Summary of the MSAM Interface

C Summary

Data Types and Constants

```
/* predefined message creator and message type */
#define kMailAppleMailCreator    'apml' /* message creator */
#define kMailLtrMsgType          'lttr' /* message type for letter, report */

/* predefined block creator and block types */
#define kMailAppleMailCreator    'apml' /* block creator */
#define kMailLtrHdrType          'lthd' /* letter header */
#define kMailContentType         'body' /* content of letter */
#define kMailEnclosureListType   'elst' /* list of enclosures */
#define kMailEnclosureDesktopType'edsk' /* Desktop Mgr info for enclosures */
#define kMailEnclosureFileType   'asgl' /* a file enclosure */
#define kMailImageBodyType       'imag' /* image of letter */
#define kMailMSAMType            'gwyi' /* MSAM-specific information */
#define kMailReportType          'rpti' /* report info */
#define kMailTunnelLtrType       'tunl' /* reserved */
#define kMailHopInfoType         'hopi' /* reserved */

/* families used for mail or related msgs */
#define kMailFamily        'mail' /* letter with content block or
                                     content enclosure */
#define kMailFamilyFile    'file' /* letter without content block or
                                     content enclosure */

#define kMailResolvedList  0     /* MailAttributeID value for resolved
                                    recipient list */

/* mask to test location flags of a slot */
#define MailLocationMask(locationNumber) (1<<((locationNumber)-1))

/* bit flags of MailAttributeID type */
enum {
   kMailLetterFlagsBit     = 1,  /* letter flags bit */
   kMailIndicationsBit     = 3,  /* indications bit */
```

```
   kMailMsgTypeBit         = 4,  /* letter creator & type bit */
   kMailLetterIDBit        = 5,  /* letter ID bit */
   kMailSendTimeStampBit   = 6,  /* send timestamp bit */
   kMailNestingLevelBit    = 7,  /* nesting level bit */
   kMailMsgFamilyBit       = 8,  /* message family bit */
   kMailReplyIDBit         = 9,  /* reply ID bit */
   kMailConversationIDBit  = 10, /* conversation ID bit */
   kMailSubjectBit         = 11, /* subject bit */
   kMailFromBit            = 12, /* From recipient bit */
   kMailToBit              = 13, /* To recipient bit */
   kMailCcBit              = 14, /* cc recipient bit */
   kMailBccBit             = 15  /* bcc recipient bit */
};

/* Values of MailAttributeMask data type. The masks are defined for use
 with the MailAttributeBitmap data type. However, because the
 MailAttributeBitmap data type is defined as a bit field structure, and the
 masks operate on variables of type long, you cannot use these masks to set
 or test the value of a bit field in a MailAttributeBitmap structure. The
 masks are included for historical reasons only. */
enum {
   kMailLetterFlagsMask          = 1L<<(kMailLetterFlagsBit-1),
   kMailIndicationsMask          = 1L<<(kMailIndicationsBit-1),
   kMailMsgTypeMask              = 1L<<(kMailMsgTypeBit-1),
   kMailLetterIDMask             = 1L<<(kMailLetterIDBit-1),
   kMailSendTimeStampMask        = 1L<<(kMailSendTimeStampBit-1),
   kMailNestingLevelMask         = 1L<<(kMailNestingLevelBit-1),
   kMailMsgFamilyMask            = 1L<<(kMailMsgFamilyBit-1),
   kMailReplyIDMask              = 1L<<(kMailReplyIDBit-1),
   kMailConversationIDMask       = 1L<<(kMailConversationIDBit-1),
   kMailSubjectMask              = 1L<<(kMailSubjectBit-1),
   kMailFromMask                 = 1L<<(kMailFromBit-1),
   kMailToMask                   = 1L<<(kMailToBit-1),
   kMailCcMask                   = 1L<<(kMailCcBit-1),
   kMailBccMask                  = 1L<<(kMailBccBit-1)
};

/* bit flags of MailIndications type */
enum {
   kMailOriginalInReportBit      = 1,
   kMailNonReceiptReportsBit     = 3,
   kMailReceiptReportsBit        = 4,
   kMailForwardedBit             = 5,
   kMailPriorityBit              = 6,
```

```
   kMailIsReportWithOriginalBit  = 8,
   kMailIsReportBit              = 9,
   kMailHasContentBit            = 10,
   kMailHasSignatureBit          = 11,
   kMailAuthenticatedBit         = 12,
   kMailSentBit                  = 13
};

/* Masks for bits of MailIndications type. Because the MailIndications data
 type is defined as a bit field structure, and the masks operate on variables
 of type long, you cannot use the masks to set or test the value of a bit
 field in a MailIndications structure. The masks are included for
 historical reasons only. */
enum {
   kMailSentMask                 = 1L<<(kMailSentBit-1),
   kMailAuthenticatedMask        = 1L<<(kMailAuthenticatedBit-1),
   kMailHasSignatureMask         = 1L<<(kMailHasSignatureBit-1),
   kMailHasContentMask           = 1L<<(kMailHasContentBit-1),
   kMailIsReportMask             = 1L<<(kMailIsReportBit-1),
   kMailIsReportWithOriginalMask = 1L<<(kMailIsReportWithOriginalBit-1),
   kMailPriorityMask             = 3L<<(kMailPriorityBit-1),
   kMailForwardedMask            = 1L<<(kMailForwardedBit-1),
   kMailReceiptReportsMask       = 1L<<(kMailReceiptReportsBit-1),
   kMailNonReceiptReportsMask    = 1L<<(kMailNonReceiptReportsBit-1),
   kMailOriginalInReportMask     = 3L<<(kMailOriginalInReportBit-1)
};

/* bit values of the originalInReport field in MailIndications */
enum {
   kMailNoOriginal         = 0,  /* do not enclose original in report */
   kMailEncloseOnNonReceipt= 3   /* enclose original with non-delivery
                                    indication */
};

/* values of MailSegmentType type*/
enum {
   kMailInvalidSegmentType    = 0,
   kMailTextSegmentType       = 1,
   kMailPictSegmentType       = 2,
   kMailSoundSegmentType      = 3,
   kMailStyledTextSegmentType = 4,
   kMailMovieSegmentType      = 5
};
```

```
enum{
   kMailTextSegmentBit,
   kMailPictSegmentBit,
   kMailSoundSegmentBit,
   kMailStyledTextSegmentBit,
   kMailMovieSegmentBit
};

/* values of MailSegmentMask type */
enum {
   kMailTextSegmentMask       = 1L<<kMailTextSegmentBit,
   kMailPictSegmentMask       = 1L<<kMailPictSegmentBit,
   kMailSoundSegmentMask      = 1L<<kMailSoundSegmentBit,
   kMailStyledTextSegmentMask = 1L<<kMailStyledTextSegmentBit,
   kMailMovieSegmentMask      = 1L<<kMailMovieSegmentBit
};

/* values of MailBlockMode type */
enum {
   kMailFromStart = 1,  /* write data at offset from start of block */
   kMailFromLEOB  = 2,  /* write data at offset from end of block */
   kMailFromMark  = 3   /* write data at offset from the current mark */
};

/* bit values of MailLetterSystemFlags type */
enum {
   kMailIsLocalBit   = 2   /* letter is available locally */
   };

enum {
   kMailIsLocalMask  = 1L<<kMailIsLocalBit
};

/* bit values of MailLetterUserFlags type */
enum {
   kMailReadBit,        /* letter has been opened */
   kMailDontArchiveBit, /* reserved */
   kMailInTrashBit      /* reserved */
   };

enum {
   kMailReadMask           = 1L<<kMailReadBit,
   kMailDontArchiveMask    = 1L<<kMailDontArchiveBit,
   kMailInTrashMask        = 1L<<kMailInTrashBit
};
```

```
#define kMailErrorLogEntryVersion 0x101

/* 'STR#' resource IDs for personal MSAM's error and action messages */
#define kMailMSAMErrorStringListID  128 /* list of error message strings */
#define kMailMSAMActionStringListID 129 /* list of action message strings */

/* values of MailLogErrorType type*/
enum {
   kMailELECorrectable     = 0,  /* error correctable by user */
   kMailELEError           = 1,  /* error not correctable by user */
   kMailELEWarning         = 2,  /* warning requiring no user intervention */
   kMailELEInformational   = 3   /* informational message */
};

/* predefined values of MailLogErrorCode type */
enum {
   kMailMSAMErrorCode= 0,     /* MSAM-defined error */
   kMailMiscError    = -1,    /* miscellaneous error */
   kMailNoModem      = -2     /* modem required, but missing */
};

#define kMailMsgSummaryVersion   1

#define kMailMaxPMSAMMsgSummaryData 128/* maximum bytes for private MSAM
                                          message summary data */

/* defines for the addressedToMe field in MailCoreData */
#define kAddressedAs_TO 0x1
#define kAddressedAs_CC 0x2
#define kAddressedAs_BCC 0x4

enum {
   kMailTimerOff        = 0,  /* no timer specified */
   kMailTimerTime       = 1,  /* timer relative to midnight */
   kMailTimerFrequency  = 2   /* frequency timer */
};

/* values of PMSAMStatus type */
enum {
   kPMSAMStatusPending  = 1,  /* for outQueue */
   kPMSAMStatusError    = 2,  /* for inQueue letters */
   kPMSAMStatusSending  = 3,  /* for outQueue */
   kPMSAMStatusCaching  = 4,  /* for inQueue letters */
   kPMSAMStatusSent     = 5   /* for outQueue */
};
```

```
#define kMailEPPCMsgVersion   3

/* values of AOCE high-level event message classes */
enum {
   kMailEPPCCreateSlot        = 'crsl',
   kMailEPPCModifySlot        = 'mdsl',
   kMailEPPCDeleteSlot        = 'dlsl',
   kMailEPPCShutDown          = 'quit',
   kMailEPPCMailboxOpened     = 'mbop',
   kMailEPPCMailboxClosed     = 'mbcl',
   kMailEPPCMsgPending        = 'msgp',
   kMailEPPCSendImmediate     = 'sndi',
   kMailEPPCContinue          = 'cont',
   kMailEPPCSchedule          = 'sked',
   kMailEPPCAdmin             = 'admn',
   kMailEPPCInQUpdate         = 'inqu',
   kMailEPPCMsgOpened         = 'msgo',
   kMailEPPCDeleteOutQMsg     = 'dlom',
   kMailEPPCWakeup            = 'wkup',
   kMailEPPCLocationChanged   = 'locc'
};

/* values of SMSAMAdminCode type */
enum {
   kSMSAMNotifyFwdrSetupChange   = 1,
   kSMSAMNotifyFwdrNameChange    = 2,
   kSMSAMNotifyFwdrPwdChange     = 3,
   kSMSAMGetDynamicFwdrParams    = 4
};

enum {
   kSMSAMFwdrHomeInternetChangedBit,
   kSMSAMFwdrConnectedToChangedBit,
   kSMSAMFwdrForeignRLIsChangedBit,
   kSMSAMFwdrMnMServerChangedBit
};

/* values of SMSAMSlotChanges type */
enum {
   kSMSAMFwdrEverythingChangedMask  = -1,
   kSMSAMFwdrHomeInternetChangedMask= 1L<<kSMSAMFwdrHomeInternetChangedBit,
   kSMSAMFwdrConnectedToChangedMask = 1L<<kSMSAMFwdrConnectedToChangedBit,
   kSMSAMFwdrForeignRLIsChangedMask = 1L<<kSMSAMFwdrForeignRLIsChangedBit,
   kSMSAMFwdrMnMServerChangedMask   = 1L<<kSMSAMFwdrMnMServerChangedBit
};
```

```
enum {
   kOCESetupLocationNone   = 0,  /* disconnect state */
   kOCESetupLocationMax    = 8   /* maximum location value */
};

typedef long MailMsgRef;      /* reference to new/open letter or message */

typedef long MSAMQueueRef;          /* reference to an open MSAM queue */

typedef unsigned short MSAMSlotID;  /* slot identifier */

typedef unsigned short MailSlotID;  /* identifies slots within a mailbox */

typedef long MailboxRef;            /* reference to an active mailbox */

typedef unsigned short MailAttributeID;/* letter attribute identifier */

/* The MailAttributeMask data type defines a set of masks for the
 MailAttributeBitmap data type. However, because the MailAttributeBitmap data
 type is defined as a bit field structure, and the masks operate on variables
 of type long, you cannot use the masks to set or test the value of a bit
 field in a MailAttributeBitmap structure. The MailAttributeMask data type is
 included for historical reasons only. */
typedef unsigned long MailAttributeMask;

typedef IPMMsgID MailLetterID;

typedef unsigned short MailNestingLevel;

typedef OCERecipient MailRecipient;

typedef unsigned short MailSegmentMask;

typedef unsigned short MailSegmentType;

typedef short MailBlockMode;

typedef unsigned short PMSAMStatus;

typedef char OCESetupLocation;            /* current system location */

typedef unsigned char MailLocationFlags;  /* slot location flags */

struct MailBuffer {
   long     bufferSize; /* size of your buffer */
   Ptr      buffer;     /* pointer to your buffer  */
   long     dataSize;   /* amount of data returned in or read out
                           of your buffer */
};

typedef struct MailBuffer MailBuffer;
```

```
struct MailReply {
   unsigned short tupleCount;
   /* tuple[tupleCount] */
};

typedef struct MailReply MailReply;

struct MSAMEnumerateOutQReply {
   long           seqNum;     /* sequence number of message */
   Boolean        done;       /* resolution of message */
   IPMPriority    priority;   /* priority of message */
   OSType         msgFamily   /* message family */
