M-API Version 2.3

Programmer's Reference Guide

February 19, 1998

SDST-090C

M-API Version 2.3

Programmer's Reference Guide

Prepared By:

Frederick Shaw, SAIC/GSC
Utilities Programmer

Date

Change Record Page

This document is baselined and has been placed under Configuration Control. Any changes to this document will need the approval of the Configuration Control Board.

Document Title: M-API Version 2.0 Programmers Reference Guide

Document Date: December 26, 1996

Issue	Date	Page Affected	Description
Original	12/26/96	All	Baseline
Revision A	3/14/97	All	CCR #285
Revision B	12/1/97	All	CCR #311
Revision C	2/19/98	All	CCR #288

M-API Version 2.3
Programmer's Reference Guide

Table of Contents

1. INTRODUCTION 1-1

1.1 Purpose.....1-1
1.2 Organization of the M-API Reference Guide.....1-1

2. RELATED DOCUMENTS.....2-1

2.1 Hierarchical Data Format-Related Documents.....2-1
2.2 EOSDIS Core System-Related Documents.....2-1
2.3 Other Documentation.....2-2

3. M-API ROUTINE DESCRIPTIONS.....3-1

3.1 Standard Format for Routines.....3-1
3.2 Prototype Listing for C and FORTRAN Routines.....3-3
3.3 Individual Routines.....3-6
closeMODISfile (CLMFIL).....3-6
createMAPIfilehandle (CMFH).....3-7
completeMODISfile (CPMFIL).....3-9
createMODISarray (CRMAR).....3-12
createMODISgroup (CRMGRP).....3-15
createMODIStable (CRMTBL).....3-17
endMODISobjaccess (EMOBJ).....3-21
getMODISardims (GMARDM).....3-23
getMODISarinfo (GMARIN).....3-26
getMODISarray (GMAR).....3-30
getMODISdiminfo (GMDMIN).....3-33
getMODISdimname (GMDNAM).....3-38
getMODISECSinfo (GMECIN).....3-40
getMODISfields (GMFLDS).....3-45
getMODISfileinfo (GMFIN).....3-49
getMODIShobjid (GMHOID).....3-52
getMODIStable (GMTBL).....3-54
MODISsizeof (MSIZE).....3-58
openMODISfile (OPMFIL).....3-60
putMODISarinfo (PMARIN).....3-62
putMODISarray (PMAR).....3-66
putMODISdiminfo (PMDMIN).....3-69
putMODISdimname (PMDNAM).....3-73
putMODISfileinfo (PMFIN).....3-76
putMODIStable (PMTBL).....3-78
releaseMAPIfilehandle (RMFH).....3-81

substrMODISECSinfo (SMECIN).....3-82

4. M-API INTERNAL AND LOW LEVEL ROUTINE DESCRIPTIONS.....4-1

4.1 Prototype Listing for Internal Routines.....4-1
4.2 M-API Internal Routines.....4-3
cadmgrp.....4-3
cclmfil.....4-4
ccmfh.....4-5
ccpmfil.....4-6
ccrmar.....4-8
ccrmgrp.....4-9
ccrmtbl.....4-10
cemobj.....4-11
cgmar.....4-12
cgmardm.....4-13
cgmarin.....4-15
cgmdmin.....4-16
cgmdnam.....4-17
cgmecin.....4-18
cgmfin.....4-19
cgmflds.....4-20
cgmhoid.....4-21
cgmtbl.....4-22
copmfil.....4-23
cpmar.....4-24
cpmarin.....4-25
cpmdmin.....4-26
cpmdnam.....4-27
cpmfin.....4-28
cpmtbl.....4-29
crmfh.....4-30
4.3 Prototype Listing for Low-Level Routines.....4-31
4.4 Individual Low-Level Routines.....4-32
addid.....4-32
addMODISgroup (ADMGRP).....4-34
datatype_to_DFNT (DT2DF).....4-35
DFNT_to_datatype (DF2DT).....4-36
emptyVdata.....4-37
getMODISarrayid.....4-38
getMODIStableid.....4-39
MPVL2ODL.....4-40
MTYPEc2f.....4-41
MTYPEf2c.....4-43
MVSfind.....4-45
NULLstr.....4-46
parse_string.....4-47

SDS_footprintOK (SDSOK).....4-48
searchid.....4-49
searchMODISgroup (SRMGRP).....4-50
set_Vhasdata.....4-52
Vdatattr_name.....4-53
VFdatatypes.....4-54

APPENDIX A: M-API-SUPPLIED CONSTANTS AND MACROS A-1

APPENDIX B: M-API UTILITIES DATA DICTIONARY.....B-1

APPENDIX C: VARIABLES FOR ROUTINES..... C-1

List of Tables

Table 3-1. C Routine Prototype Listing.....3-3
Table 3-2. FORTRAN Routine Prototype Listing.....3-5
Table 4-1. FORTRAN to C Interface Routine Prototype Listing.....4-1
Table 4-2. Low-Level C and FORTRAN Routine Prototype Listing.....4-31
Table A-1. M-API Data Type Constants.....A-1
Table A-2. SDS Metadata Constants.....A-1
Table A-3. ECS Global Inventory Metadata Names.....A-2
Table A-4. Level 1A Macros.....A-4
Table A-5. L1B/Geolocation Macros.....A-6
Table A-6. Atmosphere Macros.....A-11
Table A-7. Ocean Macros.....A-15
Table A-8. Land Macros.....A-16
Table B-1. M-API Utilities Data Dictionary.....B-1
Table C-1. Variables for C Routines.....C-1
Table C-2. Variables for FORTRAN Routines.....C-6

(This page intentionally left blank.)

M-API Version 2.3
Programmer's Reference Guide

1. INTRODUCTION

This document describes the Version 2.3 Release of the MODIS-Applications Programming Interface (M-API) developed by the MODIS Science Data Support Team (SDST), for both reading and writing of science data sets associated with geolocation, Level 1A (L1A), Level 1B (L1B), land, ocean, and atmosphere algorithms. The intended audience includes both scientists and programmers building algorithm-based software as part of the MODIS Version 2 (V2) code delivery to the EOSDIS Core System (ECS). A knowledge of C or FORTRAN programming and a general knowledge of Hierarchical Data Format (HDF) would be helpful but is not required to start using M-API. This document should be used in conjunction with the M-API Version 2.3 User's Guide (SDST-064B). The goal of this document is to give users a source to refer to when they have an idea of what function to use but are not sure of the specifics. For example, what parameters are to be passed to the function or the order of the parameter.

1.1 Purpose

M-API is designed to greatly simplify the process of reading L1B radiance bands and Level 2 (L2) science data and for writing output products and metadata to the HDF files. The metadata consists of both a core set and a product-specific set using the standards enumerated by the ECS group. M-API shields the users as much as possible from the low-level details of HDF so they can focus on their science. At the same time, M-API allows a great deal of flexibility so that the users can customize the structures of the HDF output to meet their own research needs. M-API handles the generation of low-level objects within the HDF and organizes them, as efficiently as possible, to facilitate access to their contents by other L2, as well as Level 3 (L3), algorithms.

1.2 Organization of the M-API Reference Guide

This M-API Reference Guide is organized as follows:

- Section 1 provides the purpose and organization of this reference guide.
- Section 2 provides a list of related documentation.
- Section 3 describes the M-API routines.
- Section 4 describes the M-API FORTRAN to C interfaces and low-level routines.
- Appendix A lists the M-API-supplied constants and macros.
- Appendix B lists the data dictionary for the M-API utilities.
- Appendix C lists the variables for routines.

(This page intentionally left blank.)

2. RELATED DOCUMENTS

2.1 Hierarchical Data Format-Related Documents

- HDF User's Guide - An overview of HDF and how to create programs for reading and writing HDF files. M-API users new to HDF should read this document.
- HDF Reference Manual - Detailed descriptions of the HDF C and FORTRAN subroutines.
- HDF-EOS Library User's Guide for the ECS Project, Volumes 1 and 2 - The ECS view of what capabilities they will support, including descriptions of an HDF taxonomy including swaths, grids, point data, and metadata.

2.2 EOSDIS Core System-Related Documents

- SDP Toolkit User's Guide for the ECS Project - Descriptions of routines which are part of the EOS Core System's Science Data Processing (SDP) toolkit.
- ECS Science Software Delivery Procedures - The ECS view of how science software is to be delivered to the DAAC for integration into a production system.
- ECS Metadata Syntax: Granules - A description of the metadata ECS expects to see at the granule (i.e., HDF file) level. This can be used as a reference when formulating the metadata as global attributes in an HDF file.
- Global Change Master Directory (GCMD) Data Interchange Format (DIF) manual - This is referenced by the ECS Metadata Syntax: Granules document as defining the valid values for each of the metadata items.
- PVL-ODL User's Guide - Descriptions of the Parameter Value Language (PVL) and Object Description Language (ODL) that are the basis for the format of the metadata items to be included in the HDF files.

2.3 Other Documentation

- MODIS-Applications Programming Interface User's Guide (SDST-064B). The document describing the Version 2.3 release of the M-API, developed by the MODIS SDST, for both reading and writing of science data sets associated with Geolocation, L1A, L1B, land, ocean, and atmosphere algorithms.
- MODIS Software Development Standards and Guidelines (SDST-022C) - The document describing SDST's standards.
- World Wide Web (WWW) Uniform Resource Locators (URLs):
  - http://edhs1.gsfc.nasa.gov - ECS Data Handling System (on-line repository of ECS documents).
  - http://ltpwww.gsfc.nasa.gov/MODIS/MODIS.html - MODIS home page.
  - http://hdf.ncsa.uiuc.edu/index.html - HDF home page.
  - http://hdf.ncsa.uiuc.edu/doc.html - HDF documentation
  - http://ltpwww.gsfc.nasa.gov/MODIS/SDST/Home.html - SDST home page.
  - http://ltpwww.gsfc.nasa.gov - Laboratory for Terrestrial Physics home page.
  - http://newsroom.hitc.com/sdptoolkit/toolkit.html - SDP Toolkit home page.

3. M-API ROUTINE DESCRIPTIONS

This section contains a listing of every function call available in the M-API software library. The routines are listed alphabetically.

3.1 Standard Format for Routines

Each of the routines in the Version 2.3 release of the M-API is described using a standard format. The template for the standard format for routines is shown below, along with an explanation of the information given in each section.

C Function Name (FORTRAN Name)

Purpose: What the routine does.

Description: Details the use and behavior of the routine. This includes the conditions required to use the routine and references to other routines closely associated with it.

C: `return_type function_name(type parameter1, type parameter2,...)`

Input Parameters:

parameter1 Definition of first parameter. Brief discussion of possible inputs or outputs.
parameter2 Definition of second parameter, etc.

Output Parameters:

parameter1 Definition of first parameter. Brief discussion of possible inputs or outputs.
parameter2 Definition of second parameter, etc.

FORTRAN:

```
INTEGER FUNCTION NAME (parameter1, parameter2......)
      TYPE  parameter list
      TYPE  parameter list
```

Input Parameters:

parameter1 Definition of first parameter.
parameter2 Definition of second parameter, etc.

Output Parameters:

parameter1 Definition of first parameter.
parameter2 Definition of second parameter, etc.

Returns: Description of return value. This is often, or has as part of its content, an error state indicator.

Error Messages:	**Description:**
Error: `function_name`	(descriptive text)

All M-API generated messages are written to the Log Status file. The ERROR prefix indicates that the condition prevented the routine from performing its task and that an error condition will be returned. A discussion of what circumstances will evoke an error message is included where appropriate.

Warning: `function_name`	(descriptive text)

WARNING messages will occur in unusual situations where M-API is capable of completing its task but the result may be considered undesirable.

3.2 Prototype Listing for C and FORTRAN Routines

Table 3-1 is a prototype listing of the C routines and Table 3-2 is a prototype listing of the FORTRAN routines.

Table 3-1. C Routine Prototype Listing

Return Type	Routine Name [Parameter(s)]
int	closeMODISfile (MODFIL **file)
int	completeMODISfile (MODFIL **file, PGSt_MET_all_handles mdHandles, ECSattr_names_for_all_handles HDFattrNames, long int NumHandles)
MODFIL	createMAPIfilehandle (int32 fid)
int	createMODISarray (MODFIL *file, char *arrayname, char*groupname, char *data_type, long int rank, long int dimsizes[])
int	createMODISgroup (MODFIL *file, char *groupname, char *classname)
int	createMODIStable (MODFIL *file, char *tablename, char *classname, char*groupname, char*fieldname, char*data_type)
int	endMODISobjaccess (MODFIL *file, char *name, char *group, long int type)
int	getMODISardims (MODFIL *file, char *arrayname, char *groupname, char *data_type, long int *rank, long int dimsizes[])
int	getMODISarinfo (MODFIL *file, char *arrayname, char *groupname, char *attribute, char *data_type, long int *n_elements, void *value)
int	getMODISarray (MODFIL *file, char *arrayname, char *groupname, long int start[], long int dimsizes[], void *data)
int	getMODISdiminfo (MODFIL *file, char *arrayname, char *groupname, long int dimension, char *attribute, char *data_type, long int *n_elements, void *value)
int	getMODISdimname (MODFIL *file, char *arrayname, char *groupname, long int dimension, char *dimname)
int	getMODISECSinfo (MODFIL *file, char *PVLAttrName, char *parmName, char *data_type, long int *n_elements, void *value)
int	getMODISfields (MODFIL *file, char *tablename, char *groupname, long int *stringlen, long int *recno, long int *fieldno, char *fieldname, char *data_type, char *classname)
int	getMODISfileinfo (MODFIL *file, char *attribute, char *data_type, long int *n_elements, void *value)
int	getMODIShobjid (MODFIL *file, char *name, char *group, long int type, long int *n_elements, void * value
int	getMODIStable (MODFIL *file, char *tablename, char *groupname, char*fieldname, long int start, long int recno, long int *buffsize, unsigned char *data)
long int	MODISsizeof (char *data_type)
MODFIL*	openMODISfile (char *filename, char *access)

Return Type	Routine Name [Parameter(s)]
int	putMODISarinfo (MODFIL **file*, char **arrayname*, char **groupname*, char **attribute*, char **data_type*, long int *n_elements,* void **value*)
int	putMODISarray (MODFIL **file*, char **arrayname,* char **groupname*, long int *start[],* long int *dimsizes[]*, void **data*)
int	putMODISdiminfo (MODFIL **file*, char **arrayname*, char **groupname*, long int *dimension*, char **attribute*, char **data_type*, long int *n_elements,* void **value*)
int	putMODISdimname(MODFIL **file*, char **arrayname*, char **groupname*, long int *dimension,* char **dimname*)
int	putMODISfileinfo (MODFIL **file*, char **attribute*, char **data_type*, long int *n_elements,* void **value*)
int	putMODIStable (MODFIL **file*, char **tablename,* char **groupname*, long int *start,* long int *recno,* unsigned char **data*)
int	releaseMAPIfilehandle (MODFIL ** *file*)
int	substrMODISECSinfo (*char *char_value*, long int *n_elements,* long int **n_strings,* char **substr[]*)

Table 3-2. FORTRAN Routine Prototype Listing

Return Type	Routine Name [Parameter(s)]
INTEGER FUNCTION	CLMFIL (*modfil*)
INTEGER FUNCTION	CMFH (*swfid*, *modfil*)
INTEGER FUNCTION	CPMFIL (*modfil*, *mdHandles*, *hdfattrnms*, *numhandles*)
INTEGER FUNCTION	CRMAR (*modfil, arrnm, group, dtype, rank, dims*)
INTEGER FUNCTION	CRMGRP (*modfil, group, clsnm*)
INTEGER FUNCTION	CRMTBL (*modfil*, *tblnm*, *class*, *group*, *field*, *dtype*)
INTEGER FUNCTION	EMOBJ (*modfil*, *name*, *group*, *type*)
INTEGER FUNCTION	GMAR (*modfil, arrnm, group, start, dims, data)*
INTEGER FUNCTION	GMARDM (*modfil, arrnm, group, dtype, rank, dims)*
INTEGER FUNCTION	GMARIN (*modfil*, *arrnm*, *group*, *attr, dtype*, *nms, value*)
INTEGER FUNCTION	GMDMIN (*modfil*, *arrnm*, *group*, *dim, attr, dtype*, *nms, value*)
INTEGER FUNCTION	GMDNAM (*modfil*, *arrnm*, *group*, *dim, dname*)
INTEGER FUNCTION	GMECIN (*modfil, pvlname, pname*, *nms, dtype, pvalue*)
INTEGER FUNCTION	GMFIN (*modfil*, *attr, dtype*, *nms, value*)
INTEGER FUNCTION	GMFLDS *(modfil, tblnm, group, strln, recno, fldno, fldnm, dtype, clss)*
INTEGER FUNCTION	GMHOID (*modfil, name, group, type, access)*
INTEGER FUNCTION	GMTBL (*modfil*, *tblnm*, *group, field, start, recno, bsize, data*)
INTEGER FUNCTION	MSIZE (*dtype*)
INTEGER FUNCTION	OPMFIL (*fname*, *access*, *modfil*)
INTEGER FUNCTION	PMAR (*modfil, arrnm, group, start, dims, data)*
INTEGER FUNCTION	PMARIN (*modfil*, *arrnm*, *group*, *attr*, *dtype*, *nms, value*)
INTEGER FUNCTION	PMDMIN (*modfil*, *arrnm*, *group*, *dim*, *attr*, *dtype*, *nms, value*)
INTEGER FUNCTION	PMDNAM (*modfil*, *arrnm*, *group*, *dim, dnm*)
INTEGER FUNCTION	PMFIN (*modfil*, *attr*, *dtype*, *nms, value*)
INTEGER FUNCTION	PMTBL (*modfil, tblnm, group, start, recno, data*)
INTEGER FUNCTION	RMFH (*modfil*)
INTEGER FUNCTION	SMECIN (*cvalue*, *nelmnt*, *nstrs*, *substr*)
INTEGER FUNCTION	SRMGRP (*modfil*, *group, clsnm, objnm*, *objcls*, *objtyp*)

3.3 Individual Routines

The following are explanations for each individual routine.

closeMODISfile (CLMFIL)

Purpose: Closes a file accessed by the M-API routines.

Description: closeMODISfile (CLMFIL) terminates the access of M-API routines to a MODIS HDF file opened using openMODIS file (OPMFIL). Only pre-existing files should be closed by closeMODISfile. completeMODISfile (CPMFIL) should be used to end access to a new MODIS HDF file so that the file's header information can be completed. closeMODISfile may fail to close the file if an error occurs.

C: `int closeMODISfile(MODFIL **file)`

Input Parameters:

file IN/OUT: Pointer to MODFIL structure address used to reference a file in all M-API routines. *file* is set to NULL when the file is closed.

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION CLMFIL (modfil)
      INTEGER     modfil(MODFILLEN)
```

Input Parameters:

modfil IN: Array that is used to reference a MODIS HDF file in all other M-API routines

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
`closeMODISfile cannot close a null file.`	
`closeMODISfile detected FAIL from HDF function Sdend. Unable to close` *`filename`*`.`	File could not be closed because access identifiers to data objects are still attached to the file. Changes to the file may be lost.
`WARNING: closeMODISfile closed new file` *`filename`* `without complete header information.`	The file has been successfully closed, but completeMODISfile should be used instead so that the required file header information will be included.

createMAPIfilehandle (CMFH)

Purpose: Creates a M-API file ID handle from an HDF-EOS file ID handle.

Description: createMAPIfilehandle (CMFH) provides a M-API filehandle for a previously opened HDF-EOS file (swath, grid, or point). A routine called releaseMAPIfilehandle must be called for each call to this routine or a memory leak will occur. NOTE: This file handle should not be passed to M-API routines closeMODISfile or completeMODISfile.

Users may open an HDF-EOS file using a HDF-EOS open routine; call createMAPIfilehandle to create the M-API filehandle; then use M-API routines to access data object(s) in the opened file. After access to the objects is finished, releaseMAPIfilehandle must be called to release the M-API filehandle. Also, this routine automatically closes all the objects opened with M-API routines. The last step is to call the HDF-EOS closing routines which close the file.

C: `MODFIL createMAPIfilehandle (int32 fid)`

Input Parameters:

fid IN: HDF-EOS file ID.

Output Parameters: N/A

Returns: Address of MODFILE structure that is used to reference the file in all other M-API C routines, or NULL if an error occurs.

FORTRAN:

```
INTEGER FUNCTION CMFH (modfil, swfid)
      INTEGER*4   swfid
      INTEGER     modfil(MODFILLEN)
```

Input Parameters:

swfid IN: HDF-EOS file ID handle.

Output Parameters:

modfil OUT: Array that is used to reference the file in all other M-API routines. The array will contain all zeroes if an error occurs.

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
`createMAPIfilehandle detected FAIL from HDF-EOS procedure Ehidinfo`	Possibly invalid HDF-EOS filehandle, SD interface ID, or HDF filehandle ID.
`createMAPIfilehandle detected FAIL from HDF procedure Hfidinquire while`	Routine could not retrieve the HDF-EOS

Error Message:	Description:
`retrieving filename and access mode`	filename or the access mode.
`createMAPIfilehandle is unable to allocate memory for a MODIS file structure for file` *`filename`*	The routine is unable to allocate enough memory for the M-API filehandle structure.
`createMAPIfilehandle is unable to allocate memory for the MODIS-HDF file structure's filename` *`filename`*	The routine is unable to allocate enough memory to hold the char array for the M-API filename.

completeMODISfile (CPMFIL)

Purpose: Closes a new file created and accessed by the M-API routines.

Description: completeMODISfile (CPMFIL) terminates the access of M-API routines to a MODIS HDF file created using openMODISfile (OPMFIL). In addition to closing the file, users can insert ECS PVL metadata text blocks created by PGS metadata tool kit into the MODIS file as global attributes.

A pre-existing MODIS HDF file should be closed by closeMODISfile (CLMFIL) or some of its header information will be overwritten. completeMODISfile may fail to close the file if an error occurs.

C:

```
int completeMODISfile(MODFIL **file, PGSt_MET_all_handles mdHandles,
ECSattr_names_for_all_handles HDFattrNames, long int NumHandles )
```

Input Parameters:

file IN: Address of MODFIL structure used to reference a file in all M-API routines.

mdHandles IN: A character array with size of [PGSd_MET_NUM_OF_GROUPS] [PGSd_MET_GROUP_NAME_L], where PGSd_MET_NUM_OF_GROUPS is 20 and PGSd_MET_GROUP_NAME_L is 50. This array is typedef-ed as PGSt_MET_all_handles. Each row in the array stores a handle to an internal ODL tree structure which will be written out as an ECS PVL attribute. Each handle, which is a string, should be less than 50 characters and occupy one row in the array. Therefore, the maximum number of handles should be 20.

HDFattrNames IN: A character array with size of [PGSd_MET_NUM_OF_GROUPS] [MAX_ECS_NAME_L], where PGSd_MET_NUM_OF_GROUPS is 20 and MAX_ECS_NAME_L is 256. This array is typedef-ed as ECSattr_names_for_all_handles. Each row in this array is a character string used as a global attribute name for storing an ECS PVL text block which has a handle in the corresponding row in *mdHandles* array. Each name, which is a string, should be less than MAX_ECS_NAME_L characters and occupies one row in the array.

NumHandles IN: Specifies the number of actual handles contained in mdHandles. This may be set from 0 to PGSd_MET_NUM_OF_GROUPS-1.
NOTE: No metadata will be written if this value is set to zero.

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION CPMFIL(modfil, mdHandles, HDFattrnms, NumHandles)
      INTEGER     modfil(MODFILLEN)
      CHARACTER(PGSd_MET_GROUP_NAME_L -1)
            mdHandles(PGSd_MET_NUM_OF_GROUPS)
      CHARACTER(MAX_ECS_NAME_L -1)
            HDFattrnms(PGSd_MET_NUM_OF_GROUPS)
      INTEGER NumHandles
```

Input Parameters:

modfil IN: array to reference a file in all M-API routines.

mdHandles IN: An array of character strings. The memory size of the array is PGSd_MET_NUM_OF_GROUPS * PGSd_MET_GROUP_NAME_L bytes where PGSd_MET_NUM_OF_GROUPS is 20 and PGSd_MET_GROUP_NAME_L is 50. Each string in the array stores a handle (ASCII name) to an internal ODL tree structure which will be written out as an HDF-EOS PVL global attribute. Each handle should be less than PGSd_MET_GROUP_NAME_L-1 characters. The maximum number of handles should be 20.

HDFattrnms IN: An array of character strings. The memory size of the array is PGSd_MET_NUM_OF_GROUPS * MAX_ECS_NAME_L bytes, where PGSd_MET_NUM_OF_GROUPS is 20 and MAX_ECS_NAME_L is 256. Each string in the array is a global attribute name for storing an HDF-EOS PVL text block which has a handle in the corresponding string in *mdHandles* array. Each name should be less than MAX_ECS_NAME_L-1 characters.

NumHandles IN: Specifies the number of actual handles contained in *mdHandles*. This may be set from 0 to PGSd_MET_NUM_OF_GROUPS-1.

NOTE: No metadata will be written if this value is set to zero.

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
`completeMODISfile unable to continue with empty input.`	
`closeMODISfile detected FAIL from HDF function Hclose. Unable to close file.`	
`closeMODISfile detected FAIL from HDF function Sdend. Unable to close file.`	
`completeMODISfile detected FAIL from HDF procedure Hclose. Unable to close file.`	File could not be closed because access identifiers to data objects are still attached to the file. Changes to the file may be lost.
`WARNING: completeMODISfile revised header data of pre-existing filename file.`	The file has been successfully closed, but closeMODISfile should be used instead to prevent modification to the MODIS HDF file's metadata.
`WARNING: completeMODISfile unable to revise header data of filename file open for read-only.`	The file has been successfully closed and was accessed only for reading.

createMODISarray (CRMAR)

Purpose: Initializes an array structure in a MODIS HDF file to store a multi-dimensional array of data.

Description: createMODISarray (CRMAR) creates an HDF SDS structure to store a new data array into a MODIS HDF file. It must be called before the data may be written to the file using putMODISarray (PMAR). The associated attributes for the whole array or each array dimension may be stored using putMODISarinfo (PMARIN) or putMODISdiminfo (PMDMIN).

The *groupname* string provides the facility to place the new array in an HDF 'Vgroup' data group. If a Vgroup with the name *groupname* does not exist, the array structure will not be created. The array may be placed in the file outside of any Vgroup by replacing *groupname* with NULL in C or *group* with a blank string (' ') in FORTRAN.

If an array with the name *arrayname* is written outside of a Vgroup, it must not already exist in the file. Arrays with the same name may be stored in the same file, however, if they are placed in separate Vgroups.

C:
```
int createMODISarray(MODFIL *file, char *arrayname, char *groupname,
char *data_type,  long int rank, long int dimsizes[])
```

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file receiving the new array.
arrayname	IN:	ASCII string that will be the name of the array.
groupname	IN:	ASCII string name of the data group where the new array will be placed. If set to NULL, the array will not be placed in a data group.
data_type	IN:	Data type of the array.

Permitted C data types:

```
"int8"
"uint8
"int16"
"uint16"
"int32"
"uint32"
"float32"
"float64"
```

rank	IN:	The number of dimensions in the array.
dimsizes	IN:	The size of each array dimension.

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION CRMAR (modfil, arrnm, group, dtype, rank, dims)
      INTEGER           modfil(MODFILLEN), rank, dims(*)
      CHARACTER*(*)     arrnm, group, dtype
```

Input Parameters:

modfil OUT: Array that is used to reference a MODIS HDF file where the new array structure will be installed.

arrnm IN: ASCII string that will be the name of the array.

group IN: ASCII string name of the data group where the new array will be placed. If set to ' ' (blank), the array will not be placed in a data group.

dtype IN: Data type of the array.

Recommended FORTRAN data types:

```
'CHARACTER*(*)'
'INTEGER*1'
'UINTEGER*1'
'INTEGER*2'
'UINTEGER*2'
'INTEGER*4'
'UINTEGER*4'
'INTEGER*8'
'UINTEGER*8'
'REAL*4'
'REAL*8'
```

rank IN: The number of dimensions in the array.

dims IN: The size of each array dimension.

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message: **Description:**

createMODISarray unable to make a new *arrayname* array with a NULL file MODFIL structure.

createMODISarray unable to make a new array without an array name input.

createMODISarray unable to make a new *arrayname* array without array dimension input.

createMODISarray unable to make a new *arrayname* array without array data type input.

createMODISarray unable to make a new *arrayname* array in file opened for read only.

createMODISarray found *arrayname* array already exists.

createMODISarray found *arrayname* array already exists in data group "*groupname*".

createMODISarray unable to find data group *groupname* to place new *arrayname* array in.

createMODISarray unable to create *arrayname* array of data type *data_type.*

createMODISarray unable to create *arrayname* array with *rank* dimensions.

createMODISarray detected FAIL from HDF procedure SDcreate attempting to create *arrayname* array.

createMODISarray detected FAIL from HDF procedure Sdend access while attempting to create *arrayname* array.

createMODISarray detected FAIL from HDF procedure Vattach attempting to create *arrayname* array.

createMODISarray detected FAIL from HDF procedure Vaddtagref attempting to create *arrayname* array.

createMODISarray detected FAIL from HDF procedure Vdetach attempting to create *arrayname* array.

createMODISgroup (CRMGRP)

Purpose: Creates a group structure in a MODIS HDF file to store data objects in.

Description: createMODISgroup (CRMGRP) creates an HDF Vgroup structure in a MODIS HDF file to store table and array structures. It must be called before any of the data objects to be aggregated in it are created. The use of data groups is optional, but data objects stored in them are documented in Appendix F, Examples of MODIS Data Product File Definitions, of the M-API User's Guide. A data group with the name *groupname* must be unique in a file.

C:
```
int createMODISgroup(MODFIL *file,  char *groupname, char *classname)
```

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file receiving the new group.
groupname	IN:	ASCII string that will be the name of the data group.
classname	IN:	(Optional) additional ASCII string that will be the class name of the group. Set to NULL for default.

Output Parameters: N/A

FORTRAN:
```
INTEGER FUNCTION CRMGRP (modfil, group, clsnm)
      INTEGER*4          modfil(3)
      CHARACTER*(*)      group, clsnm
```

Input Parameters:

modfil	IN:	Array that is used to reference the MODIS HDF file where the new data group structure will be installed.
group	IN:	ASCII string that will be the name of the data group.
clsnm	IN:	(Optional) additional ASCII string that will be the class name of the group. Set to a blank string ' ' (blank) for default.

Output Parameters: N/A

Returns: MAPIOK if successful, otherwise MFAIL.

Error Message:	Description:
`createMODISgroup unable to continue with an empty group name.`	
`createMODISgroup unable to continue to make a new data group` *`groupname`* `with an empty file pointer.`	
`createMODISgroup unable to make a new data group` *`groupname`* `in file opened for read only.`	
`createMODISgroup found group` *`groupname`* `already exists`	
`createMODISgroup detected FAIL from HDF procedure Vattach while creating group` *`groupname`*`.`	

createMODIStable (CRMTBL)

Purpose: Initializes a table structure in a MODIS HDF file to store a multi-column table of data.

Description: createMODIStable (CRMTBL) creates an HDF Vdata structure in a MODIS HDF file to store a new data table. It must be called before the data may be written to the file using putMODIStable (PMTBL). The text headers for each field (column) and the data type stored in each field must be provided.

The *groupname* string provides the facility to place the new table in an HDF 'Vgroup' data group. If a Vgroup with the name *groupname* does not exist, the table structure will not be created. The table may be placed in the file outside of any Vgroup by setting *groupname* = NULL in C and *group* = ' ' (blank) in FORTRAN.

An empty Vdata may not be created, so a dummy record is inserted into it. This dummy record will be overwritten with the first call from putMODIStable (PMTBL).

createMODIStable (CRMTBL) makes extensive use of dynamically allocated memory. All of this memory is freed, however, prior to exiting the routine.

If a table with the name *tablename* is created outside of a Vgroup, it must not already exist in the file. Tables with the same name may be stored in the same file, however, if they are placed in separate Vgroups.

C:
```
int createMODIStable(MODFIL *file,   char *tablename, char *classname,
char *groupname, char*fieldname, char*data_type)
```

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file receiving the new table.
tablename	IN:	ASCII string that will be the name of the table.
classname	IN:	ASCII string that will be the class name of the table. If set to NULL or an empty string, the table will have no class name.
groupname	IN:	ASCII string name of the data group where the new table will be placed. If set to NULL, the table will not be placed in a data group.
fieldname	IN:	Array of comma-delimited ASCII string table headers. The headers should be in the same order that the data for each table row will subsequently be written.

data_type IN: Array of comma-delimited data types for each table field. The data type strings should be in the same order that the data for each table row will subsequently be written.

Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
 "char *"
```

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION CRMTBL (modfil, tblnm, class, group, field,
dtype)
      INTEGER modfil(3)
      CHARACTER*(*) tblnm, class, group, field, dtype
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file receiving the new table.

tblnm IN: ASCII string that will be the name of the table.

class IN: ASCII string that will be the class name of the table. If set to an empty string, the table will have no class name.

group IN: ASCII string name of the data group where the new table will be placed. If set to a empty string, the table will not be placed in a data group.

field IN: Comma-delimited ASCII string table headers. The headers should be in the same order that the data for each table row will subsequently be written.

dtype IN: String of comma-delimited data types for each table field. The data type strings should be in the same order that the data for each table row will subsequently be written.

