LANSA Open Guide

- Introduction
- Getting Started
- Function Overview
- Guidelines for Using LANSA Open
- Quick Start for Visual Basic Programmers
- Function Details
- Troubleshooting
- Appendix A. Error Message Codes

Edition Date May 27, 2014 © LANSA

1. Introduction

1.1 What Is LANSA Open?

1.2 The LANSA Object Repository

1.3 Dynamic Link Libraries (DLLs)

1.4 How LANSA Open Works

1.5 Local Data Dictionary

1.6 Program Examples?

1.1 What Is LANSA Open?

LANSA Open is 'middleware' software that facilitates the high speed transfer of information and data between

• PC-based client applications, developed using any of the popular workstation development tools

and

• a server (LANSA), either on an IBM i, Windows or Linux platform.

An integral part of LANSA Open is the LANSA Object Repository, which contains all field and file details including the relevant business rules. This ensures that all PC applications are subjected to the same rigorous security checks, data validation and referential integrity checks as IBM i applications.

1.2 The LANSA Object Repository

The LANSA Object Repository provides significant advantages to client/server applications because it centralizes the information about the fields and files which make up the application's database. By centralizing information such as business rules, field descriptions and help text in the LANSA Repository, application development time is reduced. An application accessing data known to the Repository can be executed on the server or on a client PC. The same set of rules apply.

If you are using LANSA Open to create a client/server extension to an existing LANSA or non-LANSA application, check the LANSA Repository features before upgrading the existing application and its files. The LANSA Repository's facilities may reduce or even eliminate many of changes which, at first, appear to be necessary.

Virtual fields are just one example of using the LANSA Repository to extend your existing application without changing your files. Using virtual fields, your dates can be presented to applications in formats such as DD/MM/YY or MM/DD/YY, while they are stored in YYYYMMDD format.

Some LANSA Repository features that you can use to enhance your existing applications:

- Field validations and user error messages.
- Field defaults, standard names for labels & headings, stamping attributes, edit codes, etc.
- File level validations.
- Validations for referential integrity checking.
- Predetermined Join Fields
- Database event triggers for fields or files.
- Date, calculation, string types of virtual fields.
- Multilingual definitions.
- Automated help and error messaging.

1.2.1 I/O Modules & OAMs

The LANSA Repository details for files are included in Input/Output Modules. These modules are called I/O Modules (IOMs), or Object Access Modules (OAMs). IOMs are only used on the IBM i in partitions that are not enabled for RDMLX or where no OAM exists. OAMs are used for all other file requests.

Including process and data information in the IOM or OAM means that this information will move wherever the database moves. The information is accessed by all LANSA applications.

To learn more about the LANSA Repository and input/output modules, refer to Introduction to LANSA Object Repository in the Introduction to LANSA for *iSeries Guide*.

1.3 Dynamic Link Libraries (DLLs)

A DLL is a library that links dynamically, not statically, with an application. The library contains a group of compiled functions or programs.

Static linking of libraries results in very large programs which must be recompiled and linked each time a function in the library changes.

Dynamic library links reduce program size and recompilations, because applications link to the library to obtain the current code when they are executed.

The use of DLLs is now very common and many products provide and can use DLLs. For example, C#, Visual Basic, Microsoft Excel and Lotus Notes all call DLLs. Some products require external DLLs to have a product specific interface. These products require a DLL to be written to interface to LANSA Open.

LANSA Open provides its functions in a DLL which must be loaded onto your PC.

1.4 How LANSA Open Works

Your PC based application can be developed using C#, VB.NET, Visual Basic, Paradox, Delphi, Gupta, PowerBuilder, Excel, REXX, C or in fact any PC development tool that can reference a third party DLL.

• The application program makes calls to the LANSA Open functions which reside locally on the PC. This call may be to a function to format data, send information to the server or receive information from the server.

For example, if the application calls the LANSA Open function LceFetch (to get a record from the host), LANSA Open sends a request for data to the server.

• On the server, a LANSA Open program receives the request and determines what action to take.

If a file needs to be accessed, the appropriate LANSA OAM (or I/O Module) is called and the data is retrieved and formatted.

- Once the data is retrieved, it is transmitted back to the PC.
- The PC application can then use other LANSA Open functions to access that data which is stored on the PC.

1.5 Local Data Dictionary

LANSA Open can maintain a local data dictionary on the PC's hard disk to record the file and field information (e.g. description, type, length, etc.) normally returned by the host during the Open Session function call.

The local data dictionary avoids the constant downloading of information from the host in cases where the information has not changed.

The use of the local data dictionary will depend on your application and the user's requirements.

The local data dictionary's details are kept up to date in this way:

- For files, each file is version checked with the host and details updated locally, if necessary, on a per file basis.
- For fields, if any field definition in the Repository on the host has changed, all the fields for that file are replaced in the local data dictionary. This occurs regardless of whether any changed field is requested in the current session or not.

You specify that you want to use the local data dictionary in the Configuration File.

1.6 Program Examples?

A number of sample programs are provided with LANSA Open to assist you in understanding the use of the LANSA Open functions and to help you develop your own applications.

You will find these programs in the <LANSA Root Directory>\Open\Samples directory. If you did not elect to install Samples with LANSA Open, you can do so at any time.

All the sample programs use files from the Personnel System which is shipped as part of the DEM or demonstration partition with LANSA.

In order to execute these sample programs, you must have the DEM partition and the appropriate files loaded onto the host.

A printed sub-set of the Where to Find the Program Examples Visual Basic sample application is provided in this guide.

2. Getting Started

Before you begin development with LANSA Open, the software must be properly installed and configured on your PC. The appropriate LANSA license must be installed on the server and software configuration steps must also be performed on the server.

In this section, you will find an overview of some of the installation and configuration information for reference purposes. For detailed information, refer to the *Installing LANSA on Windows Guide*.

2.1 Installing/Upgrading LANSA Open

You will find the instructions for installing and upgrading LANSA Open in the *Installing LANSA on Windows Guide*.

If you are upgrading from an older version of LANSA Open, be sure to read the Important Note for LANSA Open Upgrade.

The installation will load the following:

- Windows 32 DLL (LCOEW32.DLL)
- A Configuration Utility
- A Help file.

Sample and demonstration programs are included with the LANSA Open software. Refer to the Guidelines for Using LANSA Open for further details about these files.

2.2 Configuration File

LANSA Open's Configuration File parameters are recorded in the Windows System Registry.

The Configuration File parameters are set to default values during the installation process. When the LANSA Open installation has been completed, you must review the default values and change them to suit your installation's requirements. For complete details of these parameters, refer to LANSA Open Configuration Parameters in the Installing LANSA on Windows Guide.

You will view and maintain the Configuration File using the LANSA Open Configuration Utility which opens from the Configurator's desktop folder.

Always use this utility to view or change this file. Never use a registry editor to change these details.

Note: You will need to log on as system administrator to install/change the Configuration File parameters.

You can override some of LANSA Open's Configuration File parameters for individual applications. The LANSA Open function to override a parameter is included with each parameter's description.

2.3 Repository on the Server

The LANSA Repository centralizes the information about the fields and files which make up your server's application database and will save you repeating this information in each client program. In order to access any file using LANSA Open, the file must have been defined to the Repository and an I/O Module created.

If you are already using Visual LANSA or LANSA for iSeries to develop your applications, your files will most likely be known to the Repository. Because of the particular benefits the Repository brings to client/server applications, it is recommended that you check your Repository definitions to ensure that all the applicable Repository features are in use.

If you have not previously used LANSA, then the fields and files must be defined and the Repository characteristics added, as described in the Introduction to *LANSA for iSeries Guide*.

If the files on the IBM i already exist but are not known to LANSA, you will need to load these files into LANSA and create the I/O Modules for them. Refer to Building the LANSA Repository - Files in the *Introduction to LANSA for iSeries Guide* for the steps required to make your files known to the Repository.

If you are creating a new application, you must define the fields and files that will make up the application's database. Building the LANSA Repository - Files and Building the LANSA Repository - Fields in the *Introduction to LANSA for iSeries Guide* tells you how to do this.

2.4 Multilingual Support

To enable LANSA Open to display messages and translate data correctly from the Host System, these two files must be converted to the appropriate language:

- LCOECHAR.DAT (Client <-> IBM i Host Character Translation)
- LCOEMESG.DAT (Error Message Text Translation)

A default LCOECHAR.DAT file translating characters between Windows Codepage 1252 and EBCDIC CCSID 1140 will automatically be installed. If this is not appropriate for your language then you must select another one of the shipped LCOECHAR.DAT files or your own custom LCOECHAR.DAT. EBCDIC CCSID 1140 is identical to CCSID 037 except that € (the Euro

symbol) replaces ¤. This supports the entire Windows 1252 character set except for the following characters:

Characters	Hex Range	Explanation
'f",†‡^‰Š<ŒŽ'"""*~(tm)š>œž	0x82 to 0x9E inclusive and 0xA0	These characters do not exist in EBCDIC CCSID 1140. For this reason they are mapped to EBCDIC Hex values below 0x40. This means the IBM i will interpret them as control characters and thus they cannot be viewed on the IBM i and may affect the display of characters following them. However, these characters may still be saved and retrieved because when these Hex values are retrieved by your LANSA Open application, they will be mapped back to their correct Windows Codepage Hex Code and thus should appear correctly in your LANSA Open application. These characters should not be used if this data will be accessed with software other than LANSA Open.
Ÿ	0x9F	Does not exist in EBCDIC CCSID 1140. Should not be used because it

maps to 0xFF which causes strings to be terminated.

0xA4 The Euro replaces this character

The full set of characters supported is:

Ø

€!#\$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVVXYZ [\]^_`abcdefghijkImnopqrstuvwxyz{|}~j¢£¥¦§°©ª«¬®[−]°±²°´µ¶·,'°»¼½%½ ÀÁÂĂĂÆÇÈÉÊÊÌÍĨĨÐÑÒÓÔŐÖרÙÚŨÜÝÞßàáâããåæçèéêêìíĩĩðñòóôőö÷øùúûüýþÿ

3. Function Overview

This chapter describes a Dynamic Link Library and how it interacts with LANSA Open.

To help you to become familiar with the LANSA Open functions, we have listed them, with a brief description, in the context that you would use them in your applications.

A generic program example is provided at the end of the chapter in order to show you how to apply the LANSA Open functions.

If you are using Visual Basic to develop your client programs, the Visual Basic application extract in Quick Start for Visual Basic Programmers will provide you with more specific examples of using the LANSA Open functions.

3.1 What Are DLLs?

The Dynamic Link Library (DLL) contains a group of compiled functions or programs which link dynamically, not statically, with an application.

LANSA Open provides its functions in a DLL. This DLL will be installed with the LANSA Open installation program.

Using DLLs

In order to use the installed DLL functions in your application, you must perform two steps:

- 1. Make the DLL functions known to the application.
- 2. Call the desired function.

3.1.1 Making DLL Functions Known to Your Application

Each PC application product has its own way of identifying and interfacing with DLLs. You should refer to the user guides of your specific PC application product when determining how to use DLL functions. You might find this information under one of these topics:

- external functions
- calling functions
- including functions
- Dynamic Link Libraries (DLL).

It is important that the DLL be located in the proper directory of the PC so that it can be located when the application executes. Refer to Important Note for LANSA Open Upgrade for notes regarding the location of the LANSA Open DLL.

With most application tools and programming languages, it will be necessary to define the parameters of each of the external functions in the DLL. The parameters for defining the functions are identified in the online Help text and the Function Details. This step need only be performed once as the resulting code is generally copied from one application to the next or is stored in a file which can be copied or imported into the application.

Once the DLL functions have been defined to the application, the functions can be called.

The Visual Basic sample programs provided with LANSA Open contain the relevant definitions to use with your Visual Basic applications.

3.1.2 Calling DLL Functions

DLL functions can be called using one of two techniques:

- 1. Call by function name.
- 2. Call by ordinal number.

Calling a function by name is recommended over calling a function by ordinal number, in case the ordinal number of the DLLs change. When a function is called by name, its position in the DLL is not important. If you call a function by ordinal number, you are identifying the function strictly by its location in the DLL.

When calling the functions, it is of value to note that the LANSA Open functions for all Windows versions are the same, even though the DLL files are different.

For more details about making DLLs known to your application, refer to the Guidelines for Using LANSA Open.

3.2 LANSA Open Function Summary

3.2.1 Which LANSA Open Function to Use and When

- 3.2.2 Session Management
- 3.2.3 Session Definition
- 3.2.4 Data Access
- 3.2.5 Miscellaneous Functions

3.2.1 Which LANSA Open Function to Use and When

When using LANSA Open functions, there will always be a consistent structure to your program or application. Your applications will have four basic sections. These are:

Session Management (Open Session)	Opens a communication session with the host and obtains the session identifier.
Session Definition	Defines the application-specific session characteristics for the host.
Data Access	Accesses data on the host.
Session Management (Close Session)	Closes an opened host session.

The majority of the LANSA Open functions are only executed in one of these sections. Which functions go where is explained in this section.

The sample programs supplied with the LANSA Open software, as well as those provided on LANSA's web site show you how these functions are used in an application.

3.2.2 Session Management

Using the session characteristics defined by the functions described in the *Session Definition* section, your application will open a communications session with the host.

An application may have more than one session open with the host.

Open Session

LceGetSessionId Obtains a valid session number.

LceOpenSession Opens a communication session with the host.

Close Session

Once your application has completed, the communications session must be closed. All sessions opened on the host by your application should be closed by your application.

Failure to close a session will result in an unnecessary resource overhead on the host.

LceEndSession Ends the specified session with the host.

3.2.3 Session Definition

The session definition functions are used to define the characteristics of your application to the host. These functions must be performed before a session is opened. This information will be passed to the host when the session opens:

LceUseSystem	Specifies the host system to be used.
LceUsePartition	Specifies the partition to be used.
LceUseLanguage	Instructs the session to use the specified language.
LceSetSessionOption	Sets the file open option and the returned data format for the session, if different to the default options specified in the Configuration File.
LceSetCommitmentOn	Turns commitment control on.
LceSetCommitmentOff	Turns commitment control off.
LceUseUserId	Specifies the host User Identifier for this session.
LceUsePassword	Specifies the password corresponding to the host User Identifier.
LceUseFile	Specifies the files to be used in the session.
LceUseField	Specifies the fields to be used in the session.
LceUsePriority	Specifies the host job priority.
LceUseProcess	Specifies the process name for transaction stamping.
LceUseFunction	Specifies the function name for transaction stamping.
LceSetSelectOptions	Specifies the options to be used for the first SELECT and FETCH functions. These options can be changed during the session and are reset after each SELECT.

LceUseExceptionalArguments Specifies exceptional arguments (X_RUN Parameters) to be used. This option should only be used with a host that supports X_RUN parameters.

3.2.4 Data Access

The majority of your applications will consist of data related functions.

Once your application has opened a communications session, it can access the data on the host. Using LANSA Open, you can get, add, change and delete information on the host. You may also execute LANSA and other host programs.

There are a wide range of data access functions. These functions have been grouped into the following functional areas:

File Processing - Single Record	These functions are used to fetch, add, change or delete a single record from the host.
File Processing - Multiple Record	These functions are used to retrieve one or more records from the host.
File Commitment Control	These functions are used if you have commitment control invoked for your host database.
Retrieving Repository Details	These functions are used to retrieve information, such as descriptions, labels and column headings, which are stored in the host Repository.
Manipulating Field Data	These functions are used to use or prepare the current value for a field.
Calling Processes on the Host	These functions are used to execute LANSA and iSeries programs or OS/400 functions.
Message/Error Handling	LANSA Open has its own automated error management facility, or alternatively, your application can perform its own error handling using the functions provided.

File Processing - Single Record

The LceInsert, LceUpdate and LceDelete functions have a validate parameter which allows a check to be made without performing the actual operation.

LceCheckFor	Checks for the existence of a record in the specified file. The validate parameter of the insert, update or delete function can be used instead of this function.
LceSetSelectOptions	Specifies the select or fetch options.
LceFetch	Fetches the values of the specified fields from the specified file.
LceInsert	Inserts the values of the specified fields into a record.
LceUpdate	Updates the record with the values in the fields specified.
LceDelete	Deletes a record from the specified file.

Note: When multiple users are accessing a database, eventually two or more users will want to update the same record at the same time.

What should you do about this?

Refer to the LceUpdate or LceDelete function details to find out.

File Processing - Multiple Record

LceRequestSelect	Selects records on the host.
LceRequestSelectWhere	Provides a 'where' condition to be used when selecting records on the host.
LceReceiveSelect	Transfers records from the host to the PC.
LceGetRecordCount	Determines how many records were transferred from the host.
LceGetSelect	Retrieves the next record from the records transferred to the PC.
LceSelect	Selects and retrieves specified fields from file.
LceReceiveNextX	Gets the next record transferred from the host.
LceDeleteSelect	Deletes the temporary file used to hold the host records transferred to the PC.

File Commitment Control

These functions are only effective if the LceSetCommitmentOn function has

been executed and Commitment Control definitions have been configured correctly on the host files.

LceCommitWork Commits the current transaction.

LceRollBackWork Rolls back the current transaction.

Retrieving Repository Details

These functions are used to retrieve information from the Repository for your PC application:

LceGetFieldDesc	Retrieves the field description.
LceGetFieldHeading	Obtains the three line field heading.
LceGetFieldIndicator	Returns the DBCS field indicator.
LceGetFieldLabel	Returns the field label.
LceGetFieldType	Returns the field type.
LceGetFieldValueH	Sets the window text field value.
LceGetFileDesc	Returns the file description.
LceGetFileType	Returns the file type.
LceGetVariable	Retrieves the system or multilingual variable.
LceGetFieldEdit	Fetches the edit code or word.
LceGetDefaultValueX	Returns the field's default value.
LceGetFieldHelpText	Returns the field's help text to a buffer or displays it in a window.
LceGetFieldAttributes	Returns the field's attributes.

Manipulating Field Data

LceGetFieldValueX	Retrieves the field's current value.
LceSetFieldValue	Sets the field value in the field information area.
LceLocalDateTimeToServer	Converts the DateTime from the client's local timezone into UTC (Universal Coordinated Time). This function is for use in creating

Where clauses which contain DateTime values.

LceGetFieldDataLength Retrieve the length of the field's current value.

Calling Processes on the Host

These functions allow you to execute processes on the host.

By re-using existing host processes and functions, significant savings can be made in development time. Consistency between client and host processes is also maintained.

For tasks requiring access to multiple files, perform them on the host, to allow your client applications to benefit from the host's fast file processing.