   long           approxSize; /* size of message */
   Boolean        tunnelForm; /* reserved */
   Byte           padByte;    /* for even byte boundary */
   NetworkSpec    nextHop;    /* reserved */
   OCECreatorType msgType;    /* message creator and type */
};

typedef struct MSAMEnumerateOutQReply MSAMEnumerateOutQReply;

struct MSAMEnumerateInQReply {
   long     seqNum;     /* letter sequence number */
   Boolean  msgDeleted; /* should letter be deleted? */
   Boolean  msgUpdated; /* was message summary updated? */
   Boolean  msgCached;  /* is letter in the incoming queue? */
   Byte     padByte;    /* for even byte boundary */
};

typedef struct MSAMEnumerateInQReply MSAMEnumerateInQReply;

struct MailAttributeBitmap {
   unsigned int               /* 32 bits */
      reservedA:16,           /* bits 17 through 32 reserved */
      reservedB:1,            /* bit 16--reserved */
      bcc:1,                  /* bit 15--blind carbon copy recipients */
      cc:1,                   /* bit 14--carbon copy recipients */
      to:1,                   /* bit 13--To recipients */
      from:1,                 /* bit 12--sender of letter */
      subject:1,              /* bit 11--subject of letter */
      conversationID:1,       /* bit 10--ID of conversation thread */
      replyID:1,              /* bit 09--ID of letter being replied to */
      msgFamily:1,            /* bit 08--message family */
      nestingLevel:1,         /* bit 07--nesting level of letter */
      sendTimeStamp:1,        /* bit 06--time letter was sent */
```

```
      letterID:1;             /* bit 05--letter's unique ID number */
      msgType:1,              /* bit 04--letter's creator and type */
      indications:1,          /* bit 03--MailIndications */
      reservedC:1,            /* bit 02--reserved */
      letterFlags:1           /* bit 01--letter flags */
};

typedef struct MailAttributeBitmap MailAttributeBitmap;

struct MailIndications {
unsigned int
   reservedB:16,
   hasStandardContent:1,/* letter has a content block */
   hasImageContent:1,   /* letter an image block */
   hasNativeContent:1,  /* letter has a content enclosure */
   sent:1,              /* letter sent, not just composed */
   authenticated:1,     /* letter was created and transported with
                           authentication */
   hasSignature:1,      /* letter was signed with a digital signature */
   hasContent:1,        /* this letter or a nested letter has content */
   isReport:1,          /* is really a report */
   isReportWithOriginal:1,
                        /* Report contains the original letter */
   priority:2,          /* letter has normal, low, or high priority */
   forwarded:1,         /* letter contains a forwarded letter */
   receiptReports:1,    /* originator requests delivery indications */
   nonReceiptReports:1, /* originator requests non-delivery indications */
   originalInReport:2   /* originator wants original letter enclosed in
                           reports */
};

typedef struct MailIndications MailIndications;

struct OCERecipient {
   RecordID*      entitySpecifier;
   OSType         extensionType;
   unsigned short extensionSize;
   Ptr            extensionValue;
};

struct OCEPackedRecipient {
   unsigned short dataLength; /* length of recipient data */
                  /* followed by recipient data of dataLength bytes */
};
```

```
struct MailOriginalRecipient {
   short    index;      /* index for recipient */
            /* followed by OCEPackedRecipient structure */
};

typedef struct MailOriginalRecipient MailOriginalRecipient;

struct MailResolvedRecipient {
   short       index;         /* index for recipient */
   short       recipientFlags;/* recipient information */
   Boolean     responsible;   /* responsible for delivery? */
   Byte        padByte;
               /* followed by OCEPackedRecipient structure */
};

typedef struct MailResolvedRecipient MailResolvedRecipient;

struct MailEnclosureInfo {
   StringPtr   enclosureName; /* name of the enclosure */
   CInfoPBPtr  catInfo;       /* HFS catalog info about enclosure */
   StringPtr   comment;       /* comment for Get-Info window */
   Ptr         icon;          /* icon for enclosure file */
};

typedef struct MailEnclosureInfo MailEnclosureInfo;

typedef unsigned short MailLogErrorType;

typedef short MailLogErrorCode;

struct MailErrorLogEntryInfo {
   short             version;             /* log entry version */
   UTCTime           timeOccurred;        /* time of error */
   Str31             reportingPMSAM;      /* which MSAM? */
   Str31             reportingMSAMSlot;   /* which slot? */
   MailLogErrorType  errorType;           /* level of error */
   MailLogErrorCode  errorCode;           /* error code */
   short             errorResource;       /* error string resource index */
   short             actionResource;      /* action string resource index */
   unsigned long     filler;              /* reserved */
   unsigned short    filler2;             /* reserved */
};

typedef struct MailErrorLogEntryInfo MailErrorLogEntryInfo;
```

```
struct MailMasterData {
   MailAttributeBitmap  attrMask;      /* indicates attributes present in
                                          MSAMMsgSummary */
   MailLetterID         messageID;     /* ID of this letter *
   MailLetterID         replyID;       /* ID of letter this is a reply to */
   MailLetterID         conversationID;/* ID of letter that started this
                                          conversation */
};

typedef struct MailMasterData MailMasterData;

struct MailCoreData {
   MailLetterFlags      letterFlags;   /* letter status flags */
   unsigned long        messageSize    /* size of letter */
   MailIndications      letterIndications;
                                       /* indications for this letter */
   OCECreatorType       messageType;   /* message creator and type of this
                                          letter */
   MailTime             sendTime;      /* time this letter was sent */
   OSType               messageFamily; /* message family */
   unsigned char        reserved;
   unsigned char        addressedToMe; /* user is To, cc, or bcc recipient */
   char                 agentInfo[6];  /* reserved */
   RString32            sender;        /* sender of this letter */
   RString32            subject;       /* subject of this letter */
};

typedef struct MailCoreData MailCoreData;

struct MSAMMsgSummary {
   short                version;       /* version of the MSAMMsgSummary */
   Boolean              msgDeleted;    /* true if letter is to be deleted by
                                          personal MSAM */
   Boolean              msgUpdated;    /* true if MSAMMsgSummary was updated
                                          by IPM Manager */
   Boolean              msgCached;     /* true if letter is in the inQueue */
   Byte                 padByte;
   MailMasterData       masterData;    /* attributes not essential to
                                          display */
   MailCoreData         coreData;      /* attributes critical to display */
   /* followed by the personal MSAM's private data: Byte PMSAMSpecData[]; */
};

typedef struct MSAMMsgSummary MSAMMsgSummary;
```

```
struct MailLocationInfo {
   OCESetupLocation     location;   /* the current location */
   MailLocationFlags    active;     /* slot's location flags */
};

typedef struct MailLocationInfo MailLocationInfo;

struct MailEPPCMsg {
   short    version;               /* message version */
   union {
      SMCA *   theSMCA;            /* pointer to SMCA */
      long     sequenceNumber;     /* letter sequence number */
      MailLocationInfo locationInfo;/* location information */
   } u;
};

typedef struct MailEPPCMsg MailEPPCMsg;

struct SMCA {
   unsigned short smcaLength; /* length of entire SMCA
                                 (including the length field) */
   OSErr          result;     /* result code */
   long           userBytes;  /* command interpreted user data */
   union{
      CreationID  slotCID;    /* creation ID of record
                                 containing slot information */
      long        msgHint;    /* message reference value */
   } u;
};

typedef struct SMCA SMCA;

typedef unsigned short SMSAMAdminCode;

struct SMSAMAdminEPPCRequest {
   SMSAMAdminCode          adminCode;        /* admin code */
   union {
      SMSAMSetupChange     setupChange;      /* setup change */
      SMSAMNameChange      nameChange;       /* reserved */
      SMSAMPasswordChange  passwordChange;   /* reserved */
      SMSAMDynamicParams   dynamicParams;    /* reserved */
   } u;
};

typedef struct SMSAMAdminEPPCRequest SMSAMAdminEPPCRequest;

typedef unsigned long SMSAMSlotChanges;
```

```
struct SMSAMSetupChange {
   SMSAMSlotChanges  whatChanged;   /* bitmap of changed parameters */
   AddrBlock         serverHint;    /* AOCE server address */
};

typedef struct SMSAMSetupChange SMSAMSetupChange;

struct SMSAMNameChange {      /* reserved data type */
   RString     newName;       /* sever MSAM's new name */
   AddrBlock   serverHint;    /* AOCE server address */
};

typedef struct SMSAMNameChange SMSAMNameChange;

struct SMSAMPasswordChange {  /* reserved data type */
   RString     newPassword;   /* server MSAM's new password */
   AddrBlock   serverHint;    /* AOCE server address */
};

typedef struct SMSAMPasswordChange SMSAMPasswordChange;

struct SMSAMDynamicParams {      /* reserved data type */
   unsigned long  curDiskUsed;   /* disk space used */
   unsigned long  curMemoryUsed; /* memory used */
};

typedef struct SMSAMDynamicParams SMSAMDynamicParams;

struct MailTime {
   UTCTime     time;    /* current UTC (GMT) */
   UTCOffset   offset;  /* offset from UTC */
};

typedef struct MailTime MailTime;

union MailTimer {
   long  frequency;     /* how often to connect */
   long  connectTime;   /* time since midnight */
};

typedef union MailTimer MailTimer;

typedef Byte MailTimerKind;
```

```
struct MailTimers {
   MailTimerKind  sendTimeKind;        /* timer kind for sending */
   MailTimerKind  receiveTimeKind;     /* timer kind for receiving */
   MailTimer      send;                /* connect time or frequency for
                                          sending letters */
   MailTimer      receive;             /* connect time or frequency for
                                          receiving letters */
};

typedef struct MailTimers MailTimers;

struct MailStandardSlotInfoAttribute {
   short             version;       /* MSAM version of the slot */
   MailLocationFlags active;        /* active at location i if
                                       MailLocationMask (i) is set */
   Byte              padByte;
   MailTimers        sendReceiveTimer;
};

typedef struct MailStandardSlotInfoAttribute MailStandardSlotInfoAttribute;

typedef unsigned short  MailLetterSystemFlags;

typedef unsigned short  MailLetterUserFlags;

struct MailLetterFlags {
   MailLetterSystemFlags   sysFlags;   /* system flags */
   MailLetterUserFlags     userFlags;  /* user flags */
};

typedef struct MailLetterFlags MailLetterFlags;

struct MailMaskedLetterFlags {
   MailLetterFlags   flagMask;   /* flags that are to be set */
   MailLetterFlags   flagValues; /* their values */
};

typedef struct MailMaskedLetterFlags MailMaskedLetterFlags;

struct MailBlockInfo {
   OCECreatorType blockType;
   unsigned long  offset;
   unsigned long  blockLength;
};

typedef struct MailBlockInfo MailBlockInfo;
```

```
# define MailParamBlockHeader
   Ptr            qLink;         /* reserved */ \
   long           reservedH1;    /* reserved */ \
   long           reservedH2;    /* reserved */ \
   ProcPtr        ioCompletion;  /* pointer to completion routine */ \
   OSErr          ioResult;      /* result code */ \
   long           saveA5;        /* pointer to global variables */ \
   short          reqCode;       /* reserved */

struct PMSAMGetMSAMRecordPB {
   MailParamBlockHeader
   CreationID  msamCID;
};

typedef struct PMSAMGetMSAMRecordPB PMSAMGetMSAMRecordPB;

struct PMSAMOpenQueuesPB {
   MailParamBlockHeader