Permitted FORTRAN data types:

```
'CHARACTER*(*)'
 'INTEGER*1'
 'UINTEGER*1'
 'INTEGER*2'
 'UINTEGER*2'
 'INTEGER*4'
 'UINTEGER*4'
 'INTEGER*8'
```

'UINTEGER*8'
'REAL*4'
'REAL*8'

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
`createMODIStable unable to make a new table without a table name input.`	
`createMODIStable unable to make a new` *`tablename`* `table with a NULL file MODFIL structure.`	
`createMODIStable unable to make a new` *`tablename`* `table without field names input.`	
`createMODIStable unable to make a new` *`tablename`* `table without field data types input.`	
`createMODIStable unable to make a new` *`tablename`* `table in file opened for read only.`	
`createMODIStable found the` *`tablename`* `table already exists.`	
`createMODIStable unable to create` *`tablename`* `table with # byte records.`	Vdata table records are limited to 32K each.
`createMODIStable unable to create` *`tablename`* `table with` *`data_type`* `data types.`	
`createMODIStable unable to allocate memory for` *`fieldname`* `temporary buffer used to create the` *`tablename`* `table.`	
`createMODIStable unable to allocate memory for` *`data_type`* `temporary buffer used to create` *`tablename`* `table.`	
`createMODIStable found the` *`tablename`* `table to have no fields in the fieldname string` *`fieldname.`*	
`createMODIStable unable to support the creation of # fields in the field name string "fieldname" for the "tablename" table.`	Vdata table records are limited to 256 fields

Error Message:	Description:
`createMODIStable found the` *`tablename`* `table to have # data types in the data type string` *`data_type`* `instead of #.`	One data type must be supplied for each field in the Vdata table.
`createMODIStable detected FAIL from HDF procedure VSattach attempting to create the` *`tablename`* `table.`	
`createMODIStable detected fail from HDF procedure VSfdefine for` *`fieldname`* `and` *`data_type`* `of the` *`tablename`* `table.`	
`createMODIStable detected FAIL from HDF procedure VSsetfields creating the` *`tablename`* `table.`	
`createMODIStable unable to allocate memory for dummy field buffer used to create the` *`tablename`* `table.`	
`createMODIStable detected FAIL from HDF procedure VSwrite creating the` *`tablename`* `table.`	
`createMODIStable detected FAIL from HDF procedure Vattach attempting to create the` *`tablename`* `table.`	
`createMODIStable detected FAIL from HDF procedure Vaddtagref attempting to create the` *`tablename`* `table.`	
`createMODIStable detected FAIL from HDF procedure Vdetach attempting to create the` *`tablename`* `table.`	

endMODISobjaccess (EMOBJ)

Purpose: This routine ends the access to an individual or a group of opened objects (close the object and release resources).

Description: A Vdata or SDS is opened and remains opened when application programs call any of M-API routines to access the object. The M-API access routines automatically call M-API internal routine addid to insert the HDF object ID and related information into the ring super structure in MODFIL. In the subsequent calls to access the same object, M-API routines will use the ID stored in the ring super structure for fast data access.

endMODISobjaccess ends the access to an individual or a group of opened objects by deleting each object's DATAID structure from the ring super structure, releasing memory, and detaching (for Vdata) or end-accessing (for SDS) the objects. This routine is called by closeMODISfile or completeMODISfile, which calls closeMODISfile, so that all opened objects will be closed automatically before the MODIS HDF file is closed. As long as an application program calls closeMODISfile or completeMODISfile, the application does not need to worry when or how to use this routine to close an object or a group of objects. However, if an application program determines an object will no longer be accessed and needs to end the access to the object to release computer resources, the application program can call this routine any time before closing the opened HDF file.

C: `int endMODISobjaccess (MODFIL *file, char *name, char *group, long int type)`

Input parameters:

file IN: Address of MODFIL structure that is used to reference a MODIS HDF file containing objects to be closed (Vdata or SDS).

name IN: The name of the object. If the name is set to NULL, all objects matched with *group* and object *type* will be closed.

group IN: The name of the group to which objects belongs. If *group* is set to NULL, all lone objects (objects belonging to no group) matched with *name* and *type* will be closed. If both *name* and *group* are NULL, all objects matched with *type* will be closed.

type IN: The object type: MODIS_ARRAY (for SDS, the numerical value is 720, the same value as DFTAG_NDG), MODIS_TABLE(for Vdata, the numerical value is 1962, the same value as DFTAG_VH), or MODIS_ALL_TYPES (the numerical values is 0). If *type* is 720, only SDS objects will be closed. If *type* is 1962, only Vdata will be closed. If *type*

is MODIS_ALL_TYPES, both Vdata and SDS objects specified by *name* and *group* will be closed. Therefore, to close all opened objects, set both *name* and *group* to NULL and set *type* to 0.

Output parameter: N/A

FORTRAN:

```
INTEGER FUNCTION EMOBJ(modfil, name, group, type)
INTEGER modfil(MODFILLEN), type
CHARACTER *(*) name, group
```

Input parameters:

modfil IN: Array that is used to refer to a MODIS HDF file containing objects to be closed (Vdata or SDS).

name IN: The name of the object. If the name is set to ' ' (blank), all objects matched with *group* and object *type* will be closed.

group IN: The name of the group to which objects belongs. If *group* is set to ' ' (blank), all lone objects (objects belonging to no group) matched with *name* and *type* will be closed. If both *name* and *group* are ' ' (blank), all objects matched with *type* will be closed.

type IN: The object type: MODIS_ARRAY(for SDS, the numerical value is 720, the same value as DFTAG_NDG), MODIS_TABLE(for Vdata, the numerical value is 1962, the same value as DFTAG_VH), or ALL_MODIS_TYPES (the nuemrical value is 0). If *type* is 720, only SDS objects will be closed. If *type* is 1962, only Vdata will be closed. If *type* is 0, both Vdata and SDS objects specified by *name* and *group* will be closed. Therefore, to close all opened objects, set both *name* and *group* to NULL and set *type* to 0.

Output parameter: N/A

Returns: Number of objects closed, MFAIL if error occurs.

Error Message:	Description:
`endMODISobjaccess detected FAIL from HDF procedure SDendaccess while closing access to array did->name.`	
`endMODISobjaccess detected FAIL from HDF procedure VSdetach while closing access to table did->name.`	

getMODISardims (GMARDM)

Purpose: Retrieves the rank, dimensions, and data type of a specified MODIS HDF file array structure.

Description: getMODISardims (GMARDM) retrieves the essential characteristics of an HDF SDS array structure contained in a MODIS HDF file. This provides the information needed to properly read data from the array structure using getMODISarray (GMAR).

The *groupname* string provides the facility to select an array structure placed in a particular HDF 'Vgroup' data group. Alternatively, the entire file will be searched for an array structure named *arrayname* if *groupname* = NULL in C or group is a blank string (' ') in FORTRAN.

There are two approaches to determine the proper dimensioning of *dimsizes* to provide sufficient elements for the dimensions of the array structure. The first is to provide a generous *dimsizes* array. The second approach is to use the *rank* variable as an input containing the number of elements in *dimsizes.* If *dimsizes* is inadequate for the multi-dimensional array structure in question, getMODISardims will fail and will return the rank of the array structure, allowing for the dimension information to be retrieved on a subsequent call.

C:
```
int getMODISardims(MODFIL *file,   char *arrayname, char *groupname,
char *data_type, long int *rank  long int dimsizes[])
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference the MODIS HDF file containing the target array structure.

arrayname IN: ASCII string name of the target array structure.

groupname IN: ASCII string name of the data group containing the target array structure. If set to NULL, the entire file will be searched for the array structure named *arrayname.*

Output Parameters:

data_type OUT: String describing the data type of the array.

Possible C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
```

"int64"
"float32"
"float64"

rank IN/OUT: The number of elements in the array *dimsizes* on input. This is replaced with the number of dimensions in the array structure for output. It is set to 0 if a functional error occurs. No dimensional information will be provided if rank = NULL.

dimsizes OUT: Array describing the size of each dimension of the the array structure. The dimensions will not be provided unless *dimsizes* contains sufficient elements for the rank of the array. If the *dimsizes* is too small, the array rank information will be returned in the rank argument and the function will return MFAIL.

FORTRAN:

```
INTEGER FUNCTION GMARDM (modfil, arrnm, group, dtype, rank, dims)
      INTEGER           modfil(MODFILLEN),rank, dims(*)
      CHARACTER*(*)     arrnm, group, dtype
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file containing the target array structure.

arrnm IN: ASCII string name of the target array structure.

group IN: ASCII string name of the data group containing the target array structure. If *group* = ' ' (blank), the entire file will be searched for the array structure named *arrayname.*

Output Parameters:

dtype OUT: Data type of the array.

Possible FORTRAN data types:

'INTEGER*1'
'UINTEGER*1'
'INTEGER*2'
'UINTEGER*2'
'INTEGER*4'
'UINTEGER*4'
'INTEGER*8'
'REAL*4'
'REAL*8'

rank IN/OUT: The number of elements in the array *dims* on input. This is replaced with the number of dimensions in the array structure for output. It is set to 0 if a functional error

occures. No dimensional information will be provided if rank = NULL.

dims OUT: Array describing the size of each dimension of the the array structure. The dimensions will not be provided unless *dims* contains sufficient elements for the rank of the array. If the *dims* is too small, the array rank information will be returned in the rank argument and the function will return MFAIL.

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	**Description:**
`getMODISardims unable to access the` *`arrayname`* `array with a NULL file MODFIL structure.`	
`getMODISardims unable to access an array without an array name input.`	
`getMODISardims unable to return the` *`arrayname`* `array's dimensions without a dimsizes array.`	
`getMODISardims cannot find the` *`arrayname`* `array.`	
`getMODISardims cannot find the` *`arrayname`* `array in the` *`groupname`* `data group.`	
`getMODISardims unable to find the` *`groupname`* `data group containing the` *`arrayname`* `array.`	
`getMODISardims cannot get an sds_id for the` *`arrayname`* `array.`	
`getMODISardims detected FAIL from HDF procedure SDgetinfo attempting to access the` *`arrayname`* `array.`	
`getMODISardims detected FAIL from HDF procedure SDendaccess attempting to detach from the` *`arrayname`* `array.`	The output from getMODISardims may not be valid if SDendaccess fails.
`Rank (if provided) is set to 0 if any of the errors associated with the above messages occur.`	
`getMODISardims unable to return the` *`arrayname`* `array's` *`sds_rank`* `dimension sizes in a` *`rank`* `element dimsizes array.`	getMODISardims will not attempt to write to the dimsizes output array, but it will return the rank of the target HDF array structure. The dimsizes array needs to have at least this many elements.

getMODISarinfo (GMARIN)

Purpose: Reads the value(s) of a local attribute attached to an SDS (array) from a MODIS HDF file.

Description: getMODISarinfo (GMARIN) retrieves the value(s) associated with an attribute = value(s) pair attached to an SDS (array) by putting in the array name and the attribute name. If the attribute cannot be found, the routine will return MFAIL, and the passed variable will be unchanged.

The routine will also fail if the provided *data_type* is found to be different from the attribute's data type or *n_elements* is found to be too small to contain the attribute's value(s). getMODISarinfo replaces this input information with the actual data type and number of elements contained in the attribute value(s) (in the case of character data, it is the length of the string, including the '\0' character). These values returned by the routine may be used to properly retrieve the attribute value(s) with a second call to the routine. If a function called by the routine fails, the routine will set *n_elements* to zero.

C:
```
int getMODISarinfo(MODFIL *file, char *arrayname, char *groupname, char
*attribute, char *data_type, long int *n_elements, void *value)
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference a MODIS HDF file containing the attribute.

arrayname IN: ASCII string name of the array. Provided macros for accepted MODIS HDF file array names are listed in Appendix A, M-API-Supplied Constants and Macros.

groupname IN: ASCII string name of the data group containing the array structure to which the attribute is attached. If set to NULL, the entire file will be searched for the array structure named *arrayname.*

attribute IN: ASCII string name of the attribute. Provided macros for accepted MODIS HDF file attribute names are listed in Appendix A, M-API-Supplied Constants and Macros.

data_type IN/OUT: Address of data type of the *value* output. Output returns with the data type of the retrieved attribute. The memory size of this argument should be at least 8 bytes long.

Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
"float64"
"char *"
```

n_elements IN/OUT: Address of the number of elements the *value* buffer can contain. Output returns with the number of elements required to contain the attribute. If a function failure occurs, the value will be set to zero.

Output Parameters:

value OUT: Address of value(s) associated with the attribute.

FORTRAN:

```
INTEGER FUNCTION GMARIN(modfil, arrnm, group, attr, dtype, nms,
value)
      INTEGER     modfil(3), nms
      CHARACTER*(*) group, arrnm, dtype, attr
      BYTE        value(*)
```

Input Parameters:

modfil IN: Array that is used to reference a MODIS HDF file containing the attribute.

arrnm IN: ASCII string name of the array. Provided macros for accepted MODIS HDF file array names are listed in Appendix A, M-API-Supplied Constants and Macros.

group IN: ASCII string name of the data group containing the array structure to which the attribute is attached. If set to NULL, the entire file will be searched for the array structure named *arrnm.*

attr IN: ASCII string name of the attribute. Provided macros for accepted MODIS HDF file attribute names are listed in Appendix A, M-API-Supplied Constants and Macros.

dtype IN/OUT: ASCII string of data type of the *value* output. Output returns with the data type of the retrieved attribute. The memory size of *dtype* should be at least 13 characters long.

Permitted FORTRAN data types:

```
'CHARACTER*(*)'
 'INTEGER*1'
 'UINTEGER*1'
 'INTEGER*2'
 'UINTEGER*2'
 'INTEGER*4'
 'UINTEGER*4'
 'INTEGER*8'
 'UINTEGER*8'
 'REAL*4'
 'REAL*8'
```

nms IN/OUT: The number of elements the *value* buffer can contain. Output returns with the number of elements required to contain the attribute. If a function failure occurs, the value will be set to zero.

Output Parameters:

value OUT: Values associated with the attribute.

Returns: MAPIOK if successful, MFAIL if *value* cannot contain the retrieved attribute value, the data type is different, the attribute cannot be found, or an error occurs.

Error Message:	**Description:**
`getMODISarinfo unable continue with empty n_elements.`	
`getMODISarinfo unable to access an array attribute without an attribute name input.`	
`getMODISarinfo unable to access the` *`attribute`* `attribute without the name of the array it is associated with.`	No *arrayname* attribute was provided.
`getMODISarinfo cannot find array "`*`arrayname`*`".`	
`getMODISarinfo unable to find the` *`groupname`* `data group containing the` *`arrayname`* `array.`	This may be preceded by one of the following three messages:
`searchMODISgroup fails to search object` *`objectname`* `in group` *`groupname`* `because Vattach fails.`	
`searchMODISgroup unable to find the specified Vgroup group` *`groupname`*`.`	
`searchMODISgroup fails to obtain` *`objectname`*`'s tag and reference number.`	

Error Message:	Description:
`getMODISarinfo cannot find the` *`arrayname`* `array in the` *`groupname`* `data group.`	The Vdata array could not be found in the specified Vgroup data group.
`getMODISarinfo detected FAIL retrieving the data type string for the` *`attribute`* `attribute using DFNT_to_datatype.`	M-API currently does not recogniize the HDF number types 3 (unsigned char), 7 (float128), 27 (unsigned int64), 28 (int128), 30 (unsigned int128), 42 (char16), 43 (unsigned char 16), or any greater than 512 (machine specific, custom, or little endian storage formats).
`getMODISarinfo cannot find local array attribute` *`attribute`*`.`	
`getMODISarinfo detected FAIL from HDF procedure SDselect attempting to read the` *`attribute`* `attribute.`	
`getMODISarinfo detected FAIL from HDF procedure SDattrinfo attempting to read the` *`attribute`* `attribute.`	
`getMODISarinfo detected FAIL from HDF procedure SDreadattr attempting to read the` *`attribute`* `attribute.`	
`getMODISarinfo unable to read local array attribute without output buffer for` *`attribute`*`.`	
`getMODISarinfo detected FAIL from HDF procedure SDendaccess attempting to read the` *`attribute`* `attribute.`	**n_elements* is set to 0 if any of the errors associated with the messages above occur.
`WARNING: Vgroup groupname contains non-existing SDS object with reference ID` *`ref_id`*`.`	Information about an SDS array structure that doesn't really exist has been found in the Vgroup data group being accessed. While this will not directly prevent reading the specified local array attribute, it does identify a probable defect in the HDF file.

getMODISarray (GMAR)

Purpose: Retrieves an array or subarray of data from a MODIS HDF file array structure.

Description: getMODISarray (GMAR) returns a multi-dimensional array of data from an HDF SDS array structure contained in a MODIS HDF file. The data array must be of the same data type as data in the target array structure. In addition, the dimensions and array region requested from the array structure must be consistent with the structure's rank and dimensions. (The array structure's data type, rank, and dimensions may be retrieved using getMODISarinfo (GMARIN)). If a getMODISarray error message occurs, the data retrieval will not be performed.

The *groupname* string provides the facility to select an array structure placed in a particular HDF 'Vgroup' data group. Alternatively, the entire file will be searched for an array structure named *arrayname* if groupname = NULL in C or *group* is a blank string (' ') in FORTRAN.

C:
```
int getMODISarray(MODFIL *file, char *arrayname, char *groupname, long
int start[], long int dimsizes[], void *data)
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference the MODIS HDF file containing the target array structure.

arrayname IN: ASCII string name of the target array structure.

groupname IN: ASCII string name of the data group containing the target array structure. If set to NULL, the entire file will be searched for the array structure named *arrayname.*

start IN: Array containing the array structure location to begin reading the data from the array structure. *start* must have the same number of elements as the target array has dimensions.

dimsizes IN: Array describing the size of the array being retrieved from the array structure. *dimsizes* must have the same number of elements as the target array structure has dimensions and the product of the array dimensions must equal the number of elements in *data.*

Output Parameters:

data OUT: Address of the data buffer.

FORTRAN:

```
INTEGER FUNCTION GMAR (modfil, arrnm, group, start, dims, data)
      INTEGER        modfil(MODFILLEN),start(*), dims(*)
      CHARACTER*(*)  arrnm, group
      BYTE           data(*)
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file containing the target array structure.

arrnm IN: ASCII string that will be the name of the array.

group IN: ASCII string name of the data group containing the target array structure. If *group* = ' ' (blank) the entire file will be searched for the array structure named *arrnm.*

start IN: Array containing the array structure location to begin reading the data from the array structure. *start* must have the same number of elements as the target array has dimensions.

dims IN: Array describing the size of the array being retrieved from the array structure. *dims* must have the same number of elements as the target array structure has dimensions and the product of the array dimensions must equal the number of elements in *data.*

Output Parameters:

data OUT: Multi-dimensional data array.

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	**Description:**
`getMODISarray unable to read from the` *`arrayname`* `array with a NULL file MODFIL structure.`	
`getMODISarray unable to read from an array without an array name input.`	
`getMODISarray unable to read from the` *`arrayname`* `array without array dimension input.`	
`getMODISarray unable to read from the` *`arrayname`* `array without a data buffer.`	
`getMODISarray cannot find the` *`arrayname`* `array.`	
`getMODISarray cannot find the` *`arrayname`* `array in the` *`groupname`* `data group.`	
`getMODISarray unable to find the` *`groupname`* `data group containing the` *`arrayname`* `array.`	

Error Message:	Description:
`getMODISarray unable to read data from invalid array structure locations in the` *`arrayname`* `array.`	This error message may be preceded by one of the following two messages:
`SDS_footprintOK detected FAIL from HDF procedure Sdgetinfo.`	
`Unable to access data at invalid array structure locations` *`"start[0] ... start[r]".`*	
`getMODISarray detected FAIL from HDF procedure SDselect while attempting to read from the` *`arrayname`* `array.`	
`getMODISarray detected FAIL from HDF procedure SDgetinfo while attempting to read from the` *`arrayname`* `array.`	
`getMODISarray detected FAIL from HDF procedure SDwritedata while attempting to read from the` *`arrayname`* `array.`	
`getMODISarray detected FAIL from HDF procedure SDendaccess while attempting to read from the` *`arrayname`* `array.`	

getMODISdiminfo (GMDMIN)

Purpose: Reads the value(s) of a local attribute attached to a specific dimension of an SDS (array) from a MODIS HDF file.

Description: getMODISdiminfo (GMDMIN) retrieves the value(s) associated with an attribute = value pair for a dimension attribute by giving the array's name and dimension number and the attribute name. If the attribute cannot be found, the routine will return MFAIL and leave the passed variable unchanged.

The routine will also fail if the provided *data_type* is found to be different from the attribute's data type or the *n_elements* is found to be too small to contain the attribute's value. getMODISdiminfo replaces this input information with the actual data type and number of elements contained in the attribute value (in the case of character data, it is the length of the string, including the '\0' character). The values returned by the routine may be used to properly retrieve the attribute value with a second call to the routine. If a function called by the routine fails, the routine will set *n_elements* to zero.

A variable of the proper data type should be passed for the *value* parameter. The data type information required to properly use either routine may be found in Appendix A, M-API-Supplied Constants and Macros, in this document and Appendix F, Examples of MODIS Data Product File Definitions, in the M-API User's Guide.

C:
```
int getMODISdiminfo(MODFIL *file, char *arrayname, char *groupname, long
int dimension, char *attribute, char *data_type, long int *n_elements,
void *value)
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference a MODIS HDF file containing the attribute.

arrayname IN: ASCII string name of the array. Provided macros for accepted MODIS HDF file array names are listed in Appendix A, M-API-Supplied Constants and Macros.

groupname IN: ASCII string name of the data group containing the array structure to which the attribute is attached. If set to NULL, the entire file will be searched for the array structure named *arrayname.*

dimension IN: The dimension number from which the attribute is retrieved (0-based).

attribute IN: ASCII string name of the attribute. Provided macros for accepted MODIS HDF file attribute names are listed in Appendix A, M-API-Supplied Constants and Macros.

data_type IN/OUT: Address of data type of the *value* output. Output returns with the data type of the retrieved attribute. The memory size of this arguments should be at least 8 bytes long.

Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
"float64"
"char *"
```

n_elements IN/OUT: Address of the number of elements the *value* buffer can contain. Output returns with the number of elements required to contain the attribute. If a function failure occurs, the value will be set to zero.

Output Parameters:

value OUT: Address of value associated with the attribute.

FORTRAN:

```
INTEGER FUNCTION GMDMIN(modfil, arrnm, group, dim, attr, dtype,
nms, value)
      INTEGER           modfil(3), nms, dim
      CHARACTER*(*) group, arrnm, dtype, attr
      BYTE        value(*)
```

Input Parameters:

modfil IN: array that is used to reference a MODIS HDF file containing the attribute.

arrnm IN: ASCII string name of the array. Provided macros for accepted MODIS HDF file array names are listed in Appendix A, M-API-Supplied Constants and Macros.

group IN: ASCII string name of the data group containing the array structure to which the attribute is attached. If set to ' ' (blank), the entire file will be searched for the array structure named *arrnm.*

dim IN: The dimension number from which the attribute values will be retrieved (0-based).

attr IN: ASCII string name of the attribute. Provided macros for accepted MODIS HDF file attribute names are listed in Appendix A, M-API-Supplied Constants and Macros.

dtype IN/OUT: ASCII string of data type of the *value* output. Output returns with the data type of the retrieved attribute. The memory size of *dtype* should be at least 13 characters long.

Permitted FORTRAN data types:

```
'CHARACTER*(*)'
 'INTEGER*1'
 'UINTEGER*1'
 'INTEGER*2'
 'UINTEGER*2'
 'INTEGER*4'
 'UINTEGER*4'
 'INTEGER*8'
 'UINTEGER*8'
 'REAL*4'
 'REAL*8'
```

nms IN/OUT: The number of elements available in the *value* array. Output returns with the number of elements required to contain the attribute. If a function failure occurs, the value will be set to zero.

Output Parameters:

value OUT: Values associated with the attribute.

Returns: MAPIOK if successful, MFAIL if *value* cannot contain the retrieved attribute value, the data type is different, the attribute cannot be found, or an error occurs.

Error Message:	**Description:**
`getMODISdiminfo unable continue with empty n_elements.`	
`getMODISdiminfo unable to access an array attribute without an attribute name input.`	
`getMODISdiminfo unable to access the attribute attribute without the name of the array it is associated with.`	No *arrayname* attribute was provided.
`getMODISdiminfo cannot find array "arrayname".`	

Error Message:	Description:
`getMODISdiminfo unable to find the` *`groupname`* `data group containing the` *`arrayname`* `array.`	This may be preceded by one of the following three messages:
`searchMODISgroup fails to search object` *`objectname`* `in group` *`groupname`* `because Vattach fails.`	
`searchMODISgroup unable to find the specified Vgroup group` *`groupname`*`.`	
`searchMODISgroup fails to obtain` *`objectname`*`'s tag and reference number.`	
`getMODISdiminfo cannot find the` *`arrayname`* `array in the` *`groupname`* `data group.`	The Vdata array could not be found in the specified Vgroup data group.
`getMODISdiminfo detected FAIL retrieving the data type string for the` *`attribute`* `attribute using DFNT_to_datatype.`	M-API currently does not recogniize the HDF number types 3 (unsigned char), 7 (float128), 27 (unsigned int64), 28 (int128), 30 (unsigned int128), 42 (char16), 43 (unsigned char 16), or any greater than 512 (machine specific, custom, or little endian storage formats).
`getMODISdiminfo cannot find local array dimension attribute` *`attribute.`*	
`getMODISdiminfo detected FAIL from HDF procedure SDselect attempting to read the` *`attribute`* `attribute`	
`getMODISdiminfo detected FAIL from HDF procedure SDgetinfo attempting to read the` *`attribute`* `attribute.`	
`getMODISdiminfo detected FAIL from HDF procedure SDattrinfo attempting to read the` *`attribute`* `attribute.`	
`getMODISdiminfo unable to retrieve an` *`attribute`* `attribute for dimension` *`dimension`*`. The` *`arrayname`* `array has` *`rank`* `dimensions.`	
`getMODISdiminfo detected FAIL from HDF procedure SDgetdimid attempting to read the` *`attribute`* `attribute.`	

Error Message:	Description:
`getMODISdiminfo detected FAIL from HDF procedure SDreadattr attempting to read the` *`attribute`* `attribute.`	
`getMODISdiminfo unable to read local array attribute without output buffer for` *`attribute`*`.`	
`getMODISdiminfo detected FAIL from HDF procedure SDendaccess attempting to read the` *`attribute`* `attribute.`	**n_elements* is set to 0 if any of the above errors associated with the messages occur.
`WARNING: Vgroup groupname contains non-existing SDS object with reference ID` *`ref_id`*`.`	Information about an SDS array structure that doesn't really exist has been found in the Vgroup data group being accessed. While this will not directly prevent reading the specified local array attribute, it does identify a probable defect in the HDF file.

getMODISdimname (GMDNAM)

Purpose: Obtains the name of a specific dimension in an array structure.

Description: getMODISdimname (GMDNAM) retrieves the name of an HDF dimension associated with an array structure given the array's name and the dimension's number. If the dimension name cannot be found, the routine will return MFAIL (-1). This routine does not retrieve a "long_name" dimension attribute. However, getMODISdiminfo can retrieve such a dimension label (if it exists).

The *groupname* string provides the facility to select an array structure placed in a particular HDF 'Vgroup' data group. Alternatively, the entire file will be searched for an array structure named *arrayname* if the argument *groupname* = NULL in C or *group* is a blank string (' ') in FORTRAN.

C:
```
int getMODISdimname(MODFIL *file, char *arrayname, char *groupname,
long int dimension, char *dimname)
```

Input parameters:

file IN: Address of MODFIL structure that is used to reference a MODIS HDF file containing the local dimension name.

arrayname IN: ASCII string name of the target array structure.

groupname IN: ASCII string name of the data group containing the target array structure. If set to NULL, the entire file will be searched for the array structure named *arrayname.*

dimension IN: The dimension number to which the dimension name is attached (0-based). The 0 dimension of an HDF SDS array structure is associated with the least rapidly varying array index.

Output parameters:

dimname OUT: ASCII string for the dimension name. Provided array should be at least 256 bytes long.

FORTRAN:
```
INTEGER FUNCTION GMDNAM(modfil, arrnm, group, dim, dname)
      INTEGER modfil(MODFILLEN), dim
      CHARACTER *(*) arrnm, group, dnam
```

Input parameters:

modfil IN: Array that is used to reference the MODIS HDF file containing the dimension name.

arrnm IN: ASCII string name of the target array structure.

group IN: ASCII string name of the data group containing the target array structure. If *group* = ' ' (blank), the entire file will be searched for the array structure named *arrnm.*

dim IN: The dimension number to which the dimension name is attached (0-based). The 0 dimension of an HDF SDS array structure is associated with the most rapidly varying array index.

Output parameters:

dname OUT: ASCII string for the dimension name. Provided array should be at least 256 bytes long.

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
`getMODISdimname unable to read the name of a dimension without the name of array it is associated with.`	
`getMODISdimname unable to read the name of a dimension in the` *`arrayname`* `array with an invalid MODIS file structure input.`	
`getMODISdimname unable to read the name of a` *`arrayname`* `array's dimension name without an output character string.`	
`getMODISdimname detected MFAIL from M-API internal function getMODISarrayid while attempting to obtain the name of dimension` *`dimension`* `in the` *`arrayname`* `array.`	
`getMODISdimname unable to read the dimension name of the non-existent dimension` *`dimension`* `of the` *`arrayname`* `array.`	
`getMODISdimname detected FAIL from HDF procedure SDgetdimid attempting to read the name of an` *`arrayname`* `array's dimension.`	
`getMODISdimname detected FAIL from HDF procedure SDsetdimname attempting to read the name of an` *`arrayname`* `array's dimension.`	

getMODISECSinfo (GMECIN)

Purpose: To retrieve the value of a parameter from an HDF PVL text block.

Description: In HDF-EOS, parameters are collected together to form a text block using PVL. Then the text block is stored in HDF as a single attribute. getMODISECSinfo (GMECIN) retrieves the value of a parameter from the PVL text block.

In order to obtain the value(s) of a parameter inside a PVL text block, the function reads the PVL text block specified by *PVLAttrName* from the MODIS file, creates the internal ODL tree structure from the PVL text block, and searches the tree structure to retrieve the value of a parameter. The tree structure is then saved internally for consecutive searches in the same PVL text block.

If multiple parameters will be retrieved from the same PVL block, perform the follwoing:

For C, set *PVLAttrName* to the HDF PVL attribute name in the first call and set it to NULL for consecutive calls. If the next call is to retrieve the value of a parameter in a different PVL text block, set the *PVLAttrName* to the new PVL attribute name. The saved old tree structure will be deleted automatically and a new ODL tree will be created and saved.

For FORTRAN, set *pvlname* to the HDF PVL attribute name in the first call and set it to blank (' ') for consecutive calls. If the next call is to retrieve the value of a parameter in a different PVL text block, set the *pvlname* to the new PVL attribute name. The saved old tree structure will be deleted automatically and a new ODL tree will be created and saved.

When all the calls to getMODISECSinfo are complete, it should be called once more with both *PVLAttrName* and *parmName set* to NULL in C or *pvlname* and *pvname* set to blank (' ') in FORTRAN. This will release any memory allocated by the function. In addition, this will reduce memory leaks.

C:
```
int getMODISECSinfo(MODFIL *file,  char *PVLAttrName, char *parmName,
char *data_type, long int *n_elements, void *value)
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference the MODIS HDF file containing the target PVL attribute.

