ipd4200maimgrmTES-10

Defense Information Infrastructure (DII)
Common Operating Environment (COE)

Application Program Interface Reference Manual (APIRM)
for the
METOC Imagery API (MAIMG) Segment
of the
Tactical Environmental Support System Next Century
[TESS(NC)]
Meteorology and Oceanography (METOC) Database

Preliminary Release

Document Version 4.2

9 October 1998

Prepared for:
Naval Research Laboratory
Marine Meteorology Division
Monterey, CA

Prepared by:
Integrated Performance Decisions
Middletown, RI

Table of Contents

1 SCOPE 1
1.1 Identification 1
1.2 System Overview 1

2 REFERENCED DOCUMENTS 5
2.1 Government Documents 5
2.2 Non-Government Documents 6

3 API OVERVIEW 7
3.1 Imagery Data Structures/Defines 8
3.1.1 Imagery Defines/Enumerations 8
3.1.2 Imagery Area of Interest (AOI) Structure 10
3.1.3 Imagery Data Structure 10
3.1.4 Satellite Imagery Data Structures 11
3.1.5 Product Imagery Data Structures 12
3.1.6 Imagery Object Reference Structure 12
3.1.7 Imagery Linked List Structure 12
3.1.8 Imagery Query Structure 12
3.1.9 Imagery Catalog Structure 14
3.1.10 Imagery TypesData Structure 15
3.1.11 Imagery Return Structure 15
3.2 MAIMG Error Definitions 16

4 IMAGERY API REFERENCE 17
4.1 MAIMGConnect 18
4.2 MAIMGDisconnect 19
4.3 MAIMGRemoteConnect 20
4.4 MAIMGRemoteDisconnect 21
4.5 MAIMGSetConnection 22
4.6 MAIMGIngest 23
4.7 MAIMGCatalog 24
4.8 MAIMGGetByID 26
4.9 MAIMGGetByQuery 27
4.10 MAIMGUpdateByID 29
4.11 MAIMGDeleteByID 30
4.12 MAIMGGetTypesInfo 31
4.13 MAIMGFreeLL 32

5 NOTES 33
5.1 Glossary of Acronyms 33

List of Figures

1-1 TESS(NC) METOC Database Conceptual Organization 3

1 SCOPE

1.1 Identification

This Application Program Interface (API) Reference Manual (APIRM) describes the APIs provided in the Imagery API (MAIMG) segment, Version 4.2.0.0, of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database. The MAIMG segment provides APIs for the storage and retrieval of Imagery Products. This software is designed to run under the Defense Information Infrastructure (DII) Common Operating Environment (COE), release 3.1, on a Hewlett-Packard computer running HP-UX 10.20 or a personal computer running the Microsoft Windows NT 4.0 operating system with Service Pack 3.

1.2 System Overview

The software described in this document forms a portion of the METOC Database component of the TESS(NC) Program (Navy Integrated Tactical Environmental Subsystem (NITES) Version I). On 29 October 1996, the Oceanographer of the Navy issued a TESS Program Policy statement in letter 3140 Serial 961/6U570953, modifying the Program by calling for five seamless software versions that are DII COE compliant, preferably to level 5.

The five versions are:

- NITES Version I — The local data fusion center and principal METOC analysis and forecast system (TESS(NC))
- NITES Version II — The subsystem on the Joint Maritime Command Information System (JMCIS) or Global Command and Control System (GCCS) (NITES/Joint METOC Segment (JMS))
- NITES Version III — The unclassified aviation forecast, briefing, and display subsystem tailored to Naval METOC shore activities (currently satisfied by the Meteorological Integrated Data Display System (MIDDS))
- NITES Version IV — The Portable subsystem composed of independent PCs/workstations and modules for forecaster, satellite, communications, and Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance (IC4ISR) functions (currently the Interim Mobile Oceanographic Support System (IMOSS))
- NITES Version V — Foreign Military Sales (currently satisfied by the Allied Environmental Support System (AESS))

NITES I acquires and assimilates various METOC data for use by US Navy and Marine Corps weather forecasters and tactical planners. NITES I provides these users with METOC data,

products, and applications necessary to support the warfighter in tactical operations and decision making. NITES I provides METOC data and products to NITES I and II applications, as well as non-TESS(NC) systems requiring METOC data, in a heterogeneous, networked computing environment.

The TESS(NC) Concept of Operations and system architecture require that the METOC Database be distributed both in terms of application access to METOC data and products and in terms of physical location of the data repositories. The organizational structure of the database is influenced by these requirements, and the components of this distributed database are described below.

In accordance with DII COE database concepts, the METOC Database is composed of six DII COE-compliant *shared database* segments. Associated with each shared database segment is an API segment. The segments are arranged by data type as follows:

Data Type	**Data Segment**	**API Segment**
Grid Fields	MDGRID	MAGRID
Latitude-Longitude-Time (LLT) Observations	MDLLT	MALLT
Textual Observations and Bulletins	MDTXT	MATXT
Remotely Sensed Data	MDREM	MAREM
Imagery	MDIMG	MAIMG
Climatology Data	MDCLIM	MACLIM

A typical client-server installation is depicted in Figure 1-1 on the next page. This shows the shared database segments residing on a DII COE database server, with a NITES I or II client machine hosting the API segments. Communication between API segments and shared database segments is accomplished over the network using ANSI-standard Structured Query Language (SQL).

