ipd4200imgpmTES-10

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

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

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
2 REFERENCED DOCUMENTS 5
2.1 Government Documents 5
3 SEGMENT OVERVIEW 7
4 SEGMENT DEVELOPMENT 9
4.1 Writing Applications Using the MAIMG APIs 9
4.1.1 Ingesting an Image Into the Database 10
4.1.2 Deleting Imagery Products From the Database 13
4.1.3 Getting a Catalog of Imagery Products From the Database 16
4.1.4 Retrieving Imagery Products From the Database by Query 17
4.1.5 Retrieving Imagery Products from the Database By Reference (ID) 20
4.1.6 Retrieving Imagery Type and SubType Information 21
4.1.7 Updating Imagery Products in the Database 23
4.2 MDIMG Database Reference Tables 26
4.2.1 MDIMG Type and SubType Tables 26
4.3 Building Applications Using the MAIMG Libraries 32
4.3.1 Makefile Using the Dynamic MAIMG Libraries on HP-UX 32
4.3.2 Makefile Using the Static MAIMG Libraries on HP-UX 32
4.3.3 Makefile Using the Dynamic MAIMG Libraries on Windows NT 33
4.3.4 Makefile Using the Static MAIMG Libraries on Windows NT 33
5 CUSTOMIZING SEGMENTS 35
6 NOTES 37
6.1 Glossary of Acronyms 37

List of Tables

4-1 MDIMGType/MDIMGSubType Join Table 27

List of Figures

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

(This page intentionally left blank.)

1 SCOPE

1.1 Identification

This Programming Manual (PM) describes the use of the Imagery Application Program Interface (API) (MAIMG) segment 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 product data. 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.

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).

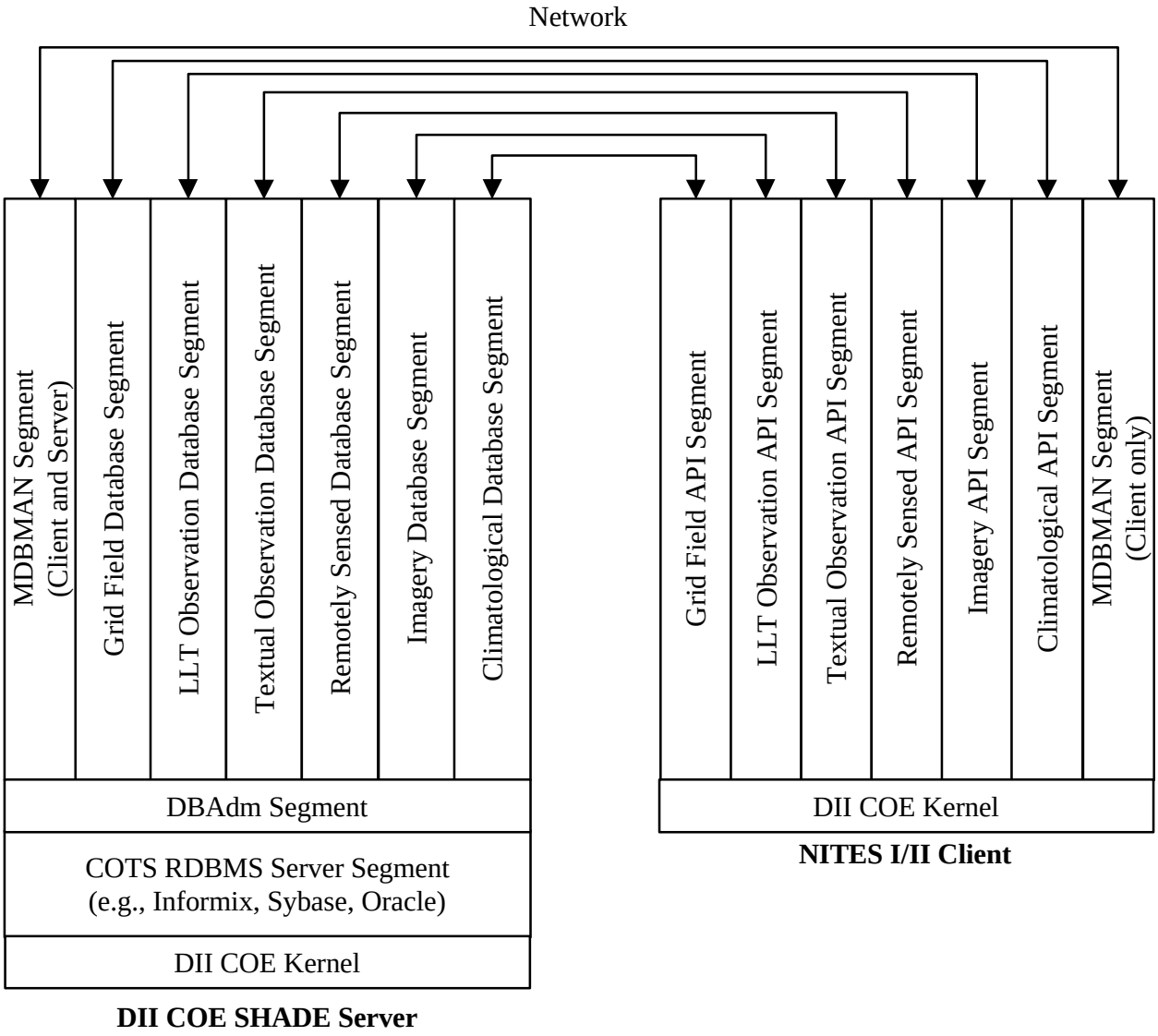

Figure 1-1. TESS(NC) METOC Database Conceptual Organization

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. The supported formats 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 29 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*
ipd4200maimgrmTES-10 9 October 1998	*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*

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*

3 SEGMENT OVERVIEW

The MAIMG segment provides APIs used to access Imagery data in the TESS(NC) METOC Database. The schema for storing these imagery products is defined by the shared database segment MDIMG. Both of these segments require the Informix Relational Database Management System (RDBMS), version 7.22 (for HP-UX machines) or 7.23 (for Windows NT machines). Both segments run in the DII COE release 3.1, hosted on the following machines and operating systems:

- Tactical Advanced Computer, TAC-3 (HP 750/755)/TAC-4 (HP J201), Operating System: HP-UX 10.20
- IBM-Compatible Personal Computer (PC), Operating System: Microsoft Windows NT 4.0, with Service Pack 3

MAIMG uses the following environment variables related to the Informix installation:

- `INFORMIXSERVER` Identifies the Informix server, typically set to `online_coe`
- `INFORMIXDIR` Path to the Informix software, typically `/opt/informix` on HP systems and `C:\informix` on Windows NT systems

The path specified in the `INFORMIXDIR` variable should also be included in the system's PATH variable.