LceLansaCall	Executes a LANSA process/function or an iSeries program with an immediate response.
LceExecute400	Executes the specified function on the iSeries.
LceSubmit	Submits a LANSA process/function or an iSeries program in batch mode.
Lce3GLDefineParameter	Defines the attributes and optionally sets the values of the parameter used in the Lce3GLCall function.
Lce3GLCall	Executes the specified 3GL program on the iSeries.
Lce3GLGetValue	Fetches the parameter value returned by the Lce3GLCall function.

Message/Error Handling

There are two ways to manage your error messages.

You can:

• let LANSA Open perform all the error message handling automatically

or

• do your own error message handling.

LceDisplayErrors	Displays error messages automatically.
------------------	--

LceGetStatus Returns the error status and message, if any.

LceGetMessageCount Counts the number of messages sent by the Host System.

LceGetMessage

Returns the specified error message.

3.2.5 Miscellaneous Functions

There are some other optional functions, in addition to those already listed. The LceASCIIToFloat and LceASCIIToInteger functions are generally not needed as most application languages provide data conversion routines. You should use the supplied routines where possible.

LceASCIIToFloat	Converts a string containing numerics to a floating point number.
LceASCIIToInteger	Converts a string containing numerics to a long integer number.
LceUseTmpDrive2	Use the second directory for the PC's temporary files.
LceSetCommsCursor	Set the cursor. This function is only available with Windows applications and must be specified during Session Definition.
LceSetCommsType	Set the communications type, if different to the default, when accessing the host. This must be specified during Session Definition.
LceVersion	Returns the LANSA Open version number or displays it in a dialogue box.
LceGetSystemAttribute	Returns the selected system attribute.

3.3 Generic Coding Example

The following example shows how to apply the LANSA Open functions and is not directed to any one language. It is supplied to illustrate the steps required, using familiar constructs.

Note that this is a simple example and detailed error handling has been omitted. Your own application programs should check the return codes from the LANSA Open functions and provide for the appropriate handling of any errors.

```
/* SESSION DEFINITION SECTION */
/* Get a session ID */
iSession = LceGetSessionId()
if not iSession
      return FALSE
endif
/* Specify LANSA partition to be used as DEM */
if not LceUsePartition ( iSession, "DEM" )
      return FALSE
endif
/* Specify the language of the partition as French */
if not LceUseLanguage (iSession, "FRA")
      return FALSE
endif
/* Set the execution priority on host to 20 */
if not LceUsePriority (iSession, 20)
      return FALSE
endif
/* Define the files and fields to be used */
if not LceUseFile (iSession, "PSLMST, DEPTAB")
      return FALSE
endif
if not LceUseField ( iSession, "EMPNO, SURNAME, GIVENAME, SALARY" )
      return FALSE
endif
/* OPEN SESSION SECTION */
if not LceOpenSession (iSession)...
      Error ("Could not open a session.")
      return FALSE
```

endif /* DATA ACCESS SECTION */ /* Fetch an employee record */ /* Set the value of the key field, EMPNO */ if not LceSetFieldValue (iSession, "EMPNO", "A1001") return FALSE endif /* Fetch the record from the host */ If not LceFetch (iSession, "SURNAME, GIVENAME, SALARY", "PSLMST", ' Error ("Employee record not found.") return FALSE else /* Get the value of the SURNAME field */ if not LceGetFieldValueX (iSession, "SURNAME", strSurname, lFlags) return FALSE endif /* Get the value of the SALARY field */ if not LceGetFieldValueX (iSession, "SALARY", strSalary, lFlags) return FALSE endif /* Convert the SALARY field from a string to a float */ LceASCIIToFloat (strSalary, nSalary) endif /* Select all employee records (No keys) */ /* Using *RECEIVEIMMED and specifying a buffer with LceReceiveNextX should result in better performance */ if not LceSetSelectOptions(iSessionId, "*RECEIVEIMMED") return FALSE endif /* Specify the records to be selected from the host */ if not LceRequestSelect (iSession, "EMPNO, SURNAME, SALARY", "PSLMS' return FALSE else /* Transfer the selected records to the PC */ Do While (LceReceiveNextX(iSessionId, strBuff, lSize,

```
fldData, iTotalFields ) = LceTrue )
```

```
strEmpNo = CopyString( strBuff, fldData(0).fieldLen )
          iStartPos = fldData(0).fieldLen
          strSurname = CopyString( strBuff, iStartPos, fldData(1).fieldLen )
          iStartPos = iStartPos + fldData(1).fieldLen
          strSalary = CopyString( strBuff, iStartPos, fldData(2).fieldLen )
     End While
endif
/* END SESSION SECTION */
LceEndSession (iSession, FALSE)
/* iSession is now invalid and the data dictionary has been removed from memory
/* Alternative method for performing the above query */
/* Specify the records to be selected from the host */
if not LceRequestSelect ( iSession, "EMPNO, SURNAME, SALARY", "PSLMS'
     return FALSE
else
     /* Transfer the selected records to the PC */
     if not LceReceiveSelect ( iSession, "PSLMST" )
          return FALSE
      else
          /* Determine how many records were transferred */
          LceGetRecordCount ( iSession, "PSLMST", nRecordCount )
          /* Loop through all transferred records */
               For (I = 1 to nRecordCount)
               /* Get a record from the data transferred */
               LceGetSelect ( iSession, "PSLMST", I, "EMPNO, SURNAME" )
               LceGetFieldValueX (iSession, "EMPNO", strEmpno, lFlags)
               LceGetFieldValueX (iSession, "SURNAME", strSurname,
lFlags)
          Next
      endif
endif
/* END SESSION SECTION */
LceEndSession ( iSession, FALSE )
/* iSession is now invalid and the data dictionary has been removed from memor
```

4. Guidelines for Using LANSA Open

A number of sample programs are provided with LANSA Open to assist you in understanding the use of the LANSA Open functions and also to help you develop your own applications.

This chapter contains some things you need to know when using specific PC application development tools.

This information is current at the time of printing.

Tip

If you visit LANSA's web site, you will find the latest tips for using LANSA Open as well as declarations and sample applications.

4.1 Where to Find the Program Examples

You will find the sample programs in the <LANSA Root Directory>\Open\Samples directory.

If you did not install the Samples initially, you can do so at any time. Sample programs are provided in Visual Basic.

All the sample programs use host files from the Personnel System which is shipped as part of the demonstration partition (DEM). In order to execute these sample programs, you must have the DEM partition and the appropriate files loaded onto the host.

For the latest tips, techniques and sample programs, visit LANSA'S WWW site at http://www.lansa.com/support

4.2 LANSA Open and Visual Basic

When creating applications in Visual Basic with LANSA Open, each of the LANSA Open functions must be defined to your Visual Basic application program. In Visual Basic, the easiest way of doing this is to add the file LCOE1632.BAS to your project using the Add Module option of the Project menu. This file is supplied in the Visual Basic samples that come with LANSA Open.

The LANSA Open functions can also be defined in any other *.BAS modules of your programs. If you want to define the functions manually, declare them using the following format:

32 bit Visual Basic

Declare Function LceGetSessionId Lib "LCOEW32.DLL" () As Integer

Special Considerations

There are some special considerations you need to take into account when calling the DLL functions from Visual Basic.

These are:

Pre-allocate Strings

When using a string variable to receive data in a DLL function, Visual Basic must be forced to reserve space for it before the call. This is done by re-assigning the variable to spaces or nulls:

```
dim s as string
s = Space (SIZE)
```

or

s = String (SIZE, chr(0))

Failure to reserve enough space will result in a GPF (General Protection Fault) error.

Testing a function's results

Return Code "True" is defined as -1 in Visual Basic, so this code:

```
if lceFunction(...) = True then ' is incorrect
```

if lceFunction(...) = LceTrue then ' is correct

```
LceTrue is a constant defined to be 1.
```
Null terminated strings

In the LANSA Open DLLs, 0 is used as the termination marker for strings. However, Visual Basic does not recognize 0 as the terminating character in a string. Your Visual Basic program will have to detect the 0 character.

In the sample programs supplied with LANSA Open, the sTrim function is used to remove any leading or trailing blanks from a given string. In addition, the sTrim function recognizes the 0 character as the termination character of a string. In the examples, the strings are always pre-allocated with nulls, rather than spaces to make the sTrim function more efficient.

You can copy the sTrim function from the UTIL.BAS module of the Employee sample project supplied with LANSA Open.

Function sTrim(s As String) As String

```
' this function trims a string of right and left spaces
' it recognizes 0 as a string terminator
Dim i As Integer
i = InStr(s, Chr$(0))
If (i > 0) Then
    sTrim = Trim(Left(s, i - 1))
Else
    sTrim = Trim(s)
End If
End Function
```

Helpful Visual Basic Routines

Visual Basic provides routines to work with and convert strings and numeric data, such as:

- Val()
- Str() and Str\$()
- Left() and Left\$()
- Right() and Right\$()

4.3 LANSA Open and C

Any C module making LANSA Open function calls should include the LCOE.H header file which is installed with the 'Development Tools'. The LCOE.H header file contains a list of constants required for the parameters used by the DLL functions. This file also contains the definition of each of the DLL functions. When using the LCOE.H header file, it is necessary to define a macro to indicate what operating system the application is being compiled for.

• For 32 bit Windows, the macro LCE_WIN32 must be defined.

These macros should be defined before the LCOE.H file is included. This step may be done inside the code or outside the code, by defining the macro when calling the compiler (Option -D on most compilers) or in your project's settings.

There are two methods to include LANSA Open when compiling your project.

- 1. The recommended method is to link the installed lcoew32.lib file to your project. See your compiler's documentation on how to perform this task. In general terms, this is performed by adding lcoew32.lib as a dependency to your project's Linker settings.
- 2. In previous versions of LANSA Open the following method was documented:

"To create an Import Library for Windows, you should create your own Import Library file using a DEF file as input to the LIB utility. The SAMPLES\C directory has an example of a DEF file for this purpose, with additional information as comments. Refer to Microsoft documentation describing the LIB utility and modify the example DEF file as required." You may continue to use this method but it is no longer the recommended method.

4.4 LANSA Open and Excel

When creating applications with Excel macros, you do not have to explicitly define the LANSA Open DLL functions to Excel. The functions are simply called directly as external functions within a macro.

The DLL functions can be called using the Excel CALL command where the command might be used as follows:

SET.NAME("nSessionId",CALL("LCOEW32","LceGetSessionId","I"))

or

```
CALL("LCOEW32","LceEndSession","AIA",nSessionId, FALSE)
```

The format of the CALL command is:

```
CALL("A","B","C",D,E,...)
```

where

A is the name of the DLL file, LCOEW32

B is the name of the LANSA Open function to be executed

C is the description of the parameters. The format is XYZ,

where

X is the type of return value

Y is the type of the first parameter

Z is the type of the second parameter, etc.

D, E ... are the parameters passed to the function

For more information about using the CALL command, refer to the Excel guides. In particular, review the data types used for parameter passing. An appendix in the Excel guides will list the data types.

In the example:

CALL("LCOEW32","LceEndSession","AIA",nSessionId, FALSE)

the parameters "AIA" indicate that the function returns a Boolean and is passed an integer and a Boolean.

Commonly used values include:

- A is a Boolean passed by value
- C is a zero-terminated ANSI string passed by reference
- I is a signed integer passed by value.

Where the function returns a boolean operator, the LANSA Open functions are commonly used in this manner:

IF(NOT(CALL("LCOEW32","LceOpenSession","AI",nSessionId)),RETURN(F/ Be sure to refer to the Excel macro samples provided.

5. Quick Start for Visual Basic Programmers

This is a step by step walk through the Microsoft Visual Basic code required to fully maintain a host file. It uses code from the sample program SIMPLE.VBP which is a subset of the Personnel System supplied with the LANSA Open software.

You will find the complete Visual Basic sample application on the LANSA Open installation media.

Before you start using LANSA Open functions in your program, please be sure to read the guidelines for using Visual Basic in Guidelines for Using LANSA Open.

5.1 Overview

This Skill File Maintenance program shows you how to use Visual Basic (VB) and LANSA Open to maintain a file residing on the host using a simple VB screen.

🛋 Skills	_ 🗆 ×	
Skill		
ACCOUNTING ADMIN PART 1 ADMIN PART 2 ADVANCED PROGRAMMING COMMUNICATIONS DEG DDS CODING & SDA USE ECONOMICS DEGREE COMPANY INTRODUCTION KEYBOARD SKILLS MANAGEMENT COURSE 1 MANAGEMENT COURSE 2 MANAGEMENT COURSE 3 MANAGMENT COURSE 4		
Skill code		
MANAGE1		
P		
Add New Save Delete	Discard	
Skill Full Description		

By following this sample code, you will see how to create an application that allows you to:

- browse
- add
- change and
- delete items from a host file.

You can cut and paste the code from this program to use in your own VB applications. You will find it in <\lansas>\SAMPLES\VB\SIMPLE.

The Skills File Maintenance program (simple.vbp) contains only one form (simple.frm) and uses the most common LANSA Open DLL calls to update the file SKLTAB.

In Simple.frm, wherever possible, the code is linked to events on the form controls. The Form_Load event is used to open the session, the Form_Unload to

close the session and the skills list click event is used to navigate through the records.

5.1.1 Program steps

To access host data from the PC, you:

Step 1: Define and Open the host session

- 1. GetSessionId
- 2. Set Session options
- 3. Open Session

Step 2: Access Data

- 1. Get Dictionary Details to use
- 2. Request Records
- 3. Retrieve Records
- 4. Process Records

Step 3: Close the host session

Close Session

5.2 Defining and Opening the Host session

Note: The LCOE1632.BAS module supplied with the sample VB programs contains all the Visual Basic V4.0 declarations to LANSA Open functions. To use the functions, insert the file into your project by using the File Add option of the File menu. All code presented in this chapter can be found in simple.frm.

5.2.1 Declare Variables

Three variables are used in the following samples: iSession, iRet, and sBuff. Declare them on the General Declarations section of your form:

Dim iSession as Integer, iRet as integer, sBuff as String

iSession is used to contain the LANSA session handle, iRet is used to receive the return code from LANSA calls, and sBuff is a general purpose buffer for data retrieved from the host.

5.2.2 GetSessionID

First, you must get a session ID to work with. This ID will be used in all subsequent LANSA Open DLL Calls (LceFunctions):

```
ISession = LceGetSessionId()
If iSession = 0 Then
MsgBox "No Session ID"
End
End If
```

5.2.3 Set Session Options

You must set the session parameters before opening the host session. Note: To run the program, replace UserID, Password and System with your own.

'Set Session ID

```
' System / User dependent parameters, replace to run program !
iRet = LceUseUserId(iSession, "USERID")
If iRet = LceTrue Then iRet = LceUsePassword(iSession, "PASSWORD")
If iRet = LceTrue Then iRet = LceUseSystem(iSession, "SYSTEM")
If iRet = LceTrue Then iRet = LceUsePartition(iSession, "DEM")
If iRet = LceTrue Then iRet = LceUseLanguage(iSession, "ENG")
' set file for session
If iRet = LceTrue Then iRet = LceUseFile(iSession, "SKLTAB")
```

```
If iRet = LceTrue Then iRet = LceUseFile(iSession, "SKLTAH

' set fields for session

If iRet = LceTrue Then

iRet = LceUseField(iSession, "SKILCODE, SKILDESC")

End If
```

If iRet = LceFalse Then MsgBox "Set Session failed" End End If

Note: Almost all LANSA functions return an integer, 1 (LceTrue) for success or 0 (LceFalse) for failure. VB True cannot be used because it is defined as -1.

5.2.4 Enabling LANSA Open Default Error Handler

LANSA Open, by itself, can trap and display the errors which can occur in a session.

To enable the default error handler:

iRet = LceDisplayErrors(LceTrue, 0)

5.2.5 Opening the Host Session

Opening the session connects the user to the host.

'open Session iRet = LceOpenSession(iSession) '------If iRet = LceFalse Then MsgBox "Open Session failed" End End If

Note: Normally, you only need to open one session. One example of when you may want more than one session is when you want to access other files while receiving records using a select with *RECEIVEIMMED mode.

5.3 Data Access

5.3.1 Using Field Labels and Descriptions from the LANSA Repository

- 5.3.2 Populating a List with the Contents of a Table
- 5.3.3 Retrieving a Specific Record
- 5.3.4 Setting a New Record's Fields to Default Values
- 5.3.5 Saving Changes / Inserting Records
- 5.3.6 Deleting Records
- 5.3.7 Retrieving Help Text

5.3.1 Using Field Labels and Descriptions from the LANSA Repository

The following example retrieves field descriptions from the LANSA Repository to be used on the screen (e.g. Skill, Skill Code and Description). Using the names & descriptions that are stored centrally in the Repository ensures that you maintain consistency between all users of the system and allows you to easily accommodate different languages.

'set field labels and lengths by reading 'their definitions from dictionary

'Fill the buffer with as many zeroes as specified in the 'FIELD_DATA_SIZE constant in LCOE1632 module sBuff = String(FIELD_DATA_SIZE, Chr(0))

```
'Get skill code label
iRet = LceGetFieldLabel(iSession, "SKILCODE", sBuff)
lbCode.Caption = sTrim(sBuff)
```

```
'Get skill description label
sBuff = String(FIELD_DATA_SIZE, Chr(0))
iRet = LceGetFieldLabel(iSession, "SKILDESC", sBuff)
lbDesc.Caption = sTrim(sBuff)
```

Note: The sTrim function used above can be found in the UTIL.BAS module of the employee sample program.

Retrieve the SKILCODE field's long description from the LANSA Repository and display it in the status bar at the bottom of the form.

```
sBuff = String(FIELD_DATA_SIZE, Chr(0)) ' pre-allocate memory
```

iRet = LceGetFieldDesc(iSession, "SKILCODE", sBuff)

'_____

status2 = sTrim(sBuff)

Read and set the maximum length for fields SKILCODE and SKILDESC.

Dim length as integer, decimals as integer

'Set skill code field length sBuff = String(FIELD_DATA_SIZE, Chr(0)) iRet = LceGetFieldType(iSession, "SKILCODE", sBuff, _ length, decimals) txtCode.MaxLength = length

·_____

'Set skill description field length

sBuff = String(FIELD_DATA_SIZE, Chr(0))

txtDesc.MaxLength = length

5.3.2 Populating a List with the Contents of a Table

To populate a list with the contents of a table, you must first request the set of records you want to use and then receive them. *RECEIVEIMMED mode is used for better performance. In this example, LceSetSelectOptions is used to set the select mode.

LceDeleteSelect can be used to clear any previously select results. This is required if a previous Select is interrupted and is not fully received.

After the file and fields in it have been selected with LceRequestSelect, a LceReceiveNextX loop is used to retrieve all selected records.