   MSAMQueueRef   inQueueRef;
   MSAMQueueRef   outQueueRef;
   MSAMSlotID     msamSlotID;
   long           filler[2];
};

typedef struct PMSAMOpenQueuesPB PMSAMOpenQueuesPB;

struct PMSAMSetStatusPB {
   MailParamBlockHeader
   MSAMQueueRef   queueRef;
   long           seqNum;
   long           msgHint;
   PMSAMStatus    status;
};

typedef struct PMSAMSetStatusPB PMSAMSetStatusPB;

struct PMSAMLogErrorPB {
   MailParamBlockHeader
   MSAMSlotID              msamSlotID; /* 0 for PMSAM errors */
   MailErrorLogEntryInfo*  logEntry;
   long                    filler[2];
};

typedef struct PMSAMLogErrorPB PMSAMLogErrorPB;
```

```
struct PMSAMCreateMsgSummaryPB {
   MailParamBlockHeader
   MSAMQueueRef      inQueueRef;
   long              seqNum;     /* sequence number of new letter */
   MSAMMsgSummary *  msgSummary; /* attributes and mask filled in */
   MailBuffer *      buffer;     /* private MSAM data to add to summary */
};

typedef struct PMSAMCreateMsgSummaryPB PMSAMCreateMsgSummaryPB;

sstruct PMSAMPutMsgSummaryPB {
   MailParamBlockHeader
   MSAMQueueRef           inQueueRef;
   long                   seqNum;
   MailMaskedLetterFlags * letterFlags;
   MailBuffer*            buffer;        /* PMSAM private data */
};

typedef struct PMSAMPutMsgSummaryPB PMSAMPutMsgSummaryPB;

struct PMSAMGetMsgSummaryPB {
   MailParamBlockHeader
   MSAMQueueRef      inQueueRef;
   long              seqNum;
   MSAMMsgSummary *  msgSummary;       /* buffer for message summary */
   MailBuffer *      buffer;           /* buffer for PMSAM private data */
   unsigned short    msgSummaryOffset; /* offset of PMSAM private data
                                          from start of message summary */
};

typedef struct PMSAMGetMsgSummaryPB PMSAMGetMsgSummaryPB;

struct SMSAMSetupPB {
     MailParamBlockHeader
     RecordIDPtr serverMSAM;
     RStringPtr  password;
     OSType      gatewayType;
     RStringPtr  gatewayTypeDescription;
     AddrBlock   catalogServerHint;
 };

typedef struct SMSAMSetupPB SMSAMSetupPB;
```

```
struct SMSAMStartupPB {
   MailParamBlockHeader
   AuthIdentity   msamIdentity;
   MSAMQueueRef   queueRef;
};

typedef struct SMSAMStartupPB SMSAMStartupPB;

struct SMSAMShutdownPB {
   MailParamBlockHeader
   MSAMQueueRef   queueRef;
};

typedef struct SMSAMShutdownPB SMSAMShutdownPB;

struct MSAMEnumeratePB {
   MailParamBlockHeader
   MSAMQueueRef   queueRef;
   long           startSeqNum;
   long           nextSeqNum;
   MailBuffer     buffer;
};

typedef struct MSAMEnumeratePB MSAMEnumeratePB;

struct MSAMDeletePB {
   MailParamBlockHeader
   MSAMQueueRef   queueRef;
   long           seqNum;
   Boolean        msgOnly; /* delete letter and message summary? */
   Byte           padByte;
   OSErr          result;  /* reserved */
};

typedef struct MSAMDeletePB MSAMDeletePB;

struct MSAMOpenPB {
   MailParamBlockHeader
   MSAMQueueRef   queueRef;
   long           seqNum;
   MailMsgRef     mailMsgRef;
};

typedef struct MSAMOpenPB MSAMOpenPB;
```

```
struct MSAMOpenNestedPB {
   MailParamBlockHeader
   MailMsgRef  mailMsgRef;
   MailMsgRef  nestedRef;
};

typedef struct MSAMOpenNestedPB MSAMOpenNestedPB;

struct MSAMClosePB {
   MailParamBlockHeader
   MailMsgRef  mailMsgRef;
};

typedef struct MSAMClosePB MSAMClosePB;

struct MSAMGetMsgHeaderPB {
   MailParamBlockHeader
   MailMsgRef        mailMsgRef;
   IPMHeaderSelector selector;
   unsigned long     offset;
   MailBuffer        buffer;
   unsigned long     remaining;
};

typedef struct MSAMGetMsgHeaderPB MSAMGetMsgHeaderPB;

struct MSAMGetAttributesPB {
   MailParamBlockHeader
   MailMsgRef           mailMsgRef;
   MailAttributeBitmap  requestMask;
   MailBuffer           buffer;
   MailAttributeBitmap  responseMask;
   Boolean              more;
};

typedef struct MSAMGetAttributesPB MSAMGetAttributesPB;

struct MSAMGetRecipientsPB {
   MailParamBlockHeader
   MailMsgRef        mailMsgRef;
   MailAttributeID   attrID;     /* kMailFromBit thru kMailBccBit */
   unsigned short    startIndex; /* starts at 1 */
   MailBuffer        buffer;
   unsigned short    nextIndex;
   Boolean           more;
};

typedef struct MSAMGetRecipientsPB MSAMGetRecipientsPB;
```

```
struct MSAMGetContentPB {
   MailParamBlockHeader
   MailMsgRef        mailMsgRef;
   MailSegmentMask   segmentMask;
   MailBuffer        buffer;
   StScrpRec *       textScrap;
   ScriptCode        script;
   MailSegmentType   segmentType;
   Boolean           endOfScript;
   Boolean           endOfSegment;
   Boolean           endOfContent;
   long              segmentLength;
   long              segmentID;
};

typedef struct MSAMGetContentPB MSAMGetContentPB;

struct MSAMGetEnclosurePB {
   MailParamBlockHeader
   MailMsgRef  mailMsgRef;
   Boolean     contentEnclosure;
   Byte        padByte;
   MailBuffer  buffer;
   Boolean     endOfFile;
   Boolean     endOfEnclosures;
};

typedef struct MSAMGetEnclosurePB MSAMGetEnclosurePB;

struct MSAMEnumerateBlocksPB {
   MailParamBlockHeader
   MailMsgRef        mailMsgRef;
   unsigned short    startIndex; /* starts at 1 */
   MailBuffer        buffer;
   unsigned short    nextIndex;
   Boolean           more;
};

typedef struct MSAMEnumerateBlocksPB MSAMEnumerateBlocksPB;

struct MSAMGetBlockPB {
   MailParamBlockHeader
   MailMsgRef     mailMsgRef;
   OCECreatorType blockType;
   unsigned short blockIndex;
```

```
   MailBuffer     buffer;
   unsigned long  dataOffset;
   Boolean        endOfBlock;
   Byte           padByte;
   unsigned long  remaining;
};

typedef struct MSAMGetBlockPB MSAMGetBlockPB;

struct MSAMMarkRecipientsPB {
   MailParamBlockHeader
   MSAMQueueRef   queueRef;
   long           seqNum;
   MailBuffer     buffer;
};

typedef struct MSAMMarkRecipientsPB MSAMMarkRecipientsPB;

struct MSAMnMarkRecipientsPB {
   MailParamBlockHeader
   MailMsgRef     mailMsgRef;
   MailBuffer     buffer;
};

typedef struct MSAMnMarkRecipientsPB MSAMMarknRecipientsPB;

struct MSAMCreatePB {
   MailParamBlockHeader
   MSAMQueueRef   queueRef;
   Boolean        asLetter;      /* create as letter or message */
   IPMMsgType     msgType;
   long           refCon;        /* for messages only */
   long           seqNum;        /* set if creating message in the inQueue */
   Boolean        tunnelForm;    /* always false */
   Boolean        bccRecipients; /* true if creating letter with bcc
                                    recipients */
   MailMsgRef     newRef;
};

typedef struct MSAMCreatePB MSAMCreatePB;

struct MSAMBeginNestedPB {
   MailParamBlockHeader
   MailMsgRef  mailMsgRef;
   long        refCon;     /* for messages only */
   IPMMsgType  msgType;
};

typedef struct MSAMBeginNestedPB MSAMBeginNestedPB;
```

```
struct MSAMEndNestedPB {
   MailParamBlockHeader
   MailMsgRef  mailMsgRef;
};

typedef struct MSAMEndNestedPB MSAMEndNestedPB;

struct MSAMSubmitPB {
   MailParamBlockHeader
   MailMsgRef     mailMsgRef;    /* message reference number */
   Boolean        submitFlag;    /* submit or delete message? */
   Byte           padByte;
   MailLetterID   msgID;         /* reserved */
};

typedef struct MSAMSubmitPB MSAMSubmitPB;

struct MSAMPutMsgHeaderPB {
   MailParamBlockHeader
   MailMsgRef           mailMsgRef;
   OCERecipient *       replyQueue;
   IPMSender *          sender;
   IPMNotificationType  deliveryNotification;
   IPMPriority          priority;
};

typedef struct MSAMPutMsgHeaderPB MSAMPutMsgHeaderPB;

struct MSAMPutAttributePB {
   MailParamBlockHeader
   MailMsgRef        mailMsgRef;
   MailAttributeID   attrID;
   MailBuffer        buffer;
};

typedef struct MSAMPutAttributePB MSAMPutAttributePB;

struct MSAMPutRecipientPB {
   MailParamBlockHeader
   MailMsgRef        mailMsgRef;
   MailAttributeID   attrID;
   MailRecipient *   recipient;
   Boolean           responsible;   /* for server and message msams only */
};

typedef struct MSAMPutRecipientPB MSAMPutRecipientPB;
```

```
struct MSAMPutContentPB {
   MailParamBlockHeader
   MailMsgRef        mailMsgRef;
   MailSegmentType   segmentType;
   Boolean           append;
   Byte              padByte;
   MailBuffer        buffer;
   StScrpRec *       textScrap;
   Boolean           startNewScript;
   ScriptCode        script;        /* valid when startNewScript is true */
};

typedef struct MSAMPutContentPB MSAMPutContentPB;

struct MSAMPutEnclosurePB {
   MailParamBlockHeader
   MailMsgRef        mailMsgRef;
   Boolean           contentEnclosure;
   Byte              padByte;
   Boolean           hfs;
   Boolean           append;
   MailBuffer        buffer;     /* unused if hfs is true */
   FSSpec            enclosure;
   MailEnclosureInfo addlInfo;
};

typedef struct MSAMPutEnclosurePB MSAMPutEnclosurePB;

struct MSAMPutBlockPB {
   MailParamBlockHeader
   MailMsgRef     mailMsgRef;
   long           refCon;        /* for messages only */
   OCECreatorType blockType;
   Boolean        append;
   MailBuffer     buffer;
   MailBlockMode  mode;
   unsigned long  offset;
};

typedef struct MSAMPutBlockPB MSAMPutBlockPB;
```

```
struct MSAMCreateReportPB {
   MailParamBlockHeader
   MailMsgRef        mailMsgRef;
   MailLetterID      msgID;   /* letter ID of letter being reported on */
   MailRecipient *   sender;  /* sender of the letter being reported on */
};

typedef struct MSAMCreateReportPB MSAMCreateReportPB;

struct MSAMPutRecipientReportPB {
   MailParamBlockHeader
   MailMsgRef  mailMsgRef;
   short       recipientIndex;   /* recipient index in the original letter */
   OSErr       result;           /* result of sending to the recipient */
};

typedef struct MSAMPutRecipientReportPB MSAMPutRecipientReportPB;

struct MailWakeupPMSAMPB {
   MailParamBlockHeader
   CreationID  pmsamCID;
   MailSlotID  mailSlotID;
};

typedef struct MailWakeupPMSAMPB MailWakeupPMSAMPB;

struct MailCreateMailSlotPB {
   MailParamBlockHeader
   MailboxRef  mailboxRef;
   long        timeout;
   CreationID  pmsamCID;
   SMCA        smca;
};

typedef struct MailCreateMailSlotPB MailCreateMailSlotPB;

struct MailModifyMailSlotPB {
   MailParamBlockHeader
   MailboxRef  mailboxRef;
   long        timeout;
   CreationID  pmsamCID;
   SMCA        smca;
};

typedef struct MailModifyMailSlotPB MailModifyMailSlotPB;
```

```
union MSAMParam {
   struct {MailParamBlockHeader} header;

   PMSAMGetMSAMRecordPB       pmsamGetMSAMRecord;
   PMSAMOpenQueuesPB          pmsamOpenQueues;
   PMSAMSetStatusPB           pmsamSetStatus;
   PMSAMLogErrorPB            pmsamLogError;

   SMSAMSetupPB               smsamSetup;
   SMSAMStartupPB             smsamStartup;
   SMSAMShutdownPB            smsamShutdown;

   MSAMEnumeratePB            msamEnumerate;
   MSAMDeletePB               msamDelete;

   MSAMOpenPB                 msamOpen;
   MSAMOpenNestedPB           msamOpenNested;
   MSAMClosePB                msamClose;
   MSAMGetMsgHeaderPB         msamGetMsgHeader;
   MSAMGetAttributesPB        msamGetAttributes;
   MSAMGetRecipientsPB        msamGetRecipients;
   MSAMGetContentPB           msamGetContent;
   MSAMGetEnclosurePB         msamGetEnclosure;
   MSAMEnumerateBlocksPB      msamEnumerateBlocks;
   MSAMGetBlockPB             msamGetBlock;
   MSAMMarkRecipientsPB       msamMarkRecipients;
   MSAMnMarkRecipientsPB      msamnMarkRecipients;

   MSAMCreatePB               msamCreate;
   MSAMBeginNestedPB          msamBeginNested;
   MSAMEndNestedPB            msamEndNested;
   MSAMSubmitPB               msamSubmit;
   MSAMPutMsgHeaderPB         msamPutMsgHeader;
   MSAMPutAttributePB         msamPutAttribute;
   MSAMPutRecipientPB         msamPutRecipient;
   MSAMPutContentPB           msamPutContent;
   MSAMPutEnclosurePB         msamPutEnclosure;
   MSAMPutBlockPB             msamPutBlock;

   MSAMCreateReportPB         msamCreateReport;
   MSAMPutRecipientReportPB   msamPutRecipientReport;
```

```
    PMSAMCreateMsgSummaryPB     pmsamCreateMsgSummary;
    PMSAMPutMsgSummaryPB        pmsamPutMsgSummary;
    PMSAMGetMsgSummaryPB        pmsamGetMsgSummary;

    MailWakeupPMSAMPB           wakeupPMSAM;
    MailCreateMailSlotPB        createMailSlot;
    MailModifyMailSlotPB        modifyMailSlot;
};

typedef union MSAMParam MSAMParam;
```

