Spécifie qu'il s'agit d'une réponse à un autre message.

CMC_MESSAGE_ROLE_OBSOLETED – Spécifie qu'il s'agit d'un message périmé.

CMC_MESSAGE_ROLE_RESENT – Spécifie qu'il s'agit de la réémission d'une copie d'un message original.

Cette propriété est du type **CMC_pv_enum**.

5.4.18 **Sensibilité** (*Sensitivity*)

NOM

Sensibilité du message

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_SENSITIVITY          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Sensitivity//EN"
```

DESCRIPTION

Cette propriété indique la sensibilité du message.

Cette propriété peut prendre les valeurs suivantes:

CMC_MESSAGE_SENSITIVITY_PERSONAL
CMC_MESSAGE_SENSITIVITY_PRIVATE
CMC_MESSAGE_SENSITIVITY_CONFIDENTIAL
CMC_MESSAGE_SENSITIVITY_NONE

CMC_MESSAGE_SENSITIVITY_PERSONAL – Spécifie que le message est personnel.

CMC_MESSAGE_SENSITIVITY_PRIVATE – Spécifie que le message est privé.

CMC_MESSAGE_SENSITIVITY_CONFIDENTIAL – Spécifie que le message est confidentiel.

CMC_MESSAGE_SENSITIVITY_NONE – Spécifie que le message est non-sensible.

Cette propriété est du type **CMC_pv_enum**.

5.4.19 **Taille** (*Size*)

NOM

Taille du message

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_SIZE          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Size//EN"
```

DESCRIPTION

Cette propriété indique la taille du message.

Cette propriété est du type **CMC_pv_uint32**.

5.4.20 **Sujet** (*Subject*)

NOM

Sujet du message

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_SUBJECT       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Subject//EN"
```

DESCRIPTION

Cette propriété décrit le sujet du message. Elle possède une valeur par défaut égale à la chaîne de caractères nulle.

Cette propriété est du type **CMC_pv_string**.

5.4.21 **Instant de réception** (*Time received*)

NOM

Instant de réception du message

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_TIME_RECEIVED       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Time Received//EN"
```

DESCRIPTION

Cette propriété indique la date et l'heure de réception du message.

Cette propriété est du type **CMC_pv_iso_date_time**.

5.4.22 **Instant d'émission** (*Time sent*)

NOM

Instant d'émission du message

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_TIME_SENT     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Time Sent//EN"
```

DESCRIPTION

Cette propriété indique la date et l'heure d'émission du message.

Cette propriété est positionnée par le service au moyen de la fonction **cmc_send_message_object()**.

Cette propriété est du type **CMC_pv_iso_date_time**.

5.4.23 **Type** (*Type*)

NOM

Type de message

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique le type du message. La prise en charge de cette propriété est optionnelle pour des mises en œuvre se conformant à la présente Recommandation. La propriété peut être créée par le système de messagerie dans le cas de messages reçus ou existants. Dans le cas de la création de nouveaux messages, elle est créée par la fonction **cmc_add_properties()**. Elle peut être modifiée pour ces mêmes messages par la fonction **cmc_add_properties()** et supprimée par la fonction **cmc_delete_properties()**.

Cette propriété peut prendre les valeurs suivantes:

CMC_MT_IPM
CMC_MT_RECEIPT
CMC_MT_EDI

CMC_MT_DIRECTORY
CMC_MT_DOCMGMT
CMC_MT_WORKFLOW
CMC_MT_CALSCHED

CMC_MT_IPM – Courrier électronique ou message interpersonnel, au sens de la Recommandation X.400.

CMC_MT_RECEIPT – Acquittement de message. Ce type de message est utilisé pour des avis d'acquittement et de non-acquittement. Ce type de message peut également être utile pour d'autres acquittements de message.

Les types de message qui suivent sont réservés pour des utilisations spécifiques. Leurs valeurs correspondent à des travaux en cours au sein de l'association XAPIA et d'autres groupes issus de l'industrie. Ces types de messages peuvent être modifiés dans de futures versions de la présente Recommandation, afin de prendre en compte l'aboutissement de ces travaux.

CMC_MT_EDI – Message de type échange de données informatisé. La présente Recommandation ne précise pas la forme et le format des messages EDI.

CMC_MT_DIRECTORY – Message de type service d'annuaire. Ce type de message permet d'utiliser les services de messagerie à des fins de consultation d'annuaire. La présente Recommandation ne précise pas la forme et le format des messages d'annuaire.

CMC_MT_DOCMGMT – Message de type gestion de documents. Ce type de message fournit les fonctions d'accès et de recherche de services de bibliothèque utilisant le service de messagerie comme mécanisme de transport pour les fonctions de gestion de consultation de documents. La présente Recommandation ne précise pas la forme et le format des messages de gestion de documents.

CMC_MT_WORKFLOW – Message de type gestion de travaux. Ce type de message facilite le traitement automatisé de processus d'exploitation utilisant le service de messagerie comme mécanisme de transport pour les messages de gestion de travaux. La présente Recommandation ne précise pas la forme et le format des messages de gestion de travaux.

CMC_MT_CALSCHED – Type de message d'agenda et d'ordonnancement. Ce type de message permet l'utilisation du service de messagerie comme mécanisme de transport pour les fonctions d'agenda et d'ordonnancement. La présente Recommandation ne précise pas la forme et le format des messages d'agenda et d'ordonnancement. Il existe d'autres spécifications du groupe XAPIA définissant la spécification d'interopérabilité d'agenda et d'ordonnancement.

Cette propriété est du type **CMC_pv_enum**.

5.5 Propriétés de l'objet conteneur de messages

Un objet conteneur de messages est une collection de propriétés de conteneur, d'objets message et éventuellement d'autres conteneurs de messages. Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés de l'objet conteneur de messages.

5.5.1 **Enfant autorisé** (*Child allowed*)

NOM

Enfant autorisé pour le conteneur de messages

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_CONTAINER_CHILD_ALLOWED                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Child Allowed//EN"
```

DESCRIPTION

Propriété autorisant ou interdisant l'existence d'un enfant pour le conteneur de messages.

Cette propriété est du type **CMC_pv_boolean**.

5.5.2 **Commentaire** (*Comment*)

NOM

Commentaire de conteneur de messages

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_CONTAINER_COMMENT              \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Comment//EN"
```

DESCRIPTION

Cette propriété fournit un commentaire décrivant le conteneur de messages.

Cette propriété est du type **CMC_pv_string**.

5.5.3 **Emplacement** (*Location*)

NOM

Emplacement du conteneur de messages

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_CONTAINER_LOCATION            \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Location//EN"
```

DESCRIPTION

Cette propriété indique l'emplacement du conteneur de messages.

Cette propriété peut prendre les valeurs suivantes:

CMC_MESSAGE_CONTAINER_LOCATION_LOCAL
CMC_MESSAGE_CONTAINER_LOCATION_SERVER
CMC_MESSAGE_CONTAINER_LOCATION_UNKNOWN

CMC_MESSAGE_CONTAINER_LOCATION_LOCAL – Spécifie que l'emplacement du conteneur de messages est local et non sur le serveur de messagerie.

CMC_MESSAGE_CONTAINER_LOCATION_SERVER – Spécifie que l'emplacement du conteneur de messages est sur le serveur de messagerie.

CMC_MESSAGE_CONTAINER_LOCATION_UNKNOWN – Spécifie que l'emplacement du conteneur de messages n'est pas connu.

Cette propriété est du type **CMC_pv_enum**.

5.5.4 **Nom** (*Name*)

NOM

Nom du conteneur de messages

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_CONTAINER_NAME      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Name//EN"
```

DESCRIPTION

Cette propriété donne le nom du conteneur de messages.

Cette propriété est du type **CMC_pv_string**.

5.5.5 **Classe d'objets** (*Object class*)

NOM

Classe d'objets conteneur de messages

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique que l'objet fait partie de la classe conteneur de messages.

Cette propriété est créée par la fonction **cmc_open_object_handle()**.

La seule valeur valide de cette propriété est CMC_PT_OBJECT_CLASS_MESSAGE_CONTAINER spécifiant que l'objet fait partie de la classe conteneur de messages.

Cette propriété est du type **CMC_pv_enum**.

5.5.6 **Parent** (*Parent*)

NOM

Parent du conteneur de messages

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_CONTAINER_PARENT                \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Parent//EN"
```

DESCRIPTION

Cette propriété indique le parent du conteneur de messages. Cette propriété indique le conteneur de messages parent, si la mise en œuvre prend en charge l'imbrication de conteneurs de messages. Cette propriété n'est pas présente si le conteneur de messages se trouve au niveau sommital, elle est obligatoire dans le cas contraire.

Cette propriété est du type **CMC_pv_object_handle**.

5.5.7 **Nom de serveur** (*Server name*)

NOM

Nom du serveur du conteneur de messages

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_CONTAINER_SERVER_NAME           \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Server Name//EN"
```

DESCRIPTION

Cette propriété donne le nom du serveur sur lequel se trouve le conteneur de messages.

Cette propriété est du type **CMC_pv_string**.

5.5.8 **Partagé** (*Shared*)

NOM

Conteneur de messages partagé

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_CONTAINER_SHARED     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Shared//EN"
```

DESCRIPTION

Cette propriété indique si plus d'un utilisateur peut accéder à ce conteneur de messages.

Cette propriété est du type **CMC_pv_boolean**.

5.5.9 **Type** (*Type*)

NOM

Type de conteneur de messages

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_MESSAGE_CONTAINER_TYPE       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Message Container Type//EN"
```

DESCRIPTION

Cette propriété indique le type du conteneur de messages.

Cette propriété peut prendre les valeurs suivantes:

CMC_MCT_DELETED
CMC_MCT_DRAFTS
CMC_MCT_FILED
CMC_MCT_INBOX
CMC_MCT_OUTBOX
CMC_MCT_SENT

CMC_MCT_DELETED – Spécifie que le conteneur de messages est destiné à des messages supprimés.

CMC_MCT_DRAFTS – Spécifie que le conteneur de messages est destiné à des messages en projet.

CMC_MCT_FILED – Spécifie que le conteneur de messages est destiné à des messages mis en fichier.

CMC_MCT_INBOX – Spécifie que le conteneur de messages constitue la boîte aux lettres en entrée. Une mise en œuvre peut avoir plus d'une boîte aux lettres en entrée.

CMC_MCT_OUTBOX – Spécifie que le conteneur de messages est destinée à des messages en sortie. Une mise en œuvre doit avoir obligatoirement au moins une boîte aux lettres en sortie. Les objets confiés à la boîte aux lettres ne sont pas modifiables. Ils ne peuvent être que supprimés ou copiés.

CMC_MCT_SENT – Spécifie que le conteneur de messages est destiné à des messages qui ont été émis. La boîte aux lettres de messages émis est optionnelle. Une mise en œuvre en utilisera une au plus.

Cette propriété est du type **CMC_pv_enum**.

5.6 Propriétés de l'objet information par destinataire

Les objets information par destinataire sont des composants d'un compte rendu généré pour indiquer le statut de remise ou de non-remise d'un message. Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés de l'objet information par destinataire.

5.6.1 **Commentaire** (*Comment*)

NOM

Commentaire de l'information par destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PRI_COMMENT                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Comment//EN"
```

DESCRIPTION

Cette propriété fournit une information supplémentaire concernant le statut du message.

Cette propriété est du type **CMC_pv_string**.

5.6.2 **Instant de remise** (*Delivery time*)

NOM

Instant de remise de l'information par destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PRI_DELIVERY_TIME                              \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Delivery Time//EN"
```

DESCRIPTION

Cette propriété indique la date et l'heure à laquelle le message concerné a été remis.

Cette propriété est positionnée par le service au moyen de la fonction **cmc_send_message_object()**.

Cette propriété est obligatoire si le type de l'information par destinataire est CMC_PRI_DR.

Cette propriété est du type **CMC_pv_iso_date_time**.

5.6.3 **Diagnostic** (*Diagnostic*)

NOM

Diagnostic de l'information par destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PRI DIAGNOSTIC                                 \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Diagnostic//EN"
```

DESCRIPTION

Cette propriété fournit l'information détaillée de diagnostic indiquant pourquoi le message concerné n'a pas été remis.

Cette propriété est du type **CMC_pv_string**.

5.6.4 Classe d'objets (*Object class*)

NOM

Classe d'objets information par destinataire

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique que l'objet fait partie de la classe information par destinataire.

Cette propriété est créée par la fonction **cmc_open_object_handle()**.

La seule valeur valide de cette propriété est CMC_PT_OC_PER_RECIPIENT_INFORMATION spécifiant que l'objet fait partie de la classe information par destinataire.

Cette propriété est du type **CMC_pv_enum**.

5.6.5 Motif (*Reason*)

NOM

Motif de l'information par destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PRI_REASON                                      \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Reason//EN"
```

DESCRIPTION

Cette propriété indique pour quel motif l'information par destinataire a été générée.

Cette propriété est obligatoire si le type de l'information par destinataire est CMC_PRI_NDR.

Cette propriété est du type **CMC_pv_string**.

5.6.6 Adresse du destinataire (*Recipient address*)

NOM

Adresse du destinataire de l'information par destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PRI_RECIPIENT_ADDRESS                           \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Recipient Address//EN"
```

DESCRIPTION

Cette propriété donne l'adresse du destinataire prévu pour le message concerné, le type de l'information par destinataire indiquant si le message a été reçu ou non. Ce n'est pas l'utilisateur origine du message qui sera en général le destinataire de ce compte rendu. Le destinataire du compte rendu ne peut pas y répondre.

Cette propriété est du type **CMC_pv_string**.

5.6.7 Nom du destinataire (*Recipient name*)

NOM

Nom du destinataire de l'information par destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PRI_RECIPIENT_NAME                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Recipient Name//EN"
```

DESCRIPTION

Cette propriété donne le nom du destinataire prévu pour le message concerné, le type de l'information par destinataire indiquant si le message a été reçu ou non. Ce n'est pas l'utilisateur origine du message qui sera en général le destinataire de ce compte rendu. Le destinataire du compte rendu ne peut pas y répondre.

Cette propriété est du type **CMC_pv_string**.

5.6.8 **Type** (*Type*)

NOM

Type d'information par destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PRI_TYPE                                              \
    "–//XAPIA/CMC/PROPERTY//NONSGML PRI Type//EN"
```

DESCRIPTION

Cette propriété indique le type de l'information par destinataire.

Cette propriété peut prendre les valeurs suivantes:

CMC_PRI_DR
CMC_PRI_NDR
CMC_PRI_UNKNOWN

CMC_PRI_DR – Spécifie un type d'avis de remise pour l'information par destinataire.

CMC_PRI_NDR – Spécifie un type d'avis de non-remise pour l'information par destinataire.

CMC_PRI_UNKNOWN – Spécifie que le type d'information par destinataire n'était pas spécifié ou ne s'appliquait pas.

Cette propriété est du type **CMC_pv_enum**.

5.7 Propriétés de l'objet conteneur de profil

L'objet conteneur de profil identifie une information spécifique de contexte de session et de configuration. Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés du conteneur de profil.

5.7.1 **Action automatique** (*Auto-Action*)

NOM

Action automatique pour le conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_CONTAINER_AUTO_ACTION      \
    "–//NONSGML Profile Container Auto Action//EN"
```

DESCRIPTION

Cette propriété indique une action automatique sur le message ou la suppression du message après son émission. La prise en charge de cette propriété est optionnelle pour des mises en œuvre se conformant à la présente Recommandation. La propriété est créée par la fonction **cmc_add_properties()** dans le cas de création de nouveaux messages. Pour ces mêmes messages, elle peut également être modifiée au moyen de la fonction **cmc_add_properties()** ou supprimée par la fonction **cmc_delete_properties()**. Cette propriété peut être remplacée, message par message, par la propriété CMC_PT_MESSAGE_AUTO_ACTION de l'objet message.

Cette propriété peut posséder la valeur suivante:

CMC_AA_DELETE

positionnée: le message spécifié peut être supprimé par le système de messagerie sous-jacent une fois qu'il a été déposé avec succès pour un transfert.

non positionnée: le message spécifié est placé dans le dossier d'émission, si ce dernier existe. Il est supprimé dans le cas contraire.

Cette propriété est du type **CMC_pv_flags**.

5.7.2 **Jeu de caractères** (*Character Set*)

NOM

Jeu de caractères du conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_CHARACTER_SET          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Character Set//EN"
```

DESCRIPTION

Cette propriété indique le jeu de caractères à utiliser pour véhiculer des données du type chaîne de caractères entre l'utilisateur et l'interface CMC. La valeur de la propriété est un tableau d'identificateurs d'objets jeu de caractères associés à la mise en œuvre. Le tableau est un tableau avec comptage. Le premier identificateur de tableau jeu de caractères est le jeu de caractères utilisé par défaut si l'utilisateur n'en spécifie pas un d'une manière explicite dans la fonction **cmc_logon()**. Prière de se référer à la clause propre à la plate-forme au B.2.4 traitant des identificateurs d'objets pour les jeux de caractères usuels.

Cette propriété est du type **CMC_pv_array_of_guid**.

5.7.3 **Conformité** (*Conformance*)

NOM

Conformité du conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_CONF          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Conf//EN"
```

DESCRIPTION

Cette propriété spécifie le niveau de conformité de la mise en œuvre. La valeur de la propriété sera soit CMC_CONF_SIMPLE_CMC si la mise en œuvre ne prend en charge que l'interface CMC simple, soit CMC_CONF_FULL_CMC si la mise en œuvre prend en charge une version autonome de l'interface CMC complète. La valeur CMC_CONF_FULL_CMC implique que la mise en œuvre prend également en charge l'interface CMC simple telle qu'elle est requise par la clause de conformité.

Cette propriété est du type **CMC_pv_enum**.

5.7.4 **Service par défaut** (*Default Service*)

NOM

Service par défaut du conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_DEFAULT_SERVICE          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Default Service//EN"
```

DESCRIPTION

Cette propriété donne le nom du service par défaut. Une valeur de pointeur NULL est utilisée si aucun nom de service par défaut n'est disponible. Cette propriété peut être utilisée, avec la propriété CMC_PT_PROFILE_DEFAULT_USER, comme valeurs par défaut du nom du service et du nom de l'utilisateur dans la fonction **cmc_logon()**. Cette information sera renvoyée dans le jeu de caractères par défaut de la mise en œuvre.

Cette propriété est du type **CMC_pv_string**.

5.7.5 **Utilisateur par défaut** (*Default User*)

NOM

Utilisateur par défaut du conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_DEFAULT_USER          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Default User//EN"
```

DESCRIPTION

Cette propriété donne le nom de l'utilisateur CMC par défaut. Une valeur de pointeur NULL est utilisée si aucun nom d'utilisateur par défaut n'est disponible. Cette propriété peut être utilisée, avec la propriété CMC_PT_PROFILE_DEFAULT_SERVICE comme valeurs par défaut du nom du fournisseur et du nom de l'utilisateur dans la fonction **cmc_logon()**. Cette information sera renvoyée dans le jeu de caractères par défaut de la mise en œuvre.

Cette propriété est du type **CMC_pv_string**.

5.7.6 **Fin de ligne** (*Line Terminator*)

NOM

Fin de ligne du conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_LINE_TERM     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Line Term//EN"
```

DESCRIPTION

Cette propriété spécifie le caractère de fin de ligne utilisé pour terminer des lignes de chaîne de caractères. La valeur de cette propriété est égale à CMC_LINE_TERM_CRLF si le délimiteur de ligne est un retour chariot suivi d'une nouvelle ligne, CMC_LINE_TERM_LF si le délimiteur de ligne est une nouvelle ligne ou CMC_LINE_TERM_CR si le délimiteur de ligne est un retour chariot.

Cette propriété est du type **CMC_pv_enum**.

5.7.7 **Classe d'objets** (*Object Class*)

NOM

Classe d'objets conteneur de profil

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique que l'objet fait partie de la classe conteneur de profil.

La seule valeur valide de cette propriété est CMC_PT_OBJECT_CLASS_PROFILE indiquant que l'objet fait partie de la classe conteneur de profil.

Cette propriété est du type **CMC_pv_enum**.

5.7.8 **Extensions d'objet prises en charge** (*Object Extensions Supported*)

NOM

Extensions d'objet prises en charge par le conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_OBJECT_EXT    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Object Ext//EN"
```

DESCRIPTION

Cette propriété indique les extensions de classes d'objets prises en charge par la mise en œuvre. Les valeurs de la propriété sont représentées par un tableau d'identificateurs globaux de classes d'objets pour les extensions de classe d'objets prises en charge par la mise en œuvre. Il n'existe pas d'ordre implicite des objets identificateur GUID au sein du tableau.

Cette propriété est du type **CMC_pv_array_guid**.

5.7.9 **Objets pris en charge** (*Objects Supported*)

NOM

Objets conteneur de profil pris en charge

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_OBJECT_SUP    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Object Sup//EN"
```

DESCRIPTION

Cette propriété spécifie les classes d'objets pris en charge par la mise en œuvre. Les valeurs de la propriété sont représentées par un tableau d'identificateurs globaux de classe d'objets pour les objets pris en charge par la mise en œuvre. Il n'existe pas d'ordre implicite pour les objets d'identificateur GUID au sein du tableau.

Cette propriété est du type **CMC_pv_array_guid**.

5.7.10 **Propriétés prises en charge** (*Properties Supported*)

NOM

Propriétés de conteneur de profil prises en charge

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_PROP_SUP      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Prop Sup//EN"
```

DESCRIPTION

Cette propriété spécifie les propriétés prises en charge par la mise en œuvre. Les valeurs de la propriété sont représentées par un tableau d'identificateurs globaux de propriété pour les propriétés d'objets prises en charge par la mise en œuvre. Il n'existe pas d'ordre implicite pour les objets d'identificateur GUID au sein du tableau.

Cette propriété est du type **CMC_pv_array_guid**.

5.7.11 **Extensions de propriété prises en charge** (*Property Extensions Supported*)

NOM

Propriétés prises en charge par le conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_PROP_EXT      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Prop Ext//EN"
```

DESCRIPTION

Cette propriété spécifie les extensions de propriété prises en charge par la mise en œuvre. Les valeurs de la propriété sont représentées par un tableau d'identificateurs globaux d'extension de propriété pour les extensions de propriété d'objet prises en charge par la mise en œuvre. Il n'existe pas d'ordre implicite pour les objets d'identificateur GUID au sein du tableau.

Cette propriété est du type **CMC_pv_array_guid**.

5.7.12 **Mot de passe exigé** (*Required Password*)

NOM

Mot de passe exigé pour le conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_REQ_PASSWORD          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Req Password//EN"
```

DESCRIPTION

Cette propriété indique si un mot de passe est exigé pour une ouverture de session avec le service. Les valeurs de la propriété sont CMC_REQUIRED_NO si le mot de passe n'est pas exigé pour ouvrir une session, CMC_REQUIRED_OPT s'il est optionnel ou CMC_REQUIRED_YES s'il est exigé.

Cette propriété est du type **CMC_pv_enum**.

5.7.13 **Service exigé** (*Required Service*)

NOM

Service exigé pour le conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_REQ_SERVICE          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Req Service//EN"
```

DESCRIPTION

Cette propriété indique si un nom de service est exigé pour une ouverture de session avec le service. Les valeurs de la propriété sont CMC_REQUIRED_NO si le nom du service n'est pas exigé pour ouvrir une session, CMC_REQUIRED_OPT s'il est optionnel ou CMC_REQUIRED_YES s'il est exigé.

Cette propriété est du type **CMC_pv_enum**.

5.7.14 **Utilisateur exigé** (*Required User*)

NOM

Utilisateur exigé pour le conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_REQ_USER     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Req User//EN"
```

DESCRIPTION

Cette propriété indique si un nom d'utilisateur est exigé pour une ouverture de session avec le service. Les valeurs de la propriété sont CMC_REQUIRED_NO si le nom de l'utilisateur n'est pas exigé pour ouvrir une session, CMC_REQUIRED_OPT s'il est optionnel ou CMC_REQUIRED_YES s'il est exigé.

Cette propriété est du type **CMC_pv_enum**.

5.7.15 **Prise en charge des chaînes avec comptage** (*Support Counted Strings*)

NOM

Prise en charge des chaînes de caractères avec comptage par le conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_SUP_COUNTED_STR      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Sup Counted Str//EN"
```

DESCRIPTION

Cette propriété indique si le service prend en charge des chaînes de caractères avec comptage. La valeur de la propriété sera positionnée sur "Vrai" si le fanion CMC_COUNTED_STRING_TYPE est pris en charge lors de l'ouverture de la session.

Cette propriété est du type **CMC_pv_boolean**.

5.7.16 **Prise en charge d'absence de marquage en lecture** (*Support No Mark As Read*)

NOM

Prise en charge d'absence de marquage en lecture par le conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_SUP_NOMKMSGREAD      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Sup NoMkMsgRead//EN"
```

DESCRIPTION

Cette propriété indique si le service prend en charge la fonction **cmc_read()** avec l'opération CMC_DO_NOT_MARK_AS_READ. La valeur de la propriété sera positionnée sur "Vrai" si le fanion CMC_DO_NOT_MARK_AS_READ est pris en charge par l'opération **cmc_read()**.

Cette propriété est du type **CMC_pv_boolean**.

5.7.17 **Interface utilisateur disponible** (*User Interface Available*)

NOM

Interface utilisateur disponible pour le conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_UI_AVAIL      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile UI Avail//EN"
```

DESCRIPTION

Cette propriété indique si une interface utilisateur est disponible pour la saisie et la résolution de paramètre. La valeur de la propriété sera positionnée sur "Vrai" s'il existe une interface utilisateur fournie par la mise en œuvre CMC.

Cette propriété est du type **CMC_pv_boolean**.

5.7.18 **Utilisateurs** (*Users*)

NOM

Utilisateurs du conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_USERS         \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Users//EN"
```

DESCRIPTION

Cette propriété identifie l'utilisateur qui est en cours de session sur le conteneur racine. Les valeurs de la propriété sont les noms de destinataire des utilisateurs actuellement en cours de session sur le conteneur racine. La prise en charge de la propriété est optionnelle pour des mises en œuvre se conformant à la présente Recommandation.

Il n'existe pas d'ordre implicite dans le tableau pour les noms supplémentaires de destinataires.

Cette propriété est du type **CMC_pv_array_string**.

5.7.19 **Version de la mise en œuvre** (*Version of the Implementation*)

NOM

Version de la mise en œuvre du conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_VER_IMPLEM    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Ver Implem//EN"
```

DESCRIPTION

Cette propriété indique la version de la mise en œuvre. La valeur de la propriété sera positionnée sur le produit du numéro de version par 100. La version 1.01 correspondra, par exemple, au nombre 101.

Cette propriété est du type **CMC_pv_uint16**.

5.7.20 **Version de la spécification** (*Version of the Specification*)

NOM

Version de la spécification du conteneur de profil

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_PROFILE_VER_SPEC      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Profile Ver Spec//EN"
```

DESCRIPTION

Cette propriété indique la version de la spécification CMC prise en charge par la mise en œuvre. La valeur de la propriété sera positionnée sur le produit du numéro de version de la spécification CMC par 100. La version 1.00 correspondra, par exemple, au nombre 100.

Cette propriété est du type **CMC_pv_uint16**.

5.8 Propriétés d'objets destinataire

Les objets destinataire identifient les utilisateurs individuels dans un service de messagerie. Un objet destinataire n'est pas un objet conteneur. Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés d'un objet destinataire.

5.8.1 **Adresse** (*Address*)

NOM

Adresse du destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_RECIPIENT_ADDRESS     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Address//EN"
```

DESCRIPTION

Cette propriété donne l'adresse du destinataire. Le format de la chaîne de caractères dépend de la mise en œuvre.

Dans des applications de passerelle, l'adresse de l'objet destinataire jouant le rôle d'expéditeur peut être ajoutée par la passerelle.

Cette propriété est du type **CMC_pv_string**.

5.8.2 **Renvoi de contenu exigé** (*Content Return Requested*)

NOM

Renvoi de contenu exigé pour le destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_RECIPIENT_CONTENT_RETURN_REQUESTED                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Content Return Requested//EN"
```

DESCRIPTION

Cette propriété indique si le message en question doit être renvoyé avec le compte rendu de remise en cas d'échec de la remise. Le message ne sera pas renvoyé si aucun compte rendu n'est demandé, quelle que soit l'indication fournie par cette propriété.

Cette propriété est du type **CMC_pv_boolean**.

5.8.3 **Nom** (*Name*)

NOM

Nom du destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_RECIPIENT_NAME        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Name//EN"
```

DESCRIPTION

Cette propriété donne le nom affichable du destinataire. Le format de la chaîne de caractères dépend de la mise en œuvre.

Cette propriété est du type **CMC_pv_string**.

5.8.4 **Classe d'objets** (*Object Class*)

NOM

Classe d'objets destinataire

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique que l'objet fait partie de la classe destinataire.

Cette propriété est créée par la fonction **cmc_open_object_handle()**.

La seule valeur valide de cette propriété est CMC_PT_OBJECT_CLASS_RECIPIENT spécifiant que l'objet fait partie de la classe destinataire.

Cette propriété est du type **CMC_pv_enum**.

5.8.5 **Acquittement demandé** (*Receipt Requested*)

NOM

Acquittement demandé au destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_RECIPIENT_RECEIPT_REQUESTED                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Receipt Requested//EN"
```

DESCRIPTION

Cette propriété indique le type d'acquittement devant être renvoyé pour le message.

Si l'objet message et l'objet destinataire spécifient tous deux une propriété d'acquittement demandé, la valeur spécifiée par l'objet destinataire aura priorité par rapport à celle spécifiée par l'objet message. La mise en œuvre n'a pas l'obligation de prendre en charge cette propriété au niveau de l'objet destinataire.

Cette propriété peut prendre les valeurs suivantes:

CMC_RECEIPT_RN
CMC_RECEIPT_NRN
CMC_RECEIPT_BOTH
CMC_RECEIPT_NONE

CMC_RECEIPT_RN – Demande le renvoi d'un avis d'acquittement si le destinataire a accepté le message en question.

CMC_RECEIPT_NRN – Demande le renvoi d'un avis de non-acquittement si le destinataire n'a pas accepté le message en question.

CMC_RECEIPT_BOTH – Demande le renvoi d'un avis d'acquittement ou de non-acquittement selon que le destinataire a accepté ou non le message en question.

CMC_RECEIPT_NONE – Demande qu'aucun avis ne soit renvoyé, indépendamment de l'acceptation du message en question par le destinataire.

Cette propriété est du type **CMC_pv_enum**.

5.8.6 **Compte rendu demandé** (*Report Requested*)

NOM

Compte rendu demandé au destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_RECIPIENT_REPORT_REQUESTED                     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Report Requested//EN"
```

DESCRIPTION

Cette propriété indique le type de compte rendu renvoyé pour le message en question. Elle est utilisée pour indiquer si le message en question a été remis ou non par les systèmes de transport de messagerie sous-jacents.

Cette propriété peut prendre les valeurs suivantes:

CMC_REPORT_DR
CMC_REPORT_NDR
CMC_REPORT_BOTH
CMC_REPORT_NONE

CMC_REPORT_DR – Spécifie qu'un compte rendu de remise est demandé.

CMC_REPORT_NDR – Spécifie qu'un compte rendu de non-remise est demandé.

CMC_REPORT_BOTH – Spécifie qu'un compte rendu de remise ou de non-remise est demandé, selon le cas.

CMC_REPORT_NONE – Spécifie qu'aucun compte rendu n'est demandé.

Cette propriété est du type **CMC_pv_enum**.

5.8.7 **Fanion de responsabilité** (*Responsibility Flag*)

NOM

Fanion de responsabilité du destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_RECIPIENT_RESPONSIBILITY_FLAG          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Responsibility Flag//EN"
```

DESCRIPTION

Cette propriété spécifie un indicateur concernant la réception d'une copie du message par le destinataire. Elle est utile pour des applications de passerelle et dans des situations dans lesquelles une application peut accéder à des versions multiples de l'interface.

La valeur par défaut pour cette propriété est CMC_TRUE.

Cette propriété est du type **CMC_pv_boolean**.

5.8.8 **Rôle** (*Role*)

NOM

Rôle du destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_RECIPIENT_ROLE         \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Role//EN"
```

DESCRIPTION

Cette propriété indique le rôle du destinataire.

Cette propriété peut prendre les valeurs suivantes:

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 – Spécifie le destinataire primaire.

CMC_RECIPIENT_ROLE_CC – Spécifie le destinataire en copie.

CMC_RECIPIENT_ROLE_BCC – Spécifie le destinataire en copie muette.

CMC_RECIPIENT_ROLE_ORIGINATOR – Spécifie l'expéditeur du message.

CMC_RECIPIENT_ROLE_AUTHORIZING_USER – Spécifie l'utilisateur donnant l'autorisation.

CMC_RECIPIENT_ROLE_REPLY_TO – Spécifie le destinataire auquel doit être envoyé la réponse.

CMC_RECIPIENT_ROLE_FORWARDED – Spécifie le destinataire d'une réexpédition

CMC_RECIPIENT_ROLE_ACTUAL – Spécifie le destinataire effectif.

CMC_RECIPIENT_ROLE_INTENDED – Spécifie le destinataire souhaité.

Cette propriété est du type **CMC_pv_enum**.

5.8.9 **Type** (*Type*)

NOM

Type de destinataire

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_RECIPIENT_TYPE        \
    "–//XAPIA/CMC/PROPERTY//NONSGML Recipient Type//EN"
```

DESCRIPTION

Cette propriété indique le type de destinataire.

Cette propriété peut prendre les valeurs suivantes:

CMC_RCT_UNKNOWN (=0)
CMC_RCT_INDIVIDUAL
CMC_RCT_GROUP
CMC_RCT_REPORT_RECIPIENT

CMC_RCT_UNKNOWN – Spécifie un type de destinataire non connu.

CMC_RCT_INDIVIDUAL – Spécifie que le destinataire est un individu.

CMC_RCT_GROUP – Spécifie que le destinataire est une liste de distribution.

CMC_RCT_REPORT_RECIPIENT – Spécifie que le destinataire est le destinataire d'un message de compte rendu.

Cette propriété est du type **CMC_pv_enum**.

5.9 Propriétés de l'objet compte rendu

L'objet compte rendu se constitue d'une collection de propriétés d'objet spécifique d'un compte rendu. Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés de l'objet compte rendu.

5.9.1 **Identificateur d'application** (*Application Id*)

NOM

Identificateur d'application du compte rendu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_APPLICATION_ID \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Application Id//EN"
```

DESCRIPTION

Cette propriété spécifie un identificateur global non-ambigu pour le compte rendu. Elle est positionnée par l'application.

Cette propriété est du type **CMC_pv_string**.

5.9.2 **Identificateur** (*Id*)

NOM

Identificateur de compte rendu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_ID     \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Id//EN"
```

DESCRIPTION

Cette propriété spécifie un identificateur global non-ambigu pour le compte rendu. Cette propriété est positionnée par la fonction **cmc_send_message_object()**, elle est définie par le service de messagerie (établi au moment du dépôt) et est non-ambiguë au sein du domaine.

L'identificateur de message peut être ajouté ou mis à jour par l'appelant dans des applications de passerelle.

Cette propriété est du type **CMC_pv_guid**.

5.9.3 Comptage d'article (*Item Count*)

NOM

Comptage d'article de compte rendu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_ITEM_COUNT                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Item Count//EN"
```

DESCRIPTION

Cette propriété indique le nombre d'articles de contenu de niveau sommital contenus dans un compte rendu. Le comptage n'inclut pas les articles de contenu imbriqués dans d'autres articles de contenu ou messages. Cette propriété est positionnée par la mise en œuvre.

Cette propriété est du type **CMC_pv_uint32**.

5.9.4 Identificateur du système de messagerie (*Messaging System Id*)

NOM

Identificateur du système de messagerie du compte rendu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_MESSAGING_SYSTEM_ID                                   \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Messaging System Id//EN"
```

DESCRIPTION

Cette propriété spécifie l'identificateur du système de transport de messages ou l'identificateur de la passerelle qui a créé le compte rendu.

Cette propriété est du type **CMC_pv_enum**.

5.9.5 Classe d'objets (*Object Class*)

NOM

Classe d'objets compte rendu

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique que l'objet fait partie de la classe compte rendu.

Cette propriété est créée par la fonction **cmc_open_object_handle()**.

La seule valeur valide de cette propriété est CMC_PT_OBJECT_CLASS_REPORT spécifiant que l'objet fait partie de la classe compte rendu.

Cette propriété est du type **CMC_pv_enum**.

5.9.6 Lu (*Read*)

NOM

Compte rendu lu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_READ                                          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Read//EN"
```

DESCRIPTION

Cette propriété indique si le compte rendu a été lu.

Cette propriété est du type **CMC_pv_boolean**.

5.9.7 **Taille** (*Size*)

NOM

Taille du compte rendu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_SIZE                                          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Size//EN"
```

DESCRIPTION

Cette propriété indique la taille du compte rendu.

Cette propriété est du type **CMC_pv_uint32**.

5.9.8 **Sujet** (*Subject*)

NOM

Sujet du compte rendu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_SUBJECT                                       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Subject//EN"
```

DESCRIPTION

Cette propriété décrit le sujet du compte rendu. Elle possède une valeur par défaut égale à la chaîne de caractères nulle.

Cette propriété est du type **CMC_pv_string**.

5.9.9 **Identificateur du message sujet** (*Subject Message Id*)

NOM

Identificateur du message sujet du compte rendu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_SUBJECT_MESSAGE_ID                            \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Subject Message Id//EN"
```

DESCRIPTION

Cette propriété identifie le message utilisateur qui est à l'origine de la génération de ce compte rendu.

La valeur de la propriété peut être une référence textuelle ou une approximation textuelle de l'identificateur de message de la correspondance précédente.

Cette propriété est du type **CMC_pv_string**.

5.9.10 **Instant de réception** (*Time Received*)

NOM

Instant de réception du compte rendu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_TIME_RECEIVED                                 \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Time Received//EN"
```

DESCRIPTION

Cette propriété indique la date et l'heure de réception du compte rendu.

Cette propriété est du type **CMC_pv_iso_date_time**.

5.9.11 **Instant d'émission** (*Time Sent*)

NOM

Instant d'émission du compte rendu

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_TIME_SENT                                    \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Time Sent//EN"
```

DESCRIPTION

Cette propriété indique la date et l'heure d'émission du compte rendu.

Cette propriété est positionnée par le service dans la fonction **cmc_send_message_object**.

Cette propriété est du type **CMC_pv_iso_date_time**.

5.9.12 **Non émis** (*Unsent*)

NOM

Compte rendu non émis

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_REPORT_UNSENT                                       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Report Unsent//EN"
```

DESCRIPTION

Cette propriété indique que le compte rendu n'a pas été émis.

Cette propriété est du type **CMC_pv_boolean**.

5.10 Propriétés de l'objet conteneur racine

La racine est le conteneur noyau de base constitué de diverses propriétés et d'autres objets conteneur. Le conteneur racine se compose des carnets d'adresses (contenant des destinataires et d'autres carnets d'adresses), d'un conteneur de profil et de conteneurs de messages. L'objet destinataire dans le conteneur racine ne peut pas être modifié.

Les sous-paragraphes qui suivent définissent, déclarent et décrivent les propriétés de l'objet conteneur racine.

5.10.1 **Enfant autorisé** (*Child Allowed*)

NOM

Enfant autorisé pour le conteneur racine

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_ROOT_CONTAINER_CHILD_ALLOWED          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Child Allowed//EN"
```

DESCRIPTION

Cette propriété autorise ou interdit l'existence d'un enfant pour le conteneur racine.

Cette propriété est du type **CMC_pv_boolean**.

5.10.2 **Commentaire** (*Comment*)

NOM

Commentaire du conteneur racine

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_ROOT_CONTAINER_COMMENT       \
    "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Comment//EN"
```

DESCRIPTION

Cette propriété fournit un commentaire décrivant le conteneur racine.

Cette propriété est du type **CMC_pv_string**.

5.10.3 **Emplacement** (*Location*)

NOM

Emplacement du conteneur racine

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_ROOT_CONTAINER_LOCATION      \
    "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Location//EN"
```

DESCRIPTION

Cette propriété indique l'emplacement du conteneur racine.

Cette propriété peut prendre les valeurs suivantes:

CMC_ROOT_CONTAINER_LOCATION_LOCAL
CMC_ROOT_CONTAINER_LOCATION_SERVER
CMC_ROOT_CONTAINER_LOCATION_UNKNOWN

CMC_ROOT_CONTAINER_LOCATION_LOCAL – Spécifie que l'emplacement du conteneur racine est local et non sur le serveur de messagerie.

CMC_ROOT_CONTAINER_LOCATION_SERVER – Spécifie que l'emplacement du conteneur racine est sur le serveur de messagerie.

CMC_ROOT_CONTAINER_LOCATION_UNKNOWN – Spécifie que l'emplacement du conteneur racine est inconnu.

Cette propriété est du type **CMC_pv_enum**.

5.10.4 **Nom** (*Name*)

NOM

Nom du conteneur racine

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_ROOT_CONTAINER_NAME          \
    "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Name//EN"
```

DESCRIPTION

Cette propriété donne le nom du conteneur racine.

Cette propriété est du type **CMC_pv_string**.

5.10.5 **Classe d'objets** (*Object Class*)

NOM

Classe d'objets du conteneur racine

DÉCLARATION EN LANGAGE C

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

DESCRIPTION

Cette propriété indique que l'objet appartient à la classe racine.

Cette propriété est créée par la fonction **cmc_open_object_handle()**.

La seule valeur valide de cette propriété est CMC_PT_OBJECT_CLASS_ROOT spécifiant que l'objet appartient à la classe racine.

Cette propriété est du type **CMC_pv_enum**.

5.10.6 **Partagé** (*Shared*)

NOM

Conteneur racine partagé

DÉCLARATION EN LANGAGE C

```
#define CMC_PT_ROOT_CONTAINER_SHARED       \
   "–//XAPIA/CMC/PROPERTY//NONSGML Root Container Shared//EN"
```

DESCRIPTION

Cette propriété indique si le conteneur racine est partagé avec une autre entité.

Cette propriété est du type **CMC_pv_boolean**.

6 Interface fonctionnelle

Le présent paragraphe définit les fonctions de l'interface commune d'appel de messagerie. Elle spécifie les fonctions de l'interface générique ainsi que celles de l'interface en langage C. Les descriptions de l'interface C sont reprises dans l'Annexe A, "Résumé des déclarations en langage C".