The following example is a simplified version of the code in the SetSkillsList subroutine in SIMPLE.VBP.

' reads all skill descriptions and populates the skill list

On Error Resume Next

' remove any previous query results

iRet = LceDeleteSelect(iSession, "SKLTAB")

'_____

'Set Select to IMMEDIATE MODE as recommended for reasons of speed iRet = LceSetSelectOptions(iSession, "*RECEIVEIMMED")

'_____

' Select fields and file from which records will be retrieved iRet = LceRequestSelect(iSession, "SKILDESC", "SKLTAB", "", False)

sBuff = String(FIELD_DATA_SIZE, Chr(0)) ' Initialize buffer
lstSkills.Clear 'Clear List

'-----

```
'Read all records selected
```

```
While (LceReceiveNextX(iSession, sBuff, FIELD_DATA_SIZE, fldData(0), 1) = LceTrue)
```

```
'_____
```

```
If (fldData(i).flags = 0) Then

' Value is not SQLNULL

lstSkills.AddItem sTrim(sBuff) ' add description to list
```

End If Wend

Notes:

All the records have to be retrieved, otherwise you must use LceDeleteSelect before you use any of the other record manipulation functions.

While not used in this case, (in this example the whole table is required) the number of retrieved records can be restricted by specifying keys in LceRequestSelect. The key values are set with LceSetFieldValue. A variation of LceRequestSelect, LceRequestSelectWhere, allows the specification of an additional SQL WHERE-like condition.

Instead of using a buffer to receive the data, LceGetFieldValueX could be used. The buffer in LceReceiveNextX is then not required.

```
While (LceReceiveNextX(iSession,"", 0, vbNullString, 0) = LceTrue)
```

!_____

```
iRet = LceGetFieldValueX(iSession, "SKILDESC", sBuff, lFlags)
lstSkills.AddItem sTrim(sBuff)
```

Wend

5.3.3 Retrieving a Specific Record

To retrieve a specific record, LceFetch is used. Before calling LceFetch, the value of the key used in the search is set using LceSetFieldValue.

If LceFetch returns LceTrue (1), LceGetFieldValueX can then be used to retrieve the data.

```
Private Sub lstSkills_Click()
  ' synchronize fields with list
  If (lstSkills.ListIndex > -1) Then
    ' Read record
   iRet = LceSetFieldValue(iSession, "SKILCODE", mycode) ' Set Key
    !_____
   iRet = LceFetch(iSession, "SKILCODE, SKILDESC", "SKLTAB", _
        "SKILCODE") ' Get Record
    !_____
    If iRet = LceTrue Then
     ' set code
     sBuff = String(FIELD_DATA_SIZE, Chr(0)) _
          ' required when calling dlls
     iRet = LceGetFieldValueX(iSession, "SKILCODE", sBuff, lFlags)
         ' Get Field
      !_____
     If lFlags = 1 Then
       txtCode = "Null value"
     Else
       txtCode = sTrim(sBuff) ' display data
     End If
     ' set desc
     iRet = LceGetFieldValueX(iSession, "SKILDESC", sBuff, lFlags)
      !_____
     If lFlags = 1 Then
       txtDesc = "Null value"
     Else
       txtDesc = sTrim(sBuff)
```

End If End If

End If

Exit Sub

End Sub

5.3.4 Setting a New Record's Fields to Default Values

A new record is added to a table using LceInsert. In this case, the Add command button is used only to set the default values of the fields retrieved from the dictionary (clears record). On Save, either LceInsert or LceUpdate is called, depending on whether the current record is being added or edited.

Private Sub cmdAdd_Click()

```
'Code
sBuff = String(FIELD_DATA_SIZE, Chr(0))
iRet = LceGetDefaultValueX(iSession, "SKILCODE", sBuff, lFlags)
1_____
If lFlags = 1 Then
  txtCode = "Null value"
Else
  txtCode = sTrim(sBuff)
End If
'Desc
sBuff = String(FIELD DATA SIZE, Chr(0))
iRet = LceGetDefaultValueX(iSession, "SKILDESC", sBuff, lFlags)
!_____
If lFlags = 1 Then
  txtDesc = "Null value"
Else
  txtDesc = sTrim(sBuff)
End If
'set flag to indicate that a new record is being added.
'this flag is used in the cmdSave_Click() subroutine
bAdding = True
```

End Sub

5.3.5 Saving Changes / Inserting Records

The command button, Save, is used to Insert or Update a record. Before updating the data entered by the user, it has to be set in the session buffer. In this case it is done in the Change Event of the form fields.

Note : A value of "" is invalid as it is not a null terminated string, chr(0) must be used instead.

```
Private Sub txtCode_LostFocus ()
```

If txtCode <> "" Then
 iRet = LceSetFieldValue(iSession, "SKILCODE", txtCode)
 '_____
Else
' note we require a null terminated string
iPot = LceSetFieldValue(iSession, "SKILCODE", Chr(0))

iRet = LceSetFieldValue(iSession, "SKILCODE", Chr(0))
End If

End Sub

```
Private Sub cmdSave_Click()
```

```
If bAdding = True Then

' user is inserting a new record

iRet = LceInsert(iSession, "SKILCODE,SKILDESC", _

"SKLTAB", LceFalse)

'______
```

Else

End If

If iRet = LceTrue Then ' if successful force a refresh so list shows changes Call RefreshSkillsList End If

End Sub

5.3.6 Deleting Records

Deleting a record is similar to finding one. The key value has to be set with SetFieldValue. In this case, this step is skipped, as it is already set.

Private Sub cmdDelete_Click()

iRet = LceDelete(iSession, "SKLTAB", "SKILCODE", LceFalse)

·_____

If iRet = LceTrue Then ' refresh list so deleted skill doesn't show Call RefreshSkillsList End If

End Sub

5.3.7 Retrieving Help Text

Displays the field's Help text from the LANSA Repository when a field gets focus and the user presses F1:

Private Sub txtDesc_KeyDown(KeyCode As Integer, Shift As Integer)

```
If KeyCode = vbKeyF1 Then
    iRet = LceGetFieldHelpText(iSession, "SKILDESC", "U", Chr(13) _
& Chr(10), "", 0)
```

End If End Sub

5.4 Error Handling

This sample program relies on the LANSA Open default handlers, but if required, error information can be retrieved using LceGetStatus and LceGetMessages. A simple function to display errors will look like this:

Sub DispLceError(iSessionID As Integer, sSource As String)

' this function displays LANSA Open errors

' Note that it only does anything if LceDisplayErrors is off

Dim i As Integer Dim sErrDesc As String, iErrNo As Long Dim sMsg As String, nMsgs As Integer

Dim msgBuff As String ' message buffer to contain final message

msgBuff = "" ' Get error status sErrDesc = String((MESSAGE_LENGTH + 1), Chr\$(0)) '----- Important force VB to reserve space for DLL reply Call LceGetStatus(iErrNo, sErrDesc, MESSAGE_LENGTH) '------

```
If iErrNo > 0 Then
```

For i = 1 To nMsgs '

```
sMsg = String(MESSAGE_LENGTH * 3, Chr(0))
```

iRet = LceGetMessage(iSessionID, i, sMsg, MESSAGE_LENGTH)

```
'_____
```

```
sMsg = sTrim(sMsg) ' sTrim detects null terminated strings
msgBuff = msgBuff + sMsg ' add message to buffer
Next
msgbox msgBuff
```

End If End Sub

5.5 Closing the Session

Before terminating, an application has to close the open session. In this case, the Form_Unload event is used to call End Session:

Private Sub Form_Unload(Cancel As Integer)

```
If iSession > 0 Then
iRet = LceEndSession(iSession, LceFalse)
End If
```

End Sub

Note: Passing LceTrue as the second parameter (Resume) of LceEndSession will allow re-use of the same Session ID. This is more efficient than getting a new Session ID, if the same Host and Partition are required.

This concludes a quick introduction to LANSA Open functions in the Visual Basic environment. Detailed information about each LANSA Open function can be found in Function Details.

6. Function Details

Each of the LANSA Open functions is described in detail in this chapter.

The functions are listed in alphabetical sequence, to make it easier for you to find the one you are looking for.

At the beginning are guidelines about the layout and terms used in the function descriptions.

If you are looking for the actual sequence in which you might use these functions, you'll find them listed in the Function Overview.

Function details and the examples given for each function in this chapter can also be found in LANSA Open's on-line help.

6.1 Function Format

Each LANSA Open function is described using this format, based on the structure of a 'C' function line:



Parameters

Char

Param-type indicates the type of value for the parameter, while parameter1 and parameter2 represent the actual variable you will declare and use in your application.

A 4 byte pointer to a character string in C or a simple character string

Valid param-types include:

far * in most o		in most o	other languages.	
0			pointer to a floating point value in C or a floating point) number in most other languages.	
	Int far *			
Long A 4 byte pointer to a long integer far * other languages.			pointer to a long integer value in C or an integer in most guages.	
Int An integer.		er.		
	BOOL A boolean variable (TRUE or FALSE).			
HWND A string representing the Window Handle or identifier.				
Some examples of parameter1 and parameter2 you will see are:				
	strNume	ericString	To represent a character string containing numerics.	
strFileName		ame	To represent a character string containing a file name.	
	isession		To represent an integer variable storing the session identifier.	
	FOK		To represent a boolean variable (TRUE/FALSE).	

The actual length of a parameter is indicated by the supplied field name as used in the LCOE.H header file, supplied for use with 'C' programs.

Return Values

Represented by 'Returns' in the function format, this is the value that the function returns when it is called.

Valid values include:

VOID No value is returned

When a void or no value is returned, you might use the function in the following manner:

LceFunction(parameter1, parameter2)

or

CALL LceFunction(parameter1, parameter2)

BOOL A boolean value is returned.

TRUE is 1

FALSE is 0. False normally results in an error code. A list of the error codes is in Appendix A of this guide.

When a boolean value is returned, you might use the function in the following manner:

BOOL fOK; fOK = LceFunction(parameter1, parameter2); if (fOK) { }; or

If LceFunction(parameter1, parameter2)

IF LceFunction(parameter1, parameter2) /* comment */ ELSE /* comment */ ENDIF

or

in Visual Basic

If LceFunction(parameter1, parameter2) = LceTrue then ' comment Else 'comment End If 'LceTrue is a constant declared with the value 1, don't confuse it with 'VB True = -1

int An integer number is returned.

Long A long integer is returned.

When an integer or long integer is returned, you might use the function in the following manner:

int iVal; iVal = LceFunction(parameter1, parameter2)

Notes

This section provides important notes regarding the use of the function.

Portability	This section will also note any differences that may be	
Considerations	s encountered between using IBM is and other platforms as	
	the hosts. A general discussion regarding host platform	
	differences may be found in Things That May Be	
	Different in the LANSA Application Design Guide's	
	chapter: Generated Code C and SQL/ODBC	
	Considerations.	

Tips

If there is a more efficient way of using the LANSA Open function, it will be described in the Tips section.

For the latest LANSA Open tips and program samples, check out LANSA's web site at http://www.lansa.com/support

Related Functions

This section lists any other related functions or important areas of this guide that you should review.

6.1.1 Examples

In general, two simple examples of how the function might be used are provided. The first example is based on the Visual Basic Quick Start Skills application which is supplied with LANSA Open. A portion of this program is printed in Quick Start for Visual Basic Programmers in this guide. The second example may be a C example or a generic example of how you might use the function.

In the Visual Basic examples, it is assumed that

- **a session is set and open** as shown in the LceOpenSession function's example.
- iSession is a global integer **variable containing the open session number.**
- iRet is an integer used to get the Lce Function return codes.
- sBuff is a string buffer used to temporarily store data retrieved with LANSA Open functions. Before using its value, call sTrim to remove padding spaces and to detect the end of the string.

6.2 Lansa Object Names

LANSA Version 13 introduced long names for LANSA objects. Each LANSA object has two names – a long name and an identifier. Refer to the LANSA Object Names in the LANSA Technical Reference Guide for further details on long names.

Lansa Open supports the use of either the long name or identifier to identify files, fields, processes and functions. If a function parameter requires a list of field names, the list may contain all long names, all identifiers or any combination of long names and identifiers.
6.2.1 C Example

BOOL fOK;

fOK = LceRequestSelect(iSession, "EMPNO, DEPTMENT, DateStartedWithCompany, AnnualSalary", "EmplyeeDetails", "DEPTMENT, PositionType", False);

6.3 Lce3GLCall

Executes the specified 3GL program on the IBM i. Parameters can be passed to and from the 3GL program using associated LANSA Open functions.

BOOL	Lce3GLCall	(int	iSession,
		char far *	strProgram,
		char far *	strLibrary,
		Int	iParam)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
----------	--

- strProgram The name of the IBM i program. The length should not exceed PROGRAM400_LENGTH as defined in the LCOE.H header file.
- strLibrary The name of the IBM i program library. The length of strLibrary should not exceed LIBRARY_LENGTH as defined in the LCOE.H header file.
- iParam The number of highest first level parameter block specified in iParam of the preceding Lce3GLDefineParameter. For example, if you defined first level parameters of 0 through 5, iParam is set to 5.

Return Values

TRUE is returned if the session can execute the program.

FALSE is returned if an error occurs.

Notes

Portability
ConsiderationsLce3GLCall is only supported if your host system is an
IBM i. An error will be returned by the host if you make
a call to any other platform.

The Lce3GLCall function permits the passing of parameter blocks both to and from the called 3GL program.

Lce3GLDefineParameter is used to define the parameters before using this call. Lce3GLDefineParameter should define a contiguous range of first level

parameters.

Lce3GLGetValue is used to retrieve the parameters in returned parameter blocks. The Lce3GLDefineParameter function indicates whether there will be any returned parameters.

Your application must wait until the IBM i program has completed. Error messages will be routed back to your application.

• This function can only be executed after a session has been opened.

Tip

Use Lce3GLCall to re-use tried-and-true 3GL programs which perform complicated calculations.

Related Functions

6.4 Lce3GLDefineParameter6.5 Lce3GLGetValue

6.3.1 VB Example

Call program PROG001, passing a 4 character alpha parameter and receiving back an 80 character parameter.

```
Sub cmdCall_Click ()
Dim Prog as String
Dim Value as String
Prog$ = "PROG001"
Value$ = Text1.Text
```

```
' Define the parameters to pass
Ret% = Lce3GLDefineParameter(SessId%, 0, 0, "P", "A", 4, 0, Value$)
Ret% = Lce3GLDefineParameter(SessId%, 1, 0, "R", "A", 80, 0, RetData$)
```

```
'Call the 3GL program
Ret% = Lce3GLCall(SessId%, Prog$, "*LIBL", 1)
```

```
' Retrieve the parameter returned from the 3GL program
RetData$ = String(256, 0)
Ret% = Lce3GLGetValue(SessId%, 1, 0, RetData$)
Label2.Caption = RetData$
End Sub
```

6.3.2 C Examples

Pass a 3 digit packed number to a program called XXXX: Lce3GLDefineParameter (iSession, 0, 0, "P", "P", 3, 0, &(value)); Lce3GLCall (iSession, "XXXX", "*LIBL", 0);

The same, but also get a 7 digits packed number back again: Lce3GLDefineParameter (iSession, 0, 0, "P", "P", 3, 0, &(value)); Lce3GLDefineParameter (iSession, 1, 0, "R", "P", 7, 0, &NULL); Lce3GLCall (iSession, "XXXX", "*LIBL", 1); Lce3GLGetValue(iSession, 1, 0, &(value2));

Call IBM program QCMDEXC to execute a WRKSBSJOB QCMN function: pszFunction = "WRKSBSJOB QCMD"; sprintf(pszSize, "%d", strlen(pszFunction)) Lce3GLDefineParameter (iSession, 0, 0, "P", "A", 256, 0, pszFunction); Lce3GLDefineParameter (iSession, 1, 0, "P", "P", 15, 5, pszSize); Lce3GLCall (iSession, "QCMDEXC", "*LIBL", 1);

Call program YYYY that receives a single data structure made up of an A(10), a P(7,3) and a S(6,0). Get back all values returned in the data structure: Lce3GLDefineParameter (iSession, 0, 0, "B", "A", 10, 0, "ABCD"); Lce3GLDefineParameter (iSession, 0, 1, "B", "P", 7, 3, "45.67"); Lce3GLDefineParameter (iSession, 0, 2, "B", "S", 6, 0, "10194"); Lce3GLCall (iSession, "YYYY", "*LIBL", 0); Lce3GLGetValue(iSession, 0, 0, pszValue1); Lce3GLGetValue(iSession, 0, 1, pszDoubleValue); Lce3GLGetValue(iSession, 0, 2, pszLongValue);

6.4 Lce3GLDefineParameter

Defines the attributes of and optionally sets the current value of a parameter. These parameters are for use by the Lce3GLCall function.

BOOL	Lce3GLDefineParameter	(int	iSession,
		int	iparam,
		int	isubParam,
		char far *	strDirection,
		char far *	strDataType,
		int	ilength,
		int	idecimal,
		char far *	strValue)

Parameters

iSession The session identi	fier as returned by LceGetSessionId.
-----------------------------	--------------------------------------

- iparam The first level number of the parameter to be set. The maximum number of first level parameters is 20. The base level starts at zero, so this parameter will be an integer in the range 0 to 19.
- isubParam The number of the sub-parameter within this first level parameter. The maximum number of sub-parameters is 256, with the base number starting at zero. This parameter will be an integer in the range 0 to 255.

Sub-parameters for a first level parameter must be defined in separate Lce3GLDefineParameter calls in numerically ascending order, as this call resets all numerically higher subparameters to "undefined" and ensures that all numerically lower sub-parameters have been defined.

strDirection The direction/s in which the entire first level parameter block will be passed. Its value should be:

P = pass the parameter block to the 3GL program.

R = the parameter block is returned by the 3GL program.

B = it is passed in both directions.

This value will override any previous strDirection value supplied for this first level parameter number.

strDataType The data type of this sub-parameter and must be one of the following:

A = Alphanumeric

P = Packed

S = Signed

B = Binary

iLength The field length of this parameter.

The aggregate data length (iLength, except for "Packed") of all sub-parameters for any first level parameter block, must not exceed MAX_PARAM_LENGTH (i.e. the sum of the length of each sub-parameter for the specified block) as defined in the LCOE.H header file.

- iDecimal The number of decimal places of this parameter.
- strValue The string equivalent of the value to be passed to the 3GL program when strDirection is "P" or "B". It is ignored if strDirection is "R".

This parameter must be supplied and may consist of a NULL string (i.e. "").

An error will occur if the resulting value of a DBCS conversion is data longer than iLength.