Environmental Variable settings to find shared MAIMG shared libraries are:

- NT Operating System
  `setenv PATH=$PATH;c:/h/MAIMG/bin`
- UNIX Operating System
  `setenv SHLIB_PATH /h/MAIMG/bin`

In the NT environment, programmers will need to define the _MDBDLL macro. This macro is used to indicate that declspec is to use dllimport.

The MAIMG segment is delivered as both archive and runtime libraries on both platforms, with filenames as follows:

Library Type	**HP-UX Filename**	**Windows NT Filename**
Archives	libMAIMGAPI.a	MAIMGAPI.lib
Runtime	MAIMGAPI.sl	MAIMGAPI.dll

The runtime libraries are typically installed in the directory `/h/MAIMG/bin` on HP systems and `C:\h\MAIMG\bin` on Windows NT systems.

The archive libraries are typically installed in the directory `/h/MAIMG/lib` on HP systems and `C:\h\MAIMG\lib` on Windows NT systems.

The individual API methods are described in the APIRM referenced in Section 2. Section 4 of this document provides instructions and programming examples for the use of the APIs.

4 SEGMENT DEVELOPMENT

Programming applications to access imagery product data are straightforward. The MAIMG 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 a listing of Type and SubType information meeting specified criteria from the database (MAIMGGetTypesInfo)
- 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)
- Free the linked lists returned by MAIMGCatalog and MAIMGGetByQuery (MAIMGFreeLL)
- Disconnect from the database (MAIMGDisconnect, MAIMGRemoteDisconnect).

Each of these methods is described in detail in the MAIMG APIRM, referenced in Section 2.

4.1 Writing Applications Using the MAIMG APIs

This section shows the use of the MAIMG APIs to perform common data access tasks. In each case, an overview of the actions to be performed is provided, along with a code example and a discussion of pertinent programming concerns.

In all cases, note that the database connect method (MAIMGConnect, or MAIMGRemoteConnect) must be called to connect the application to the database before any other operations may be performed. The database disconnect method (MAIMGDisconnect or

MAIMGRemoteDisconnect) should be called to disconnect the application from the database at the end of the session. These methods only need to be called once per session.

All structures must be initialized through use of calloc or memset. Failure to initialize a structure could cause unpredictable results. Garbage in a structure could cause a query to fail.

The MAIMGRET structure is used to return status information from each MAIMG method. See Section 3.1.10 of the APIRM for details of the information returned in this structure.

4.1.1 Ingesting an Image Into the Database

The MAIMGIngest method is used to add an Imagery record to the database. It takes as input a pointer to a MAIMGDATA structure containing the metadata for the image and the image data to be added. All data in the MAIMGDATA structure should be filled in. The code below provides an example of the procedure for adding a Satellite Imagery product as well as a non-Satellite imagery product to the database.

```
/*****************************************************************************
                            T   E   S   T   E   R

Purpose: Sample Code using MAIMGIngest.
*****************************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "MAIMGAPI.h"

#ifdef _WIN32
#define _MDBDLL
#endif

void GetNITFImageData(unsigned int *pulSize, float **pData);
void GetMIFFImageData(unsigned int *pulSize, float **pData);

/*****************************************************************************
                              M   A   I   N
*****************************************************************************/
void main(argc, argv)
int argc;
char **argv;
{

   struct tm      timeptr;
   int            nStatus = 0;
   MAIMGRET       maimgRet;
   MAIMGDATA      ImageData;

   maimgRet = MAIMGConnect();
   if (maimgRet.nStatus)
```

```
{
   printf( "\ttester: Error calling MAIMGConnect (%d) (%s)\n",
   maimgRet.nStatus, maimgRet.szErrorMessage );
}
else
{
   printf( "\tMAIMGConnect : Successful!\n\n");
   memset(&ImageData, 0, sizeof(MAIMGDATA));
   memset(&timeptr,0,sizeof(struct tm));

   /**************************************************
    Set up input to ingest a Satellite Image product
   **************************************************/
   ImageData.nType            = MAIMG_NOAA_9;
   ImageData.nSubType         = MAIMG_NOAA_MSU;
   ImageData.eImageFormatType = MAIMG_NITF;
   ImageData.nQualityIndicator = 0;
   ImageData.nDataCategory = MAIMG_BASE;

   strcpy(ImageData.szTitle,"NOAA 6 Pass");
   strcpy(ImageData.szDescription,"Hurricane Gloria");
   strcpy(ImageData.szSecurityClass, "UNCLASS");