PVLAttrName IN: ASCII string name of the HDF attribute which contains the PVL text block. Setting *PVLAttrName* to NULL while *parmName* is not NULL will result in searching the last PVL text block for the value of *parmName* parameter.

parmName IN: ASCII string name of a parameter whose value will be retrieved. Setting both *PVLAttrName* and *parmName* to NULL will release the memory occupied by the internal ODL tree. The *parmName* can be the parameter name only or a combination of the name and the class which will be represented as "name.class".

n_elements IN/OUT: Address of the number of memory elements as *data_type* available in the *value* array. The attribute's value will not be retrieved unless **n_elements* indicates that sufficient space is available in *value*. getMODISECSinfo replaces *n_elements* with the number of elements required to contain the metadata. If the parameter cannot be found, **n_element* will be left unchanged or set to 0 if a called function returns an error. This argument must not be the address of a constant.

SPECIAL CASE for multiple strings:
If there are multiple character strings for the parameters, strings will be packed together and returned in *value* . The separator between strings is '\0'. The low 16 bit of *n_elements* will return the total bytes in the values, including the '\0's between the strings and the '\0' at the end of the last string. The part above the low 16 bits will return number of strings packed - 1. To obtain how many strings were retrieved, do the calculation:

```
n_strings = *n_elements/65536 + 1
n_bytes = *n_elements%65536
```

Therefore, if **n_elements* is less than 65536, there is only one string in *value* and **n_elements* is the number of bytes (characters) in the string, including the last '\0'.

The M-API module substrMODISECSinfo can be used to unpack the ECS metadata that has been retrieved by getMODISECSinfo.

data_type IN/OUT: Data type of *value*. Output is replaced with the data type of the retrieved metadata. There are only 3 data types in PVL:

"char *"
"int32"
"float64"

However, "float32" may be used in the input. If the value of the parameter is "float64" type, this function will return the value (s) in "float32" if users set the input value of *dtype* to "float32". This argument must be a variable. The memory size for *dtype* should be at least DATATYPELENMAX bytes long

Output Parameters:

value OUT: buffer for the value. User should allocate enough memory for this buffer. If there are multiple data values in character type, the value will be placed consecutively. If the data value type is "char *", strings will be separated by '\0'.

FORTRAN:

```
INTEGER FUNCTION  GMECIN(modfil,  pvlname,  pname, nms, dtype,
      pvalue)
      INTEGER modfil(MODFILLEN), nms
      CHARACTER*(*) pvlname,  pname, dtype
      BYTE pvalue
```

Input Parameters:

modfil IN: MODFIL file array that is used to reference the MODIS HDF file containing the target PVL attribute.

pvlname IN: ASCII string name of the HDF attribute which contains the PVL text block. Setting *pvlname* to ' ' (blank) while *pname* is not equal to ' ' (blank) will result in searching the last PVL text block for the value of *pname* parameter.

pname IN: ASCII string name of a parameter whose value will be retrieved. Setting both *pvlname* and *pname* to ' ' (blank) will release the memory occupied by the internal ODL tree. The *pname* could be the parameter name only or a combination of name and class represented as "name.class".

nms IN/OUT: The number of memory elements as *dtype* available in the *value* array. The attribute's value will not be retrieved unless *nms* indicates that there is sufficient space available in *value*. getMODISECSinfo replaces this input with the number of elements required to contain the metadata. If the parameter cannot be found, **nms* will be left unchanged or set to 0 if a function error occurs. This argument must be a variable.

SPECIAL CASE for multiple strings:

If there are multiple character strings for the parameters, strings will be packed together and returned in *value* . The separator between strings is '\0' (numerical value 0). The low 16 bit of *nms* will return the total bytes in *value*, including the '\0's. The part above the low 16 bits will return (number of strings packed - 1). To obtain how many string retrieved, do the calculation:

n_strings = nms/65536 + 1
n_bytes = MOD(nms, 65536)

Therefore, if *nms* is less than 65536, there is only one string in *value* and *nms* is the number of bytes (characters) in the string.

The M-API module SMECIN can be used to unpack the ECS metadata that has been retrieved by GMECIN.

dtype IN/OUT: Data type of *pvalue*. Output is replaced with the data type of the retrieved metadata. There are only 3 data types in PVL:

```
'CHARACTER*(*) '
'INTEGER*4'
'REAL*8'
```

However, "REAL*4" may be used as input. If the value of the parameter is in 'REAL*8' type, this function will return in 'REAL*4' if users set the input value of *dtype* to 'REAL*4'. This argument must be a variable.The memory size for *dtype* should be at least DATATYPELENMAX bytes long

pvalue OUT: Buffer for the *pvalue*. User should allocate enough memory for this buffer. If there are multiple data values, the value will be placed consecutively. If the data value type is "CHARACTER*(*)" each string occupies 100 bytes, i.e, first string starts at 0 byte, the second string starts at 100 bytes. If the real length of the string is less than 100, the remaining bytes are set to ' '.

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	**Description:**
`getMODISECSinfo can not continue without the n_elements input.`	

Error Message:	Description:
getMODISECSinfo unable to access an ECS metadata without the parameter name input.	
getMODISECSinfo unable to access the *parmName* metadata without the name of the global attribute it is stored within.	
getMODISECSinfo unable to access the *parmName* metadata from ECS global attribute *PVLAttrName* without the data type input.	
getMODISECSinfo detected fails in procedure MPVL2ODL while attempting to retrieve parameter *parmName* from ECS global attribute *PVLAttrName*.	
getMODISECSinfo can not find the *parmName* metadata.	
getMODISECSinfo found the value for parameter *parmName* is undefined.	
getMODISECSinfo unable to access the *parmName* metadata without the output data buffer.	
getMODISECSinfo found unknown ODL value type *valueNode->item.type* for parameter *parmName*.	

getMODISfields (GMFLDS)

Purpose: Retrieves the number of records, number of fields, the field names, data types, and the class of a specified MODIS HDF file table structure.

Description: getMODISfields (GMFLDS) retrieves the essential characteristics of an HDF Vdata table structure contained in a MODIS HDF file. This provides the information needed to properly read data from the table structure using getMODIStable (GMTBL) or to write to it using putMODIStable (PMTBL). If any of the output parameters are set to NULL, then that parameter is not retrieved. An error (MFAIL) will be returned if:

1. The output strings are not long enough to contain the data type or field name strings for all the Vdata's fields,
2. An unknown (e.g., not supported by the M-API) number type is encountered, or
3. An HDF routine FAILs.

The data type string (if requested) will be returned truncated to the point where the fault occurred. *stringlen,* the address of the length of the *data_type* and *fieldname* output strings, is a required input if either of these strings is to be retrieved. It is normally revised to contain the actual array length required to hold the larger ot the two output strings. If error 2 or 3 occurs, however, **stringlen* is set to 0.

The *groupname* string provides the facility to select a table structure existing in a particular HDF 'Vgroup' data group. Alternatively, the entire file will be searched for a table structure named *tablename* if *groupname* = NULL in C or *group* = ' ' (blank) in FORTRAN.

C:

```
int getMODISfields(MODFIL *file, char *tablename, char *groupname,
long int *stringlen, long int *recno, long int *fieldno, char
*fieldname, char *data_type, char *classname)
```

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file containing the target table structure.
tablename	IN:	ASCII string name of the target table structure.
groupname	IN:	ASCII string name of the data group containing the target table structure. If set to NULL, the entire file will be searched for the table structure named *tablename.*
stringlen	IN/OUT:	Address of minimum length of *fieldname* and *data_type* arrays. Returns the minimum array length actually required to hold the longer of the two strings. It is set to 0 if a functional error occurs.

Ouput parameters:

recno OUT: Number of records (rows) present in the table structure.

fieldno OUT: Number of fields (columns) present in the table structure.

fieldname OUT: Array of comma-delimited ASCII string table headers.

data_type OUT: Array of comma-delimited data types for each table field.

Possible C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
"float64"
```

classname OUT: ASCII string for the class name of the table structure. Provided array should be at least VSNAMELENMAX (64) bytes long.

FORTRAN:

```
INTEGER FUNCTION GMFLDS (modfil, tblnm, group, strln, recno,
fldno, fldnm, dtype, clss)
     integer modfil(3), strln, recno, fldno
     character*(*) tblnm, group, dtype, clss, fldnm
```

Input Parameters:

modfil IN: M-API FORTRAN file array that is used to reference the MODIS HDF file containing the target table structure.

tblnm IN: ASCII string name of the target table structure.

group IN: ASCII string name of the data group containing the target table structure. If set to ' ' (blank), the entire file will be searched for the table structure named *tblnm.*

Output Parameters:

strln OUT: Returns the minimum string length actually required to hold the longer of the two strings. It is set to 0 if a functional error occurs.

recno OUT: Number of records (rows) present in the table structure.

fldno OUT: Number of fields (columns) present in the table structure.

fldnm OUT: ASCII string of comma-delimited ASCII string table headers.

dtype OUT: ASCII string of comma-delimited data types for each table field.

Permitted FORTRAN data types:

```
'CHARACTER*(*)'
'INTEGER*1'
'UINTEGER*1'
'INTEGER*2'
'UINTEGER*2'
'INTEGER*4'
'UINTEGER*4'
'INTEGER*8'
'UINTEGER*8'
'REAL*4'
'REAL*8'
```

class OUT: ASCII string for the class name of the table structure. Provided array should be at least VSNAMELENMAX (64) bytes long.

Returns: MAPIOK if successful, MFAIL if unable to retrieve all the requested outputs or unable to attach to the Vdata at all.

Error Message:	**Description:**
`getMODISfields unable to access the` *`tablename`* `table with a NULL file MODFIL structure.`	
`getMODISfields unable to access a table without a table name input.`	
`getMODISfields cannot find` *`tablename`* `table.`	
`getMODISfields unable to find the` *`groupname`* `data group containing the` *`tablename`* `table.`	This may be preceded by one of the following two messages:
`searchMODISgroup fails to search object` *`objectname`* `in group` *`groupname`* `because Vattach fails.`	
`searchMODISgroup unable to find the specified Vgroup group` *`groupname`*`.`	
`getMODISfields cannot find the` *`tablename`* `table in the` *`groupname`* `data group.`	This may be preceded by the following message:
`searchMODISgroup fails to obtain` *`objectname`*`'s tag and reference number.`	The Vdata table could not be found in the specified Vgroup data group.
`getMODISfields detected FAIL from HDF procedure VSattach attempting to access the` *`tablename`* `table.`	

Error Message:	Description:
`getMODISfields detected FAIL from HDF procedure VSgetfields.`	A problem occurred with retrieving information about the number and names of the table's fields.

`getMODISfields detected FAIL retrieving the data type string for the` *`tablename`* `table using Vfdatatypes.`
`VFdatatypes detected FAIL from HDF routine Vfnfields.`
`VFdatatypes detected unrecognized HDF number type.`
`M-API currently does not recognize number types 3 (unsigned char), 7 (float128), 27 (unsigned int64), 28 (int128), 30 (unsigned int128), 42 (char16), 43 (unsigned char 16), or any greater than 512 (machine specific, custom, or little endian storage formats).`
	This error message may be preceded by one of the following two messages:

`getMODISfields detected FAIL from HDF procedure VSinquire.`	A problem occurred with retrieving information about the number of records in the table.
`getMODISfields unable to fit` *`tablename`* `table's <string length> byte field names string into output string of unknown length.`	The length of the output string *fieldname* was not provided in the parameter *stringlen*.
`getMODISfields unable to fit the` *`tablename`* `table's <string length> byte field names into *`*`stringlen`* `byte output string.`	*stringlen* will return the array length required to hold the table's field names.
`getMODISfields unable to fit` *`tablename`* `table's data types string into output string of unknown length.`	The length of the output string *data_type* was not provided in the parameter *stringlen*.
`getMODISfields unable to fit the` *`tablename`* `table's <string length> byte data types into *`*`stringlen`* `byte output string.`	This error message will be preceded by:
`VFdatatypes unable to fit data types into output string.`	*stringlen* will return the array length required to hold the table's data type string. If both the field names and the data types were requested, the larger of the two array lengths is returned.

getMODISfileinfo (GMFIN)

Purpose: Reads the value(s) of a global attribute from a MODIS HDF file.

Description: getMODISfileinfo retrieves the value(s) associated with an attribute = value(s) pair given the attribute name. If the attribute cannot be found, the routine will return MFAIL and the passed variable unchanged.

The routine will also fail if the provided *data_type* is found to be different than the attribute's data type or the *n_elements* is found to be too small to contain the attribute's value(s). getMODISfileinfo replaces this input information with the actual data type and number of elements contained in the attribute value(s) (in the case of character data, it is the length of the string, including the '\0' character). These attribute's attributes may be used to properly retrieve the attribute value with a second call to the routine. If a function failure occurs, *n_elements* will be set to zero.

A variable of the proper data type should be passed for the *value* parameter. The data type information required to properly use either routine may be found in Appendix A, M-API-Supplied Constants and Macros, in this document and Appendix F, Example of MODIS Data Product File Definitions of the M-API User's Guide. Appendix A has a listing for each M-API provided attribute's attributes that include the data type, the format, and/or specific values associated with it.

C:
```
int getMODISfileinfo(MODFIL *file, char *attribute, char *data_type,
long int *n_elements, void *value)
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference a MODIS HDF file containing the attribute.

attribute IN: ASCII string name of the *attribute*. Provided macros for accepted MODIS HDF file attribute names are listed in Appendix A, M-API-Supplied Constants and Macros.

data_type IN/OUT: Address of data type of the *value* output. Output returns with the data type of the retrieved attribute.

Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
```

"float64"
"char *"

n_elements IN/OUT: Address of number of elements available in the *value* array. Output returns with the number of elements required to contain the attribute. If a function failure occurs, *n_elements* will be set to zero. If the requested attribute does not exist, *n_elements* will be unchanged.

Output parameters:

value OUT: Address of value(s) associated with the attribute.

FORTRAN:

```
INTEGER FUNCTION GMFIN(modfil, attr, dtype, nms, value)
   INTEGER modfil(MODFILLEN), nms
   CHARACTER*(*) dtype, attr
   BYTE     value(*)
```

Input parameters:

modfil IN: Array that is used to reference a MODIS HDF file containing the attribute.

attr IN: ASCII string name of the attribute. Provided macros for accepted MODIS HDF file attr names are listed in Appendix A, M-API-Supplied Constants.

dtype IN/OUT: Data type of the *value* output. Output replaces with the data type of the retrieved attribute. This memory size of this character should be at least 13 characters long.

Permitted FORTRAN data types:

'CHARACTER*(*)'
'INTEGER*1'
'UINTEGER*1'
'INTEGER*2'
'UINTEGER*2'
'INTEGER*4'
'UINTEGER*4'
'INTEGER*8'
'UINTEGER*8'
'REAL*4'
'REAL*8'

nms IN/OUT: Number of elements the *value* buffer can contain. Output replaces with the number of elements required to contain the attribute. If a function error is occur, the value is set to zero. If the requested attribute does not exist, nms will be unchanged.

Output parameters:

value OUT: Values associated with the attribute.

Returns: MAPIOK if successful, MFAIL if *value* cannot contain the retrieved attribute value, the attribute cannot be found, or an error occurs.

Error Message:	Description:
`getMODISfileinfo unable continue with empty n_elements.`	
`getMODISfileinfo unable to access a file attribute without an attribute name input.`	
`getMODISfileinfo unable to access the` *`attribute`* `file attribute without data type input.`	
`getMODISfileinfo unable continue with an invalid MODIS file structure input.`	
`getMODISfileinfo cannot find file attribute` *`attribute`*`.`	
`getMODISfileinfo detected FAIL from HDF procedure SDattrinfo attempting to read the` *`attribute`* `file attribute.`	
`getMODISfileinfo unable to recognize the HDF numerical data type <number type> while attempting to read the` *`attribute`* `file attribute.`	
`getMODISfileinfo unable to read file attribute without output buffer for` *`attribute`*`.`	
`getMODISfileinfo detected FAIL from HDF procedure SDreadattr attempting to read the` *`attribute`* `file attribute.`	

getMODIShobjid (GMHOID)

Purpose: Retrieve the ID of an existing HDF object (either array or table) in a MODIS HDF file so that a M-API application can directly call the HDF library functions to access the data in a MODIS HDF file.

Description: getMODIShobjid (GMHOID) returns the HDF object id. The application program can directly use this ID in the native HDF library function calls because the HDF object is already attached. If the *type* is specified as MODIS_ARRAY, the return value is an array ID (sds id). If the type is MODIS_TABLE, the return value is the table ID (vdata id). The intended use of this function is to allow M-API applications to call HDF functions directly. The call sequence in a M-API application should be:

1. Use openMODISfile to open a MODIS HDF file.
2. Call getMODIShobjid to obtain an object's HDF id.
3. Call HDF library functions by using the obtained id.
4. Call closeMODISfile or completeMODISfile to end the access.

The calls to HDF library functions can be mixed with calls to M-API functions.

NOTE: Do not use the HDF VSdetach (for table) or SDendaccess (for array) to end the access to the object. The closeMODISfile/completeMODISfile will take care of all opened objects. If an application program needs to end the access to an object while keeping the MODIS file open, use the M-API function endMODISobjaccess.

```
C: int getMODIShobjid (MODFIL *file, char *name, char *group, long int
                       type, char *access)
```

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference a MODIS HDF file containing objects to be accessed (Vdata or SDS).
name	IN:	The name of the object.
group	IN:	The name of the group to which objects belongs. If set to NULL, the entire file will be searched for the object named *name.*
type	IN:	The type of object: MODIS_ARRAY (for SDS, the numerical value is 720, the same value as DFTAG_NDG), MODIS_TABLE (for Vdata, the numerical value is 1962, the same value as DFTAG_VH).

access IN: "r" The object is attached for ready only.
"w" The object is attached for read/write.
If the MODIS file is opened for read only and the application program calls this function to obtain an object ID with write operation, the function will return an error.

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION GMHOID (modfil, name, group, type, access)
   INTEGER        modfil(MODFILLEN), type
   Character*(*)  name, group, access
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file.
tblnm IN: ASCII string name of the source table structure.
group IN: ASCII string name of the data group containing the source table structure. If set to ' ' (blank), the entire file will be searched for the table structure named *tblnm.*
type IN: The type f object: MODIS_ARRAY (for SDS, the numerical value is 720, the same value as DFTAG_NDG), MODIS_TABLE (for Vdata, the numerical value is 1962, the same as DFTAG_VH).
access IN: "r" The object is attached for ready only.
"w" The object is attached for read/write.
If the MODIS file is opened for read only and the application program calls this function to obtain an object ID with write operation, the function will return an error.

Output Parameters: N/A

Returns: The object ID or MFAIL.

getMODIStable (GMTBL)

Purpose: Retrieves data records from a MODIS HDF file table structure.

Description: getMODIStable (GMTBL) retrieves one or more fields of data from one or more records from an HDF Vdata table structure contained in a MODIS HDF file. The data are placed in the *data* buffer in consecutive records and in the order that the input *fieldname* are listed. The length of this buffer must be able to contain all the fields requested times the number of records requested. If the *buffsize* input indicates that it is too small to contain the actual quantity of data requested, getMODIStable will fail and return the actual *buffsize* required. The output *data* buffer must be at least this size.

The *groupname* string provides the facility to select a table structure placed in a particular HDF 'Vgroup' data group. Alternatively, the entire file will be searched for a table structure named *tablename* if groupname = NULL in C or *group* = ' ' (blank) in FORTRAN.

C:

```
int getMODIStable(MODFIL *file,  char *tablename, char *groupname,
char*fieldname, long int start, long int recno, long int *buffsize,
unsigned char *data)
```

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file containing the target table structure.
tablename	IN:	ASCII string name of the target table structure.
groupname	IN:	ASCII string name of the data group containing the target table structure. If set to NULL, the entire file will be searched for the table structure named *tablename.*
fieldname	IN:	Array of comma-delimited ASCII string table headers. The data from each field will appear in the same order as the headers.
start	IN:	Zero-based record location to begin reading the data from the table structure.
recno	IN:	Number of records to retrieve from the table structure.
buffsize	IN/OUT:	Address of the *data* buffer size on input, in bytes. The buffer must be at least this size. *buffsize* will normally return the number of bytes of data successfully retrieved. If the buffer is too small, however, the routine returns MFAIL and *buffsize* will contain the size the buffer must be to contain the output data. It is set to 0 if a functional error occurs, making output size determination unreliable.

Output Parameters:

data OUT: Address of the data buffer.

FORTRAN:

```
INTEGER FUNCTION GMTBL(modfil, tblnm, group, field, start, recno,
bsize, data)
  INTEGER modfil(3), start, bsize
  CHARACTER*(*) tblnm, group, field
  BYTE data(*)
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file.

tblnm IN: ASCII string name of the source table structure.

group IN: ASCII string name of the data group containing the source table structure. If set to ' ' (blank), the entire file will be searched for the table structure named *tblnm.*

field IN: Array of comma-delimited ASCII string table headers. The data from each field will appear in the same order as the headers.

start IN: Zero-based record location to begin reading the data from the table structure.

recno IN: Number of records to retrieve from the table structure.

bsize IN/OUT: *data* buffer size on input, in bytes. The buffer must be at least this size. *bsize* will normally return the number of bytes of data successfully retrieved. If the buffer is too small, however, the routine returns MFAIL and *bsize* will contain the size a buffer must be to contain the output data. It is set to 0 if a functional error occurs.

Output Parameters:

data OUT: the data buffer.

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
`getMODIStable unable continue without buffer size information.`	A location for *buffsize* information was not provided.
`getMODIStable unable to read from the tablename table with a NULL file MODFIL structure.`	
`getMODIStable unable to read from a table without a table name input.`	

Error Message:	Description:
`getMODIStable unable to read from the` *`tablename`* `table without a data buffer.`	
`getMODIStable cannot find` *`tablename`* `table.`	
`getMODIStable unable to find the` *`groupname`* `data group containing the` *`tablename`* `table.`	This may be preceded by one of the following two messages:
`searchMODISgroup fails to search object` *`objectname`* `in group` *`groupname`* `because Vattach fails.`	
`searchMODISgroup unable to find the specified Vgroup group` *`groupname`*`.`	
`getMODIStable cannot find the` *`tablename`* `table in the` *`groupname`* `data group.`	This may be preceded by the following message:
`searchMODISgroup fails to obtain` *`objectname`*`'s tag and reference number.`	The Vdata table could not be found in the specified Vgroup data group.
`getMODIStable detected FAIL from HDF procedure VSattach attempting to access the` *`tablename`* `table.`	
`getMODIStable unable to read data from the` *`tablename`* `table from invalid table structure record` *`start.`*	
`getMODIStable unable to read data from the` *`tablename`* `table from invalid table structure locations.`	Either access to some records or one or more fields requested do not exist in the table.
`getMODIStable detected FAIL from HDF procedure VSsetfields attempting to read` *`tablename`* `table.`	
`getMODIStable detected FAIL from HDF procedure VSsizeof attempting to read` *`tablename`* `table.`	
`getMODIStable detected FAIL from HDF procedure VSseek attempting to read` *`tablename`* `table.`	
`getMODIStable detected FAIL from HDF procedure VSread attempting to read` *`tablename`* `table.`	**buffsize* is set to 0 if any of the errors associated with the messages above occurs.

Error Message:	Description:
`getMODIStable detected FAIL from HDF procedure VSinquire.`	Should this error occur, getMODIStable will still return MAPIOK (because the data were successfully retrieved) and **buffsize* is set correctly.
`getMODIStable unable to fit <output size> bytes of` *`tablename`* `table's data into a` *`buffsize`* `byte output buffer.`	getMODIStable will not attempt to write to the *data* output buffer, but it will return the buffer length (in bytes) required to hold the requested records from the table.
`WARNING: Vgroup groupname contains non-exist Vdata object with reference ID` *`ref_id`*`.`	Information about a Vdata table that doesn't really exist has been found in the Vgroup data group being accessed. While this will not directly prevent reading the specified Vdata table, it does identify a probable defect in the HDF file.
`WARNING: getMODIStable retrieved dummy record from empty table` *`tablename.`*	The record retrieved from the table does not contain geophysical data. getMODIStable returns MAPIOK (0), however. This situation can only occur if NO geophysical data were written into the table or the single record in the Vdata was not written using M-API.

MODISsizeof (MSIZE)

Purpose: Returns the number of bytes required for a string of data types.

Description: M-API uses a set of standard strings to describe the data types stored in array and table structures. These strings are returned, for example, by the routine getMODISardims (GMARDM) to describe the data type of the target array structure. MODISsizeof (MSIZE) returns the number of bytes required to store a data type given this data type string. The input string may be a series of comma delimited data type strings, in which case the total number of bytes needed to store the record described by the string is returned.

C: `long int MODISsizeof(char *data_type)`

Input Parameters:

data_type IN: String of comma-delimited data types.

Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
"float64"
"char *"
```

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION MSIZE(dtype)
      CHARACTER*(*) dtype
```

Input Parameters:

dtype IN: String of comma-delimited data types.

Permitted FORTRAN data types:

```
'CHARACTER*(*)'
 'INTEGER*1'
 'UINTEGER*1'
 'INTEGER*2'
 'UINTEGER*2'
 'INTEGER*4'
 'UINTEGER*4'
 'INTEGER*8'
 'UINTEGER*8'
 'REAL*4'
 'REAL*8'
```

Output Parameters: N/A

Returns: Number of bytes required for a string of data types or MFAIL if the data type was not identified.

openMODISfile (OPMFIL)

Purpose: Opens a file for access by the M-API routines.

Description: openMODISfile (OPMFIL) opens a file and creates a HDF structure to allow the M-API routines access to the file. openMODISfile must be called to produce the MODFIL structure before any of these C routines can access it. Note that setting the file access to "w" creates a file and will overwrite a pre-existing one. openMODISfile will close the file and return null outputs if an error occurs.

C:

```
MODFIL * openMODISfile(char *filename, char *access)
```

Input Parameters:

filename IN: Complete path and filename for the file to be opened.
access IN: Standard C access mode.

Permitted access mode:
"r" Open for read only.
"w" Create for read/write.
"a" Open for read/write.

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION OPMFIL (fname, access, modfil)
      CHARACTER*(*)      fname, access
      INTEGER            modfil(MODFILLEN)
```

Input Parameters:

fname IN: Complete path and filename for the file to be opened.
access IN: Access mode.

One of:
'r' Open for read only.
'w' Create for read/write.
'a' Open for read/write.

Output Parameters:

modfil OUT: Array that is used to reference the file in all other M-API routines. The array will return all zeroes if an error occurs.

Returns: MODFIL structure that is used to reference the file in all other M-API C routines or NULL if an error occurs.

Error Message:	Description:
`openMODISfile unable to access a file without a filename input.`	
`openMODISfile unable to open file` *`filename`* `without access mode input.`	
`openMODISfile unable to allocate memory for a MODIS file structure for file` *`filename.`*	
`openMODISfile unable to recognize access type` *`access`* `to open file` *`filename`*`.`	
`openMODISfile unable to find file` *`filename.`*	
`openMODISfile detected FAIL from HDF procedure SDstart opening file` *`filename`*`.`	May be unable to open the HDF file because it is write-protected.
`openMODISfile detected NULL from HDF function SDIhandle_from_id accessing file` *`filename`*`.`	
`openMODISfile unable to allocate memory for the MODIS filename` *`filename`*`.`	

putMODISarinfo (PMARIN)

Purpose: Attach a local metadata attribute/value pair to a MODIS array.

Description: putMODISarinfo (PMARIN) stores an attribute = value(s) metadata pair to the indicated array. If the attribute already exists, the value(s) will be updated.

C:
```
int putMODISarinfo(MODFIL *file, char *arrayname, char *groupname,
char *attribute, char *data_type, long int n_elements, void *value)
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference a MODIS HDF file receiving the metadata.

arrayname IN: ASCII string name of the array. Provided macros for accepted MODIS HDF file array names are listed in Appendix A, M-API-Supplied Constants and Macros.

groupname IN: ASCII string name of the data group containing the array structure to which the metadata is attached. If set to NULL, the entire file will be searched for the array structure named *arrayname.*

attribute IN: Name to assign the attribute. Provided macros for accepted MODIS file metadata names are listed in Appendix A, M-API Supplied Constants and Macros.

data_type IN: Data Type of the value.

Permitted C data types:
```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
"float64"
"char *"
```

n_elements IN: Number of metadata values in *value*.

value IN: Address of the data to store in the attribute. If the attribute already exists, the value will be updated. Values should conform to the data types, formats and/or those values enumerated for the attribute in Appendix A, M-API-Supplied Constants and Macros.