If the parameter is defined as "Binary", the strValue "string" pointer passed in this parameter is used as the base to access exactly iLength bytes without regard for NULL string terminating characters.

Return Values

TRUE is returned if the parameter can be set.

FALSE is returned if an error occurs.

Notes

Portability Considerations

Lce3GLDefineParameter is only supported if your host system is an IBM i. If you make a call to any other platform an error will be returned by the host.

LANSA Open maintains a local parameter area which is used when passing parameters to and from the host using Lce3GLCall. On the IBM i, LANSA Open will call the 3GL program with pointers to each of the defined (and requested) blocks in the subsequent Lce3GLCall request.

- This function may be executed at any time after the session has been opened.
- This function performs no communication with the host.

Related Functions

6.3 Lce3GLCall 6.5 Lce3GLGetValue

6.4.1 Examples

See Lce3GLCall for examples of using this function.

6.5 Lce3GLGetValue

Retrieves the parameter value returned by the Lce3GLCall function.

BOOL	Lce3GLGetValue	(int	iSession,
		Int	iParam,
		Int	iSubParam,
		Char far *	strValue)

Parameters

iSession	The session identifier as	returned by	LceGetSessionId.

iParam The first level number of the parameter to be retrieved.

iSubParam The number of the sub-parameter within this first level parameter.

strValue The storage location to receive the returned value. The size of strValue should be at least iLength + 4 (as defined in Lce3GLDefineParameter).

Return Values

TRUE is returned if the session has got the parameter value.

FALSE is returned if an error occurs.

Notes

Portability
ConsiderationsLce3GLGetValue is only supported if your host system is
an IBM i. Unpredictable data will be returned by
Lce3GLGetValue if an error was returned by the host
from Lce3GLCall.

The Lce3GLGetValue function retrieves the value of a sub-parameter from a first level parameter block, following a call to Lce3GLCall, so that a parameter passed back by the called IBM i program may be used in the application.

The sub-parameter requested in this function must be in a first level parameter block which has been declared using an Lce3GLDefineParameter function with a strDirection of "R" (parameter block is return by the 3GL program) or "B" (parameter block is passed in both directions).

The sub-parameter data is converted to a string. If the parameter you requested is numeric, strValue will contain a string value representing the numeric value. You need to convert the string value to a numeric before using it in your application.

If the parameter is defined as "Binary", the strValue "string" pointer passed in this parameter is used as the base into which to return exactly iLength bytes without regard for NULL string terminating characters. (That is, NULL value bytes are copied and do not terminate the data).

If the parameter is defined as "Date", "Time" or "DateTime", strValue will contain the date and/or time string in the current locale format.

An error will be returned if the 3GL program returned data in the parameter block that is not suitable for conversion using the defined type, length and decimals.

- This function may be executed at any time after the session is opened.
- This function performs no communications to the host.

Related Functions

6.4 Lce3GLDefineParameter 6.3 Lce3GLCall

6.5.1 Examples

See Lce3GLCall for examples using this function.

6.6 LceASCIIToFloat

Converts a string containing a numeric into a floating point number.

VOID	LceASCIIToFloat	(char far *	StrNumericString,
		Float far *	pFloat)

Parameters

strNumericString	The source string containing the numeric value to be converted.
pFloat	Points to the target variable in which the value is to be returned.

Return Values

No return value.

Notes

LceASCIIToFloat is to be used when the development tool you are using does not provide the necessary conversion routines.

Most languages provide some type of conversion routine and it is recommended that you use the one provided by the application tool with which you are programming. Do not use this function if other routines are available.

For C application developers, this function is unnecessary since the standard C libraries supply conversion routines between strings and numeric variables.

• LceASCIIToFloat can be called at any time.

6.6.1 C Example

float Salary; LceASCIIToFloat((char far *) "32.456", (float *) &Salary);

6.6.2 Pseudocode Example

CALL LceASCIIToFloat("32.456", Salary)

6.7 LceASCIIToInteger

Converts a string containing numerics into a long integer number.

VOID	LceASCIIToInteger	(char far *	strNumericString,
		Long far *	pVal)

Parameters

strNumericString The source string containing the numeric value.

pVal Points to the long integer in which the value is to be returned.

Return Values

No return value

Notes

LceASCIIToInteger is to be used when the development tool you are using does not provide the necessary conversion routines.

Most languages provide some type of conversion routine and it is recommended that you use the one provided by the application tool with which you are programming. Do not use this function if other routines are available.

For C application developers, this function is unnecessary since the standard C libraries supply conversion routines between strings and numeric variables.

• This function can be called at any time.

6.7.1 C Example

LONG lAge; LceASCIIToInteger((char far *) "24", (long *) &lAge);

6.7.2 Pseudocode Example

CALL LceASCIIToInteger("24", lAge)

6.8 LceCheckFor

Checks for the existence of a record in the specified file using the keys specified.

BOOL	LceCheckFor	(int	iSession,
		char far *	strFileName,
		char far *	strKeyList)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strFileName	The name of the file. The file used must have been declared using an LceUseFile function.
strKeyList	A list of field names to be used as keys. The key fields must have been declared using the LceUseField function.

Return Values

TRUE is returned if the session can find the record.

FALSE is returned if an error occurs or the record does not exist.

Notes

Portability	An I/O error (Error 37) will result if you provide an
Considerations	empty string for the strKeyList parameter and your host
	system is a Windows, Linux or other non-IBM i host. If
	your host system is LANSA on an IBM i, no error will
	occur but the call will return FALSE.

This function is used to check if a record already exists in a file. You must know the specific key fields to the file. That is, the key list must match the actual file's keys. The values for the keys are set using the

 $\label{eq:linear} LceSetFieldValue/LCeSetFieldValueU\ function.$

You might use the LceCheckFor function when you wish to find out if a record exists but you do not want to fetch the record. For example, if checking if a file has related information on another file before deleting a record.

Tip

The LceInsert, LceUpdate and LceDelete functions have a validate parameter which allows a check to be made without performing the actual insert/update/delete operation. Use this validate parameter with these functions instead of this LceCheckFor function.

Alternatively, you can use the LANSA Repository to perform these referential integrity checks (for insert, update, and delete) provided that you have created the proper validation rules in the Repository. For example, LANSA will check that a customer number exists in the customer file before it creates a new order for that customer in the order file.

Related Functions

6.71 LceUseFile
6.58 LceSetFieldValue
LceSetFieldValueU
6.43 LceInsert
6.67 LceUnicodeToHex
6.10 LceDelete

6.8.1 VB Example

' if record doesn't already exist insert new

6.8.2 C Example

BOOL fOK; int iSession;

fOK = LceSetFieldValue(iSession, "EMPNO", "12345"); fOK = LceSetFieldValue(iSession, "DEPT", "SALES");

fOK = LceCheckFor(iSession, "PSLFIL", "EMPNO, DEPT");

Pseudocode Example

6.9 LceCommitWork

Commits the current transaction if LceSetCommitmentOn has been set.

BOOL LceCommitWork (int iSession)

Parameters

iSession The session identifier as returned by LceGetSessionId.

Return Values

TRUE is returned if the session can perform the function.

FALSE is returned if an error occurs.

Notes

This function is used to set a commit point for a transaction or set of transactions on the host.

- To use LceCommitWork, an LceSetCommitmentOn function must be executed before the session is opened.
- This function can only be executed after a session has been opened.

Commitment control allows you to recreate events from a specific point which would otherwise be lost due to a system or application failure. Host commitment control only applies to files for which it has been specified. It is your responsibility to ensure that commitment control has been properly configured on the host.

Refer to the appropriate host database manuals and Commitment Control in the *LANSA Application Design Guide* and Using Commitment Control in the *LANSA for iSeries User Guide* for more details about commitment control and commitment control processing.

Related Functions

6.54 LceSetCommitmentOff6.55 LceSetCommitmentOn6.52 LceRollBackWork

6.9.1 VB Example

iRet = LceSetCommitmentOn(iSession)
iRet = LceOpenSession(iSession) ' enable commit mode
...
if (bCommit) then
iRet = LceCommitWork (iSession)

else

iRet = LceRollBackWork (iSession)

6.9.2 C Example

BOOL fOK; int iSession;

iSession = LceGetSessionId fOK = LceSetCommitmentOn(iSession); fOK = LceOpenSession(iSession);

/* perform file updates */
fOK = LceCommitWork(iSession);

6.10 LceDelete

Deletes one or more records from the specified file using the specified keys.

BOOL	LceDelete	(int	iSession,
		char far *	strFileName,
		char far *	strKeyList,
		BOOL	fValidate)

Parameters

- iSession The session identifier as returned by LceGetSessionId.
- strFileName The name of the file. The file must have been declared using an LceUseFile function.
- strKeyList A list of key field names. The keys must always be specified in the correct order. These fields must have been declared using the LceUseField function.

You do not have to specify the complete key list. That is, you could list the first two fields in a file which has three key fields, however, if you use a subset of the key fields, you are implicitly using a "generic select" on the unspecified keys, i.e. match key1 and key2 while key3 can have any value. In this case, you are requesting that all records which match the partial key are deleted.

If you specify a unique key you will perform the delete of a single record.

fValidate This parameter provides a "check but do not delete" facility, if you wish to check that a record can be deleted before you perform the actual delete.

True = perform the validate only.

False = perform the delete immediately.

Return Values

TRUE is returned if the session can delete the record or has deleted the record.

FALSE is returned if an error occurs or the record does not exist.

Notes

When multiple users are working with a file, it is possible that two users will want to delete the same record. LANSA Open will handle this situation automatically if you leave the strKeyList blank. In this case, if a user attempts to delete a record which has already been deleted by another user, LANSA Open issues an appropriate message. Because LANSA Open does this check for you, record locking on the host is unnecessary.

• This function can only be executed after a session has been opened.

Tip

The LANSA Repository will perform referential integrity checks when deleting records, provided that you have added the relevant validation rules. For example, LANSA can check that a customer has no orders in the order file before allowing the customer record to be deleted from the customer file.

Related Functions

6.71 LceUseFile
6.58 LceSetFieldValue
6.59 LceSetFieldValueU
6.43 LceInsert
6.67 LceUnicodeToHex
6.8 LceCheckFor

6.10.1 VB Example

6.10.2 C Example

Delete a record from a file. BOOL fOK; int iSession;

fOK = LceSetFieldValue(iSession, "EMPNO", "12345")

fOK = LceDelete(iSession, "PSLMST", "EMPNO", FALSE);

6.10.3 Psuedocode Example

Check if record can be deleted. strName = "PSLMST" strKey = "EMPNO" IF NOT LceSetFieldValue(iSession, strKey, "12345") return error ENDIF IF LceDelete(iSession, strName, strKey, TRUE) delete of the record can be performed ELSE cannot delete the record display error messages ENDIF

6.11 LceDeleteSelect

LceDeleteSelect is used to break the loop when multiple records are retrieved from the host using LceReceiveNextX.

If the process is to finish normally, this function is not necessary.



Parameters

iSession The session identifier as returned by LceGetSessionId.

strFileName The name of the file.

Return Values

TRUE is returned if the session has been canceled.

FALSE is returned if an error occurs.

Notes

No other communications can be performed during the LceReceiveNextX loop process. The LceDeleteSelect does not cancel the data transfer. It must wait for all selected records to be transferred and then perform the delete.

• This function performs no communications to the host.

Related Functions

6.50 LceRequestSelect
6.49 LceReceiveSelect
6.53 LceSelect
6.51 LceRequestSelectWhere
6.48 LceReceiveNextX
6.64 LceSetSelectOptions

6.11.1 VB Example

' remove any previous select results not read.

' If any exist, the next select will fail.

iRet = LceDeleteSelect(iSession, "SECTAB")

' get the new data

- iRet = LceSetSelectOptions(iSession, "*RECEIVEIMMED")

6.11.2 C Example

BOOL fOK; int iSession;

fOK = LceDeleteSelect(iSession, "PSLMST");

6.11.3 Psuedocode Example

6.12 LceDisplayErrors

Specifies whether error message handling is to be automated, with error messages automatically displayed in a dialog box.

BOOL	LceDisplayErrors	(BOOL	fDisplay,
		HWND	hwnd)

Parameters

- fDisplay TRUE if LANSA Open is to display messages. FALSE if the application is to handle error messages.
- hwnd Handle of parent window for the error dialog box. This should be a valid window handle, usually the main window for the application. This parameter can be NULL if the application cannot supply a window handle that will always be valid. If NULL is specified, the active window at the time the error occurs will be used. Error messages will not be displayed if an invalid window handle is specified.

Return Values

TRUE if the function is successful.

FALSE is returned if an error occurs.

Notes

There are two types of error message within LANSA Open:

- 1. Messages sent by the host when a function results in errors.
- 2. Messages generated by the LANSA Open functions.

This function allows your application to use LANSA Open's automated error message display facilities, displaying both types of messages in a list box.

The default action is NOT to display messages.

If this function is not used or if fDisplay is set to FALSE, applications must handle error messages themselves using LceGetStatus and LceGetMessage.

Related Functions

6.40 LceGetStatus

6.34 LceGetIBMiSignon
6.12.1 VB Example

' enable default error handler iRet = LceDisplayErrors(LceTrue, 0)

•••

' disables default error handle iRet = LceDisplayErrors(LceFalse, 0)

' enable default error handler setting txtMsg in frmDispErr as the ' field to display error text load frmDispErr iRet = LceDisplayErrors(LceTrue, frmDispErr.txtMsg.hWnd)

6.12.2 C Example

BOOL fOK; fOK = LceDisplayErrors(TRUE, NULL);

6.12.3 Psuedocode Example

IF NOT LceDisplayErrors(TRUE, NULL) /* return error */ ENDIF

6.13 LceEndSession

Ends the specified session with the host.

BOOL	LceEndSession	(int	iSession,
		BOOL	fResume)

Parameters

iSession The session identifier as returned by LceGetSessionId.

fResume Close or keep the conversation open. Refer to Notes.

Return Values

TRUE is returned if the session has ended successfully.

FALSE is returned if an error occurs.

Notes

If fResume is set to TRUE, all the session information is de-allocated but the communications conversation is kept active. The given session identifier remains valid. The TRUE option allows the session to be opened again later with different fields, files etc., but without allocating another conversation.

If fResume is set to FALSE, then the session is ended and the communications conversation is de-allocated. At some point in your application, this function should always be called with fResume set to FALSE for every session opened, even if it has already been called with fResume set to TRUE.

Failure to terminate the session may result in a communications resource overhead on the host.

If you do not end the opened sessions, they may remain open on the host until they are terminated manually. When you are building and testing your applications, you must be careful to terminate sessions on the host which were not closed properly. For example, if your PC application fails and an LceEndSession was not executed, you may have to remember to terminate the session on the host manually.

6.13.1 VB Example

' The call below specifies fResume to be TRUE.

' This leaves the communications path open.

' All files and LANSA resources are disassociated from the

' session. The session Id is still valid.

If iSession > 0 Then

iRet = LceEndSession(iSession, LceTrue)

End If

••••

' At a later stage in the processing you may wish to re-

use the session. ' To do this you must re-open it.

' This will be fast because the underlying communications path already ' exists.

' Now you will start the processing for the new session.

' For example, you would set up a new set of files and fields for use

' over the session.

```
....
iRet = LceUseFile(iSession,"PLSMST")
iRet = LceOpenSession(iSession)
```

••••

' Finally, if you have no further need for the session,

' end it specifying fResume as FALSE.

' This will free all LANSA resources associated with it as well as

' de-allocating the underlying communications path.

iRet = LceEndSession(iSession, LceFalse)

6.13.2 C Example

End the session completely. BOOL fOK; int iSession;

LceEndSession(iSession, FALSE);

6.14 LceExecute400

Executes an IBM i command.

BOOL	LceExecute400	(int	iSession,
		char far *	strCommand)

Parameters

iSession The session identifier as returned by LceGetSessionId.

strCommand The IBM i command. For this entry, you must know the format of the command. You must ensure that the syntax of this parameter is correct and that the length does not exceed COMMAND400_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session has executed the command.

FALSE is returned if an error occurs.

Notes

Portability Considerations

LceExecute400 is only supported if your host system is an IBM i. An error will be returned by the host if you make a call to any other platform.

Any command, program or function initiated by this function must be batch oriented (i.e. have no screen interaction). Your application must wait until the command or function has completed before a return value is passed back to your application. Error messages will be routed back to your application.

• This function can only be executed after a session has been opened.

Tip

While LceExecute400 can execute a program or LANSA function on the IBM i, you will find it much easier to use the LceLansaCall, LceSubmit or Lce3GLCall function. Passing information with these functions is less complicated than it is with LceExecute400.

Related Functions

6.66 LceSubmit

6.3 Lce3GLCall 6.44 LceLansaCall

6.14.1 VB Example

6.14.2 C Example

Call a program on the IBM i. BOOL fOK; int iSession;

fOK = LceExecute400(iSession, "CALL PGM(LIBL/NAME)");

6.15 LceFetch

Retrieves the values of the specified fields for a single record from the specified file using the key list.

BOOL	LceFetch	(int	iSession,
		char far *	strFieldList,
		char far *	strFileName,
		char far *	strKeyList)

Parameters

- iSession The session identifier as returned by LceGetSessionId.
- strFieldList A list of field names to be retrieved from the file. Each field listed must have been declared using an LceUseField function.
- strFileName The name of the file. This file must have been declared using an LceUseFile function.
- strKeyList A list of field names to be used as keys. These key fields must have been declared using the LceUseField function. This list must match the sequence of the actual keys to the file, however, all file keys do not need to be specified. A partial key can be used. When a partial key is used, the first record matching the partial key will be returned.

Return Values

TRUE is returned if the session has fetched the record.

FALSE is returned if an error occurs.

Notes

The LceGetFieldValueX/ LceGetFieldValueU is used to obtain the values of the fields once this fetch has completed successfully.

For record locking, refer to the LceSetSelectOptions function.

• This function can only be executed after a session has been opened.

Warning: If you are selecting records and have used *RECEIVEIMMED on the LceSetSelectOptions function, you cannot perform a fetch in the same

session until all selected records have been read. If you need to read records from another file, you must either open another session or use a predetermined join field on the file on the host. Refer to the LceSetSelectOptions function for more details.