   strcpy(ImageData.szCompression, "NONE");
   strcpy(ImageData.szReceiptMethod, "TCP");
   strcpy(ImageData.szOriginatingSite, "NRL-STENNIS");

   ImageData.eProjectionType  = MAIMG_SPHERICAL;
   ImageData.stGeoArea.rsNLat =  30.0;
   ImageData.stGeoArea.rsWLon = -90.0;
   ImageData.stGeoArea.rsSLat =  20.0;
   ImageData.stGeoArea.rsELon =  -80.0;
   strcpy(ImageData.szAOIName, "Gulf Of Mexico");

   ImageData.stImage.stSatellite.nChannelID = MAIMG_CHANNEL_1;

   ImageData.stImage.stSatellite.nCalibrationTableSize = 6;
   ImageData.stImage.stSatellite.pCalibrationTable =
      (float *) calloc(1, sizeof(float) *
      ImageData.stImage.stSatellite.nCalibrationTableSize);

   ImageData.stImage.stSatellite.pCalibrationTable[0] = 10.0;
   ImageData.stImage.stSatellite.pCalibrationTable[1] = 20.0;
   ImageData.stImage.stSatellite.pCalibrationTable[2] = 22.0;
   ImageData.stImage.stSatellite.pCalibrationTable[3] = 35.6;
   ImageData.stImage.stSatellite.pCalibrationTable[4] = 50.9;
   ImageData.stImage.stSatellite.pCalibrationTable[5] = 60.1;

   /* Compute basetime (epoch) for Sept 1, 1994 */
   timeptr.tm_year = 94;
   timeptr.tm_mon  = 8;
   timeptr.tm_mday = 1;
   timeptr.tm_hour = 0;
   ImageData.lBaseTime = (long) mktime(&timeptr);

   /*********************************************************
    Create a dummy nitf image blob.
   *********************************************************
   GetNITFImageData(&ImageData.ulSize, &ImageData.pData);
```

```
/****************************
 Call MAIMGIngest.
****************************/
maimgRet = MAIMGIngest( &ImageData ) ;

if (maimgRet.nStatus)
{
   printf( "\tMAIMGIngest: Error!(%d) (%s)\n",
              maimgRet.nStatus, maimgRet.szErrorMessage );
}
else
{
   printf("\tMAIMGIngest: %s Successful (%d)!\n",
   ImageData.stReferenceID.szDatasetName,
   ImageData.stReferenceID.lRecordID);
}

/*****************************************************
 Set up input to ingest a Non-Satellite Image product
*****************************************************/
ImageData.nType              = MAIMG_PRODUCT;
ImageData.nSubType           = MAIMG_BRIEFING;
ImageData.eImageFormatType   = MAIMG_MIFF;
ImageData.nQualityIndicator  = 0;
ImageData.nDataCategory      = MAIMG_BASE;

strcpy(ImageData.szTitle," BRIEF PAGE 1");
strcpy(ImageData.szDescription," September 2, 1994 0000Z");
strcpy(ImageData.szSecurityClass, "UNCLASS");
strcpy(ImageData.szCompression, "NONE");
strcpy(ImageData.szReceiptMethod, "TCP");
strcpy(ImageData.szOriginatingSite, "ROTA");

ImageData.eProjectionType  =MAIMG_NO_PROJECTION;
ImageData.stGeoArea.rsNLat =MAIMG_NO_AOI;
ImageData.stGeoArea.rsWLon =MAIMG_NO_AOI;
ImageData.stGeoArea.rsSLat =MAIMG_NO_AOI;
ImageData.stGeoArea.rsELon =MAIMG_NO_AOI;

/* Compute basetime (epoch) for Sept 2, 1994 */
timeptr.tm_year = 94;
timeptr.tm_mon  = 8;
timeptr.tm_mday = 2;
timeptr.tm_hour = 0;
ImageData.lBaseTime = (long) mktime(&timeptr);

/****************************************************
 Create a dummy miff image blob.
****************************************************/
GetMIFFImageData(&ImageData.ulSize, &ImageData.pData);

/****************************
 Call MAIMGIngest.
****************************/
maimgRet = MAIMGIngest( &ImageData ) ;

if (maimgRet.nStatus)
{
   printf( "\tMAIMGIngest: Error!(%d) (%s)\n",
```

```
                    maimgRet.nStatus, maimgRet.szErrorMessage );
      }
      else
      {
         printf("\tMAIMGIngest: %s Successful (%d)!\n",
         ImageData.stReferenceID.szDatasetName,
         ImageData.stReferenceID.lRecordID);
      }

      /************************
       Call MAIMGDisconnect.
      ************************/
      maimgRet = MAIMGDisconnect();
   }
   exit( maimgRet.nStatus );

} /* End of main. */

/*************************************************************
 Dummy routines to create NITF and MIFF images.
 In real world will likely set pointers to any existing image.
*************************************************************/
void GetNITFImageData (unsigned int *pulSize, float **pData)
{
   float *pNITFData;
   *pulSize =  500*sizeof(float);   /* NITF image blob is 2000 bytes */

   /***********************************
    Create a dummy images now.
   ***********************************/
   pNITFData = calloc(500, sizeof(float));
   memset(pNITFData,1,500);
   *pData = pNITFData;
}

void GetMIFFImageData(unsigned int *pulSize, float **pData)
{
   float *pMIFFData;
   *pulSize =  1000*sizeof(float);   /* MIFF image blob is 40000 bytes */

   /***********************************
    Create a dummy images now.
   ***********************************/
   pMIFFData = calloc(1000, sizeof(float));
   memset(pMIFFData,1,1000);
   *pData = pMIFFData;
}
```