MSAM Functions

Initializing an MSAM

```
pascal OSErr PMSAMGetMSAMRecord(MSAMParam *paramBlock);
pascal OSErr PMSAMOpenQueues(MSAMParam *paramBlock);
pascal OSErr SMSAMSetup(MSAMParam *paramBlock);
pascal OSErr SMSAMStartup(MSAMParam *paramBlock);
```

Enumerating Messages in a Queue

```
pascal OSErr MSAMEnumerate(MSAMParam *paramBlock, Boolean asyncFlag);
```

Opening an Outgoing Message

```
pascal OSErr MSAMOpen(MSAMParam *paramBlock, Boolean asyncFlag);
```

Reading Header Information

```
pascal OSErr MSAMGetAttributes(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMGetRecipients(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMGetMsgHeader(MSAMParam *paramBlock, Boolean asyncFlag);
```

Reading a Message

```
pascal OSErr MSAMGetContent(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMGetEnclosure(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMEnumerateBlocks(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMGetBlock(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMOpenNested(MSAMParam *paramBlock, Boolean asyncFlag);
```

Marking a Recipient

```
pascal OSErr MSAMnMarkRecipients(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMMarkRecipients(MSAMParam *paramBlock, Boolean asyncFlag);
```

Closing a Message

```
pascal OSErr MSAMClose(MSAMParam *paramBlock, Boolean asyncFlag);
```

Creating, Reading, and Writing Message Summaries

```
pascal OSErr PMSAMCreateMsgSummary(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr PMSAMGetMsgSummary(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr PMSAMPutMsgSummary(MSAMParam *paramBlock, Boolean asyncFlag);
```

Creating a Message

```
pascal OSErr MSAMCreate(MSAMParam *paramBlock, Boolean asyncFlag);
```

Writing Header Information

```
pascal OSErr MSAMPutAttribute(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMPutRecipient(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMPutMsgHeader(MSAMParam *paramBlock, Boolean asyncFlag);
```

Writing a Message

```
pascal OSErr MSAMPutContent(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMPutEnclosure(MSAMParam *paramBlock);
pascal OSErr MSAMPutBlock(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMBeginNested(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMEndNested(MSAMParam *paramBlock);
```

Submitting a Message

```
pascal OSErr MSAMSubmit(MSAMParam *paramBlock);
```

Deleting a Message

```
pascal OSErr MSAMDelete(MSAMParam *paramBlock, Boolean asyncFlag);
```

Generating Log Entries and Reports

```
pascal OSErr PMSAMLogError(MSAMParam *paramBlock);
pascal OSErr MSAMCreateReport(MSAMParam *paramBlock, Boolean asyncFlag);
pascal OSErr MSAMPutRecipientReport(MSAMParam *paramBlock, Boolean asyncFlag);
```

Shutting Down a Server MSAM

```
pascal OSErr SMSAMShutdown(MSAMParam *paramBlock, Boolean asyncFlag);
```

Setting Message Status

```
pascal OSErr PMSAMSetStatus(MSAMParam *paramBlock, Boolean asyncFlag);
```

Personal MSAM AOCE Template Functions

```
pascal OSErr MailCreateMailSlot(MSAMParam *paramBlock);
pascal OSErr MailModifyMailSlot(MSAMParam *paramBlock);
pascal OSErr MailWakeupPMSAM(MSAMParam *paramBlock);
```

Application-Defined Function

```
void MyCompletionRoutine(MSAMParam *paramBlock);
```