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION PMARIN(modfil, arrnm, group, attr, dtype, nms,
value)
      INTEGER      modfil(3), nms
      CHARACTER*(*)      group, arrnm, attr, dtype
      BYTE         value(*)
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file containing the target array structure.

arrnm IN: ASCII string name of the array. Provided macros for accepted MODIS HDF file array names are listed in Appendix A, M-API-Supplied Constants and Macros.

group IN: ASCII string name of the data group containing the array structure to which the metadata is attached. If group = ' ' (blank), the entire file will be searched for the array structure named *arrnm.*

attr IN: Name to assign the attribute. Provided macros for accepted MODIS file metadata names are listed in Appendix A, M-API-Supplied Constants and Macros.

dtype IN: Data Type of the value.

Permitted FORTRAN data types:

```
'CHARACTER*(*)'
 'INTEGER*1'
 'UINTEGER*1'
 'INTEGER*2'
 'UINTEGER*2'
 'INTEGER*4'
 'UINTEGER*4'
 'REAL*4'
 'REAL*8'
```

nms IN: Number of metadata values in *value*.

value IN: The data to store in the attribute. If the attribute already exists, the value will be updated. Values should conform to the data types, formats and/or those values enumerated for the attribute in Appendix A, M-API-Supplied Constants and Macros.

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
putMODISarinfo unable to write an array attribute without an attribute name input.	
putMODISarinfo unable to write the *attribute* array attribute without data type information.	
putMODISarinfo unable to write the *attribute* array attribute without the value buffer.	
putMODISarinfo unable to write the *attribute* array attribute without the name of the array it is associated with.	No *arrayname* argument was provided.
putMODISarinfo unable to write *n_elements attribute* array attribute values.	
putMODISarinfo unable to write the *attribute* array attribute in a file opened for read only.	
putMODISarinfo cannot find array *arrayname*.	
putMODISarinfo unable to find the *groupname* data group containing the *arrayname* array.	This may be preceded by one of the following three messages:
searchMODISgroup fails to search object *objectname* in group *groupname* because Vattach fails.	
searchMODISgroup unable to find the specified Vgroup group *groupname*.	
searchMODISgroup fails to obtain *objectname*'s tag and reference number.	
putMODISarinfo cannot find the *arrayname* array in the *groupname* data group.	The SDS array structure could not be found in the specified Vgroup data group.
putMODISarinfo unable to write the *attribute* array attribute with a *size* byte value.	Each HDF attribute is limited to 32K of memory.
putMODISarinfo unable to write the *attribute* array attribute of data type *data_type*.	
putMODISarinfo detected FAIL from HDF procedure SDselect attempting to write the *attribute* array attribute.	
putMODISarinfo detected FAIL from HDF procedure SDsetattr attempting to write the *attribute* array attribute.	

Error Message:	Description:
`putMODISarinfo detected FAIL from HDF procedure SDendaccess attempting to write the` *`attribute`* `array attribute`	
`WARNING: Vgroup groupname contains non-existing SDS object with reference ID` *`ref_id`*`.`	Information about an SDS array structure that doesn't really exist has been found in the Vgroup data group being accessed. While this will not directly prevent writing the specified local array attribute, it does identify a probable defect in the HDF file.

putMODISarray (PMAR)

Purpose: Inserts an array or subarray of data into a MODIS HDF file array structure.

Description: putMODISarray (PMAR) places a multi-dimensional array of data into an HDF SDS array structure previously created using createMODISarray (CRMAR). The data in the array must be of the data type for which the target array structure was created. In addition, the dimensions and placement of the input array in the array structure must be consistent with the structure's rank and dimensions. If a putMODISarray error message occurs, the data insertion will not be performed. This routine may be called multiple times to fill the array structure. Data previously in the array structure may be overwritten with this routine.

The *groupname* string provides the facility to select an array structure placed in a particular HDF 'Vgroup' data group. The entire file will be searched for an array structure named *arrayname* if *groupname* = NULL in C or *group* is a blank string (' ') in FORTRAN.

C:
```
int putMODISarray(MODFIL *file,  char *arrayname, char *groupname, long
int start[],  long int dimsizes[], void *data)
```

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file containing the target array structure.
arrayname	IN:	ASCII string name of the target array structure.
groupname	IN:	ASCII string name of the data group containing the target array structure. If set to NULL, the entire file will be searched for the array structure named *arrayname.*
start	IN:	Array containing the array structure location to begin placing the data into the array structure. *start* must have the same number of elements as the target array has dimensions.
dimsizes	IN:	Array describing the size of the array being inserted into the array structure. *dimsizes* must have the same number of elements as the target array structure has dimensions. The product of the array dimensions must equal the number of elements in *data.*
data	IN:	Address of the data buffer.

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION PMAR (modfil, arrnm, group, start, dims, data)
      INTEGER           modfil(MODFILLEN), start(*), dims(*)
      CHARACTER*(*)     arrnm, group
      BYTE              data(*)
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file containing the target array structure.

arrnm IN: ASCII string that will be the name of the array.

group IN: ASCII string name of the data group to place containing the target array structure. If *group* = ' ' (blank), the entire file will be searched for the array structure named *arrayname*.

start IN: Array containing the array structure location to begin placing the data into the array structure. *start* must have the same number of elements as the target array has dimensions.

dims IN: Array describing the size of the array being inserted into the array structure. *dims* must have the same number of elements as the target array structure has dimensions. The product of the array dimensions must equal the number of elements in *data.*

data IN: Multi-dimensional data array.

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	**Description:**
`putMODISarray unable to write to the` *`arrayname`* `array with a NULL file MODFIL structure.`	
`putMODISarray unable to write to an array without an array name input.`	
`putMODISarray unable to write to the` *`arrayname`* `array without array dimension input.`	
`putMODISarray unable to write to the` *`arrayname`* `array without a data buffer.`	
`putMODISarray unable to write to the` *`arrayname`* `array in file opened for read only.`	
`putMODISarray cannot find the` *`arrayname`* `array.`	

Error Message:	Description:
putMODISarray cannot find the *arrayname* array in the *groupname* data group.	
putMODISarray unable to find the *groupname* data group containing the *arrayname* array.	
putMODISarray unable to write data to invalid array structure locations in the *arrayname* array.	This error message may be preceded by one of the following two messages:
SDS_footprintOK detected FAIL from HDF procedure SDgetinfo.	
SDS_footprintOK unable to access data at invalid array structure locations *"start[0] ... start[r]".*	
putMODISarray detected FAIL from HDF procedure SDselect while attempting to write to the *arrayname* array.	
putMODISarray detected FAIL from HDF procedure SDgetinfo while attempting to write to the *arrayname* array.	
putMODISarray detected FAIL from HDF procedure SDwritedata while attempting to write to the *arrayname* array.	
putMODISarray detected FAIL from HDF procedure SDendaccess while attempting to write to the *arrayname* array.	

putMODISdiminfo (PMDMIN)

Purpose: Attach a local attribute/value pair to a specific dimension of a MODIS array.

Description: putMODISdiminfo (PMDMIN) stores an attribute = value(s) attribute pair to the indicated dimension of a MODIS array. If the attribute already exists, the value(s) will be updated.

C:

```
int putMODISdiminfo(MODFIL *file, char *arrayname, char *groupname, long
int dimension, char *attribute, char *data_type, long int n_elements,
void *value)
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference a MODIS HDF file receiving the attribute.

arrayname IN: ASCII string name of the array. Provided macros for accepted MODIS HDF file array names are listed in Appendix A, M-API-Supplied Constants and Macros.

groupname IN: ASCII string name of the data group containing the array structure to which the attribute is attached. If set to NULL, the entire file will be searched for the array structure named *arrayname.*

dimension IN: The dimension to which the attribute is attached (0-based).

attribute IN: Name to assign the attribute. Provided macros for accepted MODIS file attribute names are listed in Appendix A, M-API-Supplied Constants and Macros.

data_type IN: Data type of the value.

Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
"float64"
"char *"
```

n_elements IN: Number of attribute values in *value*. If the *data_type* is char*, this routine will get the *n_elements* by calling strlen. In this case, this argument can be set to the any positive number.

value IN: Address of the data to store in the attribute. If the attribute already exists, the value will be updated. Values should conform to the data types, formats and/or those values enumerated for the attribute in Appendix A, M-API-Supplied Constants and Macros.

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION PMDMIN(modfil, arrnm, group, dim, attr, dtype,
nms, value)
      INTEGER     modfil(MODFILLEN), dim, nms
      CHARACTER*(*) group, arrnm, attr, dtype
      BYTE        value(*)
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file containing the target array structure.

arrnm IN: ASCII string name of the array. Provided macros for accepted MODIS HDF file array names are listed in Appendix A of the User's Guide, M-API-Supplied Constants.

group IN: ASCII string name of the data group containing the array structure to which the attribute is attached. If *group* = ' ' (blank), the entire file will be searched for the array structure named *arrnm.*

dim IN: The dimension to which the attribute will be attached (0-based).

attr IN: Name to assign the attribute. M-API provided macros for accepted MODIS file attribute names are listed in Appendix A, M-API-Supplied Constants and Macros.

dtype IN: Data type of the value.

Permitted FORTRAN data types:

```
'CHARACTER*(*)'
 'INTEGER*1'
 'UINTEGER*1'
 'INTEGER*2'
 'UINTEGER*2'
 'INTEGER*4'
 'UINTEGER*4'
 'REAL*4'
 'REAL*8'
```

nms IN: Number of attribute values in *value*.

value IN: The data to store in the attribute. If the attribute already exists, the value will be updated. Values should conform to the data types, formats and/or those values

enumerated for the attribute in Appendix A, M-API-Supplied Constants and Macros.

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
`putMODISdiminfo unable to write a dimension attribute without an attribute name input.`	
`putMODISdiminfo unable to write the` *`attribute`* `dimension attribute without data type information.`	
`putMODISdiminfo unable to write the` *`attribute`* `dimension attribute without the value buffer.`	
`putMODISdiminfo unable to write the` *`attribute`* `dimension attribute without the name of the array it is associated with.`	No *arrayname* argument was provided.
`putMODISdiminfo unable to write` *`n_elements attribute`* `dimension attribute values.`	
`putMODISdiminfo unable to write the` *`attribute`* `dimension attribute in a file opened for read only.`	
`putMODISdiminfo cannot find array` *`arrayname`*`.`	
`putMODISdiminfo unable to find the` *`groupname`* `data group containing the` *`arrayname`* `array.`	This may be preceded by one of the following three messages:
`searchMODISgroup fails to search object` *`objectname`* `in group` *`groupname`* `because Vattach fails.`	
`searchMODISgroup unable to find the specified Vgroup group` *`groupname`*`.`	
`searchMODISgroup fails to obtain` *`objectname`*`'s tag and reference number.`	
`putMODISdiminfo cannot find the` *`arrayname`* `array in the` *`groupname`* `data group.`	The SDS array structure could not be found in the specified Vgroup data group.
`putMODISdiminfo unable to write the` *`attribute`* `dimension attribute with a` *`size`* `byte value.`	Each HDF attribute is limited to 32K of memory.

Error Message:	Description:
putMODISdiminfo unable to write the *attribute* dimension attribute of data type *data_type*.	
putMODISdiminfo detected FAIL from HDF procedure SDselect attempting to write the *attribute* dimension attribute.	
putMODISdiminfo detected FAIL from HDF procedure SDgetinfo attempting to write the *attribute* dimension attribute.	
putMODISdiminfo detected FAIL from HDF procedure SDselect attempting to write the *attribute* dimension attribute.	
putMODISdiminfo unable to write the *attribute* attribute to non-existing dimension *dimension* of the *arrayname* array.	
putMODISdiminfo detected FAIL from HDF procedure SDgetdimid attempting to write the *attribute* dimension attribute.	
putMODISdiminfo detected FAIL from HDF procedure SDsetattr attempting to write the *attribute* dimension attribute.	
putMODISdiminfo detected FAIL from HDF procedure SDendaccess attempting to write the *attribute* dimension attribute.	
WARNING: Vgroup groupname contains non-existing SDS object with reference ID *ref_id*.	Information about an SDS array structure that doesn't really exist has been found in the Vgroup data group being accessed. While this will not directly prevent writing the specified local dimension attribute, it does identify a probable defect in the HDF file.

putMODISdimname (PMDNAM)

Purpose: Assigns a name to a specific dimension of an array structure..

Description: putMODISdimname (PMDNAM) associates an HDF dimension name with a specified SDS array structure dimension. The SDS array must be created [using createMODISarray (CRMAR)] before it is possible to name any of its dimensions. This routine does not create a "long_name" dimension attribute. However, putMODISdimname can produce such a dimension label.

putMODISdimname does more than apply a name to a dimension. An HDF dimension name is an independent data object. It may be shared by several array structure dimensions, but they all must be of the same size. Any dimension attribute that is associated with any one of these dimensions is immediately associated with all the dimensions having that name. Likewise, updating a dimension attribute for one dimension updates it for all dimensions having the same name (they could only have one "long name" dimension shared between them).

Naming an SDS dimension will also cause any dimension attributes currently associated with that dimension to be lost. Therefore, it is most practical to name an array's dimensions, if necessary, immediately after the array structure's creation and before creating dimension attributes for it.

The *groupname* string provides the facility to select an array structure placed in a particular HDF 'Vgroup' data group. Alternatively, the entire file will be searched for an array structure named *arrayname* if the argument *groupname* = NULL in C or *group* is a blank string (' ') in FORTRAN.

C:
```
int putMODISdimname(MODFIL *file, char *arrayname, char *groupname, long
int dimension, char *dimname)
```

Input parameters:

file	IN:	Address of MODFIL structure that is used to reference a MODIS HDF file receiving the dimension name.
arrayname	IN:	ASCII string name of the target array structure.
groupname	IN:	ASCII string name of the data group containing the target array structure. If set to NULL, the entire file will be searched for the array structure named *arrayname.*
dimension	IN:	The dimension number which is to be named (0-based). The 0 dimension of an HDF SDS array structure is associated with the least rapidly varying array index.
dimname	IN:	ASCII string name to give to the dimension.

Output parameters: N/A

FORTRAN:

```
INTEGER FUNCTION PMDNAM(modfil, arrnm, group, dim, dnm)
INTEGER modfil(MODFILLEN), dim
CHARACTER *(*) arrnm, group, dnm
```

Input parameters:

modfil	IN:	Array that is used to reference the MODIS HDF file receiving the dimension name.
arrnm	IN:	ASCII string name of the target array structure.
group	IN:	ASCII string name of the data group containing the target array structure. If *group* = ' ' (blank) the entire file will be searched for the array structure named *arrnm.*
dim	IN:	The dimension to be named (0-based). The 0 dimension of an HDF SDS array structure is associated with the most rapidly varying array index.
dnm	IN:	ASCII string name to gove to the dimension.

Output parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	**Description:**
putMODISdimname unable to name a dimension without a dimension name input.	
putMODISdimname unable to name a dimension *dimname* without the name of array the dimension is associated with.	
putMODISdimname unable to name a dimension *dimname* in the *arrayname* array with an invalid MODIS file structure input.	
putMODISdimname unable to name a dimension *dimname* in a file opened for read only.	
putMODISdimname detected MFAIL from M-API internal function getMODISarrayid while attempting to name a dimension *dimname* in the *arrayname* array.	
putMODISdimname unable to name a non-existing dimension *dimension dimname*.	
putMODISdimname detected FAIL from HDF procedure SDgetdimid attempting to name a dimension *dimname*.	

putMODISdimname detected FAIL from HDF procedure SDdiminfo attempting to name a dimension *dimname*.

WARNING: putMODISdimname detected *nattrs* attributes currently attached to the dimension. Naming the *dimension* dimension of the *arrayname* array *dimname* will lose those attributes.

putMODISdimname detected FAIL from HDF procedure SDsetdimname attempting to name a dimension *dimname*.

putMODISfileinfo (PMFIN)

Purpose: Writes a MODIS HDF file metadata attribute/value pair to a file.

Description: putMODISfileinfo (PMFIN) stores an attribute = value(s) attribute pair to the indicated MODIS HDF file. If the attribute already exists, the value(s) will be updated.

C:
```
int putMODISfileinfo(MODFIL *file, char *attribute, char *data_type,
long int n_elements, void *value)
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference a MODIS HDF file receiving the metadata.

attribute IN: Name to assign the attribute. Provided macros for accepted MODIS file metadata names are listed in Appendix A, M-API-Supplied Constants and Macros.

data_type IN: Data type of the value.

Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"float32"
"float64"
"char *"
```

n_elements IN: Number of metadata values to extract from *value*.

value IN: Address of the data to store in the attribute. If the *attribute* already exists, the *value* will be updated. Values should conform to the data types, formats and/or those values enumerated for the *attribute* in Appendix A, M-API-Supplied Constants and Macros.

Output Parameters: N/A

FORTRAN:
```
INTEGER FUNCTION PMFIN(modfil, attr, dtype, nms, value)
      INTEGER           modfil(3), nms
      CHARACTER*(*)     attr, dtype
      BYTE              value(*)
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file.

attr IN: Name to assign the attribute. Provided macros for accepted MODIS file metadata names are listed in Appendix A, M-API Supplied Constants and Macros.

dtype IN: Data Type of the value.

Permitted FORTRAN data types:

```
'CHARACTER*(*)'
 'INTEGER*1'
 'UINTEGER*1'
 'INTEGER*2'
 'UINTEGER*2'
 'INTEGER*4'
 'UINTEGER*4'
 'REAL*4'
 'REAL*8'
```

nms IN: Number of metadata values to extract from *value*.

value IN: The data to store in the attribute. If the attribute already exists, the value will be updated. Values should conform to the data types, formats and/or those values enumerated for the attribute in Appendix A, M-API-Supplied Constants and Macros.

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	**Description:**
`putMODISfileinfo unable to continue with empty input.`	
`putMODISfileinfo unable to store` *`n_elements attribute`* `global attribute values.`	
`putMODISfileinfo unable to write metadata in file opened for read only.`	
`putMODISfileinfo unable to identify data type "`*`data`*`type".`	
`putMODISfileinfo unable to write` *`attribute`* `metadata with a` *`size`* `byte value.`	
`putMODISfileinfo detected FAIL from HDF procedure SDsetattr.`	

putMODIStable (PMTBL)

Purpose: Inserts data records into a MODIS HDF file table structure.

Description: putMODIStable (PMTBL) places one or more data records into an HDF Vdata table structure previously created using createMODIStable (CRMTBL). The data to be inserted into the table must be inserted into a buffer string. The length of this string must be an integral number of the table structure's record length. The various data that make up a record should be inserted into the buffer in the same order as the field headers were ordered in the createMODIStable call. This routine may be called multiple times to fill the table structure. Data previously in the table structure may be overwritten. It should be noted that putMODIStable (PMTBL) can only write or overwrite a complete Vdata record. It is not possible to write or overwrite an individual field or a subset of a Vdata record.

An empty Vdata may not be created, so a dummy record will be inserted. This dummy record will be overwritten with the first call from putMODIStable. If this initial write is performed with a single record, an ambiguity would exist whether this record was actual data or the dummy. The ambiguity is resolved by the creation of a semaphore metadata to indicate that the dummy record has been overwritten.

The *groupname* string provides the facility to select a table structure placed in a particular HDF 'Vgroup' data group. The entire file will be searched for a table structure named *tablename* if *groupname* = NULL in C or *group* = ' ' (blank) in FORTRAN.

C: `int putMODIStable(MODFIL *file, char *tablename, char *groupname, long int start, long int recno, unsigned char *data)`

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file containing the target table structure.
tablename	IN:	ASCII string name of the target table structure.
groupname	IN:	ASCII string name of the data group containing the target table structure. If set to NULL, the entire file will be searched for the table structure named *tablename.*
start	IN:	Zero-based record location to begin placing the data into the table structure. If *start* = -1, data records will be appended to the end of the table structure.
recno	IN:	Number of records being inserted into the table structure. The product of *recno* and the table structure's

record length must have the same length as the buffer addressed by *data*

data IN: Address of the data buffer.

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION PMTBL(modfil, tblnm, group, start, recno, data)
      INTEGER modfil(3)
      CHARACTER*(*) tblnm, group
      INTEGER start, recno
      BYTE data(*)
```

Input Parameters:

modfil IN: Array that is used to reference the MODIS HDF file.

tblnm IN: ASCII string name of the table structure.

group IN: ASCII string name of the data group containing the target structure. If set to ' ' (blank), the entire file will be searched for the table structure named *tblnm.*

start IN: Zero-based record location to begin putting the data into the table structure. If *start* = -1, data records will be appended to the end of the table structure.

recno IN: Number of records to put into the table structure.

data IN: Address of the data buffer.

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
putMODIStable unable to write to the *tablename* table with a NULL file MODFIL structure.	
putMODIStable unable to write to a table without an table name input.	
putMODIStable unable to write to the *tablename* table without table dimension input.	
putMODIStable unable to write to the *tablename* table without a data buffer.	
putMODIStable unable to write to the *tablename* table in file opened for read only.	
putMODIStable cannot find the *tablename* table.	

Error Message:	Description:
putMODIStable cannot find the *tablename* table in the *groupname* data group.	
putMODIStable unable to find the *groupname* data group containing the *tablename* table.	
putMODIStable detected FAIL from HDF procedure Vattach while attempting to write to the *tablename* table.	
putMODIStable detected FAIL from HDF procedure VSattach while attempting to write to the *tablename* table.	
putMODIStable detected FAIL from HDF procedure VSinquire while attempting to write to the *tablename* table.	
putMODIStable unable to place *datasize* bytes of data into *recno record size* byte records in *tablename* table.	
putMODIStable unable to write data to table *tablename* to invalid table structure record *start.*	The *start* location must be contiguous to the location of records already in the table.
putMODIStable detected FAIL from HDF procedure VSseek while attempting to write to the *tablename* table.	
putMODIStable detected FAIL from HDF procedure VSwrite while attempting to write to the *tablename* table.	
putMODIStable detected FAIL from M-API procedure set_Vhasdata while attempting to write to the *tablename* table. Sometimes it is necessary for putMODIStable to read from the table structure before writing to it. The following two error messages may occur in these circumstances: putMODIStable memory allocation failure while attempting to write to the *tablename* table. putMODIStable detected FAIL from HDF procedure VSread while attempting to write to the *tablename* table.	The first record has successfully been written to the table, however M-API was unable to write an associated attribute into the file. This will cause a write to the table which will inadvertantly overwrite the first one.

releaseMAPIfilehandle (RMFH)

Purpose: Releases a M-API file ID handle, writes accessed objects to a file, and frees memory allocated by M-API.

Description: releaseMAPIfilehandle (RMFH) releases the MODFILE structure created by the routine createMAPIfilehandle (CMFH). release MAPIfilehandle also automatically closes all the objects opened by the M-API routines. The M-API file handle created by calling createMODISfilehandle should not be passed to the M-API routines closeMODISfile (CLMFIL) or completeMODISfile (CPMFIL). Instead, it should be passed to releaseMAPIfilehandle which will write the objects to the file and free the memory.

Users may open an HDF-EOS file using an HDF-EOS open routine, call createMAPIfilehandle to create the M-API filehandle and use M-API routines to access data object(s) in the opened file. Once all object access is completed, call releaseMAPIfilehandle to release this file handle created with createMAPIfilehandle. This also automatically closes all the objects opened with M-API routines. Finally, call a HDF-EOS close routine to close the file.

C: `int releaseMAPIfilehandle (MODFIL ** file)`

Input Parameters:

***file* IN: Address of pointer to MODFILE structure.

Output Parameters: N/A

FORTRAN:
```
INTEGER FUNCTION RMFH (modfil)
      INTEGER      modfil (MODFILLEN)
```

Input Parameters:

modfil IN: Array that is used to reference a MODIS HDF file in all other M-API routines.

Output Parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Message:	Description:
endMODISobjaccess while trying to close all opened objects for file.	An error occured when trying to write the accessed objects to the file.
releaseMAPIfilehandle can not release a NULL file handle	An invalid M-API filehandle was passed to the routine.

substrMODISECSinfo (SMECIN)

Purpose: Decomposes a multiple character string ECS metadata (retrieved by getMODISECSinfo) into individual strings.

Description: ECS metadata values may be integer, floating point, or character string values or arrays of values. Some may be multiple strings. The routine getMODISECSinfo (GMECIN) retrieves such strings into a one-dimension character array with the individual strings separated by nulls ('\0'). substrMODISECSinfo (SMECIN) breaks this 'packed' character array into its constituent substrings. substrMODISECSinfo (SMECIN) sets the pointers in a provided output array to the beginning of each substring in the *char_value* array.

C: `int substrMODISECSinfo(`*`char *char_value`*`, long int `*`n_elements`*`, long int *`*`n_strings`*`, char *`*`substr[]`*`)`

Input Parameters:

char_value IN: Character string containing the 'packed' multiple substrings of ECS metadata retrieved with getMODISECSinfo.

Do not deallocate *char_value* until *substr* array gets correct values.

n_elements IN: The composite output dimensions, from getMODISECSinfo, containing (in the case of character string metadata) the total length (in bytes) of the string in char_value in its lower two bytes and the number of substrings packed into char_value - 1 in the upper two bytes. The calculations

n_strings = n_elements/65536 + 1

n_bytes = n_elements%65536

provide the number of substrings and the total length, respectively, of the data in *char_value.* When there is only one string in *char_value, n_elements* will be less than 65536 and there is no need to use substrMODISECSinfo.

n_strings IN/OUT: Address of the number of pointers available in the *substr* array. The *substr* pointers will not be set to the substrings in *char_value* unless there are sufficient pointers available in the pointer array. substrMODISECSinfo replaces this input with the number of substring pointers that have been set string address in the *char_value* array. **n_strings* will be set to 0 if a function error occurs. This argument must not be the address of a constant.

Output parameters:

substr OUT: Array of pointers to the constituent substrings contained in the *char_value* array.

FORTRAN:

```
INTEGER FUNCTION SMECIN (cvalue, nelmnt, nstrs, substr)
      INTEGER     nelmnt, nstr
      CHARACTER*(*)     cvalue, substr(*)
```

Input Parameters:

cvalue IN: Character string containing the 'packed' multiple substrings of ECS metadata retrieved with GMECIN.

nelmnt IN: The composite output dimensions, from GMECIN, containing (in the case of character string metadata) the total length (in bytes) of the string in *cvalue* in its lower two bytes and the number of substrings packed into *cvalue* less one in the upper two bytes. The calculations

n_strings = n_elements/65536 + 1

n_bytes = n_elements%65536

provide the number of substrings and the total length, respectively, of the data in *cvalue*. When there is only one string in *cvalue*, *nelmnt* will be less than 65536 and there is no need to use SMECIN.

nstrs IN/OUT: Number of elements available in the *substr* array. The *substr* will not be set to the substrings in *cvalue* unless there are sufficient elements available in the *substr* array. SMECIN replaces this input with the number of substrings parsed from the *cvalue* array. *nstrs* will be set to 0 if a function error occurs.

Output parameters:

substr OUT: Array of substrings obtained from the*cvalue*. array.

Returns: MAPIOK (0) if the address of each substring in *char_value* is in the *substr* pointer array, MFAIL (-1) if *substr* contains insufficient pointers for the addresses of all of the substrings in *char_value* or an error occurs.

Error Message:	Description:
substrMODISECSinfo unable to continue without char_value input.	
substrMODISECSinfo unable to continue without n_strings input.	
substrMODISECSinfo unable to continue without substr pointer array.	
substrMODISECSinfo unable to continue with invalid *n_elements* n_elements.	
substrMODISECSinfo unable to fit *loc_n_strings* substrings into **n_strings* pointers *substr* array.	
substrMODISECSinfo detected MFAIL from M-API procedure parse_string attempting to parse the *char_value* char_value.	

(This page intentionally left blank.)

4. M-API INTERNAL AND LOW LEVEL ROUTINE DESCRIPTIONS

4.1 Prototype Listing for Internal Routines

Table 4-1 is a prototype listing of the internal routines.

Table 4-1. FORTRAN to C Interface Routine Prototype Listing

Return Value	Routine Name [Parameter(s)]
void	cadmgrp (intf *modfil[MODFILLEN]*, _fcd *group,* intf **lgr*, _fcd *classname,* intf **lcl,* intf **tag*, intf **ref*, intf **ret*)
void	cclmfil (intf *modfil[MODFILLEN]* , intf **ret*)
void	ccmfh(*intf *swfid, intf modfil[MODFILLEN], intf *ret*)
void	ccpmfil (intf *modfil[MODFILLEN]*, _fcd *mdHandles*, _fcd *HDFattrnms*, intf **NumHandles*, intf **ret*)
void	ccrmar (intf **modfil*, _fcd *arrnm*, intf **lar*, _fcd *group*, intf **lgr*, _fcd *dtype*, intf **ldt*, intf **rank*, intf **dims*, intf **ret*)
void	ccrmgrp (intf *modfil[MODFILLEN]*, _fcd *group,* intf **lgr*, _fcd *classname,* intf **lcl,* intf **ret*)
void	ccrmtbl (intf **modfil*, _fcd *tblnm*, intf **ltb*, _fcd *class*, intf **lcl*, _fcd *group*, intf **lgr*, _fcd *field*, intf **lfi*, _fcd *dtype*, intf **ldt*, intf **ret*)
void	cemobj(intf **modfil*, _fcd *name*, intf **lna*, _fcd *group*, intf **lgr*, intf **type*, intf **ret*)
void	cgmar(intf *modfil[]*, _fcd *arrnm*, intf **lar*, _fcd *group*, intf **lgr*, intf *start[],* intf *dims[],* void **data*, intf **ret*)
void	cgmardm (intf **modfil*, _fcd *arrnm*, intf **lar*, _fcd *group*, intf **lgr*, _fcd *dtype*, intf **ldt*, intf **rank*, intf *dims[]*, intf **ret*)
void	cgmarin (intf *modfil[MODFILLEN]* , _fcd *arrnm*, intf **lar*, _fcd *group,* intf **lgr*, _fcd *attr,* intf **lat,* _fcd *dtype*, intf **ldt*, intf **nms*, VOIDP *value*, intf **ret*)
void	cgmdmin (intf *modfil[]*, _fcd *arrnm*, intf **lar*, _fcd *group*, intf **lgr*, intf **dim,* _fcd *attrnm*, intf **lat*, _fcd *dtype*, intf **ldt*, intf **nms,* void **value*, intf **ret*)
void	cgmdnam (intf **modfil*, _fcd *arrnm*, intf **lar*, _fcd *group*, intf **lgr*, intf **dim*, _fcd *dname,* intf *ldn, intf **ret*)
void	cgmecin (intf *modfil*[MODFILLEN], _fcd *pvlname*, intf **lpv*, _fcd *pname*, intf **lpn*, _fcd *dtype*, intf **ldt*, intf **nms*, VOIDP *pvalue*, intf **ret*)
void	cgmfin (intf *modfil[MODFILLEN]*, _fcd *attr,* intf **lat,* _fcd *dtype*, intf **ldt*, intf **nms*, VOIDP *value*, intf **ret*)
void	cgmflds (intf **modfil*, _fcd *tblnm*, intf **ltb*, _fcd *group*, intf **lgr*, intf **strln*, intf **recno*, intf **fldno*, _fcd *fldnm*, intf **lfl*, _fcd *dtype*, intf **ldt*, _fcd *clss*, intf **lcl*, intf **ret*)
void	cgmhoid (intf **modfil*, _fcd *name*, intf **lna*, _fcd *group*, intf **lgr*, intf **type*, _fcd *access*, intf **lac*, intf **ret*)

Return Value	Routine Name [Parameter(s)]
void	cgmtbl (intf *modfil, _fcd tblnm, intf *ltb, _fcd group, intf *lgr, _fcd fldnm, intf *lfl, intf *start, intf *recno, intf *bsize, VOIDP data, intf *ret)
void	copmfil (_fcd fname, intf *lfn, _fcd access, intf *lac, intf modfil[MODFILLEN], intf *ret)
void	cpmar (intf modfil[], _fcd arrnm, intf *lar, _fcd group, intf *lgr, intf start[], intf *dims[], void *data, intf *ret)
void	cpmarin (intf modfil[MODFILLEN], _fcd arrnm, intf *lar, _fcd group, intf *lgr, _fcd attr, intf *lat, _fcd dtype, intf *ldt, intf *nms, VOIDP value, intf *ret)
void	cpmdmin(intf modfil[], _fcd arrnm, intf *lar, _fcd group, intf *lgr, intf *dim, _fcd attrnm, intf *lat, _fcd dtype, intf *ldt, intf *nms, void *value, intf *ret)
void	cpmdnam(intf *modfil, _fcd arrnm, intf *lar, _fcd group, intf *lgr, intf *dim, _fcd dname, intf *ldn, intf *ret)
void	cpmfin (intf modfil[MODFILLEN], _fcd attr, intf *lat, _fcd dtype, intf *ldt, intf *nms, VOIDP value, intf *ret)
void	cpmtbl (intf *modfil, _fcd tblnm, intf *ltb, _fcd group, intf *lgr, intf *start, intf *recno, VOIDP data, intf *ret)
void	crmfh (intf modfil[MODFILLEN], intf *ret)

4.2 M-API Internal Routines

The following are the explanations for the internal routines.

cadmgrp

Purpose: Provides a wrapping function interfacing between C and FORTRAN for addMODISgroup. This C function is only called by FORTRAN function ADMGRP. This function is a M-API internal routine.

