LONMARK® Resource File API Reference Guide

Revision 5
May 2013

078-0261-01C

Printed in the United States of America.

Echelon Corporation
www.echelon.com

Table of Contents

Introduction

LONMARK resource files are files that define the components of the external interface for one or more devices that communicate using the ISO/IEC 14908-1 control networking protocol. These files allow installation tools and operator interface applications to interpret data produced by a device and to correctly format data sent to a device. They also help a system integrator or system operator to understand how to use a device and to control the functional blocks on a device. Standard resource files are available that define the standard components used in the device interface of a device. Device manufacturers must create user-defined resource files for any user-defined components defined within the device interface of a device.

The *LONMARK Resource File Developer's Guide* describes the different types of resource files, and describes procedures for creating resource files. This reference guide describes application programming interface (API) that can be used to access LONMARK resource files.

The LONMARK Interoperability Association provides the standard LONMARK resource files and API as an application for 64-bit and 32-bit Windows that installs the current version of the standard resources files and an API for Windows applications to read and write resource files.

The LONMARK Resource File API has three interfaces. One of the interfaces included with the standard resource files is a standard dynamic link library named **LCADRF32.DLL** to read and write LONMARK resource files. A second standard library named **LDRF32R.DLL** is also provided for read-only access to the LONMARK resource files. Both DLLs support a C-language API, which can also be accessed from many other languages. Literals and function prototypes are provided for C programmers with the **lcadrf.h** header file. Source code is provided for the read-only version. You can port the source code to other platforms to provide read-only access to resource files on any platform.

The second interface included with the LONMARK Resource File API is a COM component that provides a language-independent interface. The COM interface is described under *LONMARK Resource File API COM Interface*. The COM Interface is language independent and can be used from applications written in any programming language that supports COM interfaces. The interface is defined with a COM type library (TLB), but programmers may find the C-language **lcadrf.h** header file helpful for orientation.

The COM interface supports the same operations as the C-language interface with minor differences to the names of the entry points, and the translation of LONMARK resource file-specific data types into COM-compliant, general-purpose, types.

The LONMARK Resource File API supports *type definitions* in the wider sense of the word. These type definitions include enumerations, network variable types, and configuration property types, which are similar to a C-language *typedef*, but include much more information about the semantics of a type, restrictions, options, and even data values for minima, maxima, initialization and so forth. These definitions are also supported by descriptive texts, which can support multiple languages.

These types are stored in type files (.**TYP** extension) and language files (with various file extensions, subject to the supported language). Definitions of enumerations also have a C-language counterpart (a .**H** file with a C-language type definition).

The LONMARK Resource File API also supports functional profile definitions. Functional profiles group a number of network variables and configuration properties into a larger entity, which is then used to implement a functional block (also known as a *LONMARK object*). Like the types, functional profile definitions are enriched by descriptive texts, values and restrictions. The functional profiles are stored in functional profile template files (**.FPT** extension) and language files.

Another file supported by the LONMARK Resource file API is the format file (.**FMT** extension). Format files define rules for the presentation of data.

The type file, functional profile template file, format file, and language files together form a *resource file set*. All files in a resource file set share the same program ID and scope, and typically share the same base file name. For example, the standard resource file set includes **standard.typ**, **standard.fpt**, **standard.enu**, **standard.eng**, and **standard.fmt** files.

The resource file sets are organized in a resource catalog. The catalog is implemented in a file called **ldrf.cat**, which typically resides in the **types** folder of the LONWORKS directory. The catalog lists individual resource files (through their file path) as well as entire folders (which can include multiple resource sets). The catalog forms the central repository of resource file sets, and supplies search operations through the LONMARK Resource File API.

The following sections describe the LONMARK Resource File API functions. Each function description specifies whether it is available in each of the three interfaces available for the API.

All functions return an error code with a prefix of **LDRF_ERR_**. The zero error code, corresponding to the **LDRF_ERR_NONE** enumeration value, means there was no error. Most functions use or return a pointer to an **ldrfFileInfo** structure. This structure encapsulates the file header contents and internal control information.

The functions are organized in logical groups, and are documented in the remainder of this document in those groups. Those groups include functions related to the catalog, functions related to type files and functional profile template files, functions related to type tree operations, and basic utility functions. Each group is briefly introduced at the beginning of each section, and the API functions in the group are listed in alphabetical order.

Notes:

The **Get()** functions in the LONMARK Resource File API specify ouput parameters using pointers. In addition, you may specify a NULL pointer for any reference parameter that you do not require to be returned by a given **Get()** function.

LONMARK Resource File API indices are one-based. This means that an index value of 0 means *"unknown, not specified."*

What's New in this Release

This release includes support for version 6 type files and version 5 functional profile template files. Version 6 type files add support for new 64-bit signed and unsigned integer data types and expand the SNVT ID to support more than 250 SNVTs. To provide backward compatibility, support for these new data types and extended SNVT IDs is not enabled by

default in the version 2.4 LONMARK Resource File API. Instead, applications must inform the API that they can support these new types. A new flag bytes hase been added to network variable types, configuration property types, and functional profiles. The new flag bytes are reserved for future expansion—no new flags have been defined for the version 2.4 LONMARK Resource File API.

Installing the LONMARK Resource File API

The LONMARK Resource File API is used by a number of tools, including all LNS® and OpenLNS based tools and the LonMaker®Integration Tool. On Windows platforms, these tools share a single LONMARK Resource File API for Windows, which is installed by the LONMARK Standard Resource Files installer.

Source code is available for an API that provides read-only access to LONMARK resource files. You can download a .**zip** file containing this source code from the LONMARK Web site at *www.lonmark.org/technical_resources/resource_files*. After you download the .**zip** file, extract the source files in the archive to a working directory, and then port the files to your target platform.

Catalog Functions

The LONMARK resource file catalog provides the central repository for LONMARK resource files and file sets. Applications can open the catalog, search the catalog, and close the catalog. Functions to change the catalog's content are also provided. While individual type files can be accessed even when not registered with the catalog, the catalog's search functions only inspect registered resource files. Opening, searching and closing the catalog is the first set of API functions that are used for most LONMARK Resource File API client applications.

LdrfOpenCatalog

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfOpenCatalog(LPCSTR directory, TBool readOnly,
           PLdrfFileInfo *ppInfo)
```

COM Interface Prototype

```
LdrfCatalog.OpenCatalog(BSTR directory, long readOnly,
       long *ppInfo, long *returnCode)
```

Purpose

This function is called to open an existing resource file catalog or create a new resource file catalog. The catalog must be opened without the **readOnly** flag being set to TRUE if it needs to be created. The catalog is typically located in the **Types** folder within your local LONWORKS folder as is the standard resource file set (however, both catalogs do not need to

be stored in the same location). The folder name containing the **standard.typ** file is the **directory** input parameter.

An **ldrfFileInfo** structure pointer is returned if successful. The folder where the catalog resides is added to the list of directories in the catalog. When a new catalog is created, or a catalog marked stale is opened, it is then automatically refreshed if the **readOnly** flag is not set. The **readOnly** flag is intended for use by multiple simultaneous applications that need only read-access to the catalog, since only one application can be a writer to the file at a time, and that write access is granted exclusively to one application by the operating system, preventing *any* other applications from having *any* access.

To prevent locking out other applications from the catalog, close the catalog at the earliest opportunity (and re-open later, if necessary) because of the exclusive nature of write access to the catalog, See **LdrfCloseCatalog()** for more information.

If the refresh fails or does not occur, the catalog will remain stale. A catalog is marked stale when one or more entries in the list of folders is added or removed. The stale property does ***not*** reflect whether the current folder *contents* (the list of files) has changed since the last refresh.

Return Values

LDRF_ERR_NOT_FOUND	No folder of that name was found, or, if readOnly is set, then no catalog file was found.
LDRF_ERR_NO_ACCESS	Can't get access to open the file.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_CATALOG	The file header of the file was not correct for a resource file catalog.
LDRF_ERR_CRC	The file data did not pass the CRC check.

LdrfGetCatalogInfo

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetCatalogInfo(PLdrfFileInfo pInfo, PBool pStale,
              PUShort pNumDirec, PUShort pNumLangFiles,
              PUShort pNumTypeFiles, PUShort pNumFPTFiles,
              PUShort pNumFormatFiles)
```

COM Interface Prototype

```
LdrfCatalog.GetCatalogInfo(long pInfo, long *pStale, long *pNumDirec,
          long *pNumLangFiles, long *pNumTypeFiles,
          long *pNumFPTFiles, long *pNumFormatFiles)
```

Purpose

This function is used to retrieve the current catalog status information and statistics needed before listing the contents of the catalog. The number of known language files, type files,

functional profile files, and format files includes those detected in monitored folders (whose number is reported through the **pNumDirec** output parameter) and those registered explicitly.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.

LdrfCloseCatalog

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCloseCatalog(PLdrfFileInfo pInfo)
```

COM Interface Prototype

```
LdrfCatalog.CloseCatalog(long pInfo, long *returnCode)
```

Purpose

This function is called to close an open resource file catalog. The **ldrfFileInfo** structure for the open catalog is the only parameter. A stale catalog can be closed, and its stale state will be remembered.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfCatalogAddDir

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCatalogAddDir(PLdrfFileInfo pInfo, LPCSTR newDir)
```

COM Interface Prototype

```
LdrfCatalog.CatalogAddDir(long pInfo, BSTR newDir, long *returnCode)
```

Purpose

This function is called to add a directory to an existing, open resource file catalog. The **ldrfFileInfo** structure pointer for the open catalog is the first parameter, and the string containing the new directory name is the second parameter. Once this is done, the catalog is stale, and a refresh operation is required to bring it up to date before using it to search for a file. Opening a stale catalog is sufficient to bring it up to date, as the catalog is self-refreshing if marked stale.

When the catalog is being refreshed, the LONMARK Resource File API scans all registered folders and detects any resource files present, which then become available to catalog search operations. This process can slow down refreshing and opening the catalog when many large directories are registered. For faster process use **LdrfCatalogAddFile()** to explicitly register resource files instead.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	Don't have write access to the file.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfCatalogGetDir

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCatalogGetDir(PLdrfFileInfo pInfo, TUShort index,
             LPSTR pDirName, PUShort pLength)
```

COM Interface Prototype

```
LdrfCatalog.CatalogGetDir(long pInfo, long index, BSTR *pDirName,
         long *returnCode)
```

Purpose

This function is called to retrieve the name of the folder that corresponds to the given index into the folder list. This is useful for listing out the folders of the catalog. The folder names are not alphabetized. You can get the total number of registered folders using **LdrfGetCatalogInfo()**.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_TRUNC	Filename string was truncated to fit buffer.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfCatalogRmvDir

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCatalogRmvDir(PLdrfFileInfo pInfo, LPCSTR oldDir)
```

COM Interface Prototype

```
LdrfCatalog.CatalogRmvDir(long pInfo, BSTR oldDir, long *returnCode)
```

Purpose

This function is called to remove a folder from an existing, open resource file catalog. The **ldrfFileInfo** structure pointer for the open catalog is the first parameter, and the string containing the directory name to be removed is the second parameter. The side effect is that all information about files in that folder are also removed from the catalog. The catalog is not marked as stale as the result of a remove operation.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No folder of that name was found.
LDRF_ERR_NO_ACCESS	Don't have write access to the file.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfCatalogRefresh

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCatalogRefresh(PLdrfFileInfo pInfo)
```

COM Interface Prototype

```
LdrfCatalog.CatalogRefresh(long pInfo, long *returnCode)
```

Purpose

This function is called to refresh an existing, open resource file catalog. A catalog that was marked stale before the refresh will no longer be stale after successful completion of the refresh. All file information is refreshed during a refresh operation. Files that were added via an earlier operation are verified, and if they don't exist, they are removed from the catalog. All new resource files in the folders in the catalog are added. The **ldrfFileInfo** structure pointer for an open catalog is the only parameter.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	Don't have write access to the file.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfCatalogAddFile

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCatalogAddFile(PLdrfFileInfo pInfo, LPCSTR newFile)
```

COM Interface Prototype

```
LdrfCatalog.CatalogAddFile(long pInfo, BSTR newFile, long *returnCode)
```

Purpose

This function is called to add a single file to an existing, open resource file catalog. The catalog does not become stale. The **ldrfFileInfo** structure pointer and the file name (full pathname) are the parameters.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No file of that name was found.
LDRF_ERR_NO_ACCESS	Don't have write access to the file.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfCatalogGetFile

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCatalogGetFile(PLdrfFileInfo pInfo, TLdrfFileType fileType,
              TUShort index, PUByte pMatchMode,
              PUByteArray pProgID, PUShort pDirIndex,
              PUShort pMajorVersion, PUByte pMinorVersion,
              PULong pLocale,
              LPSTR pFileName, PUShort pLength)
```

COM Interface Prototype

```
LdrfCatalog.CatalogGetFile(long pInfo,
          TLdrfFileType fileType, TUShort index,
          PUByte pMatchMode, PUByteArray pProgID,
          PUShort pDirIndex, PUShort pMajorVersion,
          PUByte pMinorVersion, PULong pLocale,
          BSTR pFileName)
```

Purpose

This function is called to retrieve information about a file in the catalog, given the file type and an index. You can use this function for listing all or a subset of the files in the catalog.

The function returns all the information in the catalog regarding the particular file. The associated directory name can be retrieved by calling the **LdrfCatalogGetDir()** function using the directory index returned by this function. A file may not have an associated folder, if it was placed in the catalog explicitly via the **LdrfCatalogAddFile()** function, and in that case, the associated directory index is 0.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_TRUNC	Filename string was truncated to fit buffer.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfCatalogRmvFile

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCatalogRmvFile(PLdrfFileInfo pInfo, LPCSTR oldFile)
```

COM Interface Prototype

```
LdrfCatalog.CatalogRmvFile(long pInfo, BSTR oldFile, long *returnCode)
```

Purpose

This function is called to remove a single file from an existing, open resource file catalog. The catalog does not become stale. The **ldrfFileInfo** structure pointer and the file name (full pathname) are the parameters.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No file of that name was found.
LDRF_ERR_NO_ACCESS	Don't have write access to the file.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfSearchCatalog

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfSearchCatalog(PLdrfFileInfo pInfo, PUByteArray pProgID,
             TUByte matchMode, TLdrfFileType fileType,
             TULong locale, LPSTR pFile, PUShort pLength,
             PUShort pMajorVersion, PUByte pMinorVersion)
```

COM Interface Prototype

```
LdrfCatalog.SearchCatalog(long pInfo, BSTR progID,
         long matchMode, long fileType, long locale,
         BSTR *pFile,
         long *pMajorVersion, long *pMinorVersion
         long *returnCode)
```

Purpose

This function is called to retrieve a full pathname for a resource file given a program ID, a matching scope selector (0-6) for the **matchMode**, and the type of file to retrieve from the catalog. The file types are given by the **TLdrfFileType** enumeration. An **LDRF_WILD_CARD** value can also be specified for the file type, and all files will be returned (one file name per call, sequentially, until a **LDRF_ERR_NOT_FOUND** code is returned). In the case of language resource files, you must also specify the language locale code, unless the first of all such matching files is desired. Since matching selector 0 selects the standard file, it doesn't use the program ID, and the program ID pointer can be NULL if desired in that one case. A matching selector of a specific value, for example '3', will only match on a file which also has matching selector '3'. You can use a matching selector value of '0xFF' to find the first file matching a given program ID using the matching algorithm, that is, a file with '6' that matches, if one exists, else a file with '5', else '4', etc. A pointer to a buffer capable of holding the full pathname is passed in, along with the length of the buffer. Along with the filename, the major and minor content data version numbers are returned.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_FILE_TYPE	The file type requested isn't valid.
LDRF_ERR_NOT_FOUND	No file matching the request was found.
LDRF_ERR_TRUNC	Filename string was truncated to fit buffer.
LDRF_ERR_STALE	The catalog can't be searched, it needs a refresh.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfCatalogDependencyCode

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCatalogDependencyCode(PLdrfFileInfo pInfo,
          PUByteArray pProgID, PULong pDepCode)
```

COM Interface Prototype

```
LdrfGeneral2.CatalogDependencyCode(long pInfo,
                                   BSTR progID, long *pDepCode
                                   long *returnCode)
```

Purpose

This function calculates a dependency code that reflects the state of the subset of resource files in the catalog that match the specified program ID. This code does not vary as a result of a refresh, unless the list or content of the applicable resource files also changes. An application can use this to obtain an easy check on whether resource files, based on a program ID, have changed since the last time this function was called.

Return Values

LDRF_ERR_NOT_FOUND No file of that name was found.

LdrfMatchProgID

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfMatchProgID(TUByte matchMode,
           PUByteArray pFileRefID, PUByteArray pProgID)
```

COM Interface Prototype

```
LdrfCatalog.MatchProgID(long matchMode, BSTR fileRefID, BSTR progID,
       long *returnCode)
```

Purpose

This function is called to match two reference or program IDs using the scope selector value (1-6) given for the **matchMode** parameter. If called with scope selector 0, the file reference ID must be all zeros, and the program ID is not used.

Return Values

LDRF_ERR_NOT_FOUND No file of that name was found.

General File Functions

This section introduces functions that are used with all LONMARK resource files (but not the catalog itself). Exceptions are noted where necessary. The general file functions include opening and closing files, and reading and writing global information related to each resource file.

LdrfOpenFile

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfOpenFile(LPCSTR path, TLdrfFileType fileType,
        TUShort majorVersion, TBool checkCRC,
        PLdrfFileInfo *ppInfo)
```

COM Interface Prototype

```
LdrfGeneral.OpenFile(BSTR path, long fileType,
       long majorVersion, long checkCRC,
       long *ppInfo, long *returnCode)
```

Purpose

This function is called to open an existing file for access, supplying the pathname and optionally the minimum data major-version level needed. If no minimum data version level is needed, a 0 is used instead.

Although the function may be used to open a resource file not registered with the catalog, resource files that are not in the catalog are not searched when you search for a particular resource.

The **ppInfo** reference parameter is filled in if the operation was successful. Files below the minimum version are still opened successfully. The file CRCs are checked if requested through the **checkCRC** argument, but you can check them separately (see the **LdrfCheckHeaderCRC()** and **LdrfCheckDataCRC()** functions). If a file does not pass CRC check and a CRC check was requested, it is still opened, but the CRC error is returned. It's up to the caller to decide what to do then.

Return Values

LDRF_ERR_NOT_FOUND	No file of that name was found.
LDRF_ERR_FILE_TYPE	The file type requested isn't valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_RESOURCE	The file header of the file was not correct for a resource file (if it was a resource file that was requested).
LDRF_ERR_NOT_TYPE	The file header of the file was not correct for a TYP file (if it was a type file that was requested).
LDRF_ERR_NOT_FPT	The file header of the file was not correct for a FPT file (if it was an FPT file that was requested).
LDRF_ERR_FMT_VERSION	The major format version is not supported.
LDRF_ERR_VERSION	The data content major-version is lower than the minimum.
LDRF_ERR_CRC	The header or data CRC check did not pass.

LdrfEnableExtendedSNVTID

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfEnableExtendedSNVTID()
```

COM Interface Prototype

```
LdrfMiscFns2.EnableExtendedSNVTID()
```

Purpose

This function enables extended SNVT ID support. Applications that do not call this function are limited to 250 SNVT IDs. Applications that call this function can support up to 32766 SNVT IDs. SNVT ID 255 is reserved for identifying extended SNVT IDs.

This function must be called immediately following a successful **LdrfOpenFile()** or **LdrfEditFile()** call.

This function may be called for any valid **LdrfFileInfo** pointer, but only has effect for type, functional profile, and catalog files. Client tools must call **LdrfFindEmptyNVT()** to allocate an ID to a new resource. SNVT-ID 255 appears as an empty record, but **LdrfFindEmptyNV()** does not allocate it. UNVTs can be allocated in the 1…4095 UNVT ID range.

Return Values

LDRF_ERR_NOT FOUND	No file of that name was found.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record.

LdrfEditFile

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfEditFile(LPCSTR path, TLdrfFileType fileType,
        PLdrfFileInfo *ppInfo)
```

COM Interface Prototype

```
LdrfGeneral.EditFile(BSTR path, long fileType, long *ppInfo,
       long *returnCode)
```

Purpose

This function begins the creation of a new file of the type requested, or if the file already exists, opens the file for editing. The pathname is supplied.

If successful, a **ppInfo** reference parameter is filled in (or partially filled in, if a new file, see **LDRF_ERR_NEW** below). A file's CRCs will be checked before a successful open-for-edit. If a file does not pass data CRC check and header CRC check, it is not opened, and the CRC error is returned.

Return Values

LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_FILE_TYPE	The file type requested isn't valid.
LDRF_ERR_NOT_RESOURCE	The file header of the file was not correct for a resource file (if it was a resource file that was requested).
LDRF_ERR_NOT_TYPE	The file header of the file was not correct for a type file (if it was a type file that was requested).
LDRF_ERR_NOT_FPT	The file header of the file was not correct for a function profile (if it was a functional profile that was requested).
LDRF_ERR_FMT_VERSION	The major or minor format version is not supported.
LDRF_ERR_NEW	The file was created. Caller must then use the **LdrfSetFileHdrInfo** and **LdrfSetFileVersion** functions in either order prior to closing the file.
LDRF_ERR_CRC	The header or data CRC check did not pass.