Pascal Summary

Data Types and Constants

```
CONST
{ predefined message creator and message type }
kMailAppleMailCreator         =  'apml'; { message creator }
kMailLtrMsgType               =  'lttr'; { message type for letter, report }

{ predefined block creator and block types }
kMailAppleMailCreator         =  'apml'; { block creator }
kMailLtrHdrType               =  'lthd'; { letter header }
kMailContentType              =  'body'; { content of letter }
kMailEnclosureListType        =  'elst'; { list of enclosures }
kMailEnclosureDesktopType     =  'edsk'; { Desktop Mgr info for enclosures }
kMailEnclosureFileType        =  'asgl'; { a file enclosure }
kMailImageBodyType            =  'imag'; { image of letter }
kMailMSAMType                 =  'gwyi'; { MSAM-specific information }
kMailReportType               =  'rpti'; { report info }

{ families used for mail or related msgs }
kMailFamily                   =  'mail'; { letter with content block or
                                           content enclosure }
kMailFamilyFile               =  'file'; { letter without content block or
                                           content enclosure }

kMailResolvedList             =  0;    { MailAttributeID value for resolved
                                          recipient list }
```

```
{ bit flags of MailAttributeID type }
kMailLetterFlagsBit     = 1;  { letter flags bit }
kMailIndicationsBit     = 3;  { indications bit }
kMailMsgTypeBit         = 4;  { letter creator & type bit }
kMailLetterIDBit        = 5;  { letter ID bit }
kMailSendTimeStampBit   = 6;  { send timestamp bit }
kMailNestingLevelBit    = 7;  { nesting level bit }
kMailMsgFamilyBit       = 8;  { message family bit }
kMailReplyIDBit         = 9;  { reply ID bit }
kMailConversationIDBit  = 10; { conversation ID bit }
kMailSubjectBit         = 11; { subject bit }
kMailFromBit            = 12; { From recipient bit }
kMailToBit              = 13; { To recipient bit }
kMailCcBit              = 14; { cc recipient bit }
kMailBccBit             = 15; { bcc recipient bit }

{ Values of MailAttributeMask data type. The masks are defined for use
 with the MailAttributeBitmap data type. However, because the
 MailAttributeBitmap data type is defined as a bit field structure, and the
 masks operate on variables of type LONGINT, you cannot use these masks to
 set or test the value of a bit field in a MailAttributeBitmap structure. The
 masks are included for historical reasons only. }
kMailLetterFlagsMask          = $00000001;{1<<(kMailLetterFlagsBit-1)}
kMailIndicationsMask          = $00000004;{1<<(kMailIndicationsBit-1)}
kMailMsgTypeMask              = $00000008;{1<<(kMailMsgTypeBit-1)}
kMailLetterIDMask             = $00000010;{1<<(kMailLetterIDBit-1)}
kMailSendTimeStampMask        = $00000020;{1<<(kMailSendTimeStampBit-1)}
kMailNestingLevelMask         = $00000040;{1<<(kMailNestingLevelBit-1)}
kMailMsgFamilyMask            = $00000080;{1<<(kMailMsgFamilyBit-1)}
kMailReplyIDMask              = $00000100;{1<<(kMailReplyIDBit-1)}
kMailConversationIDMask       = $00000200;{1<<(kMailConversationIDBit-1)}
kMailSubjectMask              = $00000400;{1<<(kMailSubjectBit-1)}
kMailFromMask                 = $00000800;{1<<(kMailFromBit-1)}
kMailToMask                   = $00001000;{1<<(kMailToBit-1)}
kMailCcMask                   = $00002000;{1<<(kMailCcBit-1)}
kMailBccMask                  = $00004000;{1<<(kMailBccBit-1)}

{ bit flags of MailIndications type }
kMailOriginalInReportBit      = 1;
kMailNonReceiptReportsBit     = 3;
kMailReceiptReportsBit        = 4;
kMailForwardedBit             = 5;
kMailPriorityBit              = 6;
kMailIsReportWithOriginalBit  = 8;
```

```
kMailIsReportBit              = 9;
kMailHasContentBit            = 10;
kMailHasSignatureBit          = 11;
kMailAuthenticatedBit         = 12;
kMailSentBit                  = 13;

{ Masks for bits of MailIndications type. Because the MailIndications data
 type is defined as a bit field structure, and the masks operate on variables
 of type LONGINT, you cannot use the masks to set or test the value of a bit
 field in a MailIndications structure. The masks are included for
 historical reasons only}
kMailSentMask                 = $00001000; {1<<(kMailSentBit-1),
kMailAuthenticatedMask        = $00000800; {1<<(kMailAuthenticatedBit-1),
kMailHasSignatureMask         = $00000400; {1<<(kMailHasSignatureBit-1),
kMailHasContentMask           = $00000200; {1<<(kMailHasContentBit-1),
kMailIsReportMask             = $00000100; {1<<(kMailIsReportBit-1),
kMailIsReportWithOriginalMask = $00000080;
                                       {1<<(kMailIsReportWithOriginalBit-1),
kMailPriorityMask             = $00000060; {3<<(kMailPriorityBit-1),
kMailForwardedMask            = $00000010; {1<<(kMailForwardedBit-1),
kMailReceiptReportsMask       = $00000008; {1<<(kMailReceiptReportsBit-1),
kMailNonReceiptReportsMask    = $00000004; {1<<(kMailNonReceiptReportsBit-1),
kMailOriginalInReportMask     = $00000003; {3<<(kMailOriginalInReportBit-1);

{ Bit values of the originalInReport field in MailIndications }
kMailNoOriginal         = 0;  { do not enclose original in report }
kMailEncloseOnNonReceipt= 3;  { enclose original in non-delivery
                                indication }

{ values of MailSegmentType type}
kMailInvalidSegmentType    = 0;
kMailTextSegmentType       = 1;
kMailPictSegmentType       = 2;
kMailSoundSegmentType      = 3;
kMailStyledTextSegmentType = 4;
kMailMovieSegmentType      = 5;

kMailTextSegmentBit        = 0;
kMailPictSegmentBit        = 1;
kMailSoundSegmentBit       = 2;
kMailStyledTextSegmentBit  = 3;
kMailMovieSegmentBit       = 4;
```

```
{ values of MailSegmentMask type }
kMailTextSegmentMask       = $0001; {1<<kMailTextSegmentBit}
kMailPictSegmentMask       = $0002; {1<<kMailPictSegmentBit}
kMailSoundSegmentMask      = $0004; {1<<kMailSoundSegmentBit}
kMailStyledTextSegmentMask = $0008; {1<<kMailStyledTextSegmentBit}
kMailMovieSegmentMask      = $0010; {1<<kMailMovieSegmentBit}

{ values of MailBlockMode type }
kMailFromStart = 1;  { write data at offset from start of block }
kMailFromLEOB  = 2;  { write data at offset from end of block }
kMailFromMark  = 3;  { write data at offset from the current mark }

{ bit values of MailLetterSystemFlags type }
kMailIsLocalBit   = 2;     { letter is available locally }

kMailIsLocalMask  = $0004; {1<<kMailIsLocalBit}

{ bit values of MailLetterUserFlags type }
kMailReadBit         = 0;     { letter has been opened }
kMailDontArchiveBit  = 1;     { reserved }
kMailInTrashBit      = 2;     { reserved }

kMailReadMask           = $0001; {1<<kMailReadBit}
kMailDontArchiveMask    = $0002; {1<<kMailDontArchiveBit}
kMailInTrashMask        = $0004; {1<<kMailInTrashBit}

kMailErrorLogEntryVersion = $101;

{ 'STR#' resource IDs for personal MSAM's error and action messages }
kMailMSAMErrorStringListID       =  128; { list of error message strings }
kMailMSAMActionStringListID      =  129; { list of action message strings }

{ values of MailLogErrorType type}
kMailELECorrectable     = 0;  { error correctable by user }
kMailELEError           = 1;  { error not correctable by user }
kMailELEWarning         = 2;  { warning requiring no user intervention }
kMailELEInformational   = 3;  { informational message }

{ values of MailLogErrorCode type }
kMailMSAMErrorCode= 0;     { MSAM-defined error }
kMailMiscError    = -1;    { miscellaneous error }
kMailNoModem      = -2;    { modem required, but missing }

kMailMsgSummaryVersion        =  1;
```

```
kMailMaxPMSAMMsgSummaryData   = 128; { maximum bytes for private MSAM
                                       message summary data }

{ defines for the addressedToMe field in MailCoreData }
kAddressedAs_TO   = 1;
kAddressedAs_CC   = 2;
kAddressedAs_BCC  = 4;

kMailTimerOff        = 0;  { no timer specified }
kMailTimerTime       = 1;  { timer relative to midnight }
kMailTimerFrequency  = 2;  { frequency timer }

{ values of PMSAMStatus type }
   kPMSAMStatusPending  = 1;  { for outQueue }
   kPMSAMStatusError    = 2;  { for inQueue letters }
   kPMSAMStatusSending  = 3;  { for outQueue }
   kPMSAMStatusCaching  = 4;  { for inQueue letters }
   kPMSAMStatusSent     = 5;  { for outQueue }

kMailEPPCMsgVersion     = 3;

{ values of AOCE high-level event message classes }
kMailEPPCCreateSlot        = 'crsl';
kMailEPPCModifySlot        = 'mdsl';
kMailEPPCDeleteSlot        = 'dlsl';
kMailEPPCShutDown          = 'quit';
kMailEPPCMailboxOpened     = 'mbop';
kMailEPPCMailboxClosed     = 'mbcl';
kMailEPPCMsgPending        = 'msgp';
kMailEPPCSendImmediate     = 'sndi';
kMailEPPCContinue          = 'cont';
kMailEPPCSchedule          = 'sked';
kMailEPPCAdmin             = 'admn';
kMailEPPCInQUpdate         = 'inqu';
kMailEPPCMsgOpened         = 'msgo';
kMailEPPCDeleteOutQMsg     = 'dlom';
kMailEPPCWakeup            = 'wkup';
kMailEPPCLocationChanged   = 'locc'

{ values of SMSAMAdminCode type }
kSMSAMNotifyFwdrSetupChange= 1;
kSMSAMNotifyFwdrNameChange = 2;
kSMSAMNotifyFwdrPwdChange  = 3;
kSMSAMGetDynamicFwdrParams = 4;
```

```
kSMSAMFwdrHomeInternetChangedBit = 0;
kSMSAMFwdrConnectedToChangedBit  = 1;
kSMSAMFwdrForeignRLIsChangedBit  = 2;
kSMSAMFwdrMnMServerChangedBit    = 3;

{ values of SMSAMSlotChanges type }
kSMSAMFwdrEverythingChangedMask     = -1,
kSMSAMFwdrHomeInternetChangedMask   = $00000001;
                                       {1<<kSMSAMFwdrHomeInternetChangedBit}
kSMSAMFwdrConnectedToChangedMask    = $00000002;
                                       {1<<kSMSAMFwdrConnectedToChangedBit}
kSMSAMFwdrForeignRLIsChangedMask    = $00000004;
                                       {1<<kSMSAMFwdrForeignRLIsChangedBit}
kSMSAMFwdrMnMServerChangedMask      = $00000008;
                                       {1<<kSMSAMFwdrMnMServerChangedBit}

kOCESetupLocationNone   = 0;  { disconnect state }
kOCESetupLocationMax    = 8;  { maximum location value }

TYPE

MailMsgRef        = LONGINT;     { reference to new/open letter or message }

MSAMQueueRef      = LONGINT;     { reference to an open MSAM queue }

MSAMSlotID        = INTEGER;     { slot identifier }

MailSlotID        = INTEGER;     { identifies slots within a mailbox }

MailboxRef        = LONGINT;     { reference to an active mailbox }

MailAttributeID   = INTEGER;     { letter attribute identifier }

{ The MailAttributeMask data type defines a set of masks for the
 MailAttributeBitmap data type. However, because the MailAttributeBitmap data
 type is defined as a bit field structure, and the masks operate on variables
 of type LONGINT, you cannot use the masks to set or test the value of a bit
 field in a MailAttributeBitmap structure. The MailAttributeMask data type is
 included for historical reasons only. }
MailAttributeMask = LONGINT;

MailLetterID      = IPMMsgID;

MailNestingLevel  = INTEGER;

MailRecipient     = OCERecipient;
```

```
MailSegmentMask   = INTEGER;

MailSegmentType   = INTEGER;

MailBlockMode     = INTEGER;

PMSAMStatus       = INTEGER;

OCESetupLocation  =  Byte; { current system location }

MailLocationFlags =  Byte; { slot location flags }

MailBuffer = RECORD
   bufferSize: LONGINT; { size of your buffer }
   buffer:     Ptr;     { pointer to your buffer }
   dataSize:   LONGINT; { amount of data returned in or read out
                          of your buffer }
   END;

MailReply = RECORD
   tupleCount: INTEGER;
   { tuple[1..tupleCount] }
   END;

MSAMEnumerateOutQReply = PACKED RECORD
   seqNum:     LONGINT;       { sequence number of message }
   done:       BOOLEAN;       { resolution of message }
   priority:   IPMPriority;   { priority of message }
   msgFamily:  OSType         { message family }
   approxSize: LONGINT;       { size of message }
   tunnelForm: BOOLEAN;       { reserved }
   padByte:    Byte;          { for even byte boundary }
   nextHop:    NetworkSpec;   { reserved }
   msgType:    OCECreatorType;{ message creator and type }
   END;

MSAMEnumerateInQReply = RECORD
   seqNum:     LONGINT;    { letter sequence number }
   msgDeleted: BOOLEAN;    { should letter be deleted? }
   msgUpdated: BOOLEAN;    { was message summary updated? }
   msgCached:  BOOLEAN;    { is letter in the incoming queue? }
   {padByte:   Byte;}
   END;
```

```
MailAttributeBitmap = PACKED RECORD
   reservedA:     0..65535;{ reserved }
   reservedB:     0..1;    { reserved }