4.1.2 Deleting Imagery Products From the Database

The MAIMGDeleteByID method is used to delete an Imagery product from the database. This method requires the dataset name and the record id for the imagery product to be deleted. Typically the MAIMGCatalog method is used to get a list of the candidate records for deletion, as shown in the example below. Once the list has been retrieved, the program can cycle through the list and call MAIMGDeleteByID to delete the current element.

```
/*****************************************************************************
                            T   E   S   T   E   R

Purpose: Sample Code using MAIMGDelete.
*****************************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "MAIMGAPI.h"

#ifdef _WIN32
#define _MDBDLL
#endif

/*****************************************************************************
                                M   A   I   N
*****************************************************************************/
void main(argc, argv)
int argc;
char **argv;
{

   int      nStatus = 0;
   long     lNumFound;
   int      n;

   MAIMGRET            maimgRet;
   MAIMGQUERY          ImageQuery;
   MAIMGLINKEDLIST     CatList, *pImageDataLL;
   PMAIMGCATALOG       pCatData;

   maimgRet = MAIMGConnect();
   if (maimgRet.nStatus)
   {
      printf( "\ttester: Error calling MAIMGConnect (%d) (%s)\n",
      maimgRet.nStatus, maimgRet.szErrorMessage );
   }
   else
   {
      printf( "\tMAIMGConnect : Successful!\n\n");

      /**************************************************
      Set up input Catalog Query.
      Get list of all Imagery Products
      that have any AOI association.
      Note: memset of structure will take care of
      of wildcarding strings.
      **************************************************/
      memset(&ImageQuery, 0, sizeof(MAIMGQUERY));

      ImageQuery.nType = MAIMG_QUERY_WILDCARD;
      ImageQuery.nSubType = MAIMG_QUERY_WILDCARD;
      ImageQuery.eProjectionType =  MAIMG_QUERY_WILDCARD;
      ImageQuery.eFormatType =  MAIMG_QUERY_WILDCARD;

      ImageQuery.stGeoArea.rsNLat  =   90.0;
      ImageQuery.stGeoArea.rsWLon  = -180.0;
```

```
ImageQuery.stGeoArea.rsSLat  =  -90.0;
ImageQuery.stGeoArea.rsELon  =  180.0;

ImageQuery.nQualityIndicator  = MAIMG_QUERY_WILDCARD;
ImageQuery.nDataCategory =  MAIMG_QUERY_WILDCARD;

ImageQuery.lBeginBaseTime = MAIMG_QUERY_WILDCARD;
ImageQuery.lEndBaseTime = MAIMG_QUERY_WILDCARD;
ImageQuery.lBeginReceiptTime = MAIMG_QUERY_WILDCARD;
ImageQuery.lEndReceiptTime = MAIMG_QUERY_WILDCARD;

ImageQuery.lChannelID = MAIMG_QUERY_WILDCARD;

/*************************
 Call MAIMGCatalog.
*************************/
maimgRet = MAIMGCatalog(&ImageQuery, &lNumFound, &CatList);

if (maimgRet.nStatus)
{
   printf( "\tMAIMGCatalog: Error!(%d) (%s)\n",
             maimgRet.nStatus, maimgRet.szErrorMessage );

}
else
{
   printf("\tMAIMGCatalog: Successful !\n");

   /*************************
    Display output values.
   *************************/
   if (lNumFound < 1)
   {
      printf("\nNo catalog entries were found.\n");
   }
   else
   {
      n = 1;
      pImageDataLL = &CatList;
      printf("\nNumber catalog entries were found %d.\n", lNumFound);

      while (n <= lNumFound)
      {
         pCatData = ( PMAIMGCATALOG )pImageDataLL->pData;

         printf("\n\tTableName = %s\n",
         pCatData->stReferenceID.szDatasetName);
         printf("\tObjectID = %d\n", pCatData->stReferenceID.lRecordID);
         printf("\tImageType = %d\n", pCatData->nType);
         printf("\tImageSubType = %d\n", pCatData->nSubType);
         printf("\tFormatType = %d\n", pCatData->eImageFormatType);
         printf("\tAOIName = %s\n", pCatData->szAOIName);
         printf("\tNorth Lat = %f\n", pCatData->stGeoArea.rsNLat);
         printf("\tSouth Lat = %f\n", pCatData->stGeoArea.rsSLat);
         printf("\tWest Lon = %f\n", pCatData->stGeoArea.rsWLon);
         printf("\tEast Lon = %f\n", pCatData->stGeoArea.rsELon);
         printf("\tBase Time = %d\n", pCatData->lBaseTime);
         printf("\tReceipt Time = %d\n", pCatData->lCreateTime);
         printf("\tQualityIndicator = %d\n", pCatData->nQualityIndicator);
```

```
                printf("\tData Category = %d\n", pCatData->nDataCategory);
                printf("\tTitle = %s\n", pCatData->szTitle);
                printf("\tDescription = %s\n", pCatData->szDescription);
                printf("\tSecurityClass = %s\n", pCatData->szSecurityClass);
                printf("\tReceiptMethod = %s\n", pCatData->szReceiptMethod);
                printf("\tOriginatingSite = %s\n", pCatData->szOriginatingSite);

                if ((pCatData->nType != MAIMG_PRODUCT) &&
                    (pCatData->nType != MAIMG_OTHER))
                {
                   printf("\tChannel ID = %d\n", pCatData->lChannelID);
                }

                /**********************
                 Delete Each Record.
                **********************/
                maimgRet = MAIMGDeleteByID(pCatData->stReferenceID);
                if (maimgRet.nStatus)
                {
                   printf( "\tMAIMGDeleteByID: Error!(%d) (%s)\n",
                             maimgRet.nStatus, maimgRet.szErrorMessage );
                }
                else
                {
                   printf("\tMAIMGDeleteByID: Record #%d in table %s has been
                             Successfully deleted !\n",
                   pCatData->stReferenceID.lRecordID,
                   pCatData->stReferenceID.szDatasetName);
                }

                n++;
                pImageDataLL = pImageDataLL->pNext;
             }

             /******************
              Free Linked List.
             ******************/
             maimgRet = MAIMGFreeLL(&CatList);
          }
       }

       /************************
        Call MAIMGDisconnect.
       ************************/
       maimgRet = MAIMGDisconnect();
    }
    exit( maimgRet.nStatus );

} /* End of main. */
```

4.1.3 Getting a Catalog of Imagery Products From the Database

The MAIMGCatalog method is used to retrieve a catalog of imagery products in the database that meet specific criteria. The criteria are contained in the input MAIMGQUERY structure. Most fields in this structure are optional; the only requirement is for the MAIMGGEOAREA element to be filled in with the area of interest boundaries or the MAIMG_NO_AOI constant. When the

area of interest boundaries are set to MAIMG_NO_AOI, only those images that have no associated area of interest will be considered. These may be set to global (latitude -90.0 to 90.0, longitude -180.0 to 180.0) if no specific geographic area is desired. MAIMGCatalog returns a linked list of MAIMGLINKEDLIST structures, which in turn point to the metadata for each record found that met the input criteria. It also returns the number of matching records found and the MAIMGRET status structure.

The linked list returned by MAIMGCatalog must be freed with a call to MAIMGFreeLL when no longer needed. See Section 4.1.2 for an example showing the use of the MAIMGCatalog method.

4.1.4 Retrieving Imagery Products From the Database by Query

The MAIMGGetByQuery method takes as input a MAIMGQUERY structure containing criteria for imagery products to be retrieved. It returns a linked list of MAIMGDATA structures containing the data for imagery products that matched the input criteria, the number of matching imagery products found, and a MAIMGRET return status structure. This method retrieves all imagery products matching the input criteria. The example given shows how a query using the AOI Name field works. Note that if MAIMGetByQuery is called, MAIMGFreeLL must be called afterwards to free the linked list returned, and then the head of the linked list must also be freed.