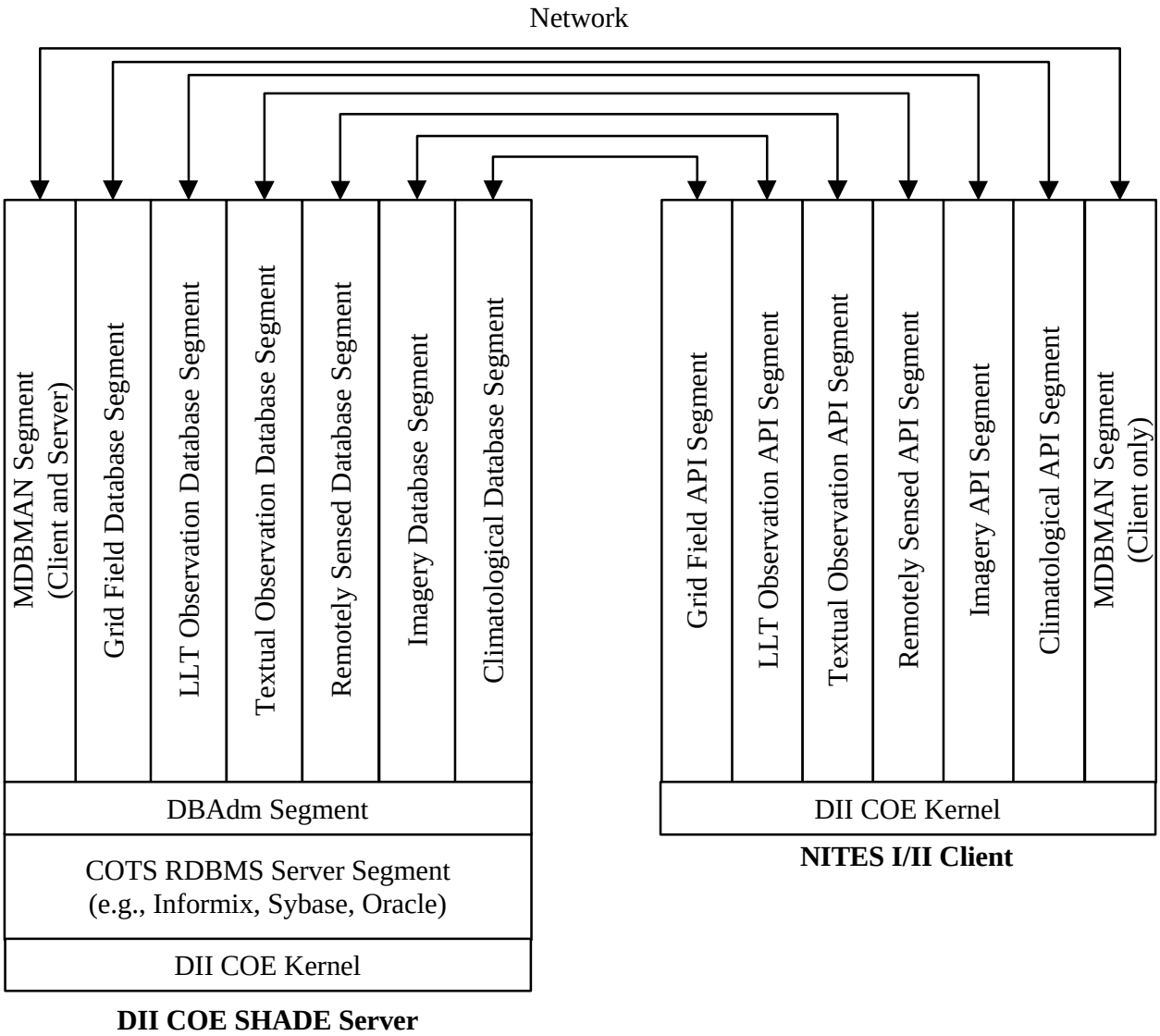

Figure 1-1. TESS(NC) METOC Database Conceptual Organization

The APIs in the MAIMG segment deal with imagery products. Imagery products can be associated with a specific geographic point/area, as well as time. A number of different image formats are supported; they are NITF, MIF, GIF, TIFF, BMP, JPEG, XWD, XBM, PBM, and MPEG.

(This page intentionally left blank.)

2 REFERENCED DOCUMENTS

2.1 Government Documents

STANDARDS

MIL-STD-498 5 December 1994	*Software Development and Documentation*

SPECIFICATIONS

Unnumbered 27 May 1997	*Performance Specification (PS) for the Tactical Environmental Support System/Next Century TESS(3)/NC (AN/UMK-3)*
Unnumbered 30 September 1997	*Software Requirements Specification for the Tactical Environmental Support System/Next Century [TESS(3)/NC] Meteorological and Oceanographic (METOC) Database*, Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC

OTHER DOCUMENTS

Unnumbered 30 September 1997	*Database Design Description for the Tactical Environmental Support System/Next Century [TESS(3)/NC)] Meteorological and Oceanographic (METOC) Database*, Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC
DII.COE.DocReqs-5 29 April 1997	*Defense Information Infrastructure (DII) Common Operating Environment (COE) Developer Documentation Requirements, Version 1.0*
ipd4200maimgpmTES-10 9 October 1998	*Programming Manual (PM) for the METOC Imagery API (MAIMG) Segment of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database*

DII.COE31.HP10.20.CIP 23 May 1997	*DII COE V3.1 HP 10.20 Consolidated Installation Procedures*
DII.3010.HP1020.KernelP1.IG-1 9 May 1997	*DII COE Kernel 3.0.1.0P1 Patch 1 for HP-UX 10.20 Installation Guide*
DII.3010.HP1020.KernelP2.IG-1 30 July 1997	*DII COE Kernel 3.0.1.0P2 Patch 2 for HP-UX 10.20 Installation Guide*
DII.3010.HP1020.KernelP3.IG-1 08 August 1997	*DII COE Kernel 3.0.1.0P3 Patch 3 for HP-UX 10.20 Installation Guide*
DII.3010.HP1020.KernelP4.IG-1 27 August 1997	*DII COE Kernel 3.0.1.0P4 Patch 4 for HP-UX 10.20 Installation Guide*

2.2 Non-Government Documents

None.

3 API OVERVIEW

The Imagery API segment provides interfaces to:

- Establish connection to the TESS(NC) MDIMG Database (MAIMGConnect, MAIMGRemoteConnect)
- Set the current connection (MAIMGSetConnection)
- Ingest an Imagery into the database (MAIMGIngest)
- Retrieve a catalog listing of Imagery meeting specified criteria from the database (MAIMGCatalog)
- Retrieve selected Imagery data from the database (MAIMGGetByQuery)
- Retrieve a single Image from the database (MAIMGGetByID)
- Update a single Image record in the database (MAIMGUpdateByID)
- Delete a selected Image from the database (MAIMGDeleteByID)
- Retrieve a listing of Type and SubType information (MAIMGGetTypesInfo)
- Free the linked lists returned by MAIMGCatalog and MAIMGGetByQuery (MAIMGFreeLL)
- Disconnect from the database (MAIMGDisconnect, MAIMGRemoteDisconnect).

This document provides developers with information needed to access Imagery data from the METOC Database. Section 4 provides the API reference in manual page format. Section 3.1 provides a list of the structures common to the Imagery APIs.