   bcc:           0..1;    { blind carbon copy recipients }
   cc:            0..1;    { carbon copy recipients }
   toRecipient:   0..1;    { to recipients }
   from:          0..1;    { sender of letter }
   subject:       0..1;    { subject of letter }
   conversationID:0..1;    { ID of conversation thread }
   replyID:       0..1;    { ID of letter being replied to }
   msgFamily:     0..1;    { message family }
   nestingLevel:  0..1;    { nesting level of letter }
   sendTimeStamp: 0..1;    { time letter was sent }
   letterID:      0..1;    { letter's unique ID number }
   msgType:       0..1;    { letter's creator and type }
   indications:   0..1;    { MailIndications }
   reservedC:     0..1;    { reserved }
   letterFlags:   0..1;    { letter flags }
   END;

MailIndications = PACKED RECORD
   reservedB:              0..65535;{ reserved }
   hasStandardContent:     0..1; { letter has a content block }
   hasImageContent:        0..1; { letter has an image block }
   hasNativeContent:       0..1; { letter has a content enclosure }
   sent:                   0..1; { letter was sent, not just composed }
   authenticated:          0..1; { letter was created and transported with
                                   authentication }
   hasSignature:           0..1; { letter was signed with digital signature }
   hasContent:             0..1; { this letter or nested letter has content }
   isReport:               0..1; { letter is a report }
   isReportWithOriginal:   0..1; { report contains the original letter }
   priority:               0..3; { letter has normal, low, or high priority }
   forwarded:              0..1; { letter contains a forwarded letter }
   receiptReports:         0..1; { originator requests delivery indications }
   nonReceiptReports:      0..1; { originator requests non-delivery
                                   indications }
   originalInReport:       0..3; { originator wants original letter
                                   enclosed in reports }
   END;
```

```
OCERecipient = RECORD
   entitySpecifier:  ^RecordID;
   extensionType:    OSType;
   extensionSize:    INTEGER;
   extensionValue:   Ptr;
   END;

OCEPackedRecipient = RECORD
   dataLength: INTEGER; { length of recipient data }
   data:       PACKED ARRAY[1..kPackedDSSpecMaxBytes] OF Byte;
   END;

MailOriginalRecipient = RECORD
   index: INTEGER;   { index for recipient }
   { Followed by OCEPackedRecipient }
   END;

MailResolvedRecipient = PACKED RECORD
   index:            INTEGER; { index for recipient }
   recipientFlags:   INTEGER; { recipient information }
   responsible:      BOOLEAN; { responsible for delivery? }
   padByte:          Byte;
         { followed by OCEPackedRecipient }
   END;

MailEnclosureInfo = RECORD
   enclosureName: StringPtr;  { name of the enclosure }
   catInfo:       CInfoPBPtr; { HFS catalog info about enclosure }
   comment:       StringPtr;  { comment for Get-Info window }
   icon:          Ptr;        { icon for enclosure file }
   END;

MailLogErrorType = INTEGER;

MailLogErrorCode = INTEGER;

MailErrorLogEntryInfo = RECORD
   version:             INTEGER;          { log entry version }
   timeOccurred:        UTCTime;          { time of error }
   reportingPMSAM:      Str31;            { which MSAM? }
   reportingMSAMSlot:   Str31;            { which slot? }
   errorType:           MailLogErrorType; { level of error }
   errorCode:           MailLogErrorCode; { error code }
```

```
   errorResource:       INTEGER;          { error string resource index }
   actionResource:      INTEGER;          { action string resource index }
   filler:              LONGINT;          { reserved }
   filler2:             INTEGER;          { reserved }
   END;

MailMasterData = RECORD
   attrMask:         MailAttributeBitmap; { indicates attributes present in
                                            MSAMMsgSummary }
   messageID:        MailLetterID;        { ID of this letter }
   replyID:          MailLetterID;        { ID of letter this is a reply to }
   conversationID:   MailLetterID;        { ID of letter that started this
                                            conversation}
   END;

MailCoreData = RECORD
   letterFlags:         MailLetterFlags;     { letter status flags }
   messageSize:         LONGINT;             { size of letter }
   letterIndications:   MailIndications;     { indications for this letter }
   messageType:         OCECreatorType;      { message creator and type of
                                               this letter }
   sendTime:            MailTime;            { time this letter was sent }
   messageFamily:       OSType;              { message family }
   reserved             unsigned char;
   addressedToMe        unsigned char;
   agentInfo:           ARRAY[1..2] OF Byte; { reserved }
   { sender and subject are variable length and even padded }
   sender:              RString32;           { sender of this letter }
   subject:             RString32;           { subject of this letter }
   END;

MSAMMsgSummary = RECORD
   version:    INTEGER;    { version of the MSAMMsgSummary }
   msgDeleted: BOOLEAN;    { true if letter is to be deleted by MSAM }
   msgUpdated: BOOLEAN;    { true if MSAMMsgSummary was updated by IPM
                             Manager }
   msgCached:  BOOLEAN;    { true if letter is in the incoming queue }
   {padByte:   Byte;}
   masterData: MailMasterData;   { attributes not essential to display }
   coreData:   MailCoreData;     { attributes critical to display }
   { followed by the personal MSAM's private data }
   END;
```

```
MailLocationInfo = RECORD
   location:   OCESetupLocation;    { the current location }
   active:     MailLocationFlags;   { slot's location flags }
   END;

MailEPPCMsg = RECORD
   version:    INTEGER; { message version }
   CASE INTEGER OF
      1. (smca:            ^SMCA);           { pointer to SMCA }
      2. (sequenceNumber:  LONGINT);         { letter sequence number }
      3. (locationInfo:    MailLocationInfo);{ location information }
   END;

SMCA = RECORD
   smcaLength: INTEGER;          { length of entire SMCA, including size of
                                   smcaLength field }
   result:     OSErr;            { result code }
   userBytes:  LONGINT;          { command interpreted user data }
   CASE INTEGER OF
      1: (slotCID: CreationID);  { creation ID of record
                                   containing slot information }
      2: (msgHint: LONGINT);     { message reference value }
   END;

SMSAMAdminCode = INTEGER;

SMSAMAdminEPPCRequest = RECORD
   adminCode: SMSAMAdminCode;                    { admin code }
   CASE INTEGER OF
      1: (setupChange:     SMSAMSetupChange);      { setup change }
      2: (nameChange:      SMSAMNameChange);       { reserved }
      3: (passwordChange:  SMSAMPasswordChange);   { reserved }
      4: (dynamicParams:   SMSAMDynamicParams);    { reserved }
   END;

SMSAMSlotChanges = LONGINT;

SMSAMSetupChange = RECORD
   whatChanged:   SMSAMSlotChanges; { bitmap of changed parameters }
   serverHint:    AddrBlock;        { AOCE server address }
   END;
```

```
SMSAMNameChange = RECORD            { reserved data type }
   newName:    RString;             { server MSAM's new name }
   serverHint: AddrBlock;           { AOCE server address }
   END;

SMSAMPasswordChange = RECORD     { reserved data type }
   newPassword:   RString;       { server MSAM's new password }
   serverHint:    AddrBlock;     { AOCE server address }
   END;

SMSAMDynamicParams = RECORD      { reserved data type }
   curDiskUsed:   LONGINT;       { disk space used }
   curMemoryUsed: LONGINT;       { memory used }
   END;

MailTime = RECORD
   time:    UTCTime;             { current UTC(GMT) }
   offset:  UTCOffset;           { offset from UTC }
   END;

MailTimer = RECORD
   CASE INTEGER OF
      1: (frequency: LONGINT);   { how often to connect }
      2: (connectTime: LONGINT); { time since midnight }
   END;

MailTimerKind = Byte;

MailTimers = PACKED RECORD
   sendTimeKind:     MailTimerKind;    { timer kind for sending }
   receiveTimeKind:  MailTimerKind;    { timer kind for receiving }
   send:             MailTimer;        { connect time or frequency
                                         for sending letters }
   receive:          MailTimer;        { connect time or frequency
                                         for sending letters }
   END;

MailStandardSlotInfoAttribute = PACKED RECORD
   version:          INTEGER;             { MSAM version of the slot }
   active:           MailLocationFlags;   { active at location i if
                                            MailLocation Mask(i) is set }
   padByte:          Byte;
   sendReceiveTimer: MailTimers;
   END;
```

```
MailLetterSystemFlags= INTEGER;

MailLetterUserFlags  = INTEGER;

MailLetterFlags = RECORD
   sysFlags:   MailLetterSystemFlags;  { system flags }
   userFlags:  MailLetterUserFlags;    { user flags }
   END;

MailMaskedLetterFlags = RECORD
   flagMask:   MailLetterFlags;  { flags that are to be set }
   flagValues: MailLetterFlags;  { their values }
   END;

MailBlockInfo = RECORD
   blockType:     OCECreatorType;
   offset:        LONGINT;
   blockLength:   LONGINT;
   END;

MailParamBlockHeader = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   END;

PMSAMGetMSAMRecordPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   msamCID:       CreationID;
   END;

PMSAMOpenQueuesPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
```

```
    ioCompletion:  ProcPtr;    { pointer to completion routine }
    ioResult:      OSErr;      { result code }
    saveA5:        LONGINT;    { pointer to global variables }
    reqCode:       INTEGER;    { reserved }
    inQueueRef:    MSAMQueueRef;
    outQueueRef:   MSAMQueueRef;
    msamSlotID:    MSAMSlotID;
    filler:        ARRAY[1..2] OF LONGINT;
    END;

PMSAMSetStatusPB = RECORD
    qLink:         Ptr;        { next queue entry }
    reservedH1:    LONGINT;    { reserved }
    reservedH2:    LONGINT;    { reserved }
    ioCompletion:  ProcPtr;    { pointer to completion routine }
    ioResult:      OSErr;      { result code }
    saveA5:        LONGINT;    { pointer to global variables }
    reqCode:       INTEGER;    { reserved }
    queueRef:      MSAMQueueRef;
    seqNum:        LONGINT;
    msgHint:       LONGINT;
    status:        PMSAMStatus;
    END;

PMSAMLogErrorPB = RECORD
    qLink:         Ptr;        { next queue entry }
    reservedH1:    LONGINT;    { reserved }
    reservedH2:    LONGINT;    { reserved }
    ioCompletion:  ProcPtr;    { pointer to completion routine }
    ioResult:      OSErr;      { result code }
    saveA5:        LONGINT;    { pointer to global variables }
    reqCode:       INTEGER;    { reserved }
    msamSlotID:    MSAMSlotID;
    logEntry:      ^MailErrorLogEntryInfo;
    filler:        ARRAY[1..2] OF LONGINT;
    END;

PMSAMCreateMsgSummaryPB = RECORD
    qLink:         Ptr;        { next queue entry }
    reservedH1:    LONGINT;    { reserved }
    reservedH2:    LONGINT;    { reserved }
    ioCompletion:  ProcPtr;    { pointer to completion routine }
    ioResult:      OSErr;      { result code }
    saveA5:        LONGINT;    { pointer to global variables }
```

```
   reqCode:       INTEGER;    { reserved }
   inQueueRef:    MSAMQueueRef;
   seqNum:        LONGINT;    { seq of the new letter }
   msgSummary:    ^MSAMMsgSummary;
   buffer:        ^MailBuffer;{ PMSAM specific data }
   END;

PMSAMPutMsgSummaryPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   inQueueRef:    MSAMQueueRef;
   seqNum:        LONGINT;
   letterFlags:   ^MailMaskedLetterFlags;
   buffer:        ^MailBuffer;   { PMSAM private data }
   END;

PMSAMGetMsgSummaryPB = RECORD
   qLink:            Ptr;        { next queue entry }
   reservedH1:       LONGINT;    { reserved }
   reservedH2:       LONGINT;    { reserved }
   ioCompletion:     ProcPtr;    { pointer to completion routine }
   ioResult:         OSErr;      { result code }
   saveA5:           LONGINT;    { pointer to global variables }
   reqCode:          INTEGER;    { reserved }
   inQueueRef:       MSAMQueueRef;
   seqNum:           LONGINT;
   msgSummary:       ^MSAMMsgSummary;
   buffer:           ^MailBuffer;{ PMSAM private data }
   msgSummaryOffset: INTEGER;    { offset of PMSAM private data  }
                                 { from start of MsgSummary }