The code example below shows the use of MAIMGGetByQuery to retrieve imagery.

```
/*****************************************************************************
                            T   E   S   T   E   R

Purpose: Sample Code using MAIMGGetByQuery.
*****************************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "MAIMGAPI.h"

#define SECONDS_PER_DAY 24*60*60

#ifdef _WIN32
#define _MDBDLL
#endif

/*****************************************************************************
                                M   A   I   N
*****************************************************************************/
void main(argc, argv)
int argc;
char **argv;
{
```

```
FILE     *fp;
int      nStatus = 0;
long     lNumFound;
int      n,i;

MAIMGRET          maimgRet;
MAIMGQUERY        ImageQuery;
PMAIMGDATA        pImageData;
MAIMGLINKEDLIST   ImageList, *pImageDataLL;

maimgRet = MAIMGConnect();
if (maimgRet.nStatus)
{
   printf( "\ttester: Error calling MAIMGConnect (%d) (%s)\n",
             maimgRet.nStatus, maimgRet.szErrorMessage );
}
else
{
   printf( "\tMAIMGConnect : Successful!\n\n");

   /**************************************************
   Set up input Image Query.
   Get list of NOAA_9 Image Products
   with an AOI Name of “Gulf Of Mexico.”
   **************************************************/
   memset(&ImageQuery, 0, sizeof(MAIMGQUERY));

   ImageQuery.nType = MAIMG_NOAA_9;
   ImageQuery.nSubType = MAIMG_QUERY_WILDCARD;
   ImageQuery.eProjectionType =  MAIMG_QUERY_WILDCARD;
   ImageQuery.eFormatType =  MAIMG_QUERY_WILDCARD;

   strcpy(ImageQuery.szAOIName, “Gulf Of Mexico”);

   ImageQuery.nQualityIndicator  = MAIMG_QUERY_WILDCARD;
   ImageQuery.nDataCategory =  MAIMG_QUERY_WILDCARD;

   ImageQuery.lBeginBaseTime = 0;
   ImageQuery.lEndBaseTime = time(0);
   ImageQuery.lBeginReceiptTime = MAIMG_QUERY_WILDCARD;
   ImageQuery.lEndReceiptTime = MAIMG_QUERY_WILDCARD;

   ImageQuery.lChannelID = MAIMG_QUERY_WILDCARD;

   /*************************
    Call MAIMGCatalog.
   *************************/
   maimgRet = MAIMGGetByQuery(&ImageQuery, &lNumFound, &ImageList);

   if (maimgRet.nStatus)
   {
      printf( "\tMAIMGGetByQuery: Error!(%d) (%s)\n",
                maimgRet.nStatus, maimgRet.szErrorMessage );

   }
   else
   {
      printf("\tMAIMGGetByQuery: Successful !\n");
```

```
/*************************
 Display output values.
*************************/
if (lNumFound < 1)
{
   printf("\nNo catalog entries were found.\n");
}
else
{
   n = 1;
   pImageDataLL = &ImageList;
   printf("\nNumber Imagery entries were found %d.\n", lNumFound);

   while (n <= lNumFound)
   {
      pImageData = ( PMAIMGDATA )pImageDataLL->pData;

      printf("\n\tTableName = %s\n",
                  pImageData->stReferenceID.szDatasetName);
      printf("\tObjectID = %d\n",
               pImageData->stReferenceID.lRecordID);
      printf("\tImage Type = %d \n", pImageData->nType);
      printf("\tImage Sub Type = %d \n", pImageData->nSubType);
      printf("\tImage Format = %d \n",
               pImageData->eImageFormatType);
      printf("\tAOIName = %s \n", pImageData->szAOIName);
      printf("\tNorthLat = %f \n", pImageData->stGeoArea.rsNLat);
      printf("\tSouthLat = %f \n", pImageData->stGeoArea.rsSLat);
      printf("\tWestLon = %f \n", pImageData->stGeoArea.rsWLon);
      printf("\tEastLon = %f \n", pImageData->stGeoArea.rsELon);
      printf("\tProduct Title = %s \n", pImageData->szTitle);
      printf("\tProduct Description = %s \n",
               pImageData->szDescription);
      printf("\tQualityIndicator =%d \n",
               pImageData->nQualityIndicator);
      printf("\tDataCategory = %d \n", pImageData->nDataCategory);
      printf("\tClassification = %s \n",
               pImageData->szSecurityClass);
      printf("\tReceipt Method = %s \n",
               pImageData->szReceiptMethod);
      printf("\tCompression = %s \n", pImageData->szCompression);
      printf("\tOriginating Site = %s\n",
               pImageData->szOriginatingSite);
      printf("\tBase Time = %d \n",
               pImageData->lBaseTime);
      printf("\tReceipt Time = %d\n",
               pImageData->stImage.stSatellite.info.lReceiptTime);
      printf("\tSatelliteName = %s\n",
               pImageData->stImage.stSatellite.info.szSatelliteName);
      printf("\tCountryOrigin = %s\n",
               pImageData->stImage.stSatellite.info.szCountryOrigin);
      printf("\tLaunchDate = %s\n",
               pImageData->stImage.stSatellite.info.szLaunchDate);
      printf("\tResponsibleAgency = %s\n",
               pImageData->stImage.stSatellite.info.szResponsibleAgency);
      printf("\tSensorName = %s\n",
               pImageData->stImage.stSatellite.info.szSensorName);
```

```
                printf("\tChannelID = %d\n",
                         pImageData->stImage.stSatellite.nChannelID);
                printf("\tChannelName= %s\n",
                         pImageData->stImage.stSatellite.info.szChannelName);
                printf("\tChannelDescription= %s\n",
                         pImageData->stImage.stSatellite.info.szChannelDescription);
                printf("\tCalibrationTableSize = %d\n",
                         pImageData->stImage.stSatellite.nCalibrationTableSize);
                for (i=0; i<pImageData->stImage.stSatellite.nCalibrationTableSize;
                     i++)
                {
                   printf("\t\tCalibrationTable[%d] = %f\n",
                            i,pImageData->stImage.stSatellite.pCalibrationTable[i]);
                }
                printf("\tImage Byte Size = %d\n",pImageData->ulSize);

                /***************************
                 Write image data to file.
                ***************************/
                fp = fopen("image.dat", "wb");
                fwrite(pImageData->pData, pImageData->ulSize, 1, fp);
                fclose(fp);

                n++;
                pImageDataLL = pImageDataLL->pNext;
             }

             /*******************
              Free Linked List.
             *******************/
             maimgRet = MAIMGFreeLL(&ImageList);
          }
       }

       /************************
        Call MAIMGDisconnect.
       ************************/
       maimgRet = MAIMGDisconnect();

    }
    exit( maimgRet.nStatus );

} /* End of main. */
```

4.1.5 Retrieving Imagery Products from the Database By Reference (ID)

A second method for retrieving imagery data is MAIMGGetByID. This method takes as input the datasetname and record id of the image to be retrieved. It returns a single MAIMGDATA structure containing the retrieved data. It also returns a MAIMGRET return status structure. Since this method requires knowledge of the datasetname and record to be retrieved, a call to MAIMGCatalog is usually made first in order to get a list of images matching specific criteria. The MAIMGDATA structure returned contains a pointer to the imagery data. This pointer should be freed after use. If the imagery data are a satellite image, then the pointer to the

calibration table may also need to be freed. Lastly, the MAIMGDATA structure should be freed if dynamically allocated. See Section 4.1.7 for an example of the MAIMGGetByID method.

4.1.6 Retrieving Imagery Type and SubType Information

The MAIMGGetTypesInfo method is used to retrieve imagery type and subtype information. contained in the database. The input criteria are the type ID and the subtype ID. This method retrieves a linked list of MAIMGTYPESDATA structures containing the data for types and subtypes that matched the input criteria. It also returns the number of matching records and the MAIMGRET structure. Note that if MAIMGGetTypesInfo is called, MAIMGFreeLL must be called afterwards to free the linked list returned, and then the head of the liked list must also be freed.