Related Functions

6.71 LceUseFile
6.58 LceSetFieldValue
6.59 LceSetFieldValueU
6.29 LceGetFieldValueX
6.30 LceGetFieldValueU
6.64 LceSetSelectOptions
6.8 LceCheckFor
6.50 LceRequestSelect
6.48 LceReceiveNextX

6.15.1 VB Example

```
Sub SetCurrentEmployee(EmpNo As String)
```

```
' Set Employee EmpNo as the active record
```

```
If EmpNo <> "" Then
```

```
' set the key value for this fetch
    iRet = LceSetFieldValue(iSession, "EMPNO", EmpNo)
    ' get the selected employee details
    If iRet = LceTrue Then
      iRet = LceFetch(iSession,
          "EMPNO,SURNAME,GIVENAME,ADDRESS1,ADDRESS2," &
          "ADDRESS3,POSTCODE,PHONEHME,PHONEBUS," &
          "DEPTMENT, SECTION, STARTDTE, TERMDATE, SALARY",
          "PSLMST", "EMPNO")
      If iRet = LceTrue Then
         ' Call GetFieldValues to retrieve values.
      End If
    End If
  Else
  ' Handle error
  End If
End Sub
```

```
Sub GetFieldValues()
Dim sBuff As String
Dim lFlags As Long
```

```
sBuff = String(FIELD_DATA_SIZE, Chr(0)) ' set buffer
```

```
' read field values and set text boxes
```

```
'EmpNo
iRet = LceGetFieldValueX(iSession, "EMPNO", sBuff, lFlags)
txtEmpNo.Text = sTrim(sBuff)
```

```
'Surname
iRet = LceGetFieldValueX(iSession, "SURNAME", sBuff, lFlags)
txtSurname.Text = sTrim(sBuff)
'Givename
iRet = LceGetFieldValueX(iSession, "GIVENAME", sBuff, lFlags)
txtGiven.Text = sTrim(sBuff)
...
'PHONEBUS
iRet = LceGetFieldValueX(iSession, "PHONEBUS", sBuff, lFlags)
if lFlags = 1 then
txtBusPhone.Text = """
Then
txtBusPhone.Text = sTrim(sBuff)
End If
End Sub
```

6.15.2 C Example

BOOLfOK;intiSession;unsigned longlFlags;

```
fOK = LceSetFieldValue(iSession, "EMPNO", "12345");
fOK = LceFetch(iSession, "SURNAME, GIVENAME", "PSLMST", "EMPNO")
if (fOK)
{
    fOK = LceGetFieldValueX(iSession, "SURNAME", strLastName, &lFlags)
    fOK = LceGetFieldValueX(iSession, "GIVENAME", strFirstName, &lFlags)
}
```

6.15.3 Psuedocode Example

6.16 LceGetDefaultValue

This function has been replaced by 6.17 LceGetDefaultValueX and is supported for backward compatibility only. It does not support RDMLX fields.

BOOL	LceGetDefaultValue	(int	iSession,
		char far *	strFieldName,
		char far *	strValue)

6.17 LceGetDefaultValueX

Retrieves the default value of a field, if it has been defined in the Repository. The retrieved default is fully evaluated. For example, if the field is the current date, the actual current date will be retrieved.

BOOL	LceGetDefaultValueX	(int	iSession,
		char far *	strFieldName,
		char far *	StrValue,
		unsigned long far *	pulFlags)

Parameters

iSession The session identifier as returned by LceGetSessionId.

- strFieldName The field name of a field previously declared with LceUseField.
- strValue Pointer to the array to receive the returned value. The field value is stored in this parameter once the function has executed successfully. It will be a zero length string if no default value is specified in the Data Dictionary. The size of strValue must be at least FIELD_DATA_SIZE + 1 as defined in the LCOE.H header file.
- pulFlags Identifies the returned value as *SQLNULL. If the field has a default value of *SQNULL pulFlags will return 1 otherwise it will return 0.

Return Values

TRUE is returned if the session has got the field's default value (or zero length string).

FALSE is returned if an error occurs.

Notes

The LceGetDefaultValueX function uses string parameters. Even if the field you are getting is numeric, it will contain a string value representing the default numeric value. You will need to convert the string value to a numeric before using it in your application. Note also that no edit code or word will be applied

and that any "decimal point" used (if appropriate) will be that which applies on the host. If the returned string is for a Date or Time, the format will be that of the current locale.

Use LceGetDefaultValueU for Unicode fields.

- This function may be executed at any time after the session is opened.
- This function communicates with the host for all initial calls for a given field name. If the field is defined as dynamic (i.e. the contents can change at any time) each subsequent call will retrieve the latest version of the field from the host.

Tip

Wherever possible, for data entry screens, default values should be defined in the Repository to reduce the amount of keying required.

Related Functions

LceGetDefaultValueU 6.29 LceGetFieldValueX 6.23 LceGetFieldHeading 6.24 LceGetFieldHelpText 6.25 LceGetFieldIndicator 6.26 LceGetFieldLabel 6.33 LceGetFileType

6.17.1 VB Example

' Declare variables.

Dim sBuff As String Dim iRet As Integer Dim lFlags As Long

' Associate fields with the session. Note that this only has to be ' done once for a particular field.

iRet = LceUseField(iSession, "DEPTMENT, DEPTDESC")

' Get the default value for Department Code. Note that the subroutine ' sTrim() removes leading and trailing blanks as well as any NULL ' terminator at the end of the string.

```
sBuff = String(FIELD_DATA_SIZE, Chr(0))
iRet = LceGetDefaultValueX(iSession, "DEPTMENT", sBuff, lFlags)
If lFlags = 1 Then
   txtCode.Text = "Null value"
Else
   txtCode.Text = sTrim(sBuff)
EndIf
```

' Get the default value for Department Description.

```
sBuff = String(FIELD_DATA_SIZE, Chr(0))
iRet = LceGetDefaultValueX(iSession, "DEPTDESC", sBuff, lFlags)
If lFlags = 1 Then
   txtDesc.Text = "Null value"
Else
   txtDesc.Text = sTrim(sBuff)
EndIf
```

6.17.2 C Example

BOOL fOK; int iSession; char strSurname[FIELD_DATA_SIZE + 1]; unsigned long ulFlags;

fOK = LceGetDefaultValueX(iSession, "SURNAME", (char far *) strSurname, &

6.17.3 Psuedocode Example

```
strValue = spaces(FIELD_DATA_SIZE + 1)
iValue = 0
lFlags = 0
IF LceGetDefaultValueX(iSession, "EMPNO", strValue, lFlags)
    IF NOT lFlags = 1
        iValue = ConvertStringToInteger (strValue)
        ENDIF
ELSE
        /* return error */
ENDIF
```

6.18 LceGetDefaultValueU

Retrieves the default value of a Unicode field, if it has been defined in the Repository. The retrieved default is fully evaluated. For example, if the field is the current date, the actual current date will be retrieved.

BOOL	LceGetDefaultValueX	(int	iSession,
		char far *	strFieldName,
		wchar_t far	strValue,
		int	iValueSize,
		unsigned long far *	pulFlags)

Parameters

- iSession The session identifier as returned by LceGetSessionId.
- strFieldName The field name of a field previously declared with LceUseField.
- strValue Pointer to the array to receive the returned value. The field value is stored in this parameter once the function has executed successfully. It will be a zero length string if no default value is specified in the Data Dictionary. The size of strValue should be at least the maximum size of the field.
- iValueSize The size (in bytes) of the strValue.
- pulFlags Identifies the returned value as *SQLNULL. If the field has a default value of *SQLNULL pulFlags will return 1 otherwise it will return 0.

Return Values

TRUE is returned if the session has got the field's default value (or zero length string).

FALSE is returned if an error occurs.

Notes

The LceGetDefaultValueU function uses string parameters. Even if the field you

are getting is numeric, it will contain a string value representing the default numeric value. You will need to convert the string value to a numeric before using it in your application. Note also that no edit code or word will be applied and that any "decimal point" used (if appropriate) will be that which applies on the host. If the returned string is for a Date or Time, the format will be that of the current locale.

If a non-Unicode field is passed to this function, it will call LceGetDefaultValueX and the returned value will be automatically converted into Unicode.

- This function may be executed at any time after the session is opened.
- This function communicates with the host for all initial calls for a given field name. If the field is defined as dynamic (i.e. the contents can change at any time) each subsequent call will retrieve the latest version of the field from the host.

Tip

Wherever possible, for data entry screens, default values should be defined in the Repository to reduce the amount of keying required.

Related Functions

6.29 LceGetFieldValueX
6.23 LceGetFieldHeading
6.24 LceGetFieldHelpText
6.25 LceGetFieldIndicator
6.26 LceGetFieldLabel
6.33 LceGetFileType

6.18.1 C Example

BOOL fOK; int iSession; char strDataType[FIELD_TYPE_LENGTH + 1]; FLDLEN length; int iDecimal; wchar_t *pwstrSurname; unsigned long ulFlags; long lDataLen;

fOK = LceGetFieldType(iSession, "SURNAME", strDataType, &length, &decimal); lDataLen = ((lDataLen + 1) * sizeof(wchar_t)); pwstrSurname = malloc(lDataLen); fOK = LceGetDefaultValueU(iSession, "SURNAME", pwstrSurname, lDataLen,

6.19 LceGetFieldAttributes

Retrieves the field's attributes. These attributes can be used to validate data before it is assigned to the field.

BOOL	LceGetFieldAttributes	(int	iSession,
		char far *	strFieldName,
		unsigned long far *	pulFlags)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strFieldName	The field name. The field must have been declared using an LceUseField function.
pulFlags	Indicates whether certain field attributes apply to the field. This field is treated as a bitmap. If the attribute applies to the field the corresponding bit is set to True (1).
	Bit 1 – Can the value be *SQLNULL Bit 2 – Is the Datetime value received from the server in UTC Bit 3 – Is the Integer value unsigned

Return Values

TRUE is returned if the session has got the field attributes. FALSE is returned if an error occurs.

Notes

- This function performs no communications to the host.
- This function must be executed after the session is opened.

Related Functions

- 6.27 LceGetFieldType
- 6.26 LceGetFieldLabel
- 6.23 LceGetFieldHeading
- 6.25 LceGetFieldIndicator
- 6.58 LceSetFieldValue

6.29 LceGetFieldValueX6.29 LceGetFieldValueX6.30 LceGetFieldValueU

6.19.1 VB Example

```
Private Sub lstSkills_Click()
```

' This subroutine sets the fields displayed to the details of the

' list's currently selected skill

Dim iRet As Integer Dim sBuff As String **Dim lFlags As Long** Dim lDataLen As Long Dim fSQLNullEnab As Boolean iRet = LceGetFieldAttributes(iSession, "SKILDESC", lFlags) If (IFlags And LCE FLDATTR SQLNULL) Then fSQLNullEnab = True Else fSQLNullEnab = False End If iRet = LceSetFieldValue(iSession, "SKILCODE", txtCode.Text) If txtDesc.Text <> "" Then iRet = LceSetFieldValue(iSession, "SKILDESC", txtDesc.Text) Else If fSQLNullEnab = True Then ' "" is not a null terminated string iRet = LceSetFieldValue(iSession, "SKILDESC", vbNullString) Else ' Display message to enter a valid description Exit Sub End If End If iRet = LceInsert(iSession, "SKILCODE, SKILDESC", "SKLTAB", LceFalse) End If End Sub

6.19.2 C Example

BOOL fOK; int iSession; int iRet; long lFlags; BOOL fSQLNullEnab;

```
fOK = LceGetFieldAttributes(iSession, " SKILDESC ", &lFlags);
fSQLNullEnab = (lFlags & LCE_FLDATTR_SQLNULL);
iRet = LceSetFieldValue(iSession, "SKILCODE", strCode);
if (fSQLNullEnab && ( strcmp(strDesc, "") == 0 ) )
```

iRet = LceSetFieldValue(iSession, "SKILDESC", Null);

else

iRet = LceSetFieldValue(iSession, "SKILDESC", strDesc);

iRet = LceInsert(iSession, "SKILCODE,SKILDESC", "SKLTAB", False);

6.19.3 Psuedocode Example

strFieldName = "" lFlags = 0

LceGetFieldDataLength(iSession, strFieldName, lFlags)

IF lFlags And LCE_FLDATTR_SQLNULL SQLNullEnabled = True ELSE SQLNullEnabled = False ENDIF

IF lFlags And LCE_FLDATTR_DATEINUTC DateInUtc = True ELSE DateInUtc = False ENDIF

IF lFlags And LCE_FLDATTR_UNSIGNED SQLNullEnabled = True ELSE SQLNullEnabled = False ENDIF

6.20 LceGetFieldDataLength

Retrieves the data length for the field's current value.

BOOL	LceGetFieldDataLength	(int	iSession,
		char far *	strFieldName,
		FLDLEN far *	pLength)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strFieldName	The field name. The field must have been declared using an LceUseField function.
pLength	Pointer to the variable to receive the field length. This indicates the number of characters or digits.

Return Values

TRUE is returned if the session has got the field data length.

FALSE is returned if an error occurs.

Notes

- This function performs no communications to the host.
- This function must be executed after the session is opened.
- If the field is currently *SQLNULL, the length will be returned as zero.

Related Functions

6.27 LceGetFieldType 6.26 LceGetFieldLabel

- 6.23 LceGetFieldHeading
- 6.25 LceGetFieldIndicator

6.20.1 VB Example

```
Private Sub lstSkills_Click()
  ' This subroutine sets the fields displayed to the details of the
  ' list's currently selected skill
  Dim iRet As Integer
  Dim sBuff As String
  Dim lFlags As Long
  Dim lDataLen As Long
  If (lstSkills.ListIndex > -1) Then ' selected
     ' Read record
    ' set key
    iRet = LceSetFieldValue(iSession, "SKILCODE", _
         SkillKey(lstSkills.ListIndex))
    ' fetch record
    iRet = LceFetch(iSession, "SKILCODE, SKILDESC", "SKLTAB",
         "SKILCODE") ' Get Record
    ' get data and set fields
    ' get code
    iRet = LceGetFieldDataLength(iSession, "SKILCODE", lDataLen)
    sBuff = String(IDataLen + 1, Chr(0))
    iRet = LceGetFieldValueX(iSession, "SKILCODE", sBuff, lFlags) 'Get Fiel
    txtCode = sTrim(sBuff) ' display data
    ' get description
    iRet = LceGetFieldDataLength(iSession, "SKILDESC", lDataLen)
    sBuff = String(IDataLen + 1, Chr(0))
    iRet = LceGetFieldValueX(iSession, "SKILDESC", sBuff, lFlags)
    txtDesc = sTrim(sBuff)
  End If
End Sub
```

6.20.2 C Example

BOOL fOK; int iSession; char *pstrSurname; long lFlags; long lDataLen;

fOK = LceGetFieldDataLength(iSession, "SURNAME", &lDataLen);
pstrSurname = malloc(lDataLen + 1);
fOK = LceGetFieldValueX(iSession, "SURNAME", (char far *) pstrSurname, &l

6.20.3 Psuedocode Example

strValue = "" iValue = 0 lFlags = 0 lDataLen = 0

LceGetFieldDataLength(iSession, "EMPNO", lDataLen) /* Allocate buffer for field value */ LceSetFieldValue(iSession, "EMPNO", szBuffer)

6.21 LceGetFieldDesc

Retrieves the field description from the Repository.

BOOL	LceGetFieldDesc	(int	iSession,
		char far *	strFieldName,
		char far *	strValue)

Parameters

iSession	The session identifier as returned by LceGetSessionId.

strFieldName The field name. The field must have been declared using an LceUseField function.

strValue The buffer location to receive the returned value. The size should be at least FIELD_DESC_LENGTH + 1 as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session has got the field description.

FALSE is returned if an error occurs.

Notes

If the partition is multilingual, the field description will be based on the language defined when the session is open.

- This function can only be executed after the session is opened.
- This function performs no communications to the host.

Tip

Always get your field descriptions from the Repository rather than hard code them in your PC application. By retrieving descriptions from the Repository, they will be consistent across both PC and host applications and, for multilingual applications, will be in the language required.

Related Functions

```
6.26 LceGetFieldLabel6.23 LceGetFieldHeading6.25 LceGetFieldIndicator
```

6.27 LceGetFieldType6.73 LceUseLanguage6.17 LceGetDefaultValueX6.18 LceGetDefaultValueU6.24 LceGetFieldHelpText
6.21.1 VB Example

Private Sub txtCode_GotFocus()

- ' sets txtCode as the current field
- ' sets form status line to contain the current field's long
- ' description
- ' sets the current field name to use with "get extended help"

Dim sBuff As String, iRet As Integer sBuff = String(FIELD_DATA_SIZE, Chr(0)) ' pre-allocate memory

' set form status line to contain the current field long description iRet = LceGetFieldDesc(iSession, "SKILCODE", sBuff) frmMain.status2 = (sTrim(sBuff))

' store the current field name to use with "get extended help" sLastField = "SKILCODE" ' for help

6.21.2 C Example

BOOL fOK; int iSession; char strDescription[FIELD_DESC_LENGTH + 1];

fOK = LceGetFieldDesc(iSession, "SURNAME", (char far *) strDescription);

6.21.3 Psuedocode Example

6.22 LceGetFieldEdit

Retrieves the field edit code **or** word specified for a field in the LANSA Repository.

BOOL	LceGetFieldEdit	(int	iSession,
		char far *	strFieldName,
		char far *	strEdit,
		char far *	strFlag)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strFieldName	The field name. This must have been declared using an LceUseField function.
strEdit	The array to receive the returned edit code or word. The size of strEdit should be at least FIELD_EDIT_LENGTH + 1 as defined in the LCOE.H header file.
	Returned strFlag
	W = strEdit will contain an "Edit Word" within single quotes " ' ".
	E = strEdit will contain a single character which is the "Edit Code".
	N = no edit code or word was defined and the contents of strEdit are undefined.
strFlag	The array to receive a flag indicating the edit type in strEdit. The size of strFlag should be at least FIELD_EDIT_FLAG_LENGTH + 1. E = the field uses an edit CODE.
	W = the field uses an edit WORD.
	N = the field has no edit specified.

Return Values

TRUE is returned if the session has got the field's edit code or word. FALSE is

returned if an error occurs.

Notes

Detailed information regarding the use of edit codes and words may be found in the *IBM i DDS Reference Manual*.

- This function performs no communications to the host.
- This function can only be executed after the session is opened.