Description: cadmgrp is a C function which is callable from FORTRAN. This function will call addMODISgroup to insert an object into a group structure. In M-API, cadmgrp is a low-level routine called only by SRMGRP.

In order to be callable from the FORTRAN in different platforms using function name cadmgrp, this function is called ncadmgrp in the actual C code. ncadmgrp is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncadmgrp will always be the object name of a FORTRAN function named cadmgrp.

C: `void cadmgrp(intf modfil[MODFILLEN], _fcd group, intf *lgr, _fcd classname, intf *lcl, intf *tag, intf *ref, intf *ret)`

Input Parameters:

modfil	IN:	FORTRAN array of MODIS file structure.
group	IN:	FORTRAN ASCII string of group name.
lgr	IN:	Address for the number of bytes in *groupname*.
classname	IN:	FORTRAN ASCII string of class name.
lcl	IN:	Address for the number of bytes in *classname*.
tag	IN:	The HDF tag of the object.
ref	IN	The HDF reference number of the object.

Output Parameters:

ret OUT: MAPIOK if successful, otherwise MFAIL

Returns: N/A

cclmfil

Purpose: Provides a wrapping function interfacing between C and FORTRAN for closeMODISfile. This C function is only called by FORTRAN function CLMFIL. This function is a M-API internal routine.

Description: cclmfil is a C function which is callable from FORTRAN. This function will call closeMODISfile to terminate the access of a MODIS HDF file. In M-API, cclmfil is a low-level routine called only by CLMFIL.

In order to be callable from the FORTRAN in different platforms using function name cclmfil, this function is called ncclmfil in the actual C code. ncclmfil is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncclmfil will always be the same as the object name of a FORTRAN function named cclmfil.

C: `void cclmfil (intf modfil[MODFILLEN] , intf *ret)`

Input Parameters:

modfil IN/OUT: FORTRAN MODIS file array. If all successful, the elements of the array will be filled with zero after closing the MODIS file.

Output Parameters:

ret OUT: if function is successful, ret is set to 0, otherwise -1

Returns: N/A

ccmfh

Purpose: Provides a wrapping function interfacing between C and FORTRAN. This C function is only called by the FORTRAN function CMFH. This function is a M-API internal routine.

Description: ccmfh is a C function which is callable from FORTRAN. This function will call createMAPIfilehandle to open/create a MODIS HDF file. In M-API, ccmfh is low-level routine which is called only by CMFH.

In order to be callable from the FORTRAN in different platforms using function name ccmfh, this function is called nccmfh in the actual C code. nccmfh is redefined in mapic.h according to compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of nccmfh will always be the object name of a FORTRAN function named ccmfh.

C: `void ccmfh(`*`intf *swfid, intf modfil[MODFILLEN], intf*ret`*`)`

Input Parameters:

swfid IN: FORTRAN integer for the HDF-EOS file ID handle.

Output Parameters:

modfil OUT: Array that is used to reference the file in all other M-API routines in FORTRAN. The array will return all zeroes if an error occurs.

ret OUT: If function is successful, *ret* is set to MAPIOK, otherwise MFAIL.

Returns: N/A

ccpmfil

Purpose: Provides a wrapping function interfacing between C and FORTRAN for completeMODISfile. This C function is only called by FORTRAN function CPMFIL. This function is a M-API internal routine.

Description: ccpmfil is a C function which is callable from FORTRAN. This function will call completeMODISfile to form an array structure. In M-API, ccpmfil is a low-level routine which is called only by CPMFIL.

In order to be callable from the FORTRAN in different platforms using function name ccpmfil, this function is called nccpmfil in the actual C code. nccpmfil is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of nccpmfil will always be the same as the object name of a FORTRAN function named ccpmfil.

C: `void ccpmfil(intf modfil[MODFILLEN], _fcd mdHandles, fcd HDFattrnms, intf *NumHandles, intf *ret)`

Input Parameters:

modfil	IN:	Array to reference a file in all M-API routines.
mdHandles	IN:	An array of character strings. The memory size of the array is PGSd_MET_NUM_OF_GROUPS * PGSd_MET_GROUP_NAME_L bytes where PGSd_MET_NUM_OF_GROUPS is 20 and PGSd_MET_GROUP_NAME_L is 50. Each string in the array stores a handle (ASCII name) to an internal ODL tree structure which will be written out as an HDF-EOS PVL global attribute. Each handle should be less than 50 characters. The maximum number of handles should be 20.
HDFattrnms	IN:	An array of character strings. The memory size of the array is PGSd_MET_NUM_OF_GROUPS * MAX_ECS_NAME_L bytes, where PGSd_MET_NUM_OF_GROUPS is 20 and MAX_ECS_NAME_L is 256. Each string in the array is a global attribute name for storing an HDF-EOS PVL text block which has a handle in the corresponding string in *mdHandles* array. Each name should be less than MAX_ECS_NAME_L characters.
NumHandles	IN:	Specifies the number of actual handles contained in *mdHandles* (excluding the MCF file which is the first item in mdHandles). This may be set from 0 to PGSd_MET_NUM_OF_GROUPS-1.

Output Parameters:

ret OUT: FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

ccrmar

Purpose: Provides a wrapping function interfacing between C and FORTRAN for createMODISarray. This C function is only called by FORTRAN function CRMAR. This function is a M-API internal routine.

Description: ccrmar is a C function which is callable from FORTRAN. This function will call createMODISarray to form an array structure. In M-API, ccrmar is a low-level routine which is called only by CRMAR.

In order to be callable from the FORTRAN in different platforms using function name ccrmar, this function is called nccrmar in the actual C code. nccrmar is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of nccrmar will always be the same as the object name of a FORTRAN function named ccrmar.

C:
```
void ccrmar(intf *modfil, _fcd arrnm, intf *lar, _fcd group,   intf
*lgr, _fcd dtype, intf *ldt, intf *rank, intf *dims, intf *ret)
```

Input Parameters:

modfil IN: FORTRAN integer array that is used to reference the MODIS HDF file receiving the new array.
arrnm IN: FORTRAN character string that will be the name of the array.
lar IN: FORTRAN integer address of the memory size of *arrnm.*
group IN: ASCII string name of the data group where the new array will be placed.
lgr IN: FORTRAN integer address of the memory size of *group.*
dtype IN: FORTRAN character string of data types for the array.
ldt IN: FORTRAN integer address of the memory size of *dtype.*
rank IN: FORTRAN integer address of the number of dimensions in the array
dims IN: The array containing the size of each dimension.

Output Parameters:

ret OUT: FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

ccrmgrp

Purpose: Provides a wrapping function interfacing between C and FORTRAN for createMODISgroup. This C function is only called by FORTRAN function CRMGRP. This function is a M-API internal routine.

Description: ccrmgrp is a C function which is callable from FORTRAN. This function will call createMODISgroup to create a group structure. In M-API, ccrmgrp is a low-level routine called only by CRMGRP.

In order to be callable from the FORTRAN in different platforms using function name ccrmgrp, this function is called nccrmgrp in the actual C code. nccrmgrp is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of nccrmgrp will always be the same as the object name of a FORTRAN function named ccrmgrp.

C:
```
int ccrmgrp(intf modfil[MODFILLEN], _fcd group, intf *lgr, _fcd classname,
intf *lcl, intf *ret)
```

Input Parameters:

modfil	IN:	FORTRAN array of MODIS file structure.
group	IN:	FORTRAN ASCII string of group name.
lgr	IN:	Address for the number of bytes in *group*.
classname	IN:	FORTRAN ASCII string of class name.
lcl	IN:	Address for the number of bytes in *classname*.

Output Parameters:

ret	OUT: MAPIOK if successful, otherwise MFAIL

Returns: N/A

ccrmtbl

Purpose: Provides a wrapping function interfacing between C and FORTRAN for createMODIStable. This C function is only called by FORTRAN function CRMTBL. This function is a M-API internal routine.

Description: ccrmtbl is a C function which is callable from FORTRAN. This function will call createMODIStable to form a table structure. In M-API, ccrmtbl is a low-level routine which is called only by CRMTBL.

In order to be callable from the FORTRAN in different platforms using function name ccrmtbl, this function is called nccrmtbl in the actual C code. nccrmtbl is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of nccrmtbl will always be the same as the object name of a FORTRAN function named ccrmtbl.

C:
```
void nccrmtbl(intf *modfil, _fcd tblnm, intf *ltb, _fcd class,  intf
*lcl, _fcd group, intf *lgr, _fcd field, intf *lfi, _fcd dtype, intf
*ldt, intf *ret)
```

Input Parameters:

modfil	IN:	Array of FORTRAN integer array that is used to reference the MODIS HDF file receiving the new table.
tblnm	IN:	FORTRAN character string that will be the name of the table.
ltb	IN:	FORTRAN integer address of the memory size of *tblnm.*
class	IN:	FORTRAN character string that will be the class name of the table.
lcl	IN:	FORTRAN integer address of the memory size of *class.*
group	IN:	ASCII string name of the data group where the new table will be placed.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
field	IN:	FORTRAN comma-delimited character string table headers.
lfi	IN:	FORTRAN integer address of the memory size of *field.*
dtype	IN:	FORTRAN character string of comma-delimited data types. for each table *field*.
ldt	IN:	FORTRAN integer address of the memory size of *dtype.*

Output Parameters:

ret OUT: FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

cemobj

Purpose: Provides a wrapping function interfacing between C and FORTRAN for endMODISobjaccess. This C function is only called by FORTRAN function EMOBJ. This function is a M-API internal routine.

Description: cemobj is a C function which is callable from FORTRAN. This function will call endMODISobjaccess to end the access to objects in a MODIS HDF file. In M-API, cemobj is low-level routine which is called only by EMOBJ.

In order to be callable from the FORTRAN in different platforms using function name cemobj, this function is called ncemobj in the actual C code. ncemobj is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncemobj will always be the object name of a FORTRAN function named cemobj.

C:
```
void cemobj (intf *modfil, _fcd name, intf *lna, _fcd group, intf *lgr,
             intf *type, intf *ret)
```

Input parameters:

modfil	IN:	FORTRAN integer array that is used to reference the MODIS HDF file.
name	IN:	FORTRAN character string that is the name of the object.
lna	IN:	FORTRAN integer address of the memory size of *name*.
group	IN:	FORTRAN character string which is the name of the data group to which the object (either an SDS or a Vdata) belongs.
lgr	IN:	FORTRAN integer address of the memory size of *group*.
type	IN:	FORTRAN integer address specifying the *type* of objects to be closed.

Output parameters:

ret	OUT:	FORTRAN integer address of the status (number of objects closed) or MFAIL if error occurs

Returns: N/A

cgmar

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODISarray. This C function is only called by FORTRAN function GMAR. This function is a M-API internal routine.

Description: cgmar is a wrapper which is callable from FORTRAN. This function will call getMODISarray to read array data. In M-API, cgmar is low-level routine which is called only by GMAR.

In order to be callable from the FORTRAN in different platforms using function name cgmar, this function is called ncgmar in the actual C code. ncgmar is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncgmar will always be the object name of a FORTRAN function named cgmar.

C:
```
void cgmar(intf modfil[], _fcd arrnm, intf *lar, _fcd group, intf *lgr,
intf start[], intf dims[], void *data, intf *ret)
```

Input parameters:

modfil	IN:	FORTRAN integer array that is used to reference the MODIS HDF file.
arrnm	IN:	FORTRAN character string that is the name of the array.
lar	IN:	FORTRAN integer address of the memory size of *arrnm.*
group	IN:	FORTRAN character string which is the name of the data group to which the array (SDS) belongs.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
start	IN:	FORTRAN integer array containing the array structure location to begin reading the data from the array structure. *start* must have the same number of elements as the target array has dimensions.
dims	IN:	FORTRAN integer array describing the size of the array being retrieved from ther array structure. *dims* must have the same number of elements as the target array structure has dimensions and the product of the array dimensions must equal the number of elements in *data.*

Output parameters:

data	OUT:	Multi-dimensional data array.
ret	OUT:	FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

cgmardm

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODISardims. This C function is only called by FORTRAN function GMARDM. This function is a M-API internal routine.

Description: cgmardm is a C function which is callable from FORTRAN. This function will call getMODISardims to read data from an array. In M-API, cgmardm is a low-level routine which is called only by GMARDM.

In order to be callable from the FORTRAN in different platforms using function name cgmardm, this function is called ncgmardm in the actual C code. ncgmardm is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncgmardm will always be the same as the object name of a FORTRAN function named cgmardm.

C:

```
void cgmardm(intf *modfil, _fcd arrnm, intf *lar, _fcd group,   intf
*lgr, _fcd dtype, intf *ldt, intf *rank, intf dims[],   intf *ret)
```

Input Parameters:

modfil IN: FORTRAN integer array that is used to reference the MODIS HDF file.

arrnm IN: FORTRAN character string that is the name of the array.

lar IN: FORTRAN integer address of the memory size of *arrnm.*

group IN: FORTRAN character string which is the name of the data group to which the array (SDS) belongs.

lgr IN: FORTRAN integer address of the memory size of *group.*

ldt IN: FORTRAN integer array to indicate the start location to retrieve.

dims IN: FORTRAN integer array to indicate the size for each dimension to retrieve.

Output Parameters:

dtype OUT: FORTRAN string of the data type of the array structure in HDF file.

rank IN/OUT: The address of a FORTRAN integer. Input with the number of elements in the *dims* array and output replaced with actual number of dimension in the array. If a function error occurs, the value is set to zero. If the *dims* is not large enough, this value will be set the actual rank in the array in the HDF and the function will return MFAIL.

dims OUT: FORTRAN integer array to place the size for each dimension to retrieve.

ret OUT: FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

cgmarin

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODISarinfo. This C function is only called by FORTRAN function GMARIN. This function is a M-API internal routine.

Description: cgmarin is a C function which is callable from FORTRAN. This function will call getMODISarinfo to get an array attribute. In M-API, cgmarin is a low-level routine called only by GMARIN.

In order to be callable from the FORTRAN in different platforms using function name cgmarin, this function is called ncgmarin in the actual C code. ncgmarin is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncgmarin will always be the same as the object name of a FORTRAN function named cgmarin.

C:

```
void cgmarin(intf modfil[MODFILLEN], _fcd arrnm, intf *lar, _fcd group, intf
*lgr, _fcd attr, intf *lat, _fcd dtype, intf *ldt, intf *nms, VOIDP value,
intf *ret)
```

Input Parameters:

modfil	IN:	FORTRAN array of MODIS file structure.
arrnm	IN:	FORTRAN ASCII string of array name.
lar	IN:	Address for the number of bytes in *arrnm*.
group	IN:	FORTRAN ASCII string of group name.
lgr	IN:	Address for the number of bytes in *group*.
attr	IN:	FORTRAN ASCII string of name of the attribute to be retrieved.
lat	IN:	Address for the number of bytes in *attr*.
dtype	IN/OUT:	FORTRAN ASCII string of data type.
ldt	IN:	Address for the number of bytes in *dtype*.
nms	IN:	Address for the number of elements can be stored in *value*. Output returns with the actual number of elements in *value*.

Output Parameters:

value	OUT: pointer to the data buffer.
ret	OUT: MAPIOK if successful, otherwise MFAIL

Returns: N/A

cgmdmin

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODISdiminfo. This C function is only called by FORTRAN function GMDMIN. This function is a M-API internal routine.

Description: cgmdmin is a wrapper which is callable from FORTRAN. This function will call getMODISdiminfo to read a dimensional attribute. In M-API, cgmdmin is low-level routine which is called only by PMAR.

In order to be callable from the FORTRAN in different platforms using function name cgmdmin, this function is called ncgmdmin in the actual C code. ncgmdmin is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncgmdmin will always be the object name of a FORTRAN function named cgmdmin.

C:
```
void cgmdmin(intf modfil[], _fcd arrnm, intf *lar, _fcd group, intf *lgr,
intf *dim, _fcd attrnm, intf *lat, _fcd dtype, intf *ldt, intf *nms, void
*value, intf *ret)
```

Input parameters:

modfil	IN:	FORTRAN integer array that is used to reference the MODIS HDF file.
arrnm	IN:	FORTRAN character string that is the name of the array.
lar	IN:	FORTRAN integer address of the memory size of *arrnm.*
group	IN:	FORTRAN character string which is the name of the data group to which the array (SDS) belongs.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
dim	IN:	The dimension from which the attribute will be read (0-based).
attrnm	IN:	Name of the attribute.
lat	IN:	FORTRAN address of the string length of *attrnm*
dtype	IN/OUT:	Data Type of the value.
ldt	IN:	FORTRAN address of the string length of *dtype*
nms	IN/OUT:	Number of attribute values in *value*.

Output parameters:

value	IN:	The data to store in the attribute.
ret	OUT:	FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

cgmdnam

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODISdimname. This C function is only called by FORTRAN function GMDNAM. This function is a M-API internal routine.

Description: cgmdnam is a wrapper which is callable from FORTRAN. This function will call getMODISdimname to read a dimension name. In M-API, cgmdnam is low-level routine which is called only by GMDNAM.

In order to be callable from the FORTRAN in different platforms using function name cgmdnam, this function is called ncgmdnam in the actual C code. ncgmdnam is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncgmdnam will always be the object name of a FORTRAN function named cgmdnam.

C:
```
void cgmdnam(intf *modfil, _fcd arrnm, intf *lar, _fcd group, intf *lgr,
intf *dim, _fcd dname, intf *ldn, intf *ret)
```

Input parameters:

modfil	IN:	FORTRAN integer array that is used to reference the MODIS HDF file.
arrnm	IN:	FORTRAN character string that is the name of the array.
lar	IN:	FORTRAN integer address of the memory size of *arrnm.*
group	IN:	FORTRAN character string which is the name of the data group to which the array (SDS) belongs.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
dim	IN:	FORTRAN integer address specifying the dimension.
ldn	IN:	FORTRAN integer address of the memory size of *dname.*

Output parameters:

dname	OUT:	FORTRAN string containing the dimension name.
ret	OUT:	FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

cgmecin

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODISECSinfo. This C function is only called by FORTRAN function GMECIN. This function is a M-API internal routine.

Description: cgmecin is a C function which is callable from FORTRAN. This function will call getMODISECSinfo to form a table structure. In M-API, cgmecin is a low-level routine which is called only by GMEATTR.

In order to be callable from the FORTRAN in different platforms using function name cgmecin, this function is called ncgmecin in the actual C code. ncgmecin is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncgmecin will always be the same as the object name of a FORTRAN function named cgmecin.

C:
```
void cgmecin(intf modfil[MODFILLEN], _fcd pvlname, intf *lpv,
_fcd pname, intf *lpn, _fcd dtype, intf *ldt, intf *nms, VOIDP pvalue,
intf *ret)
```

Input Parameters:

modfil	IN:	MODFIL file array that is used to reference the MODIS HDF file containing the target PVL attribute.
pvlname	IN:	FORTRAN ASCII string name of the HDF attribute which contains the PVL text block.
lpv	IN:	Number of bytes in *pvlname*.
pname	IN:	FORTRAN ASCII string name of a parameter whose value will be retrieved.
lpn	IN:	Number of bytes in *pname*
ldt	IN:	Number of bytes in *dtype.*
nms	IN/OUT:	Number of values in *pvalue*
dtype	IN/OUT:	Type of data in *pvalue*. There are only 3 data types: `'CHARACTER*(*) '` `'INTEGER32'` `'FLOAT32'` Users may also use 'FLOAT64', for which the function will convert the value from float32 to float64.

Output Parameters:

pvalue	OUT:	Buffer for the value. User should allocate enough memory for this buffer.
ret	OUT:	Return status of the call to C routine.

Returns: N/A

cgmfin

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODISfileinfo. This C function is only called by FORTRAN function GMFIN. This function is a M-API internal routine.

Description: cgmfin is a C function which is callable from FORTRAN. This function will call getMODISfileinfo to get a file (global) attribute. In M-API, cgmfin is a low-level routine called only by GMFIN.

In order to be callable from the FORTRAN in different platforms using function name cgmfin, this function is called ncgmfin in the actual C code. ncgmfin is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncgmfin will always be the same as the object name of a FORTRAN function named cgmfin.

C: `int cgmfin(intf modfil[MODFILLEN], _fcd attr, intf *lat, _fcd dtype, intf *ldt, intf *nms, VOIDP value, intf *ret)`

Input Parameters:

modfil IN: FORTRAN array of MODIS file structure.
attr IN: FORTRAN ASCII string of name of the attribute to be retrieved.
lat IN: Address for the number of bytes in *attr*.
dtype IN/OUT:FORTRAN ASCII string of data type.
ldt IN: Address for the number of bytes in *dtype*.
nms IN/OUT: Address for the number of elements can be stored in *value*. Output returns with the actual number of elements in *value*.

Output Parameters:

value OUT: pointer to the data buffer.
ret OUT: MAPIOK if successful, otherwise MFAIL

Returns: N/A

cgmflds

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODISfields. This C function is only called by FORTRAN function GMFLDS. This function is a M-API internal routine.

Description: cgmflds is a C function callable from FORTRAN. This function will call getMODISfields to get a table structure. In M-API, cgmflds is a low-level routine called only by FORTRAN function GMFLDS.

In order to be callable from the FORTRAN in different platforms using function name cgmflds, this function is called ncgmflds in the actual C code. ncgmflds is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncgmflds will always be the same as the object name of a FORTRAN function named cgmflds.

C:

```
void cgmflds(intf *modfil, _fcd tblnm, intf *ltb, _fcd group, intf *lgr,
intf *strln, intf *recno, intf *fldno, _fcd fldnm, intf *lfl,
_fcd dtype, intf *ldt, _fcd clss, intf *lcl, intf *ret)
```

Input Parameters:

modfil	IN:	Array of FORTRAN integer array that is used to reference the MODIS HDF file.
tblnm	IN:	FORTRAN character string of the name of the vdata (table).
ltb	IN:	FORTRAN integer address of the memory size of *tblnm.*
group	IN:	ASCII string name of the data group to which the table belongs.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
lfl	IN:	FORTRAN integer address of the memory size of *fldnm.*
ldt	IN:	FORTRAN integer address of the memory size of *dtype.*
lcl	IN:	FORTRAN integer address of the memory size of *clss.*

Output Parameters:

strln	OUT:	The minimum length required to store either *fldnm* or *dtype*. 0 if a function error occures.
recno	OUT:	FORTRAN integer address of number of records in the table.
fldno	OUT:	FORTRAN integer address of number of fields in the table.
fldnm	OUT:	FORTRAN comma-delimited character string table headers.
dtype	OUT:	FORTRAN character string of comma-delimited data types for each table field.
clss	OUT:	FORTRAN character string of table's class name.
ret	OUT:	FORTRAN interger address of the status (MFAIL, MAPIOK)

Returns: N/A

cgmhoid

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODIShobjid. This C function is only called by FORTRAN function GMHOID. This function is a M-API internal routine.

Description: cgmhoid is a C function which is callable from FORTRAN. This function will call getMODIShobjid to obtain the HDF ID for a data object in the MODIS file. In M-API, cgmhoid is low-level routine which is called only by GMHOID.

In order to be callable from the FORTRAN in different platforms using function name cgmhoid, this function is called ncgmhoid in the actual C code. ncgmhoid is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncgmhoid will always the object name of a FORTRAN function named cgmhoid.

```
C: void cgmhoid (intf *modfil, _fcd name, intf *lna, _fcd group, intf *lgr,
                 intf *type, _fcd access, intf *lac, intf *ret)
```

Input parameters:

modfil	IN: Array of FORTRAN integer array that is used to reference the MODIS HDF file.
name	IN: FORTRAN character string of the object name.
lna	IN: FORTRAN integer address of the memory size of *name.*
group	IN: ASCII string name of the data group to which the object belongs.
lgr	IN: FORTRAN integer address of the memory size of *group.*
type	IN: FORTRAN integer address of the object type.
access	IN: FORTRAN character string of the access type.
lac	IN: FORTRAN integer address of the memory size of *access*

Output parameters:

ret	OUT: FORTRAN integer address of the return value (id or MFAIL)

Returns: N/A

cgmtbl

Purpose: Provides a wrapping function interfacing between C and FORTRAN for getMODIStable. This C function is called only by FORTRAN function GMTBL. This function is a M-API low-level internal routine.

Description: cgmtbl is a C function callable from FORTRAN. This function will call getMODIStable to get a table.

In order to be callable from the FORTRAN in different platforms using function name cgmtbl, this function is named ncgmtbl in the actual C code. ncgmtbl is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that ncgmtbl compiled by C compilers will always be the same as the object name of a FORTRAN function named cgmtbl.

C:
```
void ncgmtbl(intf *modfil, _fcd tblnm, intf *ltb, _fcd group,   intf
*lgr, _fcd fldnm, intf *lfl, intf *start, intf *recno,   intf *bsize,
VOIDP data, intf *ret)
```

Input Parameters:

modfil	IN:	FORTRAN integer array that is used to reference the MODIS HDF file.
tblnm	IN:	FORTRAN character string of the name of the table.
ltb	IN:	FORTRAN integer address of the memory size of *tblnm.*
group	IN:	ASCII string name of the data group from which the table will be retrieved.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
fldnm	IN	FORTRAN character string of comma delimited name of fields to be retrieved.
lfl	IN:	FORTRAN integer address of the length of *fldnm.*
start	IN:	FORTRAN integer address of the start record, 0-based.
recno	IN:	FORTRAN integer address of the number of records to be retieved.
bsize	IN/OUT:	FORTRAN integer address containing the memory size of the *data* buffer. Output will be replaced with the actual bytes required to hold the data.

Output Parameters:

data	OUT:	The data buffer to store the retrieved table data.
ret	OUT:	FORTRAN integer address of the return status (MFAIL, MAPIOK)

Returns: N/A

copmfil

Purpose: Provides a wrapping function interfacing between C and FORTRAN for openMODISfile. This C function is only called by FORTRAN function OPMFIL. This function is a M-API internal routine.

Description: copmfil is a C function which is callable from FORTRAN. This function will call openMODISfile to open/create a MODIS HDF file. In M-API, copmfil is a low-level routine which is called only by CRMTBL.

In order to be callable from the FORTRAN in different platforms using function name copmfil, this function is called ncopmfil in the actual C code. ncopmfil is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncopmfil will always be the same as the object name of a FORTRAN function named copmfil.

C:
```
void copmfil (_fcd fname, intf *lfn, _fcd access, intf *lac,
intf modfil[MODFILLEN] , intf *ret)
```

Input Parameters:

fname IN: FORTRAN complete path and filename for the file to be opened.

lfn IN: Number of bytes in fname.

access IN: FORTRAN character string of access mode.

Permitted access mode:

"r" Open for read only.

"w" Create for read/write.

"a" Open for read/write.

lac IN: Number of bytes in access.

Output Parameters:

modfil OUT: Array that is used to reference the file in all other M-API routines in FORTRAN. The array will return all zeroes if an error occurs.

ret OUT: If function is successful, *ret* is set to 0, otherwise -1

Returns: N/A

cpmar

Purpose: Provides a wrapping function interfacing between C and FORTRAN for putMODISarray. This C function is only called by FORTRAN function PMAR. This function is a M-API internal routine.

Description: cpmar is a wrapper which is callable from FORTRAN. This function will call putMODISarray to write array data. In M-API, cpmar is low-level routine which is called only by PMAR.

In order to be callable from the FORTRAN in different platforms using function name cpmar, this function is called ncpmar in the actual C code. ncpmar is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncpmar will always be the object name of a FORTRAN function named cpmar.

C:
```
void cpmar (intf modfil[], _fcd arrnm, intf *lar, _fcd group, intf *lgr,
intf start[], intf dims[], void *data, intf *ret)
```

Input parameters:

modfil	IN:	FORTRAN integer array that is used to reference the MODIS HDF file.
arrnm	IN:	FORTRAN character string that is the name of the array.
lar	IN:	FORTRAN integer address of the memory size of *arrnm.*
group	IN:	FORTRAN character string which is the name of the data group to which the array (SDS) belongs.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
start	IN:	FORTRAN integer array containing the array structure location to begin writing the data to the array structure. *start* must have the same number of elements as the target array has dimensions.
dims	IN:	FORTRAN integer array describing the size of the array being written to the array structure. *dims* must have the same number of elements as the target array structure has dimensions and the product of the array dimensions must equal the number of elements in *data.*
data	IN:	The data array.

Output Parameters:

ret	OUT:	FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

cpmarin

Purpose: Provides a wrapping function interfacing between C and FORTRAN for putMODISarinfo. This C function is called only by FORTRAN function PMARIN. This function is a M-API low-level internal routine.

Description: cpmarin is a C function callable from FORTRAN. This function will call putMODISarinfo to attach an attribute to the array.

In order to be callable from the FORTRAN in different platforms using function name cpmarin, this function is named ncpmarin in the actual C code. ncpmarin is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of C ncpmarin will always be the same as the object name of a FORTRAN function named cpmarin.

C:

```
void cpmarin(intf modfil[MODFILLEN], _fcd arrnm, intf *lar, _fcd group,
intf *lgr, _fcd attr, intf *lat, _fcd dtype, intf *ldt, intf *nms,
VOIDP value, intf *ret)
```

Input Parameters:

modfil	IN:	FORTRAN integer array used to reference the MODIS HDF file.
arrnm	IN:	FORTRAN character string of the name of the array.
lar	IN:	FORTRAN integer address of the memory size of *arrnm.*
group	IN:	FORTRAN string name of the data group to which the array belongs.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
attr	IN:	FORTRAN character string of the attribute name.
lat	IN:	FORTRAN integer address of the memory size of *attr*.
dtype	IN:	FORTRAN character string of the memory size of the attribute data type.
ldt	IN:	FORTRAN integer address of the memory size of *dtype*.
nms	IN:	FORTRAN integer address of the number of elements in *value*.
value	IN:	*value* buffer.