LdrfGetFileHdrInfo

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFileHdrInfo(PLdrfFileInfo pInfo, PBool pUser,
              LPSTR pDesc, PUShort pDescLen,
              LPSTR pCreator, PUShort pCreLen,
              LPSTR pURL, PUShort pURLLen,
              PUByte pResDescSel, PULong pResDescIndex,
              PUByte pResCreSel,  PULong pResCreIndex);
```

COM Interface Prototype

```
LdrfGeneral.GetFileHdrInfo(long pInfo, long * pUser, BSTR * pDesc,
          BSTR * pCreator, BSTR * pURL,
          long * pResDescSel, long * pResDescIndex,
          long * pResCreSel,  long * pResCreIndex,
          long * returnCode)
```

Purpose

This function returns information strings and string resource references from an open file's header. This works for all resource file types (language files, type files, and functional profiles), as the types of information are identical. To open the file, use the appropriate open function from the appropriate file-specific sections below.

To retrieve the resource strings for extended creator information and descriptions, see the string resource functions.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_TRUNC	String or strings was/were truncated to fit buffer.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfSetFileHdrInfo

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfSetFileHdrInfo(PLdrfFileInfo pInfo, LPCSTR creator,
              LPCSTR phone, LPCSTR webid, LPCSTR URL,
              TUByte resDescSel, TULong resDescIndex,
              TUByte resCreSel,  TULong resCreIndex);
```

COM Interface Prototype

```
LdrfGeneral.SetFileHdrInfo(long pInfo,
          BSTR creator, BSTR phone, BSTR webid, BSTR URL,
          long resDescSel, long resDescIndex,
          long resCreSel,  long resCreIndex,
          long * returnCode)
```

Purpose

This function creates or modifies information strings and resource references in an open file's header. This works for all resource file types, as the information is identical. To open the file, use **LdrfEditFile()**.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfGetFileVersion

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFileVersion(PLdrfFileInfo pInfo,
              PUByte  pMajorFmtVer,  PUByte pMinorFmtVer,
              PUShort pMajorDataVer, PUByte pMinorDataVer,
              PUShort pYear, PUByte pMonth, PUByte pDay,
              PUByte pHour, PUByte pMinute, PUByte pSecond,
              PUByte pSel, PUByteArray *ppRefID);
```

COM Interface Prototype

```
LdrfGeneral.GetFileVersion(long pInfo,
          long *pMajorFmtVer, long *pMinorFmtVer,
          long *pMajorDataVer, long *pMinorDataVer,
          long *pYear, long *pMonth, long *pDay,
          long *pHour, long *pMinute, long *pSecond,
          long *pSel, BSTR *pRefID, long *returnCode)
```

Purpose

This function returns version information and timestamp data from an open file's header. The reference ID and scope values are also returned. This works for all resource file types (language files, type files, and functional profiles), as the types of information are identical. To open the file, use **LdrfOpenFile()** or **LdrfEditFile()**, as appropriate. For the C language API, the reference ID is allocated as a byte array (of 8 bytes) and is filled in. Your application must call the **LdrfFreeByteArray()** function to free the returned byte arraywhen your application is done with it. For the COM API, the reference ID is allocated as a BSTR.

Return Values

LDRF_ERR_FILE_INFO	The file info structure contents was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfSetFileVersion

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfSetFileVersion(PLdrfFileInfo pInfo,
              TUShort majorDataVer, TUByte minorDataVer,
              TUByte sel, PUByteArray pRefID);
```

COM Interface Prototype

```
LdrfGeneral.SetFileVersion(long pInfo,
          long majorDataVer, long minorDataVer,
          long sel, BSTR refID, long *returnCode)
```

Purpose

This function creates or modifies version information and reference information in an open file's header. This works for all resource file types (language files, type files, and functional profiles), as the types of information are identical. This function cannot be used to change the version number of the standard resource file set. To open the file, use **LdrfEditFile()**. For the C language API, the **pRefID** parameter must point to a byte array that contains eight bytes. For the COM API, the reference ID is a BSTR.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_PARAM	A parameter isn't valid. This can be returned if the refID format is invalid or if this function is called with the standard resource file.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfCloseFile

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCloseFile(PLdrfFileInfo pInfo)
```

COM Interface Prototype

```
LdrfGeneral.CloseFile(long pInfo, long *returnCode)
```

Purpose

This function closes a previously opened resource file. If the file was being edited or created, new header information is built (including the directory) and written to the file, the file is packed if previous editing actions created gaps, and the header and data CRCs are updated. A valid info pointer must be passed in. All memory related to the file info structure will be freed, so the info pointer must not be used again after calling this function.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	The file has not been completely created.
LDRF_ERR_WRITE	Write error, or disk is full.

LdrfGetLangFileInfo

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetLangFileInfo(PLdrfFileInfo pInfo,
               PULong pLocale, PULong pNumResources)
```

COM Interface Prototype

```
LdrfLangResource.GetLangFileInfo(long pInfo,
           long *pLocale, long *pNumResources,
           long *returnCode)
```

Purpose

This function returns the locale code and the number of resources in the open language (string) resource file.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.

LdrfExtendedDataTypeAware

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfExtendedDataTypeAware(PLdrfFileInfo pInfo,
               TNVTType dataType)
```

COM Interface Prototype

```
LdrfMiscFns1.ExtendedDataTypeAware(long pInfo,
           long dataType, long *returnCode)
```

Purpose

This function enables access to the following extended data types: **NVT_TYPE_UNSIGNED_QUAD**, **NVT_TYPE_DOUBLE_FLOAT**, **NVT_TYPE_SIGNED_INT64,** and **NVT_TYPE_UNSIGNED_INT64**. The client must specify an open resource file and must also specify the most recent data type supported. Extended data types are defined in a chronologically sorted list according to their time of introduction. Following is the list, from oldest to newest:

1. **NVT_TYPE_UNSIGNED_QUAD**
2. **NVT_TYPE_DOUBLE_FLOAT**
3. **NVT_TYPE_SIGNED_INT64**
4. **NVT_TYPE_UNSIGNED_INT64**

To specify support for all four types, specify **NVT_TYPE_UNSIGNED_INT64** for the dataType parameter.

Without calling this function, these extended types are presented in a backwards compatible fashion as described in the following table.

Extended Data Type	Backward-Compatible Presentation
NVT_TYPE_DOUBLE_FLOAT	For non-structured types: a structure containing an array of 8 unsigned short integers in big-endian order For structured types: a **data[]** array member of the existing structure with 8 unsigned short integers in big-endian order
NVT_TYPE_SIGNED_INT64	For non-structured types: a structure containing an array of 8 unsigned short integers in big-endian order For structured types: a **drf_int64[]** array member of the existing structure with 8 unsigned short integers in big-endian order
NVT_TYPE_UNSIGNED_INT64	For non-structured types: a structure containing an array of 8 unsigned short integers in big-endian order For structured types: a **drf_uint64[]** array member of the structure with 8 unsigned short integers in big-endian order
NVT_TYPE_UNSIGNED_QUAD	**NVT_TYPE_SIGNED_QUAD**

To enable extended types for a resource file set, you must call this function for each type file in the resource file set, each time you open the resource file set.

Return Values

LDRF_ERR_FILE_INFO The file info structure content was not valid.

LdrfConvertFile

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfConvertFile(LPCSTR pathIn, LPCSTR pathOut,
                TUByte toVersion, TBool checkCRC)
```

COM Interface Prototype

```
LdrfGeneral2.ConvertFile(BSTR pathIn, BSTR pathOut,
                         long toVersion, long checkCRC,
                         long *returnCode)
```

Purpose

This function converts a resource file (type, functional profile, or language file) to the specified version. If **toVersion** is set to 0, the resource file will be converted to the latest available version. This function cannot convert version 1 type files, which are not supported by the LONMARK Resource File API.

When converting a resource file from one format version A to a more recent format version B, data added in format B may be set to defaults (typically all zeroes). Conversely, if you downgrade a resource file from format version B to an earlier format version A, the resource file will lose data in the conversion process. This data will be lost unrecoverably.

Return Values

LDRF_ERR_CRC	The data did not pass the CRC check.

LdrfPurgeFile

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfPurgeFile(LPCSTR pathIn, LPCSTR pathOut,
               TBool checkCRC)
```

COM Interface Prototype

```
LdrfGeneral2.PurgeFile(BSTR pathIn, BSTR pathOut,
                       long checkCRC, long *returnCode)
```

Purpose

Resources can be flagged as deleted without actually removing them from the resource file set. This is done by marking the resource with a name that ends with a tilde '~' character. The advantage of this deletion method is that it can be undone, and that other types, which might reference the type marked for deletion, may not break as a result.

However, it is sometimes useful to purge a resource file from all resources marked deleted in this way. This removes unnecessary ballast that might have accumulated during development, frees up space, and frees up previously used indices. You can use the **LdrfPurgeFile()** function to purge resource files.

Return Values

LDRF_ERR_CRC	The data did not pass the CRC check.

LdrfEditFileVer

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfEditFileVer(LPCSTR path, TLdrfFileType fileType,
                long majFmtForCreate, PLdrfFileInfo * ppInfo)
```

COM Interface Prototype

```
LdrfGeneral2.EditFileVer(BSTR path, long fileType,
                long majFmtForCreate, long *ppInfo, long *returnCode)
```

Purpose

This function creates a resource file with the version specified by the **majFmtForCreate** parameter. The parameter is only used if the file does not already exist (in other words, an existing file won't be converted as a result of this call). If an unsupported major-format-version is requested and the file does not already exist, an **LDRF_ERR_PARAM** error is returned. If a zero is passed for this parameter, the function uses the latest format version.

Return Values

LDRF_ERR_CRC	The data did not pass the CRC check.
LDRF_ERR_PARAM	The requested major format version is not supported.

LdrfEnableEmptyEntries

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfEnableEmptyEntries(PLdrfFileInfo pInfo)
```

COM Interface Prototype

```
LdrfMiscFns1.EnableEmptyEntries(long pInfo, long *returnCode)
```

Purpose

Version 4 type files, version 3 language files, and version 3 FPT files support resource deletion, which can leave gaps in the structure of a resource file. This function informs the LONMARK Resource File API that the application supports empty type records. An empty-aware application will receive the **LDRF_ERR_EMPTY_RECORD** return code if it attempts to get an empty record. An application that is not aware of empty entries will see placeholder entries. The programmatic name for an empty record will include the text "**empty**<*index*>~", where <*index*> indicates the decimal character representation of the index number of the empty record, and the '~' character is the last character of the name (or record).

Gaps are presented as if they hold a resource flagged for deletion (for example, using a name ending with a tilde '~' character), even if the file has been successfully purged with **LdrfPurgeFile()**.

Return Values

LDRF_ERR_CRC	The data did not pass the CRC check.

Language File Functions

Language files hold strings with alphanumerical information such as descriptions, units, and names. Many aspects of LONMARK resource files, and the items defined therein, can refer to these strings through their reference ID and scope value pair and the string index.

You can use the language file functions to define, manage, and obtain localized description strings and similar textual information. See the next section, *String Service Functions*, for more information.

LdrfSetLangFileInfo

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetLangFileInfo(PLdrfFileInfo pInfo, TULong locale)
```

COM Interface Prototype

```
LdrfLangResource.SetLangFileInfo(long pInfo,
           long locale, long *returnCode)
```

Purpose

This function sets the locale code for the open language (string) resource file.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.

LdrfGetResourceString

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetResourceString(PLdrfFileInfo pInfo, TULong index,
                LPSTR pString, PUShort pLength)
```

COM Interface Prototype

```
LdrfLangResource.GetResourceString(long pInfo,
             long index, BSTR *pString,
             long *returnCode)
```

Purpose

This function retrieves a string from an open resource file. The **pInfo** parameter is supplied, along with the index of the resource string in **index**. The string pointer and length pointer parameters are supplied by the caller. The length must be set to the maximum size of the buffer available prior to the call. If the resource string exceeds the length available, it will be truncated (see error code below).

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No resource with that index was found.
LDRF_ERR_TRUNC	Resource string was truncated to fit buffer.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfDeleteResourceString

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfDeleteResourceString(PLdrfFileInfo pInfo, TULong index)
```

COM Interface Prototype

```
LdrfMiscFns1.DeleteResourceString(long pInfo, long index,
                                  long *returnCode)
```

Purpose

This function is called to delete a resource string. Deleted resources do not consume any file data space. They only have NULL entries in the resource key-access directories.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No resource with that index was found.
LDRF_ERR_TRUNC	Resource string was truncated to fit buffer.
LDRF_ERR_FMT_VERSION	The function was called on a pre-version 3 language-dependent-string resource file.

LdrfSetASCIIResource

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetASCIIResource(PLdrfFileInfo pInfo, TBool shareDup,
                TULong index, LPCSTR string,
                PULong pDupIndex)
```

COM Interface Prototype

```
LdrfLangResource.SetASCIIResource(long pInfo,
            long shareDup, long index,
            BSTR string, long *pDupIndex,
            long *returnCode)
```

Purpose

This function modifies or adds an ASCII resource string, provided the language file has been opened for editing. An existing resource string will be replaced, but it must already be an ASCII resource string.

A new ASCII resource string can be added, but only if the index is one larger than the existing number of resource strings. The index of the string is passed in, as well as the **pInfo** parameter and the string pointer. Modified strings may be added at the end of the file, and a gap in the middle of the file may result. Gaps will be remembered until the file is closed, at which time it will be packed if necessary, and directories will be rebuilt. The interface can be asked to share duplicates, or not. If the **shareDup** parameter is TRUE, the creation of a new string which duplicates one already in the file will result in the string not being created, the **pDupIndex** parameter will be used to return the index of the string duplicated, and the error code will be **LDRF_ERR_DUPLICATE**. If the **shareDup** parameter is FALSE, the **pDupIndex** parameter is set to the value of the input index that is used to set the resource string. The **index** parameter and the **pDupIndex** parameter may be the value and reference, respectively, of the same variable.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_NOT_FOUND	No resource with that index was found, or if new, was not correct.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_DUPLICATE	The string is already in the file.
LDRF_ERR_FULL	File is full, no more indices can be added.

LdrfFindEmptyResourceString

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfFindEmptyResourceString(PLdrfFileInfo pInfo, PULong pIndex)
```

COM Interface Prototype

```
LdrfMiscFns1.FindEmptyResourceString(long pInfo, long *pIndex,
                                     long *returnCode)
```

Purpose

This function returns the first empty resource string index. If there are no empty resource string records, this function returns n+1, where n is the number of resource string records in the file.

Return Values

LDRF_ERR_INTERNAL	Internal error.
LDRF_ERR_NOT_FOUND	No empty record index is available (only occurs if file is full).

LdrfValidateResourceString

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfValidateResourceString(PLdrfFileInfo pInfo, TUShort Index)
```

COM Interface Prototype

```
LdrfMiscFns1.ValidateResourceString(long pInfo, long index,
                                    long *returnCode)
```

Purpose

This returns a value that indicates the status of the specified resource string. See *Return Values* for more information.

Return Values

LDRF_ERR_PARAM	Incorrect parameters supplied.
LDRF_ERR_NOT_FOUND	The specified resource string was not found.
LDRF_ERR_INTERNAL	Internal error.
LDRF_ERR_NONE	The resource string was found and is not empty.
LDRF_ERR_EMPTY_RECORD	The resource string is an empty record (i.e. it was deleted). This error code will only be returned if the **LdrfEnableEmptyEntries()** function was called on the type file.

LdrfGetNumLanguages

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetNumLanguages(LPCSTR pInfDirectory, PUShort pNumLanguages)
```

COM Interface Prototype

```
LdrfLangResource.GetNumLanguages(BSTR pInfDirectory,
                        long *pNumLanguages, long *returnCode)
```

Purpose

This function returns the number of languages currently known to the LONMARK Resource File API. These languages can be accessed with consecutive keys from 1 to *n*, where *n* is the number of languages returned by this function. Specifying the directory containing the **lcadrf32.inf** file is optional; a registry key from **lcadrf32.inf**is normally used to find the .INF file, so the value NULL is normally used for this parameter.

Return Values

LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INTERNAL	Internal error, algorithm / unexpected error.
LDRF_ERR_PARAM	A parameter isn't valid (the **pInfDirectory** parameter doesn't contain the .INF file, or **pNumLanguages** isn't valid).
LDRF_ERR_NOT_FOUND	No language information was found.
LDRF_ERR_TRUNC	Directory/file pathname too long for buffer.

LdrfGetLanguageInfo

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetLanguageInfo(LPCSTR pInfDirectory, TULong myLocale, TUShort key,
LPCSTR pCatDirectory, PULong pMSLocaleID, PULong pLocale, PUByteArray
pFileExtension, LPSTR pString, PUShort pLength)
```

COM Interface Prototype

```
LdrfLangResource.GetLanguageInfo(BSTR *pInfDirectory,
       long myLocale, long key, BSTR *pCatDirectory, long *pMSLocaleID,
       long *pLocale, BSTR *pFileExtension, BSTR *pString,
       long *returnCode)
```

Purpose

This function returns all the language file information that corresponds to a specified key. This includes the 16-bit locale ID, the 32-bit locale value that can be used with the Catalog API, the three-letter file extension, and a locale-specific string (in the language of choice) that can be printed or displayed. Specifying the directory containing the file **lcadrf32.inf**file is optional; a registry key from **lcadrf32.inf**is normally used to find the .INF file, so the value NULL is normally used for this parameter. For a given key, this function can return the locale, the extension, or a string naming the language, or any combination of these items. Should any of the return item pointers be NULL, that item will be skipped. To return a string naming the language, the string is returned in the language requested (if possible). The string's language is requested via the **myLocale** parameter. If the **pString** parameter (the string naming the language) is NULL, then the **myLocale**, **pCatDirectory**, and **pLength** parameters are not used, and can be any value. Otherwise, the **pString** parameter must point to a buffer to receive the string, and the **pLength** parameter must point to a **TUShort** variable containing the maximum length of the buffer. The **pLength** parameter is only used with the C language API. The length variable will be modified to contain the actual number of characters in the string upon return. The LONMARK Resoure File catalog is needed to retrieve the string naming the language. The **pCatDirectory** parameter can be NULL, however; unless it is desired to override the registry key that locates the **lonworks\types** directory as the home of the **ldrf.cat** catalog file.

Return Values

LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_PARAM	A parameter isn't valid (key not in valid range).
LDRF_ERR_NOT_FOUND	No language information was found for the key.
LDRF_ERR_TRUNC	Resource string was truncated to fit the buffer, or a filename/directory name was too long.
LDRF_ERR_NO_ACCESS	Can't get access to open a file.
LDRF_ERR_NOT_CATALOG	The file header of the file was not correct for a resource file catalog.
LDRF_ERR_CRC	The file data did not pass the CRC check.
LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_FILE_TYPE	The file type requested isn't valid.
LDRF_ERR_STALE	The catalog can't be searched, it needs a refresh.
LDRF_ERR_NOT_RESOURCE	The file header of the file was not correct for a resource file (if it was a resource file that was requested).
LDRF_ERR_FMT_VERSION	The major format version is not supported.
LDRF_ERR_VERSION	The data content major-version is lower than the minimum.

LdrfGetLanguageKeyFromLocale

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetLanguageKeyFromLocale(LPCSTR pInfDirectory, TULong locale,
                             PUShort pKey)
```

COM Interface Prototype

```
LdrfLangResource.GetLanguageKeyFromLocale(BSTR *pInfDirectory,
                        long locale, long *pKey, long *returnCode)
```

Purpose

This function returns the key corresponding to the provided locale. Once the key is obtained, any other language information can also be obtained using the **LdrfGetLanguageInfo()** function. Specifying the directory containing the **lcadrf32.inf**file is optional; a registry key from **lcadrf32.inf** is normally used to find the .INF file, so the value NULL is normally used for this parameter.

Return Values

LDRF_ERR_INTERNAL	Internal error, algorithm / unexpected error.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_PARAM	A parameter isn't valid (the **pInfDirectory** parameter isn't valid, or the key is NULL).
LDRF_ERR_NOT_FOUND	No .INF file or no language information was found.
LDRF_ERR_TRUNC	File pathname was too long to fit in buffer.

LdrfGetLanguageKeyFromMSLocaleID

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetLanguageKeyFromMSLocaleID(LPCSTR pInfDirectory,
                  TULong MSLocaleID, PUShort pKey)
```

COM Interface Prototype

```
LdrfLangResource.GetLanguageKeyFromMSLocaleID(BSTR *pInfDirectory,
                        long MSLocaleID, long *pKey, long *returnCode)
```

Purpose

This returns the key corresponding to the provided Locale ID, for example the Locale ID for US English is 0x0409. Once the key is obtained, any other language information can also be obtained using the **LdrfGetLanguageInfo()** function. Specifying the directory containing

the file **lcadrf32.inf** file is optional; a registry key from **lcadrf32.inf** is normally used to find the .INF file, so the value NULL is normally used for this parameter.

Return Values

LDRF_ERR_INTERNAL	Internal error, algorithm / unexpected error.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_PARAM	A parameter isn't valid (the **pInfDirectory** parameter isn't valid, or the key is NULL).
LDRF_ERR_NOT_FOUND	No .INF file or no language information was found.
LDRF_ERR_TRUNC	File pathname was too long to fit in buffer.

LdrfGetLanguageKeyFromExtension

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetLanguageKeyFromExtension(LPCSTR pInfDirectory,
                  TUByteArray fileExtension, PUShort pKey)
```

COM Interface Prototype

```
LdrfLangResource.GetLanguageKeyFromExtension(BSTR *pInfDirectory,
                  BSTR fileExtension, long *pKey, long *returnCode)
```

Purpose

This function returns the key corresponding to the provided three-character language extension. Once the key is obtained, any other language information can also be obtained using the **LdrfGetLanguageInfo()** function. Specifying the directory containing the **lcadrf32.inf** file is optional; a registry key from **lcadrf32.inf** is normally used to find the .INF file, so the value NULL is normally used for this parameter.