6.1 Fonctions de l’interface CMC simple

L'interface CMC simple offre un ensemble de base de fonctions destinées à fournir des capacités de messagerie à des applications utilisant la messagerie. Ces fonctions ont été publiées précédemment comme version 1.0 de l'interface CMC. Le Tableau 14 donne la liste des fonctions de l'interface CMC simple.

Tableau 14/X.446 – Interface CMC simple

Fonction	Description
Emission de messages	
émission	émettre un message de courrier
émission de documents	émission de courrier à partir d'une chaîne de caractères
Réception de messages	
agir sur	exécuter une action sur un message spécifié
liste	liste contenant une information résumée pour des messages correspondant à des critères spécifiés
lecture	lire et renvoyer un message spécifié
Recherche de noms	
recherche	recherche d'information d'adressage
Administration	
libérer	libérer la mémoire allouée par le service de messagerie
fin de session	mettre fin à une session avec le service de messagerie
ouverture de session	établir une session avec le service de messagerie
demande de configuration	déterminer une information concernant le service CMC installé

Les pages de manuel correspondant à ces propriétés sont données ci-après.

6.1.1 Emission de message

6.1.1.1 Emission (*Send*)

NOM

Emission – Emettre un message.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction émet un message de courrier.

L'appelant peut, d'une manière optionnelle, fournir une liste de destinataires, un texte de sujet et un texte de note. La fonction peut émettre le message si un ou plusieurs destinataires sont présents.

Un résultat positif de cette fonction n'implique pas nécessairement la validation des destinataires.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Les identificateurs opaques de session sont créés par un appel à une fonction d'ouverture de session et invalidés par un appel à une fonction de fermeture de session.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide et qu'une session valide n'a pas été créée au moyen de l'interface utilisateur.

Message (*message*), type message

Structure de message renfermant le contenu du message à émettre. Si le fanion d'argument d'extension CMC_X_SEND_UI_REQUESTED n'est pas positionné ou n'est pas pris en charge, il est nécessaire que l'un au moins des destinataires soit du type TO, CC ou BCC.

Les autres champs sont facultatifs. Les champs instant d'émission et référence de message sont ignorés.

Les règles suivantes s'appliquent aux champs de la structure de message:

***destinataires* –** Le nombre de destinataires peut être limité par certains services. Le code erreur CMC_E_TOO_MANY_RECIPIENTS est renvoyé si cette limite est dépassée. Un pointeur de valeur NULL doit être assigné au champ *destinataires* si aucun destinataire n'est spécifié.

Le descripteur de destinataire peut contenir le nom du destinataire, une adresse ou un couple nom/adresse. S'il spécifie uniquement un nom, le nom est résolu vers une adresse en utilisant des règles de résolution définies par la mise en œuvre. Si une adresse seule est spécifiée, elle est utilisée pour la remise et comme nom en clair du destinataire. Une résolution d'adresse ne doit pas être effectuée si une adresse et un nom sont spécifiés. Le nom est ignoré si une mise en œuvre ne peut prendre en charge à la fois le nom et l'adresse. L'adresse se trouve dans un format défini par la mise en œuvre et il est fait l'hypothèse qu'elle a été fournie d'une autre manière par la mise en œuvre. Un destinataire du type expéditeur n'est pas nécessaire pour l'émission. Son effet est défini par la mise en œuvre CMC s'il est présent;

***attachements* –** Le nombre d'attachements par message peut être limité par certains services. Le code erreur CMC_E_TOO_MANY_FILES est renvoyé si la limite est dépassée. Un pointeur de valeur NULL indique l'absence d'attachements. Les fichiers d'attachement sont lus avant le retour de la fonction **cmc_send(),** de sorte que ces fichiers peuvent être modifiées ou supprimées sans affecter le message;

sujet – Un pointeur de valeur NULL indique l'absence de texte de sujet. Certaines mises en œuvre peuvent tronquer des lignes de sujet trop longues ou contenant des caractères de retour chariot, de saut de ligne ou de saut de page;

texte de note – Un pointeur de valeur NULL indique l'absence de texte. Les mises en œuvre peuvent imposer des limites à la taille du texte. Si le texte de la note excède la limite du service, ce dernier peut déclasser le corps du texte sous la forme d'un attachement ou générer le code erreur CMC_E_TEXT_TOO_LARGE;

type de message – Pointeur vers une chaîne de caractères contenant le type du message. Le type spécifie le type de message émis (voir la description de la structure de données message pour les détails). La chaîne de caractères "CMC: IPM" est utilisée pour spécifier un message interpersonnel. La valeur "CMC: IPM" est prise par défaut si le pointeur possède la valeur NULL ou pointe vers une chaîne de caractères vide;

fanions – Le fanion suivant peut être utilisé lors de l'émission d'un message:

CMC_MSG_TEXT_NOTE_AS_FILE.

Tout autre fanion sera ignoré. Prière de se référer à la description de la structure du message pour plus d'information concernant ces fanions.

Fanions d'émission (*send_flags*), type fanions

Masque binaire de fanions. Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés.

CMC_LOGON_UI_ALLOWED

CMC_SEND_UI_REQUESTED

CMC_ERROR_UI_ALLOWED

CMC_COUNTED_STRING_TYPE

CMC_LOGON_UI_ALLOWED – Positionné si la fonction doit afficher une boîte de dialogue pour demander, si nécessaire, une ouverture de session. Si le fanion n'est pas positionné, aucune boîte de dialogue n'est affichée et le code erreur CMC_E_INVALID_SESSION_ID sera renvoyé si l'utilisateur n'est pas en cours de session.

CMC_SEND_UI_REQUESTED – Positionné si la fonction doit afficher une boîte de dialogue pour demander la saisie des destinataires, des champs du message et d'autres options d'émission. La fonction n'affiche pas de boîte de dialogue si ce fanion n'est pas positionné, mais un destinataire au moins doit être spécifié.

CMC_ERROR_UI_ALLOWED – Positionné si la fonction peut afficher une boîte de dialogue lorsqu'elle rencontre une erreur pouvant être récupérée. La fonction n'affichera pas de boîte de dialogue et renverra simplement un code erreur si le fanion n'est pas positionné.

CMC_COUNTED_STRING_TYPE – Positionné si le type de la chaîne de caractères utilisée dans le message est une chaîne de caractères avec comptage. Il est fait l'hypothèse que la chaîne de caractères se termine par un caractère nul dans le cas contraire. Ce fanion est ignoré si le paramètre de session est valide.

Identificateur d'interface utilisateur (*ui_id*), type identificateur d'interface utilisateur

Descripteur opaque d'interface utilisateur (par exemple une fenêtre de dialogue) utilisée pour résoudre toute question qui se pose lorsque le service exécute la fonction, en demandant une information supplémentaire à l'utilisateur ou en acquittant une information fournie précédemment.

Cet identificateur est ignoré si l'interface utilisateur n'est pas prise en charge par la mise en œuvre CMC.

Extensions d'émission (*send_extensions*), type extensions

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'émission (*send_extensions***), type extensions**

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code***)**

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Emission de documents (*Send Documents*)

NOM

Emission de document – Fonction d'émission de courrier, utilisant une chaîne de caractères.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction émet un message de courrier. Elle est prévue en premier lieu pour être appelée à partir d'un langage de "scénario" (par exemple, une macro de tableur) qui ne peut pas traiter de structures de données.

Cette fonction essaiera d'établir une session sans interface utilisateur pour l'ouverture de session. Si cela n'est pas possible la fonction, demandera une information d'ouverture de session afin d'établir une session. La session est toujours fermée lorsque la fonction se termine.

ARGUMENTS

Adresses de destinataire (*recipient addresses*), type chaîne de caractères

Pointeur vers une chaîne de caractères contenant les adresses des destinataires du message. Lorsque des destinataires multiples sont spécifiés, ils doivent être séparés par le caractère délimiteur. Les destinataires sont supposés être des destinataires primaires, à moins qu'ils ne soient préfixés par "cc:" ou "bcc:" indiquant des destinataires en copie ou en copie muette. Le préfixe "to:" peut également être utilisé à des fins de cohérence. Un pointeur de valeur NULL indique que les destinataires doivent être demandés au moyen d'un dialogue.

Sujet (*subject*), type chaîne de caractères

Pointeur vers une chaîne de caractères contenant le sujet du message. Un pointeur de valeur NULL indique l'absence de texte de sujet.

Note de texte (*text_note*), type chaîne de caractères

Pointeur vers une chaîne de caractères contenant le texte de note devant être véhiculé avec le message. Un pointeur de valeur NULL indique l'absence de note.

Fanions d'émission de documents (*send_doc_flags*), type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés.

CMC_COUNTED_STRING_TYPE
CMC_FIRST_ATTACH_AS_TEXT_NOTE

CMC_COUNTED_STRING_TYPE – Positionné si le type de la chaîne de caractères utilisée dans le message est une chaîne de caractères avec comptage. Il est fait l'hypothèse que la chaîne de caractères se termine par un caractère nul dans le cas contraire.

CMC_FIRST_ATTACH_AS_TEXT_NOTE – Positionné si le premier attachement émis contient la note de texte de message. Dans le cas contraire, la note est contenue dans le champ note de texte.

Chemin d'accès au fichier (*file_paths*), type chaîne de caractères

Pointeur vers une chaîne de caractères contenant les noms des chemins d'accès effectifs aux fichiers d'attachement. Lorsque des chemins multiples sont spécifiés, ils doivent être séparés par le caractère délimiteur.

Titres d'attachements (*attach_titles*), type chaîne de caractères

Pointeur vers une chaîne de caractères contenant les titres des attachements devant être vus par le destinataire. Lorsque plusieurs noms sont spécifiés, ils doivent être séparés par le caractère délimiteur.

Délimiteur (*Delimiter*), type chaîne de caractères

Pointeur vers un caractère utilisé pour délimiter les noms dans les chaînes des caractères indiquant les chemins d'accès aux fichiers, les noms de fichiers et les adresses. Ce caractère doit être choisi parmi ceux qui ne sont pas utilisés pour les noms de fichier du système d'exploitation et les noms de destinataire. Ce paramètre ne peut avoir la valeur NULL.

Identificateur d'interface utilisateur (*ui_id*), type identificateur d'interface utilisateur

Pointeur vers un identificateur d'une interface utilisateur (par exemple, une fenêtre de dialogue) utilisée pour résoudre toute question qui pourrait entraîner une erreur si elle faisait défaut ou pour demander à l'utilisateur une information supplémentaire.

Ignoré si l'interface utilisateur n'est pas prise en charge par la mise en œuvre CMC.

RÉSULTATS

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Réception de messages

6.1.2.1 Agir sur (*Act On*)

NOM

Agir sur – Effectue une action sur un message spécifié

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction exécute l'action spécifiée sur le message indiqué par la référence de message.

ARGUMENTS

Session (*Session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Les identificateurs opaques de session sont créés par un appel à une fonction d'ouverture de session et invalidés par un appel à une fonction de fermeture de session.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Référence de message (*message_reference*), type référence de message

Spécifie la référence de message du message faisant l'objet de l'action.

Le code erreur CMC_E_INVALID_MESSAGE_REFERENCE est renvoyé si la référence de message n'est pas valide (ou n'est plus valide, par exemple après une suppression). Un pointeur de référence de message de valeur NULL ou une référence de message de longueur nulle sont considérés comme non-valides pour des opérations exigeant ce paramètre.

Opération (*Operation*), type énumération

Opération à effectuer sur le message. Les opérations valides sont les suivantes:

CMC_ACT_ON_EXTENDED (= 0)
CMC_ACT_ON_DELETE

CMC_ACT_ON_EXTENDED – Examiner la liste d'extensions au sujet de l'action à effectuer.

CMC_ACT_ON_DELETE – L'action demandée consiste à supprimer le message spécifié dans la boîte aux lettres. Cette opération exige un paramètre de référence de message valide.

Act On Flags (*act_on_flags*), type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés. Les positionnements de fanions sont les suivants:

CMC_ERROR_UI_ALLOWED

CMC_ERROR_UI_ALLOWED – Positionné si la fonction peut afficher une boîte de dialogue lorsqu'elle rencontre des erreurs pouvant être récupérées. Si le fanion n'est pas positionné, la fonction n'affiche pas de boîte de dialogue et renverra simplement un code erreur.

Identificateur d'interface utilisateur (*ui_id*), type identificateur d'interface utilisateur

Descripteur opaque d'une interface utilisateur (par exemple une fenêtre de dialogue) utilisée pour résoudre toute question qui pourrait entraîner une erreur si elle faisait défaut.

Ignoré si l'interface utilisateur n'est pas prise en charge par la mise en œuvre CMC.

Extensions de l'action (*act_on_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. La valeur NULL indique que l'appelant n'utilise aucune extension. Pour plus de détails, voir la structure des extensions.

RÉSULTATS

Extensions de l'action (*act_on_extensions*), type extension

Les résultats du service seront disponibles dans cette extension si des extensions en entrée ont été fournies à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extension pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Liste (*List*)

NOM

Liste – Liste contenant des informations de résumé au sujet des messages correspondant à un critère spécifié.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction fournit une liste d'informations de résumé, incluant une référence de message, au sujet des messages qui correspondent à un critère spécifié. Les références de messages renvoyées peuvent être utilisées pour un traitement ultérieur des messages au moyen des fonctions **cmc_read()** et **cmc_act_on()**.

Les critères facultatifs sont les suivants:

– le message est d'un type spécifié,

– le message n'a pas encore été lu.

La recherche commence à partir d'une référence de message "de départ" spécifiée ou à partir du début de la boîte aux lettres. Il est possible de spécifier le nombre maximal de messages à lister. La fonction renvoie le nombre effectif de messages retournés.

D'une manière optionnelle, chaque résumé de message renvoyé dans le champ "résultat" peut se limiter à la référence de message.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Les identificateurs opaques de session sont créés par un appel à une fonction d'ouverture de session et invalidés par un appel à une fonction de fermeture de session.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Type de message (*message_type*), type chaîne de caractères

L'information n'est renvoyée que pour les messages du type spécifié. Le code erreur CMC_E_UNRECOGNIZED_MESSAGE_TYPE sera renvoyé si le type n'est pas reconnu. Le format de la chaîne de caractères type de message est donné au 5.4.23.

Une valeur NULL indique que l'information doit être renvoyée pour tous les messages disponibles.

Fanions de liste (*list_flags*), type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés. Les fanions concernés sont les suivants:

MC_ERROR_UI_ALLOWED
CMC_LIST_UNREAD_ONLY
CMC_LIST_MSG_REFS_ONLY
CMC_LIST_COUNT_ONLY

CMC_ERROR_UI_ALLOWED – Positionné si la fonction peut afficher une boîte de dialogue lorsqu'elle rencontre une erreur pouvant être récupérée. La fonction n'affichera pas de boîte de dialogue et renverra simplement un code erreur si le fanion n'est pas positionné.

CMC_LIST_UNREAD_ONLY – Si ce fanion est positionné, la liste ne doit contenir que les messages non encore lus. S'il n'est pas positionné, la liste peut contenir également contenir les messages déjà lus.

CMC_LIST_MSG_REFS_ONLY – Si ce fanion est positionné, seule la référence de message figure dans la structure de résultat. Les valeurs des autres champs ne sont pas définies et doivent être ignorées. Si le fanion n'est pas positionné, la totalité de l'information de la structure sera fournie en retour.

CMC_LIST_COUNT_ONLY – Si ce fanion est positionné, la fonction ne renverra aucune structure de résumé, mais uniquement l'indication du nombre de messages satisfaisant au critère spécifié. Les structures de résumé seront fournies en retour s'il n'est pas positionné.

Placement (*seed*), type référence de message

Spécifie la référence de message du message après lequel doit débuter la recherche. Si la référence de message n'est pas valide (ou n'est plus valide, par exemple après une suppression), le code erreur CMC_E_INVALID_MESSAGE_REFERENCE est renvoyé.

Un pointeur de référence de message de début de valeur NULL indique que la recherche doit débuter avec le premier message dans la boîte aux lettres.

Comptage (*count*), type uint32

Spécifie le nombre maximal de messages à renvoyer. Une valeur nulle spécifie l'absence de maximum.

Identificateur d'interface utilisateur (*ui_id*), type identificateur d'interface utilisateur

Descripteur opaque d'une interface utilisateur (par exemple une fenêtre de dialogue) utilisée pour résoudre toute question qui pourrait entraîner une erreur si elle faisait défaut.

Ignoré si l'interface utilisateur n'est pas prise en charge par la mise en œuvre CMC.

Liste d'extensions (*list_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Comptage (*count*), type uint32

Spécifie le nombre de messages effectivement renvoyés. Une valeur nulle est renvoyée si aucun message ne correspond au critère ou si la boîte aux lettres est vide.

Résultat (*result*), type résumé de message

Le champ résultat contient l'adresse à laquelle sera renvoyé un tableau de structures de résumé de message CMC. Ce tableau de structures est alloué par le service et doit être libéré en totalité au moyen d'un appel unique de la fonction **cmc_free()**.

Le champ référence de message contenu dans chaque résumé de message CMC peut être utilisé pour identifier des messages dans des appels ultérieurs de fonctions **cmc_read()** et **cmc_act_on()**.

NOTE – Il peut être nécessaire de copier le champ référence du message avant d'invoquer la fonction **cmc_free()** avec cette structure.

Si le fanion CMC_LIST_MSG_REFS_ONLY a été positionné, les structures de message CMC renvoyées contiendront uniquement les références de message. Les valeurs des autres champs ne sont pas définies et doivent être ignorées.

Extensions de liste (*list_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Pour plus de détails, voir la structure des extensions.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Lecture (*Read*)

NOM

Lecture – Lire et renvoyer un message spécifié.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction renvoie une structure de message contenant les données d'un message indiqué par la référence de message spécifiée. La structure de message renvoyée peut, d'une manière optionnelle, ne contenir que les en-têtes du message et de l'attachement.

Le champ note de texte est contenu dans le fichier indiqué par le premier attachement si le fanion CMC_MSG_TEXT_NOTE_AS_FILE est positionné dans le message renvoyé.

Dans le cas de systèmes pouvant marquer les messages comme ayant été lus, un message sera dans l'état "LU" après la réussite de cette fonction, sauf si le fanion CMC_DO_NOT_MARK_AS_READ est positionné.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Les identificateurs opaques de session sont créés par un appel à une fonction d'ouverture de session et invalidés par un appel à une fonction de fermeture de session. Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Référence de message (*message_reference*), type référence de message

Spécifie la référence de message du message devant être lu et renvoyé. Le code erreur CMC_E_INVALID_MESSAGE_REFERENCE est renvoyé si la référence de message n'est pas valide (ou n'est plus valide, par exemple après une suppression).

Un pointeur de référence de message de valeur NULL indique que le premier message de la boîte aux lettres doit être lu et renvoyé.

Fanions de lecture (*read_flags*), type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés. Les fanions concernés sont les suivants:

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 – Positionné si la fonction peut afficher une boîte de dialogue si elle rencontre une erreur pouvant être récupérée. La fonction n'affichera pas de boîte de dialogue et renverra simplement un code erreur si le fanion n'est pas positionné.

CMC_MSG_AND_ATT_HDRS_ONLY – Si ce fanion est positionné, le ou les champ(s) de nom de fichier d'attachement auront un contenu indéfini au moment du retour de la fonction **cmc_read()** et doivent être ignorés. Ceci peut être utile pour réduire la quantité de données transférées. Les noms de fichiers d'attachement seront renvoyés normalement si ce fanion n'est pas positionné.

NOTE – Si le fanion CMC_MSG_TEXT_NOTE_AS_FILE est positionné dans le message pour indiquer que le texte de la note est stocké dans le premier attachement, le champ nom de fichier d'attachement sera renvoyé pour l'attachement indépendamment du positionnement de ce fanion.

CMC_DO_NOT_MARK_AS_READ – Si ce fanion est positionné, l'état du message n'est pas modifié comme étant lu lorsque la fonction effectue son retour. Ceci supprimera également l'émission d'un compte rendu d'acquittement. Il est possible d'interroger la mise en œuvre pour déterminer si elle prend en charge cette fonctionnalité en positionnant le fanion CMC_CONFIG_SUP_NOMKMSGREAD dans l'appel de la fonction **cmc_query_config()**.

CMC_READ_FIRST_UNREAD_MESSAGE – Ce fanion n'est disponible que si une référence de message de valeur NULL est passée pour recevoir le premier message de la boîte aux lettres. S'il est positionné, le premier message qui n'est pas marqué comme étant lu doit être renvoyé. Dans le cas contraire, le premier message de la boîte aux lettres doit être renvoyé, qu'il soit marqué comme lu ou non.

Identificateur d'interface utilisateur (*ui_id*), identificateur d'interface utilisateur

Descripteur opaque d'une interface utilisateur (par exemple une fenêtre de dialogue) utilisée pour résoudre tout problème qui se manifeste lorsque le service traite cette fonction.

Ignoré si l'interface utilisateur n'est pas prise en charge par la mise en œuvre CMC.

Extensions en lecture (*read_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Message (*message*), type message

Le champ "message" contient l'adresse à laquelle doit être renvoyé un pointeur vers la structure de message CMC. Cette structure est allouée par le service et doit être libérée en utilisant la fonction **cmc_free()**.

Les données d'attachement seront renvoyées dans des fichiers et la structure de message CMC indiquera les noms de ces fichiers.

Si le fanion CMC_MSG_AND_ATT_HDRS_ONLY est positionné (voir "fanions"), la structure de message CMC ne renverra pas les fichiers d'attachement, comme décrit ci-dessus.

Extensions en lecture (*read_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des erreurs énumérées ci-dessous.

ERREURS

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 Consultation de noms

6.1.3.1 Consultation (*Look Up*)

NOM

Consultation – Consulte l'information d'adressage dans l'annuaire.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction recherche une information d'adressage dans l'annuaire fourni par la structure de message CMC. Son utilisation principale est de permettre la résolution d'un nom convivial vers une adresse.

Des adresses multiples peuvent être renvoyées. Un tableau de descripteurs de destinataires est alloué et renvoyé, avec comme contenu une information complètement résolue pour chaque élément.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Les identificateurs opaques de session sont créés par un appel à une fonction d'ouverture de session et invalidés par un appel à une fonction de fermeture de session.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide et qu'une session valide n'a pas été créée à l'aide de l'interface utilisateur.

Destinataire en entrée (*recipient_in*), type destinataire

Dans le cas d'une résolution de nom, le champ nom de la structure contient le nom à résoudre. Le type de nom peut être positionné afin de fournir des informations sur la résolution souhaitée pour le nom. Prière de se référer à la documentation de la structure de destinataire pour ce qui est des types possibles.

Dans le cas d'un affichage des détails du destinataire, la structure de destinataire doit contenir une entrée dont la résolution conduit à un destinataire unique. Le code erreur CMC_E_AMBIGUOUS_RECIPIENT sera renvoyé dans le cas contraire.

Dans le cas d'affichage de l'interface utilisateur en vue de créer les listes d'adressage, ce champ contient un pointeur vers un tableau de destinataires terminé par le fanion CMC_RECIP_LAST_ELEMENT. La liste de destinataires sera utilisée comme valeur par défaut pour l'affichage de la liste d'adresses par l'interface utilisateur.

Toutes les structures de destinataire autres que la première seront ignorées dans le cas d'une résolution de nom combinée avec l'affichage de détails du destinataire.

Fanions de consultation (*look_up_flags*), type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés. Les fanions concernés sont les suivants:

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 – Positionné si la fonction doit afficher une boîte de dialogue pour demander, si nécessaire, une ouverture de session. Si le fanion n'est pas positionné, aucune boîte de dialogue n'est affichée et le code erreur CMC_E_INVALID_SESSION_ID sera renvoyé si l'utilisateur n'est pas en cours de session.

CMC_ERROR_UI_ALLOWED – Positionné si la fonction peut afficher une boîte de dialogue lorsqu'elle rencontre une erreur pouvant être récupérée. La fonction n'affichera pas de boîte de dialogue et renverra simplement un code erreur si le fanion n'est pas positionné.

CMC_COUNTED_STRING_TYPE – Positionné si le type de la chaîne de caractères utilisée dans le message est une chaîne de caractères avec comptage. Il est fait l'hypothèse que la chaîne de caractères se termine par un caractère nul dans le cas contraire. Ce fanion est ignoré si le paramètre de session est valide.

CMC_LOOKUP_RESOLVE_PREFIX_SEARCH – Si ce fanion est positionné, la recherche se fera par préfixe. Une recherche par préfixe signifie que tous les noms correspondant à la chaîne de caractères de préfixe, à partir du premier caractère du nom, seront considérés comme concordants. Si le fanion n'est pas positionné, la méthode de recherche sera une concordance exacte. Les mises en œuvre CMC doivent prendre en charge une recherche simple par préfixe. La fourniture de la recherche par joker ou par sous-chaîne est optionnelle.

CMC_LOOKUP_RESOLVE_IDENTITY – Si ce fanion est positionné, la fonction renverra un enregistrement de destinataire donnant l'identité de l'utilisateur au sein du système de messagerie. Une résolution de nom non-ambiguë sera effectuée si l'opération ne peut être faite d'une manière non-ambiguë. Ceci permet à l'application de déterminer l'adresse de l'utilisateur actuel.

CMC_LOOKUP_RESOLVE_UI – Positionné si la mise en œuvre CMC doit tenter de lever une ambiguïté de nom au moyen d'un dialogue avec l'utilisateur pour la résolution de nom. Si ce fanion n'est pas positionné, les résolutions qui n'aboutissent pas à un nom unique reverront le code erreur CMC_E_AMBIGUOUS_RECIPIENT dans le cas de services qui doivent effectuer une résolution de nom non-ambigüe. Des services qui peuvent renvoyer des noms multiples renverront une liste comme indiqué par d'autres paramètres de la fonction. Ce fanion est pris en charge d'une manière optionnelle par les mises en œuvre.

CMC_LOOKUP_DETAILS_UI – Si ce fanion est positionné, la fonction affichera les détails correspondant au destinataire en entrée. Ceci n'aura d'effet que pour le premier destinataire de la liste. Cette action ne sera pas effectuée et le code erreur CMC_E_AMBIGUOUS_RECIPIENT sera renvoyé si la résolution de nom conduit à plus d'une adresse.

CMC_LOOKUP_ADDRESSING_UI – Si ce fanion est positionné, la fonction affichera l'interface utilisateur afin de permettre la création d'une liste d'adressage pour un message et un balayage général de l'annuaire. La liste de destinataires passée à la fonction sera la liste originale de destinataires pour l'interface utilisateur. Cette fonction renverra la liste des destinataires choisis par l'utilisateur. Ce fanion est pris en charge d'une manière optionnelle par les mises en œuvre.

Identificateur d'interface utilisateur (*ui_id*), type identificateur d'interface utilisateur

Descripteur opaque d'une interface utilisateur (par exemple une fenêtre de dialogue) utilisée pour résoudre tout problème qui se manifeste lorsque le service traite cette fonction.

Ignoré si l'interface utilisateur n'est pas prise en charge par la mise en œuvre CMC.

Comptage (*count*), type uint32

Spécifie le nombre maximal de noms à renvoyer. Une valeur nulle spécifie l'absence de maximum. La valeur sera renvoyée dans l'emplacement adressé par ce pointeur. Le pointeur vers l'information de comptage renvoyée doit être valide.

Extensions de consultation (*look_up_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Destinataire en sortie (*recipient_out*), type destinataire

Pointeur vers un tableau d'une ou de plusieurs structures de destinataire allouées par la fonction **cmc_look_up()**. La structure peut être utilisée dans des appels ultérieurs à la fonction **cmc_send()**. Le pointeur renvoyé est passé à la fonction **cmc_free()** afin de libérer toutes les structures de destinataire.

Comptage (*count*), type uint32

Spécifie le nombre de noms effectivement renvoyés. Une valeur nulle est renvoyée, ainsi que le code erreur CMC_E_RECIPIENT_NOT_FOUND, si aucun nom correspondant au critère n'est trouvé.

Si le nombre de noms renvoyés est inférieur au nombre de noms trouvés, le fanion CMC_RECIP_LIST_TRUNCATED sera positionné dans la dernière structure de destinataire du tableau, désignée par le fanion CMC_RECIP_LAST_ELEMENT.

Extensions de consultation (*look_up_extensions*), type extension

Si les extensions de sortie sont passées à la fonction dans la liste d'extensions, les résultats du service seront disponibles dans l'extension. Pour plus de détails, voir la définition de la structure d'extensions.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Administration

Les fonctions d'administration définies dans la présente Recommandation sont la libération, l'ouverture de session, la fermeture de session et l'interrogation de la configuration.

6.1.4.1 Libération (*Free*)

NOM

Libération – Libérer la mémoire allouée par le service de messagerie.

RÉSUMÉ

```
#include <xcmc.h>

CMC_return_code
cmc_free(
    CMC_buffer       memory
);
```

DESCRIPTION

Cette fonction libère la mémoire allouée par le service de messagerie. Le pointeur vers la zone *mémoire* ne sera plus valide après l'appel et ne doit pas être utilisé ultérieurement. Lorsqu'une fonction CMC alloue et renvoie vers l'application une zone tampon, l'application libérera la mémoire en question lorsque l'appel n'en aura plus besoin.

Si une fonction CMC renvoie un pointeur de base vers une structure complexe contenant plusieurs niveaux de pointeurs, l'application appellera cette fonction en utilisant uniquement le pointeur de base pour libérer la totalité de la structure ou du tableau de structures. Les fonctions CMC qui renvoient des structures de ce type sont les suivantes:

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()

Le comportement de la fonction **cmc_free()** n'est pas défini lorsqu'elle est appelée avec un pointeur vers un bloc mémoire qui a déjà été libéré ou avec un pointeur contenu dans une structure renvoyée par la mise en œuvre CMC.

Les extensions spécifiées pour une fonction peuvent, dans certains cas, être une combinaison d'extensions en entrée et en sortie. Dans un tel cas, les extensions en sortie doivent être libérées l'une après l'autre en utilisant la fonction **cmc_free()**. L'Annexe C, "Exemples de programmation" fournit un exemple de ce type.

ARGUMENTS

Mémoire (*memory*), type tampon

Pointeur vers une zone mémoire allouée par le service de messagerie. Une valeur NULL sera ignorée.

RÉSULTATS

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

CMC_E_FAILURE
CMC_E_INVALID_MEMORY

6.1.4.2 Fermeture de session (*Logoff*)

NOM

Fermeture de session – Fermeture d'une session avec le service CMC.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction permet à l'application appelante de fermer une session établie avec le service CMC. Les utilisateurs du service CMC doivent appeler la fonction **cmc_free()** pour tous les pointeurs de mémoire alloués en cours de session par le service avant d'appeler la fonction **cmc_logoff()**. Le non-respect de cette prescription peut conduire à des pertes de contrôle de mémoire ou à un comportement non défini lors d'accès ultérieurs à ces pointeurs après la fin de la session.

NOTE – Certaines mises en œuvre CMC peuvent faire le choix de libérer tous les pointeurs créés pour la session lorsque la fonction **cmc_logoff()** est appelée. La prise en charge de ce nettoyage en fin de session est toutefois optionnelle pour le service CMC.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie. Il devient non-valide comme résultat de cet appel.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Identificateur d'interface utilisateur (*ui_id*), type identificateur d'interface utilisateur

Identificateur d'une interface utilisateur (par exemple, le descripteur opaque de la fenêtre parente pour l'application appelante) utilisée pour résoudre toute question qui pourrait entraîner une erreur si elle faisait défaut.

Ignoré si l'interface utilisateur n'est pas prise en charge par la mise en œuvre CMC.

Fanions de fermeture (*logoff_flags*) ,type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés.

CMC_ERROR_UI_ALLOWED
CMC_LOGOFF_UI_ALLOWED

CMC_ERROR_UI_ALLOWED – Positionné si la fonction peut afficher une boîte de dialogue lorsqu'elle rencontre une erreur pouvant être récupérée. La fonction n'affichera pas de boîte de dialogue et renverra simplement un code erreur si le fanion n'est pas positionné.

CMC_LOGOFF_UI_ALLOWED – Positionné si la fonction peut afficher une interface utilisateur lors de fermeture de la session de l'utilisateur en l'absence d'erreurs.

Extensions de fermeture de session (*logoff_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions de fermeture de session (*logoff_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des erreurs énumérées ci-dessous.

ERREURS

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 Ouverture de session (*Logon*)

NOM

Ouverture de session – Ouvrir une session avec le service CMC.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction permet à l'application appelante d'ouvrir une session avec le service CMC.

Cette fonction renvoie un identificateur de session que l'application peut utiliser dans des appels ultérieurs à des fonctions CMC.

ARGUMENTS

Service (*service*), type chaîne de caractères

Chaîne de caractères indiquant l'emplacement du service de messagerie sous-jacent, par exemple le chemin d'accès vers un stockage de message ou le nom de nœud d'un serveur distant. Cette valeur peut être NULL lorsque le service de messagerie sous-jacent n'exige pas de nom de service. Ceci peut être nécessaire dans certaines mises en œuvre et ignoré par d'autres.

Le service de messagerie sous-jacent à une mise en œuvre CMC ou à une installation d'une mise en œuvre peut, d'une manière optionnelle, prendre en charge simultanément plusieurs protocoles de messagerie. Si des protocoles multiples sont pris en charge par une mise en œuvre, un protocole particulier est choisi par le service sur la base de critères tels que:

– la configuration de la prise en charge du protocole;
– la disponibilité dynamique de la prise en charge du protocole;

– les capacités du destinataire (si celles-ci sont connues);
– l'analyse du format ou de la notation utilisée pour l'adresse;
– d'autres critères propres au système.

Ces critères peuvent être appliqués message par message ou destinataire par destinataire.

Utilisateur (*user*), type chaîne de caractères

Chaîne de caractères identifiant l'utilisateur CMC, par exemple un nom d'utilisateur d'un service de messagerie. Cette valeur peut être NULL si le service de messagerie sous-jacent n'exige pas de nom d'utilisateur ou si l'interface utilisateur est autorisée.

Mot de passe (*password*), type chaîne de caractères

Chaîne de caractères contenant le mot de passe exigé pour accéder au service CMC. Cette valeur peut être NULL si le service de messagerie sous-jacent n'exige pas de mot de passe ou si l'interface utilisateur est autorisée.

Jeu de caractères (*character_set*), type identificateur d'objet

Identificateur d'objet indiquant le jeu de caractères utilisé par l'appelant CMC. Les valeurs possibles disponibles pour le client sont renvoyées par la mise en œuvre CMC au moyen de la fonction **cmc_query_configuration()**. Le client peut passer la valeur NULL, auquel cas le jeu de caractères est déterminé par le service CMC.

Identificateur d'interface utilisateur (*ui_id*), type identificateur d'interface utilisateur

Identificateur d'une interface utilisateur (par exemple, le descripteur opaque de la fenêtre parente pour l'application appelante) utilisée pour résoudre toute question qui pourrait entraîner une erreur si elle faisait défaut, ou utilisée pour un dialogue d'ouverture de session si cela est autorisé et requis.

Ignoré si l'interface utilisateur n'est pas prise en charge par la mise en œuvre CMC.

Version CMC de l'appelant (*caller_cmc_version*), type uint16

Numéro de version de l'application CMC appelante, multiplié par 100. Par exemple, la version 1.01 est spécifiée par le nombre 101. La version de la présente Recommandation est 2.00, représentée par le nombre 200.

Fanions d'ouverture de session (*logon_flags*), type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés.

CMC_LOGON_UI_ALLOWED
CMC_ERROR_UI_ALLOWED
CMC_COUNTED_STRING_TYPE
CMC_FULL_CMC

CMC_LOGON_UI_ALLOWED – Positionné si la fonction doit afficher une boîte de dialogue pour demander, si nécessaire, une ouverture de session. Si le fanion n'est pas positionné, aucune boîte de dialogue n'est affichée et une erreur sera renvoyée si une information suffisante n'a pas été fournie.

CMC_ERROR_UI_ALLOWED – Positionné si la fonction peut afficher une boîte de dialogue lorsqu'elle rencontre une erreur pouvant être récupérée. La fonction n'affichera pas de boîte de dialogue et renverra simplement un code erreur si le fanion n'est pas positionné.

CMC_COUNTED_STRING_TYPE – Positionné si le type de la chaîne de caractères utilisée dans le message est une chaîne de caractères avec comptage. Il est fait l'hypothèse que la chaîne de caractères se termine par un caractère nul dans le cas contraire.

CMC_FULL_CMC – Positionné si l'application demande les fonctions de l'interface CMC complète. Si ce fanion n'est pas positionné, ceci signifie que l'application accède à l'interface CMC simple. L'interface CMC complète n'est disponible que si l'appelant spécifie une version CMC de l'appelant supérieure ou égale à 200.

Extensions d'ouverture de session (*logon_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

L'application peut déterminer, en utilisant des extensions, quelles sont les extensions disponibles et positionner celles des extensions de données qui seront actives au cours de la session. L'extension CMC_X_COM_SUPPORT_EXT est utilisée à cet effet. Toute mise en œuvre CMC prenant en charge des extensions doit prendre en charge cette extension. Prière de se référer, pour plus de détails, au B.2 traitant des extensions.

RÉSULTATS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Extensions d'ouverture de session (*logon_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Interrogation de configuration (*Query Configuration*)

NOM

Interrogation de configuration – Recherche une information concernant la configuration CMC installée.

RÉSUMÉ

```
#include <xcmc.h>

CMC_return_code
cmc_query_configuration(
    CMC_session_id        session,
    CMC_enum              item,
    CMC_buffer            reference,
    CMC_extension         *config_extensions
);
```

DESCRIPTION

Cette fonction interroge la configuration de la mise en œuvre sous-jacente et renvoie l'information demandée, en allouant la mémoire si nécessaire.

NOTE – La configuration ne peut pas être modifiée par l'interface CMC et les formats des fichiers de configuration sous-jacents sont propres à la mise en œuvre.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Les identificateurs opaques de session sont créés par un appel à une fonction d'ouverture de session et invalidés par un appel à une fonction de fermeture de session.

L'identificateur de session peut avoir la valeur NULL pour indiquer qu'il n'existe pas de session et qu'une information indépendante de la session doit être renvoyée. Une information d'ouverture de session par défaut sera fournie dans ce cas.

Une information dépendante de la session sera renvoyée si l'identificateur de session est positionné sur une valeur valide.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Article (*item*), type énumération

Cet argument indique quelle est l'information de configuration devant être renvoyée. Les valeurs possibles sont les suivantes:

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 – L'argument de référence doit être un pointeur vers un tableau d'identificateurs d'objet CMC. Un pointeur vers le tableau de chaînes de caractères contenant des identificateurs de jeu de caractères sera renvoyé dans ce cas. Le tableau sera terminé par un identificateur d'objet CMC de valeur NULL. Le premier identificateur de jeu de caractères du tableau est le jeu de caractères utilisé par défaut si l'appelant n'en définit pas un d'une manière explicite. Le paragraphe propre à la plate-forme au B.2.4 contient les valeurs d'identificateur d'objet définis pour les jeux de caractères usuels. Ce pointeur vers le tableau doit être libéré en utilisant la fonction **cmc_free()**. Cet identificateur d'objet est utilisé à l'ouverture de la session par l'appelant pour spécifier à la mise en œuvre l'utilisation d'un jeu de caractères différent du jeu par défaut.

CMC_CONFIG_LINE_TERM – L'argument de référence doit être un pointeur vers une variable CMC énumérée qui sera positionnée sur la valeur CMC_LINE_TERM_CRLF si le délimiteur de ligne est un retour chariot suivi d'un saut de ligne, CMC_LINE_TERM_LF si le délimiteur de ligne est un saut de ligne ou CMC_LINE_TERM_CR si le délimiteur de ligne est un retour chariot.

CMC_CONFIG_DEFAULT_SERVICE – L'argument de référence doit être un pointeur vers une chaîne de caractères CMC dans laquelle sera placé un pointeur vers le nom par défaut du service, s'il est disponible, suivi d'un caractère nul. Un pointeur de valeur NULL sera fourni si aucun nom de service par défaut n'est disponible. Ce pointeur doit être libéré en utilisant la fonction **cmc_free()**. Cette chaîne peut être utilisée, avec celle renvoyée par l'argument CMC_CONFIG_DEFAULT_USER, comme valeur par défaut lorsque l'utilisateur est interrogé au sujet du nom de service, du nom de l'utilisateur et du mot de passe. Ces informations seront renvoyées en utilisant le jeu de caractères par défaut de la mise en œuvre.

CMC_CONFIG_DEFAULT_USER – L'argument de référence doit être un pointeur vers une chaîne de caractères CMC dans laquelle sera placé un pointeur vers le nom par défaut de l'utilisateur, s'il est disponible, suivi d'un caractère nul. Un pointeur de valeur NULL sera fourni si aucun nom d'utilisateur par défaut n'est disponible. Ce pointeur doit être libéré en utilisant la fonction **cmc_free()**. Cette chaîne peut être utilisée, avec celle renvoyée par l'argument CMC_CONFIG_DEFAULT_SERVICE, comme valeur par défaut lorsque l'utilisateur est interrogé au sujet du nom de fournisseur, du nom de l'utilisateur et du mot de passe. Ces informations seront renvoyées en utilisant le jeu de caractères par défaut de la mise en œuvre.

CMC_CONFIG_REQ_PASSWORD – L'argument de référence doit être un pointeur vers une variable CMC énumérée, qui sera positionnée sur la valeur CMC_REQUIRED_NO si le mot de passe n'est pas exigé pour l'ouverture de la session, CMC_REQUIRED_OPT si le mot de passe est optionnel pour l'ouverture de la session, ou CMC_REQUIRED_YES si le mot de passe est exigé pour l'ouverture de la session.

CMC_CONFIG_REQ_SERVICE – L'argument de référence doit être un pointeur vers une variable CMC énumérée, qui sera positionnée sur la valeur CMC_REQUIRED_NO, si le nom du service n'est pas exigé pour l'ouverture de la session, CMC_REQUIRED_OPT si le nom du service est optionnel pour l'ouverture de la session ou CMC_REQUIRED_YES, si le nom du service est exigé pour l'ouverture de la session.

CMC_CONFIG_REQ_USER – L'argument de référence doit être un pointeur vers une variable CMC énumérée, qui sera positionnée sur la valeur CMC_REQUIRED_NO, si le nom de l'utilisateur n'est pas exigé pour l'ouverture de la session, sur la valeur CMC_REQUIRED_OPT, si le nom de l'utilisateur est optionnel pour l'ouverture de la session, ou sur la valeur CMC_REQUIRED_YES, si le nom de l'utilisateur est exigé pour l'ouverture de la session.