The code example below shows the use of MAIMGGetTypesInfo to retrieve type and subtype information.

```
/*************************************************************************
                            T   E   S   T   E   R

Purpose: Sample Code using MAIMGGetTypesInfo.
*************************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "MAIMGAPI.h"

#define SECONDS_PER_DAY 24*60*60

#ifdef _WIN32
#define _MDBDLL
#endif

/*************************************************************************
                              M   A   I   N
*************************************************************************/
void main(argc, argv)
   int argc;
   char **argv;
{
   int      nStatus = 0;
   int      n;
   long     lNumFound;
   long     lTypeID;
   long     lSubTypeID;

   MAIMGRET            maimgRet;
   MAIMGLINKEDLIST     TypesList;
   PMAIMGLINKEDLIST    pTypesDataLL;
   PMAIMGTYPESDATA     pTypesData;
```

```
maimgRet = MAIMGConnect();

if (maimgRet.nStatus)
{
   printf( "\ttester: Error calling MAIMGConnect (%d) (%s)\n",
             maimgRet.nStatus, maimgRet.szErrorMessage );
}
else
{
   printf( "\tMAIMGConnect : Successful!\n\n");

   /****************************************/
   /* Set up input Types/SubTypes Query.  */
   /* Get list of all SubTypes associated */
   /* with the DMSP-F14 satellite         */
   /****************************************/

   lTypeID = MAIMG_DMSP_F14;
   lSubTypeID = MAIMG_QUERY_WILDCARD;
   /****************************/
   /* Call MAIMGGetTypesInfo. */
   /****************************/
   maimgRet = MAIMGGetTypesInfo( lTypeID, lSubTypeID,
              &lNumFound, &TypesList );

   if (maimgRet.nStatus)
   {
      printf( "\tMAIMGGetTypesInfo: Error!(%d) (%s)\n",
                maimgRet.nStatus, maimgRet.szErrorMessage );
   }
   else
   {
      printf("\tMAIMGGetTypesInfo: Successful !\n");

      /***************************/
      /* Display output values. */
      /***************************/
      if (lNumFound < 1)
      {
         printf("\nNo entries were found.\n");
      }
      else
      {
         n = 1;
         pTypesDataLL = &TypesList;
         printf("\nNumber of entries found %d.\n", lNumFound);

         while (n <= lNumFound)
         {
            pTypesData = (PMAIMGTYPESDATA) pTypesDataLL->pData;

            printf("TypeID        = %d\n", pTypesData->lTypeID);
            printf("Type Name     = %s\n", pTypesData->szTypeName);
            printf("SubTypeID     = %d\n", pTypesData->lSubTypeID);
            printf("SubType Name  = %s\n", pTypesData->szSubTypeName);
            printf("\n\n");

            n++;
            pTypesDataLL = pTypesDataLL->pNext;
```

```
            }

            /*********************/
            /* Free Linked List. */
            /*********************/
            maimgRet = MAIMGFreeLL(&TypesList);
         }
      }

      /*************************/
      /* Call MAIMGDisconnect. */
      /*************************/
      maimgRet = MAIMGDisconnect();

   }
   exit( maimgRet.nStatus );

} /* End of main. */
```

4.1.7 Updating Imagery Products in the Database

The MAIMGUpdateByID method is used to update imagery records contained in the database. This method requires the datasetname and the record id for the imagery record to be updated. Typically the MAIMGGetByID method is used to get the imagery record for update. Once the imagery record is extracted, any field in the record may be modified with the exception of type and subtype. The MAIMGUpdateByID call is made by passing in the object identifiers (datasetname, recordid) from the MAIMGGetByID call as well as the updated imagery record. Since base records cannot be modified, an update for an imagery record where nDataCategory is set to MAIMG_BASE is stored as a new imagery record. The new record will have its nDataCategory field set to MAIMG_EDITED.

```
/*****************************************************************************
                           T   E   S   T   E   R

Purpose: Sample Code using MAIMGUpdate.
*****************************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "MAIMGAPI.h"

#ifdef _WIN32
#define _MDBDLL
#endif

/*****************************************************************************
                              M   A   I   N
*****************************************************************************/
void main(argc, argv)
int argc;
```

```
char **argv;
{
   int         nStatus = 0;
   long        lNumFound;
   int         n;
   struct tm   timeptr;

   MAIMGRET            maimgRet;
   MAIMGQUERY          ImageQuery;
   MAIMGDATA           ImageData;
   MAIMGLINKEDLIST     CatList, *pImageDataLL;
   PMAIMGCATALOG       pCatData;

   memset(&timeptr,0,sizeof(struct tm));

   maimgRet = MAIMGConnect();
   if (maimgRet.nStatus)
   {
      printf( "\ttester: Error calling MAIMGConnect (%d) (%s)\n",
                 maimgRet.nStatus, maimgRet.szErrorMessage );
   }
   else
   {
      printf( "\tMAIMGConnect : Successful!\n\n");
      /**************************************************
       Set up input Catalog Query.
       Looking for a very specific product for update.
      **************************************************/
      memset(&ImageQuery, 0, sizeof(MAIMGQUERY));

      ImageQuery.nType = MAIMG_NOAA_9;
      ImageQuery.nSubType = MAIMG_NOAA_MSU;
      ImageQuery.eProjectionType = MAIMG_SPHERICAL;
      ImageQuery.eFormatType =  MAIMG_NITF;

      ImageQuery.stGeoArea.rsNLat  =   90.0;
      ImageQuery.stGeoArea.rsWLon  = -180.0;
      ImageQuery.stGeoArea.rsSLat  =  -90.0;
      ImageQuery.stGeoArea.rsELon  =  180.0;

      ImageQuery.nQualityIndicator  = 0;
      ImageQuery.nDataCategory =  MAIMG_BASE;

      /* Compute basetime (epoch) for Sept 1, 1994 */
      timeptr.tm_year = 94;
      timeptr.tm_mon  = 8;
      timeptr.tm_mday = 1;
      timeptr.tm_hour = 0;

      ImageQuery.lBeginBaseTime   = (long) mktime(&timeptr);
      ImageQuery.lEndBaseTime    = (long) mktime(&timeptr);;
      ImageQuery.lBeginReceiptTime = MAIMG_QUERY_WILDCARD;
      ImageQuery.lEndReceiptTime = MAIMG_QUERY_WILDCARD;

      ImageQuery.lChannelID = MAIMG_CHANNEL_1;

      /*************************
       Call MAIMGCatalog.
      *************************/
```

```
maimgRet = MAIMGCatalog(&ImageQuery, &lNumFound, &CatList);

if (maimgRet.nStatus)
{
   printf( "\tMAIMGCatalog: Error!(%d) (%s)\n",
              maimgRet.nStatus, maimgRet.szErrorMessage );

}
else
{
   printf("\tMAIMGCatalog: Successful !\n");

   /************************
    Display output values.
   ************************/
   if (lNumFound < 1)
   {
      printf("\nNo catalog entries were found.\n");
   }
   else
   {
      n = 1;
      pImageDataLL = &CatList;
      printf("\n\tNumber catalog entries were found %d.\n", lNumFound);

      if (lNumFound == 1)
      {
         pCatData = ( PMAIMGCATALOG )pImageDataLL->pData;

         printf("\n\tTableName = %s ObjectID = %d\n",
                    pCatData->stReferenceID.szDatasetName,
                    pCatData->stReferenceID.lRecordID);

         /***************************************
          Get the Imagery Record Data.
         ***************************************/
         maimgRet = MAIMGGetByID( pCatData->stReferenceID,
                    &ImageData );

         if (maimgRet.nStatus)
         {
            printf( "\tMAIMGGetByID: Error!(%d:%s) (%s)\n",
                      maimgRet.nStatus, maimgRet.szSQLState,
                      maimgRet.szErrorMessage );
         }
         else
         {
            printf("\tMAIMGGetByID: Successful !\n");

            /*********************************************
             Modify some of the Calibration Table
             points and then update the Imagery Record.
            *********************************************/

            ImageData.stImage.stSatellite.pCalibrationTable[0] = 5.6;
            ImageData.stImage.stSatellite.pCalibrationTable[4] = 55.9;

            maimgRet=MAIMGUpdateByID(pCatData->stReferenceID,
                      &ImageData);
```

```
                  /*******************
                   Check for errors.
                  *******************/
                  if (maimgRet.nStatus)
                  {
                     printf( "\tMAIMGUpdateByID: Error!(%d) (%s)\n",
                               maimgRet.nStatus,
                               maimgRet.szErrorMessage );
                  }
                  else
                  {
                     printf("\tMAIMGUpdateByID: Successful !\n");
                     printf("\tRecord Updated DatasetName=%s Record Id=%d \n",
                              ImageData.stReferenceID.szDatasetName,
                              ImageData.stReferenceID.lRecordID);
                  }

                  if (ImageData.pData)
                  {
                     free(ImageData.pData);
                  }
                  if (ImageData.stImage.stSatellite.pCalibrationTable)
                  {
                     free(ImageData.stImage.stSatellite.pCalibrationTable);
                  }
               }

               /*******************
                Free Linked List
               *******************/
               maimgRet = MAIMGFreeLL(&CatList);
            }
         }
      }

      /************************
       Call MAIMGDisconnect.
      ************************/
      maimgRet = MAIMGDisconnect();
   }
   exit( maimgRet.nStatus );

} /* End of main */
```