Return Values

LDRF_ERR_INTERNAL	Internal error, algorithm / unexpected error.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_PARAM	A parameter isn't valid (the **pInfDirectory** parameter isn't valid, or the key is NULL).
LDRF_ERR_NOT_FOUND	No .INF file or no language information was found.
LDRF_ERR_TRUNC	File pathname was too long to fit in buffer.

String service functions provide a simple API to retrieve resource strings. Resource strings are always referenced through their scope and index value pairs, combined with the reference ID and scope of the file that contains the reference. You can use string service functions to locate the referenced string in the current locale, and you can use the string functions' prioritized list of locales for automatic substitutions if the first choice of languages is unavailable.

For example, an application can register French as the first choice language and Spanish as the second choice. English (US English) is always automatically used as the lowest priority choice, and does not require explicit registration. The string service API can then retrieve a given string, referenced by its index and scope value pair, in French, if available. Spanish language will be supplied as the second choice, and so forth.

LdrfStartStringService

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfStartStringService(TUlong locale, LPCSTR directory,
                           PserviceID, pID)
```

COM Interface Prototype

```
Ldrf LdrfLangResource.StartStringService(long locale, BSTR directory,
                                  long* pID, long *returnCode)
```

Purpose

This function begins a string service session. During a string service session you can request language strings in the language provided in the **locale** parameter. This function returns a service ID that is used to identify the string service in other functions.

Return Values

LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_TRUNC	Directory/filename was truncated to fit buffer.
LDRF_ERR_STALE	The catalog can't be searched, it needs a refresh.
LDRF_ERR_PARAM	A parameter isn't valid.
LDRF_ERR_STRING_SERVICE	The service ID is not a valid service (internal error in this function, since it allocates the service ID).

LdrfAddStringServiceLocale

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfAddStringServiceLocale(TServiceID id, TULong locale)
```

COM Interface Prototype

```
LdrfLangResource.AddStringServiceLocale(long id, long locale,
                                        long *returnCode)
```

Purpose

This function adds a locale to the list of locales that will be searched by the **LdrfStringServiceRequest()** function. When this function is called, any string currently in the cache is cleared.

Return Values

LDRF_ERR_PARAM	A parameter isn't valid (the id is NULL).
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_STRING_SERVICE	The service ID provided is not a valid service.

LdrfStringServiceRequest

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfStringServiceRequest(TServiceID id, PUByteArray pProgID,
            TUByte scope, TULong index, LPSTR string, PUShort pLength)
```

COM Interface Prototype

```
LdrfLangResource.StringServiceRequest(long id, BSTR *pProgID,
      long scope, long index, BSTR string, long *returnCode)
```

Purpose

This function returns the requesting string from the locale provided in the **LdrfStartStringService()** function. If multiple locales have been provided via the **LdrfAddStringServiceLocale()** function and the string is not found in the first language, it will be searched for in the second, the third, etc. If the string is not found in any of the listed locales, the string service will return "<*scope*:*index*> **Message string is unavailable**".

Return Values

LDRF_ERR_PARAM	A parameter isn't valid (e.g., the id is NULL).
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_FILE_TYPE	The file type requested isn't valid.
LDRF_ERR_STALE	The catalog can't be searched, it needs a refresh.
LDRF_ERR_NOT_RESOURCE	The file header of the file was not correct for a resource file (if it was a resource file that was requested).
LDRF_ERR_FMT_VERSION	The major format version is not supported.
LDRF_ERR_VERSION	The data content major-version is lower than the minimum.
LDRF_ERR_CRC	The header or data CRC check did not pass.
LDRF_ERR_NOT_FOUND	No resource with that index was found.
LDRF_ERR_TRUNC	Resource string was truncated to fit buffer.
LDRF_ERR_INCOMPLETE	The file has not been completely created.
LDRF_ERR_WRITE	Write error, or disk is full.

LdrfStopStringService

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfStopStringService(TServiceID id)
```

COM Interface Prototype

```
LdrfLangResource.StopStringService(long id, long *returnCode)
```

Purpose

This function terminates the string service and closes all cached files.

Return Values

LDRF_ERR_PARAM	A parameter isn't valid (the id is NULL).
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_STRING_SERVICE	The service ID provided is not a valid service.
LDRF_ERR_FILE_INFO	The file info structure content was not valid.

Type File Functions

Type files contain type information for network variable types (NVT), configuration property types (CPT), and enumerations. You can use the type file functions for general operations with type files and allow to manage the resource types. See the *Type Tree Functions* in the next section for access to the fundamental data type definitions behind NVT or CPT.

LdrfGetTypeFileInfo

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetTypeFileInfo(PLdrfFileInfo pInfo,
                PUShort pResDep0, PUShort pResDep1,
                PUShort pResDep2, PUShort pResDep3,
                PUShort pResDep4, PUShort pResDep5,
                PUShort pResDep6,
                PUShort pTypDep0, PUShort pTypDep1,
                PUShort pTypDep2, PUShort pTypDep3,
                PUShort pTypDep4, PUShort pTypDep5,
                PUShort pTypdep6,
                PUShort pNumNVTs, PUShort pNumCPTs,
                PUShort pNumEnumSets)
```

COM Interface Prototype

```
LdrfTypes.GetTypeFileInfo(long pInfo,
          long *pResDep0, long *pResDep1, long *pResDep2,
          long *pResDep3, long *pResDep4, long *pResDep5,
          long *pResDep6,
          long *pTypDep0, long *pTypDep1, long *pTypDep2,
          long *pTypDep3, long *pTypDep4, long *pTypDep5,
          long *pTypDep6, long *pNumNVTs, long *pNumCPTs,
          long *pNumEnumSets, long *returnCode)
```

Purpose

This function is called retrieves information specific to the open type file. The **pInfo** argument specifies the open type file. All other arguments to this function are pointers to output parameters, and a NULL value may be used if the caller is not interested in a particular detail.

The function reports the number of NVT, CPT, and enumerations defined in the type file. The function also supports two seven-part dependency codes. The dependency codes are not typically used; callers to this API typically specify the NULL pointer for these arguments.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	The file has not been completely created.

LdrfSetTypeFileInfo

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetTypeFileInfo(PLdrfFileInfo pInfo,
               TUShort resDep0, TUShort resDep1,
               TUShort resDep2, TUShort resDep3,
               TUShort resDep4, TUShort resDep5,
               TUShort resDep6,
               TUShort typDep0, TUShort typDep1,
               TUShort typDep2, TUShort typDep3,
               TUShort typDep4, TUShort typDep5,
               TUShort typdep6)
```

COM Interface Prototype

```
LdrfTypes.SetTypeFileInfo(long pInfo, long resDep0, long resDep1,
           long resDep2, long resDep3, long resDep4,
           long resDep5, long resDep6,
           long typDep0, long typDep1, long typDep2,
           long typDep3, long typDep4, long typDep5,
           long typDep6, long *returnCode)
```

Purpose

This function sets dependency information specific to the open type file. The **pInfo** parameter is supplied, along with the dependency data.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

Enum Set Access Functions for a Type File

Enumerations are signed 8-bit enumerated value sets with a value range of -128…+127. Each enumeration defines a tag and a header file name a set of members. Each member defines a name/value pair. Optional resource strings may be supplied for additional information.

When working with an enumeration, you must select the enumeration with the **LdrfSelectEnumSet()** function first. You can then access the enumeration and its members until you select a different enumeration.

LdrfChangeSelectedEnumSetFile

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfChangeSelectedEnumSetFile (PLdrfFileInfo pInfo,
                               LPSTR file);
```

COM Interface Prototype

```
LdrfGeneral2.ChangeSelectedEnumSetFile (long pInfo,
                                        BSTR file,
                                        long *returnCode );
```

Purpose

This function updates the selected enumeration's file name string and database key entry. This file name relates to the name of the C language header file which provides the C-language enumeration for this type. The enumeration set will remain selected.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No enumeration set with that index was found.
LDRF_ERR_DUPLICATE	An attempt was made to change the file name string to a duplicate of another enum set. The DRF API will attempt to restore the file name database key.

LdrfChangeSelectedEnumSetTag

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfChangeSelectedEnumSetTag (PLdrfFileInfo pInfo,
                              LPSTR tag);
```

COM Interface Prototype

```
LdrfGeneral2.ChangeSelectedEnumSetTag (long pInfo,
                                       BSTR tag,
                                       long *returnCode );
```

Purpose

This function updates the selected enumeration's tag string and database key entry. The enumeration set will remain selected.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No enum set with that index was found.
LDRF_ERR_DUPLICATE	An attempt was made to change the tag string to a duplicate of another enum set. The LONMARK Resource File API will attempt to restore the tag database key.

LdrfDeleteEnumMemberByIndex

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfTypes.DeleteEnumMemberByIndex (PLdrfFileInfo pInfo,
                                   TUShort index);
```

COM Interface Prototype

```
LdrfGeneral2.DeleteEnumMemberByIndex (long pInfo,
                                      long index,
                                      long *returnCode );
```

Purpose

This function deletes the indexed member of the selected enumeration set. The enumeration set will remain selected. All members with larger indices than the member being deleted will have their indices adjusted downwards (i.e., decremented) by 1. The index is 1-based (meaning an enumeration set with one single member will have a valid index of '1', and only that value).

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No enum set with that index was found.

LdrfSelectEnumSet

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfSelectEnumSet(PLdrfFileInfo pInfo, TUShort index,
            LPSTR pTag, PUShort pTagLen,
            LPSTR pFile, PUShort pFileLen)
```

COM Interface Prototype

```
LdrfMiscFns1.SelectEnumSet(long pInfo, long index, BSTR *pTag,
       BSTR *pFile, long *returnCode)
```

Purpose

This selects an enumeration set in an open type file. The **pInfo** parameter is supplied, along with the index of the enumeration set. This function does not return any enumeration data; it just selects which enumeration set to use in certain future accesses through other functions, and this selection is retained in the **info** structure. The function does return the enumeration tag and enumeration file name if desired.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No enumeration set with that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfSelectEnumSetByTag

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfSelectEnumSetByTag(PLdrfFileInfo pInfo, LPCSTR tag,
                  PUShort pIndex)
```

COM Interface Prototype

```
LdrfTypes.SelectEnumSetByTag(long pInfo, BSTR tag, long *pIndex,
              long *returnCode)
```

Purpose

This function selects an enumeration in an open type file. The **pInfo** parameter is supplied, along with a string pointer to the tag of the enumeration. This function does not return any enumeration data, it just selects which enumeration to use in certain future accesses (see below), and this selection is retained in the info structure. The index of the enumeration is returned if desired.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No enumeration with that index was found.

LdrfSelectEnumSetByFile

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfSelectEnumSetByFile(PLdrfFileInfo pInfo, LPCSTR file,
                  PUShort pIndex)
```

COM Interface Prototype

```
LdrfTypes.SelectEnumSetByFile(long pInfo, BSTR file, long *pIndex,
               long *returnCode)
```

Purpose

This function selects an enumeration in an open type file. The **pInfo** parameter is supplied, along with a string pointer to the filename key of the enumeration. This function does not return any enumeration data, it just selects which enumeration to use in certain future accesses (see below), and this selection is retained in the info structure. The index of the enumeration is returned if desired.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No enumeration with that index was found.

LdrfSelectNewEnumSet

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSelectNewEnumSet(PLdrfFileInfo pInfo,
                LPCSTR tag, LPCSTR file, PUShort pIndex)
```

COM Interface Prototype

```
LdrfTypes.SelectNewEnumSet(long pInfo, BSTR tag, BSTR file,
            long *pIndex, long *returnCode)
```

Purpose

This function creates and selects a new enumeration in an open type file. The **pInfo** parameter is supplied, along with a string pointer to the enumeration tag and a string pointer to the filename key of the enumeration. This call creates the new enumeration and selects it for future operations, and this selection is retained in the info structure. The new enumeration is created using the next available index. That index is returned in the **pIndex** reference parameter. Both the tag key and the filename key must be unique in the file. The new enumeration is initially created with no members.

If an enumeration set is deleted, leaving an empty record, this function will search for an available empty enumeration set record before creating a new record at the end of the file.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_DUPLICATE	One of the keys is already in the file.
LDRF_ERR_FULL	File is full, no more indices can be added (if editing).

LdrfDeleteEnumSet

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfDeleteEnumSet(PLdrfFileInfo pInfo, TUShort Index)
```

COM Interface Prototype

```
LdrfTypes.DeleteEnumSet(long pInfo, long index, long *returnCode)
```

Purpose

This function deletes an enumeration set. Deleted enumerations do not consume any file data space. They only have NULL entries in the resource key-access directories.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_FMT_VERSION	The function was called on a pre-version 4 type file.

LdrfGetEnumMember

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetEnumMember(PLdrfFileInfo pInfo, TByte value,
             LPSTR pString, PUShort pLength,
             PUByte pResSel, PULong pResIndex)
```

COM Interface Prototype

```
LdrfTypes.GetEnumMember(long pInfo, long value,
         BSTR *pString, long *pResSel, long *pResIndex,
         long *returnCode)
```

Purpose

This function retrieves an enumeration member from the previously selected enumeration in an open type file. The **pInfo** parameter for the file is supplied (for the C language API only), along with the key for the member of the enumeration. The member key is identical to the value. Since the enumeration members each have a programmatic string and a resource string index, both are returned. The caller must allocate a buffer to hold the string, and pass the length of the buffer through the length reference parameter (for the C language API only), which will be altered to indicate the length actually read.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_SELECTED	No selected enumeration.
LDRF_ERR_NOT_FOUND	No enumeration member with that value was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.

LdrfGetEnumValue

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetEnumValue(PLdrfFileInfo pInfo, LPCSTR string,
            PByte pValue)
```

COM Interface Prototype

```
LdrfTypes.GetEnumValue(long pInfo, BSTR string,
        long *pValue, long *returnCode)
```

Purpose

This function retrieves an enumeration member's value key using the string supplied from the previously selected enumeration in an open type file. The string may be either a resource string or the programmatic enumeration member name—both will be searched. The value reference parameter will be filled in if the search is successful.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_SELECTED	No selected enumeration.
LDRF_ERR_NOT_FOUND	No enumeration member with that value was found.

LdrfGetEnumMemberCount

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetEnumMemberCount(PLdrfFileInfo pInfo, PUShort pNumMembers)
```

COM Interface Prototype

```
LdrfGeneral2.GetEnumMemberCount(long pInfo, long *pNumMembers
                        long *returnCode)
```

Purpose

This function retrieves the number of enumeration members in the specified enumeration set.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_SELECTED	No selected enumeration.
LDRF_ERR_NOT_FOUND	No enumeration member with that value was found.

LdrfGetEnumMemberByIndex

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetEnumMemberByIndex(PLdrfFileInfo pInfo, TUShort index,
                         PByte pValue, LPSTR pString,
                         PUShort pLength, PUByte pResScope,
                         PULong pResIndex)
```

COM Interface Prototype

```
LdrfGeneral2.GetEnumMemberByIndex(long pInfo, long index,
                         long *pValue, BSTR *pString,
                         long *pResScope,
                         long *pResIndex, long *returnCode)
```

Purpose

This function gets the enumeration set member with the specified index.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_SELECTED	No selected enumeration.
LDRF_ERR_NOT_FOUND	No enumeration member with that value was found.

LdrfSetEnumMember

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetEnumMember(PLdrfFileInfo pInfo, TByte value,
            LPCSTR string, TUByte resSel, TULong resIndex)
```

COM Interface Prototype

```
LdrfTypes.SetEnumMember(long pInfo, long value, BSTR string,
         long resSel, long resIndex, long *returnCode)
```

Purpose

This function adds or modifies an enumeration member in the currently selected enumeration.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_SELECTED	No selected enumeration.

LdrfValidateEnumSet

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfValidateEnumSet(PLdrfFileInfo pInfo, TUShort Index)
```

COM Interface Prototype

```
LdrfMiscFns1.ValidateEnumSet(long pInfo, long index, long *returnCode)
```

Purpose

This function returnw a value that indicates the status of this enumeration set. See *Return Values* for more information.

Return Values

LDRF_ERR_PARAM	Incorrect parameters supplied.
LDRF_ERR_NOT_FOUND	The specified enumeration set was not found.
LDRF_ERR_INTERNAL	Internal error.
LDRF_ERR_NONE	The enumeration set was found and is not empty.

LDRF_ERR_EMPTY_RECORD	The enumeration set is an empty record (i.e. it was deleted). This error code is only returned if the **LdrfEnableEmptyEntries()** function was called on the type file.

NVT Access Functions for a Type File

The functions described in this section manage network variable types (NVTs), including NVT properties such as type name and descriptive strings. You can explore the fundamental data type using the type tree functions, discussed later in this document.

LdrfGetNVT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetNVT(PLdrfFileInfo pInfo, TUShort index,
      PLdrfTypeTree *ppTypeTree))
```

COM Interface Prototype

```
LdrfTypes.GetNVT(long pInfo, long index,
     long *ppTypeTree, long *returnCode)
```

Purpose

This function retrieves a network variable type by index from an open type file. The **pInfo** parameter for the file is supplied, along with the index for the network variable type. The type description is read in by the routine, and is returned in the **ppTypeTree** reference parameter. The type tree contains the programmatic name string for the type and three resource string indices for name, comment, and units. You must call the **LdrfFreeTypeTree()** function to free the type tree when your application is done with it. If extended SNVT-ID support is not enabled, this function hides standard network variable types whose ID exceeds 250.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No network variable type with that index was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfGetNVTEx

☑ LCADRF32.DLL ☐ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetNVTEx(PLdrfFileInfo pInfo, TUShort index,
      PLdrfTypeTree *ppTypeTree, PUSHORT pFlags))
```

Purpose

This function is identical to the **LdrfGetNVT()** function, with the addition of a new **pFlags** parameter. The **pFlags** parameter is reserved for future use and the value is set to zero by the **LdrfGetNVTEx()** function.

This function retrieves a network variable type by index from an open type file. The **pInfo** parameter for the file is supplied, along with the index for the network variable type. The type description is read in by the routine, and is returned in the **ppTypeTree** reference parameter. The type tree contains the programmatic name string for the type and three resource string indices for name, comment, and units. Your application must call the **LdrfFreeTypeTree()** function to free the type tree when your application is done with it. If extended SNVT-ID support is not enabled, this function hides standard network variable types whose ID exceeds 250.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No network variable type with that index was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfGetNVTByName

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetNVTByName(PLdrfFileInfo pInfo, LPCSTR name,
           PUShort pIndex, PLdrfTypeTree *ppTypeTree)
```

COM Interface Prototype

```
LdrfTypes.GetNVTByName(long pInfo, BSTR name, long *pIndex,
        long *ppTypeTree, long *returnCode)
```

Purpose

This function retrieves a network variable type from an open type file using the programmatic name key string for the type. The **pInfo** parameter for the file is supplied, along with a pointer to the name string for the network variable type. The index of the network variable type is returned. The type description is read in by the function, and is returned in the **ppTypeTree** reference parameter. The type tree contains the programmatic name string for the type and three resource string indices for name, comment, and units. Your application must call the **LdrfFreeTypeTree()** function to free the type tree when your application is done with it. If extended SNVT-ID support is not enabled, this function hides standard network variable types whose ID exceeds 250.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No network variable type with that name was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfGetNVTByNameEx

☑ LCADRF32.DLL ☐ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetNVTByName(PLdrfFileInfo pInfo, LPCSTR name,
            PUShort pIndex, PLdrfTypeTree *ppTypeTree, PUShort pFlags)
```

Purpose

This function is identical to the **LdrfGetNVTByName()** function, with the addition of a new **pFlags** parameter. The **pFlags** parameter is reserved for future use and the value is set to zero by the **LdrfGetNVTByNameEx()** function.

This function retrieves a network variable type from an open type file using the programmatic name key string for the type. The **pInfo** parameter for the file is supplied, along with a pointer to the name string for the network variable type. The index of the network variable type is returned. The type description is read in by the function, and is returned in the **ppTypeTree** reference parameter. The type tree contains the programmatic name string for the type and three resource string indices for name, comment, and units. Your application must call the **LdrfFreeTypeTree()** function to free the type tree when your application is done with it. If extended SNVT-ID support is not enabled, this function hides standard network variable types whose ID exceeds 250.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No network variable type with that name was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfLookupTypeNameString

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfLookupTypeNameString(LPCSTR pTypeNameString,
                         PNVTTYPE pNvtType)
```

COM Interface Prototype

```
LdrfMiscFns1.GetLookupNameString(BSTR*pString, long *pNvtType,
                                 long *returnCode)
```

Purpose

This function performs a case-insensitive search of the **TNVTType** enumeration for the provided string. If a match is found, the corresponding network variable type is returned.

See **LdrfGetTypeNameString()** for the inverse operation.

Return Values

LDRF_ERR_NOT_FOUND	The string was not found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfSetNVT

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetNVT(PLdrfFileInfo pInfo,
      TUShort index, PLdrfTypeTree pTypeTree)
```

COM Interface Prototype

```
LdrfTypes.SetNVT(long pInfo, long index, long pTypeTree,
     long *returnCode)
```

Purpose

This function adds or modifies a network variable type in a type file that has already been opened for editing. The index must be supplied, along with a pointer to the new type tree. The type tree contains the programmatic name string for the type and three resource string indices for name, comment, and units. The index is used as the key for the network variable type record to change. The name string key is checked to make sure it is not a duplicate conflicting with another record. Type trees must be constructed using the type tree functions described later in this document. Your application must call the **LdrfFreeTypeTree()** function to free the type trees when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No network variable type with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_DUPLICATE	The name key is already in use by another network variable type in the file.
LDRF_ERR_TYPE_TREE	The type tree structure is invalid.
LDRF_ERR_FMT_VERSION	The resource file format does not support the specified data type. The **NVT_TYPE_UNSIGNED_QUAD** and **NVT_TYPE_DOUBLE_FLOAT** data types require version 5 or later. The **NVT_TYPE_SIGNED_INT64** and **NVT_TYPE_UNSIGNED_INT64** data types require version 6 or later.