CMC_CONFIG_UI_AVAIL – L'argument de référence doit être un pointeur vers une variable CMC booléenne, qui sera positionnée sur "Vrai" s'il existe une interface utilisateur fournie par la mise en œuvre CMC.

CMC_CONFIG_SUP_NOMKMSGREAD – L'argument de référence doit être un pointeur vers une variable CMC booléenne, qui sera positionnée sur "Vrai" si le fanion CMC_DO_NOT_MARK_AS_READ est pris en charge par la fonction **cmc_read()**.

CMC_CONFIG_SUP_COUNTED_STR – L'argument de référence doit être un pointeur vers une variable CMC booléenne, qui sera positionnée sur "Vrai" si le fanion CMC_COUNTED_STRING_TYPE est pris en charge pendant l'ouverture de session.

CMC_CONFIG_VER_IMPLEM – L'argument de référence doit être un pointeur vers une variable CMC de type uint16, qui sera positionnée sur le numéro de version de la mise en œuvre multiplié par 100. La version 1.01 reverra, par exemple, la valeur 101.

CMC_CONFIG_VER_SPEC – L'argument de référence doit être un pointeur vers une variable CMC de type uint16, qui sera positionnée sur la valeur de la version de la spécification CMC utilisée par la mise en œuvre, multipliée par 100. La version 1.00 reverra, par exemple, la valeur 100.

Extensions de configuration (*config_extensions***), type extension**

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

L'application peut déterminer quelles sont les extensions disponibles en utilisant des extensions. L'extension CMC_X_COM_SUPPORT_EXT est utilisée à ce effet. Toute mise en œuvre CMC prenant en charge des extensions devra prendre en charge cette extension. Prière de se référer, pour plus de détails, au B.2 traitant des extensions.

RÉSULTATS

Référence (*reference***), type tampon**

Cet argument pointe vers un tampon destiné à recueillir l'information de configuration. La zone mémoire correspondant au nombre d'octets impliqués par la valeur du paramètre de l'article doit appartenir à l'appelant et doit être modifiable. Le type de la variable ou du tampon dépend de l'argument de l'article.

Extensions de configuration (*config_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Fonctions CMC complètes

L'interface CMC complète se compose d'un ensemble étendu de fonctions destinées à fournir des capacités tributaires de la messagerie à des applications utilisant la messagerie. Le Tableau 15 fournit la liste des fonctions de l'interface CMC complète.

Tableau 15/X.446 – Interface CMC complète

Fonction	Description
Fonctions d'administration	
libérer	voir la description de l'interface CMC simple
fin de session	voir la description de l'interface CMC simple
ouverture de session	voir la description de l'interface CMC simple
Fonction de liaison	
lier une mise en œuvre	créer et renvoyer une table de distribution
délier une mise en œuvre	libérer toute donnée associée à un appel de la fonction **cmc_bind_implementation()** de liaison pour une mise en œuvre CMC donnée
Fonctions de composition	
copier un objet	copier un objet source vers un objet conteneur
ajouter des propriétés	ajouter ou modifier un ensemble de propriétés pour un objet conteneur
confier un objet	confier un objet à des fins de stockage persistant dans un objet conteneur
copier un descripteur opaque d'objet	copier un descripteur opaque d'objet
supprimer des objets	supprimer les objets spécifiés dans un conteneur
supprimer des propriétés	supprimer les propriétés spécifiées dans un objet
ouvrir un descripteur opaque d'objet	ouvrir un descripteur opaque d'objet
restaurer un objet	restaurer les données d'un objet à partir d'un fichier
sauvegarder un objet	sauvegarder les données d'un objet vers un fichier
Fonctions d'énumération	
obtenir la dernière erreur	renvoyer, dans un texte en langue locale, un message d'erreur concernant la dernière erreur survenue pour l'objet
obtenir le descripteur opaque racine	renvoyer un descripteur opaque du conteneur constituant la racine de la hiérarchie du modèle d'objets pour la session
liste des propriétés contenues	liste des propriétés dans un objet conteneur

Tableau 15/X.446 – Interface CMC complète *(fin)*

Fonction	Description
Fonctions d'énumération *(suite)*	
liste du nombre de concordances	liste du nombre d'objets correspondant à un critère donné dans un conteneur
liste d'objets	liste des objets dans un objet conteneur
liste de propriétés	liste des propriétés d'un objet
ouverture de curseur	ouverture d'un curseur pour un objet conteneur
lecture de curseur	lire la position fractionnaire actuelle d'un curseur
lecture de propriétés	lire l'information de contenu d'un ensemble de propriétés
lecture de coût propriété	lire le coût relatif lié à la lecture d'un ensemble de propriétés
mise à jour de la position du curseur	mettre à jour la position fractionnaire actuelle d'un curseur
mise à jour de la position du curseur avec placement	mettre à jour la position actuelle d'un curseur par rapport à un objet dans le conteneur
Fonctions de notification d'événement	
superviser un événement	supervision d'un élément d'un service de messagerie
enregistrer un événement	enregistrer un événement que l'appelant souhaite superviser
résilier un événement	résilier l'enregistrement d'un événement que l'appelant ne souhaite plus superviser
appel de rappel automatique	appeler une ou plusieurs fonctions enregistrées, lorsque l'événement s'est manifesté
Fonctions de messagerie	
création d'un message dérivé	créer un message à des fins de réexpédition ou de réponse concernant un message donné
émission d'un objet message	dépôt d'un message pour émission auprès de l'agent MTA
Fonctions de traitement de nom	
identificateur vers nom	convertir un identificateur de propriété vers un nom de propriété
nom vers identificateur	convertir un nom de propriété vers un identificateur de propriété
Fonction de flux	
exportation de flux	exporter un flux vers un fichier de données
importation de fichier vers un flux	importer des données d'un fichier vers un flux
ouverture de flux	ouvrir une propriété pour des opérations de lecture ou d'écriture de flux
lecture de flux	lire un flux d'information de contenu
recherche dans un flux	se positionner sur un emplacement dans un flux de contenu d'information
écriture de flux	écrire un flux d'information de contenu

Les pages de manuel concernant ces fonctions sont données ci-après.

6.2.1 Fonctions de liaison

Les fonctions de liaison permettent à une mise en œuvre de créer et de renvoyer une table de distribution, ainsi que de libérer ultérieurement toute donnée associée à la fonction de mise en œuvre de liaison.

6.2.1.1 Lier la mise en œuvre (*Bind Implementation*)

NOM

Lier la mise en œuvre – Créer et renvoyer une table de distribution.

RÉSUMÉ

```
#include <xcmc.h>

CMC_return_code
cmc_bind_implementation(
    CMC_guid                    implementation_name,
    CMC_dispatch_table          **dispatch_table,
    CMC_extension               *cmc_bind_implementation_extensions
);
```

DESCRIPTION

Cette fonction crée et remplit, au profit de l'appelant, une table de distribution d'adresses de fonctions CMC. La fonction doit être prise en charge par le gestionnaire CMC et la mise en œuvre CMC. Des tâches administratives locales peuvent être exécutées à cet instant par le gestionnaire CMC, par la mise en œuvre CMC ou par les deux.

ARGUMENTS

Nom de mise en œuvre (*implementation_name***), type GUID**

Identificateur global non-ambigu représentant une mise en œuvre CMC spécifique. La distinction entre versions différentes d'une même mise en œuvre CMC doit être faite au moyen de l'indicateur GUID afin de permettre la coexistence des différentes versions.

Extensions de liaison de mise en œuvre (*bind_implementation_extensions***), type extension**

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Table de distribution (*dispatch_table***), type table de distribution**

Adresse de la table de distribution de la mise en œuvre CMC. Cette table est allouée par la mise en œuvre CMC et doit être libérée en utilisant la fonction **cmc_free()**.

Extensions de liaison de mise en œuvre (*bind_implementation_extensions***), type extension**

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code***)**

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Délier la mise en œuvre (*Unbind Implementation*)

NOM

Délier la mise en œuvre – Libère toutes les données associées au moyen de la fonction **cmc_bind_implementation()** pour une mise en œuvre CMC spécifique.

RÉSUMÉ

```
#include <xcmc.h>

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

DESCRIPTION

Cette fonction libère et délie toute donnée associée à une liaison avec une mise en œuvre CMC spécifique. Des tâches administratives locales peuvent être exécutées à cet instant par le gestionnaire CMC, la mise en œuvre CMC ou les deux.

ARGUMENTS

Implementation Name (*implementation_name***), type GUID**

Identificateur global non-ambigu représentant une mise en œuvre CMC spécifique devant être déliée de l'application ou du gestionnaire CMC.

Extensions de liaison de mise en œuvre (*bind_implementation_extensions***), type extension**

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions de suppression de liaison de mise en œuvre (*unbind_implementation_extensions***), type extension**

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code***)**

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Fonctions de composition

Les fonctions de composition fournissent la capacité de création et de manipulation d'objets CMC et de propriétés d'objet.

6.2.2.1 Ajout de propriétés (*Add_Properties*)

NOM

Ajout de propriétés – Ajoute un ensemble de propriétés à un objet.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction ajoute un ensemble de propriétés à un objet.

La propriété sera remplacée si elle existe déjà. Elle sera ajoutée si elle n'existe pas. L'interface CMC ne définit pas d'ordre pour les propriétés dans un objet. L'ordre dans lequel de nouvelles propriétés sont ajoutées à un objet est propre à la mise en œuvre (par exemple, elles peuvent être ajoutées à la fin de l'objet).

Les propriétés ajoutées à un objet ne peuvent pas être confiées tant que la fonction **cmc_commit_object()** n'a pas été appelée.

ARGUMENTS

Objet (*object*), type poignée d'objet

Descripteur opaque d'objet.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Nombre de propriétés (*number_properties*), type uint32

Nombre de propriétés dans l'argument propriétés.

Propriétés (*properties*), type propriété

Pointeur vers un tableau de structures de propriétés devant être ajoutées à l'objet.

Extensions d'ajout de propriétés (*add_properties_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'ajout de propriétés (*add_properties_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Confier un objet (*Commit Object*)

NOM

Confier un objet – Confie un objet pour un stockage persistant dans un objet conteneur.

RÉSUMÉ

```
#include <xcmc.h>

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

DESCRIPTION

Cette fonction confie un objet pour un stockage persistant dans un conteneur.

Si l'objet est confié au conteneur de messages de la boîte aux lettres en sortie, l'action rend l'objet non-modifiable. Cet objet ne peut être que supprimé ou copié.

Une fois confié à la boîte aux lettres en sortie, un objet devient candidat à tout instant pour un dépôt de message. La mise en œuvre peut émettre le message à sa propre initiative. La fonction **cmc_send_message_object()** peut être utilisée pour provoquer l'émission immédiate d'un message.

Toutes les propriétés actuelles de l'objet source seront confiées à la mémoire persistante dans un objet conteneur. L'objet doit avoir été ajouté à un conteneur au moyen d'un appel préalable à la fonction **cmc_copy_object()**.

Les curseurs du conteneur restent valides après que l'objet a été confié au conteneur. Tout objet confié au conteneur après que le curseur a été ouvert ne figurera pas sur la liste fournie pour ce conteneur par la fonction **cmc_list_objects()**. Le code erreur CMC_E_UNSUPPORTED_ACTION est renvoyé si le conteneur associé à l'objet n'accepte pas d'objets confiés.

ARGUMENTS

Objet source (*source_object*), type poignée d'objet

Descripteur opaque pour l'objet source à confier au stockage persistant du conteneur.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Extensions d'objet confié (*commit_object_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'objet confié (*commit_object_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des erreurs énumérées ci-dessous.

ERREURS

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 Copier un objet (*Copy Object*)

NOM

Copier un objet – Copie un objet source dans un objet conteneur.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction copie un objet source vers un objet conteneur spécifié. Si l'objet source est un conteneur, la fonction de copie d'objet copiera d'une manière récursive toutes les propriétés et l'objet contenu.

Toutes les propriétés actuelles de l'objet seront sauvegardées avec l'objet lui-même dans l'objet conteneur. La fonction ajoute, dans le conteneur spécifié, un nouvel objet contenant toutes les propriétés de l'objet source. Un descripteur opaque est renvoyé pour le nouvel objet dans le conteneur. L'objet source et son contenu ne sont pas modifiés. Le nouvel objet doit être confié à l'objet conteneur au moyen de la fonction **cmc_commit_object()** avant qu'il ne devienne un objet persistant au sein de l'objet conteneur.

Les curseurs du conteneur restent valides après que l'objet a été confié au conteneur. Tout objet ajouté au conteneur après que le curseur a été ouvert ne figurera pas sur la liste fournie pour ce conteneur par la fonction **cmc_list_objects()**. Le code erreur CMC_E_UNSUPPORTED_ACTION est renvoyé si le conteneur associé à l'objet est accessible en lecture seulement.

ARGUMENTS

Conteneur (*container*), type poignée d'objet

Descripteur opaque d'un objet conteneur.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Objet source (*source_object*), type poignée d'objet

Descripteur opaque de l'objet source.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Extensions de copie d'objet (*copy_object_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Nouvel objet (*new_object*), type poignée d'objet

Descripteur opaque du nouvel objet.

Extensions de copie d'objet (*copy_object_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des erreurs énumérées ci-dessous.

ERREURS

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 Copy Object Handle

NOM

Copie de descripteur opaque d'objet – Copie un descripteur opaque d'objet.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction copie un descripteur opaque d'objet. Il n'est pas créé de copie de l'objet, mais le nouveau descripteur opaque d'objet fera référence au contenu original de l'information à laquelle le descripteur opaque de l'objet source fait référence. Les descripteurs opaques de curseur ne peuvent pas être copiés.

Cette fonction fournit une méthode simple pour copier un descripteur opaque d'objet à partir d'un tableau de descripteurs opaques d'objet renvoyé par un autre appel de fonction CMC. La fonction **cmc_free()** ne libérera pas le contenu d'information auquel fait référence le nouveau descripteur opaque d'objet. La mise en œuvre ne libérera la zone mémoire associée que lorsque la dernière référence a été supprimée au moyen de la fonction **cmc_free()**, en utilisant le dernier descripteur opaque d'objet qui fait référence à l'information de contenu.

ARGUMENTS

Descripteur opaque source (*source_handle*), type poignée d'objet

Descripteur opaque de l'objet source à copier.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Extensions de copie de descripteur opaque d'objet (*copy_object_handle_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

New Handle (*new_handle*), type poignée d'objet

Nouveau descripteur opaque pour l'objet.

Extensions de copie de descripteur opaque d'objet (*copy_object_handle_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des erreurs énumérées ci-dessous.

ERREURS

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 Suppression d'objets (*Delete Objects*)

NOM

Suppression d'objets – Supprime les objets spécifiés.

RÉSUMÉ

```
#include <xcmc.h>

CMC_return_code
cmc_delete_objects(
    CMC_uint32              number_objects,
    CMC_object_handle       *object,
    CMC_extension           *delete_objects_extensions
);
```

DESCRIPTION

Cette fonction supprime les objets spécifiés. La fonction de suppression d'objets effectue une suppression récursive dans laquelle tous les objets spécifiés et tout objet contenu sont supprimés. Les descripteurs opaques d'objet ne sont plus valides après le retour de la fonction.

ARGUMENTS

Nombre d'objets (*number_objects*), type uint32

Nombre d'objets dans le paramètre **objets**.

Objets (*objects*), type poignée d'objet

Pointeur vers un tableau de descripteurs opaques des objets à supprimer.

Un code erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyé si l'un des descripteurs opaques d'objet n'est pas valide.

Extensions d'objets à supprimer (*delete_objects_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'objets à supprimer (*delete_objects_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Suppression de propriétés (*Delete Properties*)

NOM

Suppression de propriétés – Supprime l'ensemble de propriétés spécifié au sein de l'objet.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction supprime les propriétés spécifiées au sein de l'objet.

ARGUMENTS

Objet (*object*), type poignée d'objet

Descripteur opaque de l'objet.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Nombre de propriétés (*number_properties*), type uint32

Nombre de propriétés dans l'argument **propriétés**.

Identificateurs de propriété (*property_ids*), type identificateur

Pointeur vers un tableau d'identificateurs non-ambigus des propriétés à supprimer au sein de l'objet.

Extensions de suppression de propriétés (*delete_properties_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions de suppression de propriétés (*delete_properties_extensions***), type extension**

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code***)**

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des erreurs énumérées ci-dessous.

ERREURS

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 Ouverture de descripteur opaque d'objet (*Open Object Handle*)

NOM

Ouverture de descripteur opaque d'objet – Crée un nouveau descripteur opaque d'objet.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction crée un nouveau descripteur opaque d'objet. Le service alloue les ressources nécessaires au nouvel objet et renvoie le descripteur opaque qui lui est associé. Cet objet ne figure pas dans un conteneur tant qu'il n'y a pas été ajouté au moyen de la fonction **cmc_copy_object**(). L'information de contenu de cet objet n'existe pas tant que des propriétés n'y ont pas été ajoutées au moyen de la fonction **cmc_add_properties()**.

ARGUMENTS

Session (*session***), type identificateur de session**

Identificateur opaque de session représentant une session avec le service de messagerie.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Classe d'objets (*object_class***), type identificateur**

Identificateur de la classe d'objets.

Extensions d'ouverture de descripteur opaque d'objet (*open_object_handle_extensions***), type extension**

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Nouvel objet (*new_object*), type poignée d'objet

Descripteur opaque pour le nouvel objet. Le descripteur opaque de l'objet encapsule l'identificateur de session. Le descripteur opaque d'objet suffit pour faire référence à l'objet donné dans une session donnée.

Extensions d'ouverture de descripteur opaque d'objet (*open_object_handle_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Restauration d'objet

NOM

Restauration d'objet – Restaurer un objet à partir du système de fichiers.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction restaure un objet à partir d'un fichier. Elle fournit, par exemple, une méthode simple pour attacher un fichier à un message. Dans un tel cas, *l'objet restauré* représente un objet article de contenu nouvellement créé qui sera associé ultérieurement à un message en cours de composition. Cette fonction fournit, par ailleurs, une méthode pour extraire un message qui a été stocké précédemment dans le système de fichiers au moyen de la fonction **cmc_save_object()**. La représentation sur le disque des objets stockés dans le système de fichiers n'est pas définie, car elle peut varier d'un service de messagerie à l'autre. Il en résulte que les applications ne doivent pas, d'une manière générale, se fier à la possibilité d'importer des objets qui ont été exportés par d'autres systèmes de messagerie. Cette restriction ne s'applique toutefois pas au cas de l'objet attachement d'un message.

Dans le cas d'un objet article de contenu de message, cette fonction a comme effet supplémentaire d'initialiser les valeurs des propriétés suivantes:

- nom de fichier;
- date de création;
- date dernière modification.

Les autres propriétés doivent être positionnées au moyen de la fonction **cmc_add_properties()**.

NOTE – L'article de contenu de fichier doit toujours être associé à un message en cours de composition afin de mener à bien le processus d'article de contenu.

ARGUMENTS

Conteneur (*container*), type poignée d'objet

Descripteur opaque de l'objet conteneur qui contiendra l'objet restauré.

Spécification de fichier (*file_specification*), type chaîne de caractères

Spécification complète de système de fichiers du fichier contenant les données de l'objet.

Fanions de restauration (*restore_flags*), type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés.

CMC_RESTORE_OBJECT_OVERWRITE

CMC_RESTORE_OBJECT_OVERWRITE – Positionné si la fonction doit remplacer un objet existant.

Extensions de restauration d'objet (*restore_object_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Objet restauré (*restored_object*), type poignée d'objet

Descripteur opaque de l'objet restauré.

Extensions de restauration d'objet (*restore_object_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Sauvegarde d'objet (*Save Object*)

NOM

Sauvegarde d'objet – Sauvegarde un objet vers le système de fichier.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction restaure un objet à partir d'un fichier. Elle fournit, par exemple, une méthode simple pour détacher les données d'un attachement et les placer dans le système de fichiers. Dans un tel cas, *l'objet* sauvegardé représente un objet attachement qui sera associé ultérieurement à un message. Cette fonction fournit, par ailleurs, un procédé pour stocker un message dans le système de fichiers. Les données de message peuvent être restaurées à partir du fichier au moyen d'un appel ultérieur à la fonction **cmc_restore_object()**. La représentation sur le disque des objets stockés dans le système de fichiers n'est pas définie, car elle peut varier d'un service de messagerie à l'autre. Il en résulte que les applications ne doivent pas, d'une manière générale, se fier à la possibilité d'importer des objets qui ont été exportés par d'autres systèmes de messagerie.

ARGUMENTS

Objet (*object*), type poignée d'objet

Descripteur opaque de l'objet (par exemple, un objet message ou attachement) dont les données sont exportées.

Spécification de fichier (*file_specification*), type chaîne de caractères

Spécification complète de système de fichiers du fichier qui contiendra les données de l'objet.

Fanions de sauvegarde (*save_flags*), type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés.

CMC_SAVE_OBJECT_OVERWRITE
CMC_SAVE_OBJECT_NOCREATE

CMC_ SAVE _OBJECT_OVERWRITE – Positionné si la fonction doit remplacer une spécification de fichier existante correspondante.

CMC_SAVE_OBJECT_NOCREATE – Positionné si la fonction ne doit pas créer une spécification de fichier avec ce nom si le fichier n'existe pas au préalable.

Extensions de sauvegarde d'objet (*save_object_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions de sauvegarde d'objet (*save_object_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Fonctions d'énumération

Les fonctions d'énumération fournissent la possibilité de lister, de lire et de mettre à jour les objets CMC et leurs propriétés.

6.2.3.1 **Obtenir la dernière erreur** (*Get Last Error*)

NOM

Obtenir la dernière erreur – Renvoie un message de texte d'erreur en langue locale pour la dernière erreur survenue pour l'objet.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

La fonction d'obtention de la dernière erreur est utilisée par les applications clientes pour extraire et présenter à l'utilisateur une chaîne de caractères en langue locale correspondant à la dernière erreur renvoyée par un appel de fonction fait pour cet objet. La mise en œuvre alloue la mémoire pour le tampon renvoyé, le client étant responsable de sa libération. Si la fonction renvoie un code erreur (non-nul), l'application ne doit pas appeler cette fonction une deuxième fois pour obtenir des diagnostics supplémentaires. Même si la fonction renvoie le code zéro, il est toujours possible qu'il n'y ait pas de chaîne de caractères disponible. Le code retour doit être nul pour que l'application puisse utiliser la chaîne de caractères descriptive. Les mises en œuvre de la fonction cmc_get_last_error doivent traduire en langue locale les messages d'erreur formulés dans la langue du système, ce qui nécessite que l'utilisateur positionne le jeu de caractères approprié lors de l'appel de la fonction cmc_logon.

Si les paramètres de la session et de l'objet ont tous deux la valeur NULL, ceci indique une demande d'obtention de la dernière erreur pour la fonction d'ouverture de session, dans laquelle la chaîne de caractères renvoyée pour l'erreur utilisera la page de code par défaut du système. Le code erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyé si l'identificateur de session est valide et que la valeur de l'objet n'est pas valide. Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide et que la valeur de l'objet est valide.

ARGUMENTS

Session (*session*), type identificateur de session

L'identificateur de session représente la session avec le service CMC durant laquelle l'erreur s'est manifestée.

Objet (*object*), type poignée d'objet

Descripteur opaque de l'objet (par exemple, un objet message ou attachement) dont les données sont renvoyées.

Extensions d'obtention de dernière erreur (*get_last_error_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Tampon d'erreur (*error_buffer*), type chaîne de caractères

Adresse du tampon dans lequel la mise en œuvre stocke la chaîne de caractères décrivant l'erreur. Ce tampon est alloué par le service et doit être libéré en utilisant la fonction **cmc_free()**.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Obtenir le descripteur opaque racine (*Get Root Handle*)

NOM

Obtenir le descripteur opaque racine – Obtenir un descripteur opaque du conteneur qui est la racine de la hiérarchie du modèle objet.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction renvoie un descripteur opaque du conteneur qui constitue la racine de la hiérarchie du modèle d'objet pour la session. Des appels successifs à cette fonction renverront le même descripteur opaque pendant la durée de vie de la session.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Extensions d'obtention du descripteur opaque racine (*get_root_handle_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Descripteur opaque d'objet racine (*root_object_handle*), type poignée d'objet

Descripteur opaque du conteneur qui constitue la racine de la hiérarchie du modèle de l'objet.

Extensions d'obtention du descripteur opaque racine (*get_root_handle_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Liste des propriétés contenues (*List Contained Properties*)

NOM

Liste des propriétés contenues – Donne la liste des propriétés contenues dans un objet conteneur.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction donne la liste des propriétés des objets dans un objet conteneur. Une des utilisations de cette fonction consiste à extraire une information résumée concernant les objets conteneurs (par exemple, composition d'un résumé des messages de la boîte aux lettres en entrée).

ARGUMENTS

Curseur (*cursor*), type poignée de curseur

Descripteur opaque du curseur de l'objet conteneur spécifié.

Nombre d'objets (*number_objects*), type sint32

Pointeur vers le nombre maximal de descripteurs opaques d'objets à renvoyer. Une valeur nulle indique l'absence de maximum. Une valeur négative indique que les descripteurs opaques du nombre d'objets spécifiés précédant la position actuelle du curseur doivent être renvoyés dans le même ordre de tri que celui spécifié par le curseur. Par exemple, si la position actuelle du curseur correspond au huitième objet du conteneur, une valeur égale à –3 générera une liste avec les descripteurs opaques des objets de rang cinq, six et sept et le curseur sera ensuite positionné sur le cinquième objet. Une valeur égale à 4 générera une liste des objets de rang huit, neuf, dix et onze, et le curseur sera positionné sur l'objet de rang douze.

Nombre de propriétés (*number_properties*), type uint32

Pointeur vers le nombre de propriétés de l'argument identificateurs de propriété.

Identificateurs de propriété (*property_ids*), type identificateur

Pointeur vers un tableau d'identificateurs de propriété correspondant aux propriétés devant figurer dans la liste.

Liste des extensions de propriétés contenues (*list_contained_properties_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Nombre d'objets (*number_objects*), type sint32

Nombre effectif des objets dont les propriétés sont renvoyées.

Nombre de propriétés (*number_properties*), type uint32

Nombre effectif des propriétés renvoyées pour chaque objet.

Propriétés (*properties*), type propriété

Adresse d'un tableau de tableaux des structures de propriétés pour l'objet conteneur dont la liste est faite. Chaque tableau est constitué de l'ensemble des propriétés associées à un objet donné. Le nombre d'éléments du tableau est fourni dans le résultat **nombre de propriétés**. Le nombre de tableaux est fourni dans le résultat **nombre d'objets**. La mémoire de ce tableau de tableaux est allouée par le service et doit être libérée en utilisant la fonction **cmc_free()**.

Liste des extensions de propriétés contenues (*list_contained_properties_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Liste du nombre de concordances (*List Number Matched*)

NOM

Liste du nombre de concordances – Donne la liste du nombre d'éléments correspondant aux contraintes spécifiées par un curseur.

RÉSUMÉ

```
#include <xcmc.h>

CMC_return_code
cmc_list_number_matched(
    CMC_cursor_handle   cursor,
    CMC_uint32    *number_matches,
    CMC_extension*list_number_matched_extensions
);
```

DESCRIPTION

Cette fonction renvoie le nombre d'éléments appartenant à un conteneur qui correspondent aux contraintes spécifiées par un curseur. Cette valeur peut être utilisée avec la position fractionnaire actuelle du curseur pour afficher un "ascenseur" sur une barre de défilement.

ARGUMENTS

Curseur (*cursor*), type poignée de curseur

Descripteur opaque de curseur.

Le code erreur CMC_E_INVALID_CURSOR_HANDLE est renvoyé si le descripteur opaque n'est pas valide.

Liste du nombre d'extensions en correspondance (*list_number_matched_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Nombre de correspondance (*number_matches*), type uint32

Nombre d'éléments appartenant à un conteneur qui correspondent aux contraintes spécifiées par le curseur. Une valeur nulle indique qu'aucun élément ne satisfait aux contraintes spécifiées.

Liste du nombre d'extensions en correspondance (*list_number_matched_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Liste d'objets (*List Objects*)

NOM

Liste d'objets – Liste des éléments dans un conteneur.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction renvoie un pointeur vers un tableau de descripteurs opaques d'objet correspondant aux éléments se trouvant dans un conteneur. L'objet conteneur est référencé par un curseur ouvert par la fonction **cmc_open_cursor()**. Le curseur est mis à jour par le service, de sorte que des appels ultérieurs à cette fonction renverront des descripteurs opaques de fichier vers des éléments supplémentaires du conteneur sur la base de la position du curseur mise à jour.

ARGUMENTS

Curseur (*cursor*), type poignée de curseur

Descripteur opaque de curseur.

Le code erreur CMC_E_INVALID_CURSOR_HANDLE est renvoyé si le descripteur opaque de curseur n'est pas valide.

Nombre d'objets (*number_objects*), type sint32

Pointeur vers le nombre maximal de descripteurs opaques d'objets à renvoyer. Une valeur nulle indique l'absence de maximum. Une valeur négative indique que les descripteurs opaques du nombre d'objets spécifiés précédant la position actuelle du curseur doivent être renvoyés dans le même ordre de tri que celui spécifié par le curseur. Par exemple, si la position actuelle du curseur correspond au huitième objet du conteneur, une valeur égale –3 générera une liste avec les descripteurs opaques des objets de rang cinq, six et sept et le curseur sera ensuite positionné sur le huitième objet. Une valeur égale à 4 générera une liste des objets de rang huit, neuf, dix et onze, et le curseur sera positionné sur l'objet de rang douze.

Extensions de liste d'objets (*list_objects_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Curseur (*cursor*), type poignée de curseur

Descripteur opaque du curseur de l'objet conteneur spécifié. Ce descripteur opaque peut avoir été mis à jour par le service.

Nombre d'objets (*number_objects*), type sint32

Nombre effectif de descripteurs opaques d'objet renvoyés. Une valeur nulle est renvoyée si aucun élément ne satisfait aux contraintes de curseur, ou si le conteneur d'objet est vide.

Objets (*objects*), type poignée d'objet

Adresse d'un tableau de descripteurs opaques correspondant aux éléments de l'objet conteneur. La mémoire pour ce tableau est allouée par le service et doit être libérée en utilisant la fonction **cmc_free()**.

NOTE – Les descripteurs opaques d'objet individuels de ce tableau ne sont plus valides lorsque ce tableau est libéré. L'application peut retenir certains de ces descripteurs opaques avant d'invoquer la fonction **cmc_free()** pour le tableau. L'utilisation d'un descripteur opaque libéré conduit à un comportement indéfini.

Extensions de liste d'objets (*list_objects_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Liste de propriétés (*List Properties*)

NOM

Liste de propriétés – Donne la liste des propriétés d'un objet.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction renvoie une liste des identificateurs non-ambigus des propriétés contenues dans un objet. Un appel ultérieur à la fonction **cmc_read_properties()** renverra le contenu d'information de la propriété pour les objets.

ARGUMENTS

Objet (*object*), type poignée d'objet

Descripteur opaque de l'objet pour lequel une liste doit être faite.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Nombre de propriétés (*number_properties*), type uint32

Pointeur vers le nombre maximal d'identificateurs de propriété à renvoyer. Une valeur nulle indique que toutes les propriétés doivent figurer sur la liste.

Extensions de liste de propriétés (*list_properties_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Nombre de propriétés (*number_properties*), type uint32

Nombre effectif de propriétés renvoyé.

Identificateurs de propriété (*property_ids*), type identificateur

Adresse d'un tableau d'identificateurs de propriété non-ambigus correspondant aux propriétés contenu dans l'objet. Ce tableau est alloué par le service et doit être libéré en utilisant la fonction **cmc_free()**.

Extensions de liste de propriétés (*list_properties_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 **Ouverture de curseur** (*Open Cursor*)

NOM

Ouverture de curseur – Ouvre un curseur pour un objet conteneur.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction renvoie un descripteur opaque pour un curseur vers l'objet conteneur spécifié. Ce curseur peut être défini afin d'effectuer des traitements sur le conteneur en utilisant des règles de tri spécifiées.

ARGUMENTS

Objet (*object*), type poignée d'objet

Descripteur opaque d'un objet conteneur.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Contrainte (*restriction*), type contrainte de curseur

Pointeur vers une structure de contrainte de curseur destinée à être utilisée pour l'énumération des éléments au sein du conteneur. Les mises en œuvre peuvent ne pas prendre en charge tous les types de contraintes.

Nombre de clés de tri (*number_sort_keys*), type uint32

Nombre d'éléments de l'argument clés de tri. Les règles de tri du curseur ne sont pas définies si le nombre de clés est nul.

Clés de tri (*sort_keys*), type clé de tri de curseur

Pointeur vers un tableau de clés de tri de curseur destinées à servir pour le tri du conteneur. Le premier élément constitue la première clé de tri, le deuxième élément la deuxième clé de tri, et ainsi de suite. La prise en charge de plus d'une clé de tri peut ne pas être fournie par toutes les mises en œuvre. Les objets qui ne possèdent pas les propriétés désignées par la clé de tri sont placées en dernier.

Extensions d'ouverture de curseur (*open_cursor_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Curseur (*cursor*), type poignée de curseur

Descripteur opaque du curseur de l'objet spécifié. Ce descripteur opaque est alloué par le service et doit être libéré en utilisant la fonction **cmc_free()**.

Extensions d'ouverture de curseur (*open_cursor_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Lecture de curseur (*Read Cursor*)

NOM

Lecture de curseur – Lecture de la position fractionnaire actuelle du curseur dans un objet conteneur.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction renvoie la position fractionnaire actuelle du curseur spécifié. Les valeurs renvoyées dans le numérateur et le dénominateur de la position conviennent à la détermination et à l'affichage d'un "ascenseur" sur une barre de défilement. Le maximum de la barre de défilement pourrait être déterminée per un propriété d'objet spécifique de conteneur.

ARGUMENTS

Curseur (*cursor*), type poignée de curseur

Descripteur opaque de curseur.

Le code erreur CMC_E_INVALID_CURSOR_HANDLE est renvoyé si le descripteur opaque n'est pas valide.

Extensions de lecture de curseur (*read_cursor_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Numérateur de la position (*position_numerator*), type uint32

Numérateur de la fraction donnant la position souhaitée pour le curseur. Le rapport entre le numérateur et le dénominateur de la position donne la position fractionnaire du curseur au sein des éléments de l'objet conteneur.

Dénominateur de la position (*position_denominator*), type uint32

Dénominateur de la fraction donnant la position souhaitée pour le curseur. Le rapport entre le numérateur et le dénominateur de la position donne la position fractionnaire du curseur au sein des éléments de l'objet conteneur.

Extensions de lecture de curseur (*read_cursor_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Lecture de propriétés (*Read Properties*)

NOM

Lecture de propriétés – Lit le contenu d'information associé à un ensemble de propriétés dans un objet.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction renvoie l'information de contenu des propriétés spécifiées dans un objet.

Si la propriété spécifiée ne figure pas dans l'objet, le type de propriété CMC_pv_return_code sera renvoyé dans la position de cette propriété dans l'argument **propriétés**, ainsi que la valeur de propriété du code retour CMC_E_PROPERTY_ID_NOT_FOUND. L'identificateur de propriété de cette propriété n'est pas spécifié par la présente Recommandation.

ARGUMENTS

Objet (*object*), type poignée d'objet

Descripteur opaque de l'objet listé.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Nombre de propriétés (*number_properties*), type uint32

Pointeur vers le nombre de propriétés de l'argument **identificateurs de propriété**.

Identificateurs de propriété (*property_ids*), type identificateur

Pointeur vers un tableau d'identificateurs de propriété correspondant aux propriétés à lire.

Extensions de lecture de propriété (*read_properties_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Nombre de propriétés (*number_properties*), type uint32

Nombre effectif de propriétés renvoyées. Une valeur nulle est renvoyée si aucune des propriétés spécifiées ne figurait dans l'objet.

Propriétés (*properties*), type propriété

Pointeur vers un tableau de structures de propriété renfermant l'information de contenu pour les propriétés qui ont été lues. Ce tableau est alloué par le service et doit être libérée en utilisant la fonction **cmc_free()**.

Extensions de lecture de propriété (*read_properties_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Lecture de coûts de propriété (*Read Property Costs*)

NOM

Lecture de coûts de propriété – Lit le coût de lecture de propriétés individuelles dans un objet.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction renvoie le coût relatif de lecture de propriétés individuelles dans un objet.

La prise en charge de cette fonction n'est pas obligatoire pour des mises en œuvre se conformant à la présente Recommandation. Les mises en œuvre ne prenant pas en charge cette fonction renverront le code erreur CMC_E_NOT_SUPPORTED.

La base de détermination du coût de lecture d'une propriété est propre à la mise en œuvre.

ARGUMENTS

Objet (*object*), type poignée d'objet

Descripteur opaque de l'objet à lister.

Une erreur CMC_E_INVALID_OBJECT_HANDLE est renvoyée si le descripteur opaque d'objet n'est pas valide.

Nombre de propriétés (*number_properties*), type uint32

Pointeur vers le nombre d'identificateurs de propriété figurant dans l'argument **identificateurs de propriété**.

Identificateurs de propriété (*property_ids*), type identificateur

Pointeur vers un tableau d'identificateurs de propriété non-ambigus correspondant aux propriétés à lire.

Extensions de lecture de coûts de propriété (*read_property_costs_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Nombre de propriétés (*number_properties*), type uint32

Nombre effectif de coûts de propriétés renvoyés. Une valeur nulle indique qu'aucun des coûts de propriété spécifiés n'a été renvoyé.

Coûts (*costs*), type énuméré

Pointeur vers un tableau de coûts relatifs de propriétés. Les coûts individuels correspondent un par un aux noms de propriété spécifiés. Les coûts relatifs peuvent prendre l'une des valeurs suivantes:

CMC_COST_UNDETERMINED
CMC_COST_NONE
CMC_COST_MINOR
CMC_COST_MAJOR

CMC_COST_UNDETERMINED – Le coût relatif de lecture de la propriété ne peut pas être déterminé.

CMC_COST_NONE – Il n'existe pas de coût relatif associé à la lecture de cette propriété.

CMC_COST_MINOR – Le coût associé à la lecture est relativement faible.

CMC_COST_MAJOR – Le coût associé à la lecture est relativement élevé.

La mémoire pour ce tableau est allouée par le service et doit être libérée en utilisant la fonction **cmc_free()**.

Extensions de lecture de coûts de propriété (*read_property_costs_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Mise à jour de la position du curseur (*Update Cursor Position*)

NOM

Mise à jour de la position du curseur – Met à jour la position fractionnaire actuelle du curseur spécifié au sein d'un objet conteneur.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction met à jour le curseur vers une position spécifiée au sein des éléments d'un conteneur objet. La position est déterminée par le rapport entre le numérateur et le dénominateur de la position.

ARGUMENTS

Curseur (*cursor*), type poignée de curseur

Descripteur opaque de curseur. Le code erreur CMC_E_INVALID_CURSOR_HANDLE est renvoyé si le descripteur opaque n'est pas valide.

Numérateur de la position (*position_numerator*), type uint32

Numérateur de la fraction donnant la position souhaitée pour le curseur. Le rapport entre le numérateur et le dénominateur de la position donne la position fractionnaire du curseur au sein des éléments de l'objet conteneur.

Dénominateur de la position (*position_denominator*), type uint32

Dénominateur de la fraction donnant la position souhaitée pour le curseur. Le rapport entre le numérateur et le dénominateur de la position donne la position fractionnaire du curseur au sein des éléments de l'objet conteneur.

Extensions de mise à jour de la position du curseur (*update_cursor_position_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions de mise à jour de la position du curseur (*update_cursor_position_extensions***), type extension**

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code***)**

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

CMC_E_INVALID_CURSOR_HANDLE

CMC_E_FAILURE

CMC_E_INVALID_PARAMETER

CMC_E_UNSUPPORTED_FUNCTION_EXT

6.2.3.12 Mise à jour de la position du curseur avec placement (*Update Cursor Position With Seed*)

NOM

Mise à jour de la position du curseur avec placement – Mise à jour de la position actuelle du curseur spécifié, au sein de l'objet conteneur, dans une position relative par rapport à un objet de placement.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction met à jour le curseur dans une position spécifiée au sein des éléments de l'objet conteneur. La position est déterminée par rapport à un objet de placement donné situé dans le conteneur.

ARGUMENTS

Curseur (*cursor***), type poignée de curseur**

Descripteur opaque de curseur. Le code erreur CMC_E_INVALID_CURSOR_HANDLE est renvoyé si le descripteur opaque n'est pas valide.

Placement (*seed***), type poignée d'objet**

Descripteur opaque de l'objet situé dans le conteneur, par rapport auquel la position du curseur doit être mise à jour.

Extensions de mise à jour de position de curseur avec placement (*update_cursor_position_with_seed_extensions***), type extension**

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions de mise à jour de position de curseur avec placement (*update_cursor_position_with_seed_extensions***), type extension**

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code***)**

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Fonctions de notification d'événements

Les fonctions de notification d'événement permettent à une mise en œuvre de superviser des événements d'enregistrement et de résiliation, ainsi que d'effectuer des rappels automatiques.

6.2.4.1 **Supervision d'événement** (*Check Event*)

NOM

Supervision d'événement – Supervision d'un événement de service de messagerie.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction supervise un événement associé au service de messagerie. Elle fournit une variante à l'enregistrement de rappels automatiques auprès de la mise en œuvre CMC, pour celles des applications qui préfèrent scruter d'une manière synchrone l'apparition d'événements, et permet également de fournir une notification d'événement pour des mises en œuvre ne prenant pas en charge le rappel automatique.

Un fanion est associé à tout événement. Il peut également exister des paramètres en entrée et en sortie associés à l'événement. Ces structures de données d'événement sont décrites pour le type de données de rappel automatique.

Si un événement ne s'est pas manifesté et que la temporisation minimale n'est pas nulle, la mise en œuvre attend l'apparition de l'événement pendant la temporisation spécifiée avant de revenir au programme appelant. La fonction effectue un retour immédiat si l'événement survient avant l'écoulement de la temporisation. La fonction renvoie le code erreur CMC_E_NO_EVENT si l'événement n'apparaît pas avant la fin de la temporisation.

La fonction peut se terminer d'une manière anticipée avant la détection de l'événement ou l'écoulement de la temporisation, dans des circonstances définies par la mise en œuvre et sans que cela soit considéré effectivement comme une erreur. La fonction renvoie dans ce cas le code retour CMC_E_FUNCTION_INTERRUPTED.