4.2 MDIMG Database Reference Tables

4.2.1 MDIMG Type and SubType Tables

The MDIMG Type and SubType tables contain the identifiers that may be used to specify a product type and subtype. Each type has one or more subtypes associated with it. The programmer must reference this table when setting the nTypeID and nSubTypeID fields in the MAIMGDATA, MAIMGCATALOG, MAIMGQUERY, and MAIMGTYPESDATA structures.

Table 4-1. MDIMGType/MDIMGSubType Join Table

Type ID	Type Name	SubType ID	SubType Name
1	Other Product	0	None
2	PRODUCT	16	Horizontal Weather Depiction
2	PRODUCT	17	Acoustic Sensor Prediction
2	PRODUCT	18	Atmospheric Sensor Prediction
2	PRODUCT	19	Textual Image
2	PRODUCT	20	Fax
2	PRODUCT	21	Briefing
2	PRODUCT	22	Mission Planning
2	PRODUCT	23	Screen Dump
3	GOES 8	1	IMAGER
3	GOES 8	2	SOUNDER
4	GOES 9	1	IMAGER
4	GOES 9	2	SOUNDER
5	GOES 10	1	IMAGER
5	GOES 10	2	SOUNDER
6	GMS 4	3	VISSR
7	GMS 5	3	VISSR

Table 4-1. MDIMGType/MDIMGSubType Join Table

Type ID	Type Name	SubType ID	SubType Name
8	GOMS 1N	4	STR
9	NOAA 9	5	TOVS-MSU
9	NOAA 9	6	TOVS-SSU
9	NOAA 9	7	TOVS-HIRS2
9	NOAA 9	8	AVHRR2
10	NOAA 10	5	TOVS-MSU
10	NOAA 10	6	TOVS-SSU
10	NOAA 10	7	TOVS-HIRS2
10	NOAA 10	8	AVHRR2
11	NOAA 11	5	TOVS-MSU
11	NOAA 11	6	TOVS-SSU
11	NOAA 11	7	TOVS-HIRS2
11	NOAA 11	8	AVHRR2
12	NOAA 12	5	TOVS-MSU
12	NOAA 12	6	TOVS-SSU
12	NOAA 12	7	TOVS-HIRS2
12	NOAA 12	8	AVHRR2

Table 4-1. MDIMGType/MDIMGSubType Join Table

Type ID	Type Name	SubType ID	SubType Name
13	NOAA 14	5	TOVS-MSU
13	NOAA 14	6	TOVS-SSU
13	NOAA 14	7	TOVS-HIRS2
13	NOAA 14	8	AVHRR2
14	NOAA 15	5	TOVS-MSU
14	NOAA 15	6	TOVS-SSU
14	NOAA 15	7	TOVS-HIRS2
14	NOAA 15	8	AVHRR2
15	METEOSAT 3	9	VISSR
16	METEOSAT 4	9	VISSR
17	METEOSAT 5	9	VISSR
18	METEOSAT 6	9	VISSR
19	METEOSAT 7	9	VISSR
20	INSAT 1B	10	VHRR
21	INSAT 2A	10	VHRR
22	INSAT 2B	10	VHRR
23	INSAT 2C	10	VHRR