Output Parameters:

ret	OUT:	FORTRAN integer address of the return status (MFAIL, MAPIOK)

Returns: N/A

cpmdmin

Purpose: Provides a wrapping function interfacing between C and FORTRAN for putMODISdiminfo. This C function is only called by FORTRAN function PMDMIN. This function is a M-API internal routine.

Description: cpmdmin is a wrapper which is callable from FORTRAN. This function will call putMODISdiminfo to write a dimensional attribute. In M-API, cpmdmin is low-level routine which is called only by PMAR.

In order to be callable from the FORTRAN in different platforms using function name cpmdmin, this function is called ncpmdmin in the actual C code. ncpmdmin is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncpmdmin will always be the object name of a FORTRAN function named cpmdmin.

C:
```
void cpmdmin(intf modfil[], _fcd arrnm, intf *lar, _fcd group, intf *lgr,
intf *dim, _fcd attrnm, intf *lat, _fcd dtype, intf *ldt, intf *nms, void
*value, intf *ret)
```

Input parameters:

modfil	IN:	FORTRAN integer array that is used to reference the MODIS HDF file.
arrnm	IN:	FORTRAN character string that is the name of the array.
lar	IN:	FORTRAN integer address of the memory size of *arrnm.*
group	IN:	FORTRAN character string which is the name of the data group to which the array (SDS) belongs.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
dim	IN:	The dimension to which the attribute will be attached (0-based).
attrnm	IN:	Name to assign the attribute. Provided macros for accepted MODIS file attribute names are listed in Appendix A, M-API-Supplied Constants and Macros.
lat	IN:	FORTRAN address of the string length of *attrnm*
dtype	IN:	Data type of the *value*.
ldt	IN:	FORTRAN address of the string length of *dtype*
nms	IN:	Number of attribute values in *value*.
value	IN:	The data to store in the attribute. If the attribute already exists, the value will be updated.

Output parameters:

ret	OUT:	FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

cpmdnam

Purpose: Provides a wrapping function interfacing between C and FORTRAN for putMODISdimname. This C function is only called by FORTRAN function PMDNAM. This function is a M-API internal routine.

Description: cpmdnam is a C function which is callable from FORTRAN. This function will call putMODISdimname to read a dimension name. In M-API, cpmdnam is low-level routine which is called only by GMDNAM.

In order to be callable from the FORTRAN in different platforms using function name cpmdnam, this function is called ncpmdnam in the actual C code. ncpmdnam is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncpmdnam will always be the object name of a FORTRAN function named cpmdnam.

C:
```
void cpmdnam(intf *modfil, _fcd arrnm, intf *lar, _fcd group, intf *lgr,
intf *dim, _fcd dname, intf *ldn, intf *ret)
```

Input parameters:

modfil	IN:	FORTRAN integer array that is used to reference the MODIS HDF file.
arrnm	IN:	FORTRAN character string that is the name of the array.
lar	IN:	FORTRAN integer address of the memory size of *arrnm.*
group	IN:	FORTRAN character string which is the name of the data group to which the array (SDS) belongs.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
ldn	IN:	FORTRAN integer array to indicate the start location to retrieve.
dim	IN:	FORTRAN integer address specifying the dimension.
dname	IN:	FORTRAN string containing the dimension name.

Output parameters:

ret	OUT:	FORTRAN integer address of the status (MFAIL, MAPIOK)

Returns: N/A

cpmfin

Purpose: Provides wrapping function interfacing between C and FORTRAN for putMODISfileinfo. This C function is called only by FORTRAN function PMFIN. This function is a M-API low-level internal routine.

Description: cpmfin is a C function callable from FORTRAN. This function will call putMODISfileinfo to put a global (file) attribute.

In order to be callable from the FORTRAN in different platforms using function name cpmfin, this function is named ncpmfin in the actual C code. ncpmfin is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of C ncpmfin will always be the same as the object name of a FORTRAN function named cpmfin.

C:
```
void cpmfin(intf modfil[MODFILLEN], _fcd attr, intf *lat, _fcd dtype,
intf *ldt, intf *nms, VOIDP value, intf *ret)
```

Input Parameters:

modfil	IN:	FORTRAN integer array used to reference the MODIS HDF file.
attr	IN:	FORTRAN character string of the attribute name.
lat	IN:	FORTRAN integer address of the memory size of *attr*.
dtype	IN:	FORTRAN character string of the memory size of the attribute data type.
ldt	IN:	FORTRAN integer address of the memory size of *dtype*.
nms	IN:	FORTRAN integer address of the number of elements in *value*.
value	IN:	*value* buffer.

Output Parameters:

ret	OUT:	FORTRAN integer address of the return status (MFAIL, MAPIOK)

Returns: N/A

cpmtbl

Purpose: Provides a wrapping function interfacing between C and FORTRAN for putMODIStable. This C function is called only by FORTRAN function PMTBL. This function is a M-API low-level internal routine.

Description: cpmtbl is a C function callable from FORTRAN. This function will call putMODIStable towrite a table to the HDF file.

In order to be callable from the FORTRAN in different platforms using function name cpmtbl, this function is named ncpmtbl in the actual C code. ncpmtbl is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of C ncpmtbl will always be the same as the object name of a FORTRAN function named cpmtbl.

C:
```
void cpmtbl(intf *modfil, _fcd tblnm, intf *ltb, _fcd group, intf *lgr,
intf *start, intf *recno, VOIDP data, intf *ret)
```

Input Parameters:

modfil	IN:	FORTRAN integer array used to reference the MODIS HDF file.
tblnm	IN:	FORTRAN character string of the name of the table.
ltb	IN:	FORTRAN integer address of the memory size of *tblnm.*
group	IN:	ASCII string name of the data group where the table will be put.
lgr	IN:	FORTRAN integer address of the memory size of *group.*
start	IN:	FORTRAN integer address of the start record, 0-based.
recno	IN:	FORTRAN integer address of the number of records to be put into the table.
data	IN:	Data buffer of the table.

Output Parameters:

ret	OUT:	FORTRAN integer address of the return status (MFAIL, MAPIOK)

Returns: N/A

crmfh

Purpose: Provides a wrapping function interfacing between C and FORTRAN. This C function is only called by the FORTRAN function RMFH. This function is a M-API internal routine.

Description: Once an HDF-EOS file has been opened using an HDF-EOS open file routine. The user must call CMFH to create the M-API file ID handle. The M-API routines can then be used to access the data object(s) in the opened file. Once the user is finished accessing the objects they must call RMFH to release this file ID handle created by calling CMFH. RMFH writes the objects accessed using M-API routines to a file, ends access to them, and frees the allocated memory. Finally, before exiting the user must call an HDF-EOS close file routine to close the HDF-EOS file.

crmfh is a C function which is callable from FORTRAN. This function will call releaseMAPIfilehandle to release the modfil array created using createMAPIfilehandle. In M-API, crmfh is a low-level routine called only by RMFH.

In order to be callable from the FORTRAN on different platforms using function name crmfh, this function is called ncrmfh in the actual C code. ncrmfh is redefined in mapic.h according to the compiler's FORTRAN naming conventions/conversion for each platform, so that the object name of ncrmfh will always be the object name of a FORTRAN function named crmfh.

c: void crmfh (*intf modfil[MODFILLEN], intf *ret*)

Input Parameters:

modfil IN/OUT: FORTRAN MODIS file array. If successful, the elements of the array will be filled with zero after closing the MODIS file.

Output Parameters:

ret OUT: If function is successful, ret is set to MAPIOK, otherwise MFAIL.

Returns: N/A

4.3 Prototype Listing for Low-Level Routines

Table 4-2 is a prototype listing of the low-level C and FORTRAN Routines.

Table 4-2. Low-Level C and FORTRAN Routine Prototype Listing

Return Value	Routine Name [Parameter(s)]
int	`addid (MODFIL *file, char *name, char *groupname, int32 id, int32 type, char *access)`
int	`addMODISgroup (MODFIL *file, char *groupname, char *classname, int32 tag, int32 ref)`
INTEGER FUNCTION	`ADMGRP (modfil, group, clsnm, tag, ref)`
int32	`datatype_to_DFN (char *datatype)`
int	`DFNT_to_datatype (int32 dfnt, char *datatype)`
INTEGER FUNCTION	`DF2DT (dfnt, dtype)`
INTEGER FUNCTION	`DT2DF (dtype)`
short int	`emptyVdata (MODFIL *file, int32 vdata_ref)`
DATAID *	`getMODISarrayid(MODFIL *file, char *arrayname, char *groupname)`
DATAID	`getMODIStableid(MODFIL *file, char *tablename, char *groupname, char *access)`
AGGREGATE	`MPVL2ODL (MODFIL *file, char*PVLAttrName, char *aggName)`
int	`MTYPEc2f (char *c_data_type, char *f_data_type, long int *f_length)`
int	`MTYPEf2c (char *f_data_type, char *c_data_type, long int *c_length)`
int32	`MVSfind (int32 fid, char *vsname)`
int	`NULLstr (char *string)`
int	`parse_string (char *target_string, char *delimiter, int n_tokens, char *tokens[])`
short int	`SDS_footprintOK (int32 sds_id, long int start[], long int dimsizes[])`
INTEGER FUNCTION	`SDSOK (sdsid, start, dims)`
DATAID *	`searchid (MODFIL *file, char *name, char *group, int32 type, char *access)`
int32	`searchMODISgroup (MODFIL *file, char *groupname, char *classname, char *objectname, char *objectclass, int32 objecttype)`
short int	`set_Vhasdata (MODFIL *file, int32 vdata_ref)`
INTEGER FUNCTION	`SRMGRP(modfil, group, clsnm, objnm, objcls, objtyp`
short int	`Vdatattr_name (char *attribute, int32 vdata_ref, char *attibute_name)`
int	`Vfdatatypes (int32 vdata_id, long int *stringlen, char *data_type)`

4.4 Individual Low-Level Routines

The following are explanations for each individual low-level Routine.

addid

Purpose: Add newly opened data object (vdata or SDS) ID into the DATAID structure in the MODFIL structure.

Description: addid creates a DATAID structure which contains the ID of the opened object and inserts the structure to the ring super structure. In SDS, the DATAID structure also includes a void pointer which points to an SDSINFO structure created and filled by addid. The SDSINFO structure contains the rank, dimension size, and data type information for the opened SDS.

C: `int addid(MODFIL *file, char *name, char *groupname, int32 id, int32 type, char *access)`

Input parameters:

file	IN:	Address of MODFIL structure that is used to reference a MODIS HDF file containing the newly opened data object (Vdata or SDS).
name	IN:	The name of the object.
groupname	IN:	The name of the group to which the object belongs. If set to NULL, the object is a lone object.
id	IN:	The ID for the object returned from HDF object open routines such as VSattach or SDselect.
type	IN:	The object type: either DFTAG_NDG (for SDS) or DFTAG_VH (for Vdata).
access	IN:	The access type of the object: "r" or "w". This parameter is ignored when *type* is DFTAG_NDG.

Output parameters: N/A

Returns: MAPIOK if successful, MFAIL if an error occurs.

Error Messages: **Description:**

addid found the access type *access* is unrecognizable.

addid unable to allocate a DATAINFO structure while attempting to insert data ID of *name* into MODFIL structure.

addid unable to allocate a DATAID structure while attempting to insert data ID of *name* into MODFIL structure.

addid unable to allocate space for storing object's name while attempting to insert data ID of *name* into MODFIL structure.

addid unable to allocate space for storing object's group name while attempting to insert data ID of *name* into MODFIL structure.

addid unable to allocate an SDSINFO structure while attempting to insert data ID of *name* into MODFIL structure.

addid detected FAIL from HDF procedure SDgetinfo while attempting to insert data ID of *name* into MODFIL structure.

addid unable to allocate space for storing dimsizes while attempting to insert data ID of *name* into MODFIL structure.

addid unable to allocate VDINFO structure while attempting to insert data ID of *name* into MODFIL structure.

addMODISgroup (ADMGRP)

Purpose: Add an object to an HDF Vgroup.

Description: addMODISgroup (ADMGRP) is a M-API internal function which adds an HDF object into a Vgroup. The group is specified by its name and class name. However, the *classname* is an optional feature. If class name is set to NULL, the function will find the Vgroup with the same name as *groupname* without considering the *classname* of the group to insert the object into the group. The object is specified by its *tag* and reference number. The *tag* for SDS is DFTAG_NDG and for Vdata is DFTAG_VH. To obtain the reference, use SDidtoref for SDS and VSQueryref for Vdata. No check for the tag or reference is performed.

C:
```
int addMODISgroup(MODFIL *file, char *groupname, char *classname, int32 tag,
int32 ref)
```

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file.
groupname	IN:	ASCII string of the name of the data group.
classname	IN:	(Optional) ASCII string of the class name of the data group. Set to ' ' (blank) to not compare the group *clsnm*.
tag	IN:	The tag of the object. Valid values include: DFTAG_NDG, DFTAG_VH, and DFTAG_VG.
ref	IN:	The reference number of the object.

Output Parameters: N/A

FORTRAN:
```
INTEGER FUNCTION ADMGRP(modfil, group, clsnm, tag, ref)
      INTEGER           modfil(MODFILLEN), tag, ref
      CHARACTER*(*)     group, clsnm
```

Input Parameters:

modfil	IN:	Array that is used to reference the MODIS HDF file.
group	IN:	ASCII string of the name of the data group.
clsnm	IN:	(Optional) ASCII string of the class name of the data group. Set to NULL to not compare the group classname.
tag	IN:	The tag of the object. Valid values include: DFTAG_NDG for SDS, DFTAG_VH for Vdata, and DFTAG_VG for Vgroup.
ref	IN	The reference number of the object.

Output Parameters: N/A

Returns: MAPIOK if successful, otherwise, MFAIL.

datatype_to_DFNT (DT2DF)

Purpose: Returns the HDF number type associated a M-API data type string.

Description: datatype_to_DFNT (DT2DF) takes a M-API data type string and returns the HDF number type integer used by HDF routines to identify number types.

C: `int32 datatype_to_DFNT(char *datatype)`

Input Parameters:

datatype IN: Pointer to M-API data type string.

Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"float32"
"float64"
"char *"
```

Output Parameters: N/A

FORTRAN:

```
INTEGER FUNCTION DT2DF(dtype)
CHARACTER*(*)     dtype
```

Input Parameters:

dtype IN: M-API data type string.

Permitted FORTRAN data types:

```
'INTEGER*1'
'UINTEGER*1'
'INTEGER*2'
'UINTEGER*2'
'INTEGER*4'
'UINTEGER*4'
'REAL*4'
'REAL*8'
'CHARACTER*(*)'
```

Output Parameters: N/A

Returns: HDF number type or MFAIL if an error occurs.

DFNT_to_datatype (DF2DT)

Purpose: Returns the M-API data type string associated with an HDF number type.

Description: DFNT_to_datatype (DF2DT) takes an HDF number type constant (DFNT_*) used by HDF routines to identify number types and returns a M-API data type strings, if possible. If no match can be made, a null string is returned. An eight or more element character string must be provided for *datatype*.

C: `int DFNT_to_datatype (int32 dfnt, char *datatype)`

Input Parameters:

dfnt IN: HDF number type constant.

Output Parameters:

datatype OUT: Pointer to M-API data type string. This string must be at least 8 bytes long. Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"float32"
"float64"
"char *"
```

FORTRAN:

```
INTEGER FUNCTION DF2DT (dfnt, dtype)
      INTEGER          dfnt
      CHARACTER*(*)    dtype
```

Input Parameters:

dfnt IN: HDF number type constant.

Output Parameters:

dtype OUT: M-API data type string.

Permitted FORTRAN data types:

```
'INTEGER*1'
'UINTEGER*1'
'INTEGER*2'
'UINTEGER*2'
'INTEGER*4'
'UINTEGER*4'
'REAL*4'
'REAL*8'
'CHARACTER*(*)'
```

Returns: MAPIOK if *dfnt* matched to a data type string or MFAIL if an error occurs.

emptyVdata

Purpose: Determines whether the target Vdata is empty or contains data.

Description: emptyVdata (EMPTV) indicates whether a Vdata is 'empty' by searching for an HDF global metadata associated with the Vdata. An empty Vdata may not exist, so a dummy record is inserted when it is first created. This dummy record should be overwritten with the first call from putMODIStable. When the dummy record is overwritten by a single record the content of the Vdata is ambiguous, so a metadata ('NOT EMPTY') is associated with the Vdata to indicate that the Vdata is no longer empty. emptyVdata searches for the existence of this label and reads it as evidence that the dummy record has been overwritten. This check is performed ONLY if the Vdata contains one record; multiple records in a Vdata implies that the dummy record has been overwritten and no semaphore metadata needs to be checked (or written).

C: `short int emptyVdata(MODFIL *file, int32 vdata_ref)`

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file receiving the new table.
vdata_ref	IN:	Reference number of the Vdata to interrogate.

Output Parameters: N/A

Returns: 1 if the Vdata is empty, 0 if it contains data, MFAIL if an error occurs.

getMODISarrayid

Purpose: get the address of the DATAID structure for a MODIS array (SDS)

Description: getMODISarrayid returns the address of DATAID structure for the array. The DATAID structure contains the HDF access ID as well as rank and dimension information. This is an internal M-API routine used by M-API array access routines. It is suggested that getMODISarray and putMODISarray should not call but in-line the code of this routine since the performance of getMODISarray and putMODISarray is critical for the overall performace of M-API.

Because this routine is only used by M-API internally, and all inputs of this routine have been checked in the calling routines, this routine does not performace the input parameter checking.

```
C: DATAID *getMODISarrayid (MODFILE *file, char *arrayname, char *groupname)
```

input Parameters:

file	IN:	Address of MODFILE structure that is used to reference a MODIS HDF file receiving the dimension name.
arrayname	IN:	ASCII string name of the target array structure.
groupname	IN:	ASCII string name of the data group containing the target

Output Parameters: N/A

Returns: (DATAID *) if successful, NULL if an error occurs.

getMODIStableid

Purpose: Get the address of the DATAID structure for a MODIS table (Vdata).

Description: getMODIStableid returns the address of DATAID structure for a MODIS table. The DATAID structure contains the HDF access ID as well as vdata specfic information. This is an internal M-API routine used by M-API table access routines.

Because this routine is only used by M-API internally and all inputs of this routine have been checked in the calling routines, this routine does not perform input parameter checking.

C: `DATAID *getMODIStableid(MODFIL *file, char *tablename, char *groupname, char *access)`

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference a MODIS HDF file.
tablename	IN:	ASCII string name of the table structure to be searched.
groupname	IN:	ASCII string name of the data group containing the table
access	IN:	ASCII string of the table access type: "r" for read only, "w" for write.

Output Parameters: N/A

Returns: (DATAID *) if successful, NULL if an error occurs.

MPVL2ODL

Purpose: Read in a MODIS PVL (Parameter = Value) attribute in a HDF file and converts it to ODL stored in a tree structure.

Description: In HDF-EOS, attributes are collected together to form a text block using PVL. Then the text block is stored in HDF as a single attribute. MPVL2ODL reads in a MODIS PVL (Parameter = Value) attribute in a HDF file and converts it to ODL stored in a tree structure for searching individual attribute in PVL. This is a M-API internal routine called by getMODISPVLattr.

The function seaches the MODIS HDF global attributes first. If the *PVLAttrname* attributes can not be found in the global attributes, the function will then search the attributes attached to individual SDSs. Once the PVL attribute is found, the function will retrieve the attribute and convert it to ODL.

C: `AGGREGATE MPVL2ODL(MODFIL *file, char *PVLAttrName, char *aggName)`

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file containing the target PVL attribute.
PVLAttrName	IN:	ASCII string name of the HDF attribute which contains the PVL text block.
aggName	IN:	ASCII string name used to name the AGGREGATE structure.

Output Parameters: N/A

Returns: AGGREGATE if successful, NULL if an error occurs.
(Note AGGREGATE is a pointer to a tree structure defined in odldef.h)

MTYPEc2f

Purpose: Convert comma delimited M-API C data type string to M-API FORTRAN data type string .

Description: M-API uses a set of standard strings to describe the data types stored in array and table structures. For the same data type, the name is different between M-API C and FORTRAN interface. Because part of the FORTRAN interface calls the C interface by using the wrapping method, the data type string returned from the C interface is in C notation. This routine is a low level C function which will convert the C data type notation to FORTRAN data type notation. The function will return MFAIL if c_data_type contains an unrecognizable data type or the memory of f_data_type is not large enough to hold the FORTRAN data string.

C: `int MTYPEc2f(char *c_data_type, char *f_data_type, long int *f_length)`

Input Parameters:

c_data_type IN: String of comma-delimited data types.

Permitted C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
"float64"
"char *"
```

f_length IN/OUT: Address of the memory size (bytes) of the *f_data_type* string. Output will replaced with the actual length of *f_data_type* + 1. If a function error occurs, *f_length* will not be changed. If the f_data_type is not large enough, f_length will return with the actual size needed to hold the FORTRAN data type string. If the memory size of *f_data_type* is three times the string length of *f_data_type*, the memory will always be large enough. To get the actual size need to hold the FORTRAN data type string, the user may set the *f_data_type* to NULL and *f_length* to 0.

Output Parameters:

f_data_type OUT: Memory to hold comma-delimited M-API FORTRAN data types. If f_length is set to zero, this paramter can be set to NULL.

Returns: MFAIL if the *f_data_string* is not large enough or an unrecognizable data type in *c_data_type* string. MAPIOK if successful.

MTYPEf2c

Purpose: Convert comma delimited M-API C data type string to M-API FORTRAN data type string.

Description: M-API uses a set of standard strings to describe the data types stored in array and table structures. For the same data type, the name is different between M-API C and FORTRAN interface. Because part of the FORTRAN interface calls the C interface by using the wrapping method, the data type string returned from the C interface is in C notation. This routine is a low level C function which will convert the FORTRAN data type notation to C data type notation. The function will return MFAIL if *f_data_type* contains unrecognizable data type or the memory of *c_data_type* is not large enough to hold the FORTRAN data string.

C: `int MTYPEf2c(char *f_data_type, char *c_data_type, long int *c_length)`

Input Parameters:

f_data_type IN: String of comma-delimited data types.

Permitted FORTRAN data types:

```
'INTEGER*1'
'UINTEGER*1'
'INTEGER*2'
'UINTEGER*2'
'INTEGER*4'
'UINTEGER*4'
'REAL*4'
'REAL*8'
'CHARACTER*(*)'
```

c_length IN/OUT : Address of the memory size of the *c_data_type* string. Output will return with the actual length of *c_data_type* + 1. If a function error occurs, *c_length* will not be changed. If the *c_data_type* is not large enough, *c_length* will return with the actual size needed to hold the C data type string. If the memory size of *c_data_type* is twice the string length of *f_data_type*, the memory will always be large enough. To get the actual size need to hold the C data type string, user may set the *c_data_type* to NULL and *c_length* to 0.

Output Parameters:

c_data_type OUT: Memory to hold comma-delimited M-API C data types converted from *f_data_type*. If *f_length* is set to zero, this paramter can be set to NULL.

Returns: MFAIL if the *c_data_string* is not large enough or an unrecognizable data type in *f_data_type* string. MAPIOK if successful.

MVSfind

Purpose: To replace the HDF VSfind for obtaining the reference ID of a Vdata.

Description: MVSfind is a M-API internal function for obtaining the reference ID of a Vdata set. MVSfind is used to replace HDF VSfind which can not distinguish the user-defined Vdata set from attributes.

C: `int32 MVSfind(int32 fid, char *vsname)`

Input Parameters:

fid	IN:	HDF file ID returned from Hopen.
vsname	IN:	ASCII string of the name of the Vdata set.

Output Parameters: N/A

Returns: Vdata reference ID if successful, otherwise, HDF FAIL.

NULLstr

Purpose: Determines if a character pointer points to NULL or an empty string.

Description: NULLstr checks if the character pointer points to NULL or an empty string. It then returns either 1 if *string* points to NULL or a null string, otherwise 0 if *string* points to a character string.

C: `NULLstr(char *string)`

Input Parameters:

string IN: Pointer to determine if NULL or pointing to an empty string.

Output Parameters: N/A

Returns: 1 if *string* points to NULL or a null string.
0 if *string* points to a character string.

parse_string

Purpose: Breaks up character delimited string into constituent tokens.

Description: parse_string sets the *tokens* pointers to the first *n_tokens delimiter* delimited string token in *target_string.* Each terminating *delimiter* character in *target_string* is replaced by a null character so that each token pointed to in *tokens* may be handled as an independent string. The *tokens* array should have at least *n_tokens* elements. If there are fewer tokens in *target_string* than *n_tokens*, the unused pointers are set to NULL.

C:
```
int parse_string(char *target_string, char *delimiter, int n_tokens,
char *tokens[])
```

Input Parameters:

target_string	IN/OUT:	String to be atomized. Note that it is parsed *in situ.*
delimiter	IN:	String of acceptable delimiter characters.
n_tokens	IN:	Maximum number of pointers to string tokens found in *target_string.* Minimum number of available pointers in *tokens* array.

Output Parameters:

tokens	OUT:	Array of pointers to the parsed string tokens in *target_string.*

Returns: Number of string tokens found in *target_string.* This may be more than *n_tokens,* the number of string tokens made accessible through the *tokens* array. MFAIL is returned on an input error.

SDS_footprintOK (SDSOK)

Purpose: Determines whether an array access will fall within the boundaries of the target SDS.

Description: SDS_footprintOK (SDSOK) checks the start position and dimensions of a hyperslab to be accessed from an SDS against the dimensions of that SDS.

C:
```
short int SDS_footprintOK(int32 sds_id, long int start[],
long int dimsizes[])
```

Input Parameters:

sds_id	IN:	The HDF SDS data set ID returned from SDselect.
start	IN:	Array containing the array structure location to begin placing the data into the array structure. *start* must have the same number of elements as the target array has dimensions.
dimsizes	IN:	Array describing the size of the array being inserted into the array structure. *dimsizes* must have the same number of elements as the target array structure has dimensions and the product of the array dimensions must equal the number of elements in *data.*

Output Parameters: N/A

FORTRAN:
```
INTEGER FUNCTION SDSOK(sdsid, start,dims)
      INTEGER     sdsid, start(*), dims (*)
```

Input Parameters:

sdsid	IN:	The HDF SDS data set ID returned from SDselect.
start	IN:	Array containing the location where the first element of data is to be accessed from an SDS. *start* must have the same number of elements as the target array has dimensions.
dims	IN:	Array describing the size of the array being inserted into the array structure. *dims* must have the same number of elements as the rank of the target SDS.

Returns: 1 if proposed hyperslab consistent with SDS dimensions, 0 if not or an error occurs.

searchid

Purpose: Searches the ring super structure to find if an object is opened. If it is opened, returns the address of the DATAID structure.

Description: searchid searches the ring super structure to find if an object is opened. If it is opened, searchid returns the address of the DATAID structure. The function also maintains the ring so that the first object in the ring is always the current accessed object. Please note any routines using contents in the DATAID structure should neither change the contents nor release the DATAID structure.

searchid is an internal M-API routine which is used only by M-API routines. Application programs should not call this routine directly. Since this function will be only used by M-API functions and the speed of this function is important for overall performance of M-API, minimum error checking is performed in this function because all possible errors are checked in the calling routines.

C: `DATAID *searchid (MODFIL *file, char *name, char *groupname, int32 type, char *access)`

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference a MODIS HDF file containing opened data objects (Vdata or SDS).
name	IN:	The name of the object.
groupname	IN:	The name of the group to which the object belongs. If set to NULL, the object is a lone object.
type	IN:	The object type: either DFTAG_NDG (for SDS) or DFTAG_VH (for Vdata).
access	IN:	The access type, either "w" or "r". This argument is ignored when *type* is DFTAG_NDG.

Output Parameter: N/A

Returns: NULL if the object is not found in the ring super structure.
The address of the DATAID structure if the object is found.

searchMODISgroup (SRMGRP)

Purpose: To search a Vgroup to find if an HDF object is in the specified Vgroup.

Description: searchMODIS (SRMGRP) group searches an HDF Vgroup structure in a MODIS HDF file to find if an HDF object is in the Vgroup. Both the group and the object are specified by their name and class name. However, the *classname* is an optional feature. If class names are set to NULL, only name comparison is performed. Because SDS (array) has no class name, the *objectclass* for an SDS is always ignored. If the specified object exists, the function will return the reference ID for Vdata and Vgroup and index for SDS. If the object does not exist, the function will return NO_OBJECT, which is defined in mapic.h as -2.

C:
```
int32 searchMODISgroup(MODFIL *file, char *groupname, char *classname, char *objectname, char *objectclass, int32 objecttype)
```

Input Parameters:

file	IN:	Address of MODFIL structure that is used to reference the MODIS HDF file.
groupname	IN:	ASCII string of the name of the data group.
classname	IN:	(Optional) ASCII string of the class name of the data group. Set to NULL to not compare against the group's *classname*.
objectname	IN:	ASCII string of the object name to be searched for.
objectclass	IN	(Optional)ASCII string of the class name of the data object. Set to NULL to not compare against the object's class.
objecttype	IN	Type of the object; The valid objects are: DFTAG_NDG (for SDS) DFTAG_VH (for Vdata, or attribute if the object class is set to Attr0.0) DFTAG_VG (for Vgroup).

Output Parameters: N/A

FORTRAN:
```
INTEGER FUNCTION SRMGRP(modfil, group, clsnm, objnm, objcls, objtyp)
      INTEGER           modfil(MODFILLEN), objtyp
      CHARACTER*(*)     group, clsnm, objnm, objcls
```

Input Parameters:

modfil	IN:	Array that is used to reference the MODIS HDF file.
group	IN:	ASCII string of the name of the data group.

clsnm	IN:	(Optional) ASCII string of the class name of the data group. Set to NULL to not compare the group *clsnm*.
objnm	IN:	ASCII string of the object name to be searched.
objcls	IN	(Optional)ASCII string of the class name of the data object. Set to NULL blank (' ') to not compare the object class.
objtyp	IN	Type of the object; The valid objects are: DFTAG_NDG, DFTAG_VH, DFTAG_VG.

Output Parameters: N/A

Returns: NO_OBJECT if the object is not in the specified Vgroup; MFAIL if the function is failed; Reference ID if the function successfully finds the object the Vdata or Vgroup object; and SDS index if the function successfully finds the SDS object.

set_Vhasdata

Purpose: Creates a semaphore attribute to indicate that the first record of a particular Vdata is not a dummy record.

Description: set_Vhasdata (HASDAT) sets a semaphore HDF global attribute associated with a Vdata to indicate whether the Vdata is not 'empty'. An empty Vdata may not exist, so a dummy record is inserted when it is first created. This dummy record should be overwritten with the first call from putMODIStable. When the dummy record is overwritten by a single record the content of the Vdata is ambiguous, so this routine creates a metadata ('NOT EMPTY') associated with the Vdata to indicate that the Vdata is no longer empty. This routine is called ONLY if the Vdata contains one non-dummy record; multiple records in a Vdata implies that the dummy record has been overwritten and no semaphore metadata needs to be checked (or written).

C:

```
short int set_Vhasdata(MODFIL *file, int32 vdata_ref)
```

Input Parameters:

file IN: Address of MODFIL structure that is used to reference the MODIS HDF file receiving the new table.

vdata_ref IN: Reference number of the Vdata to interrogate.

Output Parameters: N/A

Returns: MAPIOK if the semaphor attribute is written successfully, MFAIL if an error occurs.

Vdatattr_name

Purpose: Generates a unique global attribute name to associate with Vdata metadata.

Description: Vdatattr_name (VDATRN) creates an attribute name that is associated with a particular Vdata. This allows the global metadata facility to be used to store Vdata metadata. The Vdata tag (DFTAG_VH) and the Vdata's ref-id are appended to the attribute name to make a Vdata-specific metadata name. If the attribute name is very long, these two integers will overwrite the last few characters of the *attribute* so that the length of *attribute_name* remains within the VSNAMELENMAX -1 (63) character limit for HDF global attribute names.

C:
```
short int Vdatattr_name(char *attribute, int32 vdata_ref,
char *attribute_name)
```

Input Parameters:

attribute IN: Standard attribute name of a Vdata metadata.

vdata_ref IN: Reference number of the Vdata with which to associate the metadata.

Output Parameters:

attribute_name OUT: Name to assign the attribute in the HDF file so that it may be associated with a particular Vdata. This array must be at least VSNAMELENMAX elements long.

Returns: MAPIOK if the attribute name is created successfully, MFAIL if an error occurs.

VFdatatypes

Purpose: Retrieves a comma delimited string of the MODIS data types of a Vdata's fields.

Description: VFdatatypes retrieves the data type for each field of a Vdata. The HDF number type of each field is converted into its corresponding M-API data type and this is appended to the output string. An error (MFAIL) will be returned if 1) The output string is not long enough to contain the data type strings for all the Vdata's fields, 2) an unknown (e.g. not supported by the M-API) number type is encountered or 3) an HDF routine FAILs. The data type string will be returned truncated to the point where the fault occurred. *stringlen,* the address of the length of the *data_type* output string, is normally revised to contain the actual array length required to hold the output string. If the latter two errors occur, however, it is set to 0.

C: `int VFdatatypes(int32 vdata_id, long int *stringlen, char *data_type)`

Input Parameters:

vdata_id IN: Vdata's access identifier returned from Vsattach.

stringlen IN/OUT: Address of the length of the *data_type* output string. This number is returned as the minimum length of an array capable of holding the output string. It is set to 0 if a functional error occurs.

Output Parameters:

data_type OUT: Array of comma-delimited data types for each table field.

Possible C data types:

```
"int8"
"uint8"
"int16"
"uint16"
"int32"
"uint32"
"int64"
"float32"
"float64"
```

Returns: MAPIOK if successful, MFAIL if an error occurs.

APPENDIX A: M-API-SUPPLIED CONSTANTS AND MACROS

The following tables show the constants that are found in the mapi.h (C) and mapi.inc (FORTRAN):

Table A-1. M-API Data Type Constants

M-API Constant	C	FORTRAN
I8	"int8"	'INTEGER*1'
I16	"int16"	'INTEGER*2'
I32	"int32"	'INTEGER*4'
I64	"int64"	'INTEGER*8'
R32	"float32"	'REAL*4'
R64	"float64"	'REAL*8'
TXT	"char *"	'CHARACTER*(*)'
UI8	"unint8"	'INTEGER*1'
UI16	"unint16"	'INTEGER*2'
UI32	"unint32"	'INTEGER*4'
UI64	"unint64"	

Table A-2. SDS Metadata Constants

Metadata Description	Metadata Name	M-API Constant
array structure and dimension label string	"long_name"	MLONG_NAME
array structure and dimension units string	"units"	MUNITS
array structure and dimension format string	"format"	MFORMAT
array structure coordinate system string	"cordsys"	MCOORD_SYS
array structure Calibration factor	"scale_factor"	MSLOPE
array structure Calibration factor error	"scale_factor_err"	MSLOPE_ERROR
array structure uncalibrated offset	"add_offset"	MOFFSET
array structure uncalibrated offset error	"add_offset_err"	MOFFSET_ERROR
array structure uncalibrated data HDF number type	"calibrated_nt"	MNUM_TYPE
standard data valid range (Sdgetrange)[minimum,]	"valid_range"	MDATA_RANGE
array structure Fill Value	"_FillValue"	MFILL_VALUE
ECS inventory metadata global attribute name	"CoreMetadata.0"	MECS_CORE
ECS archive metadata global attribute name	"ProductMetadata.0"	MECS_ARCHIVE
'Same as above' - returned for Backward compatibility	"ProductMetadata.0"	MECS_PRODUCT

Table A-3. ECS Global Inventory Metadata Names

Note: User should refer to a particular file specification for a more precise layout of the metadata for a product.

Metadata Description	Metadata Name	M-API Constant
HDFattrNames = **MECS_CORE**		
References to all ancillary input files, (i.e., all input files other than MODIS products).	"ANCILLARYINPUTPOINTER"	MCORE_ANCIL_POINTER
Indicates the results of QA performed during product generation.	"AUTOMATICQUALITYFLAG"	MCORE_AUTO_QUALITY
Easternmost longitude of the granule spatial coverage.	"EASTBOUNDINGCOORDINATE"	MCORE_EAST_BOUND
Flag indicating whether points are on an inner (exclusion) G-ring.	"EXCLUSIONGRINGFLAG"	MCORE_EXCLUS_GRING_FLG
Self-reference to granule. For V1, this field should be identical to MODISPRODUCTFILENAME.	"GRANULEPOINTER"	MCORE_GRAN_POINTER
Latitudes of a series of points representing the perimeter of the granule spatial coverage (i.e., corners).	"GRINGPOINTLATITUDE"	MCORE_GRING_POINT_LAT
Longitudes of a series of points representing the perimeter of the granule spatial coverage.	"GRINGPOINTLONGITUDE"	MCORE_GRING_POINT_LON
Sequence numbers corresponding to perimeter latitudes and longitudes.	"GRINGPOINTSEQUENCENO"	MCORE_GRING_POINT_NUM
References to other MODIS product granules used as input for this product.	"INPUTPOINTER"	MCORE_INPUT_POINTER
A descriptive name for the data collection.	"LONGNAME"	MCORE_LONG_NAME
Northernmost latitude of the granule spatial coverage.	"NORTHBOUNDINGCOORDINATE"	MCORE_NORTH_BOUND
The granule level flag applying both generally to the granule and specifically to the parameters at the granule level. When applied to a parameter, the flag refers to the quality of that parameter in the granule.	"OPERATIONALQUALITYFLAG"	MCORE_OPER_QUAL_FLAG
Number of satellite orbit during which the granule data were collected.	"ORBITNUMBER"	MCORE_ORBIT_NUM
Reference to processing history file.	"PROCESSINGHISTORYPOINTER"	MCORE_HISTORY_POINTER
Value indicating the percent of interpolated data in the granule	"QAPERCENTINTERPOLATEDDATA"	MCORE_PERCENT_INTERP
Value indicating the percent of missing data in the granule.	"QAPERCENTMISSINGDATA"	MCORE_PERCENT_MISSING
Value indicating the percent of data in the granule outside of acceptable limits.	"QAPERCENTOUTOFBOUNDSDATA"	MCORE_PERCENT_OUT
A text explanation of the criteria used to set each quality flag; including thresholds or other criteria.	"QUALITYFLAGEXPLANATION"	MCORE_QUAL_EXPL
The date and time when the temporal coverage period of this granule began.	"RANGEBEGINNINGDATETIME"	MCORE_RANGE_START
The date and time when the temporal coverage period of this granule ended.	"RANGEENDINGDATETIME"	MCORE_RANGE_END

Metadata Description	Metadata Name	M-API Constant
Indicator of what reprocessing is planned for the granule.	"REPROCESSINGPLANNED"	MCORE_TO_BE_REDONE
Indicator of the reprocessing status of the granule.	"REPROCESSINGACTUAL"	MCORE_ACTUALLY_REDONE
The granule level flag applying to the granule and to the parameters at the granule level. When applied to a parameter, the flag refers to the quality of that parameter in the granule.	"SCIENCEQUALITYFLAG"	MCORE_SCIENCE_QUAL_FLG
The identifier for the data collection.	"SHORTNAME"	MCORE_SHORT_NAME
The size of the data granule in megabytes.	"SIZEMBECSDATAGRANULE"	MCORE_SIZE_OF_GRANULE
Southernmost latitude of the granule spatial coverage.	"SOUTHBOUNDINGCOORDINATE"	MCORE_SOUTH_BOUND
Westernmost longitude of the granule spatial coverage.	"WESTBOUNDINGCOORDINATE"	MCORE_WEST_BOUND
The MODIS filename for this granule.	"MODISPRODUCTFILENAME"	MPROD_FILENAME
MODIS mode of operation.	"OPERATIONMODE"	MPROD_OPERATIONMODE
This field contains the date and time the process that created this file was started.	"PROCESSINGDATETIME"	MPROD_PROC_DATE_TIME
The SPSO parameters for all data contained in this file, as listed in the SPSO database.	"SPSOPARAMETERS"	MPROD_SPSO_PARAM
The number of this MODIS granule.	"GRANULENUMBER"	MPROD_GRANULE_NUM
HDFattrNames = **MECS_PRODUCT**		
The date this algorithm package version successfully passed AI&T procedures and was accepted as an ECS standard algorithm.	"ALGORITHMPACKAGEACCEPTANCEDATE"	MPROD_ALGO_PCK_ACPT_DATE
This specifies the maturity of the algorithm package	"ALGORITHMPACKAGEMATURITYCODE"	MPROD_ALGO_PACK_MAT_CODE
Algorithm package name	"ALGORITHMPACKAGENAME"	MPROD_ALGO_PACK_NAME
The version of the algorithm package.	"ALGORITHMPACKAGEVERSION"	MPROD_ALGO_PACK_VER
The long name by which the instrument is known.	"INSTRUMENTNAME"	MPROD_INSTR_NAME
The short name assigned to the platform carrying the instrument.	"PLATFORMSHORTNAME"	MPROD_PLATFORM_SHORT_NAM
DAAC where product is processed.	"PROCESSINGCENTER"	MPROD_PROC_CENTER

Table A-4. Level 1A Macros

Metadata Description	Metadata Name	M-API Constant
MOD01_L1A	"MOD01_L1A"	MOD01_L1A
Scan number	"Scan number"	M01SCAN_NUMBER
Frame count array	"Frame count array"	M01FRAME_COUNT_ARRAY
Scan Type	"Scan Type"	M01SCAN_TYPE
SD start time	"SD start time"	M01SD_START_TIME
SRCA start time	"SRCA start time"	M01SRCA_START_TIME
BB start time	"BB start time"	M01BB_START_TIME
SV start time	"SV start time"	M01SV_START_TIME
EV start time	"EV start time"	M01EV_START_TIME
SRCA calibration mode	"SRCA calibration mode"	M01SRCA_CALIBRATION_MODE
Packet scan count	"Packet scan count"	M01PACKET_SCAN_COUNT
CCSDS Application Identifier	"CCSDS Application Identifier"	M01CCSDS_APID
Packet Quick Look flag	"Packet expedited data flag"	M01PACKET_QL
Mirror side	"Mirror side"	M01MIRROR_SIDE
Scan quality array	"Scan quality array"	M01SCAN_QUALITY_ARRAY
Earth sector Pixel quality	"Earth sector Pixel quality"	M01EV_PIX_QUAL
SD sector Pixel quality	"SD sector Pixel quality"	M01SD_PIX_QUAL
SRCA sector Pixel quality	"SRCA sector Pixel quality"	M01SRCA_PIX_QUAL
BB sector Pixel quality	"BB sector Pixel quality"	M01BB_PIX_QUAL
SV sector Pixel quality	"SV sector Pixel quality"	M01SV_PIX_QUAL
Bands 1 and 2	"EV_250m"	M01EV_250M
Bands 3 through 7	"EV_500m"	M01EV_500M
Bands 8 through 19	"EV_1km_day"	M01EV_1KM_DAY
Bands 20 through 36	"EV_1km_night"	M01EV_1KM_NITE
Bands 1 and 2	"SD_250m"	M01SD_250M
Bands 3 through 7	"SD_500m"	M01SD_500M
Bands 8 through 19	"SD_1km_day"	M01SD_1KM_DAY
Bands 20 through 36	"SD_1km_night"	M01SD_1KM_NITE
Bands 1 and 2	"SRCA_250m"	M01SRCA_250M
Bands 3 through 7	"SRCA_500m"	M01SRCA_500M
Bands 8 through 19	"SRCA_1km_day"	M01SRCA_1KM_DAY
Bands 20 through 36	"SRCA_1km_night"	M01SRCA_1KM_NITE
Bands 1 and 2	"BB_250m"	M01BB_250M
Bands 3 through 7	"BB_500m"	M01BB_500M
Bands 8 through 19	"BB_1km_day"	M01BB_1KM_DAY
Bands 20 through 36	"BB_1km_night"	M01BB_1KM_NITE
Bands 1 and 2	"SV_250m"	M01SV_250M
Bands 3 through 7	"SV_500m"	M01SV_500M
Bands 8 through 19	"SV_1km_day"	M01SV_1KM_DAY
Bands 20 through 36	"SV_1km_night"	M01SV_1KM_NITE
FPA DCR offset data	"fpa_dcr_offst"	M01FPA_DCR_OFFST
FAM Registration sample Delays	"fam_samp_delay"	M01FAM_SAMP_DELAY

Metadata Description	Metadata Name	M-API Constant
Raw mirror encoder data	"raw_mir_enc"	M01RAW_MIR_ENC
Current/Prior HK Telem	"raw_hk_telem"	M01RAW_HK_TELEM
Sci Eng Data	"raw_sci_eng"	M01RAW_SCI_ENG
Parameter Table	"raw_param"	M01RAW_PARAM
View Sector Start	"raw_vs_start"	M01RAW_VS_START
CP Event Log	"raw_cp_event"	M01RAW_CP_EVENT
FR Event Log	"raw_fr_event"	M01RAW_FR_EVENT
Raw s/c ancill data	"raw_sc_ancil"	M01RAW_SC_ANCIL
Dump Request Info	"raw_dump_req"	M01RAW_DUMP_REQ
Dump Data	"raw_dump_data"	M01RAW_DUMP_DATA
FPA/AEM Config	"fpa_aem_config"	M01FPA_AEM_CONFIG

Table A-5. L1B/Geolocation Macros

Metadata Description	Metadata Name	M-API Constant
Product type identifier	"MOD02_L1B"	M02_PROD_ID
Software Version	"Software Version"	M02VERSION
Number of Scans	"Number of Scans"	M02NUMBER_OF_SCANS
Number of Day mode scans	"Number of Day mode scans"	M02NUMBER_OF_DAY_SCANS
Number of Night mode scans	"Number of Night mode scans"	M02NUMBER_OF_NIGHT_SCANS
Max Total Frames	"Max Total Frames"	M02MAX_TOTAL_FRAMES
Max Earth View Frames	"Max Earth Frames"	M02MAX_EARTH_FRAMES
Max SD Frames	"Max SD Frames"	M02MAX_SD_FRAMES
Max SRCA Frames	"Max SRCA Frames"	M02MAX_SRCA_FRAMES
Max BB Frames	"Max BB Frames"	M02MAX_BB_FRAMES
Max SV Frames	"Max SV Frames"	M02MAX_SV_FRAME
Scan types in product	"Scan types in product"	M02SCAN_TYPES
Dead MODIS Detectors	"Dead MODIS Detectors"	M02DEAD_DETECTORS
Noisy MODIS Detectors	"Noisy MODIS Detectors"	M02NOISY_DETECTORS
Dead Thermistors	"Dead Thermistors"	M02DEAD_THERMISTORS
Noisy Thermistors	"Noisy Thermistors"	M02NOISY_THERMISTORS
250 M Band Numbers for Reflected Solar Bands	"250 M Band Numbers for Reflected Solar Bands"	M02_250M_BAND_NUMS
500 M Band Numbers for Reflected Solar Bands	"500 M Band Numbers for Reflected Solar Bands"	M02_500M_BAND_NUMS
1000 M Band Numbers for Reflected Solar Bands	"1000 M Band Numbers for Reflected Solar Bands"	M02_1000M_REF_BAND_NUMS
Incomplete Scans	"Incomplete Scans"	M02PARTIAL_SCANS
Missing Packets	"Missing Packets"	M02MISSING_PACKETS
Packets with bad CRC	"Packets with bad CRC"	M02BAD_PACKETS
Discarded Packets	"Discarded Packets"	M02DISCARD_PACKETS
Swath Vgroup	"MODIS L1B Data"	M02SWATHWATH
num_scale_factors	"num_scale_factors"	M02NUM_SCALE_FACTORS
40*nscans	"40*nscans"	M02_40NSCANS
20*nscans	"20*nscans"	M02_20NSCANS
10*nscans	"10*nscans"	M02_10NSCANS
nscans	"nscans"	M02_NSCANS
40*nRefSBscans	M02_40NSCANS	M02_40NREFSBSCANS
20*nRefSBscans	M02_20NSCANS	M02_20NREFSBSCANS
10*nRefSBscans	M02_10NSCANS	M02_10NREFSBSCANS
Band_250M	"Band_250M"	M02BAND_250M
Band_500M	"Band_500M"	M02BAND_500M
Band_1KM_RefSB	"Band_1KM_RefSB"	M02BAND_1KM_REFSB
Band_1KM_Emissive	"Band_1KM_Emissive"	M02BAND_1KM_EMIS
4*BB frames	"4*BB_frames"	M02_4BB_FRAMES
2*BB frames	"2*BB_frames"	M02_2BB_FRAMES
BB frames	"BB_frames"	M02_BB_FRAMES

Metadata Description	Metadata Name	M-API Constant
4*EV frames	"4*EV_frames"	M02_4EV_FRAMES
2*EV frames	"2*EV_frames"	M02_2EV_FRAMES
EV frames	"EV_frames"	M02_EV_FRAMES
4*SD frames	"4*SD_frames"	M02_4SD_FRAMES
2*SD frames	"2*SD_frames"	M02_2SD_FRAMES
SD frames	"SD_frames"	M02_SD_FRAMES
4*SRCA frames	"4*SRCA_frames"	M02_4SRCA_FRAMES
2*SRCA frames	"2*SRCA_frames"	M02_2SRCA_FRAMES
SRCA frames	"SRCA_frames"	M02_SRCA_FRAMES
4*SV frames	"4*SV_frames"	M02_4SV_FRAMES
2*SV frames	"2*SV_frames"	M02_2SV_FRAMES
SV frames	"SV_frames"	M02_SV_FRAMES
Instrument Data Stored as Scientific Data Sets	"Slope_and_Offset"	M02SLOPE_AND_OFFSET
Black Body 250M Reflected Solar Bands Scaled Integer Radiance	"BB_250_RefSB_Rad"	M02BB_250
Black Body 250M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"BB_250_RefSB_Rad_Uncert"	M02BB_250_UNCERT
Earth View 250M Reflected Solar Bands Scaled Integer Radiance	"EV_250_RefSB_Rad"	M02EARTH_RAD_250
Earth View 250M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"EV_250_RefSB_Rad_Uncert"	M02EARTH_RAD_250_UNCERT
Solar Diffuser 250M Reflected Solar Bands Scaled Integer Radiance	"SD_250_RefSB_Rad"	M02DIFFUSER_250
Solar Diffuser 250M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"SD_250_RefSB_Rad_Uncert"	M02DIFFUSER_250_UNCERT
RCA 250M Reflected Solar Bands Scaled Integer Radiance	"SRCA_250_RefSB_Rad"	M02SRCA_250
SRCA 250M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"SRCA_250_RefSB_Rad_Uncert"	M02SRCA_250_UNCERT
Space View 250M Reflected Solar Bands Scaled Integer Radiance	"SV_250_RefSB_Rad"	M02SPACE_250
Space View 250M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"SV_250_RefSB_Rad_Uncert"	M02SPACE_250_UNCERT
Black Body 500M Reflected Solar Bands Scaled Integer Radiance	"BB_500_RefSB_Rad"	M02BB_500
Black Body 500M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"BB_500_RefSB_Rad_Uncert"	M02BB_500_UNCERT
Earth View 500M Reflected Solar Bands Scaled Integer Radiance	"EV_500_RefSB_Rad"	M02EARTH_RAD_500
Earth View 500M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"EV_500_RefSB_Rad_Uncert"	M02EARTH_RAD_500_UNCERT
Solar Diffuser 500M Reflected Solar Bands Scaled Integer Radiance	"SD_500_RefSB_Rad"	M02DIFFUSER_500
Solar Diffuser 500M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"SD_500_RefSB_Rad_Uncert"	M02DIFFUSER_500_UNCERT

Metadata Description	Metadata Name	M-API Constant
SRCA 500M Reflected Solar Bands Scaled Integer Radiance	"SRCA_500_RefSB_Rad"	M02SRCA_500
SRCA 500M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"SRCA_500_RefSB_Rad_Uncert"	M02SRCA_500_UNCERT
Space View 500M Reflected Solar Bands Scaled Integer Radiance	"SV_500_RefSB_Rad"	M02SPACE_500
Space View 500M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"SV_500_RefSB_Rad_Uncert"	M02SPACE_500_UNCERT
Black Body 1000M Reflected Solar Bands Scaled Integer Radiance	"BB_1000_RefSB_Rad"	M02BB_1000
Black Body 1000M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"BB_1000_RefSB_Rad_Uncert"	M02BB_1000_UNCERT
Black Body 1000M Emissive Bands Scaled Integer Radiance	"BB_1000_Emissive"	M02BB_EMIS_1000
Black Body 1000M Emissive Bands Scaled Integer Radiance Uncertainty	"BB_1000_Emissive_Uncert"	M02BB_EMIS_1000_UNCERT
Earth View 1000M Reflected Solar Bands Scaled Integer Radiance	"EV_1000_RefSB_Rad"	M02EARTH_RAD_1000
Earth View 1000M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"EV_1000_RefSB_Rad_Uncert"	M02EARTH_RAD_1000_UNCERT
Earth View 1000M Emissive Bands Scaled Integer Radiance	"EV_1000_Emissive_Rad"	M02EARTH_EMIS_RAD_1000
Earth View 1000M Emissive Bands Scaled Integer Radiance Uncertainty	"EV_1000_Emissive_Rad_Uncert"	M02EARTH_EMIS_RAD_1000_UNCERT
Solar Diffuser 1000M Reflected Solar Bands Scaled Integer Radiance	"SD_1000_RefSB_Rad"	M02DIFFUSER_1000
Solar Diffuser 1000M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"SD_1000_RefSB_Rad_Uncert"	M02DIFFUSER_1000_UNCERT
Solar Diffuser 1000M Emissive Bands Scaled Integer Radiance	"SD_1000_Emissive_Rad"	M02DIFFUSER_EMIS_1000
Solar Diffuser 1000M Emissive Bands Scaled Integer Radiance Uncertainty	"SD_1000_Emissive_Rad_Uncert"	M02DIFFUSER_EMIS_1000_UNCERT
SRCA 1000M Reflected Solar Bands Scaled Integer Radiance	"SRCA_1000_RefSB_Rad"	M02SRCA_1000
SRCA 1000M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"SRCA_1000_RefSB_Rad_Uncert"	M02SRCA_1000_UNCERT
SRCA 1000M Emissive Bands Scaled Integer Radiance	"SRCA_1000_Emissive_Rad"	M02SRCA_EMIS_1000
SRCA 1000M Emissive Bands Scaled Integer Radiance Uncertainty	"SRCA_1000_Emissive_Rad_Uncert"	M02SRCA_EMIS_1000_UNCERT
Space View 1000M Reflected Solar Bands Scaled Integer Radiance	"SV_1000_RefSB_Rad"	M02SPACE_1000
Space View 1000M Reflected Solar Bands Scaled Integer Radiance Uncertainty	"SV_1000_RefSB_Rad_Uncert"	M02SPACE_1000_UNCERT
Space View 1000M Emissive Bands Scaled Integer Radiance	"SV_1000_Emissive_Rad"	M02SPACE_EMIS_1000
Space View 1000M Emissive Bands Scaled Integer Radiance Uncertainty	"SV_1000_Emissive_Rad_Uncert"	M02SPACE_EMIS_1000_UNCERT

Metadata Description	Metadata Name	M-API Constant
Earth View 250M Reflected Solar Bands Scaled Integer Reflectance	"EV_250_RefSB_Refl"	M02EARTH_REFL_250
Earth View 250M Reflected Solar Bands Scaled Integer Reflectance Uncertainty	"EV_250_RefSB_Refl_Uncert"	M02EARTH_REFL_250_UNCERT
Earth View 500M Reflected Solar Bands Scaled Integer Reflectance	"EV_500_RefSB_Refl"	M02EARTH_REFL_500
Earth View 500M Reflected Solar Bands Scaled Integer Reflectance Uncertainty	"EV_500_RefSB_Refl_Uncert"	M02EARTH_REFL_500_UNCERT
Earth View 1000M Reflected Solar Bands Scaled Integer Reflectance	"EV_1000_RefSB_Refl"	M02EARTH_REFL_1000
Earth View 1000M Reflected Solar Bands Scaled Integer Reflectance Uncertainty	"EV_1000_RefSB_Refl_Uncert"	M02EARTH_REFL_1000_UNCERT
Eng. Packet 1 Data	"engineering_pkt_1"	M02ENG_PKT_1
Eng. Packet 2 Data	"engineering_pkt_2"	M02ENG_PKT_2
Mem. Packet 1 Data	"memory_pkt_1"	M02MEM_PKT_1
Mem. Packet 2 Data	"memory_pkt_2"	M02MEM_PKT_2
FPA DCR offset Data	"dcr_offset"	M02FPA_DCR_OFFSET
FAM Registration Sample Delays	"fam_sample_delay"	M02FAM_DELAY
Raw Mirror Encoder Data	"mirror_encoder"	M02MIRROR_ENCODER
Current/Prior HK Telemetry	"hk_telemetry"	M02HK_TELEM
Science Engineering Data	"science_engineering"	M02SCI_ENG
Parameter Table	"parameter_table"	M02PARM_TABLE
View Sector Start	"view_vector_start"	M02VIEW_START
CP Event Log	"cp_event_log"	M02CP_LOG
FR Event Log	"fr_event_log"	M02FR_LOG
Raw S/C Ancillary Data	"spacecraft_ancillary_data"	M02SC_ANCIL
Dump Request Information	"dump_request_info"	M02DUMP_REQUEST
Dump Data	"dump_data"	M02DUMP
Instrument Telemetry	"instrument_telemetry"	M02INSTR_TELEM
Level 1B Swath Metadata Written as Vdata	"Level 1B Swath Metadata"	M02SWATH_MD
Scan Number /* I32 */	"Scan number"	M02SW_SCAN_NO
Total Frames /* I32 */	"Total frames"	M02SW_TOT_FRAMES
EV Frames /* I32 */	"EV frames"	M02SW_EV_FRAMES
SD Frames /* I32 */	"SD frames"	M02SW_SD_FRAMES
SRCA Frames /* I32 */	"SRCA Frames"	M02SW_SRCA_FRAMES
BB Frames /* I32 */	"BB Frames	M02SW_BB_FRAMES
SV Frames /* I32 */	"SV frames"	M02SW_SV_FRAMES
Scan Type /* TXT */	"Scan Type"	M02SW_SCAN_TYPE
Scan Start Time /* F64 */	"Scan start time"	M02SW_SCAN_START
Mirror Side /* I32 */	"Mirror Side"	M02SW_MIR_SIDE
Missing Packets /* I32 */	"Missing Packets"	M02SW_MISS_PKTS
Packets With Bad CRC /* I32 */	"Packets With Bad CRC"	M02SW_BAD_PKTS
Discarded Packets /* I32 */	"Discarded Packets"	M02SW_DISC_PKTS
Moon in SV Port /* I32 */	"Moon in SV Port"	M02SW_MOON_OBS
On-Orbit Maneuver /* TXT */	"On-Orbit Maneuver"	M02SW_MANEUVER

Metadata Description	Metadata Name	M-API Constant
No. SV Outliers /* I32 */	"No. SV Outliers"	M02SW_NUM_SV_OUTLIERS
No. BB Outliers /* I32 */	"No. BB Outliers"	M02SW_NUM_BB_OUTLIERS
No. thermistor outliers /* I32 */	"No. thermistor outliers"	M02SW_NUM_THERM_OUTLIER
Product type identifier	"MOD03_Geolocation"	M03_PROD_ID
Mirror axis error bias (gamma)	"gamma"	M03GAMMA
Nominal mirror rotation rate	"mir_rate"	M03MIR_RATE
Sample interval for 1 km bands	"t_frame"	M03T_FRAME
Mirror side 1 encoder-to-angle conversion coefficients (quadratic)	"poly_m1"	M03POLY_M1
Mirror side 2 encoder-to-angle conversion coefficients (quadratic)	"poly_m2"	M03POLY_M2
Spacecraft-to-instrument transformation matrix	"T_inst2sc"	M03T_INST2SC
Instrument-to-mirror transformation matrix	"T_mirr2inst"	M03T_MIRR2INST
Instrument-to-telescope transformation matrix	"T_tel2inst"	M03T_TEL2INST
Focal length for detectors (0 is ideal)	"Focal_length"	M03FOCAL_LENGTH
Band readout times relative to ideal band	"T_offset"	M03T_OFFSET
ECR orbit position at scan center time	"orb_pos"	M03ORB_POS
ECR orbit velocity at scan center time	"orb_vel"	M03ORB_VEL
ECR-to-instrument frame transformation matrix at scan center time	"T_inst2ECR"	M03T_INST2ECR
Spacecraft angular velocity in instrument frame	"ang_vel"	M03ANG_VEL
Unit Sun vector in ECR frame at scan center time	"sun_ref"	M03SUN_REF
Number of mirror encoder samples for this scan	"num_impulse"	M03NUM_IMPULSE
Mirror angles from encoder data	"impulse_enc"	M03IMPULSE_ENC
Mirror encoder sample times from start of scan	"impulse_time"	M03IMPULSE_TIME
Band-to-band geometric correction coefficients (algorithm based on ATBD)	"band_geo"	M03BAND_GEO
Geodetic longitude	"longitude"	M03LONGITUDE
Geodetic latitude	"latitude"	M03LATITUDE
Height above ellipsoid	"height"	M03HEIGHT
Sensor zenith	"SensorZenith"	M03SENSOR_ZEN
Sensor azimuth	"SensorAzimuth"	M03SENSOR_AZ
Range (pixel to sensor)	"Range"	M03RANGE
Solar zenith	"SolarZenith"	M03SOLAR_ZENITH
Solar azimuth	"SolarAzimuth"	M03SOLAR_AZIMUTH
Geolocation flags	"gflags"	M03GFLAGS

Table A-6. Atmosphere Macros

Metadata Description	Metadata Name	M-API Constant
MOD04_L2	"MOD04_L2"	M04L2_PROD_ID
MOD05_L2	"MOD05_L2"	M05L2_PROD_ID
MOD06_L2	"MOD06_L2"	M06L2_PROD_ID
MOD07_L2	"MOD07_L2"	M07L2_PROD_ID
MOD08_L2	"MOD08_L2"	M08L2_PROD_ID
MOD30_L2	"MOD30_L2"	M30L2_PROD_ID
MOD35_L2	"MOD35_L2"	M35L2_PROD_ID
MOD38_L2	"MOD38_L2"	M38L2_PROD_ID
1-km_Pixels_Per_Scan_Line	"1-km_pixels"	MAPIXELS_PER_SCAN
1-km_Scan_Lines_Per_Granule	"1-km_Scan_Lines_Per_Granule"	MALINES_PER_GRANULE
Corner latitude of 10x10 pixel array	"Lat"	MACORNER_LAT
Corner longitude of 10x10 pixel array	"Lon"	MACORNER_LON
Scanline number through center of 5x5 pixel array	“Scanline_Number"	MASCANLINE_NO
Satellite zenith angle at midpoint of 5x5 array	"Sat_Zenith_Angle"	MAZENITH_SAT
Solar zenith angle at midpoint of 5x5 array	"Sun_Zenith_Angle"	MAZENITH_SOLAR
Index indicating the surface geography type as either Water(0) or Land(1)	"Land_Sea_Flag"	MAGEO_FLAG
Surface temperature at midpoint of 5x5 pixel array	"Sfc_Temp"	MATEMP_SFC
Surface pressure at midpoint of 5x5 pixel array	"Sfc_Pres"	MAPRES_SFC
Estimated tropopause height	"Height_Tropopause"	MATROPOPAUSE
long_name	“long_name"	MALONG_NAME
sampling_factor	"sampling_factor"	MASAMPLING
scale_factor	“scale_factor"	MASCALE
add_offset	"add_offset"	MAOFFSET
units	"units"	MAUNIT
valid_range	“valid_range"	MARANGE
Number Of Cells Across Swath	"Cells Across Swath"	MACELLS_ACROSS
Number Of Cells Along Swath	“Cells Along Swath"	MACELLS_ALONG
Pixels Per Scan Line	"Pixels Per Scan Line"	MAPIXELS
Number of Scan Lines	“Number of Scan Lines"	MASCANLINE
Number of Bands	"Number of Bands"	M04BANDS
Observed land reflectances averaged on 10x10 1-km pixel array	"Avg_Refl"	M04LAND_REFLS
Land aerosol optical thickness (AOT) for continental model	“Opt_Thickness_M1"	M04LAND_OPT_THICK
Standard deviation of observed land reflectances	“Std_Dev_Refl"	M04LAND_REFLS_DEV
Land AOT for corrected model	“Opt_Thickness_M2"	M04LAND_OPT_THICK_COR

Metadata Description	Metadata Name	M-API Constant
Aerosol path radiance ratio (continental model) of red to blue channel (band 3/band 1)	"Aerosol_Path_Rad_Ratio"	M04LAND_RADIANCE_RATIO
Relative contribution of smoke/sulfate particles to dust in the computation of the aerosol optical depth	"Relative_Contribution"	M04LAND_CONTRIBUTION
Number of Clear Land Pixels in Band 3	"Number_of_Pixels_B3"	M04LAND_PIXELS_B3
Number of Clear Land Pixels in Band 1	"Number_of_Pixels_B1"	M04LAND_PIXELS_B1
Identification of retrieval procedure	"Procedure_ID"	M04LAND_PROC_ID
Aerosol type in one of four categories: continental, dust, sulfate, and smoke	"Aerosol_Type"	M04LAND_AERO_TYPE
Aerosol land error flag	"Error_Flag"	M04LAND_ERROR
Ocean AOT at 0.55 micron on 10x10 1-km pixel array	"Opt_Thickness"	M04OCEAN_OPT_THICK
Small-particle ocean AOT at 0.55 micron on 10x10 pixel array	"Opt_Thickness_Small"	M04OCEAN_OPT_THICK_S
Large-particle ocean AOT at 0.55 micron on 10x10 pixel array	"Opt_Thickness_Large"	M04OCEAN_OPT_THICK_L
Weight factor for combining large and small aerosol modes during retrieval. This parameter minimizes the least-squares error summed over spectral bands	"Error_Min_Factor"	M04OCEAN_ERROR
Solution number from 1 to 36	"Solution_Number"	M04OCEAN_SOLUTION
Observed ocean reflectances averaged on 10x10 1-km pixel array	"Avg_Refl"	M04OCEAN_REFLS
Look-Up Table of Aerosol Model Parameters and Values Vdata	"LUT_Data"	M04AEROSOL_LUT
small mode aerosol mean radius	"RGSS"	M04LUT_RGSS
large mode aerosol mean radius	"RGSB"	M04LUT_RGSB
standard deviation of small mode radius	"SIGMAS"	M04LUT_SIGMAS
standard deviation of large mode radius	"SIGMAB"	M04LUT_SIGMAB
CCN	"CCNS"	M04LUT_CCNS
small mode extinction coefficient for 5 wavelengths	"EXTS"	M04LUT_EXTS
large mode extinction coefficient for 5 wavelengths	"EXTB"	M04LUT_EXTB
moments order 1-4 of small mode particle radius	"MOMENTS"	M04LUT_MOMENTS
moments order 1-4 of large mode particle radius	"MOMENTB"	M04LUT_MOMENTB
small mode backscatter ratio for 5 wavelengths	"BACKSCTS"	M04LUT_BACKSCTS
large mode backscatter ratio for 5 wavelengths	"BACKSCTB"	M04LUT_BACKSCTB
small mode asymmetry factor for 5 wavelengths	"ASSYMS"	M04LUT_ASSYMS
large mode asymmetry factor for 5 wavelengths	"ASSYMB"	M04LUT_ASSYMB
small mode albedo for 5 wavelengths	"ALBEDOS"	M04LUT_ALBEDOS

Metadata Description	Metadata Name	M-API Constant
large mode albedo for 5 wavelengths	"ALBEDOB"	M04LUT_ALBEDOB
Total column water vapor amounts over clear land, and cloud scenes over land and ocean	"Column_Water_Vapor"	M05WATER_VAPOR
Index indicating cloud(0), no cloud(1), or cloud/no cloud determination not made(-1)	"Cloud_Qualifier"	M05CLOUD_QUAL
Number_Of_1-km_Bands	"Number_Of_1-km_Bands "	M06BANDS
Number of Channel Indices	"Number of Channel Indices"	M06CHANNEL_IND
Number of Channel Differences	"Number of Channel Differences"	M06CHANNEL_DIFF
Brightness temperatures for IR channels 27 - 36 at 5x5 1-km pixel resolution	"Brightness_Temp"	M06BRIGHT_TEMP
Sufficient number of cloudy pixels (0) or too few cloudy pixels (1) to be able to process 5x5 pixel array	"Processing_Flag"	M06PROCESS_FLAG
Spectral cloud forcing for IR channels 29, and 31 - 36	"Spec_Cloud_Forcing"	M06CLOUD_FORCING
value to indicate the method of cloud height determination	"Cloud_H_Method"	M06METHOD
Cloud top effective emissivity	"Cloudtop_Eff_Emi"	M06EMISSIVITY_CT
Cloud top pressure	"Cloudtop_Pres"	M06PRES_CT
Cloud top temperature	"Cloudtop_Temp"	M06TEMP_CT
Cloud fraction at 5x5 1-km pixel resolution	"Cloud_Fraction"	M06FRACTION
Separate cloud top pressure estimates from five radiances ratios	"Cloudtop_Pres_From_Ratios"	M06PRES_CT_RATIO
Cloud top pressure from IR window	"Cloudtop_Pres_IR"	M06PRES_CT_IR
Surface type index	"Sfc_Type"	M06SFC_TYPE
Radiance variance for channels 29, 31, and 32	"Radiance_Var"	M06RADIANCE
Brightness temperature differences between IR channels 29, 31, and 32	"Brightness_Temp_Diff"	M06BRIGHT_TEMP_DIFF
Cloud thermodynamic phase derived from infrared retrieval algorithm	"Cloud_Phase_IR"	M06PHASE_IR
Effective particle radius at 1-km resolution	"Eff_Particle_Rad"	M06EFF_RADIUS
Cloud optical thickness at 1-km pixel resolution	"Cloud_Opt_Thickness"	M06CLOUD_OPT_THICK
Cloud thermodynamic phase derived from visible/SW infrared retrieval algorithm	"Cloud_Phase_VIS"	M06PHASE_VIS
Statistics at 1-km pixel resolution	"Statistics"	M06STATISTICS
Total Column Ozone at 5x5 1-km pixel resolution	"Total_Ozone"	M07TOTAL_OZONE
Total Totals Atmospheric Stability Index	"Total_Totals"	M08TOTALS
Lifted Index Atmospheric Stability Index	"Lifted_Index"	M08LIFTED_INDEX
K Index Atmospheric Stability Index	"K_Index"	M08K_INDEX
Brightness temperatures for IR channels 20, 22-25, and 27-36	"Brightness_Temp"	M30BRIGHT_TEMP

Metadata Description	Metadata Name	M-API Constant
Guess temperature profile for 20 vertical levels	"Guess_Temp_Profile"	M30TEMP_PROF
Guess dewpoint temperature profile for 15 vertical levels	"Guess_DewP_Profile"	M30DEWP_TEMP_PROF
Retrieved temperature profile for 20 vertical levels	"Retr_Temp_Profile"	M30RETR_TEMP_PROF
Retrieved dewpoint temperature profile for 15 vertical levels	"Retr_DewP_Profile"	M30RETR_DEWP_TEMP_PROF
Index of pressure levels for the 15 vertical levels	"Index_Of_Pressure_Levels"	M30PRESS_LEVEL
Bit field mask containing the results of visible and infrared radiance cloud/no cloud tests	"Cloud_Mask"	M35CLOUD_MASK
Cell Frame Number	"Cell Frame Number"	M38CELL_FRAME
Cell Line Number	"Cell Line Number"	M38CELL_LINE
Atmospheric Water Vapor Parameter at 5x5 1-km pixel resolution	"Water Vapor"	M38WATER_VAPOR

Table A-7. Ocean Macros

Metadata Description	Metadata Name	M-API Constant
MOD27 HDF output file	"MOD27 HDF output file"	M27_PROD_ID
output_file_name	"output_file_name"	M27O_F_NAME
output_file_logical file number	"output_file_logical file number"	M27O_F_L_F_NUM
units_of_output_file_logical_file_number	"units_of_output_file_logical_file_number"	U_O_O_F_L_F_NUM
product_name	"product_name"	M27P_NAME
statistics_file_name	"statistics_file_name"	M27S_F_NAME
product_sum_total_over_all_regions	"product_sum_total_over_all_regions"	M27P_SUM
units_of_product_sum_total_over_all_regions	"units_of_product_sum_total_over_all_regions"	M27U_O_P_SUM
product_variance_total_over_all_regions	"product_variance_total_over_all_regions"	M27P_VAR
units_of_product_variance_total_over_all_regions	"units_of_product_variance_total_over_all_regions"	M27P_O_P_VAR
product_area_total_over_all_regions	"product_area_total_over_all_regions"	M27P_AREA
units_of_product_area_total_over_all_regions	"units_of_product_area_total_over_all_regions"	M27U_O_P_AREA
square km	"square km"	M27SQKM
number_of_regions_for_product	"number_of_regions_for_product"	M27P_NREGS
coordinate_system	"coordinate_system"	M27COORD_SYS
units_of_coordinate_system	"units_of_coordinate_system"	M27U_O_COORD_SYS
range_of_coordinate_system	"range_of_coordinate_system"	M27R_O_COORD_SYS
character_counter	"character_counter"	M27KCHAR
region_counter	"region_counter"	M27JREG
limit_of_region_counter	"limit_of_region_counter"	M27KLIM
function_order_counter	"function_order_counter"	M27KORD
product_cell_counter	"product_cell_counter"	M27KCELLS
name_of_regions	"name_of_regions"	M27NAME_R
limit_of_regions-deg_lat_and_deg_long	"limit_of_regions-deg_lat_and_deg_long"	M27LIM_R
area_of_regions-km_squared	"area_of_regions-km_squared"	M27AREA_R
independent_variables_of_regions	"independent_variables_of_regions"	M27IV_R
functions_used_in_regions	"functions_used_in_regions"	M27FUNCTIONS_R
order_of_functions_used_in_regions	"order_of_functions_used_in_regions"	M27ORD_R
coefficients_used_in_regions	"coefficients_used_in_regions"	M27COEFF_R
error_in_regions-gr_per_m3_per year	"error_in_regions-gr_per_m3_per year"	M27ERR_R
sum_in_regions-gr_per_m3_per_year	"sum_in_regions-gr_per_m3_per_year"	M27SUM_R
variance_in_regions-gr2_per_m6_per_year2	"variance_in_regions-gr2_per_m6_per_year2"	M27VAR_R
product_y-gr_per_m3_per_year	"product_y-gr_per_m3_per_year"	M27P_Y
product_error_ey-gr_per_m3_per_year	"product_error_ey-gr_per_m3_per_year"	M27P_EY

Table A-8. Land Macros

Metadata Description	Metadata Name	M-API Constant
Pixels_per_scan_line	"Pixels_per_scan_line"	MLPIXELS_PER_SCAN
Number_of_scan_lines	"Number_of_scan_lines"	MLNUMBER_OF_LINES
Pixels_per_line	"Pixels_per_line"	MLPIXELS_PER_LINE
Lines_per_tile	"Lines_per_tile"	MLLINES_PER_TILE
Total_observations	"Total Observations"	MLTOTAL_OBSERVATIONS
Num_parameters	"Num_parameters"	MLNUMBER_OF_PARAMS
Maximum_observations	"Maximum Observations"	MLMAX_OBSERVATIONS
Number_of_granules	"Number of Granules"	MLNUMBER_OF_GRANULES
Granule_IDs	"Granule_IDs"	MLGRANULE_IDS
File_Format	"L2G Storage Format"	MLFILE_FORMAT
Parameter1	"Parameter1"	MLPARM1
Parameter2	"Parameter2"	MLPARM2
Parameter3	"Parameter3"	MLPARM3
Parameter4	"Parameter4"	MLPARM4
Parameter5	"Parameter5"	MLPARM5
Parameter6	"Parameter6"	MLPARM6
Parameter7	"Parameter7"	MLPARM7
Year	"Year"	MLYEAR
Day_of_year	"Day_of_year"	MLDOY
nrow	"nrow"	MLNUMBER_OF_ROWS
nest_lev	"Grid Nesting Level"	MLNEST_LEVEL
ref_lon_in_deg	"ref_lon_in_deg"	MLREF_LONGITUDE
ang_size_in_arcsec	"Characteristic Bin Angular Dimension"	MLANGULAR_SIZE
irow_start	"irow_start"	MLIROW_START
ncol_max	"ncol_max"	MLNCOL_MAX
itile_horiz	"itile_horiz"	MLITILE_HORIZ
itile_vert	"itile_vert"	MLITILE_VERT
ntile_horiz	"ntile_horiz"	MLNTILE_HORIZ
ntile_vert	"ntile_vert"	MLNTILE_VERT
L2G number of observations per pixel contained within L2G file	"num_observations"	MLNUMBER_OF_OBS
The number of columns in the full ISCCP grid for each row (line) contained within the L2G file	"ncol"	MLNUMBER_OF_COLS
The start column in the full ISCCP grid for each row (line) contained within the L2G file (starting at zero).	"icol_start"	MLSTART_COLUMN
The number of columns in each row (line) contained within the L2G file.	"ncol_tile"	MLCOLS_PER_ROW
The start pixel of the first valid column in each row (line) contained within the L2G file (starting at zero).	"ipix_start"	MLSTART_PIX
Number of observations per line	"nobs_line"	MLOBS_PER_LINE

Metadata Description	Metadata Name	M-API Constant
SPSO_parameter	"SPSO_parameter"	MLSPSO_PARAMETERS
Product type identifier: MOD09_ANG_L2G_1KM	"MOD.AM1.geoang.L2G"	M09ANG_PROD_ID
Zenith angle to sensor	"SensorZenith"	M09SENSOR_ZENITH
Azimuth angle to sensor	"SensorAzimuth"	M09SENSOR_AZIMUTH
Distance to sensor	"Range"	M09SENSOR_DISTANCE
Zenith angle to sun	"SolarZenith"	M09SOLAR_ZENITH
Azimuth angle to sun	"SolarAzimuth"	M09SOLAR_AZIMUTH
Product type identifier: MOD09_PNT_L2G_1KM	"MOD.AM1.pntr_1km.L2G"	M09PNT1K_PROD_ID
Product type identifier: MOD09_PNT_L2G_500M	"MOD.AM1.pntr_500m.L2G"	M09PNT500_PROD_ID
Product type identifier: MOD09_PNT_L2G_250M	"MOD.AM1.pntr_250m.L2G"	M09PNT250_PROD_ID
Pointer to granule IDs from which the observation came. Zero relative. Fill value is 255.	"granule_pnt"	M09GRANULE_PNT
Sample number of observation (1 km spatial element) in granule	"sample"	M09OBS_IN_GRANULE
Sub-pixel (delta) line location of cell center in observation footprint. Relative to center of observation specified by (line, sample).	"dline"	M09CELL_CENTER
Sub-pixel (delta) line location of cell center in observation footprint SDS. Relative to center of observation specified by (line, sample).	"dsample"	M09SAMPLE_CENTER
Observation coverage SDS: area of intersection between observation footprint and cell divided by area of observation.	"obscov"	M09OBS_COVERAGE
Cell coverage SDS: area of intersection between observation footprint and cell divided by area of cell.	"cellcov"	M09CELL_COVERAGE
Product type identifier: MOD09_L2 and MOD13_L2	"MOD.AM1.V1.srefl_500m.L2G"	M09_L2G_500M_PROD_ID
Surface Reflectance for MODIS Band 3	"sur_refl_b03"	M09BAND3_SURF_REFL
Surface Reflectance for MODIS Band 4	"sur_refl_b04"	M09BAND4_SURF_REFL
Surface Reflectance for MODIS Band 5	"sur_refl_b05"	M09BAND5_SURF_REFL
Surface Reflectance for MODIS Band 6	"sur_refl_b06"	M09BAND6_SURF_REFL
Surface Reflectance for MODIS Band 7	"sur_refl_b07"	M09BAND7_SURF_REFL
Indicators of the quality of the 500 m reflectance data	"QC_500m"	M09QUALITY_500
Product type identifier: MOD09_L2 and MOD13_L2	"MOD.AM1.V1.srefl_250m.L2G"	M09_L2G_250M_PROD_ID
Surface Reflectance for MODIS Band 1	"sur_refl_b01"	M09BAND1_SURF_REFL
Surface Reflectance for MODIS Band 2	"sur_refl_b02"	M09BAND2_SURF_REFL
Indicators of the quality of the 250 m reflectance and VI data integrity.	"QC_250m"	M09QUALITY_250

Metadata Description	Metadata Name	M-API Constant
Product type identifier: MOD09_L2 and MOD13_L2 MOD09SUBS_L2G_16DY	"MOD.AM1.brdfsubs.L3"	M09_REFLDB_PROD_ID
ang_size (in arcsec)	"ang_size (in arcsec)"	M09_REFLDB_ANGULAR_SIZE
General information on observational basis M09_OBS_INFO words	"Obs_Info_Items"	M09_OBS_INFO_WORDS
Viewing and illumination angles	"Angles"	M09_ANGLES
N_obs_dy	"Num_Obs_Max"	M09_ANGLES_OBS
N_angles	"Num_Angles"	M09_ANGLES_NUM
Surface reflectances	"Surface_Refl"	M09_REFLDB_SURF_REFL
N_obs_dy	"Num_Obs_Max"	M09_SURF_REFL_OBS
N_bands	"Num_Land_Bands"	M09_SURF_REFL_BANDS
Quality and weights of the respective observations	"Weights_QC"	M09_QUALITY_WEIGHTS
N_obs_dy	"Num_Obs_Max"	M09_QUALITY_OBS
words	"Num_Weights_QC"	M09_QUALITY_WORDS
Product type identifier: MOD09_BARS	"MOD.AMI.bars_16dy.L3"	M09BARS_PROD_ID
Nadir-equivalent surface reflectances for MODIS bands 1-7	"BARS"	M09BARS
Overall quality of the BRDF-adjusted surface reflectances	"BARS_QC"	M09BARS_QC
The number of columns in the full ISCCP grid for each row (line) contained within this L2G file.	"ncols"	M09NCOL
The start column in the full ISCCP grid for each row (line) contained within this L2G file (starting at zero).	"icol_start"	M09ICOL_START
The number of columns in each row (line) contained within this L2G file.	"ncol_tile"	M09NCOL_TILE
The start pixel of the first valid column in each row (line) contained within this L2G file (starting at zero).	"ipix_start"	M09IPIX_START
Product type identifier: MOD09_L3_16DY_G	"MOD_AM1.brdf_16dy.L3"	M09_L3_PROD_ID
Identifier for BRDF models chosen	"BRDF_Model_ID"	M09BRDF_MODEL_ID
RMSE for BRDF models chosen	"BRDF_Model_RMSE"	M09BRDF_MODEL_RMSE
BRDF quality control	"Quality_Control"	M09QUALITY
BRDF parameters for the seven land bands	"BRDF_Parameters"	M09BRDF_PARAMETERS
Albedo parameters for broadband, < 0.7 mu-m, > 0.7 mu-m, and the seven land bands.	"Albedo"	M09ALBEDO
A measure of fit from RMSE and sampling of all models tested.	"Fit_Assessments"	M09FIT_ASSESS
The number of columns in the full ISCCP grid for each row (line) contained within this L3 file.	"ncol"	M09NCOL
The start column in the full ISCCP frid for each row (line) contained within this L3 file (starting at zero).	"icol_start"	M09ICOL_START
The number of columns in each row (line) contained within this L3 file.	"ncol_tile"	M09NCOL_TILE

Metadata Description	Metadata Name	M-API Constant
The start pixel of the first valid column in each row (line) contained within this L3 file (starting at zero).	"ipix_start"	M09IPIX_START
N_select_models	"N_select_models"	M09N_SELECT_MODELS
words	"words"	M09WORDS
land_bands	"Num_Land_Bands"	M09LAND_BANDS
number_parameters	"Num_BRDF_Parameters"	M09NUMBER_PARAMETERS
land_bands_and_broadband_and_<>_0.7mu-m	"land_bands_and_broadband_and_<>_0.7mu-m"	M09LANDBANDS_BROADBAND_OTHER
N_models	"N_models"	M09N_MODELS
Product type identifier: MOD09_L2 and MOD13_L2	MOD.AM1.srefl.L2	M0913_L2_PROD_ID
SurfaceReflectance for MODIS Band 1 SDS	"sur_refl.b01"	M0913BAND1_SURF_REFL
SurfaceReflectance for MODIS Band 2 SDS	"sur_refl.b02"	M0913BAND2_SURF_REFL
SurfaceReflectance for MODIS Band 3 SDS	"sur_refl.b03"	M0913BAND3_SURF_REFL
SurfaceReflectance for MODIS Band 4 SDS	"sur_refl.b04"	M0913BAND4_SURF_REFL
SurfaceReflectance for MODIS Band 5 SDS	"sur_refl.b05"	M0913BAND5_SURF_REFL
SurfaceReflectance for MODIS Band 6 SDS	"sur_refl.b06"	M0913BAND6_SURF_REFL
SurfaceReflectance for MODIS Band 7 SDS	"sur_refl.b07"	M0913BAND7_SURF_REFL
NDVI index at 250m	"NDVI_index"	M0913_NDVI_INDEX
MVI index at 250m	"MVI_index"	M0913_MVI_INDEX
Indicators of the quality of the 250m reflectance and VI data integrity.	"QC_250m"	M0913QUALITY_250
Indicators of the quality of the 500m reflectance and VI data integrity.	"QC_500m"	M0913QUALITY_500
num_detectors	"num_detectors"	M0913NUM_DETECTORS
sampling	"sampling"	M0913SAMPLING
Number_of_pixels_processed	"Number_of_pixels_processed"	M10PROCESSED_PIXELS
Total_snow_pixels	"Total_snow_pixels	M10SNOW_PIXELS
Percentage_snow	"Percentage_snow"	M10PERCENT_SNOW
Percentage_not_snow	"Percentage_not_snow"	M10PERCENT_NOT_SNOW
Above_range_NDSI	"Above_range_NDSI"	M10NDSI_ABOVE
Below_range_NDSI	"Below_range_NDSI"	M10NDSI_BELOW
Division_by_zero	"Division_by_zero"	M10ZERO_DIVIDE
Out_of_range_input	"Out_of_range_input"	M10OUT_OF_RANGE_INPUT
No_decision	"No_decision"	M10NO_DECISION
L2/L2G Identification of daily snow cover on the land surface	"daily_snow_cover"	M10DAILY_SNOW
Product type identifier: MOD10_L2G	"MOD.AM1.V1.snow.L2G"	M10L2G_PROD_ID
Product type identifier: MOD10_L3_DY_G	"MOD.AM1.V1.snow_dy.L3"	M10L3_PROD_ID
L3 Identification of daily snow cover on the land surface	"Daily_Gridded_Snow_Cover"	M10GRIDDED_SNOW
Product type identifier: MOD11_L2	"MOD.AM1.V2.lst.L2"	M11L2_PROD_ID
L2/L2G Identification of Land Surface Temperature	"LST"	M11SURF_TEMP
L2/L2G LST Quality Indicator	"QC"	M11QUALITY

Metadata Description	Metadata Name	M-API Constant
L2/L2G Error in land surface temperature measurements	"Error_LST"	M11ERRORS
L2/L2G/L3 Band 31 emissivity	"Emis_31"	M11BAND31_EMIS
L2/L2G/L3 Band 32 emissivity	"Emis_32"	M11BAND32_EMIS
L2/L2G Band 29 or band 20 emissivity	"Emis_29"	M11BAND29OR20_EMIS
Product type identifier: MOD11_L2G	"MOD11_L2G"	M11L2G_PROD_ID
Product type identifier: MOD11_L3_WK_G	"MOD.AM1.V2.lst_1dy_cmg.L3"	M11L3_1DY_PROD_ID
L3 Identification of Land Surface Temperature	"LST"	M11L3SURF_TEMP
Land surface temperature in view within 45deg	"LST_view<45d"	M11NARROW_LST
L3 LST Quality Indicator	"QC"	M11L3QUALITY
Land-Surface Temperature Standard Deviation	"Stdv_LST"	M11STD_DEV
L3 Band 29 or band 20 emissivity	"Emis_29"	M11L3BAND29OR20_EMIS
Angular coefficients for Band 31 emissivity	"Ang_Coef_Emis_31"	M11BAND31_ANG_COEFS
Angular coefficients for Band 32 emissivity	"Ang_Coef_Emis_32"	M11BAND32_ANG_COEFS
Product type identifier: MOD12_L3_3MN_D/MOD12_L3_3MN_F	"MOD.AM1.V2.lc_1km.L3.3m"	M12L3_PROD_ID
ang_size (in arcsec)	"ang_size (in arcsec)"	M12ANGULAR_SIZE
Identification of land cover type	"Land Cover Type"	M12LAND_COVER
Identification of Overall quality of the land cover	"Type Overall QC"	M12QUALITY
Identification of Number of products generated since last classification update	"Num Product Gen"	M12PRODS_GENERATED
Identification of Number of snow months over pervious 12 months	"Snow Months"	M12SNOW_MONTHS
Identification of Number of BRDFs used for classification that have been derived within the past 12 months	"Num BRDF"	M12BRDFS_USED
Identification of Confidence in BRDF/reflectance correction	"BRDF Internal QC"	M12BRDF_STOCK
Identification of Number of LST values used for classification	"Num LST"	M12LST_VALS_USED
Identification of Confidence in VI over 12 months	"VI Internal QC"	M12VI_STOCK
Identification of TBD quality control for land cover type	"Land_cover_TBD_1"	M12QUALITY1
Identification of TBD quality control for land cover type	"Land_cover_TBD_2"	M12QUALITY2
Identification of Land cover change	"Land Cover Change"	M12LAND_COVER_CHANGE
Identification of Quality control for land cover change	"Land Cover Change QC"	M12CHANGE_QUALITY
Product type identifier: MOD14_L2	"MOD14_L2"	M14L2_PROD_ID
L2/L2G Identification of fire on the land surface	"fire_mask"	M14LAND_FIRE

Metadata Description	Metadata Name	M-API Constant
L2/L2G/L3 Total emmitted energy detected	"power"	M14ENERGY
L2/L2G/L3 Class of fire detected	"smold"	M14FIRE_CLASS
Fire quality control	"fire_qc"	M14QUALITY
Product type identifier: MOD14_L2G	"MOD.AM1.V2.fire.L2G"	M14L2G_PROD_ID
L2G/L3 Fire quality control	"fire_qc"	M14L2GQUALITY
Product type identifier: MOD14_L3	"MOD.AM1.V2.fire_daily.L3"	M14L3_PROD_ID
Product type identifier: MOD29_L2	"MOD.AM1.V2.seaice_max.L2"	M29L2_PROD_ID
Total_sea_ice_pixels	"Total_sea_ice_pixels"	M29SEA_ICE_PIXELS
Percentage_sea_ice	"Percentage_sea_ice"	M29SEA_ICE_PERCENT
Percentage_not_sea_ice	"Percentage_not_sea_ice"	M29NOT_SEA_ICE_PERCENT
Above_range_NDSI	"Above_range_NDSI"	M29NDSI_ABOVE
Below_range_NDSI	"Below_range_NDSI"	M29NDSI_BELOW
Division_by_zero	"Division_by_zero"	M29ZERO_DIVIDE
Out_of_range_input	"Out_of_range_input"	M29OUT_OF_RANGE
No_decision	"No_decision"	M29NO_DECISION
Identification of daily sea ice cover	"daily_sea_ice_cover"	M29DAILY_SEA_ICE
Product type identifier: MOD29_L2G	"MOD29_L2G"	M29L2G_PROD_ID
Daily Ice Cover	"daily_ice_cover"	M29L2GDAILY_SEA_ICE
Product type identifier: MOD29_L3_DY_G	"MOD.AM1.seaice_max_dy.L3"	M29L3_PROD_ID
Identification of daily sea ice cover	"daily_gridded_sea_ice_cover"	M29L3DAILY_SEA_ICE
Product type identifier: MOD33_L3_WK_G	"MOD.AM1.V2.snow_10dy.L3"	M33L3_PROD_ID
Weekly Snow Cover	"Composite_Snow_Cover"	M33WEEKLY_SNOW
Product type identifier: MOD34_L3_MN	"MOD.AM1.v1_1m.L3"	M34L3_PROD_ID
NDVI	"NDVI_250_M"	M34NDVI
MVI	"MVI_250_M"	M34MVI
View zenith angles for NDVI	"VZ_NDVI"	M34NDVI_ZENITH_ANGLES
View zenith angles for MVI	"VZ_MVI"	M34MVI_ZENITH_ANGLES
Quality control for NDVI	"NDVI_250_M_QC"	M34NDVI_QUALITY
Quality control for MVI	"MVI_250_M_QC"	M34MVI_QUALITY
Product type identifier: MOD42_L3_WK_G	"MOD.AM1.V1.seaice_10dy.L3"	M42L3_PROD_ID
Weekly Sea Ice Cover	"Composite_Ice_Cover"	M42WEEKLY_SEA_ICE

(This page intentionally left blank.)