Related Functions

- 6.26 LceGetFieldLabel
- 6.23 LceGetFieldHeading
- 6.27 LceGetFieldType
- 6.65 LceSetSessionOption

6.22.1 VB Example

Function LceSetValue(iSession As Integer, sFldName As String, _ sStrValue As String) As Integer

' this function is a replacement for LceSetFieldValue to be

- ' used when you have to strip edit codes from date and value fields
- ' it also replaces empty fields "" with null terminated strings

```
Dim sFlag As String
Dim sEdit As String
Dim sBuff As String
Dim sValue As String
Dim i As Integer
Dim iRet As Integer
If sStrValue = "" Then
  ' null nothing to do
  iRet = LceSetFieldValue(iSession, sFldName, Chr(0))
Else
  ' Get Field edit type to remove formatting characters if required
  sEdit = String(FIELD EDIT LENGTH, 0)
  sFlag = String(FIELD EDIT FLAG LENGTH, 0)
  iRet = LceGetFieldEdit(iSession, sFldName, sEdit, sFlag)
  If iRet = LceTrue Then
     Select Case sFlag
       Case "E"
         sValue = sStrValue ' copy value
         sEdit = sTrim(sEdit)
         Select Case sEdit
            Case "Y" '
              ' date so Remove date Separator
              i = InStr(sValue, sDtSep)
              While i > 0
                sValue = Left(sValue, i - 1) + _
                      Mid(sValue, i + 1, 99)
                i = InStr(sValue, sDtSep)
              Wend
```

```
Case "1" '
                 ' currency remove Thousand separator
                 i = InStr(sValue, sTwSep)
                 While i > 0
                   sValue = Left(sValue, i - 1) + _
                        Mid(sValue, i + 1, 99)
                   i = InStr(sValue, sTwSep)
                 Wend
            End Select
            ' Save field
            iRet = LceSetFieldValue(iSession, sFldName, sValue)
         Case "W"
         Case "N"
            ' nothing to do
            iRet = LceSetFieldValue(iSession, sFldName, _
                              sStrValue)
       End Select
    End If
  End If
  ' set return code
  LceSetValue = iRet
End Function
```

6.22.2 C Example

BOOL fOK; int iSession; char strEdit[FIELD_EDIT_LENGTH + 1]; char strFlag[FIELD_EDIT_FLAG_LENGTH + 1];

fOK = LceGetFieldEdit(iSession, "SURNAME", (char far *) strEdit, (char far *)

6.22.3 Psuedocode Example

6.23 LceGetFieldHeading

Retrieves a field's column headings from the Repository.

BOOL	LceGetFieldHeading	(int	iSession,
		char far *	strFieldName,
		int	iHeading,
		char far *	strValue)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strFieldName	The field name. The field must have been declared using an LceUseField function.
iHeading	The heading level (1, 2 or 3). Heading levels are the equivalent of the Column Heading lines in the Repository.
strValue	The array to receive the returned value. The size should be at least FIELD_HEADING_LENGTH + 1 as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session has got the field heading.

FALSE is returned if an error occurs.

Notes

A field's column heading can be stored in the Repository in up to three separate lines. The three lines (or levels) make up the complete column heading. For example, the name for the Corporate Customer Number might be stored as:

Corporate

Customer

Number

If you need a shorter field description, use either of these functions:

• LceGetFieldLabel to get the field's label

• LceGetFieldDesc to get the field's description.

To see how these lines are recorded, refer to Building the LANSA Repository - Fields in the *Introduction to LANSA for iSeries* Guide.

- This function performs no communications to the host.
- This function can only be executed after the session is opened.

Tip

Use this function to get the fields' column headings from the Repository rather than hard coding them in your applications. By retrieving the Repository information, all your PC applications will be consistent and your maintenance will be reduced.

If the partition is multilingual, the field's column headings will be based on the language defined when the session was opened.

Related Functions

6.71 LceUseFile
6.21 LceGetFieldDesc
6.27 LceGetFieldType
6.73 LceUseLanguage
6.17 LceGetDefaultValueX
6.18 LceGetDefaultValueU

6.23.1 VB Example

```
' set Surname column headings using LANSA Repository information
```

```
dim sBuff as string

sBuff = String(FIELD_HEADING_LENGTH, Chr(0))

iRet = LceGetFieldHeading(iSession, "SURNAME", 1, sBuff)

lbHead1.Caption = sTrim (sBuff)

iRet = LceGetFieldHeading(iSession, "SURNAME", 2, sBuff)

lbHead2.Caption = sTrim (sBuff)

iRet = LceGetFieldHeading(iSession, "SURNAME", 3, sBuff)

lbHead3.Caption = sTrim (sBuff)
```

6.23.2 C Example

BOOL fOK; int iSession; char strHeading[FIELD_HEADING_LENGTH + 1];

fOK = LceGetFieldHeading(iSession, "SURNAME", 2, strHeading);

6.23.3 Psuedocode Example

6.24 LceGetFieldHelpText

Retrieves the Help Text for a field from the LANSA Repository and either returns it in the specified buffer or displays it in a window.

BOOL	LceGetFieldHelpText	(int	iSession,
		char far *	strFieldName,
		char far *	strType,
		char far *	strEOL,
		char far *	strHelpBuf,
		long	lHelpBufLen)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strFieldName	The field name. The field must have been declared using an LceUseField function.
strType	The type of help requested. U = Return User help text. D = Return Developer & User help text.
strEOL	The formatting characters to be added to the end of each help text line. You may use this parameter to specify how carriage returns and line feeds are to be inserted (e.g. "\n\r"). A zero length string indicates no characters are to be inserted. The size of strEOL should not exceed HELP_EOL_SIZE as defined in the LCOE.H header file.
strHelpBuf	The buffer to receive the returned text.
lHelpBufLen	The size of the buffer. Help Text is potentially very large. If you specify a buffer length of zero, LANSA Open will display a dialogue box containing the returned help text, up to the maximum size permitted by the operating system. If you specify a size, it will limit the amount of text returned. In this case, the additional text is truncated.

Return Values

TRUE is returned if the session has got the help text.

FALSE is returned if an error occurs.

Notes

The text returned does not include any host display attributes. To maintain consistency with the text as displayed on a host terminal, these attributes are converted to spaces. This means that there will be 2 additional spaces at the beginning of each line. This is also the case when the help text is presented on a 5250 terminal.

The font and character size used to present the help text is specified in the Configuration File, described in Configuration File.

• This function may only be executed after the session is opened.

Tip

Use this function to retrieve the LANSA Repository's Help text, rather than record it in a PC file or hard code it in a program. This will ensure that your PC and host applications are consistent and will reduce your on-going application maintenance. If the partition is multilingual, the help text will be based on the language defined when the session is opened.

Related Functions

6.26 LceGetFieldLabel
6.23 LceGetFieldHeading
6.25 LceGetFieldIndicator
6.27 LceGetFieldType
6.73 LceUseLanguage

6.24.1 VB Example

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)

```
' This function displays LANSA Repository Help text when the user ' presses F1 in a form
```

Dim sBuff As String

If KeyCode = vbKeyF1 Then ' f1 pressed

```
If sLastField <> "" Then ' focus is in a field
' Display field help text
```

```
sBuff = String(2049, Chr(0)) ' set buffer
```

iRet = LceGetFieldHelpText(iSession, _

```
sLastField, "U", Chr(13), sBuff, 2048)

' display help text

Call ShowText("Data Entry Help", sTrim(sBuff), _

"", "", "") ' see util.bas

End If

End If

End Sub
```

6.24.2 C Example

BOOL fOK; int iSession; char strHelpText[4096 + 1];

6.25 LceGetFieldIndicator

Retrieves the DBCS (Double Byte Character Set) flag from the LANSA Repository for non-Unicode string fields such as Alpha, Char and String. It is irrelevant for other filed types including Unicode fields (NCHAR, NVARCHAR).

BOOL	LceGetFieldIndicator	(int	iSession,
		char far *	strFieldName,
		char far *	strValue)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
100331011	The session rachance as retained by Lee Getsessionia.

- strFieldName The field name. The field must have been declared using an LceUseField function.
- strValue The array to receive the returned value. The size should be at least FIELD_IND_LENGTH + 1.
 - Y = the field uses DBCS characters.
 - N = the field uses single byte characters.

Return Values

TRUE is returned if the session has got the field indicator. FALSE is returned if an error occurs.

Notes

- Non-string fields and Unicode code fields will return 'N'.
- This function performs no communications to the host.
- This function can only be executed after the session is opened.

Related Functions

6.26 LceGetFieldLabel6.23 LceGetFieldHeading6.27 LceGetFieldType

6.25.1 VB Example

Sub TranslateField(sFieldName as string, sValue as string)

' if required this function translates an IBM i Double Byte' Character string into a Unicode string using a hypothetical' Translate DCB function

Dim sInd as String

sInd = String (FIELD_IND_LENGTH, chr(0))
' get field indicator
If LceGetFieldIndicator(iSession, sFieldName, strInd) = LceTrue Then
If strInd ="Y" then ' is a DBC
Call TranslateDBC (sValue) ' call translator
End If
End if
End Sub

6.25.2 C Example

BOOL fOK; int iSession; char strIndicator[FIELD_IND_LENGTH + 1];

fOK = LceGetFieldIndicator(iSession, "SURNAME", (char far *) strIndicator);

6.25.3 Psuedocode Example

6.26 LceGetFieldLabel

Retrieves the field's label from the LANSA Repository.

BOOL	LceGetFieldLabel	(int	iSession,
		char far *	strFieldName,
		char far *	strValue)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
----------	--

strFieldName The field name. The field must have been declared using an LceUseField function.

strValue Pointer to the array to receive the returned value. The size should be at least FIELD_LABEL_LENGTH + 1 as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session has got the field label.

FALSE is returned if an error occurs.

Notes

- This function performs no communications to the host.
- This function can only be executed after the session is opened.

Tip

Using the Repository's label rather than coding it in your application will ensure that your PC and host applications are consistent and your maintenance is reduced.

If the partition is multilingual, the label will be based on the language defined when the session is open.

If your application needs a longer name, instead of using this function, use the LceGetFieldHeading function to get the field's 3 line Column headings from the Repository.

Related Functions

6.21 LceGetFieldDesc

6.23 LceGetFieldHeading
6.27 LceGetFieldType
6.25 LceGetFieldIndicator
6.73 LceUseLanguage
6.17 LceGetDefaultValueX
6.18 LceGetDefaultValueU

6.26.1 VB Example

Private Sub Form_Load()

' initializes the form setting field labels and sizes

Dim iRet As Integer, sBuff As String, length As Integer Dim decimals As Integer

```
. . .
'read and set field labels
'Code
sBuff = String(FIELD DATA SIZE, Chr(0))
iRet = LceGetFieldLabel(iSession, "SKILCODE", sBuff)
lbCode.Caption = sTrim(sBuff)
'Desc
sBuff = String(FIELD_DATA_SIZE, Chr(0))
iRet = LceGetFieldLabel(iSession, "SKILDESC", sBuff)
lbDesc.Caption = sTrim(sBuff)
. . .
sBuff = String(FIELD DATA SIZE, Chr(0))
iRet = LceGetFieldType(iSession, "SKILCODE", sBuff, length,
decimals)
txtCode.MaxLength = length
'note this is valid only if no edit codes are used
```

End Sub

6.26.2 C Example

BOOL fOK; int iSession; char strLabel[FIELD_LABEL_LENGTH + 1];

fOK = LceGetFieldLabel(iSession, "SURNAME", strLabel);

6.26.3 Psuedocode Example

6.27 LceGetFieldType

Retrieves the field's type.

BOOL	LceGetFieldType	(int	iSession,
		char far *	strFieldName,
		char far *	strDataType,
		FLDLEN far *	pLength,
		int far *	piDecimal)

Parameters

- iSession The session identifier as returned by LceGetSessionId.
- strFieldName The field name. The field must have been declared using an LceUseField function.
- strDataType The variable which will receive the Data Type. The size of strDataType should be at least FIELD_TYPE_LENGTH + 1.
 - A = Alpha
 - P = Packed
 - S = Signed.
 - H = String, Char
 - E = Date
 - M = Time
 - Z = DateTime
 - F = Float
 - I = Integer
 - Y = Boolean
 - 8 = NChar
 - 9 = NVarChar

pLength Pointer to the variable to receive the field length. This indicates the number of characters or digits.

piDecimal Pointer to the variable to receive the number of decimal places. Only used with numeric fields.

Return Values

TRUE is returned if the session has got the field type. FALSE is returned if an error occurs.

Notes

- This function performs no communications to the host.
- This function must be executed after the session is opened.

Related Functions

6.21 LceGetFieldDesc

- 6.26 LceGetFieldLabel
- 6.23 LceGetFieldHeading
- 6.25 LceGetFieldIndicator

6.17 LceGetDefaultValueX

6.18 LceGetDefaultValueU

6.27.1 VB Example

iRet = LceGetFieldType(iSession, "SKILCODE", sBuff, length,_ decimals) txtCode.MaxLength = length ' note this is valid only if no _

' edit codes are used

6.27.2 C Example

BOOL fOK; int iSession; char strDataType[FIELD_TYPE_LENGTH + 1]; FLDLEN length; int iDecimal;

fOK = LceGetFieldType(iSession, "SURNAME", strDataType, &length, &iDecir

6.27.3 Psuedocode Example

strDataType = "" iLength = 0 iDecimal = 0

IF NOT LceGetFieldType(iSession, "SURNAME", strDataType, iLength, iDecim /* return error */

ENDIF

6.28 LceGetFieldValue

This function has been replaced by 6.29 LceGetFieldValueX and is supported for backward compatibility only. It does not support RDMLX fields.

BOOL	LceGetFieldValue	(int	iSession,
		char far *	strFieldName,
		char far *	strValue)

6.29 LceGetFieldValueX

Retrieves the value of a field.

BOOL	LceGetFieldValueX	(int	iSession,
		char far *	strFieldName,
		char far *	srValue,
		unsigned long far *	pulFlags)

Parameters

iSession	The session identifier as returned by LceGetSessionId.

- strFieldName The field name. The field used in this function must have been declared using an LceUseField function.
- strValue The buffer location for the returned value. The field value is stored here once the function has executed successfully. The size should be at least the length of the data + 1. The data length may be obtained using LceGetFieldDataLength. The maximum field size is RDMLX_FIELD_DATA_SIZE as defined in the LCOE.H header file.
- pulFlags Identifies the returned value as *SQLNULL. If the field has a value of *SQNULL pulFlags will return 1 otherwise it will return 0.

Return Values

TRUE is returned if the session has got the field value.

FALSE is returned if an error occurs.

Notes

This function uses string parameters. Even if the field you are getting is numeric or date/time, it will contain a string value representing the respective numeric or date/time value. You will need to convert the string value to a numeric or date/time before using it in your application. This conversion should be done using the facilities provided with your development toolset. If none are provided, refer for example to the LceASCIIToFloat and LceASCIIToInteger functions.

If this function is used for Unicode fields, only characters that can be displayed in the current codepage will be translated correctly. Characters which cannot be translated will appear as question marks ("?"). To avoid this restriction use LceGetFieldValueU.

- This function may be executed at any time after the session is opened.
- This function performs no communications to the host.

Related Functions

- 6.30 LceGetFieldValueU
- 6.58 LceSetFieldValue
- 6.59 LceSetFieldValueU
- 6.20 LceGetFieldDataLength

6.29.1 VB Example

```
Private Sub lstSkills_Click()
  ' This subroutine sets the fields displayed to the details of the
  ' list's currently selected skill
  Dim iRet As Integer
  Dim sBuff As String
  Dim lFlags As Long
  Dim lDataLen As Long
  If (lstSkills.ListIndex > -1) Then ' selected
    ' Read record
    ' set key
    iRet = LceSetFieldValue(iSession, "SKILCODE", _
        SkillKey(lstSkills.ListIndex))
    ' fetch record
    iRet = LceFetch(iSession, "SKILCODE, SKILDESC", "SKLTAB",
        "SKILCODE") ' Get Record
    ' get data and set fields
    ' get code
    iRet = LceGetFieldDataLength(iSession, "SKILCODE", lDataLen)
    sBuff = String(lDataLen + 1, Chr(0))
    iRet = LceGetFieldValueX(iSession, "SKILCODE", sBuff, lFlags) ' Get Fiel
    If lFlags = 1 Then
      txtCode.Text = "Null value"
    Else
      txtCode.Text = sTrim(sBuff)
    EndIf
    ' get description
    sBuff = String(RDMLX FIELD DATA SIZE + 1, Chr(0))
    iRet = LceGetFieldValueX(iSession, "SKILDESC", sBuff, lFlags)
    If lFlags = 1 Then
      txtDesc.Text = "Null value"
    Else
      txtDesc.Text = sTrim(sBuff)
```
EndIf End If End Sub

6.29.2 C Example

BOOL fOK; int iSession; char *pstrSurname; long lFlags; long lDataLen;

```
fOK = LceGetFieldDataLength(iSession, "SURNAME", &lDataLen);
pstrSurname = malloc(lDataLen + 1);
fOK = LceGetFieldValueX(iSession, "SURNAME", (char far *) pstrSurname, &l
if (lFlags == 1)
{
    realloc(pStrSurname, 21);
    strcpy(pStrSurname, "Surname Not Provided");
```

strcpy(pStr5

}

6.29.3 Psuedocode Example

strValue = "" iValue = 0 lFlags = 0 lDataLen = 0

LceGetFieldDataLength(iSession, "EMPNO", lDataLen) /* Allocate buffer for field value */ LceGetFieldValueX(iSession, "EMPNO", szBuffer, lFlags)

6.30 LceGetFieldValueU

Retrieves the value of a Unicode field. Unicode fields are field types NCHAR and NVARCHAR.

BOOL	LceGetFieldValueU	(int	iSession,
		char far *	strFieldName,
		wchar_t far	strFieldValue,
		int	iFieldValueSize,
		unsigned long far *	pulFlags)

Parameters

strFieldName The field name. The field used in this function must have been declared using an LceUseField function.

- strFieldValue The buffer location for the returned value. The field value is stored here once the function has executed successfully. The size should be at least the length of the data in bytes plus the width of Unicode character. The data length may be obtained using LceGetFieldDataLength (see notes). The maximum field size is RDMLX_FIELD_DATA_SIZE as defined in the LCOE.H header file.
- iFieldValueSize The size (in bytes) of strFieldValue.
- pulFlags Identifies the returned value as *SQLNULL. If the field has a value of *SQNULL pulFlags will return 1 otherwise it will return 0.

Return Values

TRUE is returned if the session has got the field value.

FALSE is returned if an error occurs.

Notes

This function uses string parameters. Even if the field you are getting is numeric

or date/time, it will contain a string value representing the respective numeric or date/time value. You will need to convert the string value to a numeric or date/time before using it in your application. This conversion should be done using the facilities provided with your development toolset. If none are provided, refer for example to the LceASCIIToFloat and LceASCIIToInteger functions.

If a non-Unicode field is passed to this function, it will call LceGetFieldValueX and the returned value will be automatically converted into Unicode.

If using LceGetFieldDataLength to determine the size of the buffer, remember that LceGetFieldDataLength returns the numbers of characters. When allocating the buffer for Unicode and DBCS fields, you may need to adjust the length if the function you are using to allocate the buffer requires the length in bytes rather than characters. To convert characters to bytes you must multiply the number of characters by the size of a Unicode character (normally 2 bytes). For example in C use sizeof(wchar_t) to get the size of a Unicode character.