   END;

SMSAMSetupPB = RECORD
   qLink:            Ptr;        { next queue entry }
   reservedH1:       LONGINT;    { reserved }
   reservedH2:       LONGINT;    { reserved }
   ioCompletion:     ProcPtr;    { pointer to completion routine }
   ioResult:         OSErr;      { result code }
   saveA5:           LONGINT;    { pointer to global variables }
```

```
   reqCode:          INTEGER;    { reserved }
   serverMSAM        RecordIDPtr;
   password       RStringPtr;
   gatewayType    OSType;
   gatewayTypeDescription  RStringPtr;
   catalogServerHint       AddrBlock;
   END;

SMSAMStartupPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   msamIdentity:  AuthIdentity;
   queueRef:      MSAMQueueRef;
   END;

SMSAMShutdownPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   queueRef:      MSAMQueueRef;
   END;

MSAMEnumeratePB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   queueRef:      MSAMQueueRef;
   startSeqNum:   LONGINT;
   nextSeqNum:    LONGINT;
   buffer:        MailBuffer;
   END;
```

```
MSAMDeletePB = PACKED RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   queueRef:      MSAMQueueRef;
   seqNum:        LONGINT;
   msgOnly:       BOOLEAN;    { only valid for PMSAM & inQueue }
   padByte:       Byte;
   result:        OSErr;      { reserved }
   END;

MSAMOpenPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   queueRef:      MSAMQueueRef;
   seqNum:        LONGINT;
   mailMsgRef:    MailMsgRef;
   END;

MSAMOpenNestedPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   nestedRef:     MailMsgRef;
   END;

MSAMClosePB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
```

```
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   END;

MSAMGetMsgHeaderPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   selector:      IPMHeaderSelector;
   offset:        LONGINT;
   buffer:        MailBuffer;
   remaining:     LONGINT;
   END;

MSAMGetAttributesPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   requestMask:   MailAttributeBitmap;
   buffer:        MailBuffer;
   responseMask:  MailAttributeBitmap;
   more:          BOOLEAN;
   END;

MSAMGetRecipientsPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
```

```
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   attrID:        MailAttributeID;
   startIndex:    INTEGER;
   buffer:        MailBuffer;
   nextIndex:     INTEGER;
   more:          BOOLEAN;
   END;

MSAMGetContentPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   segmentMask:   MailSegmentMask;
   buffer:        MailBuffer;
   textScrap:     ^StScrpRec;
   script:        ScriptCode;
   segmentType:   MailSegmentType;
   endOfScript:   BOOLEAN;
   endOfSegment:  BOOLEAN;
   endOfContent:  BOOLEAN;
   segmentLength: LONGINT;
   segmentID:     LONGINT;
   END;

MSAMGetEnclosurePB = PACKED RECORD
   qLink:            Ptr;        { next queue entry }
   reservedH1:       LONGINT;    { reserved }
   reservedH2:       LONGINT;    { reserved }
   ioCompletion:     ProcPtr;    { pointer to completion routine }
   ioResult:         OSErr;      { result code }
   saveA5:           LONGINT;    { pointer to global variables }
   reqCode:          INTEGER;    { reserved }
   mailMsgRef:       MailMsgRef;
   contentEnclosure: BOOLEAN;
   padByte:          Byte;
   buffer:           MailBuffer;
   endOfFile:        BOOLEAN;
   endOfEnclosures:  BOOLEAN;
   END;
```

```
MSAMEnumerateBlocksPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   startIndex:    INTEGER;    { starts at 1 }
   buffer:        MailBuffer;
   nextIndex:     INTEGER;
   more:          BOOLEAN;
   END;

MSAMGetBlockPB = PACKED RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   blockType:     OCECreatorType;
   blockIndex:    INTEGER;
   buffer:        MailBuffer;
   dataOffset:    LONGINT;
   endOfBlock:    BOOLEAN;
   padByte:       Byte;
   remaining:     LONGINT;
   END;

MSAMMarkRecipientsPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   queueRef:      MSAMQueueRef;
   seqNum:        LONGINT;
   buffer:        MailBuffer;
   END;
```

```
MSAMnMarkRecipientsPB = RECORD
   qLink:          Ptr;        { next queue entry }
   reservedH1:     LONGINT;    { reserved }
   reservedH2:     LONGINT;    { reserved }
   ioCompletion:   ProcPtr;    { pointer to completion routine }
   ioResult:       OSErr;      { result code }
   saveA5:         LONGINT;    { pointer to global variables }
   reqCode:        INTEGER;    { reserved }
   mailMsgRef:     MailMsgRef;
   buffer:         MailBuffer;
   END;

MSAMCreatePB = RECORD
   qLink:          Ptr;        { next queue entry }
   reservedH1:     LONGINT;    { reserved }
   reservedH2:     LONGINT;    { reserved }
   ioCompletion:   ProcPtr;    { pointer to completion routine }
   ioResult:       OSErr;      { result code }
   saveA5:         LONGINT;    { pointer to global variables }
   reqCode:        INTEGER;    { reserved }
   queueRef:       MSAMQueueRef;
   asLetter:       BOOLEAN;    { create as letter or message? }
   msgType:        IPMMsgType;
   refCon:         LONGINT;    { for non-letter messages only }
   seqNum:         LONGINT;
   tunnelForm:     BOOLEAN;    { always false }
   bccRecipients:  BOOLEAN;    { true if creating letter with bcc recipients }
   newRef:         MailMsgRef;
   END;

MSAMBeginNestedPB = RECORD
   qLink:          Ptr;        { next queue entry }
   reservedH1:     LONGINT;    { reserved }
   reservedH2:     LONGINT;    { reserved }
   ioCompletion:   ProcPtr;    { pointer to completion routine }
   ioResult:       OSErr;      { result code }
   saveA5:         LONGINT;    { pointer to global variables }
   reqCode:        INTEGER;    { reserved }
   mailMsgRef:     MailMsgRef;
   refCon:         LONGINT;    { for messages only }
   msgType:        IPMMsgType;
   END;
```

```
MSAMEndNestedPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   END;

MSAMSubmitPB = PACKED RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef; { message reference number }
   submitFlag:    BOOLEAN;    { submit or delete message? }
   padByte:       Byte;
   msgID:         MailLetterID;  { reserved }
   END;

MSAMPutMsgHeaderPB = PACKED RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   replyQueue:    ^OCERecipient;
   sender:        ^IPMSender;
   deliveryNotification: IPMNotificationType;
   priority:      IPMPriority;
   END;
```

```
MSAMPutAttributePB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   attrID:        MailAttributeID;
   buffer:        MailBuffer;
   END;

MSAMPutRecipientPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   attrID:        MailAttributeID;
   recipient:     ^MailRecipient;
   responsible:   BOOLEAN;    { for server and message msams only }
   END;

MSAMPutContentPB = PACKED RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailMsgRef:    MailMsgRef;
   segmentType:   MailSegmentType;
   append:        BOOLEAN;
   padByte:       Byte;
   buffer:        MailBuffer;
   textScrap:     ^StScrpRec;
   startNewScript:BOOLEAN;
   script:        ScriptCode;
   END;
```

```
MSAMPutEnclosurePB = RECORD
   qLink:          Ptr;        { next queue entry }
   reservedH1:     LONGINT;    { reserved }
   reservedH2:     LONGINT;    { reserved }
   ioCompletion:   ProcPtr;    { pointer to completion routine }
   ioResult:       OSErr;      { result code }
   saveA5:         LONGINT;    { pointer to global variables }
   reqCode:        INTEGER;    { reserved }
   mailMsgRef:     MailMsgRef;
   contentEnclosure: BOOLEAN;
   padByte:        BOOLEAN;
   hfs:            BOOLEAN;    { true = in file system, false = in memory }
   append:         BOOLEAN;
   buffer:         MailBuffer;
   enclosure:      FSSpec;
   addlInfo:       MailEnclosureInfo;
   END;

MSAMPutBlockPB = RECORD
   qLink:          Ptr;        { next queue entry }
   reservedH1:     LONGINT;    { reserved }
   reservedH2:     LONGINT;    { reserved }
   ioCompletion:   ProcPtr;    { pointer to completion routine }
   ioResult:       OSErr;      { result code }
   saveA5:         LONGINT;    { pointer to global variables }
   reqCode:        INTEGER;    { reserved }
   mailMsgRef:     MailMsgRef;
   refCon:         LONGINT;    { for messages only }
   blockType:      OCECreatorType;
   append:         BOOLEAN;
   buffer:         MailBuffer;
   mode:           MailBlockMode;
   offset:         LONGINT;
   END;

MSAMCreateReportPB = RECORD
   qLink:          Ptr;           { next queue entry }
   reservedH1:     LONGINT;       { reserved }
   reservedH2:     LONGINT;       { reserved }
   ioCompletion:   ProcPtr;       { pointer to completion routine }
   ioResult:       OSErr;         { result code }
   saveA5:         LONGINT;       { pointer to global variables }
   reqCode:        INTEGER;       { reserved }
   queueRef:       MSAMQueueRef;  { to distinguish personal and server MSAMs }
```

```
   mailMsgRef:    MailMsgRef;
   msgID:         MailLetterID;  { of letter being reported upon }
   sender:        ^MailRecipient;{ sender of the letter you're reporting on }
   END;

MSAMPutRecipientReportPB = RECORD
   qLink:            Ptr;        { next queue entry }
   reservedH1:       LONGINT;    { reserved }
   reservedH2:       LONGINT;    { reserved }
   ioCompletion:     ProcPtr;    { pointer to completion routine }
   ioResult:         OSErr;      { result code }
   saveA5:           LONGINT;    { pointer to global variables }
   reqCode:          INTEGER;    { reserved }
   mailMsgRef:       MailMsgRef;
   recipientIndex:   INTEGER;    { recipient index in the original letter }
   result:           OSErr;      { result of sending the recipient }
   END;

MailWakeupPMSAMPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   pmsamCID:      CreationID;
   mailSlotID:    MailSlotID;
   END;

MailCreateMailSlotPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailboxRef:    MailboxRef;
   timeout:       LONGINT;
   pmsamCID:      CreationID;
   smca:          SMCA;
   END;
```

```
MailModifyMailSlotPB = RECORD
   qLink:         Ptr;        { next queue entry }
   reservedH1:    LONGINT;    { reserved }
   reservedH2:    LONGINT;    { reserved }
   ioCompletion:  ProcPtr;    { pointer to completion routine }
   ioResult:      OSErr;      { result code }
   saveA5:        LONGINT;    { pointer to global variables }
   reqCode:       INTEGER;    { reserved }
   mailboxRef:    MailboxRef;
   timeout:       LONGINT;
   pmsamCID:      CreationID;
   smca:          SMCA;
   END;

MSAMParam = RECORD
   CASE INTEGER OF
      1: (header: MailParamBlockHeader);
      2: (pmsamGetMSAMRecord: PMSAMGetMSAMRecordPB);
      3: (pmsamOpenQueues: PMSAMOpenQueuesPB);
      4: (pmsamSetStatus: PMSAMSetStatusPB);
      5: (pmsamLogError: PMSAMLogErrorPB);
      6: (smsamSetup: SMSAMSetupPB);
      7: (smsamStartup: SMSAMStartupPB);
      8: (smsamShutdown: SMSAMShutdownPB);
      9: (msamEnumerate: MSAMEnumeratePB);
      10: (msamDelete: MSAMDeletePB);
      11: (msamOpen: MSAMOpenPB);
      12: (msamOpenNested: MSAMOpenNestedPB);
      13: (msamClose: MSAMClosePB);
      14: (msamGetMsgHeader: MSAMGetMsgHeaderPB);
      15: (msamGetAttributes: MSAMGetAttributesPB);
      16: (msamGetRecipients: MSAMGetRecipientsPB);
      17: (msamGetContent: MSAMGetContentPB);
      18: (msamGetEnclosure: MSAMGetEnclosurePB);
      19: (msamEnumerateBlocks: MSAMEnumerateBlocksPB);
      20: (msamGetBlock: MSAMGetBlockPB);
      21: (msamMarkRecipients: MSAMMarkRecipientsPB);
      22: (msamnMarkRecipients: MSAMnMarkRecipientsPB);
      23: (msamCreate: MSAMCreatePB);
      24: (msamBeginNested: MSAMBeginNestedPB);
      25: (msamEndNested: MSAMEndNestedPB);
      26: (msamSubmit: MSAMSubmitPB);
      27: (msamPutMsgHeader: MSAMPutMsgHeaderPB);
      28: (msamPutAttribute: MSAMPutAttributePB);
```

```
    29: (msamPutRecipient: MSAMPutRecipientPB);
    30: (msamPutContent: MSAMPutContentPB);
    31: (msamPutEnclosure: MSAMPutEnclosurePB);
    32: (msamPutBlock: MSAMPutBlockPB);
    33: (msamCreateReport: MSAMCreateReportPB);
    34: (msamPutRecipientReport: MSAMPutRecipientReportPB);
    35: (pmsamCreateMsgSummary: PMSAMCreateMsgSummaryPB);
    36: (pmsamPutMsgSummary: PMSAMPutMsgSummaryPB);
    37: (pmsamGetMsgSummary: PMSAMGetMsgSummaryPB);
    38: (wakeupPMSAM: MailWakeupPMSAMPB);
    39: (createMailSlot: MailCreateMailSlotPB);
    40: (modifyMailSlot: MailModifyMailSlotPB);
END;
```