LdrfSetNVTEx

☑ LCADRF32.DLL ☐ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetNVTEx(PLdrfFileInfo pInfo,
      TUShort index, PLdrfTypeTree pTypeTree, TUShort flags)
```

Purpose

This function is identical to the **LdrfSetNVT()** function, with the addition of a new **flags** parameter. The **flags** parameter is reserved for future use and the value must be set to zero when calling the **LdrfSetNVTEx()** function.

This function adds or modifies a network variable type in a type file that has already been opened for editing. The index must be supplied, along with a pointer to the new type tree. The type tree contains the programmatic name string for the type and three resource string indices for name, comment, and units. The index is used as the key for the network variable type record to change. The name string key is checked to make sure it is not a duplicate conflicting with another record. Type trees must be constructed using the type tree functions described later in this document. Your application must call the **LdrfFreeTypeTree()** function to free the type trees when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LDRF_ERR_NOT_FOUND	No network variable type with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_DUPLICATE	The name key is already in use by another network variable type in the file.
LDRF_ERR_TYPE_TREE	The type tree structure is invalid.
LDRF_ERR_FMT_VERSION	The resource file format does not support the specified data type. The **NVT_TYPE_UNSIGNED_QUAD** and **NVT_TYPE_DOUBLE_FLOAT** data types require version 5 or later. The **NVT_TYPE_SIGNED_INT64** and **NVT_TYPE_UNSIGNED_INT64** data types require version 6 or later.

LdrfSetNVTObsolete

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetNVTObsolete(PLdrfFileInfo pInfo, TUShort index)
```

COM Interface Prototype

```
LdrfGeneral2.SetNVTObsolete(long pInfo, long index, long *returnCode)
```

Purpose

This function marks the specified network variable type as obsolete. Marking a type as obsolete does not affect the processing of the file. It is up to the calling application to interpret the obsolete mark. The obsolete mark is cleared when the network variable type is edited. To edit a network variable type and leave the obsolete mark intact, you must check for the mark using **LdrfGetNVTObsolete()** before making any changes and call this function after you are done editing.

You cannot use obsolete types within new definitions, but existing definitions may continue referencing obsolete types.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No network variable type with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_TYPE_TREE	The type tree structure is invalid.
LDRF_ERR_FMT_VERSION	The resource file format does not support the obsolete mark. The obsolete mark is supported in version 3 and later.

LdrfGetNVTObsolete

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetNVTObsolete(PLdrfFileInfo pInfo, TUShort index, PboolByte
                   pObsolete)
```

COM Interface Prototype

```
LdrfGeneral2.GetNVTObsolete(long pInfo, long index,
                            long *pObsolete, long *returnCode)
```

Purpose

This function checks for the obsolete flag on the specified network variable type. Marking a type as obsolete does not affect the processing of the file. It is up to the calling application to interpret the obsolete mark. This function will return FALSE if called on a resource file that does not support the obsolete mark; no error will be returned.

You cannot use obsolete types within new definitions, but existing definitions may continue referencing obsolete types. If extended SNVT-ID support is not enabled, this function will hide standard network variable types whose ID exceeds 250.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No network variable type with that index was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfFindEmptyNVT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfFindEmptyNVT(PLdrfFileInfo pInfo, PUShort pIndex)
```

COM Interface Prototype

```
LdrfMscFns1.FindEmptyNVT(long pInfo, long *pIndex,
                                     long *returnCode)
```

Purpose

This function returns the first empty network variable type index. If there are no empty network variable type records, this function returns n+1, where n is the number of network variable type records in the file.

Empty records may be a result from marking a previously exising record as deleted, and having purged the type file. This function allows reclaiming the index that has been freed.

The function supports the traditional SNVT-ID range 1…250 unless the extended SNVT-ID support is enabled.

Return Values

LDRF_ERR_INTERNAL	Internal error.
LDRF_ERR_NOT_FOUND	No empty record index is available (only occurs if file is full).

LdrfDeleteNVT

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfDeleteNVT(PLdrfFileInfo pInfo, TUShort index)
```

COM Interface Prototype

```
LdrfMiscFns1.DeleteNVT(long pInfo, long index, long *returnCode)
```

Purpose

This function deletes a network variable type. Deleted resources do not consume any file data space. They only have NULL entries in the resource key-access directories.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No network variable type with that index was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.
LDRF_ERR_FMT_VERSION	The function was called on a pre-version 4 type file.

LdrfValidateNVT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfValidateNVT(PLdrfFileInfo pInfo, TUShort Index)
```

COM Interface Prototype

```
LdrfMiscFns1.ValidateNVT(long pInfo, long index, long *returnCode)
```

Purpose

This function returns a value that indicates the status of the specified network variable type. See *Return Values* for more information. If extended SNVT-ID support is not enabled, this function hides standard network variable types whose ID exceeds 250.

Return Values

LDRF_ERR_PARAM	Incorrect parameters supplied.
LDRF_ERR_NOT_FOUND	The specified network variable type was not found.
LDRF_ERR_INTERNAL	Internal error.
LDRF_ERR_NONE	The network variable type was found and is not empty.
LDRF_ERR_EMPTY_RECORD	The network variable type is an empty record (i.e. it was deleted). This error code will only be returned if the **LdrfEnableEmptyEntries()** function was called on the type file.

CPT Access Functions for a Type File

The functions discussed in this section manage configuration property types, including CPT properties such as type name, descriptive strings, and similar aspects. The fundamental data type can be explored using the Type Tree Functions, discussed later in this document.

LdrfGetCPT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetCPT(PLdrfFileInfo pInfo, TUShort index,
      PLdrfTypeTree *ppTypeTree, PBool pInheritable,
      PUByteArray *ppMin, PUByteArray *ppMax,
      PUByteArray *ppInit, PUShort pByteArrayLen)
```

COM Interface Prototype

```
LdrfTypes.GetCPT(long pInfo, long index, long *ppTypeTree,
     long *pInheritable, BSTR *ppMin, BSTR *ppMax,
     BSTR *ppInit, long *returnCode)
```

Purpose

This function retrieves a configuration property type from an open type file. The **pInfo** parameter for the file is supplied, along with the index for the configuration property type. Each configuration property type has a programmatic string and several resource string indices, all are returned. Three byte arrays are returned, representing the default min, max, and initial values for the configuration property type. The type description is read in by the function, and is returned in the **ppTypeTree** reference parameter. Your application must call the **LdrfFreeTypeTree()** function to free the type tree when done with it. Your application must call the **LdrfFreeByteArray()** function to free each of the min, max, and initial value arrays when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfGetCPTEx

 LCADRF32.DLL  COM Interface  LDRF32R.DLL

C Language API

```
LdrfGetCPTEx(PLdrfFileInfo pInfo, TUShort index,
      PLdrfTypeTree *ppTypeTree, PBool pInheritable,
      PUByteArray *ppMin, PUByteArray *ppMax,
      PUByteArray *ppInit, PUByteArray *ppInvalid,
      PUShort pByteArrayLen)
```

COM Interface Prototype

```
LdrfTypes.GetCPTEx(long pInfo, long index, long *ppTypeTree,
     long *pInheritable, BSTR *ppMin, BSTR *ppMax,
     BSTR *ppInit, BSTR *ppInvalid, long *returnCode)
```

Purpose

This function is identical to the **LdrfGetCpt()** function except that it returns an additional reference parameter for a byte array containing the default invalid value for the configuration property type.

This function retrieves a configuration property type from an open type file. The **pInfo** parameter for the file is supplied, along with the index for the configuration property type. Each configuration property type has a programmatic string and several resource string indices, all are returned. Four byte arrays are returned, representing the default min, max, initial, and invalid values for the configuration property type. The type description is read in by the function, and is returned in the **ppTypeTree** reference parameter. You must call the **LdrfFreeTypeTree()** function to free the type tree when your application is done with it. Your application must call the **LdrfFreeByteArray()** function to free each of the min, max, initial, and invalid value arrays when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that index was found.

LDRF_ERR_TRUNC	The string was truncated to fit the buffer.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfGetCPTEx2

☑ LCADRF32.DLL ☐ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetCPTEx(PLdrfFileInfo pInfo, TUShort index,
      PLdrfTypeTree *ppTypeTree, PBool pInheritable,
      PUByteArray *ppMin, PUByteArray *ppMax,
      PUByteArray *ppInit, PUByteArray *ppInvalid,
      PUShort pByteArrayLen, PUShort pFlags)
```

Purpose

This function is identical to the **LdrfGetCptEx()** function, with the addition of a new **pFlags** parameter. The **pFlags** parameter is reserved for future use and the value is set to zero by the **LdrfGetCptEx2()** function.

This function retrieves a configuration property type from an open type file. The **pInfo** parameter for the file is supplied, along with the index for the configuration property type. Each configuration property type has a programmatic string and several resource string indices, all are returned. Four byte arrays are returned, representing the default min, max, initial, and invalid values for the configuration property type. The type description is read in by the function, and is returned in the **ppTypeTree** reference parameter. You must call the **LdrfFreeTypeTree()** function to free the type tree when your application is done with it. Your application must call the **LdrfFreeByteArray()** function to free each of the min, max, initial, and invalid value arrays when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfGetCPTByName

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetCPTByName(PLdrfFileInfo pInfo, LPCSTR name,
            PUShort pIndex, PLdrfTypeTree *ppTypeTree,
            PBool pInheritable,
            PUByteArray *ppMin, PUByteArray *ppMax,
            PUByteArray *ppInit, PUShort pByteArrayLen)
```

COM Interface Prototype

```
LdrfTypes.GetCPTByName(long pInfo, BSTR name, long *pIndex,
        long *ppTypeTree, long *pInheritable,
        BSTR *ppMin, BSTR *ppMax,
        BSTR *ppInit, long *returnCode)
```

Purpose

This retrieves a configuration property type from an open type file using the programmatic name key string for the type. The **pInfo** parameter for the file is supplied, along with a pointer to the name string for the configuration property type. The index of the configuration property type is returned. Each configuration property type has several resource string indices, all are returned. Three byte arrays are returned, representing the default min, max, and initial values for the configuration property type. The type description is read in by the function, and is returned in the **ppTypeTree** reference parameter. You must call the **LdrfFreeTypeTree()** function to free the type tree when your application is done with it. Your application must call the **LdrfFreeByteArray()** function to free each of the min, max, and initial value arrays when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that name was found.

LdrfGetCPTByNameEx

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetCPTByNameEx(PLdrfFileInfo pInfo, LPCSTR name,
            PUShort pIndex, PLdrfTypeTree *ppTypeTree,
            PBool pInheritable,
            PUByteArray *ppMin, PUByteArray *ppMax,
            PUByteArray *ppInit, PUByteArray *ppInvalid ,
            PUShort pByteArrayLen)
```

COM Interface Prototype

```
LdrfTypes.GetCPTByNameEx(long pInfo, BSTR name, long *pIndex,
        long *ppTypeTree, long *pInheritable,
        BSTR *ppMin, BSTR *ppMax,
        BSTR *ppInit, BSTR *ppInvalid, long *returnCode)
```

Purpose

This function is identical to the **LdrfGetCPTByName()** function except that it returns an additional reference parameter for a byte array containing the default invalid value for the configuration property type.

This function retrieves a configuration property type from an open type file using the programmatic name key string for the type. The **pInfo** parameter for the file is supplied, along with a pointer to the name string for the configuration property type. The index of the configuration property type is returned. Each configuration property type has several resource string indices, all are returned. Four byte arrays are returned, representing the default min, max, initial, and invalid values for the configuration property type. The type description is read in by the function, and is returned in the **ppTypeTree** reference parameter. You must call the **LdrfFreeTypeTree()** function to free the type tree when your application is done with it. Your application must call the **LdrfFreeByteArray()** function to free each of the min, max, initial, and invalid value arrays when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that name was found.

LdrfGetCPTByNameEx2

☑ LCADRF32.DLL ☐ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetCPTByNameEx2(PLdrfFileInfo pInfo, LPCSTR name,
            PUShort pIndex, PLdrfTypeTree *ppTypeTree,
            PBool pInheritable,
            PUByteArray *ppMin, PUByteArray *ppMax,
            PUByteArray *ppInit, PUByteArray *ppInvalid ,
            PUShort pByteArrayLen, PUShort pFlags)
```

Purpose

This function is identical to the **LdrfGetCptByNameEx()** function, with the addition of a new **pFlags** parameter. The **pFlags** parameter is reserved for future use and the value is set to zero by the **LdrfGetCptByNameEx2()** function.

This function retrieves a configuration property type from an open type file using the programmatic name key string for the type. The **pInfo** parameter for the file is supplied,

along with a pointer to the name string for the configuration property type. The index of the configuration property type is returned. Each configuration property type has several resource string indices, all are returned. Four byte arrays are returned, representing the default min, max, initial, and invalid values for the configuration property type. The type description is read in by the function, and is returned in the **ppTypeTree** reference parameter. You must call the **LdrfFreeTypeTree()** function to free the type tree when your application is done with it. Your application must call the **LdrfFreeByteArray()** function to free each of the min, max, initial, and invalid value arrays when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that name was found.

LdrfFreeByteArray

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfFreeByteArray(PUByteArray pByteArray)
```

Purpose

This function frees a byte array that was allocated by **LdrfGetCPT()**, **LdrfGetCPTEx()**, **LdrfGetCPTEx2()**, **LdrfGetCPTByName()**,**LdrfGetCPTByNameEx(),** or **LdrfGetCPTByNameEx2()**.

Return Values

There are no error codes returned by this function.

LdrfSetCPT

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetCPT(PLdrfFileInfo pInfo, TUShort index,
      PLdrfTypeTree pTypeTree, TBool inheritable,
      PUByteArray pMin, PUByteArray pMax,
      PUByteArray pInit, TUShort byteArrayLen)
```

COM Interface Prototype

```
LdrfTypes.SetCPT(long pInfo, long index,
     long pTypeTree, long inheritable,
     BSTR pMin, BSTR pMax, BSTR pInit, long *returnCode)
```

Purpose

This function adds or modifies a configuration property type in a type file that has been opened for editing. The type tree contains the programmatic name string for the type and three resource string indices for name, comment, and units. Pointers to new byte arrays for min, max, and init values must be supplied. The index is used as the key for the configuration property type record to change. The name key is checked to make sure it is not a duplicate conflicting with another record. Type trees must be constructed using the type tree functions described later in this document. You must call the **LdrfFreeTypeTree()** function to free type trees when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_DUPLICATE	The name key is already in use by another configuration property type in the file.
LDRF_ERR_PARAM	An incorrect parameter was supplied.
LDRF_ERR_TYPE_TREE	The type tree structure is invalid.
LDRF_ERR_FMT_VERSION	The resource file format does not support the specified data type. The **NVT_TYPE_UNSIGNED_QUAD** and **NVT_TYPE_DOUBLE_FLOAT** data types require version 5 or later. The **NVT_TYPE_SIGNED_INT64** and **NVT_TYPE_UNSIGNED_INT64** data types require version 6 or later.

LdrfSetCPTEx

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetCPTEx(PLdrfFileInfo pInfo, TUShort index,
      PLdrfTypeTree pTypeTree, TBool inheritable,
      PUByteArray pMin, PUByteArray pMax,
      PUByteArray pInit, PUByteArray pInvalid,
      TUShort byteArrayLen)
```

COM Interface Prototype

```
LdrfTypes.SetCPTEx(long pInfo, long index,
     long pTypeTree, long inheritable,
     BSTR pMin, BSTR pMax, BSTR pInit, BSTR pInvalid,
     long *returnCode)
```

Purpose

This function is identical to the **LdrfSetCpt()** function except that it takes an additional pointer to a byte array containing the default invalid value for the configuration property type.

This function adds or modifies a configuration property type in a type file that has been opened for editing. The type tree contains the programmatic name string for the type and three resource string indices for name, comment, and units. Pointers to new byte arrays for min, max, init, and invalid values must be supplied. The index is used as the key for the configuration property type record to change. The name key is checked to make sure it is not a duplicate conflicting with another record. Type trees must be constructed using the type tree functions described later in this document. You must call the **LdrfFreeTypeTree()** function to free type trees when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_DUPLICATE	The name key is already in use by another configuration property type in the file.
LDRF_ERR_TYPE_TREE	The type tree structure is invalid.
LDRF_ERR_FMT_VERSION	The resource file format does not support the specified data type. The **NVT_TYPE_UNSIGNED_QUAD** and **NVT_TYPE_DOUBLE_FLOAT** data types require version 5 or later. The **NVT_TYPE_SIGNED_INT64** and **NVT_TYPE_UNSIGNED_INT64** data types require version 6 or later.

LdrfSetCPTEx2

☑ LCADRF32.DLL ☐ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetCPTEx2(PLdrfFileInfo pInfo, TUShort index,
      PLdrfTypeTree pTypeTree, TBool inheritable,
      PUByteArray pMin, PUByteArray pMax,
      PUByteArray pInit, PUByteArray pInvalid,
      TUShort byteArrayLen, TUShort flags)
```

Purpose

This function is identical to the **LdrfSetCPTEx()** function, with the addition of a new **flags** parameter. The **flags** parameter is reserved for future use and the value must be set to zero when calling the **LdrfSetCPTEx2()** function.

This function adds or modifies a configuration property type in a type file that has been opened for editing. The type tree contains the programmatic name string for the type and three resource string indices for name, comment, and units. Pointers to new byte arrays for min, max, init, and invalid values must be supplied. The index is used as the key for the configuration property type record to change. The name key is checked to make sure it is not a duplicate conflicting with another record. Type trees must be constructed using the type tree functions described later in this document. You must call the **LdrfFreeTypeTree()** function to free type trees when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_DUPLICATE	The name key is already in use by another configuration property type in the file.
LDRF_ERR_TYPE_TREE	The type tree structure is invalid.
LDRF_ERR_FMT_VERSION	The resource file format does not support the specified data type. The **NVT_TYPE_UNSIGNED_QUAD** and **NVT_TYPE_DOUBLE_FLOAT** data types require version 5 or later. The **NVT_TYPE_SIGNED_INT64** and **NVT_TYPE_UNSIGNED_INT64** data types require version 6 or later.

LdrfSetCPTObsolete

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfSetCPTObsolete(PLdrfFileInfo pInfo, TUShort index)
```

COM Interface Prototype

```
LdrfGeneral2.SetCPTObsolete(long pInfo, long index, long *returnCode)
```

Purpose

This function marks the specified configuration property type as obsolete. Marking a type as obsolete does not affect the processing of the file. It is up to the calling application to interpret the obsolete mark. To edit a configuration property type and leave the obsolete mark intact, you must check for the mark using **LdrfGetCPTObsolete()** before making any changes and call this function after you are done editing.

You cannot use obsolete types with new definitions, but you may continue using existing definitions that reference obsolete types.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No configuration property type with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_DUPLICATE	The name key is already in use by another configuration property type in the file.
LDRF_ERR_TYPE_TREE	The type tree structure is invalid.
LDRF_ERR_FMT_VERSION	The resource file format does not support the obsolete mark. The obsolete mark is supported in version 3 and later.

LdrfGetCPTObsolete

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetCPTObsolete(PLdrfFileInfo pInfo, TUShort index, PboolByte
                   pObsolete)
```

COM Interface Prototype

```
LdrfGeneral2.GetCPTObsolete(long pInfo, long index,
                            long *pObsolete, long *returnCode)
```

Purpose

This function checks for the obsolete flag on the specified configuration property type. Marking a type as obsolete does not affect the processing of the file. It is up to the calling application to interpret the obsolete mark. This function will return FALSE if called on a resource file that does not support the obsolete mark; no error will be returned.

You cannot use obsolete types with new definitions, but you may continue using existing definitions that reference obsolete types.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No configuration property type with that index was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfFindEmptyCPT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfFindEmptyCPT(PLdrfFileInfo pInfo, PUShort pIndex)
```

COM Interface Prototype

```
LdrfMiscFns1.FindEmptyCPT(long pInfo, long *pIndex,
                                  long *returnCode)
```

Purpose

This function returns the first empty configuration property type index. If there are no empty configuration property type records, this function returns n+1, where n is the number of configuration property type records in the file.

Empty records may be a result from marking a previously exising record as deleted, and having purged the type file. This function allows reclaiming the index that has been freed.

Return Values

LDRF_ERR_INTERNAL	Internal error.
LDRF_ERR_NOT_FOUND	No empty record index is available (only occurs if file is full).