3.1 Imagery Data Structures/Defines

The following structures are used by the Imagery APIs and are provided for reference:

3.1.1 Imagery Defines/Enumerations

```
#define MAIMG_STRING_LEN        32
#define MAIMG_DESCRIP_LEN       97

#define MAIMG_QUERY_WILDCARD  -9999  /* Denotes wildcard value for Imagery  */

#define MAIMG_ALL_SATELLITE   -9998  /* Used in Query Structure nType
                                        field, indicates Query is for all
                                        Satellite derived Images            */

#define MAIMG_ALL_PRODUCTS    -9997  /* Used in Query Structure nType
                                        field, indicates Query is for all
                                        Non - Satellite derived Images      */

#define MAIMG_ALL_OTHER       -9996  /* Used in Query Structure nType
                                        field, indicates Query is for all
                                        OTHER derived Images                */

#define MAIMG_NO_AOI          -9995  /* Denotes Imagery With no AOI         */

/* Product Types */

#define  MAIMG_OTHER                  1
#define  MAIMG_PRODUCT                2
#define  MAIMG_GOES_8                 3
#define  MAIMG_GOES_9                 4
#define  MAIMG_GOES_10                5
#define  MAIMG_GMS_4                  6
#define  MAIMG_GMS_5                  7
#define  MAIMG_GOMS_1N                8
#define  MAIMG_NOAA_9                 9
#define  MAIMG_NOAA_10                10
#define  MAIMG_NOAA_11                11
#define  MAIMG_NOAA_12                12
#define  MAIMG_NOAA_14                13
#define  MAIMG_NOAA_15                14
#define  MAIMG_METEOSAT_3             15
#define  MAIMG_METEOSAT_4             16
#define  MAIMG_METEOSAT_5             17
#define  MAIMG_METEOSAT_6             18
#define  MAIMG_METEOSAT_7             19
#define  MAIMG_INSAT_1B               20
#define  MAIMG_INSAT_2A               21
#define  MAIMG_INSAT_2B               22
#define  MAIMG_INSAT_2C               23
#define  MAIMG_DMSP_F8                24
#define  MAIMG_DMSP_F9                25
#define  MAIMG_DMSP_F10               26
#define  MAIMG_DMSP_F11               27
#define  MAIMG_DMSP_F12               28
#define  MAIMG_DMSP_F13               29
```

```
#define  MAIMF_DMSP_F14                30
#define  MAIMG_DMSP_COMP               31


/* Satellite Image Subtypes */
#define  MAIMG_GOES_IMAGER             1
#define  MAIMG_NOAA_SOUNDER            2
#define  MAIMG_GMS_VISSR               3
#define  MAIMG_GOMS_STR                4
#define  MAIMG_NOAA_MSU                5
#define  MAIMG_NOAA_SSU                6
#define  MAIMG_NOAA_HIRS2              7
#define  MAIMG_NOAA_AVHRR2             8
#define  MAIMG_METEOSAT_VISSR          9
#define  MAIMG_INSAT_VHRR              10
#define  MAIMG_DMSP_OLS2_SMOOTH        11
#define  MAIMG_DMSP_OLS2_FINE          12
#define  MAIMG_DMSP_SSMI               13
#define  MAIMG_DMSP_SSMT               14
#define  MAIMG_DMSP_SSMT2              15

/* Product Image Subtypes */
#define  MAIMG_HOR_WEATHER_DEPICTION   16
#define  MAIMG_ACOUSTIC_SENSOR_PROD    17
#define  MAIMG_ATMOSPHERIC_SENSOR_PROD 18
#define  MAIMG_TEXTUAL                 19
#define  MAIMG_FAX                     20
#define  MAIMG_BRIEFING                21
#define  MAIMG_MISSION_PLANNING        22
#define  MAIMG_SCREEN_DUMP             23

/* Satellite Channel Types */
#define  MAIMG_CHANNEL_1               1
#define  MAIMG_CHANNEL_2               2
#define  MAIMG_CHANNEL_3               3
#define  MAIMG_CHANNEL_4               4
#define  MAIMG_CHANNEL_5               5
#define  MAIMG_CHANNEL_6               6
#define  MAIMG_CHANNEL_7               7
#define  MAIMG_CHANNEL_8               8
#define  MAIMG_CHANNEL_9               9
#define  MAIMG_CHANNEL_10              10
#define  MAIMG_CHANNEL_11              11
#define  MAIMG_CHANNEL_12              12
#define  MAIMG_CHANNEL_13              13
#define  MAIMG_CHANNEL_14              14
#define  MAIMG_CHANNEL_15              15
#define  MAIMG_CHANNEL_16              16
#define  MAIMG_CHANNEL_17              17
#define  MAIMG_CHANNEL_18              18
#define  MAIMG_CHANNEL_19              19
#define  MAIMG_CHANNEL_20              20

/* Format Values */
typedef enum tagEMAIMGFORMAT
{
  MAIMG_OTHER_FMT = 0,
  MAIMG_NITF,
  MAIMG_MIF,
  MAIMG_GIF,
  MAIMG_TIFF,
```

```
  MAIMG_BMP,
  MAIMG_JPEG,
  MAIMG_XWD,
  MAIMG_XBM,
  MAIMG_PBM,
  MAIMG_MPEG
} EMAIMGTYPE;

/* Projection Values */
typedef enum tagEMAIMGPROJECTION
{
  MAIMG_NO_PROJECTION = 0,
  MAIMG_POLAR,
  MAIMG_LAMBERT,
  MAIMG_MERCATOR,
  MAIMG_SPHERICAL
} EMAIMGPROJECTION;

/* Data Category Values */
#define MAIMG_BASE    0
#define MAIMG_EDITED  1
#define MAIMG_DERIVED 2
```

3.1.2 Imagery Area of Interest (AOI) Structure

This structure defines the Geographical AOI. This structure is defined in `MAIMGAPI.h`. To specify a product with no AOI association, set all the fields in the `MAIMGGEOAREA` structure to `MAIMG_NO_AOI`.

```
typedef struct tagmaimgGeo {

   float rsNLat; /* North Latitude */
   float rsSLat; /* South Latitude */
   float rsWLon; /* West Longitude */
   float rsELon; /* East Longitude */

} MAIMGGEOAREA, *PMAIMGGEOAREA;
```