NOTE – D'autres erreurs peuvent également se manifester et provoquer le retour prématuré de cette fonction. Le code retour CMC_E_FUNCTION_INTERRUPTED n'est pas utilisé dans un tel cas, et le code erreur CMC adéquat est renvoyé.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Type d'événement (*event_type*), type événement

Masque binaire d'événements que l'appelant souhaite superviser. Les événements non spécifiés sont toujours passés avec une valeur nulle. Les événements non documentés sont réservés. La définition des événements CMC est donnée dans la description du type de données événement.

Temporisation minimale (*minimum_time out*), type uint32

Temps, exprimé en secondes, après lequel la fonction se termine, même si l'événement ne s'est pas produit.

Une valeur nulle a pour effet que la fonction effectue uniquement la supervision de l'événement et se termine immédiatement.

La valeur CMC_NO_TIMEOUT indique que la fonction doit attendre l'apparition de l'événement sans aucune limite de temps.

Si une valeur autre que CMC_NO_TIMEOUT est utilisée, le temps minimal passé effectivement dans cette fonction dépend de la mise en œuvre.

Donnée de supervision d'événement (*check_event_data*), type tampon

Pointeur vers une structure de données de supervision associée à l'événement. Prière de se référer au type de données de rappel automatique en ce qui concerne la structure spécifique des données de supervision. Le choix de faire réaliser l'allocation de la mémoire tampon par la mise en œuvre ou l'application est propre à l'événement et est indiqué dans la description du type de données.

Extensions de supervision d'événement (*check_event_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Données de rappel automatique (*callback_data*), type tampon

Adresse d'une structure de données de supervision associée à l'événement. Dans ce type d'appel, la structure est renvoyée directement à l'application plutôt que d'être envoyée à une fonction de rappel automatique. Prière de se référer à la description de la table de distribution de rappel automatique en ce qui concerne la structure spécifique des données de supervision. Le choix de réaliser l'allocation de la mémoire du tampon par la mise en œuvre ou l'application est propre à l'événement et indiqué dans la description du type de données.

Extensions de supervision d'événement (*check_event_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extension pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Enregistrement d'événement (*Register Event*)

NOM

Enregistrement d'événement – Enregistre des événements auxquels s'intéresse l'appelant.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction spécifie les événements du service de messagerie pour lesquels l'appelant souhaite recevoir une alerte.

L'appelant peut recevoir une notification de la part d'un événement soit au moyen d'une fonction de rappel automatique, soit en utilisant la fonction de supervision d'événement pour interroger d'une manière synchrone des événements pour lesquels il s'est enregistré. Les mises en œuvre CMC n'ont pas l'obligation de prendre en charge le rappel automatique.

Il peut également exister des paramètres en entrée et en sortie associés à un événement. Ces paramètres sont contenus dans les données clientes. La structure des données clientes pour les événements est indiquée dans le type de données de rappel automatique.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Type d'événement (*event_type*), type événement

Masque binaire d'événements que l'appelant souhaite superviser. Les événements non spécifiés sont toujours passés avec une valeur nulle. Les événements non documentés sont réservés. La définition des événements CMC est donnée dans la description du type de données événement.

Rappel automatique (*callback*), type rappel automatique

Procédure cliente devant être appelée par le service pour traiter l'activité de rappel automatique. Une valeur NULL indique qu'aucune fonction de rappel automatique n'est fournie et que l'événement doit être signalé par la fonction de supervision d'événement. Le code erreur CMC_E_CALLBACK_NOT_SUPPORTED est renvoyé si le rappel automatique n'est pas pris en charge par la mise en œuvre.

Données d'enregistrement (*register_data*), type tampon

Pointeur vers une structure de données d'enregistrement associée à l'événement. Prière de se référer au type de données de rappel automatique en ce qui concerne la structure spécifique des données d'enregistrement. Le choix de faire réaliser l'allocation de la mémoire du tampon par la mise en œuvre ou l'application est propre à l'événement et indiqué dans la description du type de données.

Extensions d'enregistrement d'événement (*register_event_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'enregistrement d'événement (*register_event_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extension pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Résiliation d'événement (*Unregister Event*)

NOM

Résiliation d'événement – Résilie l'enregistrement pour des événements au sujet desquels l'appelant n'est plus intéressé.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction spécifie les événements du service de messagerie pour lesquels l'appelant souhaite ne plus recevoir de notification.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie.

Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Type d'événement (*event_type*), type fanions

Masque binaire indiquant les événements que l'appelant ne souhaite plus superviser. Les événements non spécifiés sont toujours passés avec une valeur nulle. Les événements non documentés sont réservés. La définition des événements CMC est donnée dans la description du type de données événement.

Rappel automatique (*callback*), type rappel automatique

Procédure cliente devant être appelée par le service pour traiter l'activité de rappel automatique. Une valeur NULL indique qu'aucune fonction de rappel automatique n'est fournie. Le code erreur CMC_E_CALLBACK_NOT_SUPPORTED est renvoyé si le rappel automatique n'est pas pris en charge par la mise en œuvre.

Données de résiliation (*unregister_data*), type tampon

Pointeur vers une structure de données de résiliation pouvant être utilisée pour passer des données d'événement qui seront nécessaires à la fonction de rappel automatique afin de fournir un contexte pour annuler l'enregistrement. La structure des données de résiliation est indiquée dans la description du type de données rappel automatique.

Extensions d'événement de résiliation (*unregister_event_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'événement de résiliation (*unregister_event_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Appel de rappels automatiques (*Call Callbacks*)

NOM

Appel de rappel automatique – Appelle la ou les fonctions de rappel automatique enregistrées si l'événement s'est manifesté.

RÉSUMÉ

```
#include <xcmc.h>

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

DESCRIPTION

Cette fonction a pour effet de faire appeler, par le service de messagerie, les fonctions de rappel automatique enregistrées associées à l'événement ou aux événements de rappel automatique spécifiés. Le service de messagerie traitera tout événement de rappel automatique spécifié et appellera les fonctions de rappel automatique enregistrées s'il y a eu des modifications qui déclenchent le rappel automatique correspondant à cet événement. L'ordre dans lequel sont invoqués les rappels automatiques est propre à la mise en œuvre.

Cette fonction est utile dans des environnements où une mise en œuvre ne peut effectuer l'appel du programme à rappeler lorsque le code de la mise en œuvre est en cours d'exécution, c'est-à-dire dans le cas de mises en œuvre pour lesquelles les rappels automatiques peuvent être effectués uniquement comme effet de bord lors de l'appel d'une fonction CMC appartenant à la mise en œuvre.

La prise en charge de cette fonction est optionnelle pour une mise en œuvre se conformant à la spécification de l'interface CMC. Le code erreur CMC_E_NOT_SUPPORTED est renvoyé si la fonction n'est pas prise en charge.

ARGUMENTS

Session (*session*), type identificateur de session

Identificateur opaque de session représentant une session avec le service de messagerie. Les fonctions de rappel automatiques enregistrées pour cette session seront invoquées si un descripteur opaque est spécifié. Le code erreur CMC_E_INVALID_SESSION_ID est renvoyé si l'identificateur de session n'est pas valide.

Type d'événement (*event_type*), type événement

Masque binaire d'événements. Les événements non spécifiés sont toujours passés avec une valeur nulle. Les événements non documentés sont réservés. La définition des événements CMC est donnée dans la description du type de données événement.

Extensions de rappel automatique (*call_callbacks_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions de rappel automatique (*call_callbacks_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Fonctions de messagerie

Les fonctions de messagerie fournissent la capacité de créer des messages dérivés à partir d'autres messages, d'émettre des messages et d'attendre l'arrivée de nouveaux messages.

6.2.5.1 **Création d'objet message dérivé** (*Create Derived Message Object*)

NOM

Création d'objet message dérivé – Crée un objet message susceptible de faire l'objet d'un renvoi ou d'une réponse.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction est utilisée pour créer un message à renvoyer à un autre destinataire ou pour répondre à un message donné. Le paramètre "objet" doit être un objet message avec au moins un destinataire.

Le message dérivé peut contenir des propriétés supplémentaires par rapport à celles du message original. De même, les propriétés du message dérivé peuvent ne pas avoir la même valeur que les propriétés correspondantes du message original. Certaines mises en œuvre modifieront, par exemple, le sujet d'un message de réponse tel que "résultats financiers trimestriels" en "Re: résultats financiers trimestriels". La mise en œuvre doit définir les règles applicables à cette fonction, indiquant quels sont les attributs modifiés et quels sont les attributs supplémentaires générés dans le message dérivé.

Un objet destinataire expéditeur est nécessaire pour répondre au message.

ARGUMENTS

Message original (*original_message***), type poignée d'objet**

Descripteur opaque de l'objet message devant faire l'objet d'une réexpédition ou d'une réponse

Action dérivée (*derived_action***), type énuméré**

Indique si le message dérivé doit faire l'objet d'une réexpédition ou d'une réponse. Ce paramètre peut prendre les valeurs suivantes:

CMC_DERIVED_ACTION_FORWARD
CMC_DERIVED_ACTION_REPLY_ORIGINATOR
CMC_DERIVED_ACTION_REPLY_ALL

CMC_DERIVED_ACTION_FORWARD – Le message est destiné à être réexpédié.

CMC_DERIVED_ACTION_REPLY_ORIGINATOR – Le message est destiné à une réponse à l'expéditeur.

CMC_DERIVED_ACTION_REPLY_ALL – Le message est destiné à une réponse à l'ensemble des destinataires du message original.

Héritage du contenu (*inherit_contents***), type booléen**

La totalité du contenu du message original est soit ignorée, soit copiée et incluse dans le message dérivé. Le nouvel objet hérite de la totalité du contenu si le paramètre est "Vrai".

Message modifié (*modified_message***), type booléen**

Indique si le message original doit être modifié.

Extensions de création d'objet message dérivé (*create_derived_message_object_extensions***), type extension**

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Message dérivé (*derived_message***), type poignée d'objet**

Nouveau descripteur opaque d'un objet message pouvant être réexpédié ou faire l'objet d'une réponse.

Extensions de création d'objet message dérivé (*create_derived_object_extensions***), type extension**

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code***)**

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Emission d'objet message (*Send Message Object*)

NOM

Emission d'objet message – Emission d'un message à partir de la boîte aux lettres en sortie.

RÉSUMÉ

```
#include <xcmc.h>

CMC_return_code
cmc_send_message_object(
    CMC_object_handle   object,
    CMC_extensions   *send_message_object_extensions
);
```

DESCRIPTION

Cette fonction est utilisée pour émettre un message à partir de la boîte aux lettres en sortie, si le conteneur de boîte aux lettres en sortie est pris en charge. La fonction essaiera également de transférer tous les autres messages confiés à la boîte aux lettres en sortie. Le paramètre "objet" doit faire référence à un objet message possédant au moins un destinataire.

ARGUMENTS

Objet (*object***), type poignée d'objet**

Descripteur opaque référençant l'objet message à déposer auprès du service de messagerie.

Extensions d'émission d'objet message (*send_message_object_extensions***), type extension**

Pointeur vers un tableau de structures d'extension CMC. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'émission d'objet message (*send_message_object_extensions***), type extension**

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Fonctions de manipulation de nom

Les fonctions de manipulation de nom fournissent la capacité de convertir un identificateur de propriété en un nom de propriété et réciproquement.

6.2.6.1 Identificateur vers nom (*Identifier To Name*)

NOM

Identificateur vers nom – Effectue la conversion d'un identificateur vers le nom non-ambigu qui lui est associé.

RÉSUMÉ

```
#include <xcmc.h>

CMC_return_code
cmc_identifier_to_name(
    CMC_id       identifier,
    CMC_name     *name,
    CMC_extension*identifier_to_name_extensions
);
```

DESCRIPTION

Cette fonction effectue la conversion d'un identificateur vers le nom non-ambigu qui lui est associé. Elle peut être utilisée pour les identificateurs de classes d'objets et les identificateurs de propriété.

Le nom est un identificateur public formel, tel qu'il est défini par l'ISO 9070. Il est non-ambigu et propre à la mise en œuvre. L'identificateur est utilisé pour identifier d'une manière non-ambiguë une propriété ou une classe d'objets dans la structure de propriétés CMC.

ARGUMENTS

Identificateur (*identifier*), type identificateur

Identificateur à convertir en un nom.

Extensions identificateur vers nom (*identifier_to_name_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Nom (*name*), type nom

Nom de la chaîne de caractères de l'identificateur. La chaîne de caractères est allouée par le service et doit être libérée en utilisant la fonction **cmc_free()**.

Extensions identificateur vers nom (*identifier_to_name_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Nom vers identificateur (*Name To Identifier*)

NOM

Nom vers identificateur – Effectue la conversion d'un nom non-ambigu vers l'identificateur qui lui est associé.

RÉSUMÉ

```
#include <xcmc.h>
CMC_return_code
cmc_name_to_identifier(
    CMC_name      name,
    CMC_id        *identifier,
    CMC_extension *name_to_identifier_extensions
);
```

DESCRIPTION

Cette fonction effectue la conversion d'un nom non-ambigu vers l'identificateur qui lui est associé. Elle peut être utilisée pour les identificateurs de classes d'objets et les identificateurs de propriété.

Le nom est un identificateur public formel, tel qu'il est défini par l'ISO 9070. Il est non-ambigu et propre à la mise en œuvre. L'identificateur est utilisé pour identifier d'une manière non-ambiguë une propriété ou une classe d'objets.

ARGUMENTS

Nom (*name*), type nom

Nom à convertir en un identificateur.

Extensions nom vers identificateur (*name_to_identifier_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Identificateur (*identifier*), type identificateur

Identificateur correspondant au nom.

Extensions nom vers identificateur (*name_to_identifier_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Fonctions de flux

Certaines propriétés CMC peuvent être définies sous la forme d'informations de contenu de taille importante. Ces propriétés ont besoin d'un groupe de fonctions permettant d'accéder à l'information de contenu sous la forme d'entrées ou de sorties par flux.

6.2.7.1 **Exportation de flux** (*Export stream*)

NOM

Exportation de flux – Exporte des données de flux vers le système de fichiers.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction exporte des données de flux vers un fichier.

ARGUMENTS

Flux (*stream*), type poignée de flux

Descripteur opaque du flux à partir duquel les données sont exportées.

Spécification de fichier (*file_specification*), type chaîne de caractères

Spécification complète de système de fichiers du fichier qui contiendra les données du flux.

Comptage (*count*), type uint32

Spécifie le nombre d'octets à exporter

Fanions d'exportation (*export_flags*), type fanions

Masque binaire de fanions. Les fanions non spécifiés doivent toujours avoir une valeur nulle au moment de l'appel. Les fanions non documentés sont réservés.

CMC_EXPORT_STREAM_OVERWRITE

CMC_EXPORT_STREAM_NOCREATE

CMC_EXPORT_STREAM_APPEND

CMC_EXPORT_STREAM_OVERWRITE – Positionné si la fonction doit remplacer un fichier existant correspondant à la spécification de fichier.

CMC_EXPORT_STREAM_NOCREATE – Positionné si la fonction ne doit pas créer un fichier correspondant à la spécification de fichier, si ce dernier n'existe pas déjà.

CMC_EXPORT_STREAM_APPEND – Positionné si la fonction doit ajouter les données du flux à un fichier existant correspondant à la spécification de fichier.

Extensions d'exportation de flux (*export_stream_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'exportation de flux (*export_stream_extensions*), type extensions

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Importation de fichier vers un flux (*Import File To Stream*)

NOM

Importation de fichier vers un flux – Importe des données depuis le système de fichiers vers un flux.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction importe, à destination d'un flux, des données en provenance d'un fichier.

ARGUMENTS

Flux (*stream*), type poignée de flux

Descripteur opaque d'un flux vers lequel les données doivent être importées.

Spécification de fichier (*file_specification*), type chaîne de caractères

Spécification complète du fichier à partir duquel les données sont importées.

Déplacement dans le fichier (*file_offset*), type uint32

Spécifie le déplacement à partir du début du fichier, exprimé en octets, à partir duquel les données sont lues.

Extensions d'importation de fichier vers flux (*import_file_to_stream_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'importation de fichier vers flux (*import_file to_stream_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Ouverture de flux (*Open Stream*)

NOM

Ouverture de flux – Ouvre une propriété pour des opérations de lecture ou d'écriture de flux.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction ouvre un flux pour la lecture ou l'écriture d'informations de contenu de taille importante depuis ou vers une propriété.

ARGUMENTS

Objet (*object*), type poignée d'objet

Descripteur opaque d'objet. Le descripteur encapsule l'identificateur de session.

Identificateur de propriété (*property_id*), type identificateur de propriété

Propriété devant être lue ou écrite au moyen du flux.

Operation (*operation*), type énuméré

Opération pour laquelle est utilisé le flux. Les opérations valides sont les suivantes:

CMC_OPEN_READ
CMC_OPEN_WRITE

CMC_OPEN_READ – Ouverture du flux pour une opération de lecture.

CMC_OPEN_WRITE – Ouverture du flux pour une opération d'écriture.

Extensions ouvertes de flux (*open_stream_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Flux (*stream*), type poignée de flux

Descripteur opaque alloué pour l'accès à la propriété spécifiée. La valeur renvoyée est passée à la fonction **cmc_free()** en vue de libérer le descripteur opaque et toutes les informations spécifiques du service concernant le flux, lorsque ce dernier n'est plus utilisé.

Extensions ouvertes de flux (*open_stream_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Lecture de flux (*Read Stream*)

NOM

Lecture de flux – Lit un flux d'information de contenu à partir de la propriété spécifiée.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction lit une information de contenu à partir de la propriété spécifiée vers un tampon géré par l'utilisateur.

ARGUMENTS

Flux (*stream*), type poignée de flux

Descripteur opaque de flux. Ce descripteur encapsule les descripteurs opaques de session et d'objet.

Comptage (*count*), type uint32

Spécifie le nombre maximal d'octets à lire. Une valeur nulle indique l'absence de maximum.

Extensions de lecture de flux (*read_stream_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Comptage (*count*), type uint32

Spécifie le nombre d'octets d'information de contenu effectivement lus. Une valeur nulle indique qu'aucune information n'a été lue.

Information de contenu (*content_information*), type tampon

Tampon renfermant l'information de contenu qui a été lue. Le tampon est alloué par le service et doit être libéré dans sa totalité au moyen d'un unique appel à la fonction **cmc_free()**. Ce tampon est géré par l'utilisateur de l'interface API et non par le service.

Extensions de lecture de flux (*read_stream_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Recherche dans un flux (*Seek Stream*)

NOM

Recherche dans un flux – Se positionne sur l'emplacement spécifié au sein du contenu d'information spécifié de la propriété de flux spécifiée.

RÉSUMÉ

```
#include <xcmc.h>

CMC_return_code
cmc_seek_stream(
    CMC_stream_handle   stream,
    CMC_enum     operation,
    CMC_uint32   *location,
    CMC_extension*seek_stream_extensions
);
```

DESCRIPTION

Cette fonction effectuera un déplacement vers la position spécifiée dans un flux de propriété. La position est spécifiée par un déplacement exprimé en octets à partir du début, de la fin ou de la position actuelle au sein de l'information de contenu.

ARGUMENTS

Flux (*stream*), type poignée de flux

Descripteur opaque de flux. Ce descripteur encapsule les descripteurs opaques de session et d'objet.

Operation (*operation*), type énuméré

Direction de la recherche. L'opération indiquera une recherche à partir du début de l'information de contenu, à partir de la fin ou à partir de la position actuelle. Les opérations valides sont les suivantes:

CMC_SEEK_BEGINNING
CMC_SEEK_END
CMC_SEEK_CURRENT_POSITION

CMC_SEEK_BEGINNING – Positionnement en utilisant le déplacement spécifié à partir du début de l'information de contenu.

CMC_SEEK_END – Positionnement en utilisant le déplacement spécifié à partir de la fin de l'information de contenu.

CMC_SEEK_CURRENT_POSITION – Positionnement en utilisant le déplacement spécifié à partir de la position actuelle dans l'information de contenu.

Emplacement (*location*), type uint32

Pointeur vers le déplacement ou la position exprimée en octets au sein du flux.

Extensions de recherche dans un flux (*seek_stream_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Emplacement (*location*), type uint32

Valeur effective du déplacement en octets.

Extensions de recherche dans le flux (*seek_stream_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 **Ecriture de flux** (*Write Stream*)

NOM

Ecriture de flux – Ecrire un flux d'information dans la propriété spécifiée.

RÉSUMÉ

```
#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
);
```

DESCRIPTION

Cette fonction écrit l'information de contenu vers la propriété spécifiée.

ARGUMENTS

Flux (*stream*), type poignée de flux

Descripteur opaque de flux. Ce descripteur encapsule les descripteurs opaques de session et d'objet.

Comptage (*count*), type uint32

Spécifie le nombre d'octets à écrire dans la propriété.

Information de contenu (*content_information*), type tampon

Pointeur vers un tampon renfermant l'information de contenu à écrire.

Extensions d'écriture de flux (*write_stream_extensions*), type extension

Pointeur vers un tableau de structures d'extensions CMC pour cette fonction. Ce tableau peut contenir aussi bien des extensions en entrée fournissant à la fonction des informations supplémentaires, que des extensions en sortie destinées à recevoir des informations en provenance de la fonction. Une valeur NULL indique que l'appelant n'utilise pas d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

RÉSULTATS

Extensions d'écriture de flux (*write_stream_extensions*), type extension

Les résultats du service seront disponibles dans l'extension si des extensions en sortie ont été transmises à la fonction dans la liste d'extensions. Prière de se référer à la définition de la structure d'extensions pour plus de détails.

Code retour (*return_code*)

Indique si la fonction a réussi ou, dans le cas contraire, la raison de l'échec. La valeur renvoyée peut être une indication de réussite ou l'une des ERREURS énumérées ci-dessous.

ERREURS

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 Codes retour

Le présent paragraphe définit les codes retour de l'interface CMC. Les codes retour de l'interface générique sont définis ci-dessous, ceux de l'interface C sont définis dans l'Annexe A "résumé des déclarations en langage C". Les tableaux 16 à 21 donne la liste des codes retour génériques et des fonctions auxquelles s'appliquent ces codes. La définition des codes retour est donnée à la suite des tableaux.

La mise en œuvre CMC ne renverra que les valeurs possibles s'appliquant à une fonction spécifique. La mise en œuvre peut renvoyer en cas de besoin d'autres indications d'erreur figurant dans la liste d'erreurs, qui ne sont pas attribuées d'une manière spécifique à une fonction. Il n'est pas recommandé de renvoyer des codes erreur autres que ceux indiqués dans la liste ci-dessous.

TABLEAU 16/X.446 – CODES RETOUR DE L'INTERFACE CMC SIMPLE

Code retour	Act	Free	List	Logoff	Logon	Query	Read	Look	Send	SndDoc
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

TABLEAU 16/X.446 – CODES RETOUR DE L'INTERFACE CMC SIMPLE *(FIN)*

Code retour	Act	Free	List	Logoff	Logon	Query	Read	Look	Send	SndDoc
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

TABLEAU 17/X.446 – CODES RETOUR DE LA FONCTION D'ADMINISTRATION ET DE LIAISON DE L'INTERFACE CMC COMPLETE

Code retour	Free	Logoff	Logon		Bind	Unbind
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		–	–

TABLEAU 17/X.446 – CODES RETOUR DE LA FONCTION D'ADMINISTRATION ET DE LIAISON DE L'INTERFACE CMC COMPLETE *(FIN)*

Code retour	Free	Logoff	Logon		Bind	Unbind
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	–		–	–

TABLEAU 18/X.446 – CODES RETOUR DE LA FONCTION DE COMPOSITION DE L'INTERFACE CMC COMPLETE

Code retour	Add Props	Comt Obj	Copy Obj	Copy Obj Hdl	Del Objs	Del Props	Open Obj Hdl	Restore Obj	Save Obj
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	–	–	–	–	–	–	–	–	–

TABLEAU 18/X.446 – CODES RETOUR DE LA FONCTION DE COMPOSITION DE L'INTERFACE CMC COMPLETE *(FIN)*

Code retour	Add Props	Comt Obj	Copy Obj	Copy Obj Hdl	Del Objs	Del Props	Open Obj Hdl	Restore Obj	Save Obj
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	–	–	–	–	–	–	–	–	–

TABLEAU 19/X.446 – CODES RETOUR DE LA FONCTION D'ENUMERATION DE L'INTERFACE CMC COMPLETE

Code retour	Get Last Err	Get Root Hdle	List Cont Props	List No Matched	List Objs	List Props	Open Cur	Read Cur	Read Props	Read Prop Costs	Upd Cur Pos	Upd Cur Pos w/ Sd
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	–	–	–	–	–	–	–	–	–	–	–	–

TABLEAU 19/X.446 – CODES RETOUR DE LA FONCTION D'ENUMERATION DE L'INTERFACE CMC COMPLETE *(FIN)*

Code retour	Get Last Err	Get Root Hdle	List Cont Props	List No Matched	List Objs	List Props	Open Cur	Read Cur	Read Props	Read Prop Costs	Upd Cur Pos	Upd Cur Pos w/ Sd
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	–	–	–	–	–	–	–	–	–	–	–	–

TABLEAU 20/X.446 – CODES RETOUR DE LA FONCTION DE NOTIFICATION D'EVENEMENT ET DE MESSAGERIE DE L'INTERFACE CMC COMPLETE

Code retour	Ck Event	Reg Event	Unreg Event	Call Clbks		Cr Der Msg	Snd Msg Obj
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	–	–	–	–		–	–

TABLEAU 20/X.446 – CODES RETOUR DE LA FONCTION DE NOTIFICATION D'EVENEMENT ET DE MESSAGERIE DE L'INTERFACE CMC COMPLETE *(FIN)*

Code retour	Ck Event	Reg Event	Unreg Event	Call Clbks		Cr Der Msg	Snd Msg Obj
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	–	–	–	–		–	–

TABLEAU 21/X.446 – CODES RETOUR DE LA FONCTION DE FLUX ET DE MANIPULATION DE NOM DE L'INTERFACE CMC COMPLETE

Code retour	Id to Name	Name to Id		Exp Str	Imp Str	Open Str	Read Str	Seek Str	Wrt Str
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	–	–		–	–	–	–	–	–

TABLEAU 21/X.446 – CODES RETOUR DE LA FONCTION DE FLUX ET DE MANIPULATION DE NOM DE L'INTERFACE CMC COMPLETE *(FIN)*

Code retour	Id to Name	Name to Id		Exp Str	Imp Str	Open Str	Read Str	Seek Str	Wrt Str
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	–	–		–	–	–	–	–	–

Les codes retour sont définis comme suit:

CMC_E_ACCESS_DENIED	L'accès a été refusé.
CMC_E_AMBIGUOUS_RECIPIENT	Le nom du destinataire est ambigu, des concordances multiples ont été trouvées.
CMC_E_ATTACHMENT_NOT_FOUND	L'attachement spécifié n'a pas été trouvé comme spécifié.
CMC_E_ATTACHMENT_OPEN_FAILURE	L'attachement spécifié a été trouvé, mais n'a pas pu être ouvert ou le fichier d'attachement n'a pas pu être créé.
CMC_E_ATTACHMENT_READ_FAILURE	Le fichier d'attachement spécifié a été trouvé et ouvert, mais une erreur est survenue lors de sa lecture.
CMC_E_ATTACHMENT_WRITE_FAILURE	Le fichier d'attachement a été créé correctement, mais une erreur est survenue lors de son écriture.
CMC_E_BIND_FAILURE	Impossible de lier l'application à la mise en œuvre.
CMC_E_CALLBACK_NOT_SUPPORTED	Le rappel automatique spécifié n'est pas pris en charge par la mise en œuvre.
CMC_E_COUNTED_STRING_UNSUPPORTED	La mise en œuvre ne prend pas en charge le type de chaîne de caractères avec comptage.
CMC_E_DISK_FULL	Espace disque insuffisant pour mener à bien l'opération demandée (il peut s'agir d'un disque local ou partagé).
CMC_E_FAILURE	Faute générale ne correspondant pas à un autre code erreur.
CMC_E_FUNCTION_INTERRUPTED	Fonction interrompue.
CMC_E_ID_NOT_FOUND	Identificateur non trouvé.
CMC_E_INSUFFICIENT_MEMORY	Mémoire insuffisante pour mener à bien l'opération demandée.
CMC_E_INVALID_CONFIGURATION	Configuration du service de messagerie sous-jacent non-valide, impossible d'ouvrir la session.
CMC_E_INVALID_CONTAINER_OBJECT	Objet conteneur non-valide.
CMC_E_INVALID_CURSOR_HANDLE	Descripteur opaque de curseur non-valide.
CMC_E_INVALID_ENUM	Valeur CMC énumérée non-valide.
CMC_E_INVALID_EVENT	Evénement spécifié non-valide.
CMC_E_INVALID_FILE_OFFSET	Déplacement non-valide spécifié dans un fichier.
CMC_E_INVALID_FILE_SPECIFICATION	Nom de fichier non-valide.
CMC_E_INVALID_FLAG	Valeur de fanion non-valide dans le paramètre de fanions.
CMC_E_INVALID_FUNCTION_EXT	Extension de fonction non-valide.
CMC_E_INVALID_MEMORY	Pointeur mémoire non-valide.
CMC_E_INVALID_MESSAGE_PARAMETER	Un des paramètres du message n'est pas valide.

CMC_E_INVALID_MESSAGE_REFERENCE	La référence du message n'est pas valide, ou n'est plus valide (par exemple: après une suppression).
CMC_E_INVALID_OBJECT_HANDLE	Descripteur opaque d'objet non-valide.
CMC_E_INVALID_PARAMETER	Paramètre fonctionnel non-valide.
CMC_E_INVALID_PROPERTY_ID	Identificateur de propriété non-valide.
CMC_E_INVALID_PROPERTY_NAME	Nom de propriété non-valide.
CMC_E_INVALID_RESTRICTION	Contrainte non-valide.
CMC_E_INVALID_SESSION_ID	L'identificateur de session n'est pas valide, ou n'est plus valide (par exemple: après une suppression).
CMC_E_INVALID_SOURCE_OBJECT	Objet source non-valide.
CMC_E_INVALID_STREAM_HANDLE	Descripteur opaque de flux non-valide.
CMC_E_INVALID_UI_ID	L'identificateur d'interface utilisateur n'est pas valide, ou n'est plus valide.
CMC_E_INVALID_VALUE	Valeur non-valide.
CMC_E_LOGON_FAILURE	Echec de l'ouverture de session dû à un nom d'utilisateur, ou à un mot de passe non-valide.
CMC_E_MESSAGE_IN_USE	L'action demandée n'a pu être effectuée, car le message est actuellement en cours d'utilisation.
CMC_E_NAME_NOT_FOUND	Nom non trouvé.
CMC_E_NO_EVENT	Evénement inexistant.
CMC_E_NO_MORE_BYTES_TO_WRITE	Il n'y a plus d'octets à écrire pour le flux.
CMC_E_NOT_SUPPORTED	Opération non prise en charge par cette mise en œuvre.
CMC_E_PASSWORD_REQUIRED	Ce service de messagerie exige un mot de passe.
CMC_E_PROPERTY_DATA_TYPE_NOT_ SUPPORTED	Type de données de propriété non pris en charge par cette mise en œuvre.
CMC_E_PROPERTY_ID_NOT_FOUND	Identificateur de propriété non trouvé.
CMC_E_PROPERTY_NAME_NOT_FOUND	Nom de propriété non trouvé.
CMC_E_PROPERTY_PROBLEMS	Problèmes de propriété.
CMC_E_RECIPIENT_NOT_FOUND	Un ou plusieurs destinataires non trouvés.
CMC_E_REQUIRED_PROPS_MISSING	Une ou plusieurs propriétés absentes.
CMC_E_RESTRICTION_NOT_SUPPORTED	Contrainte trop complexe, non prise en charge par cette mise en œuvre.

CMC_E_SERVICE_UNAVAILABLE	Service demandé indisponible.
CMC_E_TEXT_TOO_LARGE	Taille de chaîne de caractères de texte trop importante.
CMC_E_TOO_MANY_CONTENT_ITEMS	Dépassement du nombre maximal d'articles.
CMC_E_TOO_MANY_FILES	La mise en œuvre ne peut pas prendre en charge le nombre de fichiers spécifiés.
CMC_E_TOO_MANY_RECIPIENTS	La mise en œuvre ne peut pas prendre en charge le nombre de destinataires spécifiés.
CMC_E_UNABLE_TO_NOT_MARK_READ	Le fanion CMC_E_UNABLE_TO_NOT_MARK_READ ne peut pas être pris en charge.
CMC_E_UNBIND_FAILURE	Erreur survenue lors d'une tentative de suppression de liaison entre application et mise en œuvre.
CMC_E_UNRECOGNIZED_IDENTIFIER	Identificateur non reconnu.
CMC_E_UNRECOGNIZED_MESSAGE_TYPE	Type de message non pris en charge par cette mise en œuvre.
CMC_E_UNSUPPORTED_ACTION	Action non prise en charge par cette mise en œuvre.
CMC_E_UNSUPPORTED_CHARACTER_SET	Jeu de caractères demandé non pris en charge.
CMC_E_UNSUPPORTED_DATA_EXT	Extension de données demandée non prise en charge.
CMC_E_UNSUPPORTED_FLAG	Fanion demandé non pris en charge.
CMC_E_UNSUPPORTED_FUNCTION_EXT	Extension de fonction non prise en charge.
CMC_E_UNSUPPORTED_KEYS	Les clés de tri spécifiées ne sont pas prises en charge.
CMC_E_UNSUPPORTED_VALUE	Valeur non prise en charge.
CMC_E_UNSUPPORTED_VERSION	La version spécifiée dans l'appel ne peut pas être prise en charge par cette mise en œuvre CMC.
CMC_E_USER_CANCEL	Opération interrompue par l'utilisateur.
CMC_E_USER_NOT_LOGGED_ON	L'utilisateur n'a pas ouvert de session et le fanion CMC_E_USER_NOT_LOGGED_ON n'est pas positionné.

8 Conformité

Une mise en œuvre de l'interface API d'appel commun de messagerie devra satisfaire aux critères suivants en vue de se conformer à la présente Recommandation.

- toutes les fonctions et structures de données doivent être mises en œuvre telles qu'elles sont spécifiées. Des prescriptions d'autres clauses de la présente Recommandation décrivant des fonctions optionnelles, ou comportant des exceptions, ont priorité par rapport à ce critère;
- la mise en œuvre devra être en mesure de transporter au minimum des messages CMC de type interpersonnel;
- la prise en charge d'applications utilisant les prescriptions de l'association XAPIA pour la version CMC 1.0 est recommandée pour des mises en œuvre des interfaces CMC simple et CMC complète;
- la prise en charge des interfaces CMC simple et CMC complète est obligatoire pour des mises en œuvre de l'interface CMC complète;

- toutes les classes d'objets énumérées dans le paragraphe 3 doivent être mises en œuvre telles qu'elles sont définies. Des prescriptions d'autres clauses de la Recommandation décrivant des fonctions optionnelles ou comportant des exceptions ont priorité par rapport à ce critère;
- les propriétés d'objets qui sont mentionnées comme obligatoires dans les tableaux de caractéristiques de propriétés seront prises en charge;
- la prise en charge de jeux de caractères est l'affaire de la mise en œuvre sous-jacente. La prise en charge d'un jeu de caractères défini par défaut par la mise en œuvre est obligatoire. D'autres jeux de caractères peuvent être pris en charge d'une manière optionnelle. La prise en charge des chaînes de caractères avec comptage n'est pas exigée;
- toutes les extensions sont optionnelles. Les fournisseurs sont encouragés à prendre en charge l'ensemble d'extensions normalisées CMC spécifié dans la présente Recommandation. Il est en outre recommandé de réaliser des ensembles d'extensions normalisés pour tout service de messagerie propriétaire, propre au fournisseur ou non, pour lequel une interface CMC est fournie, en vue de traiter les fonctions propres à ce service de messagerie et d'enregistrer ces extensions d'une manière externe;
- la conformité minimale pour une extension sera définie par le créateur de l'ensemble d'extensions;
- le gestionnaire CMC et la mise en œuvre CMC doivent fournir une mise en œuvre des fonctions cmc_bind_implementation() et de cmc_unbind_implementation(), et l'appel de la fonction cmc_bind_implementation() doit renvoyer un pointeur vers une table de distribution. Si des mises en œuvre multiples sont prises en charge, le gestionnaire CMC peut fournir une fonction d'énumération des mises en œuvre CMC connues pour une plate-forme donnée (avec des moyens éventuels de recherche) ainsi qu'une fonction d'enregistrement des mises en œuvre CMC;
- la mise en œuvre CMC doit pouvoir prendre en charge les appels directement adressés à sa fonction ainsi que ceux qui sont adressés indirectement par la table de distribution.

Annexe A

Résumé des déclarations en langage C

A.1 Résumé des déclarations en langage C

Le présent sous-paragraphe donne la liste des déclarations définissant l'interface CMC pour le langage de programmation C. Toutes les déclarations, à l'exception des déclarations de constantes symboliques, figurent également au paragraphe 4 "Structures de données" ou au paragraphe 6 "Fonctions d'interface".

Les déclarations regroupées ici constituent le contenu d'un fichier d'en-tête qui doit être fourni aux programmeurs d'application. La référence de ce fichier est **<xcmc.h>**. Les symboles figurant dans les déclarations sont les seuls dont le service fournit la visibilité à l'application.