MSAM Functions

Initializing an MSAM

```
FUNCTION PMSAMGetMSAMRecord(VAR paramBlock: MSAMParam): OSErr;
FUNCTION PMSAMOpenQueues(VAR paramBlock: MSAMParam): OSErr;
FUNCTION SMSAMSetup(VAR paramBlock: MSAMParam): OSErr;
FUNCTION SMSAMStartup(VAR paramBlock: MSAMParam): OSErr;
```

Enumerating Messages in a Queue

```
FUNCTION MSAMEnumerate(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN): OSErr;
```

Opening an Outgoing Message

```
FUNCTION MSAMOpen(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN): OSErr;
```

Reading Header Information

```
FUNCTION MSAMGetAttributes(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                           OSErr;
FUNCTION MSAMGetRecipients(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                           OSErr;
FUNCTION MSAMGetMsgHeader(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                           OSErr;
```

Reading a Message

```
FUNCTION MSAMGetContent(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
FUNCTION MSAMGetEnclosure(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
FUNCTION MSAMEnumerateBlocks(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
FUNCTION MSAMGetBlock(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN): OSErr;
FUNCTION MSAMOpenNested(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
```

Marking a Recipient

```
FUNCTION MSAMnMarkRecipients(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
FUNCTION MSAMMarkRecipients(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
```

Closing a Message

```
FUNCTION MSAMClose(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN): OSErr;
```

Creating, Reading, and Writing Message Summaries

```
FUNCTION PMSAMCreateMsgSummary(VAR paramBlock: MSAMParam; asyncFlag:
                              BOOLEAN): OSErr;
FUNCTION PMSAMGetMsgSummary(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
FUNCTION PMSAMPutMsgSummary(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
```

Creating a Message

```
FUNCTION MSAMCreate(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN): OSErr;
```

Writing Header Information

```
FUNCTION MSAMPutAttribute(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
FUNCTION MSAMPutRecipient(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
FUNCTION MSAMPutMsgHeader(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                              OSErr;
```

Writing a Message

```
FUNCTION MSAMPutContent(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                            OSErr;
FUNCTION MSAMPutEnclosure(VAR paramBlock: MSAMParam): OSErr;
FUNCTION MSAMPutBlock(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN): OSErr;
FUNCTION MSAMBeginNested(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                            OSErr;
FUNCTION MSAMEndNested(VAR paramBlock: MSAMParam): OSErr;
```

Submitting a Message

```
FUNCTION MSAMSubmit(VAR paramBlock: MSAMParam): OSErr;
```

Deleting a Message

```
FUNCTION MSAMDelete(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN): OSErr;
```

Generating Log Entries and Reports

```
FUNCTION PMSAMLogError(VAR paramBlock: MSAMParam): OSErr;
FUNCTION MSAMCreateReport(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                            OSErr;
FUNCTION MSAMPutRecipientReport(VAR paramBlock: MSAMParam; asyncFlag:
                            BOOLEAN): OSErr;
```

Shutting Down a Server MSAM

```
FUNCTION SMSAMShutdown(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN): OSErr;
```

Setting Message Status

```
FUNCTION PMSAMSetStatus(VAR paramBlock: MSAMParam; asyncFlag: BOOLEAN):
                            OSErr;
```

Personal MSAM AOCE Template Functions

```
FUNCTION MailCreateMailSlot(VAR paramBlock: MSAMParam): OSErr;
FUNCTION MailModifyMailSlot(VAR paramBlock: MSAMParam): OSErr;
FUNCTION MailWakeupPMSAM(VAR paramBlock: MSAMParam): OSErr;
```

Application-Defined Routine

```
PROCEDURE MyCompletionRoutine(VAR paramBlock: MSAMParam);
```

Assembly-Language Summary

Trap Macros

Trap Macros Requiring Routine Selectors

`_oceTBDispatch`

Selector	Routine
$0500	`PMSAMOpenQueues`
$0501	`SMSAMStartup`
$0502	`SMSAMShutdown`
$0503	`MSAMEnumerate`
$0504	`MSAMDelete`
$0505	`MSAMMarkRecipients`
$0506	`PMSAMGetMSAMRecord`
$0507	`MailWakeupPMSAM`
$0508	`MSAMOpen`
$0509	`MSAMOpenNested`
$050A	`MSAMClose`
$050B	`MSAMGetAttributes`
$050C	`MSAMGetRecipients`
$050D	`MSAMGetContent`
$050E	`MSAMGetEnclosure`
$050F	`MSAMEnumerateBlocks`
$0510	`MSAMGetBlock`
$0511	`MSAMGetMsgHeader`
$0512	`MSAMnMarkRecipients`
$0514	`MSAMCreate`
$0515	`MSAMBeginNested`
$0516	`MSAMEndNested`
$0517	`MSAMSubmit`
$0518	`MSAMPutAttribute`
$0519	`MSAMPutRecipient`
$051A	`MSAMPutContent`
$051B	`MSAMPutEnclosure`
$051C	`MSAMPutBlock`
$051D	`MSAMPutMsgHeader`
$051F	`MSAMCreateReport`

Selector	Routine
$0520	MSAMPutRecipientReport
$0521	PMSAMLogError
$0522	PMSAMCreateMsgSummary
$0523	SMSAMSetup
$0525	PMSAMPutMsgSummary
$0526	PMSAMGetMsgSummary
$0527	PMSAMSetStatus
$052B	MailCreateMailSlot
$052C	MailModifyMailSlot

Result Codes

noErr	0	No error
corErr	–3	PowerShare mail server not running
dskFulErr	–34	All allocation blocks on the volume are full
kOCEParamErr	–50	Invalid parameter
memFullErr	–108	Not enough memory
noRelErr	–1101	Timer expired before MSAM responded
kOCEToolboxNotOpen	–1500	Collaboration toolbox is shutting down
kOCEInvalidRef	–1502	Invalid message reference number
kOCEBufferTooSmall	–1503	Buffer is too small
kOCEVersionErr	–1504	Wrong version of nested message
kOCEInternalErr	–1506	Serious internal error
kOCEAlreadyExists	–1510	Duplicate recipient type
kIPMMsgTypeReserved	–1511	Message creator and/or type specified not allowed
kOCEInvalidRecipient	–1514	Bad recipient
kOCERefIsClosing	–1516	IPM Manager is shutting down the personal MSAM, or server MSAM's mail server is shutting down
kOCEWriteAccessDenied	–1541	Identity lacks write access privileges
kOCETargetDirectoryInaccessible	–1613	Target catalog is not currently available
kOCENoSuchDNode	–1615	Can't find specified dNode
kOCENoDupAllowed	–1641	Duplicate record name and type
kMailInvalidOrder	–15040	Content already closed
kMailInvalidSeqNum	–15041	Invalid message sequence number
kMailHdrAttrMissing	–15043	Required attribute not added to message
kMailBadEnclLengthErr	–15044	Invalid data length
kMailInvalidRequest	–15045	Reference number invalid with this request
kMailInvalidPostItVersion	–15046	Message summary is wrong version
kMailNotASlotInQ	–15047	Invalid value for a slot's incoming queue
kMailIgnoredErr	–15053	MSAM ignored high-level event
kMailLengthErr	–15054	Error occurred in sending the event
kMailTooManyErr	–15055	IPM Manager or MSAM too busy to handle event
kMailNoMSAMErr	–15056	No such MSAM
kMailSlotSuspended	–15058	Slot is suspended
kMailMSAMSuspended	–15059	MSAM is suspended
kMailBadSlotInfo	–15060	Invalid slot information
kMailMalformedContent	–15061	Content data malformed
kMailNoSuchSlot	–15062	No such slot
kMailBadMSAM	–15066	MSAM unusable for unspecified reason
kMailBadState	–15068	Invalid status setting
kIPMInvalidMsgType	–15091	Only `kIPMOSFormatType` allowed when creating a letter
kIPMBlkNotFound	–15107	No such block