

DriveMap Objects

Introduction

A **DriveMap** object is used to access information in a drive map file.

Methods

Create, Delete, Validate Methods

[meiDriveMapCreate](#)

[meiDriveMapDelete](#)

[meiDriveMapValidate](#)

Configuration and Information Methods

[meiDriveMapParamCount](#)

[meiDriveMapParamList](#)

[meiDriveMapConfigCount](#)

[meiDriveMapConfigList](#)

Data Types

[MEIDriveMapMessage](#)

[MEIDriveMapParamAccess](#)

[MEIDriveMapParamInfo](#)

[MEIDriveMapParamType](#)

[MEIDriveMapParamValue](#)

Constants

[MEIDriveMapParamMAX_STRING_LENGTH](#)

meiDriveMapCreate

Declaration `meiDriveMapCreate (char *fileName)`

Required Header `stdmei.h`

Description `DriveMapCreate` creates a DriveMap object associated with the file specified by *fileName*.

DriveMapCreate is the equivalent of a C++ constructor.

*filename	the drive map file.
------------------	---------------------

Return Values

handle	to a DriveMap object. After creating a DriveMap object, it must be validated using <code>meiDriveMapValidate()</code> .
MPIHandleVOID	if the object could not be created.

See Also [meiDriveMapDelete](#) | [meiDriveMapValidate](#)

meiDriveMapDelete

Declaration `meiDriveMapDelete(MEIDriveMap driveMap);`

Required Header `stdmei.h`

Description **DriveMapDelete** deletes a DriveMap object and invalidates its handle.

DriveMapDelete is the equivalent of a C++ destructor.

driveMap	a handle of the DriveMap object to delete in the reverse order to avoid memory leaks.
-----------------	---

Return Values

MPIMessageOK	if <i>DriveMapDelete</i> successfully deleted the object.
---------------------	---

See Also [meiDriveMapCreate](#) | [meiDriveMapValidate](#)

meiDriveMapValidate

Declaration long `meiDriveMapValidate`([MEIDriveMap](#) `driveMap`);

Required Header `stdmei.h`

Description **DriveMapValidate** validates a DriveMap object and its handle.

DriveMapValidate is the equivalent of a C++ constructor.

driveMap	a handle to SqNode object.
-----------------	----------------------------

Return Values

MPIMessageOK	if <i>driveMap</i> is a handle to a valid object.
---------------------	---

See Also [meiDriveMapCreate](#) | [meiDriveMapDelete](#)

meiDriveMapParamCount

Declaration

```
long meiDriveMapParamCount (MEIDriveMap driveMap,
                             char *nodeName,
                             char *firmwareVersion,
                             long *paramsCount) ;
```

Required Header stdmei.h

Description

DriveMapParamCount scans the drive map file for a drive entry that matches a particular drive. If an entry is found, then this function returns the number of drive parameters that need to be preserved for the configuration of the drive.

This function is normally used with the [meiDriveMapParamList](#) function. First, this function is called in order to get the size of the drive parameter list. Then the user can use this size to allocate enough memory to hold the complete parameter list before calling `meiDriveMapParamList` to fill in the list.

driveMap	a handle to a DriveMap object.
nodeName	the product/manufacturing text string of the node to search for. The nodeName of an SqNode object can be retrieved by calling <code>meiSqNodeInfo</code> . See MEISqNodeInfo .
firmwareVersion	The firmware version of the drive to search for. This information can be retrieved from an SqNode object by calling <code>meiSqNodeDriveInfo</code> . See MEISqNodeDriveInfo .
*paramsCount	pointer to the variable that will be set by this function.

Return Values

MPIMessageOK	if <i>meiDriveMapParamCount</i> successfully scans the drive map file for a drive entry that matches this node on the network and returns the number of drive parameters that need to be preserved for the configuration of the drive.
---------------------	--

See Also [meiDriveMapParamList](#)

meiDriveMapParamList

Declaration

```
long meiDriveMapParamList (MEIDriveMap      driveMap,
                           char             *nodeName,
                           char             *firmwareVersion,
                           long             paramCount,
                           MEIDriveParamInfo *driveParamInfo);
```

Required Header stdmei.h

Description

SqNodeDriveMapScanParamsList scans the drive map file for an entry that matches a particular drive. If a drive entry is found, this function writes the drive parameter information about each of the drive parameters to the *driveParamInfo* list.

This function is normally used with the meiDriveMapParamCount function. The meiDriveMapParamCount function is called first to get the size of the parameter list, the user can then use this size to allocate enough memory to hold the complete parameter list before calling this function to fill in the parameter list.

driveMap	a handle to a DriveMap object.
*nodeName	the product/manufacturing text string of the node to search for. The nodeName of an SqNode object can be retrieved by calling meiSqNodeInfo. See MEISqNodeInfo .
*firmwareVersion	The firmware version of the drive to search for. This information can be retrieved from an SqNode object by calling meiSqNodeDriveInfo. See MEISqNodeDriveInfo .
paramCount	the number of drive parameter information records that can be written to the driveParamInfo list.
*driveParamInfo	pointer to the list of drive parameter information records that will be filled in by this function.