3.1.3 Imagery Data Structure

This structure contains the data for a single image record. This structure is defined in `MAIMGAPI.h`.

```
typedef struct tagmaimgData
{
   int           nType                                /* Product Type ID     */
   int           nSubType;                            /* Sub Type ID         */
   EMAIMGFORMAT  eImageFormatType;                    /* Format Type ID      */
   MAIMGGEOAREA  stGeoArea;                           /* Geography Descrip   */
   char          szAOIName[MAIMG_STRING_LEN];         /* AOI Descriptor
                                                          String              */
   char          szTitle[MAIMG_DESCRIP_LEN];          /* Product Title       */
   char          szDescription[MAIMG_DESCRIP_LEN];    /* Product Descrip     */
   int           nQualityIndicator;                   /* Quality Indicator   */
   int           nDataCategory;                       /* Data Category       */
   char          szSecurityClass[MAIMG_STRING_LEN];   /* Security Descrip    */
```

```
   char          szReceiptMethod[MAIMG_STRING_LEN];   /* Receipt Method      */
   char          szCompression[MAIMG_STRING_LEN];     /* Compression         */
   char          szOriginatingSite[MAIMG_STRING_LEN]; /* Location where
                                                         Imagery Product
                                                         creation occurred   */

   long          lBaseTime;                           /* time relative to
                                                         product (pass time
                                                         for satellites) in
                                                         epoch time          */

   EMAIMGPROJECTION     eProjectionType;              /* Projection Type ID  */
   MAIMGREFERENCEID     stReferenceID;                /* Object Identifiers
                                                         (returned)          */
   union {
     MAIMGSATELLITE     stSatellite;                  /* Satellite-specific
                                                         data                */
     MAIMGPRODUCT_INFO  stProduct;                    /* Non-Satellite
                                                         Product-specific
                                                         data                */
   } stImage;

   unsigned int         ulSize;                       /* Size in Bytes of
                                                         Image               */
   float                *pData;                       /* Pointer to image
                                                         data                */

} MAIMGDATA, *PMAIMGDATA;
```

3.1.4 Satellite Imagery Data Structures

These structures contains the data for a single Satellite Imagery record. These structures are defined in `MAIMGAPI.h`.

```
typedef struct tagmaimgSatellite
{
   int    nChannelID;              /* Channel indicator                    */
   int    nCalibrationTableSize;   /* Number of points in calibration table */
   float  *pCalibrationTable;      /* Calibration Points                   */
   MAIMGSATELLITE_INFO info;       /* Returned Satellite Information       */

} MAIMGSATELLITE,*PMAIMGSATELLITE;

  /************************************************************************ /
  /* Satellite Information that is returned on retrieval                  */
  /* Fields are not used during ingest.                                   */
  /************************************************************************/
typedef struct tagmaimgSatelliteInfo
{
   char   szSatelliteName[MAIMG_STRING_LEN];
   char   szCountryOrigin[MAIMG_STRING_LEN];
   char   szResponsibleAgency[MAIMG_STRING_LEN];
   char   szSensorName[MAIMG_STRING_LEN];
   char   szChannelName[MAIMG_STRING_LEN];
   char   szChannelDescription[MAIMG_STRING_LEN];
   long   lReceiptTime;            /* Time (epoch) data are stored in
                                      database                             */
```

```
} MAIMGSATELLITE_INFO, *PMAIMGSATELLITE_INFO;
```

3.1.5 Product Imagery Data Structures

This structure contains the data for a single Product Imagery record. These structures are defined in `MAIMGAPI.h`.

```
 /************************************************************************ /
 /* Product Imagery Information that is returned on retrieval              */
 /* Fields are not used during ingest.                                     */
 /************************************************************************/
typedef struct tagmaimgProductInfo
   char  szProductName[MAIMG_STRING_LEN];  /* Product Type Description      */
   long  lReceiptTime;                     /* Time (epoch) at which product
                                              was ingested                  */
} MAIMGPRODUCT_INFO, PMAIMGPRODUCT_INFO;
```

3.1.6 Imagery Object Reference Structure

This structure contains the data required to directly reference an Imagery record. These structures are defined in `MAIMGAPI.h`.

```
typedef struct tagmaimgReferenceID
{
   char szDatasetName[MAIMG_STRING_LEN]; /* Name of dataset table
                                            containing image              */
   long lRecordId;                       /* Record offset within the
                                            dataset table                 */
} MAIMGREFERENCEID, *PMAIMGREFERENCEID;
```

3.1.7 Imagery Linked List Structure

This is a generic structure used to keep linked lists. This structure is defined in the `MAIMGAPI.h`.

```
typedef struct tagmaimgLinkedList {
  void    *pNext;                        /* Pointer to next Link           */
  void    *pPrev;                        /* Pointer to the parent Link     */
  void    *pData;                        /* Pointer to the actual data     */
  int     nInternalMask;                 /* Internal usage only            */
} MAIMGLINKEDLIST, *PMAIMGLINKEDLIST;
```