LdrfDeleteCPT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfDeleteCPT(PLdrfFileInfo pInfo, TUShort index)
```

COM Interface Prototype

```
LdrfMiscFns1.DeleteCPT(long pInfo, long index, long *returnCode)
```

Purpose

This function deletes a configuration property type. Deleted resources do not consume any file data space. They only have NULL entries in the resource key-access directories.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No configuration property type with that index was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_FMT_VERSION	The function was called on a pre-version 4 type file.

LdrfValidateCPT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfValidateCPT(PLdrfFileInfo pInfo, TUShort Index)
```

COM Interface Prototype

```
LdrfMiscFns1.ValidateCPT(long pInfo, long index, long *returnCode)
```

Purpose

This function returns a value that indicates the status of the specified configuration property type. See *Return Values* for more information.

Return Values

LDRF_ERR_PARAM	Incorrect parameters supplied.
LDRF_ERR_NOT_FOUND	The specified configuration property type was not found.
LDRF_ERR_INTERNAL	Internal error.
LDRF_ERR_NONE	The configuration property type was found and is not empty.
LDRF_ERR_EMPTY_RECORD	The configuration property type is an empty record (i.e. it was deleted). This error code will only be returned if the **LdrfEnableEmptyEntries()** function was called on the type file.

Type Tree Functions

Type trees represent the structure and definition of data types. The type tree for a scalar is a single node. The type tree for a more complex data type, such as a union, struct, or array, is a recursive type tree. You can use the functions described in this section to traverse through an existing type tree, and to build a new type tree. Type tree nodes contain state information permitting multiple calls to construct a type tree, or to scan a type tree.

LdrfFreeTypeTree

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfFreeTypeTree(PLdrfTypeTree pTypeTree)
```

COM Interface Prototype

```
LdrfTypeTree.FreeTypeTree(long pTypeTree, long *returnCode)
```

Purpose

This function frees a type tree linked structure that was allocated by the **LdrfGetNVT()**,**LdrfGetNVTEx()**,**LdrfGetNVTByName()**,**LdrfGetNVTByNameEx()**, **LdrfGetCPT()**, **LdrfGetCPTEx()**,**LdrfGetCPTEx2()**, **LdrfGetCPTByName()**, **LdrfGetCPTByNameEx()**, or **LdrfGetCPTByNameEx2()** functions, and to free a type tree created by other functions.

Return Values

LDRF_ERR_TYPE_TREE	The type tree structure is invalid.

LdrfGetNextSupportedNVTType

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetNextSupportedNVTType(PLdrfFileInfo pInfo, TNVTType *pNvtType)
```

COM Interface Prototype

```
LdrfMiscFns1.GetNextSupportedNVTType(long pInfo, long *pNvtType,
                                     long *returnCode)
```

Purpose

This function returns the next base type supported by the specified type tree node for a specified type file version. You can use this function to enumerate all supported fundamental data types supported by a given type file. Fundamental data types are the types that are used to define a network variable or configuration property type, and include types such as **NVT_TYPE_UNSIGNED_CHAR** or **NVT_TYPE_SIGNED_LONG**, but also include arrays, structures, unions, or references to other types. Extended data types are included if the client has previously called **LdrfExtendedDataTypeAware()**. Only the data types specified as supported are returned.

If **pInfo** is set to NULL, this function returns the next available network variable type, irrespective of required version.

Return Values

LDRF_ERR_FILE_INFO	The file info structure contents was not valid.
LDRF_ERR_NOT_FOUND	The network variable type was not found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfGetTypeNameString

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetTypeNameString(TNVTType nvtType, LPSTR pTypeNameString,
                      PUShort pLength)
```

COM Interface Prototype

```
LdrfMiscFns1.GetTypeNameString(long nvtType, BSTR*pString,
                                    long *returnCode)
```

Purpose

This function returns the name of the given base type in U.S. English (from the **TNVTType** enumeration, defined in **lcadrf.h**). The names provided are similar to C type names, but names such as “reference,” “bitfield”, or “array,” which are not C language type names, may also be returned.

The type names provided by this function are designed for exposure in graphical user interfaces, and similar applications.

See **LdrfLookupTypeNameString()** for the inverse operation.

Return Values

LDRF_ERR_PARAM	The network variable type was not found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_TRUNC	Name truncated (output buffer too short)

LdrfNewTypeTreeNode

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfNewTypeTreeNode(TNVTType newNodeType, LPCSTR name,
               TUByte resNmSel,  TULong resNmIndex,
               TUByte resCmtSel, TULong resCmtIndex,
               TUByte resUntSel, TULong resUntIndex,
               PLdrfTypeTree *ppTypeTree)
```

COM Interface Prototype

```
LdrfTypeTree.NewTypeTreeNode(long newNodeType, BSTR name,
           long resNmSel,  long resNmIndex,
           long resCmtSel, long resCmtIndex,
           long resUntSel, long resUntIndex,
           long *pTypeTree, long *returnCode)
```

Purpose

This function adds a new node to an existing type tree, or creates a new type tree. The node is added in the next position, for example, if the last thing done to the tree was to create an array type, then a call to this function adds the definition for the array element. Type nodes are added in depth first order. For example, if adding a structure's fields, and one of the fields is itself a structure, the *nested* structure's fields must all be added before returning to add fields to the outer structure. This is identical to the order of declarations in the C language.

If the **ppTypeTree** parameter is a pointer to an existing type tree, the new node is added to the specified tree. If the **ppTypeTree** parameter is NULL, a new type tree is created and the new type tree pointer is returned via the first reference parameter. The other parameters are a node type for the new node, a string pointer to a programmatic name for the type element, and three optional resource string indices for language-dependent name, comment, and units for the type element.

The type tree is marked as incomplete if the definition isn't finished. Once the node is added, it must be set to proper values using one of the set details functions below. At that point, if the tree is complete, an **LDRF_ERR_NONE** value will be returned.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a set details function was expected, or the type is already complete.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	New node accepted; a call to a set function is required next.

LdrfResolveAllTypeTreeRefs

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfResolveAllTypeTreeRefs  ( PLdrfFileInfo pCatalogInfo,
                              PUByteArray pProgID,
                              PLdrfTypeTree pTypeTree,
                              PUShort pTypeSize );
```

COM Interface Prototype

```
ILdrfGeneral2.ResolveAllTypeTreeRefs (long pCatalogInfo,
                                      BSTR progID,
                                      long pTypeTree,
                                      long * pTypeSize,
                                      long * returnCode );
```

Purpose

This function resolves all references and grafts into the type tree, flattening the type tree. This does not affect the contents of the type inside the resource file. The **pTypeSize** parameter refers to the initial size of the type, not the current size. To get the current size of the type, you must perform a comprehensive type tree walk using the **LdrfScanTypeTree()**, **LdrfReadTypeTreeNode()**, and **LdrfGetDetails()** functions, as necessary.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_FILE_INFO	The file info structure contents was not valid.
LDRF_ERR_NOT_FOUND	One of the referenced types was not found.

LdrfSetScalarDetails

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetScalarDetails(PLdrfTypeTree pTypeTree,
                TLong minValid, TLong maxValid,
            TShort scaleA, TShort scaleB, TShort scaleC)
```

COM Interface Prototype

```
LdrfTypeTree.SetScalarDetails(long pTypeTree,
            long minValid, long maxValid,
            long scaleA, long scaleB, long scaleC,
            long *returnCode)
```

Purpose

This function sets the details for an 8-, 16-, or 32-bit scalar node in a type tree (not including an enum, bitfield, or float). The parameters are min and max validation constants, in raw form, and scaling values. This type may complete the tree, depending on nested context.

If you are using 32-bit extended date types, this function will accept signed and unsigned quad values. Unsigned long 32-bit values must be cast to signed long in the call to the function. See the **LdrfSetScalar64Details()** function for setting details for 64-bit scalar types.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfSetScalar64Details

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetScalar64Details(PLdrfTypeTree pTypeTree,
                __DRF_INT64 minValid, __DRF_INT64 maxValid,
            TShort scaleA, TShort scaleB, TShort scaleC)
```

COM Interface Prototype

```
LdrfScalar64Types.SetScalar64Details(long pTypeTree,
            hyper minValid, hyper maxValid,
            long scaleA, long scaleB, long scaleC,
            long *returnCode)
```

Purpose

This function sets the details for a 64-bit scalar node in a type tree (not including an enum, bitfield, or float). The parameters are min and max validation constants, in raw form, and scaling values. This type may complete the tree, depending on nested context.

This function will accept signed and unsigned 64-bit values. Unsigned 64-bit values must be cast to signed 64-bit values in the call to the function. See the **LdrfSetScalarDetails()** function for setting details for 8-, 16-, and 32-bit scalar types.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfSetScalarInvalidValue

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetScalarInvalidValue(PLdrfTypeTree pTypeTree,
                          TLong invalidValue)
```

COM Interface Prototype

```
LdrfGeneral2.SetScalarInvalidValue(long pTypeTree,
                          long invalidValue, long *returnCode)
```

Purpose

This function sets the invalid value for the specified 8-, 16-, and 32-bit scalar. Invalid values are supported in type files of version 3 or later. This function must be called with **LdrfSetScalarDetails()**. See **LdrfSetScalar64InvalidValue()** for setting the invalid value for 64-bit scalar types.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfSetScalar64InvalidValue

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetScalar64InvalidValue(PLdrfTypeTree pTypeTree,
                            __DRF_INT64 invalidValue)
```

COM Interface Prototype

```
LdrfScalar64Types.SetScalar64InvalidValue(long pTypeTree,
                           hyper invalidValue, long *returnCode)
```

Purpose

This function sets the invalid value for the specified scalar for 64-bit scalar types. Invalid values are supported in type files of version 3 or later. This function must be called with **LdrfSetScalar64Details()**. See **LdrfSetScalarInvalidValue()** for setting the invalid value for 8-, 16-, and 32-bit scalar types.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfSetFloatDetails

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFloatDetails(PLdrfTypeTree pTypeTree,
              float minValid, float maxValid)
```

COM Interface Prototype

```
LdrfTypeTree.SetFloatDetails(long pTypeTree,
           long minValid, long maxValid,
           long *returnCode)
```

Purpose

This function sets the details for a float scalar node in a type tree. The parameters are min and max validation constants. This type may complete the tree, depending on nested context.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call toa new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfSetDoubleFloatDetails

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetDoubleFloatDetails(PLdrfTypeTree pTypeTree,
              double minValid, double maxValid)
```

COM Interface Prototype

```
LdrfMiscFns1.SetDoubleFloatDetails(long pTypeTree,
           double minValid, double maxValid,
           long *returnCode)
```

Purpose

This function sets the details for a double float scalar node in a type tree. The parameters are min and max validation constants. This type may complete the tree, depending on nested context.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfSetBitfieldDetails

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetBitfieldDetails(PLdrfTypeTree pTypeTree, TUByte bitfSize,
                  TUByte bitfOffset, TBool bitfSigned,
                  TLong minValid, TLong maxValid,
                  TShort scaleA, TShort scaleB, TShort scaleC)
```

COM Interface Prototype

```
LdrfTypeTree.SetBitfieldDetails(long pTypeTree, long bitfSize,
              long bitfOffset, long bitfSigned,
              long minValid, long maxValid,
              long scaleA, long scaleB, long scaleC,
              long *returnCode)
```

Purpose

This function sets the details for a bitfield scalar node in a type tree. The parameters are offset, size, and signedness of the bitfield within an 8-bit byte, min and max validation constants, in raw form, and scaling factors. This type may complete the tree, depending on nested context.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.
LDRF_ERR_PARAM	Bad parameters (e.g. bitfield offset plus size greater than eight). The minium and maximum values are not validated against the bitfield size and signedness.

LdrfSetEnumDetails

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetEnumDetails(PLdrfTypeTree pTypeTree,
              TUByte enumSel, TUShort enumIndex,
              TLong minValid, TLong maxValid)
```

COM Interface Prototype

```
LdrfTypeTree.SetEnumDetails(long pTypeTree,
          long enumSel, long enumIndex,
          long minValid, long maxValid, long *returnCode)
```

Purpose

This function is called to set the details for an enumeration scalar node in a type tree. The parameters are enumeration selector and index and min and max validation constants, in raw form. This type may complete the tree, depending on nested context.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfSetArrayDetails

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetArrayDetails(PLdrfTypeTree pTypeTree,
               TUShort numElements)
```

COM Interface Prototype

```
LdrfTypeTree.SetArrayDetails(long pTypeTree, long numElements,
           long *returnCode)
```

Purpose

This function sets the details for an array node in a type tree. The parameter is the number of elements in the array. The type tree node or nodes for an element must be added next.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.

LDRF_ERR_SEQUENCE	Incorrect sequence of calls, was expecting a call to a new node function, or a different set details function.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfSetStructUnionDetails

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetStructUnionDetails(PLdrfTypeTree pTypeTree,
                    TUShort numFields)
```

COM Interface Prototype

```
LdrfTypeTree.SetStructUnionDetails(long pTypeTree, long numFields,
                 long *returnCode)
```

Purpose

This function sets the details for a structure or union node in a type tree. The parameter is the number of fields in the aggregate. The type tree node or nodes for each field in the aggregate must be added next.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfSetReferenceDetails

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetReferenceDetails(PLdrfTypeTree pTypeTree,
                  TUByte typeSelector, TUShort typeIndex,
                  TUShort typeSize)
```

COM Interface Prototype

```
LdrfTypeTree.SetReferenceDetails(long pTypeTree,
               long typeSel, long typeIndex,
               long typeSize, long *returnCode)
```

Purpose

This function sets the details for a reference node in a type tree. Only standard or user-defined network variable types may be referenced. The parameters are the selector, the type index of the SNVT or UNVT (0 selector for SNVT), and the size of the referent type in bytes. This type may complete the tree, depending on nested context.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to a new node function or a different set details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Set call accepted; a call to add a new node is required next.

LdrfScanTypeTree

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfScanTypeTree(PLdrfTypeTree pTypeTree, PUShort pTypeSize,
            PBool pHasRefs)
```

COM Interface Prototype

```
LdrfTypeTree.ScanTypeTree(long pTypeTree,
        long *pTypeSize, long *pHasRefs,
        long *returnCode)
```

Purpose

This function demarcates the beginning of a sequence of type reading calls. Each subsequent call returns information about the next type node in the tree, and returns **LDRF_ERR_INCOMPLETE** until the last tree node is read, and at that time the last call returns **LDRF_ERR_NONE**. If the reading calls are out of sequence, **LDRF_ERR_SEQUENCE** is returned, and that is a non-recoverable error. The initial size type is also returned (to get the current size, you must perform a comprehensive type tree walk using the **LdrfScanTypeTree()**, **LdrfReadTypeTreeNode()**, and **LdrfGetDetails()** functions, as necessary). A flag is returned that indicates if a type tree contains references.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.

LdrfFindTypeTreeNode

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfFindTypeTreeNode(PLdrfTypeTree pTypeTree, TBool relative,
                LPCSTR pFieldName)
```

COM Interface Prototype

```
LdrfTypeTree.FindTypeTreeNode(long pTypeTree,
            long relative, BSTR fieldName,
            long *returnCode)
```

Purpose

This function sets the context of the type tree to a particular branch for a sequence of type reading calls. The context can be set with a fully qualified name from the beginning of the tree, or it can be set with a name relative to the current context. Each subsequent call returns information about the next type node in the tree starting with the branch, and returns **LDRF_ERR_INCOMPLETE** until the last tree node is read, and at that time the last call returns **LDRF_ERR_NONE**. If the reading calls are out of sequence, the return code **LDRF_ERR_SEQUENCE** is returned, and that is a non-recoverable error.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_NOT_FOUND	No field with that name was found.

LdrfReadTypeTreeNode

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfReadTypeTreeNode(PLdrfTypeTree pTypeTree, PNVTType pNodeType,
                PUShort pTypeOffset, PUShort pTypeSize,
                LPSTR pName, PUShort pLength,
                PUByte pResNmSel,  PULong pResNmIndex,
                PUByte pResCmtSel, PULong pResCmtIndex,
                PUByte pResUntSel, PULong pResUntIndex)
```

COM Interface Prototype

```
LdrfTypeTree.ReadTypeTreeNode(long pTypeTree, nvtType *pNodeType,
            long *pTypeOffset, long *pTypeSize,
            BSTR *pName,
            long *pResNmSel,  long *pResNmIndex,
            long *pResCmtSel, long *pResCmtIndex,
            long *pResUntSel, long *pResUntIndex,
            long *returnCode)
```

Purpose

This function reads the next node in the type tree. This call returns the node type, the programmatic name string, and the three resource string indices for name, comment, and units. Callers to this function can furnish a buffer to receive a copy of the programmatic name string, and for the C language API the max length of that buffer must be placed in length value pointed to by **pLength**. The length will be updated to reflect the length of the string. The string doesn't have to be read - a length of 0 and a NULL string pointer can be supplied to not read the string. The byte offset of this type within the overall type is returned.

The call to this function must be followed by a read details call. This function always returns **LDRF_ERR_INCOMPLETE** since it must always be followed by a read details call unless it returns another of the error codes, or returns **LDRF_ERR_NO_MORE_NODES**.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—the tree must be put into scan state by a previous call.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.
LDRF_ERR_NO_MORE_NODES	The scan has reached the end of the type tree.
LDRF_ERR_INCOMPLETE	New node accepted; a call to set details is required next.

LdrfGetScalarDetails

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetScalarDetails(PLdrfTypeTree pTypeTree,
            PLong pMinValid, PLong pMaxValid,
            PShort pScaleA, PShort pScaleB, PShort pScaleC)
```

COM Interface Prototype

```
LdrfTypeTree.GetScalarDetails(long pTypeTree,
            long *pMinValid, long *pMaxValid,
            long *pScaleA, long *pScaleB, long *pScaleC,
            long *returnCode)
```

Purpose

This function gets the details for an 8-, 16, or 32-bit scalar node in a type tree (not including an enum, a bitfield, or a float). The parameters are min and max validation constants, in raw form, and scaling constants. See **LdrfGetScalar64Details()** for getting 64-bit scalar type detals.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get call done; a call to read another type node is required next.

LdrfGetScalar64Details

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetScalar64Details(PLdrfTypeTree pTypeTree,
            __DRF_PINT64 pMinValid, __DRF_PINT64 pMaxValid,
            PShort pScaleA, PShort pScaleB, PShort pScaleC)
```

COM Interface Prototype

```
LdrfScalar64Types.GetScalar64Details(long pTypeTree,
            hyper *pMinValid, hyper *pMaxValid,
            long *pScaleA, long *pScaleB, long *pScaleC,
            long *returnCode)
```

Purpose

This function gets the details for a 64-bit scalar node in a type tree (not including an enum, a bitfield, or a float). The parameters are min and max validation constants, in raw form, and scaling constants. See **LdrfGetScalarDetails()** for getting 8-, 16-, and 32-bitt scalar type detals.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get call done; a call to read another type node is required next.

LdrfGetScalarInvalidValue

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetScalarInvalidValue(PLdrfTypeTree pTypeTree,
                          PBoolByte pInvalidValuePresent,
                          PLong pInvalidValue)
```

COM Interface Prototype

```
LdrfGeneral2.GetScalarInvalidValue(long pTypeTree,
                          long *pInvalidValuePresent,
                          long *pInvalidValue, long *returnCode)
```

Purpose

This function gets the invalid value of the specified 8-, 16-, or 32-bit scalar. Invalid values are supported in version 3 or later type files. This function must be called with **LdrfGetScalarDetails()**. See **LdrfGetScalar64InvalidValue()** for getting the invalid value for 64-bit scalar types.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get function completed; a call to read another type node is required next.

LdrfGetScalar64InvalidValue

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetScalar64InvalidValue(PLdrfTypeTree pTypeTree,
                          PBoolByte pInvalidValuePresent,
                          __DRF_PINT64 pInvalidValue)
```

COM Interface Prototype

```
LdrfScalar64Types.GetScalar64InvalidValue(long pTypeTree,
                          long *pInvalidValuePresent,
                          hyper *pInvalidValue,
                          long *returnCode)
```

Purpose

This function gets the invalid value of the specified 64-bit scalar. 64-bit invalid values are supported in version 6 or later type files. This function must be called with **LdrfGetScalar64Details()**. See **LdrfGetScalarInvalidValue()** for getting the invalid value for 8-, 16-, and 32-bit scalar types.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get function completed; a call to read another type node is required next.

LdrfGetFloatDetails

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFloatDetails(PLdrfTypeTree pTypeTree,
              float *pMinValid, float *pMaxValid)
```

COM Interface Prototype

```
LdrfTypeTree.GetFloatDetails(long pTypeTree,
            float *pMinValid, float *pMaxValid,
            long *returnCode)
```

Purpose

This function gets the details for a float scalar node in a type tree. The parameters are min and max validation constants.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get function completed; a call to read another type node is requied next.

LdrfGetDoubleFloatDetails

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetDoubleFloatDetails(PLdrfTypeTree pTypeTree,
              double * pMinValid, double * pMaxValid)
```

COM Interface Prototype

```
LdrfMiscFns1.GetDoubleFloatDetails(long pTypeTree,
            double *pMinValid, double *pMaxValid,
            long *returnCode)
```

Purpose

This function gets the details for a double float scalar node in a type tree. The parameters are min and max validation constants.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get function completed; a call to read another type node is required next.

LdrfGetBitfieldDetails

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetBitfieldDetails(PLdrfTypeTree pTypeTree, PUByte pBitfSize,
                  PUByte pBitfOffset, PBool pBitfSigned,
                  PLong pMinValid, PLong pMaxValid,
                  PShort pScaleA, PShort pScaleB, PShort pScaleC)
```

COM Interface Prototype

```
LdrfTypeTree.GetBitfieldDetails(long pTypeTree, long *pBitfSize,
              long *pBitfOffset, long *pBitfSigned,
              long *pMinValid, long *pMaxValid,
              long *pScaleA, long *pScaleB, long *pScaleC,
              long *returnCode)
```

Purpose

This function gets the details for a bitfield scalar node in a type tree. The parameters are offset, size, and signedness of bitfield within an 8-bit byte, min, and max validation constants, in raw form, and scaling factors.