Table 4-1. MDIMGType/MDIMGSubType Join Table

Type ID	Type Name	SubType ID	SubType Name
24	DMSP F8	11	OLS SMOOTH
24	DMSP F8	12	OLS FINE
24	DMSP F8	13	SSMI
24	DMSP F8	14	SSMT
25	DMSP F9	11	OLS SMOOTH
25	DMSP F9	12	OLS FINE
25	DMSP F9	13	SSMI
25	DMSP F9	14	SSMT
26	DMSP F10	11	OLS SMOOTH
26	DMSP F10	12	OLS FINE
26	DMSP F10	13	SSMI
26	DMSP F10	14	SSMT
27	DMSP F11	11	OLS SMOOTH
27	DMSP F11	12	OLS FINE
27	DMSP F11	13	SSMI
27	DMSP F11	14	SSMT
28	DMSP F12	11	OLS SMOOTH

Table 4-1. MDIMGType/MDIMGSubType Join Table

Type ID	Type Name	SubType ID	SubType Name
28	DMSP F12	12	OLS FINE
28	DMSP F12	13	SSMI
28	DMSP F12	14	SSMT
28	DMSP F12	15	SSMT2
29	DMSP F13	11	OLS SMOOTH
29	DMSP F13	12	OLS FINE
29	DMSP F13	13	SSMI
29	DMSP F13	14	SSMT
29	DMSP F13	15	SSMT2
30	DMSP F14	11	OLS SMOOTH
30	DMSP F14	12	OLS FINE
30	DMSP F14	13	SSMI
30	DMSP F14	14	SSMT
30	DMSP F14	15	SSMT2

4.3 Building Applications Using the MAIMG Libraries

This section contains four makefiles that can be used to build the MAIMG APIs. Static and dynamic makefiles are provided for the HP-UX environment and the Windows NT environment. The dynamic makefiles use the run-time libraries (*.sl) on HP-UX or (*.dll) on Windows NT to build the APIs. The static makefiles use the static libraries (*.a) on HP-UX, (*.lib) on Windows NT) to build the APIs.

4.3.1 Makefile Using the Dynamic MAIMG Libraries on HP-UX

```
PROGRAM = MAIMGIngestSample

CFILE = main.c

OFILE = $(CFILE:.c=.o)

RM = /bin/rm -f
INFORMIXDIR = /opt/informix

LIBPATHS = -L$(MAIMG_HOME)/bin -L$(INFORMIXDIR)/lib -L$(INFORMIXDIR)/lib/esql

INFORMIXLIBS = -lixsql -lixgen -lixos -lixasf -lixgls \
   $(INFORMIXDIR)/lib/esql/checkapi.o -lnsl_s -lm -lV3 -lcl -lsec

LIBS = \
   -lMAIMGAPI

INCLUDES = -I$(MAIMG_HOME)/include

all : $(PROGRAM)

CFLAGS = $(INCLUDES) -Aa
$(PROGRAM) : $(OFILE)
   $(CC) $(OFILE) \
   $(LIBPATHS) $(LIBS) $(INFORMIXLIBS) -lm -lc -o $(PROGRAM)

clean :
   $(RM) $(OFILE) $(PROGRAM) core
```

4.3.2 Makefile Using the Static MAIMG Libraries on HP-UX

```
PROGRAM = MAIMGIngestSample
CFILE = main.c

OFILE = $(CFILE:.c=.o)

RM = /bin/rm -f
INFORMIXDIR = /opt/informix

LIBPATHS = -L$(MAIMG_HOME)/lib -L$(INFORMIXDIR)/lib -L$(INFORMIXDIR)/lib/esql
```

```
INFORMIXLIBS = -lixsql -lixgen -lixos -lixasf -lixgls \
   $(INFORMIXDIR)/lib/esql/checkapi.o -lnsl_s -lm -lV3 -lcl -lsec

LIBS = \
   -lMAIMGAPI

INCLUDES = -I$(MAIMG_HOME)/include

all : $(PROGRAM)

CFLAGS = $(INCLUDES) -Aa

$(PROGRAM) : $(OFILE)
   $(CC) $(OFILE) \
$(LIBPATHS) $(LIBS) $(INFORMIXLIBS) -lm -lc -o $(PROGRAM)

clean :
   $(RM) $(OFILE) $(PROGRAM) core
```

4.3.3 Makefile Using the Dynamic MAIMG Libraries on Windows NT

```
PROGRAM = MAIMGIngestSample.exe

CFILE = main.c

OFILE = $(CFILE:.c=.obj)

MAIMG_HOME = ..\..\..\..
RM = -@del /Q /F /S
LD = link

LIBS = $(MAIMG_HOME)\bin\MAIMGAPI.lib

INFORMIXLIB = $(INFORMIXDIR)\lib\isqlt07c.lib

INCLUDES = -I$(MAIMG_HOME)\include

all : $(PROGRAM)

CFLAGS = $(INCLUDES) -MD

$(PROGRAM) : $(OFILE)
   $(LD) $(OFILE) \
   $(LIBS) $(INFORMIXLIB) -OUT:$@
clean :
   $(RM) $(OFILE) $(PROGRAM)
```

4.3.4 Makefile Using the Static MAIMG Libraries on Windows NT

```
PROGRAM = MAIMGIngestSample.exe

CFILE = main.c

OFILE = $(CFILE:.c=.obj)
```

```
MAIMG_HOME = ..\..\..\..
RM = -@del /Q /F /S
LD = link

LIBS = $(MAIMG_HOME)\lib\MAIMGAPI.lib

INFORMIXLIB = $(INFORMIXDIR)\lib\isqlt07c.lib

INCLUDES = -I$(MAIMG_HOME)\include

all : $(PROGRAM)

CFLAGS = $(INCLUDES)

$(PROGRAM) : $(OFILE)
   $(LD) /NODEFAULTLIB:LIBC.LIB /NODEFAULTLIB:LIBCMT.LIB $(OFILE) \
   $(LIBS) $(INFORMIXLIB) -OUT:$@
clean :
   $(RM) $(OFILE) $(PROGRAM)
```

5 CUSTOMIZING SEGMENTS

This section is tailored out.

(This page intentionally left blank.)

6 NOTES

6.1 Glossary of Acronyms

AESS	Allied Environmental Support System
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 Mobile Oceanographic Support System
JMCIS	Joint Maritime Command Information System
JMS	Joint METOC Segment
LLT	Latitude-Longitude-Time
MAIMG	Imagery API Segment of the TESS(NC) METOC Database
METOC	Meteorology and Oceanography
MIDDS	Meteorological Integrated Data Display System
NITES	Navy Integrated Tactical Environmental Subsystem
PC	Personal Computer
PM	Programming Manual
PS	Performance Specification
RDBMS	Relational Database Management System
SQL	Structured Query Language
TESS(NC)	Tactical Environmental Support System Next Century

(This page intentionally left blank.)