3.1.8 Imagery Query Structure

The following is an input structure used to submit a catalog/imagery query. All fields must either be set or wildcarded. The area of interest cannot be wildcarded. This structure is defined in the file `MAIMGAPI.h`.

```
typedef struct tagmaimgQuery
{
   int    nType;                            /* nType may be Wild carded with:
                                               MAIMG_ALL_SATELLITE
                                               MAIMG_ALL_PRODUCTS
                                               MAIMG_ALL_OTHER
                                               MAIMG_QUERY_WILDCARD or set
                                               explicitly with a specified
                                               type as listed in
                                               section 3.1.1              */

   int    nSubType;                         /* Sub Type Ids maybe wild carded
                                               with: MAIMG_QUERY_WILDCARD or
                                               set explicitly with a
                                               specified sub type as listed
                                               in section 3.1.1           */

   EMAIMGPROJECTION  eProjectionType;       /* Projection Types as listed in
                                               section 3.1.1 or maybe set to
                                               MAIMG_QUERY_WILDCARD       */

   EMAIMGFORMAT      eFormatType;           /* Format Types as listed in
                                               section 3.1.1 or
                                               MAIMG_WILDCARD             */

   MAIMGGEOAREA      stGeoArea;             /* Geography Description, Fields
                                               may be set to look for a point
                                               (Corner points are set to same
                                               lat/long position), a bounding
                                               area, and to MAIMG_NO_AOI (all
                                               lat/long corner points set to
                                               this constant).

   char szAOIName[MAIMG_STRING_LEN];        /* AOI Descriptor String      */

   char szTitle[MAIMG_DESCRIP_LEN];         /* May be NULLED (Wildcarded) or
                                               set to a known product title*/

   char szDescription[MAIMG_DESCRIP_LEN];   /* May be NULLED (WildCarded) or
                                               set to a known product
                                               description                */

   int  nQualityIndicator;                  /* May be set to
                                               MAIMG_QUERY_WILDCARD or may be
                                               set to a user defined quality
                                               indicator value            */

   int  nDataCategory;                      /* Category
                                               MAIMG_QUERY_WILDCARD,
                                               MAIMG_BASE, MAIMG_EDITED,
                                               MAIMG_DERIVED              */
   char szSecurityClass[MAIMG_STRING_LEN];  /* May be NULLED (Wildcarded) or
                                               set to a known classification
                                               string                     */

   char szReceiptMethod[MAIMG_STRING_LEN];  /* May be NULLED (Wildcarded) or
                                               set to a known receipt method
                                               string                     */
```

```
   char szOriginatingSite[MAIMG_STRING_LEN]; /* May be NULLED (Wildcarded) or
                                                set to a known origination
                                                site string                */

   long lBeginBaseTime;                      /* Begin Base Time (epoch) lower
                                                bounds of time criteria
                                                search. May be set to
                                                MAIMG_QUERY_WILDCARD       */

   long lEndBaseTime;                        /* End Base Time (epoch) upper
                                                bounds of time criteria
                                                search. May be set to
                                                MAIMG_QUERY_WILDCARD       */

   long lBeginReceiptTime;                   /* Begin Receipt Time (epoch)
                                                lower bounds of time criteria
                                                search. May be set to
                                                MAIMG_QUERY_WILDCARD       */

   long lEndReceiptTime;                     /* End BaseTime (epoch) (lower
                                                bounds of time criteria
                                                search. May be set to
                                                MAIMG_QUERY_WILDCARD       */

   long lChannelID;                          /* Channel Id may be set when
                                                nType is set to a satellite
                                                product type or
                                                MAIMG_ALL_SATELLITE Or
                                                MAIMG_QUERY_WILDCARD,
                                                Otherwise this field is
                                                ignored                    */

} MAIMGQUERY, *PMAIMGQUERY;
```

3.1.9 Imagery Catalog Structure

The following structure is returned from a catalog query. This structure is defined in `MAIMGAPI.h`.

```
typedef struct tagmaimgCatalog
{
   int               nType;                      /* Image Type indicator   */
   int               nSubType;                   /* Image SubType
                                                    indicator             */
   MAIMGREFERENCEID  stReferenceID;              /* Image reference        */
   EMAIMGFORMAT      eImageFormatType;           /* Image format type      */
   MAIMGGEOAREA      stGeoArea;                  /* Image area of
                                                    interest              */
   char              szAOIName[MAIMG_STRING_LEN]; /* AOI Descriptor String */
   long              lCreateTime;                /* Time stored to
                                                    database (epoch)
                                                    (NOTE: This variable
                                                    is the same value as
                                                    1ReceiptTime, which
                                                    is used in other
                                                    Imagery structures)   */
   long              lBaseTime                   /* Base Time (epoch) of
                                                    product               */
```

```
   int                nQualityIndicator;           /* Quality indicator
                                                      value                 */
   int                nDataCategory;               /* Data Category value   */

   char               szTitle[MAIMG_DESCRIP_LEN];
   char               szDescription[MAIMG_DESCRIP_LEN];
   char               szSecurityClass[MAIMG_STRING_LEN];
   char               szOriginatingSite[MAIMG_STRING_LEN];
   char               szReceiptMethod[MAIMG_STRING_LEN];

   long               lChannelID;                  /* Set when Satellite image
                                                      specified             */
} MAIMGCATALOG, *PMAIMGCATALOG;
```

3.1.10 Imagery TypesData Structure

The following is returned from a type/subtype query. This structure is defined in `MAIMGAPI.h`.

```
typedef struct tagmaimgTypesData
{
   long     lTypeID;                            /* Image Type Indicator     */
   long     lSubTypeID;                         /* Image SubType Indicator  */
   char     szTypeName[MAIMG_STRING_LEN];       /* Image Type Name          */
   char     szSubTypeName[MAIMG_STRING_LEN];    /* Image SubType Name       */
}  MAIMGTYPESDATA, *PMAIMGTYPESDATA;
```

3.1.11 Imagery Return Structure

Each of the Imagery APIs returns this structure containing status data. The *nStatus* field will contain a zero upon successful completion of the API call. If the field is set to 1, there is an SQL error, and the `szSQLState` field must be examined. Any value of nStatus greater than 1 indicates an MAIMG error that maps to a define in the file `MAIMGErr.h`.

```
typedef struct tagmaimgret
{
   int                nStatus;       /* if nStatus == 0 ==> Success          */
                                     /* if nStatus == 1 ==> Check SQLState   */
                                     /* if nStatus >= 2 ==> Segment specific
                                        error                               */
   char               szSQLState[6]; /* First 2 characters represent the
                                        class, last 3 characters the
                                        subclass where an SQL error occurred */
   char   szErrorMessage[291];
}  MAIMGRET, *PMAIMGRET;
```