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get function completed; a call to read another type node is required next.

LdrfGetEnumDetails

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetEnumDetails(PLdrfTypeTree pTypeTree,
              PUByte pEnumSel, PUShort pEnumIndex,
              PLong pMinValid, PLong pMaxValid)
```

COM Interface Prototype

```
LdrfTypeTree.GetEnumDetails(long pTypeTree,
          long *pEnumSel, long *pEnumIndex,
          long *pMinValid, long *pMaxValid,
          long *returnCode)
```

Purpose

This function gets the details for an enumeration scalar node in a type tree. The parameters are the enumeration selector and index and min and max validation constants.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get function completed; a call to read another type node is required next.

LdrfGetArrayDetails

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetArrayDetails(PLdrfTypeTree pTypeTree, TBool multiple,
                PUShort pNumElements)
```

COM Interface Prototype

```
LdrfTypeTree.GetArrayDetails(long pTypeTree, long multiple,
           long *pNumElements, long *returnCode)
```

Purpose

This function gets the details for an array node in a type tree. The parameter is the number of elements in the array. The type tree node or nodes for the element must be read next.

To linearize a type, retrieve the element nodes *n* times, where *n* corresponds to the array bound. You can use this method to convert a data item from formatted to binary, or vice versa. This can be accomplished by setting the **multiple** boolean parameter to TRUE, and the **LdrfReadTypeTreeNode()** function will take care of the rest, including controlling the type node offset as each element of the array is walked through. Setting **multiple** to FALSE indicates that the subsequent type nodes of array elements should only be returned once. The offset will be for the last element of the array, in that case.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get function completed; a call to read another type node next.

LdrfGetStructUnionDetails

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetStructUnionDetails(PLdrfTypeTree pTypeTree,
                      PUShort pNumFields)
```

COM Interface Prototype

```
LdrfTypeTree.GetStructUnionDetails(long pTypeTree, long *pNumFields,
                 long *returnCode)
```

Purpose

This function gets the details for a structure or union node in a type tree. The parameter is the number of fields in the aggregate. The type tree node or nodes for each field in the aggregate must be read next.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get function completed; a call to read another type node is required next.

LdrfGetReferenceDetails

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetReferenceDetails(PLdrfTypeTree pTypeTree,
                    PUByte pTypeSel, PUShort pTypeIndex)
```

COM Interface Prototype

```
LdrfTypeTree.GetReferenceDetails(long pTypeTree,
                long *pTypeSel, long *pTypeIndex,
                long *returnCode)
```

Purpose

This function sets the details for a reference node in a type tree. The output parameters are the selector and the type index of the SNVT or UNVT (0 selector for SNVTs). This type may complete the tree, depending on nested context.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—a call to the read function or a different get details function was expected.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Get function completed; a call to read another type node is required next.

LdrfGraftReference

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGraftReference(PLdrfTypeTree pTypeTree,
              PLdrfTypeTree pReferentTree)
```

COM Interface Prototype

```
LdrfTypeTree.GraftReference(long pTypeTree, long pReferentTree,
          long *returnCode)
```

Purpose

This function changes a reference node in a type tree into the nodes that make up the referent type. This change cannot be undone, and the knowledge that this node is a reference is lost. The parameters are the type tree containing the type reference, and the referent type, respectively. Following this call, the referent type must not be accessed further, nor can it be freed separately, as it has been grafted into the main type tree.

There are two sets of programmatic names, and groups of resource string indices being combined for this node. The ones in the reference will supersede the ones in the referent. If the reference does not contain resource string indices, however, the ones in the referent will be used.

LdrfGraftReference() is an in-memory operation that does not change the type tree in the resource file.

Return Values

LDRF_ERR_TYPE_TREE	Invalid type tree or referent tree passed in.
LDRF_ERR_SEQUENCE	Incorrect sequence of calls—this call must be the next operation on a type tree following the **LdrfReadTypeTreeNode()** operations that returned a node type of **NVT_TYPE_REFERENCE**, and then a call to the **LdrfGetReferenceDetails()** access function. The caller can use the selector to resolve the reference and obtain a type tree for the referent.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	Paste function completed; a call to read another type node next to get the first node of the referent type is required next.

LdrfApplyValOverride

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfApplyValOverride(PLdrfTypeTree pTypeTree,
                PUByteArray pValMin, PUByteArray pValMax)
```

COM Interface Prototype

```
LdrfTypeTree.ApplyValOverride(long pTypeTree, BSTR valMin, BSTR valMax.
            Long *returnCode)
```

Purpose

This function applies a validation override to the nodes in a type tree. This change cannot be undone. The parameters are the type tree and one or both pointers to byte arrays containing min and max validation information (either or both pointers may be NULL).

Return Values

LDRF_ERR_TYPE_TREE Invalid type tree passed in.

LdrfApplyValOverrideEx

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfApplyValOverrideEx(PLdrfTypeTree pTypeTree,
                  PUByteArray pValMin, PUByteArray pValMax, PUByteArray
                  pValInvalid)
```

COM Interface Prototype

```
LdrfTypeTree.ApplyValOverrideEx(long pTypeTree, BSTR valMin,
                  BSTR valMax, BSTR valInvalid.Long *returnCode)
```

Purpose

This function is identical to the **LdrfApplyValOverride()** function except that it takes an additional pointer to a byte array containing the invalid validation information.

This function applies a validation override to the nodes in a type tree. This change cannot be undone. The parameters are the type tree and one, two, or all three pointers to byte arrays containing min, max, and invalid validation information (either or all pointers may be NULL).

Return Values

LDRF_ERR_TYPE_TREE Invalid type tree passed in.

Functional Profile Template File Functions

Functional profiles group related network variables and configuration properties, optionally combined with additional details such as descriptive strings of overrides for certain aspects, into a single entity. Functional profiles are the type definitions for functional blocks, which are also known as *LonMark objects*. Functional profiles are defined within resource files called *functional profile template files*.

LdrfGetFPTFileInfo

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTFileInfo(PLdrfFileInfo pInfo,
              PUShort pResDep0, PUShort pResDep1,
              PUShort pResDep2, PUShort pResDep3,
              PUShort pResDep4, PUShort pResDep5,
              PUShort pResDep6,
              PUShort pTypDep0, PUShort pTypDep1,
              PUShort pTypDep2, PUShort pTypDep3,
              PUShort pTypDep4, PUShort pTypDep5,
              PUShort pTypdep6,
              PUShort pNumFPTs)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTFileInfo(long pInfo,
          long *pResDep0, long *pResDep1,
          long *pResDep2, long *pResDep3, long *pResDep4,
          long *pResDep5, long *pResDep6,
          long *pTypDep0, long *pTypDep1, long *pTypDep2,
          long *pTypDep3, long *pTypDep4, long *pTypDep5,
          long *pTypDep6, long *pNumFPTs,
          long *returnCode)
```

Purpose

This function gets information specific to the open functional profile template file. The **pInfo** parameter is supplied, along with pointers to the different types of data.

Two sets of dependency data are supplied, but this data is not commonly used. The **pNumFPTs** output parameter returns the number of functional profiles defined in the functional profile file.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	The file has not been completely created.

LdrfSetFPTFileInfo

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTFileInfo(PLdrfFileInfo pInfo,
              TUShort resDep0, TUShort resDep1,
              TUShort resDep2, TUShort resDep3,
              TUShort resDep4, TUShort resDep5,
              TUShort resDep6,
              TUShort typDep0, TUShort typDep1,
              TUShort typDep2, TUShort typDep3,
              TUShort typDep4, TUShort typDep5,
              TUShort typdep6)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.SetFPTFileInfo(long pInfo,
          long resDep0, long resDep1,
          long resDep2, long resDep3, long resDep4,
          long resDep5, long resDep6, long typDep0,
          long typDep1, long typDep2, long typDep3,
          long typDep4, long typDep5, long typDep6,
          long *returnCode)
```

Purpose

This function is sets dependency information specific to the open functional profile file. The **pInfo** parameter is supplied, along with the dependency data.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfGetFPT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPT(PLdrfFileInfo pInfo, TUShort index,
      PUShort pKey, LPSTR pName, PUShort pLength,
      PUByte pResNmSel,  PULong pResNmIndex,
      PUByte pResCmtSel, PULong pResCmtIndex,
      PUByte pManNVs, PUByte pOptNVs,
      PUByte pManCPs, PUByte pOptCPs,
      PUByte pPrincipalNV)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPT(long pInfo, long index,
     long *pKey, BSTR *pName,
     long *pResNmSel,  long *pResNmIndex,
     long *pResCmtSel, long *pResCmtIndex,
     long *pManNVs, long *pOptNVs,
     long *pManCPs, long *pOptCPs,
     long *pPrincipalNV, long *returnCode)
```

Purpose

This function retrieves a functional profile from an open functional profile template file. The **pInfo** parameter for the file is supplied, along with the index for the functional profile. Each functional profile also has a 16-bit numeric key, and this is returned. This key is used to identify the profile's implementation in the node's SD string. Each functional profile has a programmatic string and two resource string indices, all are returned. The caller must allocate a buffer to hold the string, and pass the length of the buffer through the length reference parameter (C language API only), which will be altered to indicate the length actually read. Four byte values are returned, respectively representing the number of mandatory network variables, optional network variables, mandatory configuration properties, and optional configuration properties in the functional profile. Lastly, the principal network variables's index (if one is designated) is returned. The principal network variable's index is the index within this functional profile's set of member network variables. Each network variable and configuration property record is obtained separately by calling other access functions documented below.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfGetFPTEx

☑ LCADRF32.DLL ☐ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTEx(PLdrfFileInfo pInfo, TUShort index,
      PUShort pKey, LPSTR pName, PUShort pLength,
      PUByte pResNmSel,  PULong pResNmIndex,
      PUByte pResCmtSel, PULong pResCmtIndex,
      PUByte pManNVs, PUByte pOptNVs,
      PUByte pManCPs, PUByte pOptCPs,
      PUByte pPrincipalNV, PUShort pFlags)
```

Purpose

This function is identical to the **LdrfGetFPT()** function, with the addition of a new **pFlags** parameter. The **pFlags** parameter is reserved for future use and the value is set to zero by the **LdrfGetFPTEx()** function.

This function retrieves a functional profile from an open functional profile template file. The **pInfo** parameter for the file is supplied, along with the index for the functional profile. Each functional profile also has a 16-bit numeric key, and this is returned. This key is used to identify the profile's implementation in the node's SD string. Each functional profile has a programmatic string and two resource string indices, all are returned. The caller must allocate a buffer to hold the string, and pass the length of the buffer through the length reference parameter (C language API only), which will be altered to indicate the length actually read. Four byte values are returned, respectively representing the number of mandatory network variables, optional network variables, mandatory configuration properties, and optional configuration properties in the functional profile. Lastly, the principal network variables's index (if one is designated) is returned. The principal network variable's index is the index within this functional profile's set of member network variables. Each network variable and configuration property record is obtained separately by calling other access functions documented below.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfGetFPTByName

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTByName(PLdrfFileInfo pInfo, LPCSTR name,
            PUShort pIndex, PUShort pKey,
            PUByte pResNmSel,  PULong pResNmIndex,
            PUByte pResCmtSel, PULong pResCmtIndex,
            PUByte pManNVs, PUByte pOptNVs,
            PUByte pManCPs, PUByte pOptCPs,
            PUByte pPrincipalNV)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTByName(long pInfo, BSTR name,
        long *pIndex, long *pKey,
        long *pResNmSel,  long *pResNmIndex,
        long *pResCmtSel, long *pResCmtIndex,
        long *pManNVs, long *pOptNVs,
        long *pManCPs, long *pOptCPs,
        long *pPrincipalNV, long *returnCode)
```

Purpose

This function retrieves a functional profile from an open functional profile template file using the programmatic name key string for the type. The **pInfo** parameter for the file is supplied, along with a pointer to the name string for the functional profile. The index of the functional profile is returned. Each functional profile has two resource string indices, both are returned. Five byte values are returned, respectively representing the number of mandatory network variables, optional network variables, mandatory configuration properties, and optional configuration properties in the functional profile, and the index of the principal network variable.

Return Values

LDRF_ERR_FILE_INFO	The file info structure contents was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found.

LdrfGetFPTByNameEx

☑ LCADRF32.DLL ☐ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTByName(PLdrfFileInfo pInfo, LPCSTR name,
            PUShort pIndex, PUShort pKey,
            PUByte pResNmSel,  PULong pResNmIndex,
            PUByte pResCmtSel, PULong pResCmtIndex,
            PUByte pManNVs, PUByte pOptNVs,
            PUByte pManCPs, PUByte pOptCPs,
            PUByte pPrincipalNV, PUShort pFlags)
```

Purpose

This function is identical to the **LdrfGetFPTByName()** function, with the addition of a new **pFlags** parameter. The **pFlags** parameter is reserved for future use and the value is set to zero by the **LdrfGetFPTByNameEx()** function.

This function retrieves a functional profile from an open functional profile template file using the programmatic name key string for the type. The **pInfo** parameter for the file is supplied, along with a pointer to the name string for the functional profile. The index of the functional profile is returned. Each functional profile has two resource string indices, both are returned. Five byte values are returned, respectively representing the number of mandatory

network variables, optional network variables, mandatory configuration properties, and optional configuration properties in the functional profile, and the index of the principal network variable.

Return Values

LDRF_ERR_FILE_INFO	The file info structure contents was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found.

LdrfGetFPTByKey

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTByKey(PLdrfFileInfo pInfo, TUShort key,
           PUShort pIndex, LPSTR pName, PUShort pLength,
           PUByte pResNmSel,  PULong pResNmIndex,
           PUByte pResCmtSel, PULong pResCmtIndex,
           PUByte pManNVs, PUByte pOptNVs,
           PUByte pManCPs, PUByte pOptCPs,
           PUByte pPrincipalNV)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTByKey(long pInfo, long key,
       long *pIndex, BSTR *pName,
       long *pResNmSel, long *pResNmIndex,
       long *pResCmtSel, long *pResCmtIndex,
       long *pManNVs, long *pOptNVs,
       long *pManCPs, long *pOptCPs,
       long *pPrincipalNV, long *returnCode)
```

Purpose

This function retrieves a functional profile from an open function profile template file using the numeric key for the type. The **pInfo** parameter for the file is supplied, along with the numeric key for the functional profile. The index of the functional profile is returned, as is the programmatic name. Each functional profile has two resource string indices, both are returned. Five byte values are returned, respectively representing the number of mandatory network variables, optional network variables, mandatory configuration properties, and optional configuration properties in the functional profile, and the index of the principal network variable.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found.

LdrfGetFPTByKeyEx

☑ LCADRF32.DLL ☐ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTByKeyEx(PLdrfFileInfo pInfo, TUShort key,
           PUShort pIndex, LPSTR pName, PUShort pLength,
           PUByte pResNmSel,  PULong pResNmIndex,
           PUByte pResCmtSel, PULong pResCmtIndex,
           PUByte pManNVs, PUByte pOptNVs,
           PUByte pManCPs, PUByte pOptCPs,
           PUByte pPrincipalNV, PUShort pFlags)
```

Purpose

This function is identical to the **LdrfGetFPTByKey()** function, with the addition of a new **pFlags** parameter. The **pFlags** parameter is reserved for future use and the value is set to zero by the **LdrfGetFPTByKeyEx()** function.

This function retrieves a functional profile from an open function profile template file using the numeric key for the type. The **pInfo** parameter for the file is supplied, along with the numeric key for the functional profile. The index of the functional profile is returned, as is the programmatic name. Each functional profile has two resource string indices, both are returned. Five byte values are returned, respectively representing the number of mandatory network variables, optional network variables, mandatory configuration properties, and optional configuration properties in the functional profile, and the index of the principal network variable.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found.

LdrfSetFPT

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPT(PLdrfFileInfo pInfo, TUShort index, TUShort key,
      LPCSTR name,
      TUByte resNmSel,  TULong resNmIndex,
      TUByte resCmtSel, TULong resCmtIndex,
      TUByte manNVs, TUByte optNVs,
      TUByte manCPs, TUByte optCPs, TUByte principalNV)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.SetFPT(long pInfo,
     long index, long key, BSTR name,
     long resNmSel,  long resNmIndex,
     long resCmtSel, long resCmtIndex,
     long manNVs, long optNVs, long manCPs, long optCPs,
     long principalNV, long *returnCode)
```

Purpose

This function adds or modifies a functional profile in a functional profile template file that has been opened for editing. The index, numeric key, programmatic name key, and the two resource string selectors and indices must be supplied. Also, counts for mandatory network variables, optional network variables, mandatory configuration properties, and optional configuration properties must all be supplied. If a network variable is to be designated as a principal network variable, its index within the profile's set of member network variables must be specified; otherwise, 0 must be used. The index is used as the key for the functional profile record to change. The numeric key and name keys are checked to make sure they are not a duplicate conflicting with another record. The specific member network variables and configuration properties will be created as blank, and must be added in order in following calls to the access methods documented below.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_DUPLICATE	The name key is already in use by another functional profile in the file.

LdrfSetFPTEx

☑ LCADRF32.DLL ☐ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTEx(PLdrfFileInfo pInfo, TUShort index, TUShort key,
      LPCSTR name,
      TUByte resNmSel,  TULong resNmIndex,
      TUByte resCmtSel, TULong resCmtIndex,
      TUByte manNVs, TUByte optNVs,
      TUByte manCPs, TUByte optCPs, TUByte principalNV, TUShort flags)
```

Purpose

This function is identical to the **LdrfSetFPT()** function, with the addition of a new **flags** parameter. The **flags** parameter is reserved for future use and the value must be set to zero when calling the **LdrfSetFPTEx()** function.

This function adds or modifies a functional profile in a functional profile template file that has been opened for editing. The index, numeric key, programmatic name key, and the two resource string selectors and indices must be supplied. Also, counts for mandatory network variables, optional network variables, mandatory configuration properties, and optional configuration properties must all be supplied. If a network variable is to be designated as a principal network variable, its index within the profile's set of member network variables must be specified; otherwise, 0 must be used. The index is used as the key for the functional profile record to change. The numeric key and name keys are checked to make sure they are not a duplicate conflicting with another record. The specific member network variables and configuration properties will be created as blank, and must be added in order in following calls to the access methods documented below.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_DUPLICATE	The name key is already in use by another functional profile in the file.

LdrfGetFPTNV

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTNV(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort NVindex,
        LPSTR pNVName, PUShort pLength,
        PBool pMandatory,
        PUByte pResNmSel,  PULong pResNmIndex,
        PUByte pResCmtSel, PULong pResCmtIndex,
        PUByte pNVTSel, PUShort pNVTIndex,
        PUByte pDirPollServ, PUShort pByteArrayLen,
        PUByteArray *ppValMin, PUByteArray *ppValMax)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTNV(long pInfo,
       long FPTindex, long NVindex,
       BSTR *pNVName, long *pMandatory,
       long *pResNmSel,  long *pResNmIndex,
       long *pResCmtSel, long *pResCmtIndex,
       long *pNVTSel, long *pNVTIndex,
       long *pDirPollServ, BSTR *pValMin, BSTR *pValMax,
       long *returnCode)
```

Purpose

This function gets a functional profile's network variable member record from a functional profile template file that has been opened. The file info, and the index (starting from 1) of the member network variable must be specified. The network variable programmatic name is returned in a buffer, the length of the buffer must be passed in (for the C language API only), and the length is modified to reflect the number of actual bytes in the name. A Boolean value that indicates whether the network variable is mandatory or optional is returned. Language resource string selectors and indices are returned for the network variable's language-dependent name and an additional info/comment string. The network variable's type selector and index are returned. An encoding of the direction, polledness, and default service type is returned. Two pointers to byte arrays containing the optional min and max overrides of the validation range are returned if the validation range is overridden (otherwise a NULL is returned). The pointer to a length for each byte array precedes the byte array pointers. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them. If extended SNVT-ID support is not enabled, this function will hide standard network variable types whose ID exceeds 250.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.