- This function may be executed at any time after the session is opened.
- This function performs no communications to the host.

Related Functions

6.59 LceSetFieldValueU6.20 LceGetFieldDataLength

6.30.1 C Example

BOOL fOK; int iSession; wchar_t *pwstrSurname; long lFlags; long lDataLen;

```
fOK = LceGetFieldDataLength(iSession, "SURNAMEU", &lDataLen);
lDataLen = ( (lDataLen + 1) * sizeof(wchar_t) );
pwstrSurname = malloc(lDataLen);
fOK = LceGetFieldValueU(iSession, "SURNAME", pwstrSurname,
lDataLen, &lFlags);
```

```
if (lFlags == 1)
{
    realloc(pStrSurname, 21);
    strcpy(pStrSurname, "Surname Not Provided");
}
```

6.30.2 Psuedocode Example

strValue = "" iValue = 0 lFlags = 0 lDataLen = 0

LceGetFieldDataLength(iSession, "EMPNO", lDataLen) /* Allocate buffer for field value */ LceGetFieldValueX(iSession, "EMPNO", szBuffer, lFlags)

6.31 LceGetFieldValueH

Sets the text of the given window handle (hwndValue).

BOOL	LceGetFieldValueH	(int	iSession,
		char far *	strFieldName,
		HWND	hwndValue)

Parameters

iSession The session identifier returned by LceGetSessionId.

strFieldName The field name.

hwndValue The handle to a window which will receive the value.

Return Values

TRUE is returned if the session has got the field value.

FALSE is returned if an error occurs.

Notes

This function is similar to an LceGetFieldValueX or LceGetFieldValueU except that it allows you to retrieve the value of the field into a window handle value, i.e. a window field.

Refer to the LceGetFieldValueX or LceGetFieldValueU for details of how this function works. The field will be blank if the value is *SQLNULL.

- LceGetFieldValueH must be executed after the session is opened and data has been retrieved from the host.
- This function performs no communications to the host.

Related Functions

6.29 LceGetFieldValueX LceGetFieldValueU

6.31.1 VB Example

iRet = LceGetFieldValueH(iSession, "SURNAME", _
 frmDetails.txtSurname.hWnd)

6.31.2 C Example

BOOL fOK; int iSession; HWND hwndField;

fOK = LceGetFieldValueH(iSession, "SURNAME", hwndField);

6.32 LceGetFileDesc

Retrieves the file description from the Repository.

BC	DOL	LceGetFileDesc	(int	iSession,
			char far *	strFileName,
			char far *	strValue)

Parameters

iSession	The session identifier returned by LceGetSessionId.

strFileName The file name. This file must have been declared using an LceUseFile function.

strValue The variable which will receive the returned value. The size should be at least FILE_DESC_LENGTH + 1 as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session has retrieved the file description.

FALSE is returned if an error occurs.

Notes

- This function performs no communications with the host.
- This function must be executed after the session is opened.

Tip

By retrieving the description from the LANSA Repository rather than including it in your PC application, your PC descriptions will be consistent with your host applications and your maintenance will be reduced. If the partition is multilingual, the description will be in the required language.

Related Functions

6.71 LceUseFile6.33 LceGetFileType6.73 LceUseLanguage

6.32.1 VB Example

dim sDescription as String

' get the LANSA Repository Description to use in list label

sDescription = string (FILE_DESC_LENGTH + 1 ,chr(0))
iRet = LceGetFileDesc(iSession, "PSLMST", sDescription)
lbFileList.caption = sTrim (sDescription)

6.32.2 C Example

BOOL fOK; int iSession; char strDescription[FILE_DESC_LENGTH + 1];

fOK = LceGetFileDesc(iSession, "PSLMST", strDescription);

6.32.3 Psuedocode Example

6.33 LceGetFileType

Retrieves the file type.

BOOL	LceGetFileType	(int	iSession,
		char far *	strFileName,
		char far *	strValue)

Parameters

iSession	The session identifier returned by LceGetSessionId.
strFileName	The file name. The file must have been declared using an LceUseFile function.
strValue	The buffer location for the returned value. The size should be at least FILE_TYPE_LENGTH + 1.
	The file types returned:

- P = a physical file
- L = a logical file.

Return Values

TRUE is returned if the session has got the file type. FALSE is returned if an error occurs.

Notes

- This function performs no communications to the host.
- This function can only be executed after the session is opened.

Related Functions

6.71 LceUseFile6.32 LceGetFileDesc

6.33.1 VB Example

dim sFileType as string
' get file type and append it to list label
sFileType = string (FILE_TYPE_LENGTH + 1, chr(0))
iRet = LceGetFileType(iSession, "PSLMST", sFileType)
lbFileList.Caption = lbFileList.Caption + " Type:" + sFileType

6.33.2 C Example

BOOL fOK; int iSession; char strFileType[FILE_TYPE_LENGTH + 1];

fOK = LceGetFileType(iSession, "PSLMST", strFileType);

6.33.3 Psuedocode Example

6.34 LceGetIBMiSignon

Returns the status of the user profile on the IBM i host and the date the password expires.

BOOL	LceGetIBMiSignon	(char far *	strServer,
		int	iMapperPort,
		BOOL	fSSLRequired
		char far *	strUserid,
		char far *	strPassword,
		BOOL	fEncryptPassword,
		char far *	strReturnCode
		char far *	strExpiryDate)

Parameters

strServer	The IBM i server name or IP address. The length should not exceed SYSTEM_LENGTH as defined in LCOE.h.
iMapperPort	Defaults to 449 if passed as 0.
fSSLRequired	This parameter provides the ability to use SSL to communicate with the IBM i server. True = Use SSL False = Do not use SSL.
strUserid	The User Identifier. The length should not exceed USERID_LENGTH as defined in the LCOE.h header file.
strPassword	The password. The length should not exceed PASSWORD_LONG_LENGTH as defined in the LCOE.H header file
fEncryptPassword	This parameter provides the ability to encrypt the password in communications with the IBM i server. True = Encrypt the password.

False = Do not encrypt the password
The buffer location for the return code. The size should be at least RETURN_CODE_LENGTH + 1.
The return codes returned:
OK - Signon is OK
CE - Communications error
LE - Local Encryption error
NR - User name does not exist
SE - Server error
WP - Wrong Password
UD - User name is disabled
LP – Password is too long
The buffer location for the returned expiry date. The size should be at least EXPIRY_DATE_LENGTH + 1. The format of the date is YYYYMMDD. Only valid if strReturnCode is OK. Returned as "99991231" if the password never expires

Return Values

TRUE indicates that request completed without error.

FALSE indicates an error occurred.

Notes

Because of the way that the IBM i operating system handles user names and short passwords (Password level 0 or 1) with the US English (CCSID 037) characters '@', '#' and '\$', this facility will only work with such user names and short passwords if the IBM i is operating in US English (CCSID 037).

The current implementation of SSL used for this facility ensures that encryption is negotiated and used for communication between the client and the IBM i server. It does not verify that the IBM i server is that specified on the security certificate that has been downloaded.

The interplay between SSL Required and Encrypt Password is interesting. If SSL is available and SSL Required is TRUE, then strictly speaking password encryption is not needed because the entire communication stream is encrypted,

so Encrypt Password could be specified as FALSE. If SSL Required is FALSE, then we recommend that Encrypt Password be specified as TRUE. The reasons for Return Code CE -Communications error can include:

- a misspelling in the IBM i Server name;
- the IBM i Server name not being locatable by your DNS;
- a firewall between the local computer and the IBM i server;
- the IBM i server being offline;
- TCP/IP not being started on the IBM i server;
- TCP/IP host servers not being started on the IBM i server;
- SSL Required TRUE and the SSL TCP/IP host servers not being started on the IBM i server;
- SSL Required FALSE and the non-SSL TCP/IP host servers not being started on the IBM i server.

If Return Code 'SE - Server Error' is returned, a review of the joblog for the QZSOSIGN job on the IBM i server should show the reason in detail.

Related Functions

6.12 LceDisplayErrors6.61 LceSetIBMiSignon

6.34.1 VB Example

Declare Function LceGetIBMiSignon Lib "LCOEW32" (_ ByVal sServer As String, _ ByVal iMapperPort As Integer, _ ByVal bSslRequired As Byte, _ ByVal sUserId As String, _ ByVal sPassword As String, _ ByVal bEncryptPassword As Byte, _ ByVal sReturnCode As String, _ ByVal sExpiryDate As String _) As Integer

Sub LopenTest1()

Dim sServer As String Dim iMapperPort As Integer Dim bSslRequired As Byte Dim sUserId As String Dim sPassword As String Dim bEncryptPassword As Byte Dim sReturnCode As String Dim sExpiryDate As String

```
sServer = "myserver"

iMapperPort = 0

bSslRequired = False

sUserId = "myuser"

sPassword = "mypasswd"

bEncryptPassword = True
```

sReturnCode = String(3, Chr(0))
sExpiryDate = String(20, Chr(0))

Call LceGetIBMiSignon(sServer, iMapperPort, bSslRequired, sUserId, sPassword, bEncryptPassword, sReturnCode, sExpiryDate)

Sheet1.Columns("B").ColumnWidth = 20

```
Sheet1.Cells(2, "B") = "Return Code"
Sheet1.Cells(2, "C") = sReturnCode
Sheet1.Cells(3, "B") = "Expiry Date"
Sheet1.Cells(3, "C") = sExpiryDate
```

End Sub

6.34.2 C Example

```
char strServer[ SYSTEM_LENGTH + 1 ] = "myserver";
int iMapperPort = 0;
BOOL fSSLRequired = false;
char strUserid[ USERID_LENGTH + 1 ] = "myuser";
char strPassword[ PASSWORD_LONG_LENGTH + 1 ] = "mypasswd";
BOOL fEncryptPassword = true;
char strReturnCode[ RETURN_CODE_LENGTH + 1 ];
char strExpiryDate[ EXPIRY_DATE_LENGTH + 1 ];
BOOL rc;
```

rc = LceGetIBMiSignon(strServer, iMapperPort, fSSLRequired, strUserid, strPassword, fEncryptPassword, strReturnCode, strExpiryDate);

```
if (rc)
ł
 if ( strcmp( strReturnCode, "OK" ) == 0 )
  {
   printf( "Signon details successfully retrieved. Expiry Date: %s\n",
strExpiryDate );
  }
  else
  ł
    printf( "The request to the server has failed with return code: %s\n",
strReturnCode );
  }
}
else
{
 printf( "The function failed to run successfully\n" );
}
```

6.34.3 C# Example

```
namespace LansaOpenNative
{
 class LansaOpen
  {
   [DllImport( "lcoew32.dll", CharSet = CharSet.Ansi )]
   public static extern int LceGetIBMiSignon(
     StringBuilder
                                    server,
                               mapperPort,
     int
     int
                               sslRequired,
     StringBuilder
                                    userId,
     StringBuilder
                                    password,
                               encryptPassword,
     int
     StringBuilder
                                    returnCode,
     StringBuilder
                                    expiryDate );
 }
}
private void LOpenSample()
 String strServer = "myuser";
 int iMapperPort = 0;
 bool fSSLRequired = false;
 String strUserid = "myuser";
 String strPassword = "mypasswd";
 bool fEncryptPassword = true;
 StringBuilder strReturnCode = new StringBuilder( 3 );
 StringBuilder strExpiryDate = new StringBuilder( 20 );
 bool rc;
 rc = LansaOpen.LceGetIBMiSignon(
   new StringBuilder( strServer ),
   iMapperPort,
   (fSSLRequired?1:0),
   new StringBuilder( strUserid ),
```

```
new StringBuilder( strPassword ),
```

```
(fEncryptPassword?1:0),
   strReturnCode,
   strExpiryDate );
 if (rc)
 {
   if ( strReturnCode.ToString() == "OK" )
   {
     txtStatusField.Text = String.Format( "Signon details successfully retrieved.
Expiry Date: {0}", strExpiryDate );
   }
   else
   {
     txtStatusField.Text = String.Format( "The request to the server has failed
with return code: {0}", strReturnCode );
   }
  }
 else
  {
   txtStatusField.Text = "The function failed to run successfully";
 }
}
```

6.35 LceGetMessage

This function retrieves an error message sent by the last function executed on the Host System.

The LceGetMessage function is useful if you are planning to program your own error handling. Host error messages generally result from database operations and operations such as LceExecute400, LceSubmit or LceLansaCall.

BOOL	LceGetMessage	(int	iSession,
		int	iMessageNum,
		char far *	strMessageBuf,
		int	iMessageBufLen)

Parameters

iSession	The session identifier returned by LceGetSessionId.
iMessageNum	The sequence number of the error message.
strMessageBuf	The buffer to return the message in.
iMessageBufLen	The size of the buffer.

Return Values

TRUE indicates that error messages were successfully retrieved. FALSE indicates an error occurred.

Notes

Portability Considerations The diagnostic and error messages returned by OAMs running on a Windows, Linux or other non-IBM i host system may be different to those returned by LANSA I/O Modules on the IBM i, as many of them are issued by the native DBMS. For example, DB2/400 on the IBM i, ODBC with SQL Server or SQL Anywhere on Windows.

The number of errors returned from the Host System can be determined by calling the LceGetMessageCount function.

• This function performs no communications to the host.

Related Functions

6.12 LceDisplayErrors6.36 LceGetMessageCount

6.35.1 VB Example

Sub DispLceError(iSessionId As Integer, sSource As String)

' this function displays LANSA Open errors.

' Note that it is only applicable if LceDisplayErrors is off

' otherwise it will never get called, as LANSA takes care of its

' own error messages

Dim i As Integer Dim sErrDesc As String, iErrNo As Long Dim sMsg As String, nMsgs As Integer

Dim msgBuff As String ' message buffer to contain final message

```
msgBuff = ""
  ' Get error status
  sErrDesc = String((MESSAGE_LENGTH + 1), Chr$(0))
  Call LceGetStatus(iErrNo, sErrDesc, MESSAGE LENGTH)
  ' if there was an error
  If iErrNo > 0 Then
    ' prepare header message with error description
    msgBuff = sGetText("Error : ") & str(iErrNo) & " " & sSource
    msgBuff = msgBuff & Chr(13) & sErrDesc & Chr(13) & Chr(13)
    ' read remaining messages
    iRet = LceGetMessageCount(iSessionId, nMsgs)
    For i = 1 To nMsgs '
      sMsg = String(MESSAGE LENGTH * 3, Chr(0))
      iRet = LceGetMessage(iSessionId, i, sMsg, MESSAGE LENGTH)
      sMsg = sTrim(sMsg) ' sTrim detects null terminated strings
      msgBuff = msgBuff + sMsg ' add message to buffer
    Next
  Else
    ' no error just display message
    msgBuff = sGetText("Error") & " : " & sSource
  End If
  msgBox msgBuff ' display errors
End Sub
```

6.35.2 C Example

6.36 LceGetMessageCount

This function returns the number of messages which have been returned by the Host System.

BOOL	LceGetMessageCount	(int	iSession,
		int far *	piMessageCount)

Parameters

iSession The session identifier as returned by LceGetSessionId.

piMessageCount Pointer to the variable to hold the number of messages.

Return Values

TRUE indicates a successful return. FALSE indicates an error occurred.

Notes

This function is useful if you are planning to program your own error handling. Host error messages generally result from database operations and operations such as LceExecute400, LceSubmit, or LceLansaCall.

The value returned by this function can be used as a loop parameter when retrieving messages using the LceGetMessage function. For example, if an LceUpdate failed, the host system may have sent a number of validation errors. This function can be used to determine the number.

• This function performs no communications to the host.

Tip

If you use the LceDisplayErrors function, you will not need to use this function as all the error handling will be performed by LANSA Open.

Related Functions

6.12 LceDisplayErrors6.34 LceGetIBMiSignon

6.36.1 VB Example

Refer to the LceGetMessage 6.36.1 VB Example .

6.36.2 C Example

6.37 LceGetRecordCount

Returns the number of records which were retrieved as a result of the LceReceiveSelect function.

BOOL	LceGetRecordCount	(int	iSession,
		char far *	strFileName,
		int far *	piValue)

Parameters

iSession	The session identifier returned by LceGetSessionId.
strFileName	The file name. This file name must be the same as the file name
	used in the preceding LceReceiveSelect function.

piValue The buffer location which will receive the returned value.

Return Values

TRUE is returned if the session has retrieved the record count of the select. FALSE is returned if an error occurs.

Notes

The number retrieved can be used as a loop counter when processing the received records. The LceGetSelect function requires the number of the record it is to get. You can create a loop containing the LceGetSelect function, where the end condition is the value returned by this function.

This LceGetRecordCount function is used with a number of other select processing functions. The general pattern of these functions is as follows:

- LceSetSelectOptions sets the selection search characteristics. The *RECEIVEIMMED option cannot be used with this function when it is used with this group of functions.
- LceRequestSelect selects records on the host.
- LceReceiveSelect transfers the records from the host to the PC.
- LceGetRecordCount determines how many records were transferred.
- LceGetSelect retrieves a single record from the group of records transferred.
- This function can only be executed after the LceRequestSelect and

LceReceiveSelect functions have been issued.

• This function performs no communications to the host.

Related Functions

6.64 LceSetSelectOptions

- 6.50 LceRequestSelect
- 6.51 LceRequestSelectWhere
- 6.49 LceReceiveSelect
- 6.48 LceReceiveNextX

6.53 LceSelect

6.11 LceDeleteSelect

6.37.1 VB Example

- iRet = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", _ "EMPNO", FALSE);
- iRet = LceReceiveSelect(iSession, "PSLMST");
- iRet = LceGetRecordCount(iSession, "PSLMST", &iRecords);
6.37.2 C Example

```
BOOL fOK;
int iSession;
int iRecords;
fOK = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25");
fOK = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", " ", FA
fOK = LceReceiveSelect(iSession, "PSLMST");
fOK = LceGetRecordCount(iSession, "PSLMST", &iRecords);
for (iCount=0; iCount < iRecords; i++)
{
    fOK = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME, SALARY
}
```

6.38 LceGetSelect

Retrieves the specified fields with the values in the given record from the previous LceReceiveSelect.

BOOL	LceGetSelect	(int	iSession,
		char far *	strFileName,
		int	iRecord,
		char far *	strFieldList)

Parameters

- iSession The session identifier as returned by LceGetSessionId.
- strFileName The name of the file. This name must be the same as the file name used in the preceding LceReceiveSelect function.
- iRecord The required, or next, record number. This number indicates which record is to be retrieved from the file. This value must not be greater than the value returned by the LceGetRecordCount function.
- strFieldList A list of fields to be retrieved from the file. This list must be based on the initial list defined in the LceRequestSelect or LceSelect function. The list may be a subset. You do not have to retrieve all the fields which were requested but you cannot request a field which was not defined in the LceRequestSelect.