3.2 MAIMG Error Definitions

The following are the MAIMG error definitions contained in the file `MAIMGErr.h`.

```
#ifndef    MAIMGERR_OFFSET
#define    MAIMGERR_OFFSET    10000
#define    MAIMG_NULL_PTR                    1 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_IMAGE_TYPE          2 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_SUB_TYPE            3 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_FORMAT              4 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_LAT                 5 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_LON                 6 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_DATASIZE            7 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_TIME                8 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_INDICATOR           9 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_DATACATAGORY       10 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_PROD_DESCRIP       11 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_SECURITY_CLASS     12 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_ORIGIN_SITE        13 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_RECEIPT_METHOD     14 + MAIMGERR_OFFSET
#define    MAIMG_DATA_NOT_FOUND             15 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_OBJECTID           16 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_TABLENAME          17 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_FLAG               18 + MAIMGERR_OFFSET
#define    MAIMG_RECORD_ENTRY_FAILED        19 + MAIMGERR_OFFSET
#define    MAIMG_DELETE_RECORD_FAILED       20 + MAIMGERR_OFFSET
#define    MAIMG_RECORD_UPDATE_FAILED       21 + MAIMGERR_OFFSET
#define    MAIMG_RETRIEVE_IMAGE_FAILED      22 + MAIMGERR_OFFSET
#define    MAIMG_INVALIDLOCKMODE            23 + MAIMGERR_OFFSET
#define    MAIMG_INVALIDROLENAME            24 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_TITLE              25 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_COMPRESSION        26 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_PROJECTION         27 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_CHAN_ID            28 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_TABLE_SIZE         29 + MAIMGERR_OFFSET
#define    MAIMG_CALTABL_NULLPTR            30 + MAIMGERR_OFFSET
#define    MAIMG_NO_SATNAME                 31 + MAIMGERR_OFFSET
#define    MAIMG_NO_CNTRY_ORIG              32 + MAIMGERR_OFFSET
#define    MAIMG_NO_LAUNCHDATE              33 + MAIMGERR_OFFSET
#define    MAIMG_NO_AGENCY                  34 + MAIMGERR_OFFSET
#define    MAIMG_NO_SENSOR_NAME             35 + MAIMGERR_OFFSET
#define    MAIMG_NO_CHANNEL_NAME            36 + MAIMGERR_OFFSET
#define    MAIMG_NO_CHAN_DESCRIP            37 + MAIMGERR_OFFSET
#define    MAIMG_INVALID_PRODNAME           38 + MAIMGERR_OFFSET
#define    MAIMG_BEGIN_GRTRTHAN_END         39 + MAIMGERR_OFFSET
#define    MAIMG_PREPARE_FAIL               40 + MAIMGERR_OFFSET
#define    MAIMG_NO_CHANID                  41 + MAIMGERR_OFFSET
#define    MAIMG_NO_SATELLITE               42 + MAIMGERR_OFFSET
#define    MAIMG_BAD_TYPE_OR_SUBTYPE        43 + MAIMGERR_OFFSET
#define    MAIMG_MEMORY_ALLOC_FAILED        44 + MAIMGERR_OFFSET
#endif
```

4 IMAGERY API REFERENCE

Information about each API is presented in manual page format as follows:

NAME
Function Name – Provides a brief description of the function.

SYNOPSIS
Presents the calling syntax for the routine, including the declarations of the arguments and the return type. Also lists the necessary include files for each routine.

INPUT PARAMETERS
Describes each of the parameters used by the function.

OUTPUT PARAMETERS
Describes each of the parameters output by the function.

DESCRIPTION
Describes what the function does and what events or side effects it causes.

RETURNS
Describes what the function returns.

NOTE
1. Provides any applicable notes about the function.

SEE ALSO
Provides a reference to related functions.

Examples showing the proper usage of the APIs are presented in the *Imagery API Programming Manual*, referenced in Section 2.

4.1 MAIMGConnect

NAME

`MAIMGConnect` – Connects the application to the database.

SYNOPSIS

```
#include “MAIMGAPI.h”
MAIMGRET MAIMGConnect(void);
```

INPUT PARAMETERS

None.

OUTPUT PARAMETERS

None.

DESCRIPTION

This subroutine connects the application to the database.

RETURNS

MAIMGRET structure – See Section 3.1.11 for details.

NOTE

1. This interface must be called before any other MAIMG APIs may be called. It should only be called once per session.

SEE ALSO

`MAIMGDisconnect`, `MAIMGRemoteConnect`

4.2 MAIMGDisconnect

NAME

`MAIMGDisconnect()` – Disconnects the application from the database.

SYNOPSIS

```
#include "MAIMGAPI.h"
MAIMGRET MAIMGDisconnect(void);
```

INPUT PARAMETERS

None.

OUTPUT PARAMETERS

None.

DESCRIPTION

This subroutine disconnects the application from the database, ending the database session.

RETURNS

MAIMGRET structure – See Section 3.1.11 for details.

NOTE

1. `MAIMGDisconnect` should be called to end each database session started by `MAIMGConnect`.

SEE ALSO

`MAIMGConnect`

4.3 MAIMGRemoteConnect

NAME

`MAIMGRemoteConnect` – Allows the application to connect to a database residing on a remote server and/or to establish multiple connections.

SYNOPSIS

```
#include "MAIMGAPI.h"
MAIMGRET MAIMGConnect
(
      char *szDBServer,
      char *szConnectionName
);
```

INPUT PARAMETERS

`char *szDBServer` – Database server name (COE default is `$INFORMIXSERVER`)

`char *szConnectionName` – Connection name for MDIMG database `MAIMG_DEFAULT_CONNECTION` is default)

OUTPUT PARAMETERS

None.

DESCRIPTION

The `MAIMGRemoteConnect` routine allows an application to connect to a remote database server and/or to maintain multiple open connections on the same or different servers.

RETURNS

MAIMGRET structure – See Section 3.1.11 for details.

NOTES

1. This interface must be called before any other MAIMG APIs may be called. It should only be called once per session with the same arguments.
2. Passing in NULL for either argument assumes that the default settings are to be used for server (`$INFORMIXSERVER`) and connection name (`MAIMG_DEFAULT_CONNECTION`).
3. Applications connecting to only one server and one database at a time should use the simpler `MAIMGConnect` and `MAIMGDisconnect` routines.

SEE ALSO

`MAIMGRemoteDisconnect`, `MAIMGSetConnection`, `MAIMGConnect`

4.4 MAIMGRemoteDisconnect

NAME

`MAIMGRemoteDisconnect` – Disconnects the application from the database residing on a remote server.

SYNOPSIS

```
#include "MAIMGAPI.h"
MAIMGRET MAIMGRemoteDisconnect
(
char *szConnectionName
);
```

INPUT PARAMETERS

`char *szConnectionName` – Connection to be disconnected.

OUTPUT PARAMETERS

None.

DESCRIPTION

This subroutine disconnects the application from the database connection specified, ending the database session.

RETURNS

`MAIMGRET` structure – See Section 3.1.11 for details.

NOTE

1. `MAIMGRemoteDisconnect` should be called to end each database connection session started by `MAIMGRemoteConnect`. It should only be called once per session with the same arguments.

SEE ALSO

`MAIMGRemoteConnect`, `MAIMGSetConnection`

4.5 MAIMGSetConnection

NAME

`MAIMGSetConnection` – Set the connection context for multiple connections to different database servers and/or databases.

SYNOPSIS

```
#include "MAIMGAPI.h"
MAIMGRET MAIMGSetDisconnect
(
char *szConnectionName
);
```

INPUT PARAMETERS

`char *szConnectionName` – Connection desired for database transactions.

OUTPUT PARAMETERS

None.

DESCRIPTION

The `MAIMGSetConnect` routine is used to set the connection context for multiple connections to different database servers and/or databases.