Return Values

MPIMessageOK	if <i>meiDriveMapParamList</i> successfully scans the drive map file for an entry that matches the node on the network and writes the information about each of the drive parameters to the <i>driveParamInfo</i> list.
---------------------	---

See Also [meiDriveMapParamCount](#)

meiDriveMapConfigCount

Declaration

```
long meiDriveMapConfigCount (MEIDriveMap driveMap,
                             char *nodeName,
                             char *firmwareVersion,
                             long *configCount) ;
```

Required Header stdmei.h

Description

DriveMapConfigCount scans the drive map file for a drive entry that matches a particular drive. If an entry is found, this function returns the number of drive parameters that need to be preserved for the configuration of the drive.

This function is normally used with the `meiDriveMapConfigList` function. This function is called first in order to get the size of the drive configuration list. Then the user can use this size to allocate enough memory to hold the complete configuration list before calling `meiDriveMapConfigList` to fill in the list.

driveMap	a handle to a DriveMap object.
nodeName	the product/manufacturing text string of the node to search for. The nodeName of an SqNode object can be retrieved by calling <code>meiSqNodeInfo</code> . See MEISqNodeInfo .
firmwareVersion	The firmware version of the drive to search for. This information can be retrieved from an SqNode object by calling <code>meiSqNodeDriveInfo</code> . See MEISqNodeDriveInfo .
*configCount	pointer to the variable that will be set by this function.

Return Values

MPIMessageOK	if <i>meiDriveMapConfigCount</i> successfully scans the drive map file for a drive entry that matches this node on the network and returns the number of drive parameters that need to be preserved for the configuration of the drive.
---------------------	---

See Also [meiDriveMapConfigList](#)

meiDriveMapConfigList

Declaration

```
long meiDriveMapConfigList (MEIDriveMap driveMap,
                             char *nodeName,
                             char *firmwareVersion,
                             long configCount,
                             long *configList);
```

Required Header `stdmei.h`

Description

DriveMapConfigList scans the drive map file for a drive entry that matches a particular drive. If an entry is found, this function returns the list of drive parameters that need to be preserved for the configuration of the drive.

This function is normally used with the [meiDriveMapConfigCount](#) function. The `meiDriveMapConfigCount` function is called first in order to get the size of the drive configuration list. Then the user can use this size as a guide to allocate enough memory to hold the complete configuration list before calling this function to fill in the list.

driveMap	a handle to a DriveMap object.
*nodeName	the product/manufacturing text string of the node to search for. The nodeName of an SqNode object can be retrieved by calling <code>meiSqNodeInfo</code> . See MEISqNodeInfo .
*firmwareVersion	The firmware version of the drive to search for. This information can be retrieved from an SqNode object by calling <code>meiSqNodeDriveInfo</code> . See MEISqNodeDriveInfo .
configCount	the number of drive parameter information records that can be written to the configList list.
*configList	pointer to the list of drive parameters that make up the drive configuration that will be filled in by this function.

Return Values

MPIMessageOK	if <i>meiDriveMapConfigList</i> successfully scans the drive map file for a drive entry that matches this node on the network and returns the list of drive parameters that need to be preserved for the configuration of the drive.
---------------------	--

See Also [meiDriveMapConfigCount](#)

MEIDriveMapMessage

MEIDriveMapMessage

```
typedef enum {  
    MEIDriveMapMessageMAP_FILE_OPEN_ERROR,  
    MEIDriveMapMessageMAP_FILE_FORMAT_INVALID,  
    MEIDriveMapMessageNODE_NOT_FOUND_IN_MAP,  
    MEIDriveMapMessageVERSION_NOT_FOUND_IN_MAP,  
    MEIDriveMapMessageDRIVE_PARAM_READ_ONLY,  
} MEIDriveMapMessage;
```

Description

MEIDriveMapMessageMAP_FILE_OPEN_ERROR

There was an error when opening the drive map file. The file may not exist, or access to the directory may not be allowed.

MEIDriveMapMessageMAP_FILE_FORMAT_INVALID

The format of the drive map file is invalid.

MEIDriveMapMessageNODE_NOT_FOUND_IN_MAP

The node type specified by the nodeName parameter was not found in the driveMap file.

MEIDriveMapMessageVERSION_NOT_FOUND_IN_MAP

The drive firmware version specified by the firmwareVersion parameter was not found in the driveMap file.

MEIDriveMapMessageDRIVE_PARAM_READ_ONLY

A read-only parameter is included in the configuration list for the specified drive in the driveMap file.