LdrfGetFPTNVEx

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTNVEx(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort NVindex,
        LPSTR pNVName, PUShort pLength,
        PBool pMandatory,
        PUByte pResNmSel,  PULong pResNmIndex,
        PUByte pResCmtSel, PULong pResCmtIndex,
        PUByte pNVTSel, PUShort pNVTIndex,
        PUByte pDirPollServ, PUShort pByteArrayLen,
        PUByteArray *ppValMin, PUByteArray *ppValMax
        PUByteArray *ppValInvalid)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTNVEx(long pInfo,
       long FPTindex, long NVindex,
       BSTR *pNVName, long *pMandatory,
       long *pResNmSel,  long *pResNmIndex,
       long *pResCmtSel, long *pResCmtIndex,
       long *pNVTSel, long *pNVTIndex,
       long *pDirPollServ, BSTR *pValMin, BSTR *pValMax,
       BSTR *pValInvalid, long *returnCode)
```

Purpose

This function is identical to the **LdrfGetFPTNV()** function except that it returns an additional reference parameter for a byte array containing the default invalid value for the network variable type.

This function gets a functional profile's network variable member record from a functional profile template file that has been opened. The file info, and the index (starting from 1) of the member network variable must be specified. The network variable programmatic name is returned in a buffer, the length of the buffer must be passed in (for the C language API only), and the length is modified to reflect the number of actual bytes in the name. A Boolean value that indicates whether the network variable is mandatory or optional is returned. Language resource string selectors and indices are returned for the network variable's language-dependent name and an additional info/comment string. The network variable's type selector and index are returned. An encoding of the direction, polledness, and default service type is returned. Three pointers to byte arrays containing the optional min and max overrides of the validation range and the optional invalid value are returned if the validation range and invalid value are overridden (otherwise a NULL is returned). The pointer to a length for each byte array precedes the byte array pointers. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them. If extended SNVT-ID support is not enabled, this function will hide standard network variable types whose ID exceeds 250.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.

LdrfGetFPTNVEx2

☑ LCADRF32.DLL ☐ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTNVEx2(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort NVindex,
        LPSTR pNVName, PUShort pLength,
        PBool pMandatory,
        PUByte pResNmSel,  PULong pResNmIndex,
        PUByte pResCmtSel, PULong pResCmtIndex,
        PUByte pNVTSel, PUShort pNVTIndex,
        PUByte pDirPollServ, PUShort pByteArrayLen,
        PUByteArray *ppValMin, PUByteArray *ppValMax
        PUByteArray *ppValInvalid, PUShort pFlags)
```

Purpose

This function is identical to the **LdrfGetFPTNVEx()** function, with the addition of a new **pFlags** parameter. The **pFlags** parameter is reserved for future use and the value is set to zero by the **LdrfGetFPTNVEx2()** function.

This function gets a functional profile's network variable member record from a functional profile template file that has been opened. The file info, and the index (starting from 1) of the member network variable must be specified. The network variable programmatic name is returned in a buffer, the length of the buffer must be passed in (for the C language API only), and the length is modified to reflect the number of actual bytes in the name. A Boolean value that indicates whether the network variable is mandatory or optional is returned. Language resource string selectors and indices are returned for the network variable's language-dependent name and an additional info/comment string. The network variable's type selector and index are returned. An encoding of the direction, polledness, and default service type is returned. Three pointers to byte arrays containing the optional min and max overrides of the validation range and the optional invalid value are returned if the validation range and invalid value are overridden (otherwise a NULL is returned). The pointer to a length for each byte array precedes the byte array pointers. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them. If extended SNVT-ID support is not enabled, this function will hide standard network variable types whose ID exceeds 250.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.

LdrfGetFPTCP

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTCP(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPindex,
        PUShort pAppliesTo,
        LPSTR pCPName, PUShort pLength,
        PBool pMandatory,
        PUByte pResNmSel,  PULong pResNmIndex,
        PUByte pResCmtSel, PULong pResCmtIndex,
        PUByte pCPTSel, PUShort pCPTIndex,
        PUByte pModifyArray,
        PUShort pByteArrayLen, PUByteArray *ppDefault,
        PUByteArray *ppValMin, PUByteArray *ppValMax
        PUBByteArray *ppValInvalid)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTCP(long pInfo, long FPTindex, long CPindex,
       long *pAppliesTo, BSTR *pCPName, long *pMandatory,
       long *pResNmSel,  long *pResNmIndex,
       long *pResCmtSel, long *pResCmtIndex,
       long *pCPTSel, long *pCPTIndex,
       long *pModifyArray, BSTR *pDflt,
       BSTR *pValMin, BSTR *pValMax, long *returnCode)
```

Purpose

This function gets a functional profile's configuration property member record from a functional profile template file that has been opened. The file info, and the index (starting from 1) of the member configuration property must be specified. The **appliesTo** value is returned. The **appliesTo** parameter is set to zero if the configuration property applies to the whole functional block. It is set to the index of the network variable within the functional profile's set of member network variables if the functional profile is defined in an functional profile template file with file format version 1 or 2, and it is set to the this network variable's member number if the profile is defined in an functional profile template file with file format version 3 or better. In the latter case, which supports inheriting functional profiles, you can OR a value of 0x8000 with the member number to indicate that the CP applies to a network variable (by its member number) that is defined in the inherited profile, rather than the one containing the CP.

The configuration property programmatic name is returned in a buffer, the length of the buffer must be passed in (for the C language API only), and the length is modified to reflect the number of actual bytes in the name. A Boolean value that indicates whether the network variable is mandatory or optional is returned. Language resource string selectors and indices are returned for the network variable's language-dependent name and an additional info/comment string. The configuration properties's type selector and index are returned. An encoding of the modification restrictions and array indicator is returned. A pointer to a byte array containing the optional default value is returned, if the default value is given (otherwise NULL is returned). Two pointers to byte arrays containing the optional min and max overrides of the validation range are returned, if the validation range is overridden

(otherwise NULL is returned). The pointer to a length for each byte array precedes the byte array pointers. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.

LdrfGetFPTCPEx

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTCPEx(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPindex,
        PUShort pAppliesTo,
        LPSTR pCPName, PUShort pLength,
        PBool pMandatory,
        PUByte pResNmSel,  PULong pResNmIndex,
        PUByte pResCmtSel, PULong pResCmtIndex,
        PUByte pCPTSel, PUShort pCPTIndex,
        PUByte pModifyArray,
        PUShort pByteArrayLen, PUByteArray *ppDefault,
        PUByteArray *ppValMin, PUByteArray *ppValMax
        PUByteArray *ppValInvalid)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTCPEx(long pInfo, long FPTindex, long CPindex,
       long *pAppliesTo, BSTR *pCPName, long *pMandatory,
       long *pResNmSel,  long *pResNmIndex,
       long *pResCmtSel, long *pResCmtIndex,
       long *pCPTSel, long *pCPTIndex,
       long *pModifyArray, BSTR *pDflt,
       BSTR *pValMin, BSTR *pValMax, BSTR *pValInvalid,
       long *returnCode)
```

Purpose

This function is identical to the **LdrfGetFPTCP()** function except that it returns an additional reference parameter for a byte array containing the default invalid value for the configuration property type.

This function gets a functional profile's configuration property member record from a functional profile template file that has been opened. The file info, and the index (starting from 1) of the member configuration property should be specified. The **appliesTo** value is

returned. The **appliesTo** parameter is set to zero if the configuration property applies to the whole functional block. It is set to the index of the network variable within the functional profile's set of member network variables if the functional profile is defined in an functional profile template file with file format version 1 or 2, and it is set to the this network variable's member number if the profile is defined in an functional profile template file with file format version 3 or better. In the latter case, which supports inheriting functional profiles, you can OR a value of 0x8000 with the member number to indicate that the CP applies to a network variable (by its member number) that is defined in the inherited profile, rather than the one containing the CP.

The configuration property programmatic name is returned in a buffer, the length of the buffer must be passed in (for the C language API only), and the length is modified to reflect the number of actual bytes in the name. A Boolean value that indicates whether the network variable is mandatory or optional is returned. Language resource string selectors and indices are returned for the network variable's language-dependent name and an additional info/comment string. The configuration properties's type selector and index are returned. An encoding of the modification restrictions and array indicator is returned. A pointer to a byte array containing the optional default value is returned if the default value is given (otherwise NULL is returned). Three pointers to byte arrays containing the optional min and max overrides of the validation range and the optional invalid value are returned if the validation range and invalid value are overridden (otherwise NULL is returned). The pointer to a length for each byte array precedes the byte array pointers. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.

LdrfGetFPTCPEx2

☑ LCADRF32.DLL ☐ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTCPEx2(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPindex,
        PUShort pAppliesTo,
        LPSTR pCPName, PUShort pLength,
        PBool pMandatory,
        PUByte pResNmSel,  PULong pResNmIndex,
        PUByte pResCmtSel, PULong pResCmtIndex,
        PUByte pCPTSel, PUShort pCPTIndex,
        PUByte pModifyArray,
        PUShort pByteArrayLen, PUByteArray *ppDefault,
        PUByteArray *ppValMin, PUByteArray *ppValMax
        PUByteArray *ppValInvalid, PUShort pFlags)
```

Purpose

This function is identical to the **LdrfGetFPTCPEx()** function, with the addition of a new **pFlags** parameter. The **pFlags** parameter is reserved for future use and the value is set to zero by the **LdrfGetFPTCPEx2()** function.

This function gets a functional profile's configuration property member record from a functional profile template file that has been opened. The file info, and the index (starting from 1) of the member configuration property should be specified. The **appliesTo** value is returned. The **appliesTo** parameter is set to zero if the configuration property applies to the whole functional block. It is set to the index of the network variable within the functional profile's set of member network variables if the functional profile is defined in an functional profile template file with file format version 1 or 2, and it is set to the this network variable's member number if the profile is defined in an functional profile template file with file format version 3 or better. In the latter case, which supports inheriting functional profiles, you can OR a value of 0x8000 with the member number to indicate that the CP applies to a network variable (by its member number) that is defined in the inherited profile, rather than the one containing the CP.

The configuration property programmatic name is returned in a buffer, the length of the buffer must be passed in (for the C language API only), and the length is modified to reflect the number of actual bytes in the name. A Boolean value that indicates whether the network variable is mandatory or optional is returned. Language resource string selectors and indices are returned for the network variable's language-dependent name and an additional info/comment string. The configuration properties's type selector and index are returned. An encoding of the modification restrictions and array indicator is returned. A pointer to a byte array containing the optional default value is returned if the default value is given (otherwise NULL is returned). Three pointers to byte arrays containing the optional min and max overrides of the validation range and the optional invalid value are returned if the validation range and invalid value are overridden (otherwise NULL is returned). The pointer to a length for each byte array precedes the byte array pointers. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.

LdrfGetFPTNVMemberNumber

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTNVMemberNumber(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort NVindex,
        PUSHORT pNVMemberNumber)
```

COM Interface Prototype

```
LdrfGeneral2.GetFPTNVMemberNumber(long pInfo,
       long FPTindex, long NVindex,
       long *pNVMemberNumber, long *returnCode)
```

Purpose

This function gets a functional profile's network variable's member number. The file info, functional profile template index, and the network variable index (starting from 1) of the member network variable must be specified.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that index was found.

LdrfGetFPTCPMemberNumber

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTCPMemberNumber(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPindex,
        PUSHORT pCPMemberNumber)
```

COM Interface Prototype

```
LdrfGeneral2.GetFPTCPMemberNumber(long pInfo,
       long FPTindex, long CPindex,
       long *pCPMemberNumber, long *returnCode)
```

Purpose

This function gets a functional profile's configuration properties's member number. The file info, functional profile template index, and the configuration property index (starting from 1) of the member configuration property must be specified.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that index was found.

LdrfGetFPTNVIndex

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTNVIndex(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort NVMemberNumber,
        PUSHORT pNVIndex)
```

COM Interface Prototype

```
LdrfGeneral2.GetFPTNVIndex(long pInfo,
       long FPTindex, long NVMemberNumber,
       long *pNVIndex, long *returnCode)
```

Purpose

This function gets a functional profile's network variable's index, given the member number. The file info, functional profile template index, and the network variable member number of the network variable must be specified.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that index was found.

LdrfGetFPTCPIndex

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTCPIndex(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPMemberNumber,
        PUSHORT pCPIndex)
```

COM Interface Prototype

```
LdrfGeneral2.GetFPTCPIndex(long pInfo,
       long FPTindex, long CPMemberNumber,
       long *pCPIndex, long *returnCode)
```

Purpose

This function get a functional profile's configuration property's index, given the member number. The file info, functional profile template index, and the configuration property member number of the configuration property must be specified.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, like for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that index was found.

LdrfGetFPTCPByAttributes

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTCPByAttributes(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort appliesTo,
        TUByte CPTSel, TUShort CPTIndex,
        PUShort pCPindex,
        LPSTR pCPName, PUShort pLength,
        PBool pMandatory,
        PUByte pResNmSel,  PULong pResNmIndex,
        PUByte pResCmtSel, PULong pResCmtIndex,
        PUByte pCPTSel, PUShort pCPTIndex,
        PUByte pModifyArray,
        PUShort pByteArrayLen, PUByteArray *ppDefault,
        PUByteArray *ppValMin, PUByteArray *ppValMax)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTCPByAttributes(long pInfo,
                long FPTindex, long appliesTo,
                long CPTSel, long CPTIndex, long *pCPIndex,
                BSTR *pCPName, long *pMandatory,
                long *pResNmSel,  long *pResNmIndex,
                long *pResCmtSel, long *pResCmtIndex,
                long *pModifyArray, BSTR *pDflt,
                BSTR *pValMin, BSTR *pValMax,
                long *returnCode)
```

Purpose

This function gest a functional profile's configuration property member record from a functional profile template file that has been opened, using a matching algorithm rather than the index of the configuration property. The file info, and the type (selector and index) and the **appliesTo** value of the member configuration property must be specified. Each configuration property either is declared to apply to a particular network variable, or to the object as a whole. A configuration that applies to the object uses an **appliesTo** value of 0. Otherwise, the index of the network variable is used. The matching algorithm treats a configuration property that applies to the principal network variable as synonymous with a configuration property that applies to the object as a whole. (For example, if the principal network variable is index 1, and you are looking for a configuration property whose type is <s>:<i> but you are not sure whether the configuration property applies to the object or to the principal network variable—if you pass these values in to this routine, using either **appliesTo**=0 or **appliesTo**=1, the configuration property will be found.)

The configuration property index is returned. The programmatic name is returned in a buffer, the length of the buffer must be passed in (for the C language API only), and the length is modified to reflect the number of actual bytes in the name. A Boolean value that indicates whether the network variable is mandatory or optional is returned. Language resource string selectors and indices are returned for the network variable's language-dependent name and an additional info/comment string. An encoding of the modification restrictions and array indicator is returned. A pointer to a byte array containing the optional default value is returned if the default value is given (otherwise NULL is returned). Two pointers to byte arrays containing the optional min and max overrides of the validation range are returned if the validation range is overridden (otherwise NULL is returned). The pointer to a length for each byte array precedes the byte array pointers. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.

LdrfGetFPTCPByAttributesEx

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTCPByAttributesEx(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort appliesTo,
        TUByte CPTSel, TUShort CPTIndex,
        PUShort pCPindex,
        LPSTR pCPName, PUShort pLength,
        PBool pMandatory,
        PUByte pResNmSel,  PULong pResNmIndex,
        PUByte pResCmtSel, PULong pResCmtIndex,
        PUByte pCPTSel, PUShort pCPTIndex,
        PUByte pModifyArray,
        PUShort pByteArrayLen, PUByteArray *ppDefault,
        PUByteArray *ppValMin, PUByteArray *ppValMax,
        PUByteArray *ppInvalid)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTCPByAttributesEx(long pInfo,
                long FPTindex, long appliesTo,
                long CPTSel, long CPTIndex, long *pCPIndex,
                BSTR *pCPName, long *pMandatory,
                long *pResNmSel,  long *pResNmIndex,
                long *pResCmtSel, long *pResCmtIndex,
                long *pModifyArray, BSTR *pDflt,
                BSTR *pValMin, BSTR *pValMax, BSTR *pValInvalid,
                long *returnCode)
```

Purpose

This function is identical to the **LdrfFPTCPByAttributes()** function except that it returns an additional reference parameter for a byte array containing the default invalid value for the configuration property type.

This function gets a functional profile's configuration property member record from a functional profile template file that has been opened, using a matching algorithm rather than the index of the configuration property. The file info, and the type (selector and index) and the **appliesTo** value of the member configuration property must be specified. Each configuration property either is declared to apply to a particular network variable, or to the object as a whole. A configuration that applies to the object uses an **appliesTo** value of 0. Otherwise, the index of the network variable is used. The matching algorithm treats a configuration property that applies to the principal network variable as synonymous with a configuration property that applies to the object as a whole. (For example, if the principal network variable is index 1, and you are looking for a configuration property whose type is <*s*>**:**<*i*> but you are not sure whether the configuration property applies to the object or to the principal network variable—if you pass these values in to this function, using either **appliesTo**=0 or **appliesTo**=1, the configuration property will be found.)

The configuration property index is returned. The programmatic name is returned in a buffer, the length of the buffer must be passed in (for the C language API only), and the

length is modified to reflect the number of actual bytes in the name. A Boolean value that indicates whether the network variable is mandatory or optional is returned. Language resource string selectors and indices are returned for the network variable's language-dependent name and an additional info/comment string. An encoding of the modification restrictions and array indicator is returned. A pointer to a byte array containing the optional default value is returned if the default value is given (otherwise NULL is returned). Three pointers to byte arrays containing the optional min and max overrides of the validation range and the optional invalid value are returned if the validation range and invalid value are overridden (otherwise NULL is returned). The pointer to a length for each byte array precedes the byte array pointers. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that index was found.
LDRF_ERR_TRUNC	The string was truncated to fit the buffer.

LdrfSetFPTNV

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTNV(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort NVMemberNumber,
        LPCSTR NVName, TBool mandatory,
        TUByte resNmSel,  TULong resNmIndex,
        TUByte resCmtSel, TULong resCmtIndex,
        TUByte NVTSel, TUShort NVTIndex,
        TUByte dirPollServ, TUShort byteArrayLen,
        PUByteArray pValMin, PUByteArray pValMax,
        PUByteArray pValInvalid)
```

COM Interface Prototype

```
LdrfFuncProf.SetFPTNV(long pInfo,
       long FPTindex, long NVMemberNumber,
       BSTR NVName, long mandatory,
       long resNmSel,  long resNmIndex,
       long resCmtSel, long resCmtIndex,
       long NVTSel, long NVTIndex,
       long dirPollServ, BSTR valMin, BSTR valMax,
       long *returnCode)
```

Purpose

This function modifies a functional profile's network variable member record in a functional profile template file that has been opened for editing. The file info, and the member number

(starting from 1) of the member network variable must be specified, and the network variable programmatic name, and resource string selectors and indices must all be specified. After the programmatic name parameter is the Boolean **mandatory** parameter, indicating whether the network variable is a mandatory or optional part of the functional profile.

The network variable's type selector and index are also supplied, as is an encoding of the direction, polledness, and default service type. Two pointers to byte arrays containing the optional min and max overrides of the validation range are supplied if the validation range is overridden (otherwise NULL is supplied). A byte array length is also passed in. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that member number was found.
LDRF_ERR_DUPLICATE	The name key is already in use by another network variable in the functional profile.

LdrfSetFPTNVEx

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTNVEx(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort NVMemberNumber,
        LPCSTR NVName, TBool mandatory,
        TUByte resNmSel,  TULong resNmIndex,
        TUByte resCmtSel, TULong resCmtIndex,
        TUByte NVTSel, TUShort NVTIndex,
        TUByte dirPollServ, TUShort byteArrayLen,
        PUByteArray pValMin, PUByteArray pValMax
        PUByteArray pValInvalid)
```

COM Interface Prototype

```
LdrfFuncProf.SetFPTNVEx(long pInfo,
       long FPTindex, long NVMemberNumber,
       BSTR NVName, long mandatory,
       long resNmSel,  long resNmIndex,
       long resCmtSel, long resCmtIndex,
       long NVTSel, long NVTIndex,
       long dirPollServ, BSTR valMin, BSTR valMax, BSTR ValInvalid,
       long *returnCode)
```

Purpose

This function is identical to the **LdrfSetFPTNV()** function except that it takes an additional pointer to a byte array containing the default invalid value for the network variable type.

This function modifies a functional profile's network variable member record in a functional profile template file that has been opened for editing. The file info, and the index (starting from 1) of the member network variable must be specified, and the network variable programmatic name, and resource string selectors and indices must all be specified. After the programmatic name parameter is the **mandatory** Boolean parameter, indicating whether the network variable is a mandatory or optional part of the functional profile.

The network variable's type selector and index are also supplied, as is an encoding of the direction, polledness, and default service type. Three pointers to byte arrays containing the optional min and max overrides of the validation range and the optional invalid value are supplied if the validation range or invalid value are overridden (otherwise NULL is supplied). A byte array length is also passed in. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that index was found.
LDRF_ERR_DUPLICATE	The name key is already in use by another network variable in the functional profile.

LdrfSetFPTNVEx2

☑ LCADRF32.DLL ☐ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTNVEx2(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort NVMemberNumber,
        LPCSTR NVName, TBool mandatory,
        TUByte resNmSel,  TULong resNmIndex,
        TUByte resCmtSel, TULong resCmtIndex,
        TUByte NVTSel, TUShort NVTIndex,
        TUByte dirPollServ, TUShort byteArrayLen,
        PUByteArray pValMin, PUByteArray pValMax
        PUByteArray pValInvalid, TUShort flags)
```

Purpose

This function is identical to the **LdrfSetFPTNVEx()** function, with the addition of a new **flags** parameter. The **flags** parameter is reserved for future use and the value must be set to zero when calling the **LdrfSetFPTNVEx 2()** function.

This function modifies a functional profile's network variable member record in a functional profile template file that has been opened for editing. The file info, and the index (starting from 1) of the member network variable must be specified, and the network variable programmatic name, and resource string selectors and indices must all be specified. After the programmatic name parameter is the **mandatory** Boolean parameter, indicating whether the network variable is a mandatory or optional part of the functional profile.

The network variable's type selector and index are also supplied, as is an encoding of the direction, polledness, and default service type. Three pointers to byte arrays containing the optional min and max overrides of the validation range and the optional invalid value are supplied if the validation range or invalid value are overridden (otherwise NULL is supplied). A byte array length is also passed in. Your application must call the **LdrfFreeByteArray()** function to free the returned byte array pointers when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that index was found.
LDRF_ERR_DUPLICATE	The name key is already in use by another network variable in the functional profile.