RETURNS

`MAIMGRET` structure – See Section 3.1.11 for details.

NOTE

1. Passing in NULL for the connection name assumes that the default connection is to be used (`MAIMG_DEFAULT_CONNECTION`).

SEE ALSO

`MAIMGRemoteConnect`, `MAIMGRemoteDisconnect`

4.6 MAIMGIngest

NAME

`MAIMGIngest` – Ingests an Imagery record into the database.

SYNOPSIS

```
#include "MAIMGAPI.h"
MAIMGRET MAIMGIngest
(
      PMAIMGDATA  pMAIMGData
);
```

INPUT PARAMETERS

`PMAIMGDATA` `pMAIMGData` – A pointer to a `MAIMGDATA` structure (Section 3.1.3) containing Imagery description and data to be ingested.

OUTPUT PARAMETERS

None.

DESCRIPTION

The `MAIMGIngest` function ingests Imagery into the database. The routine takes as input a pointer to the `MAIMGDATA` structure containing the Imagery description and data. The routine returns the reference identifiers (`stReferenceID.szDatasetName`, `stReferenceID.lRecordID`) after the imagery object has been added to the database (contained in the `MAIMGDATA` structure).

RETURNS

`MAIMGRET` structure – See Section 3.1.11 for details.

NOTE

1. `MAIMGConnect` or `MAIMGRemoteConnect` must have been called to start a database session before `MAIMGIngest` may be called.

SEE ALSO

MAIMGConnect, MAIMGRemoteConnect

4.7 MAIMGCatalog

NAME

`MAIMGCatalog` – Retrieve catalog listing of Imagery records meeting specified criteria.

SYNOPSIS

```
#include "MAIMGAPI.h"
 MAIMGRET MAIMGCatalog
 (
       PMAIMGQUERY pImageryQuery,
       long *plNumFound,
       PMAIMGLINKEDLIST pCatList
);
```

INPUT PARAMETERS

`PMAIMGQUERY pImageryQuery` – A pointer to a `MAIMGQUERY` structure (Section 3.1.8) containing query criteria for the catalog retrieval.

OUTPUT PARAMETERS

`long *plNumFound` – A pointer to the number of records found that match the query criteria

`PMAIMGLINKEDLIST pCatList` – A pointer to a linked list (Section 3.1.7) of `MAIMGCATALOG` (Section 3.1.9) structures containing the catalog data retrieved.

DESCRIPTION

The function retrieves a catalog listing of imagery in the database that satisfies the query criteria provided in the input `MAIMGQUERY` structure. It returns the number of catalog items found, a linked list of the catalog items, and the return status structure, `MAIMGRET`.

RETURNS

`MAIMGRET` structure – See Section 3.1.11 for details.

NOTES

1. `MAIMGFreeLL` should be called to free the linked list upon completion of processing.
2. Wildcards are permitted on all fields except for the `pImageryQuery` –> `stGeoArea` field. The area of interest must be specified as either a point, a bounding box, or `MAIMG_NO_AOI` (indicating that you desire images with no geographic association). Wildcarding of most fields is done with `MAIMG_QUERY_WILDCARD`; the exception is for string fields, which use a null setting (memset of structure will set them initially to NULL).

3. While wildcarding is permitted and maximum flexibility is allowed, the programmer should use some discretion when using them. Wildcarding is likely to result in a large volume of data being returned, which may tax system/network resources.

SEE ALSO

`MAIMGConnect`, `MAIMGRemoteConnect`, `MAIMGFreeLL`

4.8 MAIMGGetByID

NAME

`MAIMGGetByID` – Retrieve a single image from the database given the object identifiers for the imagery record of interest.

SYNOPSIS

```
#include "MAIMGAPI.h"
 MAIMGRET  MAIMGGetByID
 (
      MAIMGREFERENCEID stReference,
      PMAIMGDATA pImageData
);
```

INPUT PARAMETERS

`MAIMGREFERENCEID stReference` – Object identifiers (Section 3.1.6) for the image record of interest.

OUTPUT PARAMETERS

`PMAIMGDATA pImageData` – A pointer to the `MAIMGDATA` structure (Section 3.1.3) containing the retrieved image.

DESCRIPTION

The `MAIMGGetByID` routine takes as input the object identifier for the imagery record to be retrieved. The reference identifier structure uniquely identifies an image in the database. It returns the requested image in the `MAIMGDATA` structure.

RETURNS

`MAIMGRET` structure – See Section 3.1.11 for details.

NOTE

1. `MAIMGConnect` must have been called to start a database session before `MAIMGGetByID` may be called.

SEE ALSO

`MAIMGConnect`, `MAIMGRemoteConnect`, `MAIMGCatalog`

4.9 MAIMGGetByQuery

NAME

`MAIMGGetByQuery` – Retrieve one or more images from the database that match the specified query criteria.

SYNOPSIS

```
include "MAIMGAPI.h"
MAIMGRET  MAIMGGetByQuery
(
      PMAIMGQUERY pImageryQuery,
      long *plNumFound,
      PMAIMGLINKEDLIST pImageData
);
```

INPUT PARAMETERS

`PMAIMGQUERY pImageryQuery` – A pointer to a `MAIMGQUERY` structure (Section 3.1.8) containing query criteria for the imagery retrieval.

OUTPUT PARAMETERS

`long *plNumFound` – A pointer to the number of records found that match the query criteria

`PMAIMGLINKEDLIST pImageData` – A pointer to the `MAIMGLINKEDLIST` structure (Section 3.1.7) that points to the `MAIMGDATA` structure (Section 3.1.3) containing the retrieved images.

DESCRIPTION

The `MAIMGGetByQuery` routine takes as input a `MAIMGQUERY` structure containing the query criteria. It returns the number of matching images found, a linked list that points to `MAIMGDATA` structures containing the images, and the `MAIMGRET` return status structure.

RETURNS

`MAIMGRET` structure – See Section 3.1.11 for details.

NOTES

1. `MAIMGConnect` must have been called to start a database session before `MAIMGGet` may be called.
2. `MAIMGFreeLL` should be called to free the linked list upon completion of processing.
3. Wildcards are permitted on all fields except for the `pImageryQuery` –> `stGeoArea` field. The area of interest must be specified as either a point, a bounding box, or `MAIMG_NO_AOI` (indicating that you desire images with no geographic association). Wildcarding of most fields

is done with `MAIMG_QUERY_WILDCARD`; the exception is for string fields, which use a null setting (memset of structure will set them initially to NULL).