See Also

MEIDriveMapParamAccess

MEIDriveMapParamAccess

```
typedef enum MEISqNodeDriveParamAccess {  
    MEIDriveMapParamAccessREAD_WRITE,  
    MEIDriveMapParamAccessREAD_ONLY,  
} MEIDriveMapParamAccess;
```

Description

The **DriveMapParamAccess** structure indicates what type of access is possible with the drive parameter.

This field of the [MEIDriveMapParamInfo](#) structure indicates what type of access is possible with this drive parameter.

See Also

[MEIDriveMapParamInfo](#)

MEIDriveMapParamInfo

MEIDriveMapParamInfo

```
typedef struct MEISqNodeDriveParamInfo {
    unsigned long           parameter;
    char                   name[MEISqNodeDriveParamMAX_STRING_LENGTH];
    MEIDriveMapParamAccess access; /* 0=>rw, 1=>ro */
    MEIDriveMapParamType  type;
    char                   validValues[MEISqNodeDriveParamMAX_STRING_LENGTH];
    long                   defaultValue;
    char                   help[MEISqNodeDriveParamMAX_STRING_LENGTH];
} MEIDriveMapParamInfo;
```

Description

The [DriveMapParamInfo](#) structure holds a set of information describing a drive parameter. This structure is read from the drives map file "drives.dm."

parameter	the number used to address this drive parameter.
name	a null terminated string giving a name for this drive parameter.
access	defines if this drive parameter is read-only or read-write.
type	the data type for this drive parameter. (ex: signed32, unsigned32, float, etc.)
validValues	a string describing the possible valid values for this drive parameter.
defaultValue	The factory default value this drive parameter.
help	A string describing this drive parameter.

See Also

[MEIDriveMapParamType](#)

MEIDriveMapParamType

MEIDriveMapParamType

```
typedef enum MEIDriveParamType {
    MEISqNodeDriveParamTypeSIGNED32,
    MEISqNodeDriveParamTypeSIGNED16,
    MEISqNodeDriveParamTypeSIGNED8,
    MEISqNodeDriveParamTypeUNSIGNED32,
    MEISqNodeDriveParamTypeUNSIGNED16,
    MEISqNodeDriveParamTypeUNSIGNED8,
    MEISqNodeDriveParamTypeHEX,
    MEISqNodeDriveParamTypeENUMERATED,
    MEISqNodeDriveParamTypeMASK,
    MEISqNodeDriveParamTypeCHARACTER,
    MEISqNodeDriveParamTypeSTRING,
    MEISqNodeDriveParamTypeSINGLE,
    MEISqNodeDriveParamTypeACTION,
} MEIDriveMapParamType;
```

Description

DriveMapParamType is an enumeration that indicates which data format should be used with the drive parameter.

MEISqNodeDriveParamTypeSIGNED32	A signed 32bit integer.
MEISqNodeDriveParamTypeSIGNED16	A signed 16bit integer.
MEISqNodeDriveParamTypeSIGNED8	A signed 8bit integer.
MEISqNodeDriveParamTypeUNSIGNED32	An unsigned 32bit integer.
MEISqNodeDriveParamTypeUNSIGNED16	An unsigned 16bit integer.
MEISqNodeDriveParamTypeUNSIGNED8	An unsigned 8bit integer.
MEISqNodeDriveParamTypeHEX	An integer represented in hexadecimal notation.
MEISqNodeDriveParamTypeENUMERATED	An integer where each value can be represented by a different name.
MEISqNodeDriveParamTypeMASK	An integer where each bit has a unique name.
MEISqNodeDriveParamTypeCHARACTER	An single ASCII character.
MEISqNodeDriveParamTypeSTRING	A null terminated string of characters.
MEISqNodeDriveParamTypeSINGLE	A single precision floating point number.
MEISqNodeDriveParamTypeACTION	Writing to this parameter will perform an action on the drive. No data is associated with this type.

MEIDriveMapParamValue

MEIDriveMapParamValue

```
typedef union {
    long          signed32;
    short        signed16;
    char         signed8;
    unsigned long unsigned32;
    unsigned short unsigned16;
    unsigned char unsigned8;
    unsigned long enumerated;
    unsigned long hex;
    unsigned long mask;
    char         character;
    char         * string;
    float        single;
} MEIDriveMapParamValue;
```

Description

The **MEIDriveMapParamValue** union holds the value of a drive parameter. The different fields allow this data type to hold all the different types of drive parameters. The MEISqNodeDriveParamType enumeration is used to identify which of the fields within the MEISqNodeDriveParamValue union to use.

See Also [MEIDriveMapParamType](#)

MEIDriveMapParamMAX_STRING_LENGTH

MEIDriveMapParamMAX_STRING_LENGTH

```
#define MEIDriveMapParamMAX_STRING_LENGTH (256)
```

Description

The **DriveMapParamMAX_STRING_LENGTH** macro defines the maximum length of a string that can be read from or written to a drive parameter.

See Also