```
/*DÉBUT DE L'INTERFACE CMC 2.0*/

#ifndef_XCMC_H
#define_XCMC_H

#ifdef_cplusplus
extern "C" {
#endif

/*TYPES DE DONNÉES DE BASE*/
#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;

/*DÉFINITION DE LA TAILLE DES CARACTÈRES*/
#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)

/*STRUCTURES DE DONNÉES*/

/*CHAÎNES DE CARACTÈRES AVEC COMPTAGE*/
typedef struct {
      CMC_uint32                length;
      CMC_CHAR                  string[1];
} CMC_counted_string;

/*IDENTIFICATEUR DE SESSION*/
typedef CMC_uint32              CMC_session_id;
```

```
#ifndef DIFFERENT_PLATFORM
/*DESCRIPTEUR OPAQUE DE CURSEUR*/
typedef CMC_uint32              CMC_cursor_handle;

/*DESCRIPTEUR OPAQUE D'OBJET*/
typedef CMC_uint32              CMC_object_handle;

/*DESCRIPTEUR OPAQUE DE FLUX*/
typedef CMC_uint32              CMC_stream_handle;

/*DESCRIPTEUR OPAQUE NULL*/
#define CMC_NULL_OBJECT_HANDLE ((CMC_object_handle) 0)
#endif

/*DONNÉES OPAQUES*/
typedef struct CMC_TAG_OPAQUE_DATA {
       CMC_size                 size;
       CMC_byte                 *data;
} CMC_opaque_data;

/*TEMPS*/
/* les champs unusedX non utilisés sont nécessaires */
/* pour aligner la structure sur une frontière de 4 octets */
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)

/*IDENTIFICATEUR D'INTERFACE UTILISATEUR*/
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;

/*IDENTIFICATEUR DE PROPRIÉTÉ*/
typedef CMC_uint32              CMC_id;

/*NOM DE PROPRIÉTÉ*/
typedef CMC_string              CMC_name;

/*DÉFINITIONS DE PROPRIÉTÉS À VALEURS MULTIPLES*/

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;
```

```
/*PROPRIÉTÉ*/
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;

/*ÉVÉNEMENT/
typedef CMC_uint32              CMC_event;

/* TYPES D'ÉVÉNEMENTS */
#define CMC_EVENT_NEW_MESSAGES            ((CMC_enum) 0)

/*RAPPEL AUTOMATIQUE*/

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
);
/*CONTRAINTE DE CURSEUR*/
```

```
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;

/* TYPES DE CONTRAINTES ET CONSTANTES */
#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)

/*CLÉ DE TRI DE CURSEUR*/
typedef struct TAG_CURSOR_SORT_KEY{
      CMC_id                        property;
      CMC_enum                      order;
} CMC_cursor_sort_key;

/* CONSTANTES DE CLÉ DE TRI DE CURSEUR */
#define CMC_SORT_DEFAULT                        ((CMC_enum) 0)
#define CMC_SORT_ASCEND                         ((CMC_enum) 1)
#define CMC_SORT_DESCEND                        ((CMC_enum) 2)

/*ATTACHEMENT*/

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;

/* FANIONS D'ATTACHEMENT */
#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"

/*RÉFÉRENCE DE MESSAGE*/
typedef CMC_counted_string      CMC_message_reference;

/*DESTINATAIRE*/
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;

/* TYPES DE NOM */
#define CMC_TYPE_UNKNOWN                        ((CMC_enum) 0)
#define CMC_TYPE_INDIVIDUAL                     ((CMC_enum) 1)
#define CMC_TYPE_GROUP                          ((CMC_enum) 2)

/* RÔLES */
#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)

/* FANIONS DE DESTINATAIRE */
#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;

/* FANIONS DE MESSAGE */
#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)

/* TYPES DE MESSAGE */
#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"

/*RÉSUMÉ DE MESSAGE*/
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;

/* FANIONS DE RÉSUMÉ DE MESSAGE */
#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)

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

/* FANIONS DE COMPTE RENDU */
#define CMC_REPORT_LAST_ELEMENT             ((CMC_flags) 0x00000001)

/* TYPES DE COMPTE RENDU */
#define CMC_X400_DR                         ((CMC_enum) 0)
#define CMC_X400_NDR                        ((CMC_enum) 1)

/*FONCTIONS CMC*/
/*FANIONS COMMUNS AUX FONCTIONS*/
#define CMC_ERROR_UI_ALLOWED                ((CMC_flags) 0x01000000)
#define CMC_LOGON_UI_ALLOWED                ((CMC_flags) 0x02000000)
#define CMC_COUNTED_STRING_TYPE             ((CMC_flags) 0x04000000)

/*CLASSES D'OBJETS*/
#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"

/*PROPRIÉTÉS DES OBJETS*/

/* Classe d'objets, s'applique à tous les objets */

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

/* Carnet d'adresses */

#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"

/* Type de contenu */

#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"

/* Liste de distribution */

#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"

/* Conteneur de messages */

#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"

/* Destinataire */

#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"

/* Compte rendu */

#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"

/* Conteneur racine */

#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"

/* Information par destinataire */

#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"

/* Conteneur de profile */

#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"

/* Constante de valeur de propriété, valeurs d'identificateur CMC */

/* Classe d'objets, s'applique à tous les objets */

#define CMC_PV_OBJECT_CLASS                              0

/* Carnet d'adresses */
#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
```

```
/* Article de contenu */
#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
/* Liste de distribution */
#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
/* Conteneur de messages */
#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
/* Destinataire */
#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
/* Compte rendu */
#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
```

```
/* Conteneur racine */
#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
/* Information par destinataire */
#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
/* Profil */
#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

/*VALEURS DE CONSTANTES DE PROPRIÉTÉ D'OBJET*/

/* Carnet d'adresses */
#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)

/* Information de contenu */
#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"

/* Type de codage */
#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"

/* Type de contenu */
#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"

/* Type d'article de contenu */
#define CMC_IT_NOTE                             ((CMC_enum) 0)
#define CMC_IT_ATTACHMENT                       ((CMC_enum) 1)
#define CMC_IT_ANNOTATION                       ((CMC_enum) 2)

/* Types de message et constantes de message */
#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)

/*Statut de message client*/
#define CMC_MESSAGE_STATUS_DRAFT                ((CMC_enum) 0)

/*Statut de message en sortie*/
#define CMC_MESSAGE_STATUS_DELETED              ((CMC_enum) 0)
#define CMC_MESSAGE_STATUS_SUBMITTED            ((CMC_enum) 1)
#define CMC_MESSAGE_STATUS_SENT                 ((CMC_enum) 2)

/*Statut de message en entrée*/
#define CMC_MESSAGE_STATUS_NEW                  ((CMC_enum) 0)
#define CMC_MESSAGE_STATUS_READ                 ((CMC_enum) 1)
#define CMC_MESSAGE_STATUS_CHANGED              ((CMC_enum) 2)

/* Types et constantes de conteneur de messages */

#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)

/* Destinataire */
#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)

/* Compte rendu */
#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)

/* Conteneur racine */
#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)

/* Information par destinataire */
#define CMC_PRI_DR                              ((CMC_enum) 0)
#define CMC_PRI_NDR                             ((CMC_enum) 1)
#define CMC_PRI_UNKNOWN                         ((CMC_enum) 2)

/* Profil */
#define CMC_CONF_SIMPLE_CMC                     ((CMC_enum) 0)
#define CMC_CONF_FULL_CMC                       ((CMC_enum) 1)
```

```
/* FANIONS D'EXTENSION */
#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

/* ÉMISSION */
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
);

/* ÉMISSION DE 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
);

/* AGIR SUR */
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
);

/* LISTE */
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
);

/* LIRE */
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
);
```

```
/* RECHERCHER */
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
);

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

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

/* OUVERTURE DE SESSION */
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
);

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

/* CMC COMPLET */

/* COPIER UN OBJET */
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
);

/* AJOUTER DES PROPRIÉTÉS */
CMC_return_code
cmc_add_properties(
       CMC_object_handle         object,
       CMC_uint32                number_properties,
       CMC_property              *properties,
       CMC_extension             *add_properties_extensions
);
```

```
/* CONFIER UN OBJET */
CMC_return_code
cmc_commit_object(
       CMC_object_handle         source_object,
       CMC_extension             *commit_object_extensions
);

/* COPIE DE DESCRIPTEUR OPAQUE D'OBJETS */
CMC_return_code
cmc_copy_object_handle(
       CMC_object_handle         source_handle,
       CMC_object_handle         *new_handle,
       CMC_extension             *copy_object_handle_extensions
);

/* CRÉER UN OBJET MESSAGE DÉRIVÉ */
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
);

/* SUPPRIMER DES OBJETS */
CMC_return_code
cmc_delete_objects(
       CMC_uint32                number_objects,
       CMC_object_handle         *object,
       CMC_extension             *delete_objects_extensions
);

/* SUPPRIMER DES PROPRIÉTÉS */
CMC_return_code
cmc_delete_properties(
       CMC_object_handle         object,
       CMC_uint32                number_properties,
       CMC_id                    *property_ids,
       CMC_extension             *delete_properties_extensions
);

/* OBTENIR LE DESCRIPTEUR OPAQUE RACINE */
CMC_return_code
cmc_get_root_handle(
       CMC_session_id            session,
       CMC_object_handle         *root_object_handle,
       CMC_extension             *get_root_handle_extensions
);

/* IDENTIFICATEUR VERS NOM */
CMC_return_code
cmc_identifier_to_name(
       CMC_id                    identifier,
       CMC_name                  *name,
       CMC_extension             *identifier_to_name_extensions
);

/* LISTE DES PROPRIÉTÉS CONTENUES */
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
);
```

```
/* LISTE DU NOMBRE DE CONCORDANCES */
CMC_return_code
cmc_list_number_matched(
       CMC_cursor_handle         *cursor,
       CMC_uint32                *number_matches,
       CMC_extension             *list_number_matched_extensions
);

/* LISTE D'OBJETS */
CMC_return_code
cmc_list_objects(
       CMC_cursor_handle         *cursor,
       CMC_sint32                *number_objects,
       CMC_object_handle         **objects,
       CMC_extension             *list_objects_extensions
);

/* LISTE DE PROPRIÉTÉS */
CMC_return_code
cmc_list_properties(
       CMC_object_handle         *object,
       CMC_uint32                *number_properties,
       CMC_id                    **property_ids,
       CMC_extension             *list_properties_extensions
);

/* NOM VERS IDENTIFICATEUR */
CMC_return_code
cmc_name_to_identifier(
       CMC_name                  name,
       CMC_id                    *identifier,
       CMC_extension             *name_to_identifier_extensions
);

/* OUVERTURE DE CURSEUR */
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
);

/* OUVERTURE DE DESCRIPTEUR OPAQUE D'OBJET */
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
);

/* LECTURE DE CURSEUR */
CMC_return_code
cmc_read_cursor(
       CMC_cursor_handle         *cursor,
       CMC_uint32                *position_numerator,
       CMC_uint32                *position_denominator,
       CMC_extension             *read_cursor_extensions
);

/* LECTURE DE PROPRIÉTÉS */
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
);
```

```
/* LECTURE DE COÛTS DE PROPRIÉTÉS */
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
);

/* RESTAURATION D'OBJET */
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
);

/* SAUVEGARDE D'OBJET */
CMC_return_code
cmc_save_object(
       CMC_object_handle         object,
       CMC_string                file_specification,
       CMC_flags                 save_flags,
       CMC_extension             *save_object_extensions
);

/* ÉMISSION D'UN OBJET MESSAGE */
CMC_return_code
cmc_send_message_object(
       CMC_object_handle         message_to_send,
       CMC_extension             *send_message_object_extensions
);

/* MISE À JOUR DE LA POSITION DU CURSEUR */
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
);

/* MISE À JOUR DU CURSEUR AVEC PLACEMENT */
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
);

/* SUPERVISION D'ÉVÉNEMENT */
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
);

/* ENREGISTREMENT D'ÉVÉNEMENT */
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
);
```

```
/* RÉSILIATION D'ÉVÉNEMENT */
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
);

/* APPEL DE RAPPEL AUTOMATIQUE */
CMC_return_code
cmc_call_callbacks(
       CMC_session_id            session,
       CMC_event                 event_type,
       CMC_extension             *call_callbacks_extensions
);

/* EXPORTER UN FLUX */
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
);

/* IMPORTER UN FICHIER VERS UN FLUX */
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
);

/* OUVERTURE DE FLUX */
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
);

/* LECTURE DE FLUX */
CMC_return_code
cmc_read_stream(
       CMC_stream_handle         stream,
       CMC_uint32                *count,
       CMC_buffer                content_information,
       CMC_extension             *read_stream_extensions
);

/* RECHERCHE DANS UN FLUX */
CMC_return_code
cmc_seek_stream(
       CMC_stream_handle         stream,
       CMC_enum                  operation,
       CMC_uint32                *location,
       CMC_extension             *seek_stream_extensions
);

/* ÉCRITURE DE FLUX */
CMC_return_code
cmc_write_stream(
       CMC_stream_handle         *stream,
       CMC_uint32                *count,
       CMC_buffer                *content_information,
       CMC_extension             *write_stream_extensions
);
```

```
/* OBTENIR LA DERNIÈRE ERREUR */
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
);

/* TABLE DE DISTRIBUTION POUR DES MISES EN ŒUVRE MULTIPLES */

typedef struct {
       CMC_extension                 *dispatch_table_extensions;

       /* ÉMETTRE */
       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
       );

       /* ÉMETTRE UN 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
       );

       /* AGIR SUR */
       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
       );

       /* LISTE */
       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
       );

       /* LECTURE */
       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
       );
```

```
/* CONSULTATION */
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
);

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

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

/* OUVERTURE DE SESSION */
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
);

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

/* CMC COMPLET */

/* COPIE D'OBJET */
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
);

/* AJOUT DE PROPRIÉTÉS */
CMC_return_code
(*cmc_add_properties)(
   CMC_object_handle         object,
   CMC_uint32                number_properties,
   CMC_property              *properties,
   CMC_extension             *add_properties_extensions
);
```

```
/* CONFIER UN OBJET */
CMC_return_code
(*cmc_commit_object)(
   CMC_object_handle          source_object,
   CMC_extension              *commit_object_extensions
);

/* COPIE DE DESCRIPTEUR OPAQUE D'OBJET */
CMC_return_code
(*cmc_copy_object_handle)(
   CMC_object_handle          source_handle,
   CMC_object_handle          *new_handle,
   CMC_extension              *copy_object_handle_extensions
);

/* CRÉATION D'OBJET MESSAGE DÉRIVÉ */
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
);

/* SUPPRESSION D'OBJETS */
CMC_return_code
(*cmc_delete_objects)(
   CMC_uint32                 number_objects,
   CMC_object_handle          *object,
   CMC_extension              *delete_objects_extensions
);

/* SUPPRESSION DE PROPRIÉTÉS */
CMC_return_code
(*cmc_delete_properties)(
   CMC_object_handle          object,
   CMC_uint32                 number_properties,
   CMC_id                     *property_ids,
   CMC_extension              *delete_properties_extensions
);

/* OBTENIR LE DESCRIPTEUR OPAQUE RACINE */
CMC_return_code
(*cmc_get_root_handle)(
   CMC_session_id             session,
   CMC_object_handle          *root_object_handle,
   CMC_extension              *get_root_handle_extensions
);

/* IDENTIFICATEUR VERS NOM */
CMC_return_code
(*cmc_identifier_to_name)(
   CMC_id                     identifier,
   CMC_name                   *name,
   CMC_extension              *identifier_to_name_extensions
);

/* LISTE DES PROPRIÉTÉS CONTENUES */
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
);
```

```
/* LISTE DU NOMBRE DE CONCORDANCES */
CMC_return_code
(*cmc_list_number_matched)(
    CMC_cursor_handle          *cursor,
    CMC_uint32                 *number_matches,
    CMC_extension              *list_number_matched_extensions
);

/* LISTE D'OBJETS */
CMC_return_code
(*cmc_list_objects)(
    CMC_cursor_handle          *cursor,
    CMC_sint32                 *number_objects,
    CMC_object_handle          **objects,
    CMC_extension              *list_objects_extensions
);

/* LISTE DE PROPRIÉTÉS */
CMC_return_code
(*cmc_list_properties)(
    CMC_object_handle          *object,
    CMC_uint32                 *number_properties,
    CMC_id                     **property_ids,
    CMC_extension              *list_properties_extensions
);

/* NOM VERS IDENTIFICATEUR */
CMC_return_code
(*cmc_name_to_identifier)(
    CMC_name                   name,
    CMC_id                     *identifier,
    CMC_extension              *name_to_identifier_extensions
);

/* OUVERTURE DE CURSEUR */
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
);

/* OUVERTURE DE DESCRIPTEUR OPAQUE D'OBJET */
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
);

/* LECTURE DE CURSEUR */
CMC_return_code
(*cmc_read_cursor)(
    CMC_cursor_handle          *cursor,
    CMC_uint32                 *position_numerator,
    CMC_uint32                 *position_denominator,
    CMC_extension              *read_cursor_extensions
);

/* LECTURE DE PROPRIÉTÉS*/
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
);
```

```
/* LECTURE DE COÛTS DE PROPRIÉTÉ*/
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
);

/* RESTAURATION D'OBJET */
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
);

/* SAUVEGARDE D'OBJET */
CMC_return_code
(*cmc_save_object)(
    CMC_object_handle          object,
    CMC_string                 file_specification,
    CMC_flags                  save_flags,
    CMC_extension              *save_object_extensions
);

/* ÉMISSION D'OBJET MESSAGE*/
CMC_return_code
(*cmc_send_message_object)(
    CMC_object_handle          message_to_send,
    CMC_extension              *send_message_object_extensions
);

/* MISE À JOUR DE LA POSITION DU CURSEUR*/
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
);

/* MISE À JOUR DE LA POSITION DU CURSEUR AVEC PLACEMENT */
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
);

/* SUPERVISION D'ÉVÉNEMENT */
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
);

/* ENREGISTREMENT D'ÉVÉNEMENT */
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
);
```

```
/* RÉSILIATION D'ÉVÉNEMENT */
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
);

/* APPEL DE RAPPEL AUTOMATIQUE */
CMC_return_code
(*cmc_call_callbacks)(
    CMC_session_id              session,
    CMC_event                   event_type,
    CMC_extension               *call_callbacks_extensions
);

/* EXPORTATION DE FLUX */
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
);

/* IMPORTER UN FICHIER VERS UN FLUX */
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
);

/* OUVERTURE DE FLUX */
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
);

/* LECTURE DE FLUX */
CMC_return_code
(*cmc_read_stream)(
    CMC_stream_handle           stream,
    CMC_uint32                  *count,
    CMC_buffer                  content_information,
    CMC_extension               *read_stream_extensions
);

/* RECHERCHE DANS UN FLUX */
CMC_return_code
(*cmc_seek_stream)(
    CMC_stream_handle           stream,
    CMC_enum                    operation,
    CMC_uint32                  *location,
    CMC_extension               *seek_stream_extensions
);

/* ÉCRITURE DE FLUX */
CMC_return_code
(*cmc_write_stream)(
    CMC_stream_handle           *stream,
    CMC_uint32                  *count,
    CMC_buffer                  *content_information,
    CMC_extension               *write_stream_extensions
);
```

```
       /* OBTENIR LA DERNIÈRE ERREUR */
       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;

/* LIER LA MISE EN ŒUVRE */
CMC_return_code
cmc_bind_implementation(
       CMC_guid                  implementation_name,
       CMC_dispatch_table        **dispatch_table,
       CMC_extension             *cmc_bind_extensions
);

/* DÉLIER LA MISE EN ŒUVRE */
CMC_return_code
cmc_unbind_implementation(
       CMC_guid                  implementation_name,
       CMC_extension             *cmc_unbind_implementation_extensions
);

#else   /* Contrepartie de fonction WCHAR / UNICODE */

/* ÉMISSION */
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
);

/* ÉMISSION DE 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
);

/* AGIR SUR */
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
);

/* LISTE */
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
);
```

```
/* LECTURE */
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
);

/* CONSULTATION */
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
);

/* LIBÉRER */
CMC_return_code
cmc_free_W(
       CMC_buffer                memory
);

/* FERMETURE DE SESSION */
CMC_return_code
cmc_logoff_W(
       CMC_session_id            session,
       CMC_ui_id                 ui_id,
       CMC_flags                 logoff_flags,
       CMC_extension             *logoff_extensions
);

/* OUVERTURE DE SESSION */
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
);

/* INTERROGATION DE CONFIGURATION */
CMC_return_code
cmc_query_configuration_W(
       CMC_session_id            session,
       CMC_enum                  item,
       CMC_buffer                reference,
       CMC_extension             *config_extensions
);

/* CMC COMPLET */

/* COPIE D'OBJET */
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
);
```

```
/* AJOUT DE PROPRIÉTÉS */
CMC_return_code
cmc_add_properties_W(
       CMC_object_handle         object,
       CMC_uint32                number_properties,
       CMC_property              *properties,
       CMC_extension             *add_properties_extensions
);

/* CONFIER UN OBJET */
CMC_return_code
cmc_commit_object_W(
       CMC_object_handle         source_object,
       CMC_extension             *commit_object_extensions
);

/* COPIE DE DESCRIPTEUR OPAQUE D'OBJET */
CMC_return_code
cmc_copy_object_handle_W(
       CMC_object_handle         source_handle,
       CMC_object_handle         *new_handle,
       CMC_extension             *copy_object_handle_extensions
);

/* CRÉATION D'OBJET MESSAGE DÉRIVÉ */
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
);

/* SUPPRESSION D'OBJETS */
CMC_return_code
cmc_delete_objects_W(
       CMC_uint32                number_objects,
       CMC_object_handle         *object,
       CMC_extension             *delete_objects_extensions
);

/* SUPPRESSION DE PROPRIÉTÉS */
CMC_return_code
cmc_delete_properties_W(
       CMC_object_handle         object,
       CMC_uint32                number_properties,
       CMC_id                    *property_ids,
       CMC_extension             *delete_properties_extensions
);

/* OBTENIR LE DESCRIPTEUR OPAQUE RACINE */
CMC_return_code
cmc_get_root_handle_W(
       CMC_session_id            session,
       CMC_object_handle         *root_object_handle,
       CMC_extension             *get_root_handle_extensions
);

/* IDENTIFICATEUR VERS NOM */
CMC_return_code
cmc_identifier_to_name_W(
       CMC_id                    identifier,
       CMC_name                  *name,
       CMC_extension             *identifier_to_name_extensions
);
```

```
/* LISTE DES PROPRIÉTÉS CONTENUES */
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
);

/* LISTE DU NOMBRE DE CONCORDANCES */
CMC_return_code
cmc_list_number_matched_W(
       CMC_cursor_handle         *cursor,
       CMC_uint32                *number_matches,
       CMC_extension             *list_number_matched_extensions
);

/* LISTE D'OBJETS */
CMC_return_code
cmc_list_objects_W(
       CMC_cursor_handle         *cursor,
       CMC_sint32                *number_objects,
       CMC_object_handle         **objects,
       CMC_extension             *list_objects_extensions
);

/* LISTE DE PROPRIÉTÉS */
CMC_return_code
cmc_list_properties_W(
       CMC_object_handle         *object,
       CMC_uint32                *number_properties,
       CMC_id                    **property_ids,
       CMC_extension             *list_properties_extensions
);

/* NOM VERS IDENTIFICATEUR */
CMC_return_code
cmc_name_to_identifier_W(
       CMC_name                  property_name,
       CMC_id                    *property_id,
       CMC_extension             *name_to_identifier_extensions
);

/* OUVERTURE DE CURSEUR */
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
);

/* OUVERTURE DE DESCRIPTEUR OPAQUE D'OBJET */
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
);

/* LECTURE DE CURSEUR */
CMC_return_code
cmc_read_cursor_W(
       CMC_cursor_handle         *cursor,
       CMC_uint32                *position_numerator,
       CMC_uint32                *position_denominator,
       CMC_extension             *read_cursor_extensions
);
```

```
/* LECTURE DE PROPRIÉTÉS*/
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
);

/* LECTURE DE COÛTS DE PROPRIÉTÉ*/
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
);

/* RESTAURATION D'OBJET */
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
);

/* SAUVEGARDE D'OBJET */
CMC_return_code
cmc_save_object_W(
       CMC_object_handle         object,
       CMC_string                file_specification,
       CMC_flags                 save_flags,
       CMC_extension             *save_object_extensions
);

/* ÉMISSION D'OBJET MESSAGE*/
CMC_return_code
cmc_send_message_object_W(
       CMC_object_handle         message_to_send,
       CMC_extension             *send_message_object_extensions
);

/* MISE À JOUR DE LA POSITION DU CURSEUR*/
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
);

/* MISE À JOUR DE LA POSITION DU CURSEUR AVEC PLACEMENT */
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
);

/* SUPERVISION D'ÉVÉNEMENT */
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
);
```

```
/* ENREGISTREMENT D'ÉVÉNEMENT */
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
);

/* RÉSILIATION D'ÉVÉNEMENT */
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
);

/* APPEL DE RAPPEL AUTOMATIQUE */
CMC_return_code
cmc_call_callbacks_W(
       CMC_session_id            session,
       CMC_event                 event_type,
       CMC_extension             *call_callbacks_extensions
);

/* EXPORTATION DE FLUX */
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
);

/* IMPORTER UN FICHIER VERS UN FLUX */
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
);

/* OUVERTURE DE FLUX */
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
);

/* LECTURE DE FLUX */
CMC_return_code
cmc_read_stream_W(
       CMC_stream_handle         stream,
       CMC_uint32                *count,
       CMC_buffer                content_information,
       CMC_extension             *read_stream_extensions
);

/* RECHERCHE DANS UN FLUX */
CMC_return_code
cmc_seek_stream_W(
       CMC_stream_handle         stream,
       CMC_enum                  operation,
       CMC_uint32                *location,
       CMC_extension             *seek_stream_extensions
);
```

```
/* ÉCRITURE DE FLUX */
CMC_return_code
cmc_write_stream_W(
      CMC_stream_handle         *stream,
      CMC_uint32                *count,
      CMC_buffer                *content_information,
      CMC_extension             *write_stream_extensions
);

/* OBTENIR LA DERNIÈRE ERREUR */
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
);

/* TABLE DE DISTRIBUTION UNICODE, MISES EN ŒUVRE MULTIPLES */

typedef struct {
      CMC_extension                 *dispatch_table_extensions;
      /* ÉMISSION */
      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
      );

      /* ÉMISSION DE 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
      );

      /* AGIR SUR */
      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
      );

      /* LISTE */
      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
      );
```

```
/* LECTURE */
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
);

/* CONSULTATION */
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
);

/* LIBÉRER */
CMC_return_code
(*cmc_free_W)(
      CMC_buffer              memory
);

/* FERMETURE DE SESSION */
CMC_return_code
(*cmc_logoff_W)(
      CMC_session_id          session,
      CMC_ui_id               ui_id,
      CMC_flags               logoff_flags,
      CMC_extension           *logoff_extensions
);

/* OUVERTURE DE SESSION */
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
);

/* INTERROGATION DE CONFIGURATION */
CMC_return_code
(*cmc_query_configuration_W)(
      CMC_session_id          session,
      CMC_enum                item,
      CMC_buffer              reference,
      CMC_extension           *config_extensions
);

/* CMC COMPLET */

/* COPIE D'OBJET */
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
);
```

```
/* AJOUT DE PROPRIÉTÉS */
CMC_return_code
(*cmc_add_properties_W)(
       CMC_object_handle       object,
       CMC_uint32              number_properties,
       CMC_property            *properties,
       CMC_extension           *add_properties_extensions
);

/* CONFIER UN OBJET */
CMC_return_code
(*cmc_commit_object_W)(
       CMC_object_handle       source_object,
       CMC_extension           *commit_object_extensions
);

/* COPIE DE DESCRIPTEUR OPAQUE D'OBJET */
CMC_return_code
(*cmc_copy_object_handle_W)(
       CMC_object_handle       source_handle,
       CMC_object_handle       *new_handle,
       CMC_extension           *copy_object_handle_extensions
);

/* CRÉATION D'OBJET MESSAGE DÉRIVÉ */
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
);

/* SUPPRESSION D'OBJETS */
CMC_return_code
(*cmc_delete_objects_W)(
       CMC_uint32              number_objects,
       CMC_object_handle       *object,
       CMC_extension           *delete_objects_extensions
);

/* SUPPRESSION DE PROPRIÉTÉS */
CMC_return_code
(*cmc_delete_properties_W)(
       CMC_object_handle       object,
       CMC_uint32              number_properties,
       CMC_id                  *property_ids,
       CMC_extension           *delete_properties_extensions
);

/* OBTENIR LE DESCRIPTEUR OPAQUE RACINE */
CMC_return_code
(*cmc_get_root_handle_W)(
       CMC_session_id          session,
       CMC_object_handle       *root_object_handle,
       CMC_extension           *get_root_handle_extensions
);

/* IDENTIFICATEUR VERS NOM */
CMC_return_code
(*cmc_identifier_to_name_W)(
       CMC_id                  identifier,
       CMC_name                *name,
       CMC_extension           *identifier_to_name_extensions
);
```

```
/* LISTE DES PROPRIÉTÉS CONTENUES */
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
);

/* LISTE DU NOMBRE DE CONCORDANCES */
CMC_return_code
(*cmc_list_number_matched_W)(
       CMC_cursor_handle       *cursor,
       CMC_uint32              *number_matches,
       CMC_extension           *list_number_matched_extensions
);

/* LISTE D'OBJETS */
CMC_return_code
(*cmc_list_objects_W)(
       CMC_cursor_handle       *cursor,
       CMC_sint32              *number_objects,
       CMC_object_handle       **objects,
       CMC_extension           *list_objects_extensions
);

/* LISTE DE PROPRIÉTÉS */
CMC_return_code
(*cmc_list_properties_W)(
       CMC_object_handle       *object,
       CMC_uint32              *number_properties,
       CMC_id                  **property_ids,
       CMC_extension           *list_properties_extensions
);

/* NOM VERS IDENTIFICATEUR */
CMC_return_code
(*cmc_name_to_identifier_W)(
       CMC_name                name,
       CMC_id                  *identifier,
       CMC_extension           *name_to_identifier_extensions
);

/* OUVERTURE DE CURSEUR */
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
);

/* OUVERTURE DE DESCRIPTEUR OPAQUE D'OBJET */
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
);

/* LECTURE DE CURSEUR */
CMC_return_code
(*cmc_read_cursor_W)(
       CMC_cursor_handle       *cursor,
       CMC_uint32              *position_numerator,
       CMC_uint32              *position_denominator,
       CMC_extension           *read_cursor_extensions
);
```

```
/* LECTURE DE PROPRIÉTÉS */
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
);

/* LECTURE DE COÛTS DE PROPRIÉTÉ */
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
);

/* RESTAURATION D'OBJET */
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
);

/* SAUVEGARDE D'OBJET */
CMC_return_code
(*cmc_save_object_W)(
       CMC_object_handle       object,
       CMC_string              file_specification,
       CMC_flags               save_flags,
       CMC_extension           *save_object_extensions
);

/* ÉMISSION D'OBJET MESSAGE */
CMC_return_code
(*cmc_send_message_object_W)(
       CMC_object_handle       message_to_send,
       CMC_extension           *send_message_object_extensions
);

/* MISE À JOUR DE LA POSITION DU CURSEUR */
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
);

/* MISE À JOUR DE LA POSITION DU CURSEUR AVEC PLACEMENT */
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
);

/* SUPERVISION D'ÉVÉNEMENT */
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
);
```

```
/* ENREGISTREMENT D'ÉVÉNEMENT */
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
);

/* RÉSILIATION D'ÉVÉNEMENT */
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
);

/* APPEL DE RAPPEL AUTOMATIQUE */
CMC_return_code
(*cmc_call_callbacks_W)(
       CMC_session_id          session,
       CMC_event               event_type,
       CMC_extension           *call_callbacks_extensions
);

/* EXPORTATION DE FLUX */
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
);

/* IMPORTER UN FICHIER VERS UN FLUX */
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
);

/* OUVERTURE DE FLUX */
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
);

/* LECTURE DE FLUX */
CMC_return_code
(*cmc_read_stream_W)(
       CMC_stream_handle       stream,
       CMC_uint32              *count,
       CMC_buffer              content_information,
       CMC_extension           *read_stream_extensions
);

/* RECHERCHE DANS UN FLUX */
CMC_return_code
(*cmc_seek_stream_W)(
       CMC_stream_handle       stream,
       CMC_enum                operation,
       CMC_uint32              *location,
       CMC_extension           *seek_stream_extensions
);
```

```
	/* ÉCRITURE DE FLUX */
	CMC_return_code
	(*cmc_write_stream_W)(
		CMC_stream_handle	*stream,
		CMC_uint32		*count,
		CMC_buffer		*content_information,
		CMC_extension		*write_stream_extensions
	);

	/* OBTENIR LA DERNIÈRE ERREUR */
	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;

/* LIER LA MISE EN ŒUVRE */
CMC_return_code
cmc_bind_implementation_W(
	CMC_guid		implementation_name,
	CMC_dispatch_table	**dispatch_table,
	CMC_extension		*cmc_bind_extensions
);

/* DÉLIER LA MISE EN ŒUVRE */
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
);

/* Constantes de fonctions */

/* ÉMISSION */
#define CMC_SEND_UI_REQUESTED			((CMC_flags) 1)

/* ÉMISSION DE DOCUMENT */
#define CMC_FIRST_ATTACH_AS_TEXT_NOTE		((CMC_flags) 2)

/* AGIR SUR */
#define CMC_ACT_ON_EXTENDED			((CMC_enum)  0)
#define CMC_ACT_ON_DELETE			((CMC_enum)  1)

/* LISTE */
#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

/* LECTURE */
#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)

/* CONSULTATION */
#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)
```

```
/* FERMETURE DE SESSION */
#define CMC_LOGOFF_UI_ALLOWED                   ((CMC_flags) 1)

/* OUVERTURE DE SESSION */
#define CMC_VERSION                             ((CMC_uint16) 100)

/* ÉNUMÉRATION D'INTERROGATION DE CONFIGURATION */
#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)

/* ÉNUMÉRATION DE CONFIGURATION DE TERMINAISON DE LIGNE */
#define CMC_LINE_TERM_CRLF                      ((CMC_enum) 0)
#define CMC_LINE_TERM_CR                        ((CMC_enum) 1)
#define CMC_LINE_TERM_LF                        ((CMC_enum) 2)

/* ÉNUMÉRATION DE CONFIGURATION DE PARAMÈTRE D'OUVERTURE DE SESSION */
#define CMC_REQUIRED_NO                         ((CMC_enum) 0)
#define CMC_REQUIRED_YES                        ((CMC_enum) 1)
#define CMC_REQUIRED_OPT                        ((CMC_enum) 2)

/* CRÉATION D'OBJET MESSAGE DÉRIVÉ */
#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)

/* LECTURE DE COÛTS DE PROPRIÉTÉ */
#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)

/* FANIONS DE RESTAURATION D'OBJET */
#define CMC_RESTORE_OBJECT_OVERWRITE            ((CMC_flags) 1)

/* FANIONS DE SAUVEGARDE D'OBJETS */
#define CMC_SAVE_OBJECT_OVERWRITE               ((CMC_flags) 1)
#define CMC_SAVE_OBJECT_NOCREATE                ((CMC_flags) 2)

/* EXPORTATION DE FLUX */
#define CMC_EXPORT_STREAM_OVERWRITE             ((CMC_flags) 1)
#define CMC_EXPORT_STREAM_NOCREATE              ((CMC_flags) 2)
#define CMC_EXPORT_STREAM_APPEND                ((CMC_flags) 3)

/* OUVERTURE DE FLUX */
#define CMC_OPEN_READ                           ((CMC_enum) 0)
#define CMC_OPEN_WRITE                          ((CMC_enum) 1)

/* RECHERCHE DANS UN FLUX */
#define CMC_SEEK_BEGINNING                      ((CMC_enum) 0)
#define CMC_SEEK_END                            ((CMC_enum) 1)
#define CMC_SEEK_CURRENT_POSITION               ((CMC_enum) 2)

/* IDENTIFICATEUR D'OBJETS DÉFINIS POUR DES JEUX DE CARACTÈRES */
#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"

/* FANIONS DE CODE RETOUR */
#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)
```

```
/* CODES RETOUR */
#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 */
```

Annexe B

Extensions de fournisseurs CMC

B.1 Extensions CMC faites par les fournisseurs

La présente Recommandation permet aux fournisseurs de réaliser des extensions dans de nombreux domaines. Les fournisseurs peuvent ajouter des extensions à certaines structure de données CMC et toutes les fonctions CMC contiennent un paramètre susceptible de véhiculer des extensions fonctionnelles. Les fournisseurs peuvent définir de nouvelles classes d'objets, étendre l'ensemble des propriétés associées à une classe d'objets, ajouter des valeurs énumérées et associer un identificateur de mise en œuvre CMC à une mise en œuvre. En outre, certaines des fonctions de la présente Recommandation ont été définies au moyen d'extensions communes définies dans la présente Recommandation dans le but de préserver une compatibilité amont avec l'interface CMC 1.0 de l'association XAPIA. De nouveaux ensembles d'extensions peuvent également être définis dans de futures versions de la présente Recommandation. Il en résulte qu'il est important de disposer d'un ensemble de directives au sujet de la dénomination et de la définition d'extensions. Ces directives sont les suivantes:

1) les domaines de valeur des codes articles pour les extensions seront attribués par blocs de 256, à des fins de création d'ensembles d'extensions, aux fournisseurs ou groupes de fournisseurs. Un fournisseur ou groupe de fournisseurs peut obtenir plus d'un domaine de code article si l'ensemble d'extensions le nécessite. L'identificateur de l'ensemble d'extensions pour l'étendue du domaine de code article sera égal à la valeur du premier bloc attribué. Cet identificateur d'ensemble d'extensions est utilisé pour interroger le service au sujet de la prise en charge d'un ensemble d'extensions donné;

   Les extensions du groupe de fournisseurs X peuvent, par exemple, posséder les valeurs 0x00000400, 0x00000900, et 0x00004300 et l'identificateur de l'ensemble d'extensions sera égal à 0x00000400 si ce bloc était le premier attribué au fournisseur X. Les applications interrogeront un service au sujet du groupe d'extensions de ce fournisseur pour savoir s'il prend en charge l'ensemble d'extensions 0x00000400;

2) un ensemble d'extensions se verra également attribuer un préfixe donné à des fins d'utilisation dans les noms de toutes les extensions dans l'ensemble d'extensions. Le format de préfixe sera le suivant:

CMC_XS_[identificateur de fournisseur]	pour l'identificateur de l'ensemble d'extensions
CMC_X_[identificateur de fournisseur]_[nom de l'extension]	pour les codes article de l'ensemble d'extensions.

   Dans l'exemple du groupe de fournisseurs X ci-dessus, si l'identificateur du fournisseur est CX, il définira ses extensions sous la forme suivante:

```
#define CMC_XS_CX          0x00000400
#define CMC_X_CX_EXT1      0x00000401
#define CMC_X_CX_EXT2      0x00000402
```

3) les ensembles d'extensions définis par la présente Recommandation se verront attribuer un numéro d'ensemble d'extensions et un préfixe d'extension par l'association API X.400. Les réalisateurs peuvent également obtenir un préfixe d'ensemble d'extensions et un bloc de codes d'extension sur demande écrite à l'association API X.400. L'Annexe B donne des numéros des ensembles d'extensions pré-définis. La prise en charge de divers ensembles d'extensions est donnée par la configuration de la mise en œuvre CMC et peut être interrogée au moyen de la fonction **cmc_query_configuration()** en utilisant l'extension CMC_X_COM_SUPPORT_EXT;

4) une valeur d'ensemble d'extensions BILATERAL a également été allouée. Toute mise en œuvre peut définir des extensions dans l'ensemble BILATERAL. Il n'est pas exigé d'enregistrement du numéro d'ensemble d'extensions. Cet ensemble est fourni afin de permettre aux réalisateurs de définir des extensions sans avoir besoin d'un enregistrement formel. Cette liberté implique que des extensions de fournisseurs différents peuvent entrer en conflit et faire obstacle à la portabilité des applications et à la coexistence de différentes mises en œuvre CMC. Le préfixe de ces extensions sera CMC_X_BLT, et l'identificateur de l'ensemble correspondant sera CMC_XS_BLT;

5) de nombreux objets sont nommés au moyen d'identificateurs globalement non-ambigus (GUID). Les identificateurs GUID peuvent être attribués par les fournisseurs sous des noms propres à ce dernier. Un fournisseur se voit également attribuer une ramification dans l'espace de nom des identificateurs GUID lorsqu'il enregistre un ensemble d'extension:

   –//XAPIA/CMC20/OBJECT CLASS/VENDOR [identificateur de fournisseur]//NONSGML [nom d'extension]//EN pour les classes d'objets,

   –//XAPIA/CMC20/PROPERTY/VENDOR [identificateur de fournisseur]//NONSGML [nom d'extension]//EN pour les noms de propriétés,

–//XAPIA/CMC20/CONTENT TYPE/VENDOR [identificateur de fournisseur]//NONSGML [nom d'extension]//EN pour les types de contenu,

–//XAPIA/CMC20/CHARSET/VENDOR [identificateur de fournisseur]//NONSGML [nom d'extension]//EN pour les jeux de caractères,

–//XAPIA/CMC20/ENCODING TYPE/VENDOR [identificateur de fournisseur]//NONSGML [nom d'extension]//EN pour les types de codage.

NOTE – La spécification d'extensions de fournisseur n'implique pas que ces extensions seront véhiculées telles quelles par les protocoles de messagerie et les passerelles. La documentation du fournisseur doit spécifier les contraintes sur les protocoles et les passerelles résultant de ces extensions.

6) Les fournisseurs peuvent également étendre les valeurs énumérées définies par la présente Recommandation. Les valeurs énumérées de 0 à 512 sont réservées pour la présente Recommandation. Le fournisseur peut réutiliser des valeurs de code article pour les valeurs énumérées. Le fournisseur doit utiliser le préfixe CMC_X_[identificateur_de_fournisseur]_[enum] pour la définition des constantes associées à ces valeurs, ce qui garantit que les noms de constantes n'entrent pas en conflit avec les noms d'extensions.

Les réalisateurs sont encouragés, en vue de minimiser les problèmes de portabilité, à spécifier les extensions d'une manière aussi générique que possible et à soumettre ces extensions comme proposition d'ajout à l'ensemble d'extensions définis par l'interface CMC. Cette procédure permettra de faire évoluer l'interface API d'appel CMC dans une direction favorable tout en continuant à assurer une portabilité maximale.

B.1.1 Extensions de fonctions

B.1.1.1 Extension CMC_X_COM_SUPPORT_EXT

Cette extension est utilisée par des applications clientes pour interroger la mise en œuvre CMC au sujet des extensions prises en charge. Ceci peut se faire avant l'établissement de la session pour obtenir une information préliminaire concernant la prise en charge avant la procédure d'ouverture de la session. Lorsque cette extension est utilisée dans la fonction **cmc_logon()**, elle indiquera également quelles sont les extensions de données que le client souhaite ajouter aux structures de données pour la durée de la session.

NOTE – Certaines mises en œuvre peuvent prendre en charge des extensions différentes en fonction du service avec lequel l'application cliente établit une session, de sorte que l'utilisation de cette extension est recommandée au moment de l'ouverture de la session afin de vérifier la prise en charge des extensions.

Cette extension doit être prise en charge par une mise en œuvre CMC, lorsqu'une extension quelle qu'elle soit est prise en charge.

UTILISÉ PAR

```
cmc_query_config()
cmc_logon()
```

ENTRÉE

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

données de l'article

Nombre d'articles se trouvant dans le tableau vers lequel pointe la référence d'article.

référence de l'article

Pointeur vers le premier élément d'un tableau de structures donnant la liste des extensions dont l'application demande la prise en charge par la mise en œuvre. La déclaration de cette structure en langage C est la suivante:

```
typedef struct {
   CMC_uint32              item_code;
   CMC_flags               flags;
} CMC_X_COM_support;
```

Le code article (*item_code*) de cette structure est positionné sur la valeur du code article au sujet duquel l'application interroge le service. Il peut s'agir d'ensembles d'extensions ou d'extensions individuelles. Un code article nul sera ignoré. Les fanions (*flags*) pour les structures utilisées en entrée sont les suivants:

CMC_X_COM_SUP_EXCLUDE – Exclure cet article lors de l'interrogation faite pour déterminer si la mise en œuvre prend en charge un ensemble d'extensions. L'article ne doit pas être attaché aux structures pour cette session, même si d'autres entités en font la demande. Ce fanion n'est utilisé qu'avec des ensembles d'extensions.

SORTIE

fanions d'extension

inchangés

données de l'article

inchangées

référence de l'article

Les fanions dans les structures sont positionnés par la mise en œuvre pour indiquer la prise en charge de l'extension. Ces fanions ne seront pas positionnés si le fanion CMC_X_COM_SUP_EXCLUDE était positionné en entrée. Les valeurs possibles sont les suivantes:

CMC_X_COM_SUPPORTED – L'extension correspondant à ce code article est prise en charge. S'il s'agit d'une extension de données passée au moment de l'ouverture de la session, elle fera partie des structures utilisées pour cette session. Dans le cas d'ensembles d'extension, la fonction demandée et les extensions de données de l'ensemble sont prises en charge.

CMC_X_COM_NOT_SUPPORTED – Le code article n'est pas pris en charge. Dans le cas d'ensembles d'extension, la totalité des fonctions et des extensions de données demandées ne sont pas toutes prises en charge. S'il s'agit d'une extension de données ou un ensemble d'extensions contenant des extensions de données, les données ne seront pas attachées aux structures pour cette session.

CMC_X_COM_DATA_EXT_SUPPORTED – Uniquement pour des ensembles d'extensions. Ce fanion peut être renvoyé par la mise en œuvre pour indiquer que toutes les extensions de données de l'ensemble sont prises en charge, mais pas toutes les extensions de fonction demandées. Comme dans le cas du fanion CMC_X_COM_SUPPORTED, si ce fanion est renvoyé au moment de l'appel à la fonction **cmc_logon()**, les extensions de donnée feront partie des structures de données pendant la session concernée.

CMC_X_COM_FUNC_EXT_SUPPORTED – Uniquement pour des ensembles d'extensions. Ce fanion peut être renvoyé par la mise en œuvre pour indiquer que les extensions de fonction de l'ensemble qui ont été demandées sont prises en charge, mais pas toutes les extensions de données demandées. Contrairement au cas du fanion CMC_X_COM_SUPPORTED, si ce fanion est renvoyé au moment de l'appel à la fonction **cmc_logon(),** les extensions de données NE seront PAS incluses dans les structures de données pour cette session et devront être demandées de manière explicite.

B.1.1.2 Extension CMC_X_UI_ID_EXT

Cette extension est utilisée par des applications clientes pour spécifier aux fonctions CMC l'information propre à la plate-forme concernant l'interface utilisateur donnée. L'information d'interface utilisateur peut être utilisée par la mise en œuvre CMC pour mener à bien des dialogues utilisateur dans le cas de la résolution d'arguments supplémentaires concernant l'appel CMC ou pour résoudre toute autre problème pouvant se poser lorsque le service traite la fonction. Dans un environnement utilisant des fenêtres, par exemple, cette information peut être le descripteur opaque de la fenêtre parente de l'application appelante.

NOTE – La prise en charge de l'interface utilisateur n'est pas exigée pour les mise en œuvre CMC, et la fourniture de cette interface pour une fonctionnalité donnée n'implique pas nécessairement que l'interface utilisateur est disponible pour toutes les fonctionnalités CMC.

Les codes erreur générés à la suite de l'utilisation de cette extension de fonction seront renvoyés sous la forme de codes erreur dans le processus normal de renvoi du code retour.

UTILISÉ PAR

Toutes les fonctions de l'interface CMC complète

ENTRÉE

fanions d'extension

Tous les fanions d'extension CMC sont valides. Les fanions non spécifiés doivent toujours avoir une valeur nulle. Les fanions supplémentaires utilisés par cette fonction sont les suivants:

CMC_X_ERROR_UI_ALLOWED

CMC_X_ERROR_UI_ALLOWED – Positionné si la fonction peut faire un affichage au moyen de l'interface utilisateur lorsqu'elle rencontre des erreurs pouvant être récupérées. Si ce fanion n'est pas positionné, la fonction peut ne pas faire d'affichage et renverra un code erreur. Ce fanion est valide pour toutes les fonctions CMC prenant cette extension en charge.

données de l'article

néant

référence de l'article

Pointeur vers un identificateur d'une interface utilisateur (par exemple, une fenêtre de dialogue), à des fins d'utilisation pour la résolution de toute question qui pourrait entraîner une erreur si elle faisait défaut, ainsi que pour demander à l'utilisateur une information supplémentaire en cas de besoin.

SORTIE

fanions d'extension

inchangés

données de l'article

inchangées

référence de l'article

inchangée

B.1.1.3 CMC_X_COM_CONFIG_DATA

Obtient toute valeur disponible dans une structure au moyen de la fonction **cmc_query_configuration()**.

UTILISÉ PAR

cmc_query_configuration()

ENTRÉE

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini

données de l'article

néant

référence de l'article

NULL

SORTIE

fanions d'extension

Le fanion CMC_EXT_OUTPUT sera positionné si une structure est renvoyée correctement.

données de l'article

inchangées

référence de l'article

Pointeur vers une structure contenant toute l'information disponible fournie par l'appel d'interrogation de configuration. La déclaration de cette structure en langage C est la suivante:

```
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;
```

Les définitions de chacun des membres de la structure correspondent aux données renvoyées dans l'argument de référence par la fonction **cmc_query_configuration()** pour la valeur de même nom de l'argument d'article. Cette structure doit être libérée en utilisant la fonction **cmc_free().**

B.1.1.4 CMC_X_COM_PROPERTY_HINTS

Cette extension de fonction fournit à la fonction **cmc_list_objects()** un renseignement au sujet des propriétés dont l'appelant aura besoin à court terme. Ce renseignement permet aux mises en œuvre d'optimiser l'extraction de propriétés en obtenant toutes les propriétés en même temps.

UTILISÉ PAR

cmc_list_objects()

ENTRÉE

fanions d'extension

Tous les fanions d'extension sont valides. Les fanions non spécifiés doivent toujours avoir une valeur nulle. Aucun fanion nouveau n'est défini.

données de l'article

Nombre de noms de propriétés CMC figurant dans le tableau de structures sur lequel pointe la référence de l'article.

référence de l'article

Pointeur vers un tableau de noms de propriétés CMC. Ces identificateurs spécifient les propriétés au sujet desquelles le renseignement est fourni.

SORTIE

fanions d'extension

inchangés

données de l'article

inchangées

référence de l'article

inchangée

B.1.1.5 CMC_X_COM_CAN_SEND_RECIP

Vérifie si le service de messagerie est prêt à émettre vers le destinataire spécifié.

UTILISÉ PAR

cmc_look_up()

ENTRÉE

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

données de l'article

néant

référence de l'article

NULL

Le paramètre destinataire en entrée de la fonction **cmc_look_up()** contiendra le destinataire au sujet duquel le service est questionné. L'extension ne recherchera que le premier destinataire, qui sera également le seul renvoyé, même s'il y en existe plusieurs.

SORTIE

fanions d'extension

inchangés

données de l'article

Positionnées sur CMC_X_COM_NOT_READY si aucun transport n'est disponible pour ce type de destinataire, CMC_X_COM_READY si l'émission vers le destinataire est possible immédiatement et CMC_X_COM_DEFER si le message sera accepté mais différé jusqu'à ce qu'un transport soit prêt.

référence de l'article

inchangée

B.1.1.6 CMC_X_COM_SAVE_MESSAGE

Sauvegarde une structure de message dans la boîte aux lettres en entrée.

UTILISÉ PAR

cmc_act_on()

ENTRÉE

fanions d'extension

Doivent contenir CMC_EXT_REQUIRED afin d'indiquer que la fonction doit effectuer une action de sauvegarde et non une action de suppression. Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

données de l'article

néant

référence de l'article

Pointeur vers la structure de message dans la boîte aux lettres en entrée. Le fanion CMC_MSG_UNSENT de ce message sera positionné par la mise en œuvre CMC afin d'indiquer qu'il n'a pas été émis.

Le paramètre opération de la fonction **cmc_act_on()** doit être positionné sur CMC_ACT_ON_EXTENDED pour indiquer que l'opération est contenue dans les extensions.

SORTIE

fanions d'extension

Le fanion sera positionné CMC_EXT_OUTPUT pour indiquer que le message a été sauvegardé correctement et la référence de message renvoyée.

données de l'article

inchangé

référence de l'article

Pointeur vers référence de message du message sauvegardé dans la boîte aux lettres en entrée. Ce pointeur doit être libéré en utilisant la fonction **cmc_free()**.

B.1.1.7 CMC_X_COM_SENT_MESSAGE

Renvoie une structure de message contenant toutes les informations concernant le message qui vient d'être émis. Ceci est utile pour accéder à une information qui a été placée dans la structure du message par l'intermédiaire de l'interface utilisateur plutôt que par l'application appelante.

UTILISÉ PAR

cmc_send()

ENTRÉE

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

données de l'article

néant

référence de l'article

NULL

SORTIE

fanions d'extension

Le fanion CMC_EXT_OUTPUT est positionné si la référence de l'événement contient un pointeur vers un message.

données de l'article

inchangées

référence de l'article

Pointeur vers une structure de message contenant toute l'information au sujet du message qui vient d'être émis. Ce pointeur doit être libéré en utilisant la fonction **cmc_free()**.

B.1.1.8 CMC_X_COM_PROP_STATUS

Cette extension de fonction indique que l'opération effectuée doit renvoyer un statut par propriété. Une erreur résultant d'une tentative de modification ou de suppression de propriété est appelée problème de propriété. Cette extension permet à l'appelant de recevoir des comptes rendus au sujet de problèmes de propriété qui se manifestent lorsqu'une opération portant sur des propriétés multiples rencontre des problèmes qui l'empêchent de traiter certaines de ces propriétés.

UTILISÉ PAR

cmc_add_properties()

cmc_delete_properties()

ENTRÉE

fanions d'extension

Tous les fanions d'extension sont valides. Les fanions non spécifiés doivent toujours avoir une valeur nulle. Aucun fanion nouveau n'est défini.

données de l'article

néant

référence de l'article

NULL

SORTIE

fanions d'extension

Le fanion CMC_EXT_OUTPUT est positionné si un compte rendu de problème de propriété est rapporté.

données de l'article

Nombre d'articles du tableau sur lequel pointe la référence de l'article. Une valeur nulle indique l'absence de compte rendu de problème de propriété.

référence de l'article

Pointeur vers un tableau de structures donnant la liste des problèmes de propriétés rapportés. La déclaration de cette structure en langage C est la suivante:

```
typedef struct {
   CMC_uint32              index;
   CMC_id                  id;
   CMC-return_code         error_code;
} CMC_X_COM_prop_problem;
```

dans laquelle:

- la variable index spécifie la valeur de la propriété impliquée dans les tableaux de *propriétés* ou *d'identificateurs de propriété* fournis en entrée à la fonction;
- la variable id spécifie la propriété en question;
- la variable error_code spécifie l'erreur rencontrée lors du traitement de la demande concernant cette propriété.

Le tableau est alloué par le service, et doit être libéré en utilisant la fonction **cmc_free()**.

La fonction renvoie le code erreur CMC_E_PROPERTY_PROBLEMS lorsque cette extension rencontre un problème de propriété. On peut, dans un tel cas, faire l'hypothèse que toute propriété pour laquelle aucun problème n'a été rapporté a été traitée correctement.

ERREURS

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 Extensions de données

B.1.2.1 CMC_X_COM_TIME_RECEIVED

Extension de données pour une structure de temps contenant l'instant de remise du message.

Le code article est passé lors de l'ouverture de session dans le tableau CMC_X_COM_SUPPORT_EXT afin d'indiquer que ce membre de données doit être attaché pour la durée de la session aux structures de message et de résumé de message.

UTILISÉ PAR

Message CMC
Résumé de message CMC

ENTRÉE

Cette extension est ignorée pour une structure de message en entrée.

SORTIE

fanions d'extension

NULL

données de l'article

néant.

référence de l'article

Pointeur vers une structure de temps indiquant l'instant de réception du message. Prière de se référer à la description de la structure CMC_time pour plus de détails.

B.1.2.2 CMC_X_COM_RECIP_ID

Extension de données permettant d'ajouter à la structure un identificateur opaque non-ambigu de destinataire. Elle est fournie par la mise en œuvre au moment de la résolution de nom et peut être utilisée pour éviter, dans certains services, de nouvelles résolutions de noms pendant l'émission. Ceci est analogue à la référence de message.

Le code article est transmis dans le tableau CMC_X_COM_SUPPORT_EXT au moment de l'ouverture de la session afin d'indiquer que ce membre de données doit être attaché à la structure de destinataire pour la durée de la session.

UTILISÉ PAR

Destinataire CMC

ENTRÉE

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

données de l'article

longueur de l'identificateur de destinataire.

référence de l'article

pointeur vers l'identificateur de destinataire.

SORTIE

fanions d'extension

inchangés.

données de l'article

longueur de l'identificateur de destinataire.

référence de l'article

pointeur vers l'identificateur de destinataire.

B.1.2.3 CMC_X_COM_ATTACH_CHARPOS

Extension de données prenant en charge l'affichage d'une représentation graphique de l'attachement dans la note de texte du message. L'extension contient la position de caractère pour la représentation.

Le code article est passé lors de l'ouverture de session dans le tableau CMC_X_COM_SUPPORT_EXT afin d'indiquer que ce membre de données doit être attaché pour la durée de la session aux structures de message et de résumé de message.

UTILISÉ PAR

Attachement CMC

ENTRÉE

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

données de l'article

Déplacement de base nulle de la position de caractère de l'attachement au sein des données de la note de texte.

NOTE – Il s'agit d'un déplacement exprimé en caractères et non en octets, cette distinction étant importante en cas d'utilisation de jeux de caractères à plusieurs octets.

référence de l'article

NULL

SORTIE

fanions d'extension

inchangés.

données de l'article

Déplacement de base nulle de la position de caractère de l'attachement au sein des données de la note de texte.

référence de l'article

inchangée.

B.1.2.4 CMC_X_COM_PRIORITY

Extension de données pour la priorité du message.

Le code article est passé lors de l'ouverture de session dans le tableau CMC_X_COM_SUPPORT_EXT afin d'indiquer que ce membre de données doit être attaché pour la durée de la session aux structures de message et de résumé de message.

UTILISÉ PAR

Message CMC
Résumé de message CMC

ENTRÉE

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

données de l'article

positionnées sur l'une des valeurs CMC_X_COM_URGENT, CMC_X_COM_NORMAL ou CMC_X_COM_LOW en fonction de l'urgence du message.

référence de l'article

NULL

SORTIE

fanions d'extension

inchangés

données de l'article

positionnées sur l'une des valeurs CMC_X_COM_URGENT, CMC_X_COM_NORMAL ou CMC_X_COM_LOW en fonction de l'urgence du message.

référence de l'article

inchangée

B.2 Résumé des déclarations en langage C de l'ensemble d'extensions

Le présent sous-paragraphe donne la liste des déclarations en langage C définissant l'interface CMC pour l'ensemble commun d'extensions.

Les déclarations regroupées ici constituent le contenu d'un ficher d'en-tête qui doit être fourni aux programmeurs d'application. La référence de ce fichier est **<xcmcext.h>**. Les symboles figurant dans les déclarations sont les seuls pour lesquels le service fournit une visibilité à l'application.

```
/* DÉCLARATIONS D'EXTENSIONS COMMUNES */

/* IDENTIFICATEUR D'ENSEMBLE D'EXTENSIONS */

#define CMC_XS_COM                    ((CMC_uint32) 0)

/* EXTENSIONS DE FONCTION */

/* Interrogation pour la prise en charge de l'extension par la mise en œuvre */
```

```
#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)

/* Obtenir en retour une structure contenant les données de configuration */

#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;

/* Déterminer si l'émission peut être faite pour un destinataire */

#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)

/* Sauvegarder un message dans la boîte aux lettres en entrée */

#define CMC_X_COM_SAVE_MESSAGE      ((CMC_uint32) 19)

/* Obtenir en retour une structure pour le message venant d'être émis */

#define CMC_X_COM_SENT_MESSAGE      ((CMC_uint32) 20)

/* EXTENSIONS DE DONNÉES */

/* attacher les données reçues aux structures message et résumé de message */

#define CMC_X_COM_TIME_RECEIVED     ((CMC_uint32) 128)

/* attacher un identificateur non-ambigu */
/* aux structures destinataires résolues */

#define CMC_X_COM_RECIP_ID          ((CMC_uint32) 129)

/* indiquer la position de caractère au sein du texte du message afin */
/* d'afficher une icône associée à un attachement donné               */

#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 Ensemble d'extensions X.400

Les identificateurs d'ensemble d'extensions qui suivent sont en cours d'enregistrement auprès de l'association XAPIA pour une utilisation avec le service de messagerie 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 Structure de compte rendu CMC

La structure suivante, définie en langage C, est utilisée dans l'extension 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;

/* type de compte rendu */

      #define CMC_X400_DR       ((CMC_enum) 0)
      #define CMC_X400_NDR      ((CMC_enum) 1)

/* fanions de compte rendu */

      #define CMC_REPORT_LAST_ELEMENT ((CMC_flags) 0x80000000)
```

B.2.1.2 Code erreur: CMC_EX_X400_STD

Un nouveau code erreur est défini afin de qualifier d'une manière plus complète l'échec d'une fonction CMC résultant d'une condition d'exception X.400, d'interruption X.400 ou d'erreur X.400. Ce code sera utilisé par les 16 bits d'ordre supérieur, le code retour CMC afin d'indiquer que l'erreur est liée à des services de messagerie mettant en œuvre les Recommandations X.400-X.420 (1988).

La définition de cette erreur en langage C est la suivante:

```
      #define CMC_EX_X400_STD   ((CMC_uint16) 400)
```

Il en résulte que si une mise en œuvre CMC souhaite classifier une situation d'erreur qui est causée par le service de messagerie X400 sous-jacent et si des extensions CMC optionnelles liées au service X.400 sont renvoyées, le code retour CMC peut être positionné de la manière suivante:

code_retour_CMC.<16 bits d'ordre inférieur> = CMC_E_FAILURE, ou le code erreur le mieux approprié.
code_retour_CMC.<16 bits d'ordre supérieur> = CMC_EX_X400_STD

B.2.2 Extensions supplémentaires pour un mappage CMC simple vers X400

B.2.2.1 CMC_X_X400_ERROR

Si la fonction CMC échoue à la suite de l'échec du service X.400, l'erreur résultant du fonctionnement du service X.400 est renvoyée à l'application, de manière à ce que cette dernière soit en mesure de trouver l'origine première de l'erreur. Cette extension contient les erreurs spécifiques définies par les Recommandations X.400-X.420 (1988). Prière de se référer au document correspondant pour l'explication et la valeur du code erreur.

NOTE – Si l'application CMC souhaite que la mise en œuvre CMC renvoie cette extension en cas d'apparition d'une erreur X.400, elle doit fournir la mémoire nécessaire à cette extension lorsqu'elle invoque la fonction CMC; faute de quoi la mise en œuvre CMC ne pourrait renvoyer cette extension, car l'argument extension présent dans toute fonction CMC représente seulement l'adresse d'un tampon alloué par l'application. Si la spécification CMC V1.0 n'est pas modifiée afin de permettre l'utilisation de l'argument extension de la fonction comme argument en entrée et en sortie, une variante consisterait à faire fournir par la mise en œuvre CMC une nouvelle fonction "information d'erreur" permettant à l'application d'obtenir dans cette extension le détail de l'erreur après un échec d'une fonction CMC avec une erreur liée au service X.400.

UTILISÉ PAR

cmc_act_on(), cmc_list(), cmc_logon(), cmc_logoff(), cmc_read(), cmc_send()

SORTIE

code article

CMC_X_X400_ERROR

données de l'article

données_de_l'article.<16 bits d'ordre supérieur> = numéro d'opération défini par X.400.

données_de_l'article.<16 bits d'ordre inférieur > = codes retour de l'opération définis par X.400.

référence de l'article

NULL

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

B.2.2.2 CMC_X_X400_MSG_PARENT

La mémoire de messages X.400 prend en charge des messages imbriqués au moyen du concept de message parent et enfant. Dans l'exemple du corps d'un message interpersonnel contenant un message interpersonnel réexpédié, ce dernier est un message enfant et le message interpersonnel effectuant la réexpédition est le message parent, ou, dans le cas d'un compte rendu, le message interpersonnel renvoyé est le message enfant et le compte rendu lui-même est le message parent. Une nouvelle extension sera utilisée afin de permettre à l'application de déterminer si un message est un parent ou un enfant.

Identification utilisée pour indiquer si la référence de message du message CMC ou du résumé de message CMC est un message parent au sens de la Recommandation X.413 ou un message enfant. Cette extension sera renvoyée si le message associé est un message parent.

UTILISÉ PAR

Message CMC et résumé de message CMC

SORTIE

code article

CMC_X_X400_MSG_PARENT

données de l'article

numéro de séquence parent X.413 pour le message enfant.

référence de l'article

NULL

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

B.2.2.3 CMC_X_X400_MSG_ID

Le service X.400 crée, lors de l'émission d'un message, un identificateur non ambigu pour ce message appelé identificateur MTS. Cet identificateur est utilisé pour effectuer un suivi des messages et rendre compte de leur remise ou non-remise. Cet identificateur est renvoyé à l'application CMC au moyen de l'extension d'identification de message à l'occasion des opérations de lecture, de liste et d'émission du message. L'application peut ainsi, par une action appropriée, répondre à un message donné ou effectuer une action sur ce message.

Identificateur non ambigu fourni par le service de messagerie sous-jacent lorsque l'application effectue une opération de lecture, de liste ou d'émission du message.

UTILISÉ PAR

Message CMC, résumé de message CMC et **cmc_send()**.

Lecture et liste de messages:

l'extension d'identification de message est attachée à la structure, lorsqu'une structure de message (ou une structure de résumé de message) est renvoyée après un appel de la fonction **cmc_read()** [ou **cmc_list()**]. Les données de l'article pour l'identification de message ne sont pas significatives et sont donc nulles. La référence de l'article pointe vers une structure de chaîne de caractères qui est allouée par le service et contient une forme lisible de l'identificateur MTS non-ambigu.

Emission de messages:

lorsque l'utilisateur émet un message au moyen de la fonction **cmc_send(),** le service CMC a la possibilité de renvoyer à l'appelant dans la structure d'extension d'émission l'identificateur MTS alloué pour ce message, si l'appelant alloue de la mémoire pour l'extension avec le code article positionné sur CMC_X_X400_MSG_ID. Le service CMC ne renverra pas l'identificateur MTS si cette extension est absente. Le service CMC renvoie l'identificateur MTS en allouant une chaîne de caractères CMC contenant les données requises et en attachant

un pointeur pour ces données dans la référence de l'article. Les fanions d'extension contiennent un fanion CMC_EXT_OUTPUT positionné. Ceci indique à l'appelant qu'il doit libérer la référence de l'article, une fois qu'il l'a utilisée, en appelant la fonction **cmc_free()**.

SORTIE

code article

CMC_X_X400_MSG_ID

données de l'article

NULL

référence de l'article

pointeur vers une chaîne de caractères CMC d'un identificateur MTS

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

B.2.2.4 CMC_X_X400_MSG_REPORT_ID

Lorsqu'il lit un compte rendu de remise ou de non-remise, le service de messagerie sous-jacent renvoie un identificateur non-ambigu pour le compte rendu (identificateur MTS), qui diffère de celui du message original faisant l'objet du compte rendu.

Identificateur non-ambigu d'un compte rendu de remise ou de non-remise fourni par le service de messagerie sous-jacent lorsque le compte rendu est lu par l'application.

UTILISÉ PAR

Message CMC

SORTIE

code article

CMC_X_X400_MSG_REPORT_ID

données de l'article

NULL

référence de l'article

Pointeur vers une chaîne de caractères donnant la forme lisible de l'identificateur MTS non-ambigu.

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

B.2.2.5 CMC_X_X400_REPORT

Cette extension est utilisée pour véhiculer, à destination de l'application CMC, l'information X.400 spécifique de remise ou de non-remise lorsque l'information de base à renvoyer est un compte rendu X.400. Cette extension est renvoyée comme extension du message CMC.

Renvoi d'une information spécifique de compte rendu de remise ou de non-remise définie dans la Recommandation X.411 (1988). Prière de se référer à ce document en ce qui concerne les définitions et les valeurs des codes motif et diagnostic.

UTILISÉ PAR

Message CMC

SORTIE

code article

CMC_X_X400_REPORT

données de l'article

NULL

référence de l'article

Pointeur vers la structure de compte rendu CMC.

fanions d'extension

Tous les fanions d'extension sont valides, aucun fanion nouveau n'est défini.

B.2.3 Autres ensembles d’extension

D'autres ensembles d'extensions seront définis par l'association XAPIA et des groupes de fournisseurs afin de prendre en charge divers protocoles de messagerie. Des ensembles d'extensions sont en cours de définition pour les services de télécopie G3, de télécopie G3-64, de télécopie G4, de télex et télétex utilisant la Recommandation T.611. Prière de contacter l'association XAPIA pour déterminer quels sont les ensembles d'extensions disponibles.

B.2.4 Informations propres aux plates-formes, y compris les associations d'exécution

Les réalisateurs de mises en œuvre CMC sont encouragés à fournir des interfaces d'association de fonctions d'exécution à leurs mises en œuvre de service CMC. Ces interfaces sont en général dépendantes de la plate-forme ou du système d'exploitation ou des deux. Le présent sous-paragraphe fournit un certain nombre de prescriptions générales et de prescriptions propres à la plate-forme pour plusieurs plates-formes et systèmes d'exploitation usuels.

Sauf indication contraire, les définitions suivantes s'appliquent à l'ensemble des plates-formes:

```
octet                              CMC_sint8
entier à 16 bits                   CMC_sint16
entier long à 32 bits              CMC_sint32
entier à 16 bits sans signe        CMC_uint16
entier long à 32 bits sans signe   CMC_uint32
pointeur à 32 bits                 CMC_buffer
pointeur de caractère à 32 bits    CMC_string
CMC_uint32                         CMC_ui_id
CMC_uint32                         CMC_session_id
```

B.2.4.1 Associations implicites et explicites

Toutes les fonctions de l’interface API CMC doivent pouvoir faire l’objet de liaisons implicites ou explicites. Les liaisons implicites construisent une association entre l'application et la mise en œuvre du service CMC dans l'application. Les liaisons explicites nécessitent la présence dans l’application d’un code exécutable qui crée l'association avec une mise en œuvre du service CMC.

Il est également recommandé que toutes les fonctions d'extension soient chargées d'une manière explicite, car leur absence dans certaines mises en œuvre CMC peut faire obstacle au chargement de l'application.

Des mécanismes de liaisons statiques et dynamiques sont définis ci-dessous pour diverses plates-formes usuelles.

B.2.4.2 Associations pour la plate-forme Macintosh d’Apple

Les applications doivent utiliser les conventions d'appel Pascal et des pointeurs plats à 32 bits pour les liaisons statiques faisant appel à une mise en œuvre CMC sur la plate-forme Macintosh d'Apple.

Prière de contacter Apple Computer, Inc. en ce qui concerne les liaisons dynamiques.

La mise en œuvre CMC doit toujours tenter de fournir les chaînes de caractères internationales Apple (ISTRING).

B.2.4.3 Liaisons pour la plate-forme MS-DOS

Les applications doivent utiliser pour les liaisons statiques des appels du type "far", les conventions d'appel du langage C et des pointeurs segmentés de 32 bits pour faire appel à une mise en œuvre CMC sous MS-DOS. Cette convention est compatible avec le modèle de mémoire "large" du compilateur C de Microsoft. Toute modification ultérieure de ce mécanisme sera publiée par Microsoft.

La mise en œuvre CMC doit toujours tenter de fournir les pages de code 437 ou 850.

B.2.4.4 Liaisons pour la plate-forme MS-Windows 3.x

Les mises en œuvre CMC sous MS-Windows 3.x doivent utiliser, pour des liaisons dynamiques, des bibliothèques de liaison dynamiques (DLL, *dynamic linked libraries*) et des liaisons par nom avec les fonctions CMC.

Pour déterminer si un service CMC est disponible au moment de l'exécution, les applications doivent appeler la fonction GetProfileInt() pour rechercher la variable CMC dans la clause [MAIL] du fichier WIN.INI. Si cette variable est présente et non-nulle, elle indique qu'une bibliothèque dynamique CMC.DLL est disponible. Les fonctions CMC ne peuvent pas être appelées si la variable CMC n'est pas trouvée ou si sa valeur est nulle. Toute modification future de ce mécanisme sera publiée par Microsoft.

Les fonctions CMC doivent être accédées par un appel du type "far" en utilisant les conventions d'appel Pascal et des pointeurs segmentés de 32 bits du type "far".

Les structures CMC seront cadrées sur des frontières multiples de 4 octets (32 bits). Ceci ne s'applique pas aux champs d'octets dans la structure de temps ou dans la structure de chaîne de caractères avec comptage.

La mise en œuvre CMC doit toujours tenter de fournir la page de code 1252.

B.2.4.5 Liaisons pour la plate-forme MS-Windows NT

Les mises en œuvre CMC sous MS-Windows NT doivent utiliser, pour des liaisons dynamiques, des bibliothèques de liaison dynamiques (DLL) et des liaisons par nom avec les fonctions CMC.

Pour déterminer au moment de l'exécution si un service CMC est disponible, les applications doivent consulter la base de registres pour établir si le service CMC est disponible. Le mécanisme exact à utiliser pour cette consultation sera publié par Microsoft.

Les fonctions CMC doivent être appelées en utilisant les conventions d'appel STDCALL.

B.2.4.6 Liaisons DLL pour les plates-formes OS/2 1.x et 2.x 16 bits

Pour établir des liaisons dynamiques sous OS/2 1.x ou 2.x 16 bits, les mises en œuvre CMC doivent utiliser des bibliothèques de liaison dynamiques (DLL) et des liaisons par nom avec les fonctions.

Pour déterminer au moment de l'exécution si un service CMC est disponible, les applications doivent appeler la fonction WinQueryProfileInt() pour rechercher la variable CMC dans la clause [MAIL] du fichier OS2.INI. Cette variable indiquera si la bibliothèque DLL est du type 16 bits ou 32 bits. Si cette variable est présente et non-nulle, elle indique qu'une bibliothèque dynamique CMC.DLL est disponible. Les fonctions CMC ne peuvent pas être appelées si la variable CMC n'est pas trouvée ou si sa valeur est nulle. Toute modification future de ce mécanisme sera publiée par IBM.

L'accès aux fonctions CMC doit être fait par un appel du type "far" en utilisant les conventions d'appel système et des pointeurs segmentés de 32 bits du type "far".

La mise en œuvre CMC doit toujours tenter de fournir la page de code 850.

B.2.4.7 Liaisons DLL pour la plate-forme OS/2 2.0 32 bits

Pour établir des liaisons dynamiques, les mises en œuvre CMC sous OS/2 2.0 32 bits doivent utiliser des bibliothèques de liaisons dynamiques (DLL) et des liaisons par nom avec les fonctions.

Pour déterminer au moment de l'exécution si un service CMC est disponible, les applications doivent appeler la fonction WinQueryProfileInt() pour rechercher la variable CMC dans la clause [MAIL] du fichier OS2.INI. Cette variable indiquera si la bibliothèque DLL est du type 16 bits ou 32 bits. Si cette variable est présente et non-nulle, elle indique qu'une bibliothèque dynamique CMC.DLL est disponible. Les fonctions CMC ne peuvent pas être appelées si la variable CMC n'est pas trouvée ou si sa valeur est nulle. Toute modification future de ce mécanisme sera publiée par IBM.

L'accès aux fonctions CMC doit être fait par un appel du type "far" en utilisant les conventions d'appel système et des pointeurs plats de 32 bits du type "far".

La mise en œuvre CMC doit toujours tenter de fournir la page de code 850

B.2.4.8 Liaisons pour la plate-forme UNIX SVR4

Pour établir des liaisons dynamiques, les mises en œuvre doivent se conformer à la spécification d'interface binaire d'application (ABI, *application binary interface*) UNIX System V Release 4.0, définie pour le système UNIX System V Release 4.0, et établir des liaisons par nom avec les fonctions.

Pour déterminer au moment de l'exécution si un service CMC est disponible, les applications doivent rechercher la mise en œuvre CMC sur le chemin d'accès absolu /usr/lib/XAPI/libCMC.so. La mise en œuvre pour le système sera localisée dans ce répertoire. Toute modification future de ce mécanisme sera publiée par le fournisseur de la plate-forme.

Les fonctions et structures CMC doivent utiliser les conventions d'appel système.

La mise en œuvre CMC doit toujours tenter de fournir la page de code 850.

B.2.5 Utilisation des services de l'ossature X.400 par l'interface CMC simple

Le présent sous-paragraphe décrit de quelle manière des fonctions de la version 1.0 de l'interface API d'appel commun de messagerie (CMC, *common messaging call*) sont mappées vers un système sous-jacent de messagerie X.400 au niveau de l'interface de mémoire de messages (MS, *message store*), et de quelle manière les messages CMC sont mappés vers des messages X.400. La présente Recommandation ne traite pas des points suivants:

mappage de l'annuaire X.500 (adresses) pouvant être accédé au moyen de la fonction cmc_look_up;

dialogue au niveau de l'interface utilisateur (UI, *user interface*) qui est une option de la fonction **cmc_send_documents()**, car celle-ci ne joue pas un rôle fondamental dans l'interaction entre l'application CMC utilisant la messagerie et le service de messagerie X.400.

La présente Recommandation fait l'hypothèse que le lecteur est familiarisé avec l'élément de service d'opérations distantes (ROSE, *remote operation service element*), les protocoles et éléments de service P1, P2, P22, P3 et P7, ainsi qu'avec les spécifications et les objectifs de la version 1.0 de l'interface API d'appel CMC. Les Recommandations suivantes sont utilisées en référence:

– Recommandations X.200-X.219 (Modèle et notation OSI, définition du service);

– Recommandations X.220-X.229 (Spécifications du protocole OSI);

– Recommandations X.400-X.420 (Système MHS X.400 1984);

– Recommandations X.400-X.420 (Système MHS X.400 1988);

– Recommandation F.401 (1988), Annexe B (Représentation d'adresses expéditeur et destinataire pour un usage humain) ou son équivalent de contenu identique;

– ISO/CEI 10021-2:1990/Amd.1 Annexe F (Représentation d'adresses expéditeur et destinataire pour un usage humain);

– Interface API CMC version 1.0 de l'association XAPIA;

– Interface API CMC version 2.0 de l'association XAPIA.

Le mappage entre la version 1.0 de l'interface CMC et la messagerie est décrit dans la présente Recommandation dans un double but:

- conformément aux objectifs simples et de haut niveau de la version 1.0 de l'interface CMC, le mappage n'utilise pas la totalité de l'ensemble des fonctions X.400, de sorte que seul un profil de base est recommandé;
- il représente, sous l'aspect principal de leur interopérabilité, un compromis entre différentes mises en œuvre de la version 1.0 de l'interface CMC utilisant comme transport de message divers systèmes de messagerie X.400.

B.2.5.1 Introduction

L'interface de programme d'application du système commun de messagerie fournit un ensemble de fonctions de haut niveau permettant à des applications utilisant la messagerie d'émettre et de recevoir des messages électroniques. Cette interface nécessite une prise en charge par des services de messagerie. Un des principaux services de messagerie est le système de messagerie MHS X.400 défini par l'OSI. Ce document est destiné aux personnes souhaitant intégrer l'interface API d'appel CMC avec le système MHS X.400.

La vue et les capacités de toute mise en œuvre CMC doivent être mappées avec la vue et les capacités du service de messagerie sous-jacent. L'association XAPIA propose un certain nombre de directives en vue de maximiser l'interopérabilité entre applications CMC utilisant des services de messagerie sous-jacents différents. Les chaînes de caractères des messages doivent être converties chaque fois que possible vers des chaînes de caractères internationales et les types d'attachement de messages doivent être mappés vers des types d'attachement reconnus d'une manière générale chaque fois que cela est approprié ou possible.

Il convient de comprendre et d'utiliser de la manière la plus judicieuse le service de messagerie sous-jacent (MHS, *message handling system*) afin de réaliser le mappage. La suite de ce document traite des points suivants:

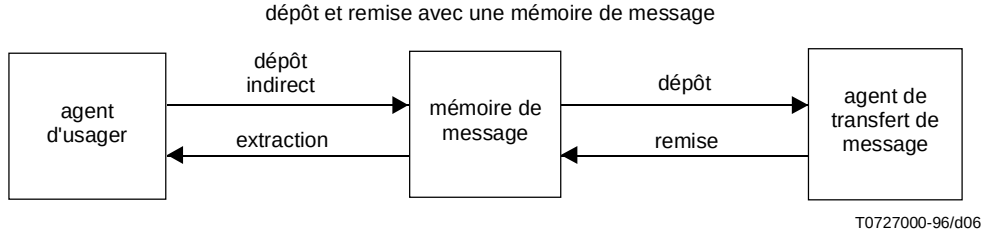
- présentation à haut niveau du système de la Recommandation X.400;
- présentation générale utilisant un mappage simple de l'interface API d'appel CMC vers le service de messagerie X.400, ainsi que les options et les extensions pour la prise en charge d'un ensemble plus riche de fonctions pour les application utilisant la messagerie et les applications tributaires de la messagerie;
- profil de mappage de base pour une interface API CMC simple fournissant une interopérabilité simple en vue de l'émission et de la réception de messages pour un certain nombre de services de communication MHS X.400.

B.2.5.2 Présentation à haut niveau du système X.400

Les services de messagerie fournis par le système de messagerie X.400 se constituent d'un service de messagerie interpersonnelle (IPM, *interpersonal messaging*) et d'un service de transfert de messages. Ces services permettent à des abonnés d'échanger des messages par commutation de messages. Le service de messagerie définit un ensemble de types de messages et de capacités qu'un expéditeur peut envoyer à des destinataires.

Un expéditeur prépare un message avec l'assistance d'un agent d'usager (UA, *user agent*). L'agent d'usager est une application qui interagit avec le système de transfert de messages (MTS, *message transfer system*) afin de déposer des messages. Le système de transfert de messages (MTS) se constitue d'un certain nombre d'agents de transfert de messages (MTA, *message transfer agent*). La coopération entre agents MTA assure un relais vers les agents d'usager destinataires qui mettent ensuite ces messages à la disposition des destinataires souhaités.

La version 1988 de la messagerie X.400 contient une fonction optionnelle mémoire de messages (MS). Un usager peut déposer des messages par l'intermédiaire de la mémoire de messages et recevoir des messages qui ont été remis à la mémoire de messages. L'enregistrement de message n'agit que pour le compte d'usages individuels.



Les échanges entre une mémoire de messages et le système de transfert de messages se font au moyen du protocole P3. Les échanges entre un agent d'usager et le système de transfert de messages se font au moyen du protocole P7. Les opérations fournies par le protocole P7 sont les suivantes:

- service d'extraction (résumé, liste, recherche, suppression, enregistrement dans la mémoire de messages avec possibilité de signal asynchrone, alerte);
- dépôt indirect utilisant les services de dépôt X.411 (dépôt de message, dépôt à l'essai, remise avec suppression différée et commande de dépôt);
- services d'administration (références d'enregistrement et de modification);
- utilisation des opérations MS-Bind et MS-Unbind pour la liaison et la fin de liaison aux services MS. L'opération de liaison est utilisée pour identifier, authentifier et mettre en place le contexte de sécurité d'un utilisateur du service MS.

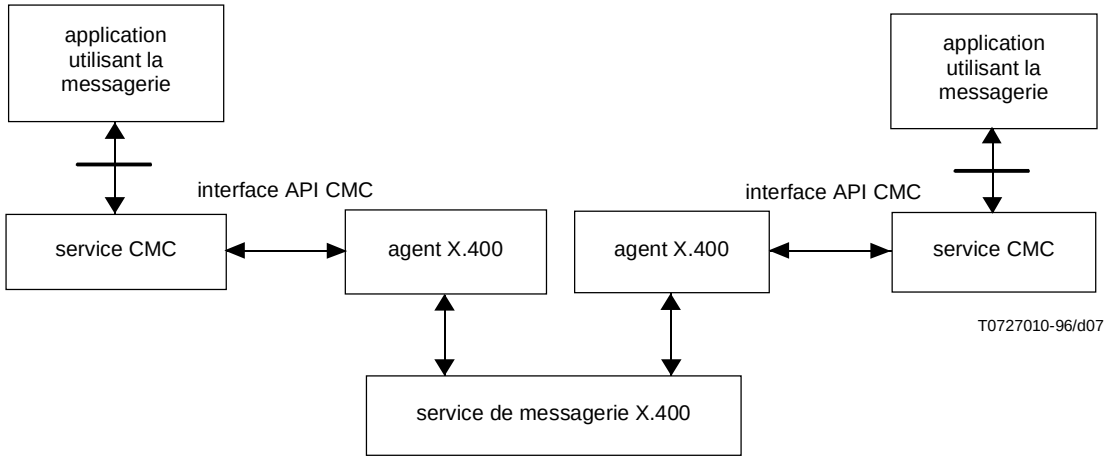
Les opérations de protocole P7 sont invoquées par l'agent d'usager au moyen de l'élément de service d'opérations distantes (élément ROSE, défini dans les Recommandation X.219 et X.229). Le modèle de l'élément ROSE se constitue d'interactions de demandes et de réponses. Ceci permet à l'application de l'agent d'usager de demander une opération P7 et d'en obtenir le résultat.

B.2.5.3 Démarche et généralités

Le présent sous-paragraphe présente une vue générique de mappage CMC v1.0 (appelé également CMC v2.0 simple) vers la messagerie MHS X.400, ainsi que certaines des considérations et des options possibles. La présentation ne traite pas de l'interface humaine, parce que les applications utilisant la messagerie sont considérées comme pouvant être exploitées sans interaction humaine, tout en pouvant également être exécutées au moyen de commandes ou de scénarios de l'utilisateur. Ceci signifie que les applications ne nécessitent pas d'interface utilisateur graphique et peuvent être exploités sous forme de processus en arrière plan.

B.2.5.3.1 Fonctions CMC et opérations MS X.400

Compte tenu des modèles fonctionnels de l'interface CMC v1.0 et de la mémoire de messages MS X.400, les fonctions CMC correspondent relativement bien aux services MS. Les ouvertures et fermetures de session CMC correspondent aux opérations MS de connexion et déconnexion. L'émission CMC correspond au dépôt MS indirect. La lecture CMC correspond à l'extraction MS. Les fonctions CMC "agir sur" et "liste" sont couvertes par les fonctions MS de suppression, de résumé et de liste.



service de messagerie CMC d'application à application

Les deux objectifs de l'interface API CMC consistent à fournir non seulement un ensemble générique de capacités de messagerie indépendantes de tout système d'exploitation, mais également un nombre minimal d'appels de fonction nécessaires pour émettre et recevoir un message. Le nombre réduit d'appels de fonction et l'interopérabilité sont des exigences fondamentales. Il est possible d'appliquer deux démarches de réalisation d'une interface API d'appel de fonctions pour la mise en correspondance des fonctions CMC avec les opérations MS et la mise en œuvre de ces prescriptions. Le choix entre ces deux styles détermine la manière dont les appels CMC sont mis en œuvre:

- fonctions simplifiées d'appel CMC sans extensions spéciales pour l'environnement local;
- appel de fonctions CMC masquant la complexité interne et prenant en charge des extension génériques.

B.2.5.3.2 Messages CMC et messages X.400

Une grande partie de la tâche de traduction entre les messages CMC et X.400 implique la conversion entre les structures de messages respectives. Il en résulte que les noms et les adresses d'utilisateurs du service de messagerie CMC doivent être convertis vers des noms et des adresses d'utilisateurs X.400 (expéditeur et destinataire) qui sont connus comme noms expéditeur/distributeur (noms expéditeur/distributeur incluant les adresses d'expéditeur et de destinataire).

Les autres parties d'une structure de message CMC nécessitant une conversion sont le type de message, l'instant d'émission, les destinataires et les attachements. Un sujet de message CMC est simplement le sujet du message IPM X.400 (avec certaines limitations concernant la taille), alors que la note de texte nécessite un traitement spécial étant donné les différences entre les systèmes CMC et X.400. Les fanions, extensions et autres paramètres originaux d'entrée ne sont utilisés que comme des aides lors de la conversion. Ils ne font pas partie du message X.400 émis, de sorte que l'information correspondante est perdue dans la plupart des cas, lorsque le message X.400 remis est converti à nouveau en message CMC pour être fourni à l'application CMC destinataire.

Les conversions, conventions et autre choix nécessaires à l'utilisation de la messagerie X.400 comme service de messagerie sous-jacent sont, entre autres, les suivants:

Conversion de textes:

- conversion du jeu de caractères;
- conversions de nom et d'adresse.

Conventions du système d'exploitation local et du service de messagerie:

- conventions de connexion et de déconnexion;
- connaissance des erreurs du service de messagerie sous-jacent;
- conventions et prescription particulières du système de mémoire de messages.

Conventions de messages sortants:

- conventions natives ou génériques, telles que les options de conversion de texte;
- conversions propres à la destination finale.

Conversions de messages entrants:

- traitement de messages CMC ne pouvant être traités localement;
- traitement de messages CMC contenant des parties qui ne peuvent pas être traitées localement;
- traitement de messages CMC incorrects;
- traitement de messages non-CMC.

Extensions spéciales génériques et locales

- extensions traitant des caractéristiques du service de messagerie sous-jacent qui ne sont pas génériques pour l'interface CMC;
- extensions permettant d'ajouter de fonctions spéciales X.400 telles que la priorité ou l'avis de remise.
- extension permettant d'utiliser des parties spécifiques du corps du message X.400 comme attachements;
- extensions utilisées pour assortir les systèmes locaux expéditeur et destinataire.

B.2.5.3.3 Attachements de message CMC et parties de corps de message X.400

Les attachements (de texte et binaires) d'un message CMC sont équivalents à des parties de corps de message X.400. La partie la plus adéquate pour constituer un attachement CMC diffère pour chaque "version" de la messagerie de la Recommandation X.400 (c'est-à-dire 1984, 1988 ou 1992). La liste ci-dessous indique les équivalences recommandées.

ASCII CMC	<=>	texte IA5	
Note de texte CMC sous forme de fichier	<=>	partie de corps de texte IA5	
Attachement de texte CMC	<=>	X.400 1984	partie de corps de texte IA5
Attachement de texte CMC	<=>	X.400 1988	partie de corps définie d'une manière externe
Attachement de texte CMC	<=>	X.400 1992	partie de corps de transfert de fichier
Attachement binaire CMC	<=>	X.400 1984	partie de corps définie d'une manière bilatérale
Attachement binaire CMC	<=>	X.400 1988	partie de corps définie d'une manière externe
Attachement binaire CMC	<=>	X.400 1992	partie de corps de transfert de fichier

B.2.5.3.4 Conventions et prescriptions d'entrée sortie.

Le jeu de caractères commun pratiquement universel utilisé dans la messagerie X.400 est l'alphabet international numéro 5 (texte IA5) qui est proche de l'alphabet ASCII. Des caractères ASCII, tels que "@", "%" et "_", ne figurent pas dans l'alphabet IA5 de base, mais se trouvent dans la version internationale de référence (IRV, *international reference version*). Les autres versions (nationales) de l'alphabet IA5 nécessitent une convention à des fins d'entrée sortie. L'atelier OIW de l'institut NIST a défini un algorithme de conversion pour l'échange de textes IA5 et ASCII et son utilisation est recommandée si la majorité des textes traités par l'application locale utilisant la messagerie utilisent une version de l'alphabet IA5 différente de la version internationale de référence. Une convention comparable est nécessaire pour d'autres jeux de caractères, tels que les alphabets EBCDIC, ISO 10646, UNICODE, etc., ainsi que pour traiter les noms de fichier contenant des espaces qui ont été communiqués dans des chaînes de caractères CMC avec comptage.

Les noms et adresses X.400 sont organisés d'une manière interne par la messagerie de la Recommandation X.400 sous forme d'ensembles structurés d'objets de données qui sont comparables à la structure de destinataire CMC, mais plus complexes. Il existe une convention de présentation de nom et d'adresse X.400, par exemple sur une carte de visite. Cette convention doit être utilisée pour la représentation des adresses de destinataires X.400 sous la forme d'un texte contenu dans une chaîne de caractères CMC, de même que les autres conventions normalisées de présentation contenues dans l'Annexe B de la Recommandation F.401 (faisant référence à l'Annexe F de l'ISO/CEI 10021-2). L'utilisation de la fonction CMC de recherche peut fournir un moyen simple de passer d'un nom connu tel que "eowens" à une chaîne de caractères d'adresse (de nom), comme indiqué ci-dessous, ou à une structure de destinataire CMC.

> Les destinataires CMC (expéditeur dans une structure de résumé de message CMC ou destinataires dans des structures de message CMC) utilisent des chaînes de caractères d'adresse correspondant à celles recommandées dans la Recommandation F.401. Dans l'exemple, la chaîne de caractères du destinataire est "S=Owens; G=Edward; P=ccmail; A=telemail; C=US".

Des conventions multiples seront nécessaires pour les représentations de nom et d'adresse si une combinaison de services de messagerie sous-jacents est prise en charge. Dans l'exemple précédent, on fait l'hypothèse d'un service de messagerie sous-jacent unique (X.400). Il existe de même des prescriptions de conversion pour les noms et les adresses qui sont passés à travers le service de messagerie lorsque le jeu de caractères utilisé par ce dernier ne prend pas en charge les conventions originales de nom et d'adresse. Si l'information nom@adresse ne peut être transférée dans un texte IA5 affichable qu'au moyen d'une convention spéciale telle que nom(a)adresse, la convention inverse doit être appliquée à l'autre extrémité, ou bien la convention doit être compréhensible pour le destinataire. Si par exemple, l'adresse X.400 implique des attributs définis pour le domaine tel qu'un attribut DDA pour une adresse Internet, le nom et l'adresse peuvent se présenter comme suit:

DDA:RFC-882=fred(a)widget.co.uk;O=gateway;P=abc;C=gb

Les conventions de taille de mot et d'ordre de succession des octets peuvent également créer des problèmes qui rendent inutilisables certains attachements et en brouillent de nombreux autres. La solution usuelle de la plupart de ces problèmes consiste à normaliser le contenu et les formats des informations transmises vers l'extérieur. Si toutefois, cette information est indépendante du service de messagerie sous-jacent et que le nombre de ces derniers n'est pas connu (ou seulement en partie, par l'ensemble des applications CMC utilisant la messagerie), il est nécessaire de trouver une autre solution. Une solution simple consiste à assurer que les applications de l'émetteur utilisant la messagerie soient au courant des capacités des applications destinataires utilisant la messagerie, et, de ce fait, de créer des messages sur mesure. Ceci simplifie le transfert entre systèmes possédant des capacités similaires. Une variante consiste à envoyer avec le message des informations supplémentaires indiquant au système destinataire les capacités de l'émetteur et le formatage particulier du message ainsi que d'autres détails spéciaux.

S'il est possible d'établir des conventions spéciales, l'utilisation des extensions CMC fournit un moyen pour indiquer aux deux mises en œuvre CMC comment transmettre en émission l'information à travers le service de messagerie sous-jacent à destination des mises en œuvre CMC réceptrices, de sorte que l'application utilisant la messagerie puisse choisir la façon de lire le message. Il est recommandé, à l'heure actuelle, que l'interface CMC 1.0 laisse ce choix à la mise en œuvre locale, car ces problèmes sont le mieux résolus lorsque toutes les exigences locales sont connues.

B.2.5.3.5 Connexion et déconnexion X.400 et prescriptions de mémoire MS

La connexion et la déconnexion avec un service de messagerie X.400 sous-jacent sont soumises à des variations locales et dépendent également du type de service à interfacer. L'hypothèse faite à l'heure actuelle est que la connexion se fait au moyen du service de mémoire de messages X.413. Ce service englobe les prescriptions CMC et fournit des fonctions supplémentaires. Les diverses caractéristiques et fonctions de l'interface X.413 nécessitent une interprétation de manière à ce que les services CMC attendus soient fournis d'une manière compréhensible.

Pour que l'opération X.413 de liaison puisse servir d'ouverture de session CMC vers une messagerie X.400, les arguments "usager" et "mot de passe" doivent être des chaînes de caractères contenant le nom de l'utilisateur du service de messagerie et le mot de passe lui permettant d'accéder au service sous-jacent. Dans la terminologie X.413, il s'agit des paramètres ORAddressAndOrDirectoryName et InitiatorCredentials. Si on fait l'hypothèse que seule une authentification simple est demandée par le service X.413, il est recommandé que l'argument usager contienne la version de texte affichable, au sens de la Recommandation F.401, du nom d'usager et de l'adresse X.400.

Dans le cas d'une authentification plus complexe, ou pour prendre en charge d'autres arguments de liaison X.413, les extensions spéciales CMC fournissent un moyen local d'ajouter un contexte de sécurité, des contraintes d'extraction et des demandes de configuration de la mémoire de message. D'autres extensions CMC peuvent également être nécessaires pour traiter localement les résultats renvoyés par un appel de liaison au service MS ou les erreurs de liaison renvoyées. Le choix le plus simple consiste à ignorer le résultat de l'appel de liaison au service MS, et de faire renvoyer par la fonction d'ouverture de session CMC un code erreur CMC_E_FAILURE, si l'appel de liaison MS renvoie une erreur. Un autre choix consiste à utiliser une extension CMC optionnelle pour rendre compte de l'erreur X.400 correspondante.

Les messages X.400 entrants sont stockés dans la mémoire de messages et lus soit par une demande explicite d'un message de courrier donné (dont le numéro est connu), soit en demandant le prochain élément de courrier non encore lu. Les attachements sont renvoyés soit dans un répertoire temporaire avec comme nom de fichier leurs noms de fichier d'attachement, soit comme fichiers nommés d'une manière temporaire dans un répertoire, avec une certaine indication du titre de l'émetteur. Les parties de messages de courrier, qui ne peuvent pas être mappées dans la structure de message CMC, doivent être rejetées.

Le service de mémoire de messages doit disposer d'une interface P7 et mettre en œuvre les deux fonctions X.413 permettant de faire des interrogations au sujet des message de courrier client adéquats, permettant ainsi la remise des articles demandés, ainsi que des articles "X.420" tels que l'en-tête et diverses parties du corps contenues dans un message interpersonnel X.420.

L'opération MS de fin de liaison ferme l'association avec l'usager (ou avec l'application utilisant la messagerie) et ne possède pas d'argument (résultats ou erreurs). Il s'ensuit que la fermeture de session CMC n'entraîne pas de complications additionnelles causées par le service de messagerie X.400 sous-jacent.

Les applications utilisant la messagerie (ou les usagers) doivent avoir connaissance d'une fonction de la mémoire de message. La mémoire de messages peut contenir des entrées filles en plus des entrées principales correspondant aux messages stockés. Les entrées filles apparaissent sur la liste tout comme leurs parents, mais l'application utilisant la messagerie (ou l'usager) ne peut supprimer les entrées filles qu'en supprimant l'entrée parente. La suppression d'une entrée fille ne marche pas comme on pourrait s'y attendre.

B.2.5.4 Conversion de message

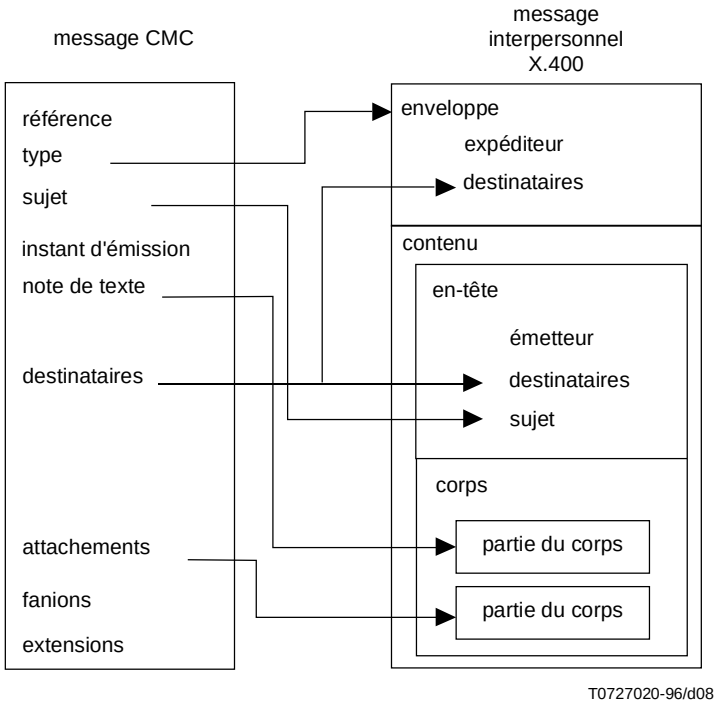
Le présent sous-paragraphe traite de la conversion de messages CMC en messages X.400, de la conversion de messages X.400 en messages CMC et de la conversion de messages X.400 non-CMC.

B.2.5.4.1 Conversion de messages CMC en messages X.400

Une structure de message CMC contient un type de message, un sujet, un ou plusieurs destinataires et, d'une manière optionnelle, une note ou un ensemble d'attachements ou les deux. L'équivalent de ce message CMC pour la messagerie X.400 est une unité de donnée de protocole de message (MPDU, *message protocol data unit*). La structure de base d'une unité MPDU 1984 se constitue d'une enveloppe et d'un contenu. Les agents d'usager (UA) fournissent des services supplémentaires aux individus souhaitant communiquer entre eux. Ce service constitue un service de messagerie interpersonnelle (IPMS) avec un type de contenu assigné pour ces messages. Les agents d'usager de la messagerie IPM communiquent entre eux. Le contenu d'un message IPM se constitue d'un en-tête et d'un corps. Il en résulte que le message CMC est découpé en enveloppe, en-tête et corps à des fins de conversion.

L'enveloppe contient l'identité de l'expéditeur (nom et adresse), les identités des destinataires et des détails propres à l'enveloppe de message X.400. L'en-tête de message IPM contient l'identité de l'expéditeur, un ensemble d'usagers donnant des autorisations, un ensemble d'identités de destinataire (primaire, copie, copie muette), un sujet et d'autres détails spécifiques X.400. Le corps est une suite de parties de corps. Toute partie de corps est caractérisée par un type. Les types les plus pertinents sont le texte IA5 et la partie de corps définie d'une manière bilatérale (type 14). D'autres types pertinents ont été définis pour des versions ultérieures de la messagerie de la Recommandation X.400 (1988 et 1992).

Des types de partie ajoutées ultérieurement sont la partie de corps définie d'une manière externe (1988) et la partie de corps de transfert de fichier (1992). Ces deux types conviennent bien mieux à une utilisation par l'interface CMC car plus de choses, autres que les données, peuvent être converties (et non perdues). Il en résulte qu'il existe une étendue de choix possibles pour la note et les attachements.



La liste ci-dessous donne quelques uns de ces choix:

- conversion de tous les messages CMC en messages X.400 (1984) avec texte (note et attachement) contenus dans des parties de corps du type texte IA5 et des attachements "binaires" contenus dans des parties BDBP/non identifiées (type 14);

- conversion de tous les messages CMC en messages X.400 (1988) avec note de texte du type texte IA5 et des attachements contenus dans des parties EDBP (type 15);

- convertir les attachements en fonction du millésime (version) de la messagerie de la Recommandation X.400;

- convertir en utilisant une extension CMC spéciale indiquant quelle partie de corps X.400 est à utiliser dans chaque cas impliquant l'utilisation d'autres parties de corps.

Les versions récentes de la messagerie de la Recommandation X.400 fournissent un domaine de choix plus étendus pour les parties de corps. Ces parties de corps récentes prennent en charge aussi bien une partie de corps du style texte qu'une partie de corps du style chaîne d'octets. Les parties de corps des types texte IA5, définies bilatéralement, ne peuvent véhiculer que du texte ou des chaînes "binaires", mais en aucun cas l'information supplémentaire de titre d'attachement ni de nom de fichier d'attachement. Une convention telle que l'utilisation d'une partie de corps supplémentaire pour l'information supplémentaire est possible mais non recommandée.

Les séries de Recommandations X.400 1988 déconseillent l'utilisation de la partie de corps définie d'une manière bilatérale, et recommandent l'utilisation de la partie de corps définie d'une manière externe. Les capacités supplémentaires de la partie de corps définie d'une manière externe permettent de véhiculer le titre de l'attachement avec la partie de corps. La version 1992 de la messagerie X.400 a défini une partie de corps de transfert de fichier permettant de transférer un fichier stocké et, d'une manière optionnelle, ses attributs. La portion contenu est similaire à la partie de corps définie d'une manière externe, alors que la partie optionnelle du paramètre véhicule des attributs tels que le nom correspondant du fichier stocké, le type de contenu, les relations et les attributs de fichier. Il en résulte que la partie de corps de transfert de fichier convient d'une manière idéale pour le transport d'attachement CMC.

La version 1984 de la messagerie X.400 est le système de messagerie X.400 le plus répandu, de sorte qu'un compromis est nécessaire. Un profil de base suggéré utilise des parties de corps X.400 1984 et prescrit que si les parties de corps plus récentes sont utilisées, elle peuvent être déclassées pour être rendues équivalentes à des parties de corps 1984. De même, toutes les applications utilisant la messagerie X.400 qui utilisent les parties de corps plus récentes, doivent être en mesure d'accepter à leur place l'ensemble de profil de base. Ceci signifie que les titres d'attachement nécessitent une substitution automatique à l'extrémité réceptrice, car ils ne sont pas transférés.

Le système MHS X.400 prescrit un ensemble d'attributs obligatoires faisant partie du dépôt d'un message, qui contiennent entre autres l'expéditeur (nom expéditeur/destinataire de l'émetteur). Ceci est également prescrit pour l'ouverture de session MS, de sorte que si le récepteur du message CMC n'en contient pas un avec un rôle CMC_ROLE_ORIGINATOR, l'usager indiqué par les paramètres d'ouverture de session MS sera utilisé par défaut. Un nom de destinataire est également prescrit et le dépôt du message doit être refusé si aucun n'est fourni.

La messagerie X.400 qui est émis sous la forme d'un message interpersonnel (IPM) doit avoir un type de contenu indiquant un message interpersonnel 1984 ou 1988. D'autres attributs X.400 obligatoires doivent être fournis comme valeurs par défaut (priorité normale positionnée), ou, dans le cas contraire, avec les indicateurs par message positionnés avec les bits indiquant une révélation des destinataires non autorisée, une conversion implicite interdite, pas de destinataire de remplacement et pas de renvoi de contenu. Un autre attribut obligatoire en entrée doit demander le positionnement des bits de demande de compte rendu expéditeur pour indiquer un compte rendu de non-remise.

L'utilisation d'extensions CMC optionnelles est recommandée pour la prise en charge de l'utilisation de la partie de corps la plus adéquate ou l'addition d'autres attributs X.400 à un message. Elles sont traités dans un sous-paragraphe ci-dessous. Pour une utilisation initiale et de base du service de messagerie X.400, le profil de base ne prescrivant pas d'extensions supplémentaires est toutefois recommandé, car ceci conduit à une interopérabilité mais pas au meilleur échange entre deux systèmes prenant chacun en charge les extensions de l'autre.

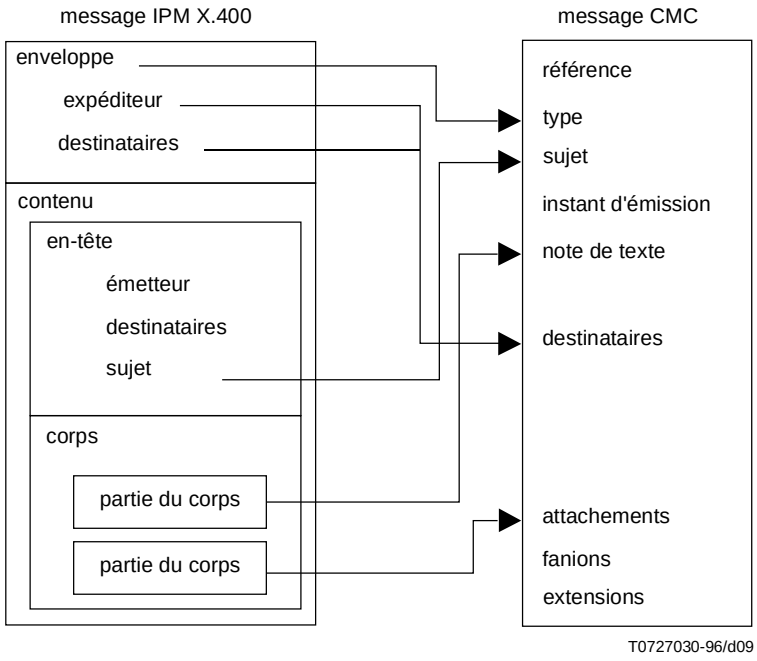
B.2.5.4.2 Conversion de messages X.400 en messages CMC

Les messages X.400 se présentent dans de nombreux types et versions. Le message interpersonnel (IPM) est le plus approprié pour la conversion CMC du point de vue du type et de la taille, mais une mémoire de messages X.400 peut contenir tout type de message X.400, y compris ceux qui sont altérés. Les principales variétés sont les messages, les essais et les comptes rendus. Parmi les messages se trouvent les messages interpersonnels "P1" et d'autres messages "P2" tels que des messages EDI X.435. Les messages IPM incluent les notifications (IPN) et des versions basées sur les normes 1984, 1988 et 1992.

Plusieurs choix simples sont faciles à faire:

- l'usager CMC voit à la fois les messages CMC et les autres messages X.400 contenus dans la mémoire de message;
- l'usager CMC peut également lire partiellement les autres messages X.400;
- l'usager CMC peut supprimer ces autres messages (sauf s'ils sont des enfants d'autres messages).

La conversion des messages CMC en messages X.400 a été décrite dans le sous-paragraphe précédent, de sorte que la conversion en sens inverse vers des messages CMC est assez simple, sauf pour l'information perdue au cours du processus. Les parties de corps X.400 1984 utilisées pour véhiculer des attachements ne peuvent pas, par exemple, véhiculer le nom d'attachement, de sorte que celui-ci est perdu.



Les comptes rendus générés par la remise ou la non-remise d'un message CMC X.400 apparaissent comme comptes rendus dans la mémoire de messages de l'émetteur. Le compte rendu contiendra en retour, sur demande de l'expéditeur, le message interpersonnel qui ne peut être remis. Il est possible également qu'un compte rendu X.400 contienne un compte rendu de remise pour certains destinataires et de non-remise pour d'autres en fonction de l'endroit où la faute a eu lieu. Ces deux types de compte rendu font partie du compte rendu, du point de vue de la mémoire de message, ce qui est totalement différent du point de vue de la messagerie CMC. Les comptes rendus doivent être convertis d'une manière adéquate, de façon à permettre à un utilisateur CMC d'établir une corrélation avec le message qui a été émis.

Les autres types de message X.400 pouvant se trouver dans la mémoire de messages sont des messages CMC envoyés par des utilisateurs CMC qui emploient une autre méthode de conversion de leurs messages X.400 en messages CMC. Il est, en conséquence, nécessaire de gérer toute une série de conversions possibles. Quelle que soit la méthode choisie, elle doit permettre à l'utilisateur destinataire de choisir d'ignorer la conversion, d'en accepter des parties ou de l'accepter complètement, compte tenu d'autres fonctions disponibles pour la conversion.

Il existe d'autres problèmes en relation avec le surclassement et le déclassement de messages X.400 ainsi que d'autres conversions de messages X.400. Tout cela signifie qu'il existe, soit un nombre réduit d'importations d'un message X.400 "non-CMC" permettant à un usager CMC de "voir" ce message, soit un ensemble de règles complexes. Dans le cas de conversion simple, les parties de corps en texte IA5 sont des notes ou des attachements. Toutes les autres parties de corps peuvent être considérées comme "binaires" et l'usager dispose du type et éventuellement du titre comme indices concernant le contenu de l'attachement. Tous les messages qui ne peuvent pas être convertis peuvent être indiqués par un texte dans la ligne de sujet.

La version X.400 locale (1984, 1988, ou 1992) sera souvent le facteur déterminant pour la possibilité de conversion d'un message X.400 ou d'un corps d'un message IPM. Le type de message ou d'une partie de corps peut être utilisé pour rejeter cet élément et le remplacer par un indicateur fourni à l'usager CMC.

Une conversion simple est recommandée pour le profil de base (minimal). Le profil du côté émetteur du message contient une demande de compte rendu de non-remise positionnée, de sorte que les comptes rendus peuvent apparaître en clair dans la mémoire de message. Etant donné que l'information majeure contenue dans le compte rendu est la remise ou la non-remise d'un message, cette information est la seule devant être renvoyée à l'application utilisant la messagerie. En conséquence, une conversion demandée est le remplacement du compte rendu par un message contenant la note de texte "ce message a été (n'a pas été) remis" ainsi que les destinataires, etc.

D'autres conversions seront nécessaires si le type de contenu concerne une version différente (millésime X.400) et que les parties de corps ne sont pas celles attendues. Par exemple, pour une partie de corps définie d'une manière bilatérale à la place d'une partie de corps définie d'une manière externe ou d'une partie de corps de transfert de fichier à la place d'une partie de corps de texte IA5. Les règles de conversion sont simples dans ce cas, quoique la conversion d'une partie de corps en une autre, nécessite un code supplémentaire.

B.2.5.4.3 Conversion de messages X.400 non CMC

Lorsqu'une application utilisant la messagerie, ou son utilisateur, possède une adresse X.400, la mémoire de messages X.400 stockera tout message émis vers cette adresse, par exemple un message EDI, un message interpersonnel ou un compte rendu. Il en résulte que l'utilisateur peut émettre de nombreux messages qui ne sont pas des messages créés par l'interface CMC ou d'autres messages utilisant des conventions différentes, que le système de messagerie CMC local ne peut traiter. Il est nécessaire de disposer, pour traiter ces problèmes, d'un ensemble de conventions sur ce qui est rejeté, sur ce qui est converti en partie de manière à ce que l'utilisateur puisse prendre connaissance de certains détails du message, et sur les parties pouvant être remplacées ou supprimées.

Les possibilités de conversion simples sont les suivantes:

- tous les messages non-CMC sont rejetés lorsqu'ils sont lus par une application CMC;
- tous les messages CMC se trouvant dans une forme différentes (telle qu'un message PEM comme partie de corps P22 X.400) qui ne peut pas être traité par le système local (par exemple X.400 1984) sont rejetés. Celles des parties de corps qui peuvent être traitées par les mises en œuvre CMC locales et X.400 sont remplacées par une simple note de texte, un fichier de chaîne de caractères ou un fichier binaire;
- convertir toutes les parties de corps inconnues sous forme de fichier utilisable partiellement en utilisant une conversion simple d'affichage de tous les caractères imprimables tels quels, et en convertissant tous les caractères non imprimables dans une forme d'affichage "\'hex"hex"', de sorte qu'un "usager" puisse déterminer si ce fichier peut être récupéré.

Des extensions CMC spéciales optionnelles peuvent être utilisées comme précédemment pour véhiculer en retour une information supplémentaire vers l'application utilisant la messagerie, ou bien pour fournir à la fonction CMC des indices et des demandes à la fonction CMC sur la manière de convertir ou d'interpréter divers attributs et parties de corps du message reçus. En plus du traitement d'étranges messages, l'interface CMC locale doit avoir des conventions traitant des retours d'erreurs en provenance du service de messagerie sous-jacent. La question est de savoir si ces erreurs du système de messagerie sont passées ou non, à l'application CMC utilisant la messagerie. Et s'il en est ainsi, sous quelle forme: converties vers une convention CMC d'erreur commune ou laissées sous leur forme de base?

Les choix d'erreurs de la messagerie sous-jacente sont les suivants:

- remplacer par CMC_E_FAILURE en cas d'appel, ignorer sinon;
- si présente, mettre l'erreur brute dans une extension CMC spéciale d'erreur X.400;
- convertir l'erreur vers une extension CMC normalisée d'erreur de communication.

B.2.5.4.4 Extensions CMC supplémentaires

Bien des choses pourraient être réalisées au moyen d'extensions. La liste qui suit en donne un aperçu rapide:

- extensions pour sélectionner lequel de plusieurs services de messagerie sous-jacents utiliser;
- extensions pour configurer et établir l'environnement utilisé par le système CMC et le système de messagerie;
- extensions pour ajouter des attributs X.400 supplémentaires, au-delà de l'ensemble de base utilisé par l'interface CMC;
- extensions déterminant quelles parties de corps X.400 utiliser, en tenant compte de certains facteurs;
- extensions indiquant à la fonction CMC de lecture comment traiter des messages entrants qui ne sont pas des messages de base;
- extensions passées avec le message pour indiquer au destinataire comment le traiter.

L'utilisation d'extensions soulève un certain nombre de problèmes. Le premier point est de savoir si elles sont réellement nécessaires. Une configuration de base pourrait, ou ne devrait pas, s'appuyer sur des extensions, mais se limiter à traiter d'une manière acceptable les prescriptions minimales définies pour les appels CMC simples. Les diverses mises en œuvre peuvent traiter d'une manière spécifique les prescriptions locales et adapter ainsi leurs actions et leurs comportements.

Il existe beaucoup d'extensions CMC possibles, et de façons de les caractériser. Certaines extensions sont nécessaires pour adapter la version CMC 1.0 locale au service de messagerie sous-jacent local, et même pour sélectionner lequel des systèmes locaux de messagerie utiliser.

Ces extensions sont dites locales. D'autres extensions spéciales traitent de l'utilisation de fonctions non-CMC (au moins dans la version CMC 1.0). De telles extensions sont nécessaires pour l'interopérabilité.

Dans le cas de la messagerie de la Recommandation X.400, il existe des extensions exigées pour le traitement de base de la messagerie X.400 et qui sont donc génériques. Les extensions X.400 génériques traitent des résultats X.400 et des retours d'erreur. Ces extensions peuvent être même généralisées davantage pour englober les résultats et les retours d'erreurs d'autres services de messagerie non X.400.

Les retours d'erreur des appels X.400 peuvent être renvoyés, dans une extension CMC spécifique, à l'application utilisant la messagerie au moyen d'appels CMC. Ceci est essentiel pour ceux des clients d'appel CMC, qui ont besoin de se rétablir en déterminant à quel endroit l'appel a été perdu.

D'autres extensions génériques CMC X.400 traitent de la priorité des messages et de la demande de comptes rendus de livraison, de non-livraison ou des deux. D'autres extensions véhiculent des certificats de l'utilisateur X.400 et des contraintes de mémoire de message. Le type de message du message X.400 véhicule des identificateurs, tels que des identificateurs d'objet, permettant à l'expéditeur du message CMC de spécifier toute forme de message X.400. Une autre extension "générique" sera nécessaire pour renvoyer cette information d'erreur vers l'utilisateur de l'appel CMC, si les fonctions CMC ou le service de messagerie X.400 sous-jacent ne prennent pas en charge ce type particulier de message X.400 (ou exigent d'une manière spéciale que ce type ne soit pas utilisé). Des problèmes similaires se posent pour l'addition d'une extension spéciale permettant la sélection de la partie de corps X.400 adéquate pour le type d'attachement de l'attachement CMC.

D'autres extensions possibles peuvent être utilisées pour assister le côté récepteur dans le traitement de messages entrants, impliquant le fait que des comptes rendus doivent être supprimés ou ne doivent pas être lus, mais plutôt faire l'objet d'un compte rendu spécial. Le transfert de bout en bout des extensions (ou de leur information), afin d'assister le destinataire dans le traitement du message, appelle une étude ultérieure.

B.2.5.5 Profil de mappage de base pour l'interface CMC simple (CMC 1.0)

Lorsque le système MHS X.400 est utilisé comme service de messagerie sous-jacent, l'émission et la réception du message de base doivent être prises en charge par toutes les mises en œuvre de la version CMC 1.0. Un profil de base est défini à cet effet en vue de fournir cette fonctionnalité.

Une mise en œuvre CMC utilisant la démarche de "profil de base" doit être en mesure d'émettre tout message qui n'utilise que le format de message interpersonnel de base, et éventuellement les deux parties de corps X.400 recommandées pour les attachements (et la note de texte). De même, toutes celles des applications utilisant la messagerie X.400 qui utilisent des parties de corps plus évoluées doivent être en mesure d'accepter à leur place l'ensemble de profil de base. Les mises en œuvre CMC qui reçoivent des messages remis par la messagerie X.400 doivent être en mesure de lire correctement et de traiter ces messages de "profil de base". Tous les autres messages X.400 peuvent être rejetés.

Les règles suivantes doivent s'appliquer au profil de base: si le message entrant ou la structure de message sortant ne correspondent pas à un modèle normalisé, le message en question sera rejeté et un avertissement ou une erreur peuvent être renvoyés d'une manière optionnelle à l'application utilisant la messagerie. De même, lorsque les paramètres d'entrée ne correspondent pas à l'ensemble pouvant être autorisé ou ne peuvent pas être traités par le service de messagerie sous-jacent, une erreur (ou un avertissement si cela n'est pas faisable) est renvoyée à l'application utilisant la messagerie.

La suite du présent sous-paragraphe décrit le mappage de base minimal prescrit pour l'utilisation simplifiée des services de messagerie X.400. L'utilisation des services complets de messagerie X.400 disponibles peut être fournie par des extensions additionnelles ou optionnelles supplémentaires qui ne figurent pas dans ce profil de base. Il est donc nécessaire de prendre en considération l'interopérabilité éventuelle entre le profil de base et ce même profil étendu par des extensions.

NOTE – Une convention de nom spéciale est utilisée pour faire référence aux divers champs définis par la messagerie X.400. Tout champ possède un nom constitué de la désignation de la norme, suivi du nom du champ tel qu'il apparaît dans la norme. Un caractère "." est utilisé pour décrire un champ défini à l'intérieur d'un autre champ, par exemple X.420.en-tête.utilisateurs_donnant_l'autorisation.

B.2.5.5.1 Mappage du destinataire CMC

Le destinataire CMC est mappé vers un nom d'expéditeur et destinataire X.411 en utilisant la représentation textuelle d'une adresse expéditeur/destinataire pour une utilisation humaine telle qu'elle est décrite dans l'Annexe F de l'ISO/CEI 10021-2: 1990/Amd.1. Toute forme valide de nom d'expéditeur et destinataire, tel qu'il est décrit dans la Recommandation X.411, peut être utilisée.

Les mappages spécifiques sont les suivants:

destinataire_CMC.nom

correspond à X.411.nom_expéditeur/destinataire.nom_d'annuaire si la mise en œuvre prend en charge le nom d'annuaire, ignoré dans le cas contraire.

destinataire_CMC.type_de_nom

supposé être positionné sur la valeur INDIVIDUAL pour une opération de sortie X.400 vers CMC.

destinataire_CMC.addresse

correspond à X.411.nom_expéditeur/destinataire.adresse_expéditeur/destinataire.

destinataire_CMC.rôle

correspond à X.411.nom_d'expéditeur.nom_expéditeur/destinataire.adresse_expéditeur/destinataire si sa valeur est égale à CMC_ROLE_ORIGINATOR. Il correspond également à X.420.en-tête.expéditeur lorsque le message est un message interpersonnel ou une notification interpersonnelle.

correspond à X.411.nom_de_destinataire.adresse_expéditeur/destinataire si sa valeur est égale à CMC_ROLE_TO. Il correspond également à X.420.en-tête.destinataires_primaires lorsque le message est un message interpersonnel ou une notification interpersonnelle.

correspond à X.411.nom_de_destinataire.adresse_expéditeur/destinataire si sa valeur est égale à CMC_ROLE_CC. Il correspond également à X.420.en-tête.destinataires_en_copie lorsque le message est un message interpersonnel ou une notification interpersonnelle.

correspond à X.411.nom_de_destinataire.adresse_expéditeur/destinataire si sa valeur est égale à CMC_ROLE_BCC. Il correspond également à X.420.en-tête.destinataires_en_copie_muette lorsque le message est un message interpersonnel ou une notification interpersonnelle.

correspond à X.420.en-tête.utilisateurs_donnant_l'autorisation si sa valeur est égale à CMC_ROLE_AUTHORIZING_USER, lorsque le message est un message interpersonnel ou une notification interpersonnelle.

destinataire_CMC.fanions_de_destinataire

sont vérifiés pour trouver le dernier.

destinataire_CMC.extensions_de_destinataire

sont ignorées.

B.2.5.5.2 Mappage du message CMC

Le résumé de message CMC est mappé vers la base d'information liée à une opération X.413 de dépôt de message ou une opération X.413 de recherche.

Les mappages spécifiques sont les suivants:

message_CMC.référence_de_message

correspond à X.413.numéro_séquentiel_d'entrée.

message_CMC.type_de_message

correspond à X.413.type_d'entrée; avec "CMC:IPM" = message remis, "CMC:REPORT" = compte rendu remis, et "CMC:IPM" = contenu renvoyé avec la nouvelle extension CMC_X_X400_MSG_PARENT indiquant qu'il s'agit d'un message imbriqué.

message_CMC.sujet

correspond à X.413.contenu_de_X.420.en-tête.sujet.

message_CMC.instant_d'émission

de valeur NULL pour **cmc_send()**, ou correspond à X.413.instant_de_dépôt_du_message pour **cmc_read()**.

message_CMC.note_de_texte
: est NULL, ou pour **cmc_send()** et si le fanion CMC_TEXT_NOTE_AS_FILE est positionné, correspond à la première partie de corps de X.413.contenu_de_X.420.corps.texte_ia5.données, ou pour **cmc_read()** s'il correspond à la première partie de corps de texte ia5 disponible.

  L'information X.420.corps.texte_ia5.répertoire est ignorée.

message_CMC.destinataire
: correspond à X.411.nom_de_destinataire et X.420.en-tête.expéditeur, utilisateurs donnant l'autorisation, destinataires primaires, destinataires en copie, destinataires en copie muette, conformément au positionnement du rôle du destinataire CMC.

message_CMC.attachement
: correspond à X.420.corps.partie_de_corps. Tout attachement est mappé vers la clause de partie de corps correspondante. Voir B.2.5.5.4 "Mappage d'attachement CMC".

message_CMC.fanions_de_message
: positionné conformément à la valeur de X.413.statut_de_l'entrée pour les valeurs CMC_SUM_READ et CMC_SUM_UNSENT, détermination du dernier élément pour CMC_SUM_LAST_ELEMENT, CMC_MSG_TEXT_NOTE_AS_FILE pour le traitement du premier attachement et de la première partie de corps de texte IA5.

message_CMC.extensions_de_message
: NULL ou renvoie d'une manière optionnelle l'extension CMC_X_X400_MSG_PARENT.

B.2.5.5.3 Résumé de message CMC

Le résumé de message CMC est mappé vers les paramètres d'une opération X.413 de liste ou de résumé.

Les mappages spécifiques sont les suivants:

résumé_de_message_CMC.référence_de_message
: correspond à X.413.numéro_séquentiel_d'entrée.

résumé_de_message_CMC.type_de_message
: correspond à X.413.type_d'entrée; avec "CMC:IPM" = message remis, "CMC: REPORT" = compte rendu de remise, et "CMC:IPM" = contenu renvoyé.

résumé_de_message_CMC.sujet
: correspond à X.413.contenu_de_X.420.en-tête.sujet.

résumé_de_message_CMC.instant_d'émission
: correspond à X.413.instant_de_dépôt_du_message.

résumé_de_message_CMC.longueur_d'octet
: correspond à X.413.longueur_du_contenu.

résumé_de_message_CMC.expéditeur
: correspond à X.413.nom_de_l'expéditeur.

message_CMC.fanions_de_résumé
: positionnés conformément à X.413.statut_de_l'entrée et dernier élément.

résumé_de_message_CMC.extensions_de_résumé_de_message
: NULL ou renvoie d'une manière optionnelle l'extension CMC_X_X400_MSG_REPORT.

B.2.5.5.4 Mappage des attachements CMC

Les attachements CMC correspondent à des corps de message associés à une opération X.413 de dépôt ou à une opération X.413 de recherche. Si le message CMC possède une note, celle-ci est utilisée comme première partie de corps de texte IA5. Tout attachement est mappé vers une partie de corps X.420.

Les mappages spécifiques sont les suivants:

attachement_CMC.titre_d'attachement
: NULL, ou correspond à
  X.420.corps.partie_de_corps_définie_d'une_manière_externe.données
  .descripteur_de_valeur_de_données.

attachement_CMC.attach_type

NULL, ou correspond à un identificateur d'objet X.420.corps .partie_de_corps_définie_d'une_manière_externe.données.référence_directe.

Un identificateur d'objet de type CMC_ATT_TEXT est mappé vers le type pré-défini ou un identificateur d'objet d'une X.420.corps.partie_de_corps_de_texte_IA5. Le champ partie_de_corps_de_texte_IA5 .données.répertoire n'est pas utilisé.

Un type CMC_ATT_BINARY est mappé vers le type pré-déterminé ou un identificateur d'objet d'une X.420.corps.partie_de_corps_définie_d'une_manière_bilatérale.

D'autres types d'attachements correspondent à une X.420.corps.partie_de_corps_définie_d'une_manière_externe. Les champs de type EXTERNAL sont mappés comme suit:

- le paramètre (optionnel) n'est pas utilisé;
- les données de référence directe correspondent à l'identificateur d'objet spécifié;
- les données de référence indirecte (optionnelles) ne sont pas utilisées;
- le descripteur de valeurs de données (optionnel) n'est pas utilisé;
- le codage est positionné sur quelconque.

attachement_CMC.nom_de_fichier_d'attachement

correspond au nom de fichier externe de la mise en œuvre et n'est pas passé à la messagerie X.400. Le contenu de ficher est stocké sous forme de données dans des parties de corps X.400.

attachement_CMC.fanions_d'attachement

mappé conformément aux caractéristiques du propriétaire du nom de fichier d'attachement et au dernier élément d'un attachement.

attachement_CMC.extensions_d'attachement

NULL.

B.2.5.6 Mappage des fonctions CMC

La plupart des fonctions CMC sont mappés vers une opération de l'élément ROSE contenant une opération X.413. Les identificateurs d'invocation utilisés dans une opération ROSE doivent être des nombres non-ambigus. Les identificateurs d'invocation peuvent être générés par la mise en œuvre. Les identificateurs liés ne sont pas utilisés. Si un type d'extension ou de message ne peut pas être pris en charge par ce profil de base, une erreur est renvoyée aux fonctions CMC et les messages correspondants (ou certaines de leurs portions) seront rejetés ou ignorés.

B.2.5.6.1 Fonction CMC agir sur (*CMC Act on*)

La fonction **cmc_act_on()** correspond à une enveloppe d'élément ROSE contenant une opération de suppression X.413. L'opération de suppression X.413 a la structure suivante:

[type d'information de base, articles (choix de sélecteurs ou de numéros de séquence)]

Le type d'information de base est X.413.stockage_de_message (défaut). Les articles sont supposés être le numéro de séquence d'entrée X.413 fournie par la référence de message.

Mappage de paramètres:

```
CMC_return_code
cmc_act_on(
CMC_session_id              session,              identificateur local de
                                                  session
CMC_message_reference       *message_reference,   numéro de séquence d'entrée
                                                  x.413
CMC_enum                    operation,            ne prend en charge que
                                                  CMC_ACT_ON_DELETE résultant
                                                  d'opérations X.400
                                                  sous-jacentes
CMC_flags                   act_on_flags,         NULL ou ignoré
CMC_ui_id                   ui_id,                NULL ou ignoré
CMC_extension               *act_on_extensions    NULL ou CMC_X_X400_ERROR en
                                                  sortie
);
```

Commentaires supplémentaires:

néant.

B.2.5.6.2 Libération CMC (*CMC Free*)

La fonction **cmc_free()** ne nécessite aucun mappage vers des appels X.400.

Mappage de paramètres:

```
CMC_return_code
cmc_free(
CMC_buffer                memory
);
```

Commentaires supplémentaires:

néant.

B.2.5.6.3 Liste CMC (*CMC List*)

La fonction **cmc_list()** correspond à une enveloppe ROSE contenant une opération de liste ou de résumé X.413. Les opérations de liste ou de résumé X.413 ont la structure suivante:

[type d'information de base, sélecteur, (attributs demandés ou demande de résumé)]

Le type d'information de base est X.413.stockage_de_message (par défaut). Les articles sont supposés être le numéro de séquence d'entrée X.413 fournie par la référence de message.

Mappage de paramètres:

```
    CMC_session_id          session,              identificateur local de
                                                  session
    CMC_string              message_type,         filtre de liste sur le type
                                                  d'élément
    CMC_flags               list_flags,           UNREAD, REF_ONLY, COUNT_ONLY
    CMC_message_reference   *seed,                X.413.EntrySequenceNumber
    CMC_uint32              *count,               compteur
    CMC_ui_id               ui_id,                NULL
    CMC_message_summary     **result,             CMC:IPM ou CMC: compte rendu
                                                  ou OID
    CMC_extension           *list_extensions      Null ou CMC_X_X400_ERROR
);
```

Commentaires supplémentaires:

1) les entrées filles (optionnelles) ne sont pas utilisées;

2) si un placement de départ (*seed*) est fourni dans l'appel CMC, il peut être utilisé pour spécifier un domaine de valeurs: c'est à dire sélectionner un domaine de numéro de séquence FROM et utiliser le positionnement de départ fourni. La séquence de domaines TO (optionnelle) n'est pas utilisée;

3) un filtre est utilisé si le paramètre de type de message est spécifié OU si le fanion CMC_LIST_UNREAD_ONLY est positionné. Lorsque les deux conditions sont présentes, il convient d'utiliser un opérateur logique ET. Si le fanion CMC_LIST_UNREAD_ONLY est positionné, un élément du filtre est positionné sur "pas d'égalité des valeurs de statut d'entrée de l'article (traité)". Si un type de message est spécifié, un élément du filtre est positionné sur "égalité des valeurs de type d'entrée de l'article (message OU compte rendu)";

4) une limite est spécifiée si le paramètre de comptage de liste n'est pas nul, et dans ce cas le comptage de liste correspond directement à la valeur entière limite;

5) l'option de remplacement n'est pas utilisée;

6) les attributs suivants doivent être renvoyés depuis l'opération de liste:

   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 Fermeture de session CMC (*CMC Logoff*)

La fonction **cmc_logoff()** correspond à une enveloppe d'élément ROSE contenant une opération X.413 de fin de liaison. L'opération X.413 de fin de liaison ne possède ni argument, ni résultat, ni erreur.

Mappage de paramètres:

```
CMC_return_code
cmc_logoff(
   CMC_session_id        session,              identificateur local de
                                               session
   CMC_ui_id             ui_id,                NULL
   CMC_flags             logoff_flags,         NULL
   CMC_extension         *logoff_extensions    NULL
);
```

Commentaires supplémentaires:

néant.

B.2.5.6.5 Ouverture de session (*CMC Logon*)

La fonction **cmc_logon()** correspond à une enveloppe d'élément ROSE contenant une opération X.413 de liaison. L'opération X.413 de liaison a la structure suivante:

> [nom de l'expéditeur, accréditation de l'expéditeur, contexte de sécurité, restrictions de recherche, demande de configurations de mémoire de message]

Mappage de paramètres:

```
CMC_return_code
cmc_logon(
   CMC_string            service,              NULL, ou service local de
   CMC_string            user,                 chemin d'accès pour la forme
                                               texte du nom de l'expéditeur
                                               voir B.2.4.4.1 "Mappage
                                               destinataire CMC"
   CMC_string            password,             X.411.initiator-
                                               credentials.simple
   CMC_object_identifier character_set,        mot de passe NULL ou
                                               recherche locale
   CMC_ui_id             ui_id,                Configuration NULL
   CMC_uint16            caller_cmc_version,   numéro local de version v1.0
   CMC_flags             logon_flags,          NULL ou non utilisé
   CMC_session_id        *session,             identificateur local de
                                               session
   CMC_extension         *logon_extensions     NULL, ou CMC_X_X400_ERROR
);
```

Commentaires supplémentaires:

les éléments optionnels suivants ne sont pas utilisés:

a) contexte de sécurité de mémoire de message;

b) contraintes d'extraction;

c) demande de configuration de mémoire de message.

B.2.5.6.6 Consultation CMC (*CMC Look Up*)

La fonction **cmc_look_up()** ne nécessite aucun mappage ou appel X.400. Cette fonction n'est pas obligatoire pour la prise en charge du service de messagerie X.400.

Mappage de paramètres:

```
CMC_return_code
cmc_look_up(
   CMC_session_id        session               identificateur local de
                                               session
   CMC_recipient         *recipient_in         voir le mappage de
                                               destinataire CMC
   CMC_flags             look_up_flags         tous nuls, ou mise en œuvre
                                               locale
```

```
    CMC_ui_id             ui_id                 NULL
    CMC_uint32            *count                sortie pour une mise en œuvre
                                                locale
    CMC_recipient         **recipient_out       voir le mappage de
                                                destinataire CMC
    CMC_extension         *look_up_extensions   NULL
);
```

Commentaires supplémentaires:

néant.

B.2.5.6.7 Lecture CMC (*CMC Read*)

La fonction **cmc_read()** correspond à une enveloppe d'élément ROSE contenant une opération X.413 de recherche. L'opération X413 de recherche a la structure suivante:

> [type d'information de base, choix de recherche de l'article (ensemble de numéros de séquence) ou précis (numéro de séquence), attributs demandés]

Le type d'information de base est X.413.stockage_de_message (par défaut). Les articles sont supposés être le numéro de séquence d'entrée X.413 fournie par la référence de message.

Mappage de paramètres:

```
CMC_return_code
cmc_read(
    CMC_session_id        session,              identificateur local de
                                                session
    CMC_message_reference *message_reference,   NULL pour le premier message
                                                NON LU ou numéro de séquence
                                                d'entrée X.413
    CMC_flags             read_flags,           ne peut pas prendre en charge
                                                CMC_DO_NOT_MARK_AS_READ
    CMC_message           **message             pointeur vers un message
                                                stocke, voir le mappage des
                                                messages CMC
    CMC_ui_id             ui_id                 NULL
    CMC_extension         *read_extensions      NULL, ou CMC_X_X400_ERROR
);
```

Commentaires supplémentaires:

choisir une recherche précise si le paramètre de référence de message CMC est spécifié, sinon choisir une recherche par extraction.

Recherche précise:

la référence de message fournie peut être utilisée comme numéro de séquence de mémoire de message.

Recherche par extraction:

les entrées filles (optionnelles) ne sont pas utilisées;

choisir un domaine de numéro de séquence "FROM" et utiliser le numéro de séquence "0";

le numéro de séquence "TO" (optionnel) n'est pas utilisé;

un filtre est spécifié si le fanion CMC_READ_FIRST_UNREAD_ONLY est positionné. Cet élément de filtre est positionné sur "pas d'égalité des valeurs de statut d'entrée de l'article (traité)";

la limite (optionnelle) n'est pas utilisée;

remplacer (optionnel) n'est pas utilisé;

Les attributs suivants doivent être renvoyés:

id-att-parent-sequence-number

id-att-entry-type

id-att-message-submission-time

id-att-content-type (IPM, IPN, défini d'une manière externe, etc.)

B.2.5.6.8 Emission CMC (*CMC Send*)

La fonction **cmc_send()** correspond à une enveloppe d'élément ROSE contenant une opération X.413 de dépôt. L'opération X.413 de dépôt a la structure suivante:

[enveloppe(enveloppe de dépôt de message avec un contenu de message interpersonnel(contenu)]

Mappage de paramètres:

```
CMC_return_code
cmc_send(
   CMC_session_id        session,              identificateur de session
                                               local
   CMC_message           *message,             voir le mappage de message
                                               CMC
   CMC_flags             send_flags,           toujours nuls
   CMC_ui_id             ui_id,                NULL
   CMC_extension         *send_extensions      NULL, ou CMC_X_COM_PRIORITY,
                                               ou CMC_X_X400_ERROR
);
```

Commentaires supplémentaires:

enveloppe de dépôt de message X.411 (Enveloppe P3):

le champ expéditeur contient le nom d'expéditeur/destinataire de l'expéditeur (voir le mappage du nom d'expéditeur/destinataire). Si aucun expéditeur n'est spécifié, le nom d'expéditeur/destinataire utilisé dans la fonction **cmc_logon()** sera utilisé;

le type d'information originale codée (optionnel) n'est pas utilisé;

le type de contenu est positionné sur pré-déterminé et le type sur IPM-84;

l'identificateur de contenu (optionnel) n'est pas utilisé;

la priorité (optionnelle) est normale (par défaut) ou mappée à partir de CMC_X_COM_PRIORITY si celle-ci est présente;

l'indicateur par message (optionnel) utilise les valeurs par défaut:

- révélation du destinataire interdite (défaut);
- conversion implicite interdite autorisée (défaut);
- autre destinataire autorisé interdit (défaut);
- retour de contenu demandé non demandé (défaut);

la date de remise différée (optionnelle) n'est pas utilisée;

les extensions (optionnelles) ne sont pas utilisées;

toutes les adresses de nom d'expéditeur/destinataire de tous les destinataires sont remplies (voir le mappage de destinataire CMC). Les champs suivants dont positionnés après que chaque nom d'expéditeur/destinataire est rempli:

- demande de compte rendu de l'expéditeur interdite (défaut);
- conversion explicite (optionnelle) interdite (défaut);
- extensions (optionnelles) non utilisées.

Le contenu de l'enveloppe P3 est un message interpersonnel P2 constitué d'une enveloppe P2 et d'une ou de plusieurs parties de corps.

En-tête de message interpersonnel X.420:

l'utilisateur est rempli avec le nom d'expéditeur/destinataire (voir le mappage du destinataire CMC);

l'identificateur relatif d'utilisateur est créé en ajoutant à l'identificateur correspondant généré pour l'enveloppe de l'élément ROSE la chaîne de caractères "CMC:IPM";

les destinataires primaires, les destinataires en copie, les destinataires en copie muette, les demandes de notification (optionnelles) et demandes de réponse (optionnelles) ne sont pas utilisés;

le message interpersonnel en réponse à (optionnel) n'est pas utilisé;

les messages interpersonnels périmés (optionnels) ne sont pas utilisés;

les messages interpersonnels en relation (optionnels) ne sont pas utilisés;

le sujet du message correspond directement au champ sujet du message CMC;

la date d'expiration (optionnelle) n'est pas utilisée;

la date de réponse (optionnelle) n'est pas utilisée;

les destinataires en réponse (optionnels) ne sont pas utilisés;

l'importance (optionnelle) est normale (défaut).

la sensibilité (optionnelle) n'est pas utilisée;

la réexpédition automatique (optionnelle) n'est pas utilisée;

les extensions (optionnelles) ne sont pas utilisées;

Corps de message interpersonnel X.420

Si le message CMC possède une note de texte, celle-ci est utilisée comme première partie de corps de texte IA5. Le champ répertoire (optionnel) est ignoré lors de la création d'une partie de corps de texte IA5.

Une partie de corps X.400 est créée pour tous les autres attachements CMC. Si un attachement est du type CMC_ATT_TEXT, il est mappé vers une partie de corps de texte IA5. S'il est du type CMC_ATT_BINARY, il est mappé vers une partie de corps définie d'une manière bilatérale. Pour d'autres types CMC, la valeur d'identificateur d'objet fournie est utilisée, et donne lieu à la création d'une partie de corps créée d'une manière externe (EDBP).

B.2.5.6.9 Emission de documents CMC (*CMC Send Documents*)

La fonction **cmc_send_documents()** est mappée sur des enveloppes d'élément ROSE contenant chacune une opération X.413 de liaison de mémoire de message, une opération X.413 de dépôt de mémoire de messages et une opération de fin de liaison de mémoire de message, constituant une combinaison des fonctions successives **cmc_logon()**, **cmc_send()** et **cmc_logoff()**.

Mappage de paramètres:

```
CMC_return_code
cmc_send_documents(
    CMC_string          recipient_addresses,   X.411.enveloppe.destinataires
                                               et usagers X.420 donnant
                                               l'autorisation; destinataires
                                               primaires, en copie, en copie
                                               aveugle; voir le mappage du
                                               destinataire CMC
    CMC_string          subject,               X.420.en-tête.sujet
    CMC_string          text_note,             première partie
                                               X.420.corps.par-
                                               tie_de_corpsIA5Text.données
    CMC_flags           send_doc_flags,        fanions fournis par
                                               l'appelant
    CMC_string          file_paths,            noms locaux de fichiers, non
                                               transmis dans le message
                                               X.400
    CMC_string          attach_titles          X.420.EDBP.don-
                                               nées.descripteur_de_va-
                                               leur_de_données
    CMC_string          delimiter,             caractère délimiteur
    CMC_ui_id           ui_id,                 NULL
);
```

Commentaires supplémentaires:

néant.

B.2.5.6.10 Demande de configuration CMC (*CMC Query Configuration*)

Cet appel ne nécessite pas de mappage ni d'appels X.400. Il se borne à renvoyer la configuration de l'entité spécifiée.

Mappage de paramètres:

```
CMC_return_code
cmc_query_configuration(
   CMC_session_id        session,              identificateur local de
                                               session
   CMC_enum              item,                 mise en œuvre locale
   CMC_buffer            reference,            mise en œuvre locale
   CMC_extension         *config_extensions    NULL
);
```

Commentaires supplémentaires:

néant.

Annexe C

Exemples de programmation

C.1 Exemples de programmation

La présente Recommandation donne les exemples de programmation suivants:

C.1.1 Demande de configuration, ouverture et fermeture de session

```
/* variables locales utilisées */

CMC_return_code      Status;
CMC_boolean          UI_available;
CMC_session_id       Session;

/* rechercher si une interface utilisateur est disponible */
/* avant de démarrer     */

Status = cmc_query_configuration(
          NULL,                          /* pas d'identificateur de
                                            session.                    */
          CMC_CONFIG_UI_AVAIL,           /* interface utilisateur
                                            disponible?                 */
          &UI_available,                 /* valeur de retour.           */
          NULL);                         /* pas d'extensions.           */
       /* traitement d'erreur */

/* ouverture d'une session avec le système en utilisant l'interface UI */

Status = cmc_logon(
          NULL,                          /* service par défaut.         */
          NULL,                          /* demander le nom de l'usager.  */
          NULL,                          /* demander le mot de passe.     */
          NULL,                          /* jeu de caractères par défaut. */
          (CMC_ui_id)NULL,               /* identificateur d'interface UI
                                            par défaut.                 */
          CMC_VERSION,                   /* appel CMC en version 1.     */
          CMC_LOGON_UI_ALLOWED |         /* interface UI d'ouverture de
                                            session.                    */
          CMC_ERROR_UI_ALLOWED,          /* utiliser l'interface UI pour
                                            les erreurs.                */
          &Session,                      /* identificateur de session
                                            renvoyé.                    */
          NULL);                         /* pas d'extensions.           */
       /* traitement d'erreur */

/* effectuer divers appel CMC */

/* fermeture de session avec la mise en œuvre */

Status = cmc_logoff(

          Session,                       /* identificateur de session.    */
          (CMC_ui_id)NULL,               /* pas d'interface UI utilisée.  */
          0,                             /* pas de fanions.               */
          NULL);                         /* pas d'extensions.             */
       /* traitement d'erreur */
```

C.1.2 Fonctions d'émission et d'émission de documents

```
/* variables locales utilisées */

CMC_attachment       Attach;
CMC_session_id       Session;
CMC_message          Message;
CMC_recipient        Recip[2];
CMC_return_code      Status;

/* construire une liste de deux destinataires, ajouter un destinataire "TO". */

Recip[0].name           = "Bob Weaver";          /* envoyer à Bob Weawer.    */
Recip[0].name_type      = CMC_TYPE_INDIVIDUAL;   /* Bob est une personne.    */
Recip[0].address        = NULL;                  /* rechercher l'adresse
                                                    de Bob.                  */
```

```
Recip[0].role           = CMC_ROLE_TO;          /* il est un destina-
                                                   taire "TO".              */
Recip[0].flags          = 0;                    /* pas le dernier élément
                                                   dest.                    */
Recip[0].extensions     = NULL;                 /* pas d'extens. destina-
                                                   taire.                   */

/* ajouter un destinataire CC. */

Recip[1].name           = "Mary Yu";            /* envoyer à Mary Yu.       */
Recip[1].name_type      = CMC_TYPE_INDIVIDUAL;  /* Mary est une personne.   */
Recip[1].address        = NULL;                 /* rechercher l'adresse
                                                   de Mary.                 */
Recip[1].role           = CMC_ROLE_CC;          /* elle est un destina-
                                                   taire "C".               */
Recip[1].flags          = CMC_RECIP_LAST_ELEMENT;/* dernier élément destina-
                                                   taire.                   */
Recip[1].extensions     = NULL;                 /* pas d'extensions
                                                   destinataire.            */

/* attacher un fichier. */

Attach.attach_title     = "stock.wks";          /* nom de fichier original. */
Attach.attach_typ       = NULL;                 /* pas de type spécifique.  */
Attach.attach_filename  = "tmp22.tmp";          /* fichier à attacher.      */
Attach.attach_flags     = CMC_ATT_LAST_ELEMENT; /* dernier attachement.     */
Attach.attach_extensions= NULL;                 /* pas d'extension
                                                   d'attachement.           */

/* assembler dans la structure de message. */

Message.message_reference  = NULL;          /* ignoré pour un appel
                                               cmc_send.                    */
Message.message_type       = NULL;          /* type de msg interpersonnel. */
Message.subject            = "Stock";       /* sujet du message.           */
Message.time_sent          = NULL;          /* ignorée pour cmc_send.      */
Message.text_note          = "Time to buy"; /* note de message.            */
Message.recipients         = Recip;         /* destinataires du message.   */
Message.attachments        = &Attach;       /* attachments du message.     */
Message.message_flags      = 0;             /* pas de fanions.             */
Message.message_extensions = NULL;          /* pas d'extensions de
                                               message.                     */

/* Emettre le message! */

Status = cmc_send(
          Session,                /* id. de session – Dans l'appel d'ouver-
                                     ture de session.                       */
          &Message,               /* structure de message.                  */
          0,                      /* pas de fanions.                        */
          (CMC_ui_id)NULL,        /* pas d'interface UI utilisée.           */
          NULL);                  /* pas d'extensions.                      */
       /* traitement d'erreur */

/* faire maintenant la même chose avec l'envoi de documents et l'interface UI */

Status = cmc_send_documents(
          "to:Bob Weaver,cc:Mary Yu",       /* destinataire du message.    */
          "Stock",                          /* sujet du message.           */
          "Time to buy",                    /* note du message.            */
          CMC_LOGON_UI_ALLOWED |
          CMC_SEND_UI_REQUESTED |
          CMC_ERROR_UI_ALLOWED,             /* fanions (autorisent
                                               diverses UI).               */
          "stock.wks",                      /* fichier à attacher.         */
          "tmp22.tmp",                      /* nom de fich. à véhiculer avec
                                               l'attachement.              */
          ",",                              /* délimiteur de valeur multiple.*/
          NULL);                            /* identificateur par défaut de
                                               l'interface UI.             */
       /* traitement d'erreur */
```

C.1.3 Lister, lire et supprimer le premier message non lu

```
/* variables locales utilisées */

CMC_message_summary  *pMsgSummary;
CMC_message          *pMessage;
CMC_uint32           iCount;

/* lire le premier message non lu et le supprimer */

iCount = 5;

Status = cmc_list(
          Session,                /* identificateur de session.              */
          NULL,                   /* lister TOUS les types de message.       */
          CMC_LIST_UNREAD_ONLY,   /* ne recevoir que les messages non lus.   */
          NULL,                   /* partir du haut.                         */
          &iCount,                /* compteur de messages entrée/sortie.     */
          (CMC_ui_id)NULL,        /* pas d'interface UI utilisée.            */
          &pMsgSummary,           /* retour de la liste résumée des messages.*/
          NULL);                  /* pas d'extensions.                       */
       /* traitement d'erreur */

Status = cmc_read(
          Session,                            /* identificateur de session.    */
          pMsgSummary[0]->message_reference, /* message à lire.               */
          CMC_MSG_AND_ATT_HDRS_ONLY,          /* pas de fichiers d'attachement.*/
          &pMessage,                          /* message renvoyé.              */
          (CMC_ui_id)NULL,                    /* pas d'interface utilisateur.  */
          NULL);                              /* pas d'extensions.             */
       /* traitement d'erreur */

Status = cmc_act_on(
          Session,                            /* identificateur de session.    */
          pMsgSummary[0]->message_reference, /* message à supprimer.          */
          CMC_ACT_ON_DELETE,                  /* message à lire.               */
          0,                                  /* pas de fanions.               */
          (CMC_ui_id)NULL,                    /* pas d'interface utilisateur.  */
          NULL);                              /* pas d'extensions.             */
       /* traitement d'erreur */

/* libérer la mémoire allouée par la mise en œuvre */

Status = cmc_free(pMsgSummary);
Status = cmc_free(pMessage);

/* faire de même, sans la liste d'appel, car l'appel de lecture peut obtenir */
/* le premier message de courrier non lu */

Status = cmc_read(
          Session,                            /* identificateur de session.    */
          NULL,                               /* lire le premier message.      */
          CMC_READ_FIRST_UNREAD_MESSAGE |     /* recevoir le message non lu.   */
          CMC_MSG_AND_ATT_HDRS_ONLY,          /* pas de fichiers d'attachement.*/
          &pMessage,                          /* message renvoyé.              */
          (CMC_ui_id)NULL,                    /* pas d'interface utilisateur.  */
          NULL);                              /* pas d'extensions.             */
       /* traitement d'erreur */

Status = cmc_act_on(
          Session,                            /* identificateur de session.    */
          pMessage->message_reference,        /* message à supprimer.          */
          CMC_ACT_ON_DELETE,                  /* message à lire.               */
          0,                                  /* pas de fanions.               */
          (CMC_ui_id)NULL,                    /* pas d'interface utilisateur.  */
          NULL);                              /* pas d'extensions.             */
       /* traitement d'erreur */

/* libérer la mémoire allouée par la mise en œuvre */

Status = cmc_free(pMessage);
```

C.1.4 Recherche un destinataire donné et obtenir ses détails

```
/* variables locales utilisées */

CMC_session_id       Session;
CMC_recipient        *pRecipient;
CMC_recipient        Recip;
CMC_return_code      Status;

/* recherche de nom pour prendre le destinataire correct */

Recip.name              = "Bob Stack";          /* envoyer à Bob Weawer.     */
Recip.name_type         = CMC_TYPE_INDIVIDUAL;  /* Bob est une personne.     */
Recip.address           = NULL;                 /* rechercher l'adresse
                                                   de Bob.                   */
Recip.role              = NULL;                 /* le rôle n'est pas
                                                   utilisé.                  */
Recip.recip_flags       = 0;                    /* pas de fanions.           */
Recip.recip_extensions  = NULL;                 /* pas d'extension
                                                   destinataire.             */

Status = cmc_look_up(
         Session,                           /* identificateur de session.    */
         &Recip,                            /* nom à rechercher.             */
         CMC_LOOKUP_RESOLVE_UI |            /* résoudre à l'aide de l'UI.    */
         CMC_ERROR_UI_ALLOWED,              /* afficher les erreurs sur
                                               l'UI.                         */
         (CMC_ui_id)NULL,                   /* id. interface UI par défaut.  */
         1,                                 /* un seul destinataire renvoyé. */
         pRecipient,                        /* point. destinataire en retour.*/
         NULL);                             /* pas d'extensions.             */

/* afficher les détails stockés pour ce destinataire */

Status = cmc_look_up(
         Session,                           /* identificateur de session .   */
         pRecipient,                        /* nom concerné par les détails
                                               demandés.                     */
         CMC_LOOKUP_DETAILS_UI |            /* afficher les détails sur
                                               l'interface UI.               */
         CMC_ERROR_UI_ALLOWED,              /* afficher les erreurs sur
                                               l'interface UI.               */
         (CMC_ui_id)NULL,                   /* identificateur par défaut de
                                               l'interface UI.               */
         0,                                 /* pas de limite de comptage de
                                               retour.                       */
         NULL,                              /* pas d'enregistrements
                                               renvoyés.                     */
         NULL);                             /* pas d'extensions.             */

/* libérer la mémoire allouée par la mise en œuvre */

cmc_free(pRecipient);
```

C.1.5 Utilisation d'extensions

```
/* variables locales utilisées */

CMC_return_code      Status;
CMC_session_id       Session;
CMC_extension        Extension;
CMC_X_COM_support    Supported[2];
CMC_uint16           index;

/* trouver si l'ensemble commun d'extensions est pris en charge,      */
/* mais la prise en charge de COM_X_CONFIG_DATA n'est pas nécessaire  */

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,                              /* pas d'identificateur de
                                               session.                    */
         CMC_CONFIG_UI_AVAIL,               /* voir si une interface UI est
                                               disponible.                 */
         &UI_available,                     /* valeur renvoyée.            */
         &Extension);                       /* passer dans des extensions. */
      /* traitement d'erreur */
if (Supported[0].flags & CMC_X_COM_NOT_SUPPORTED)
      return FALSE;  /* extensions communes nécessaires non dispo */

/* ouverture de session et obtenir les extensions de données pour la 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,                              /* Service par défaut.         */
         NULL,                              /* demander le nom de l'usager.*/
         NULL,                              /* demander le mot de passe.   */
         NULL,                              /* jeu de caractères par défaut.*/
         (CMC_ui_id)NULL,                   /* id. de l'interface utilisateur
                                               par défaut.                 */
         CMC_VERSION,                       /* appels CMC version 1.       */
         CMC_LOGON_UI_ALLOWED |             /* interface UI pour l'ouverture
                                               de session.                 */
         CMC_ERROR_UI_ALLOWED,              /* utiliser l'UI pour afficher
                                               les erreurs.                */
         &Session,                          /* identificateur de session
                                               renvoyé.                    */
         &Extension);                       /* extensions d'ouverture de
                                               session.                    */
      /* traitement d'erreur */
if (Supported[0].flags & CMC_X_COM_NOT_SUPPORTED)
      return FALSE;  /* extensions communes non disponibles */
      /* les extensions de données communes seront utilisées */
      /* dans cette session */

/* exemple de libération de données renvoyées par l'interface CMC */
/* dans des extensions de fonction en sortie.                     */

for (index = 0; ; index++)  {
      if (Extensions[index].extension_flags & CMC_EXT_OUTPUT) {
         if (cmc_free(Extensions[index].item_reference) != CMC_success)     {
            /* les erreurs non attendues sont traitées à cet endroit */
            }
            }
         (Extensions[index].extension_flags & CMC_EXT_LAST_ELEMENT)
            break;
      }

/* effectuer divers appel CMC */

/* fermeture de session avec la mise en œuvre */

Status = cmc_logoff(
         Session,                /* identificateur de session.             */
         (CMC_ui_id)NULL,        /* pas d'interface UI utilisée.           */
         0,                      /* pas de fanions.                        */
         NULL);                  /* pas d'extensions.                      */
      /* traitement d'erreur */
```

C.1.6 Mise en œuvre de la liaison CMC

```
   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")))
       {
       /* traitement d'erreur */
       }

if (!(lpfnCMCBindImplementation =
   (CMC_P_BIND_IMPLEMENTATION)GetProcAddress (hlibCMC, "cmc_bind_implement-
    ation")))
       {
       /* traitement d'erreur */
       }

/* appeler un gestionnaire CMC sélectionné afin d'établir une liaison */
/* avec une mise en œuvre CMC spécifique donnée.                      */

Status = lpfnCMCBindImplementation(selected_implementation_name,
                            &pDispatchTable,
                            NULL);

if (pDispatchTable == NULL)
       {
       /* traitement d'erreur */
       }

/* vérifier si une interface utilisateur est disponible pour cette */
/* mise en œuvre avant de commencer. Attention, mélange d'appels. */

Status = pDispatchTable->cmc_query_configuration(
       NULL,                              /* pas d'identificateur de
                                             session.                  */
       CMC_CONFIG_UI_AVAIL,               /* voir si une interface UI
                                             est disponible.           */
       &UI_available,                     /* valeur renvoyée.          */
       NULL);                             /* pas d'extensions.         */
   /* traitement d'erreur */

/* ouvrir une session sur le système en utilisant l'interface utilisateur. */

Status = pDispatchTable->cmc_logon(
       NULL,                              /* service par défaut.       */
       NULL,                              /* demander le nom de l'usager.  */
       NULL,                              /* demander le mot de passe.     */
       NULL,                              /* jeu de caractères par défaut. */
       (CMC_ui_id)NULL,                   /* identificateur de l'inter-
                                             face UI par défaut.       */
       CMC_VERSION,                       /* appels CMC version 1.     */
       CMC_LOGON_UI_ALLOWED |             /* interface UI pour l'ouverture
                                             de session.               */
       CMC_ERROR_UI_ALLOWED,              /* utiliser l'UI pour afficher
                                             les erreurs.              */
       &Session,                          /* identificateur de session
                                             renvoyé.                  */
       NULL);                             /* pas d'extensions.         */
   /* traitement d'erreur */

/* effectuer les appels à destination d'une mise en œuvre CMC donnée. */

Status = pDispatchTable->cmc_get_root_handle(Session,
                            &root_object_handle,
                            NULL);
   /* traitement d'erreur */

/* fermeture de la session avec la mise en œuvre */

Status = pDispatchTable->cmc_logoff(
       Session,                /* identificateur de session.  */
       (CMC_ui_id)NULL,        /* pas d'interface utilisée.   */
       0,                      /* pas de fanions.             */
       NULL);                  /* pas d'extensions.           */
   /* traitement d'erreur */
```

```
   /* faire libérer par la mise en œuvre la table de distribution */
   /* qu'elle a créée.                                             */

   pDispatchTable->cmc_free(pDispatchTable);

   /* délier la mise en œuvre CMC, libérer la mémoire utilisée */
   /* par le gestionnaire CMC et/ou la mise en œuvre,          */
   /* sauf la table de distribution CMC.                       */

   cmc_unbind_implementation(selected_implementation_name,
                             NULL);

      /* traitement d'erreur */

   /* libérer l'instance de bibliothèque DLL. */

   FreeLibrary (hlibCMC);
```

C.2 Exemple de mise en œuvre de liaison de mise en œuvre CMC

```
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

      /* mémorisation locale pour un traitement ultérieur par cmc_free. */

   /* remplir la table de distribution. */

   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;

    /* charger la variable de sortie. */

    *dispatch_table = pDispatchTable;

    return CMC_SUCCESS;
}
```

C.3 Composition d'un message

```
    #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];

    /* remplir la structure de propriétés du destinataire. */

    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;
```

```
/* remplir la structure de propriétés du message. */

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;

/* remplir la structure de propriétés d'article de contenu de message. */

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;

/* Créer un objet destinataire. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                     &Recipient,
                                     CMC_TYPE_OC_RECIPIENT,
                                     NULL);
   /* traitement d'erreur */
```

```
/* remplir l'objet destinataire avec certaines propriétés. */

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 - Le paramètre d'ajout de propriétés de l'appel */
   /* cmc_add_properties ci-dessous aurait pu être utilisé */
   /* pour obtenir l'information d'erreur par propriété.   */
   /* traitement d'erreur */
   }

/* Créer un objet message. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                    &Message,
                                    CMC_TYPE_OC_MESSAGE,
                                    NULL);
   /* traitement d'erreur */

/* remplir l'objet message avec certaines propriétés. */

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 – Le paramètre d'ajout de propriétés de l'appel    */
   /* cmc_add_properties ci-dessous aurait pu être utilisé    */
   /* pour obtenir l'information d'erreur par propriété.      */
   /* traitement d'erreur */
   }

/* Créer un objet article de contenu. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                    &ContentItem,
                                    CMC_TYPE_OC_CONTENT_ITEM,
                                    NULL);
   /* traitement d'erreur */

/* remplir l'objet article de contenu avec certaines propriétés. */

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 - Le paramètre d'ajout de propriétés de l'appel    */
   /* cmc_add_properties ci-dessous aurait pu être utilisé    */
   /* pour obtenir l'information d'erreur par propriété.      */
   /* traitement d'erreur */
   }
```

```
/* transférer maintenant les objets destinataire et article de contenu */
/* dans l'objet message.                                                */

Status = pDispatchTable->cmc_copy_object(Message,
                                         Recipient,
                                         &Message,
                                         NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                         Message,
                                         &error_buf,
                                         NULL);
   /* traitement d'erreur */
   }

Status = pDispatchTable->cmc_copy_object(Message,
                                         ContentItem,
                                         &Message,
                                         NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                         Message,
                                         &error_buf,
                                         NULL);
   /* traitement d'erreur */
   }

/* tentative d'envoi de message. */

Status = pDispatchTable->cmc_send_message_object(Message,
                                         NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                         Message,
                                         &error_buf,
                                         NULL);
   /* traitement d'erreur */
   }

/* nettoyage. */

cfree(MessageBody.data);

pDispatchTable->cmc_free(ContentItem);
pDispatchTable->cmc_free(Recipient);
pDispatchTable->cmc_free(Message);
pDispatchTable->cmc_free(error_buf);
```

C.4 Recherche de nouveaux messages

```
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;
```

```
/* obtenir le descripteur opaque racine. */

Status = pDispatchTable->cmc_get_root_handle(
                                    Session,
                                    &root_object_handle,
                                    NULL);
   /* traitement d'erreur */

/* ouvrir un descripteur opaque pour le conteneur racine, commencer   */
/* par établir une contrainte afin de trouver le dossier de la boîte  */
/* aux lettres en entrée, dans l'hypothèse où il en existe une seule. */

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);
   /* traitement d'erreur */

/* enregistrement pour la supervision de nouveaux messages          */

/* définir les conteneurs pour lesquelle l'arrivée de messages dans */
/* la boîte aux lettres en entrées est vérifiée                     */

CheckData.number_containers = 1;
CheckData.containers = &RootCursor;

Status = pDispatchTable->cmc_register(
                         Session,
                         NewMessageFlag,
                         NULL,
                         (CMC_buffer) &CheckData,
                         NULL);
   /* traitement d'erreur */

/* superviser l'arrivée de nouveaux messages */

Status = pDispatchTable->cmc_check_event(
                         Session,
                         NewMessageFlag,
                         MinTimeOut,
                         (CMC_buffer) &CheckData,
                         CallBackData,
                         NULL);
   /* traitement d'erreur */

if (CheckData != NULL) {
   printf("You have new mail!\n");
}

/* positionner le rappel automatique pour les nouveaux messages */

Status = pDispatchTable->cmc_register_event(
                              Session,
                              NewMessageFlag,
                              NewMessageCallBack,
                              (CMC_buffer) &CheckData,
                              NULL);
   /* traitement d'erreur */

/* forcer la fonction de rappel automatique */

Status = pDispatchTable->cmc_call_callbacks(
                              Session,
                              NewMessageFlag,
                              NULL);
   /* traitement d'erreur */
```

```
   /* résilier pour l'événement de nouveaux messages */

   Status = pDispatchTable->cmc_unregister_event(
                               Session,
                               NewMessageFlag,
                               NewMessageCallBack,
                               (CMC_buffer) &CheckData,
                               NULL);
      /* traitement d'erreur */

   /* nettoyage. */

   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 Remplissage d'un message

```
   #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];
```

```
/* obtenir le descripteur opaque d'objet racine. */

Status = pDispatchTable->cmc_get_root_handle(Session,
                                &root_object_handle,
                                NULL);
   /* traitement d'erreur */

/* ouvrir un curseur pour le conteneur racine, commencer par établir */
/* une contrainte pour trouver les dossiers projets en faisant       */
/* l'hypothèse de l'existence de ces dossiers.                       */

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);
   /* traitement d'erreur */

Status = pDispatchTable->cmc_list_objects(&RootCursor,
                         &FolderCount,
                         &hFolder,
                         NULL);
   /* traitement d'erreur */

/* créer et remplir un message.                          */

/* charger une structure de propriétés de destinataire.    */

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;

/* charger une structure de propriétés de message. */

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;
```

```
/* charger une structure de propriétés d'article de contenu de message. */

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;

/* créer un objet destinataire. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                    &Recipient,
                                    CMC_TYPE_OC_RECIPIENT,
                                    NULL);

   /* traitement d'erreur */

/* remplir l'objet destinataire avec certaines propriété. */

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 - Le paramètre d'ajout de propriétés de l'appel */
   /* cmc_add_properties ci-dessous aurait pu être utilisé */
   /* pour obtenir l'information d'erreur par propriété.   */
   /* traitement d'erreur */
   }
```

```
/* créer un objet message. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                    &Message,
                                    CMC_TYPE_OC_MESSAGE,
                                    NULL);
   /* traitement d'erreur */

/* remplir l'objet message avec certaines propriétés. */

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 - Le paramètre d'ajout de propriétés de l'appel */
   /* cmc_add_properties ci-dessous aurait pu être utilisé */
   /* pour obtenir l'information d'erreur par propriété.   */
   /* traitement d'erreur */
   }

/* créer un objet d'article de contenu. */

Status = pDispatchTable->cmc_open_object_handle(Session,
                                    &ContentItem,
                                    CMC_TYPE_OC_CONTENT_ITEM,
                                    NULL);
   /* traitement d'erreur */

/* remplir l'objet message avec certaines propriétés. */

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 - Le paramètre d'addition de propriétés de l'appel */
   /* cmc_add_properties ci-dessous aurait pu être utilisé    */
   /* pour obtenir l'information d'erreur par propriété.      */
   /* traitement d'erreur */
   }

/* Transférer maintenant les objets destinataire */
/* et article de contenu dans l'objet message.   */

Status = pDispatchTable->cmc_copy_object(Message,
                             Recipient,
                             &Message,
                             NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                             Message,
                             &error_buf,
                             NULL);
   /* traitement d'erreur */
   }
```

```
Status = pDispatchTable->cmc_copy_object(Message,
                              ContentItem,
                              &Message,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              Message,
                              &error_buf,
                              NULL);
   /* traitement d'erreur */
   }

/* transférer le message dans le dossier projets. */

Status = pDispatchTable->cmc_copy_object(hFolder,
                              Message,
                              &Message,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              hFolder,
                              &error_buf,
                              NULL);
   /* traitement d'erreur */
   }

Status = pDispatchTable->cmc_commit_object(Message,
                              NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                              hFolder,
                              &error_buf,
                              NULL);
   /* traitement d'erreur */
   }

/* nettoyage. */

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 Suppression d'un message

```
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; /* hypothèse 1 dossier projet. */
CMC_sint32              MessageCount = 1; /* hypothèse une entrée      */
CMC_string              error_buf = NULL;
CMC_enum                DeletedTypeFlag = CMC_MCT_DELETED;

/* ouvrir un curseur pour le conteneur racine, commencer par établir */
/* une contrainte pour trouver le dossier projet à supprimer         */
/* en faisant l'hypothèse de l'existence de ce dossier.              */
```

```
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);
   /* traitement d'erreur */

Status = pDispatchTable->cmc_list_objects(&RootCursor,
                                    &FolderCount,
                                    &hDeletedFolder,
                                    NULL);
   /* traitement d'erreur */

/* NOTE - Dans l'hypothèse où le code de l'interface utilisateur a */
/* positionné un curseur de dossier sur une entrée de message dans */
/* une boîte de liste correspondant au message à supprimer.        */

/* obtenir le message à supprimer. */

Status = pDispatchTable->cmc_list_objects(&FolderCursor,
                                    &MessageCount,
                                    &Message,
                                    NULL);
   /* traitement d'erreur */

/* déplacer d'abord le message dans le dossier de suppression. */

Status = pDispatchTable->cmc_copy_object(hDeletedFolder,
                                    Message,
                                    &MessageInDeleted,
                                    NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                    hFolder,
                                    &error_buf,
                                    NULL);
   /* traitement d'erreur */
   }

Status = pDispatchTable->cmc_commit_object(MessageInDeleted,
                                    NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                    hDeletedFolder,
                                    &error_buf,
                                    NULL);
   /* traitement d'erreur */
   }

/* supprimer maintenant le message d'une manière permanente dans le    */
/* dossier source, le descripteur opaque du message n'est plus valide. */

Status = pDispatchTable->cmc_delete_objects(MessageCount,
                                    &Message,
                                    NULL);

if (Status != CMC_SUCCESS)
   {
   pDispatchTable->cmc_get_last_error(Session,
                                    Message,
                                    &error_buf,
                                    NULL);
   /* traitement d'erreur */
   }
```

```
/* nettoyage. */

pDispatchTable->cmc_free(hDeletedFolder);
pDispatchTable->cmc_free(MessageInDeleted);
pDispatchTable->cmc_free(RootCursor);
pDispatchTable->cmc_free(error_buf);
```

C.7 Extraction d'un message

```
#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;

/* obtenir le descripteur opaque d'objet racine. */

Status = pDispatchTable->cmc_get_root_handle(Session,
                                     &root_object_handle,
                                     NULL);
   /* traitement d'erreur */

/* ouvrir un curseur pour le conteneur racine, commencer par établir */
/* une contrainte pour trouver le dossier de boîte aux lettres en    */
/* entrée dans l'hypothèse d'un seul tel dossier.                    */

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);
   /* traitement d'erreur */

Status = pDispatchTable->cmc_list_objects(&RootCursor,
                              &FolderCount,
                              &hFolder,
                              NULL);
   /* traitement d'erreur */

/* construire une contrainte sur le dossier, extraire tous les 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;

/* ouvrir un conteneur pour le dossier. */

Status = pDispatchTable->cmc_open_cursor(hFolder,
                              &FolderRestriction,
                              0,
                              NULL,
                              &FolderCursor,
                              NULL);
   /* traitement d'erreur */
```

```
/* énumérer tous les messages présents dans la boîte aux lettres */
/* en entrée en lots de taille MAX_CACHE.                        */

while (MessageCount != 0)
   {
Status = pDispatchTable->cmc_list_objects(&FolderCursor,
                              &MessageCount,
                              &Messages,
                              NULL);
   /* traitement d'erreur */

/* construire un tableau de propriétés contenant les propriétés   */
/* désirées (voir l'exemple de composition de message), appeler   */
/* cmc_read_properties et afficher dans la boîte de liste pour    */
/* chaque objet message renvoyé.                                  */

/* NOTE - Les descripteurs opaques d'objets individuels n'ont pas */
/* besoin d'être copiés par un appel à cmc_copy_object_handle()   */
/* avant l'invocation de cmc_free() avec ce pointeur.             */

pDispatchTable->cmc_free(Messages);

}

/* nettoyage. */

pDispatchTable->cmc_free(RootCursor);
pDispatchTable->cmc_free(FolderCursor);
pDispatchTable->cmc_free(hFolder);
```

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	**Réseaux pour données et communication entre systèmes ouverts**
Série Z	Langages de programmation