4. While wildcarding is permitted and maximum flexibility is allowed, the programmer should use some discretion when using them. Wildcarding is likely to result in a large volume of data being returned, which may tax system/network resources.

SEE ALSO

`MAIMGConnect`, `MAIMGRemoteConnect`, `MAIMGFreeLL`

4.10 MAIMGUpdateByID

NAME

`MAIMGUpdateByID` – Updates an imagery record in the database.

SYNOPSIS

```
include "MAIMGAPI.h"
MAIMGRET  MAIMGUpdateByID
(
      MAIMGREFERENCEID stReference,
      PMAIMGDATA pImageData
);
```

INPUT PARAMETERS

`MAIMGREFERENCEID stReference` – Object identifiers (Section 3.1.6) for the image record of interest.

`PMAIMGDATA pImageData` – A pointer to a `MAIMGDATA` structure (Section 3.1.3) containing the data with which to update the record in the database.

OUTPUT PARAMETERS

None.

DESCRIPTION

The `MAIMGUpdateByID` routine updates an imagery record in the database with new data contained in the input `MAIMGDATA` structure. The record to be updated is identified in the `MAIMGREFERENCEID` structure specified. It returns a `MAIMGRET` structure indicating the status of the operation.

RETURNS

`MAIMGRET` Structure – See Section 3.1.11 for details.

NOTE

1. `MAIMGConnect` must have been called to start a database session before `MAIMGUpdateByID` may be called.
2. An update to a `MAIMG_BASE` imagery record will automatically generate a new `MAIMG_EDITED` imagery record. Base imagery records may not be modified. The new record id is contained within the `pImageData` structure.
3. Typically a call to `MAIMGGetByID` must be done first in order to get the imagery record to be modified.

SEE ALSO

`MAIMGConnect`, `MAIMGRemoteConnect`, `MAIMGGetByID`

4.11 MAIMGDeleteByID

NAME

`MAIMGDeleteByID` – Delete an imagery record from a dataset in the database.

SYNOPSIS

```
include "MAIMGAPI.h"
 MAIMGRET  MAIMGDeleteByID
 (
      MAIMGREFERENCEID stReference
);
```

INPUT PARAMETERS

`MAIMGREFERENCEID stReference` – Object identifier (Section 3.1.6) for the image record of interest.

OUTPUT PARAMETERS

None.

DESCRIPTION

The `MAIMGDeleteByID` routine deletes an imagery record. It returns a `MAIMGRET` structure indicating the status of the operation.

RETURNS

`MAIMGRET` Structure – See Section 3.1.11 for details.

NOTE

1. `MAIMGConnect` must have been called to start a database session before `MAIMGDeleteByID` may be called.
2. Typically a call to `MAIMGCatalog` to get the list of imagery records is made before the `MAIMGDeleteByID` call.

SEE ALSO

`MAIMGConnect`, `MAIMGRemoteConnect`, `MAIMGCatalog`

4.12 MAIMGGetTypesInfo

NAME

`MAIMGGetTypesInfo` – Retrieves the list of types and associated subtypes that match the given query criteria.

SYNOPSIS

```
include "MAIMGAPI.h"
MAIMGRET MAIMGGetTypesInfo
(
    long lTypeID,
    long lSubTypeID,
    long *plNumFound,
    PMAIMGLINKEDLIST pTypesList
);
```

INPUT PARAMETERS

`long lTypeID` – Image Type Indicator.

`long lSubTypeID` – Image subType Indicator.

OUTPUT PARAMETERS

`long *plNumFound` – Number of records found.

`PMAIMGLIKEDLIST pTypesList` – A pointer to a `MAIMGLINKEDLIST` (Section 3.1.7) that points to the `MAIMGTYPESDATA` structure (Section 3.1.10) containing the retrieved types and subtypes information.

DESCRIPTION

This subroutine retrieves the list of types/subtypes records found to match the type ID and subtype ID specified (may be wildcarded). The information retrieved in each record is the type ID, type name, subtype ID, and subtype name.

RETURNS

`MAIMGRET` structure – See Section 3.1.11 for details.

NOTES

1. `MAIMGConnect` must have been called to start a database session before `MAIMGGetTypesInfo` may be called.
2. `lTypeID` and `lSubTypeID` may be wildcarded by using `MAIMG_QUERY_WILDCARD`. When both are wildcarded, the retrieval will return all parameter definitions known to the system (common and type/subtype specific parameters). A setting of both with specific values will return one record as found for the specific type/subtype ID combination.
3. `MAIMGFreeLL` should be called to free the linked list upon completion of processing.

SEE ALSO

`MAIMGGetByQuery`, `MAIMGCatalog`

4.13 MAIMGFreeLL

NAME

`MAIMGFreeLL` – Frees the linked list associated with catalog retrieval and imagery retrieval.

SYNOPSIS

```
include "MAIMGAPI.h"
 int MAIMGFreeLL
(
    PMAIMGLINKEDLIST pLL
);
```

INPUT PARAMETERS

`PMAIMGLINKEDLIST pLinkedList` – A pointer to a `MAIMGLINKEDLIST` (Section 3.1.7) linked list of data.

OUTPUT PARAMETERS

`int nReturnValue` – A status value is returned. A status value of zero indicates a successful completion.

DESCRIPTION

`MAIMGFreeLL` frees a linked list that has been returned by calls to `MAIMGCatalog` or `MAIMGGetByQuery`. It returns a status value.

RETURNS

`MAIMGRET` structure – See Section 3.1.11 for details.

SEE ALSO

`MAIMGGetByQuery`, `MAIMGCatalog`

5 NOTES

5.1 Glossary of Acronyms

AESS	Allied Environmental Support System
AOI	Area of Interest
API	Application Program Interface
APIRM	API Reference Manual
COE	Common Operating Environment
DII	Defense Information Infrastructure
GCCS	Global Command and Control System
IC4ISR	Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance
IMOSS	Interim Mobil Oceanographic Support System
JMCIS	Joint Maritime Command Information System
JMS	Joint METOC Segment
LLT	Latitude-Longitude-Time
MAIMG	METOC Imagery API Segment
METOC	Meteorology and Oceanography
MIDDS	Meteorological Integrated Data Display System

NITES	Navy Integrated Tactical Environmental Subsystem
PM	Programming Manual
PS	Performance Specification
SQL	Structured Query Language
TESS(NC)	Tactical Environmental Support System Next Century