LdrfChangeFPTNVMemberNumber

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfChangeFPTNVMemberNumber(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort oldMemberNumber,
        TUShort newMemberNumber)
```

COM Interface Prototype

```
LdrfGeneral2.Change FPTNVMemberNumber(long *pInfo,
       long FPTindex, long oldMemberNumber,
       long newMemberNumber, long *returnCode)
```

Purpose

This function changes the member number of a network variable in a functional profile. The member number must be unique; attempting to duplicate an existing member number returns **LDRF_ERR_DUPLICATE**.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no network variable member of that member number was found.
LDRF_ERR_DUPLICATE	An attempt was made to duplicate an existing member number.

LdrfSetFPTCP

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTCP(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPMemberNumber,
        TUShort appliesTo, LPCSTR CPName, TBool mandatory,
        TUByte resNmSel,  TULong resNmIndex,
        TUByte resCmtSel, TULong resCmtIndex,
        TUByte CPTSel, TUShort CPTIndex,
        TUByte modifyArray,
        TUShort byteArrayLen, PUByteArray pDefault,
        PUByteArray pValMin, PUByteArray pValMax
        PUByteArray pValInvalid)
```

COM Interface Prototype

```
LdrfFuncProf.SetFPTCP(long pInfo, long FPTindex,
       long CPMemberNumber,
       long appliesTo, BSTR CPName, long mandatory,
       long resNmSel,  long resNmIndex,
       long resCmtSel, long resCmtIndex,
       long CPTSel, long CPTIndex,
       long modifyArray, BSTR dflt,
       BSTR valMin, BSTR valMax, long *returnCode)
```

Purpose

This function modifies a functional profile's configuration property member record in a functional profile template file that has been opened for editing. The file info, and the member number (starting from 1) of the member configuration property must be specified. The **appliesTo** parameter is set to zero if the configuration property applies to the whole functional block. It is set to the index of the network variable within the functional profile's member network variables if the functional profile is defined in an functional profile

template file with file format version 1 or 2, and it is set to this network variable's member number if the profile is defined in an functional profile template file with file format version 3 or better. In the latter case, which supports inheriting functional profiles, you can OR a value of 0x8000 with the member number to indicate that the CP applies to a network variable (by its member number) that is defined in the inherited profile, rather than the one containing the CP. The configuration property programmatic name and resource string selectors and indices must all be specified. After the programmatic name parameter is a **mandatory** Boolean parameter that indicates whether the configuration property is mandatory or optional in the functional profile.

The configuration property's type selector and index are supplied, as is an encoding of the modification restrictions and array indication. A pointer to a byte array containing the optional default value is supplied if desired, otherwise NULL is supplied. Two pointers to byte arrays containing the optional min and max overrides of the validation range are supplied, if the validation range is overridden (otherwise NULL is supplied). A byte array length is also passed in. Your application must call the **LdrfFreeByteArray()** function to free the returned byte arrays when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that member number was found.
LDRF_ERR_DUPLICATE	The name key is already in use by another configuration property in the functional profile.

LdrfSetFPTCPEx

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTCPEx(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPMemberNumber,
        TUShort appliesTo, LPCSTR CPName, TBool mandatory,
        TUByte resNmSel,  TULong resNmIndex,
        TUByte resCmtSel, TULong resCmtIndex,
        TUByte CPTSel, TUShort CPTIndex,
        TUByte modifyArray,
        TUShort byteArrayLen, PUByteArray pDefault,
        PUByteArray pValMin, PUByteArray pValMax
        PUByteArray pValInvalid)
```

COM Interface Prototype

```
LdrfFuncProf.SetFPTCPEx(long pInfo, long FPTindex,
       long CPMemberNumber,
       long appliesTo, BSTR CPName, long mandatory,
       long resNmSel,  long resNmIndex,
       long resCmtSel, long resCmtIndex,
       long CPTSel, long CPTIndex,
       long modifyArray, BSTR dflt,
       BSTR ValMin, BSTR ValMax, BSTR ValInvalid,
       long *returnCode)
```

Purpose

This function is identical to the **LdrfSetFPTCP()** function except that it takes an additional pointer to a byte array containing the default invalid value for the configuration property type.

This function modifies a functional profile's configuration property member record in a functional profile template file that has been opened for editing. The file info, and the index (starting from 1) of the member configuration property must be specified. The **appliesTo** parameter is set to zero if the configuration property applies to the whole functional block. It is set to the index of the network variable within the functional profile's member network variables if the functional profile is defined in an functional profile template file with file format version 1 or 2, and it is set to the this network variable's member number if the profile is defined in an functional profile template file with file format version 3 or better. In the latter case, which supports inheriting functional profiles, you can OR a value of 0x8000 with the member number to indicate that the CP applies to a network variable (by its member number) that is defined in the inherited profile, rather than the one containing the CP. The configuration property programmatic name and resource string selectors and indices must all be specified. After the programmatic name parameter is a **mandatory** Boolean parameter that indicates whether the configuration property is mandatory or optional in the functional profile.

The configuration property's type selector and index are supplied, as is an encoding of the modification restrictions and array indication. A pointer to a byte array containing the optional default value is supplied if desired, else NULL is supplied. Three pointers to byte arrays containing the optional min and max overrides of the validation range and the optional invalid value are supplied if the validation range or invalid value are overridden (otherwise NULL is supplied). A byte array length is also passed in. Your application must call the **LdrfFreeByteArray()** function to free the returned byte arrays when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that index was found.
LDRF_ERR_DUPLICATE	The name key is already in use by another configuration property in the functional profile.

LdrfSetFPTCPEx2

☑ LCADRF32.DLL ☐ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTCPEx(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPMemberNumber,
        TUShort appliesTo, LPCSTR CPName, TBool mandatory,
        TUByte resNmSel,  TULong resNmIndex,
        TUByte resCmtSel, TULong resCmtIndex,
        TUByte CPTSel, TUShort CPTIndex,
        TUByte modifyArray,
        TUShort byteArrayLen, PUByteArray pDefault,
        PUByteArray pValMin, PUByteArray pValMax
        PUByteArray pValInvalid, TUShort flags)
```

Purpose

This function is identical to the **LdrfSetFPTCPEx()** function, with the addition of a new **flags** parameter. The **flags** parameter is reserved for future use and the value must be set to zero when calling the **LdrfSetFPTCPEx 2()** function.

This function modifies a functional profile's configuration property member record in a functional profile template file that has been opened for editing. The file info, and the index (starting from 1) of the member configuration property must be specified. The **appliesTo** parameter is set to zero if the configuration property applies to the whole functional block. It is set to the index of the network variable within the functional profile's member network variables if the functional profile is defined in an functional profile template file with file format version 1 or 2, and it is set to the this network variable's member number if the profile is defined in an functional profile template file with file format version 3 or better. In the latter case, which supports inheriting functional profiles, you can OR a value of 0x8000 with the member number to indicate that the CP applies to a network variable (by its member number) that is defined in the inherited profile, rather than the one containing the CP. The configuration property programmatic name and resource string selectors and indices must all be specified. After the programmatic name parameter is a **mandatory** Boolean parameter that indicates whether the configuration property is mandatory or optional in the functional profile.

The configuration property's type selector and index are supplied, as is an encoding of the modification restrictions and array indication. A pointer to a byte array containing the optional default value is supplied if desired, else NULL is supplied. Three pointers to byte arrays containing the optional min and max overrides of the validation range and the optional invalid value are supplied if the validation range or invalid value are overridden (otherwise NULL is supplied). A byte array length is also passed in. Your application must call the **LdrfFreeByteArray()** function to free the returned byte arrays when your application is done with them.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.

LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that index was found.
LDRF_ERR_DUPLICATE	The name key is already in use by another configuration property in the functional profile.

LdrfChangeFPTCPMemberNumber

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfChangeFPTCPMemberNumber(PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort oldMemberNumber,
        TUShort newMemberNumber)
```

COM Interface Prototype

```
LdrfGeneral2.ChangeFPTCPMemberNumber(long *pInfo,
       long FPTindex, long oldMemberNumber,
       long newMemberNumber, long *returnCode)
```

Purpose

This function changes the member number of a configuration property in a functional profile. The member number must be unique; attempting to duplicate an existing member number returns **LDRF_ERR_DUPLICATE**.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that member number was found.
LDRF_ERR_DUPLICATE	An attempt was made to duplicate an existing member number.

LdrfSetFPTCPArrayDetails

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTCPArrayDetails (PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPIndex,
        TUShort minArraySize, TUShort maxArraySize)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.SetFPTCPArrayDetails (long pInfo,
       long FPTindex, long CPIndex, long minArraySize
       long maxArraySize, long *returnCode)
```

Purpose

This function sets the minimum and maximum size of the specified configuration property array for a functional profile. This function is only available on version 4 type files and later.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that member number was found.
LDRF_ERR_VERSION	The function was called on a pre-version 4 type file.

LdrfGetFPTCPArrayDetails

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTCPArrayDetails (PLdrfFileInfo pInfo,
        TUShort FPTindex, TUShort CPIndex,
        PUShort minArraySize, PUShort maxArraySize,
        Pbool pDetailsAreDefaults)
```

COM Interface Prototype

```
LdrfFuncProfTmplt.GetFPTCPArrayDetails (long pInfo,
       long FPTindex, long CPIndex, long *pMinArraySize
       long *pMaxArraySize, long *pDetailsAreDefaults,
       long *returnCode)
```

Purpose

This function gets the minimum and maximum size of the specified configuration property array for a functional profile. This function is only available on version 4 type files and later. If the values obtained are the automatically generated default values, **pDetailsAreDefaults** will be set toTrue. If the values are obtained from the type file, this value is set to False.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if it was a valid functional profile index, then no configuration property member of that member number was found.
LDRF_ERR_VERSION	The function was called on a pre-version 4 type file.

LdrfGetFPTInherit

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfGetFPTInherit (     PLdrfFileInfo pInfo,
                        PUSHORT pInherit,
                        TUSHORT index)
```

COM Interface Prototype

```
LdrfGeneral2.GetFPTInherit(   long pInfo,
                                    long index,
                                    long *pInherit,
                                    long *returnCode)
```

Purpose

This function gets the value of the functional profile Inherit flag. If the functional profile template file version does not support inheritance, a value of FALSE is returned.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	The file has not been completely created.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfSetFPTInherit

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTInherit (     PLdrfFileInfo pInfo,
                        TUSHORT index)
```

COM Interface Prototype

```
LdrfGeneral2.SetFPTInherit(   long pInfo,
                                    long index,
                                    long *returnCode)
```

Purpose

This function sets the functional profile Inherit flag. By default, this flag is not set.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	The file has not been completely created.
LDRF_ERR_FMT_VERSION	The format version of the resource file does not support inheritance.

LdrfClearFPTInherit

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfClearFPTInherit (   PLdrfFileInfo pInfo,
                        TUSHORT index)
```

COM Interface Prototype

```
LdrfGeneral2.ClearFPTInherit( long pInfo,
                                  long index,
                                  long *returnCode)
```

Purpose

This function is called to clear the functional profile Inherit flag. By default, this flag is not set.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_INCOMPLETE	The file has not been completely created.

LdrfSetFPTObsolete

☑ LCADRF32.DLL ☑ COM Interface ☐ LDRF32R.DLL

C Language API

```
LdrfSetFPTObsolete(PLdrfFileInfo pInfo, TUShort index)
```

COM Interface Prototype

```
LdrfGeneral2.SetFPTObsolete(long pInfo, long index, long *returnCode)
```

Purpose

This function marks the specified functional profile as obsolete. Marking a functional profile as obsolete does not affect the processing of the file. It is up to the calling applicatin to interpret the obsolete mark. To edit a functional profile and leave the obsolete mark intact, you must check for the mark using **LdrfGetFPTObsolete()** before making any changes and call this function after you are done editing.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NO_ACCESS	The file wasn't opened in edit mode.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found, or if adding, the new index is not correct (must be contiguous).
LDRF_ERR_FMT_VERSION	The resource file format does not support the obsolete mark. The obsolete mark is supported in functional profile template files of version 3 or later.

LdrfGetFPTObsolete

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetFPTObsolete(PLdrfFileInfo pInfo, TUShort index, PboolByte
                   pObsolete)
```

COM Interface Prototype

```
LdrfGeneral2.GetFPTObsolete(long pInfo, long index,
                        long *pObsolete, long *returnCode)
```

Purpose

This function checks for the obsolete flag on the specified functional profile. Marking a functional profile as obsolete does not affect the processing of the file. It is up to the calling application to interpret the obsolete mark. This function will return FALSE if called on a resource file that does not support the obsolete mark; no error will be returned.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No functional profile template with that index was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_EMPTY_RECORD	The function attempted to access an empty record. This error is only returned if the **LdrfEnableEmptyEntries()** function has been called.

LdrfFindEmptyFPT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfFindEmptyFPT(PLdrfFileInfo pInfo, PUShort pIndex)
```

COM Interface Prototype

```
LdrfMiscFns1.FindEmptyFPT(long pInfo, long *pIndex,
                                  long *returnCode)
```

Purpose

This function returns the first empty functional profile index. If there are no empty functional profile records, this function returns *n*+1, where *n* is the number of functional profile records in the file.

Return Values

LDRF_ERR_INTERNAL	Internal error.
LDRF_ERR_NOT_FOUND	No empty record index is available (only occurs if file is full).

LdrfDeleteFPT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfDeleteFPT(PLdrfFileInfo pInfo, TUShort index)
```

COM Interface Prototype

```
LdrfMiscFns1.DeleteFPT (long pInfo, long index, long *returnCode)
```

Purpose

This function deletes a functional profile template. Deleted resources do not consume any file data space. They only have NULL entries in the resource key-access directories.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_NOT_FOUND	No functional profile with that index was found.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.

LdrfValidateFPT

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfValidateFPT(PLdrfFileInfo pInfo, TUShort Index)
```

COM Interface Prototype

```
LdrfMiscFns1.ValidateFPT(long pInfo, long index, long *returnCode)
```

Purpose

This function returns the status of a functional profile. See *Return Values* for more information.

Return Values

LDRF_ERR_PARAM	Incorrect parameters supplied.
LDRF_ERR_NOT_FOUND	The specified functional profile was not found.
LDRF_ERR_INTERNAL	Internal error.
LDRF_ERR_NONE	The functional profile was found and is not empty.
LDRF_ERR_EMPTY_RECORD	The functional profile is an empty record (i.e. it was deleted). This error code is only be returned if the **LdrfEnableEmptyEntries()** function was called on the type file.

LONMARK Resource File API COM Interface

The LONMARK Resource File API COM interface provides a standard Windows COM interface to the resource file API functions. The COM interface provides a programming language interface to the API functions on Win32 platforms. The functions are described in the API reference; this topic describes general attributes of the functions. Windows applications can use either the C-language interface or the language-independent COM interface. Both interfaces offer the same functionality.

The COM interfaces are all named to make matching with the ANSI C API interface easy, adding the COM interface object and deleting the initial "Ldrf" portion of the function names. For example, the **LdrfCheckHeaderCRC()** function has a corresponding COM interface object and method named **LdrfCRC.CheckHeaderCRC**.

There is also a correspondence between the parameter lists of the ANSI C interfaces and the parameter lists of the COM interfaces. The ANSI C functions all return an error code chosen from an enumeration. Since the COM interfaces all return a value dealing with the COM interface itself, the error code return value is returned via an additional parameter at the end of the parameter list of the COM interface. This additional parameter is a pointer to a long value that contains the return value after the COM interface function returns. For example, the following interfaces describe the ANSI C and COM interfaces, respectively, for the **LdrfCheckHeaderCRC()** function:

ANSI C: `LdrfCheckHeaderCRC (PLdrfFileInfo pInfo)`

COM: `CheckHeaderCRC (long pInfo, long *pReturnCode)`

All other interface parameters are as similar as possible in number, order, name, and meaning between the ANSI C functional interfaces and the COM interfaces. The COM interface only allows certain types as described in the API reference. Following is a summary of the key differences:

- The ANSI C **LPCSTR** and **LPSTR** types are replaced with the COM **BSTR** type. The **BSTR** type is a string of 16-bit characters, one character per 16-bit word. **BSTR** arguments to be passed into a procedure must be allocated and freed by the caller. **BSTR** arguments returned by a procedure are allocated using **SysAllocString()** by the callee, and must be freed by the caller using **SysFreeString()**.
- Writable strings in the API function arguments are accompanied by length parameters. In the COM interfaces, these length parameters are unnecessary and do not exist.
- All integer values such as **TULong**, **TUShort**, and **TBool** are converted to and from the long data type in the COM interface.
- The pointer to the **PLdrfFileInfo** file information structure and the pointer to the **PLdrfTypeTree *** type tree structure are converted to and from the long data type in the COM interface.
- Pointers to byte arrays are passed as **BSTR**s containing the hex encoding of the bytes. For example, a three byte string containing '0x01,0x02,0x03' becomes the following six character **BSTR**: "010203". On the input side, the COM interfaces support optional spaces between each byte. For example, the "01 02 03" **BSTR** would be equivalent to the previous BSTR
  - On the output side, the COM interfaces return strings with the bytes separated by spaces.

Utility Functions

You can use the utility functions discussed in this section with many LONMARK Resource File API operations, but they are not required in the general case. For example, you can use the **LdrfCheckHeaderCRC()** function to enforce an explicit CRC check, but the API itself automatically verifies any checksums at the appropriate moments (for example, when opening a type file).

LdrfCheckHeaderCRC

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCheckHeaderCRC(PLdrfFileInfo pInfo)
```

COM Interface Prototype

```
LdrfGeneral.CheckHeaderCRC(long pInfo, long *returnCode)
```

Purpose

This function checks the CRC for a resource file header. The resource file must be open. The **ldrfFileInfo** structure pointer is passed in.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_CRC	The file header did not pass the CRC check.

LdrfCheckDataCRC

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCheckDataCRC(PLdrfFileInfo pInfo)
```

Purpose

This function checks the CRC for resource file data. The resource file must be open. The **ldrfFileInfo** structure pointer is passed in.

Return Values

LDRF_ERR_FILE_INFO	The file info structure content was not valid.
LDRF_ERR_SYS	System error, for example due to exceeding available file handles, disk space, or memory.
LDRF_ERR_CRC	The file data did not pass the CRC check.

LdrfCheckCRC

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfCheckCRC(PUByteArray pBlock, TULong size, TUShort oldCRC)
```

COM Interface Prototype

```
LdrfGeneral.CheckCRC(BSTR pBlock, long oldCRC, long *returnCode)
```

Purpose

This function computes the CRC on a block of data and compares it to a known CRC value. The starting point of the block of bytes and the size of the block of bytes (as a number of bytes) is passed in. You can use this function to verify that an in memory copy of a resource file is valid. To use this function call the function with a pointer to the first byte following the data CRC, and specify the size of the data block (subtracting the two bytes for the CRC itself).

Return Values

LDRF_ERR_CRC	The data did not pass the CRC check.

LdrfGetDRFAPIErrorString

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetDRFAPIErrorString(TldrfErrCodes errCode, LPSTR pErrorString,
                         PUShort pLength)
```

COM Interface Prototype

```
LdrfMistFns1.GetDRFAPIErrorString(long errCode, BSTR *pErrorString,
long *returnCode)
```

Purpose

This function returns the string associated with a specified LONMARK Resource File API error code. Set **pErrorLength** to the length of the string buffer available at **pErrorString** (for the C language API only); **pErrorLength** will be set to the actuall length of the error string when the string is returned. If the value passed in has no associated error message, **pString** will be set to "Unknown error code [DRF#<error number>]".

Alphanumeric error descriptions returned by this function are not localized. Error descriptions are always provided in (US) English.

Return Values

LDRF_ERR_TRUNC	The error code string is longer than the size specified by **pErrorLength**. The error code is truncated to fit.
LDRF_ERR_PARAM	An invalid parameter was specified.

LdrfGetDRFAPIVersion

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfGetDRFAPIVersion(PUShort pMajor, PUShort pMinor, PUShort pFix);
```

COM Interface Prototype

```
LdrfMiscFns1.GetDRFAPIVersion(long *pMajor, long *pMinor,
long *pFix, long *returnCode)
```

Purpose

This function returns the version number of the LONMARK Resource File API. A LONMARK Resource File API version # might be "2.21.03", where the "2" is the major version number, ".21" is the minor version number, and "03" is the build number. The minor version number is incremented when minor improvements are made to capability or interface. The minor version number may be incremented to the next ten (for example from ".10" or ".11" to ".20") when the improvement is "more significant". The build number is reset to "00" when the minor or major number is incremented, and the build number is incremented when there is any modification to the code base.

Return Values

LDRF_ERR_NONE

LdrfSupportedFormats

☑ LCADRF32.DLL ☑ COM Interface ☑ LDRF32R.DLL

C Language API

```
LdrfSupportedFormats(TLdrfFileType fileType,
                     PUByte pMajFmtLow, PUByte pMajFmtHigh)
```

COM Interface Prototype

```
LdrfGeneral2.SupportedFormats(TLdrfFileType fileType,
                     long *pMajFmtLow, long *pMajFmtHigh,
                     long *returnCode)
```

Purpose

For a given file type, this function returns the low and high supported version numbers, indicating the range of format-versions this LONMARK Resource File API supports for the requested file type.

Return Values

LDRF_ERR_CRC	The data did not pass the CRC check.