Return Values

TRUE is returned if the session selected the records.

FALSE is returned if an error occurs.

Notes

Use the LceGetFieldValueX/LceGetFieldValueU function to obtain the value of a specific field.

The LceGetSelect function is used with a number of other select processing functions. The general pattern of these functions is as follows:

• LceSetSelectOptions sets the selection search characteristics. The

*RECEIVEIMMED option cannot be used with this group of functions.

- LceRequestSelect selects records on the host.
- LceReceiveSelect transfers the records from the host to the PC.
- LceGetRecordCount determines how many records were transferred.
- LceGetSelect retrieves a single record from the group of records transferred.
- This function can only be executed after a session has been opened.

Related Functions

- 6.17 LceGetDefaultValueX
- 6.18 LceGetDefaultValueU
- 6.37 LceGetRecordCount
- 6.64 LceSetSelectOptions
- 6.50 LceRequestSelect
- 6.49 LceReceiveSelect
- 6.48 LceReceiveNextX
- 6.51 LceRequestSelectWhere
- 6.53 LceSelect
- 6.11 LceDeleteSelect

6.38.1 VB Example

dim iRet As Integer, iRecords As Integer, iCount As Integer, _ sBuff As String, lFlags As Long

iRet = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25")

iRet = LceReceiveSelect(iSession, "PSLMST")

```
iRet = LceGetRecordCount(iSession, "PSLMST", iRecords)
```

sBuff = space(256, chr(0))

For iCount=1 to iRecords

```
iRet = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME")
iRet = LceGetFieldValueX(iSession, "SURNAME", sBuff, lFlags)
lstNames.AddItem sTrim (sBuff)
```

lstNames.ItemData (lstNames.NewIndex) = iCount

Next

6.38.2 C Example

```
BOOL fOK;

int iSession;

int iRecords;

char strSurname[FIELD_DATA_SIZE + 1];

long lFlags;

fOK = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25");

fOK = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", "", FA

fOK = LceReceiveSelect(iSession, "PSLMST");

fOK = LceGetRecordCount(iSession, "PSLMST", &iRecords);

for (iCount=1; iCount <= iRecords; iCount++)

{

fOK = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME");

fOK = LceGetFieldValueX(iSession, "SURNAME", (char far *) strSurname

}
```

6.39 LceGetSessionId

Obtains a valid session identifier. This is usually the first function issued by your application.



Parameters

None

Return Values

The returned value is a positive session identifier.

0 is returned if a new session cannot be allocated.

Notes

The session ID is used as a parameter in the following functions to indicate which session is being referenced.

More than one session ID can be retrieved and then opened using LceOpenSession.

• All functions that use the session id must be executed after this function.

Related Functions

6.46 LceOpenSession 6.13 LceEndSession

6.39.1 VB Example

Global iSession as Integer 'declare as global as all functions 'will use it as a parameter

... 'C-of

'Get Session ID iSession = LceGetSessionId() If iSession = 0 Then MsgBox "No Session ID" End ' nothing you can do End If

6.39.2 C Examples

C Example 1

```
int iSession;
iSession = LceGetSessionId();
if (iSession = 0)
{
    /* return error */
}
```

C Example 2

Get two session IDs: int iSession1; int iSession2;

```
iSession1 = LceGetSessionId();
iSession2 = LceGetSessionId();
```

6.40 LceGetStatus

Returns the error status and any messages from LANSA Open functions which result in an error return code.

VOID	LceGetStatus	(long far *	plErrorCode,
		char far *	strErrorMessage,
		int	iMsgLen)

Parameters

plErrorCode	Pointer to the variable to receive the error status.
	0 = a successful return.
	Non-zero = an error occurred.
strErrorMessage	Array to receive error messages, if any. This array will be empty if no error message text is found.
iMsgLen	Maximum number of characters to copy to strErrorMessage.
_	

Return Values

None

Notes

Portability	The Validation error (36) and I/O error (37) returned by
Considerations	OAMs on Windows, Linux or other non-IBM i platforms,
	may be different to those returned by LANSA I/O
	Modules on an IBM i, as many of them translate from the
	native DBMS. For example, DB2/400 on the IBM i,
	ODBC with SQL Server or SQL Anywhere on Windows.

LANSA Open maintains a local error message area. When a LANSA Open function returns an error, the error message associated with the error is saved, provided the error message area is empty. If it is not empty, the message is ignored.

LceGetStatus retrieves the error message(s) and at the same time clears the error message area. LceGetStatus, therefore, needs to be executed after every LceGetFunction function.

• This function performs no communications with the host.

Tip

An alternative to LceGetStatus is LceDisplayErrors, LANSA Open's automated error message handing facility.

Related Functions

6.12 LceDisplayErrors

6.40.1 VB Example

Refer to the LceGetMessage 6.35.1 VB Example .

6.40.2 C Example

long lErrorCode; char strErrorMessage[100];

LceGetStatus(&lErrorCode, strErrorMessage, sizeof(strErrorMessage)); if (lErrorCode != 0) { process error} /* Issue a block of related LANSA Open calls */ /* Process first error which may have occurred */ LceGetStatus(&lErrorCode, strErrorMessage, sizeof(strErrorMessage)); if (lErrorCode != 0) { process error }

6.41 LceGetSystemAttribute

Retrieve information about the host system.

BOOL	LceGetVariable	(int	iSession,
		int	iAttrbiute,
		int far *	iValue)

Parameters

- iSession The session identifier as returned by LceGetSessionId.
- iAttribute The attribute to retrieve. Available attributes are listed in the Notes.
- iValue The value of the option. Refer to the Notes for the available values.

Return Values

TRUE is returned if the session can retrieve the requested value.

FALSE is returned if an error occurs.

Notes

This function can only be executed after a session has been opened.

The following system attributes may be retrieved:

LCE_SYS_CHARSET_TYPE

Specifies the Host Type. Available values are LCE_HOST_EBCDIC and LCE_HOST_ASCII as defined in the LCOE.H header file.

LCE_SYS_OS_TYPE

Specifies the Operating System type. Available values are LCE_HOST_ISERIES, LCE_HOST_WINDOWS, LCE_HOST_UNIX as defined in the LCOE.H header file.

6.41.1 VB Example

Dim iValue As Integer iRet = LceGetSystemAttribute (iSession LCE_SYS_CHARSET_TYPE, iValue)

6.41.2 C Example

int iSession; int iValue; fOK = LceGetVariable(iSession, LCE_SYS_OS_TYPE, &iValue);

6.42 LceGetVariable

Retrieve the value associated with a given system or multilingual variable.

BOOL	LceGetVariable	(int	iSession,
		char far *	strVariableName,
		char far *	strValue)

Parameters

iSession The session identifier as returned by LceGetSessionId.

strVariableName The system or multilingual variable name.

strValue Pointer to the array to receive the returned value. This will be a zero length string if no value is returned from the host. The size should be at least FIELD_DATA_SIZE + 1 as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can retrieve the requested value.

FALSE is returned if an error occurs.

Notes

If the request is for a multilingual variable, the partition must be multilingual. The value returned will be based on the language defined when the session is open.

This function uses string parameters. Even if the variable you are getting is nonnumeric, it will contain a string value representing the variable's numeric value. You will need to convert the string value to a numeric before using it in your application. Note also that any "decimal point" used (if appropriate) will be that which applies on the host.

- This function may be executed at any time after the session is opened.
- LceGetVariable communicates with the host for all initial calls for the given variable. Subsequent calls will result in communication with the host only if the variable is defined as dynamic.

6.42.1 VB Example

Dim sBuff As String sBuff = String(FIELD_DATA_SIZE, Chr(0)) ' set buffer iRet = LceGetVariable (iSession "*COMPANY", Company) Form.Caption = Company

6.42.2 C Example

BOOL fOK; int iSession; char strValue[FIELD_DATA_SIZE + 1]; fOK = LceGetVariable(iSession, "*DATE", strValue);

6.43 LceInsert

Inserts a record into the specified file, using the values contained in the fields.

BOOL	LceInsert	(int	iSession,
		char far *	strFieldList,
		char far *	strFileName
		BOOL	fValidate)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strFieldList	A list of fields to be inserted when the record is added. Each field must have been declared using an LceUseField function.
strFileName	The name of the file. This file must have been previously declared using an LceUseFile function.
fValidate	This parameter provides a "check but do not insert" facility. True = Perform validate only.
	False = Perform insert immediately.

Return Values

TRUE is returned if the session can insert or has inserted the record. FALSE is returned if an error occurs.

Notes

Only the fields specified in the field list will be inserted into the file. Other fields will have their default values assigned, provided the default values are set in the LANSA Repository.

• This function may only be executed after a session has been opened.

Tip

If you record the appropriate validation rules in the LANSA Repository, LANSA will perform referential integrity checks when inserting new records. For example, LANSA can check that a customer record exists in the customer file before attempting to insert a new order record for that customer in the order file.

Related Functions

6.71 LceUseFile
6.58 LceSetFieldValue
6.59 LceSetFieldValueU
6.67 LceUnicodeToHex
6.10 LceDelete
6.8 LceCheckFor

6.43.1 VB Example

Private Sub cmdSave_Click()

' Saves changes or inserts a new record. Note that the fields are

' updated as the user changes them.

' Errors are automatically handled by LANSA

Dim iRet As Integer

If bAdding = True Then 'user is inserting a new record iRet = LceInsert(iSession, "SKILCODE,SKILDESC", "SKLTAB", _ LceFalse) Else 'user is saving changed record End If If iRet = LceTrue Then 'if successful force a refresh so list shows changes Call RefreshSkillsList End If End Sub

6.43.2 C Example

Insert a record into the file. BOOL fOK; int iSession;

fOK = LceSetFieldValue(iSession, "EMPNO", "12345"); fOK = LceSetFieldValue(iSession, "SURNAME", "Buckley");

fOK = LceInsert(iSession, "EMPNO, SURNAME", "PSLMST", FALSE);

Psuedocode Example

```
Checks if a record can be inserted.

strName = "PSLMST"

strList = "EMPNO, SURNAME"

IF NOT LceSetFieldValue(iSession, "EMPNO", "12345")

/* return error */

ENDIF

IF NOT LceSetFieldValue(iSession, "SURNAME", "Buckley")

/* return error */

ENDIF

IF LceInsert(iSession, strList, strName, TRUE )

/* insert the new record can be performed */

ELSE

/* cannot insert the new record */

/* display error messages */

ENDIF
```

6.44 LceLansaCall

This function is used to initiate the immediate execution of a LANSA process/function or an IBM i program. If the program you call expects user interaction, this function will fail.

When using this function, your application must wait until the called program or function has been completed. Error messages will be routed back to your application.

BOOL	LceLansaCall	(int	iSession,
		char far *	strProcess,
		char far *	strFunction,
		char far *	strProgram,
		char far *	strExchangeList)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strProcess	The name of the LANSA process.
strFunction	The name of the LANSA function.
	or
strProgram	The name of the IBM i program.
strExchangeList	A list of field names to be passed to and from the program. Refer to Notes.

Return Values

TRUE is returned if the session has executed the process/function or program. FALSE is returned if an error occurs.

Notes

Portability	The format	
Considerations	your host sy	
	by the host	

The format which uses strProgram is only supported if your host system is an IBM i. An error will be returned by the host if you make a call with strProgram to any other platform.

When calling a program on the IBM i, the exchange list may only contain fields of type A, P, and S. If exchanging numeric fields, the maximum length is 30 and the maximum decimals is 9.

Parameters can be passed to the called function or program via the exchange list of field names in strExchangeList. Each field used in this exchange list must have been declared using an LceUseField function. The value for the field is assigned using the LceSetFieldValue/LceSetFieldValueU function. The LANSA process and function to be executed do not have to be declared previously by an Lce function call.

Parameters can be returned from the called function or program via the exchange fields. The returned value for the field is stored and may be accessed using the LceGetFieldValueX/LceGetFieldValueU function.

• This function can only be executed after a session has been opened.

Tip

This function is ideal to

- re-use an existing host process or IBM i program.
- perform a task that requires access to multiple files.

If you don't want to wait until the host process completes, use the LceSubmit function instead.

If you wish to call a 3GL program that either passes or receives parameters, use Lce3GLCall instead.

Related Functions

6.66 LceSubmit6.14 LceExecute4006.3 Lce3GLCall

6.44.1 VB Example

Private Sub cmdPrintDepartment_Click()

' ask the host to print a list of the department's

' employees

' set department to print

iRet = LceSetFieldValue(iSession, "DEPTMENT", DeptKey(cmbDepartment.I If iRet = LceTrue Then

iRet = LceLansaCall(iSession, "PSLSYS", "EMPLIST", "", "DEPTMENT") End If

End Sub

6.44.2 C Example

BOOL fOK; int iSession; long lFlags; fOK = LceSetFieldValue(iSession, "MONTH", "JAN"); fOK = LceSetFieldValue(iSession, "YEAR", "1994");

fOK = LceLansaCall(iSession, "PROCREP", "SALES01", "", "MONTH, YEAR'
fOK = LceGetFieldValueX(iSession, "SURNAME", strLastName, &lFlags);

6.45 LceLocalDateTimeToServer

Converts the DateTime value from the client's local timezone into UTC (Universal Coordinated Time). This function must be used when creating a where clause for LceRequestSelectWhere, if DateTime values are required in the where clause.

BOOL	LceLocalDateTimeToServer	(int	iSession,
		char far *	strFieldName,
		char far *	strDate)

Parameters

- iSession The session identifier as returned by LceGetSessionId.
- strFieldName The field name of a DateTime field previously declared with LceUseField.
- strDateTime Pointer to the array that specifies the DateTime to be converted. The converted DateTime will be returned in this parameter.

Return Values

TRUE is returned if the session has successfully converted the DateTime. FALSE is returned if an error occurs.

Notes

This function should only be used when specifying a value for a DateTime field in a where clause. It must not be used to convert a DateTime value which is then passed to LceSetFieldValue as this API will automatically do the conversion.

• This function may be executed at any time after the session is opened.

Related Functions

6.29 LceGetFieldValueX6.33 LceGetFileType6.51 LceRequestSelectWhere

6.45.1 VB Example

```
Public Type FIELD_DETAIL
  fieldLen As Long
  flags As Long
End Type
Sub SetEmployeesList()
  Dim i As Integer
  Dim fldData(2) As FIELD DETAIL
  ' clear list
  lstOrders.Clear
  If cmbSection.ListIndex = -1 Then
Exit Sub ' no department do nothing
  End If
  ' Select data
  iRet = LceSetSelectOptions(iSession, "*RECEIVEIMMED")
  ' Request data
  If iRet = LceFalse then exit sub ' Failed
  iRet = LceLocalDateTimeToServer(iSession, "ORDRTIME", sOrderStartDate)
  iRet= LceRequestSelectWhere(iSession,
     "ORDERNUM, ORDRTIME", "ORDERS", "", "ORDRTIME > " + sOrder
  If iRet = LceTrue Then
    sBuff = String(40, Chr(0))
    i = 0
    ' Read all records
    While (LceReceiveNextX(iSession, sBuff, FIELD DATA SIZE * 2, fldData
      lstOrders.AddItem (sTrim(Mid(sBuff, 1, fldData(0).fieldLen))
              & " " & sTrim(Mid(sBuff, fldData(0).fieldLen + 1, fldData(1).fiel
      i = i + 1
    Wend
    ' Select first on list
    If lstEmployees.ListCount > 0 Then lstEmployees.ListIndex = 0
```

else ' failed ... End If End Sub

6.45.2 C Example

BOOL fOK; int iSession; int iCount; char szOrderStartDate[DATETIME_LEN]; char szWhereClause[256];

```
fOK = LceSetSelectOptions(iSession, "*BLOCKBYRRN");
fOK = LceLocalDateTimeToServer(iSession, "ORDRTIME", szOrderStartDate)
sprintf( szWhereClause, "ORDRTIME > %s", szOrderStartDate );
fOK = LceRequestSelectWhere(iSession,
        "ORDERNUM, ORDRTIME", "ORDERS", "", szWhereClause, FALSE);
fOK = LceReceiveSelect(iSession, "ORDERS");
fOK = LceGetRecordCount(iSession, "ORDERS", &iRecords);
for (iCount=0; iCount < iRecords; i++)
{
        fOK = LceGetSelect(iSession, "ORDERS", iCount, "ORDERNUM, ORDR]
}
```

6.46 LceOpenSession

Opens a communications session with the host.

BOOL LceOpenSession (int iSession)

Parameters

iSession The session identifier returned by LceGetSessionId.

Return Values

TRUE is returned if the session can be opened.

FALSE is returned if an error occurs.

Notes

LceOpenSession is used in conjunction with a number of LANSA Open "session definition" functions that set the session's options.

The time taken to open a session will depend upon many factors including the client and host system's hardware and software.

• The more fields and files used, the longer the time it will take to open the session. A theoretical maximum of 256 files and 2800 fields may be used in a session.

•

• This function must be executed after an LceGetSessionId function.

Tip

For consecutive file processing, set the fResume parameter to TRUE when calling LceEndSession. This keeps the communications session with the host system active. You can then use the same session identifier (for processing the other files) in LceOpenSession.

You can also have more than one session open (via LceOpenSession) if you need to access more than one file concurrently when using the *RECEIVEIMMED option for selecting records.

Related Functions

```
6.39 LceGetSessionId6.13 LceEndSession6.69 LceUseExceptionalArguments
```

6.71 LceUseFile
6.72 LceUseFunction
6.73 LceUseLanguage
6.74 LceUsePartition
6.75 LceUsePassword
6.76 LceUsePriority
6.77 LceUseProcess
6.78 LceUseSystem
6.79 LceUseTmpDrive2

6.46.1 VB Example

iRet = LceOpenSession(iSession)

6.46.2 C Example

```
if LceOpenSession(iSession)
{
    /* return okay */
}
else
{
    /* return error as session failed to open */
}
```

6.47 LceReceiveNext

This function has been replaced by 6.48 LceReceiveNextX and is supported for backward compatibility only. It does not support RDMLX fields.

BOOL	LceReceiveNext	(int	iSession,
		char *	strBuffer,
		int	iBufferSize)

6.48 LceReceiveNextX

Retrieves the selected record which is being sent immediately from the host.

BOOL	LceReceiveNextX	(int	iSession,
		char *	strBuffer,
		BUFLEN	bufferSize,
		PFIELD_DETAILS	pDataDetails,
		int	iFields)

Parameters

iSession	The session identifier.
strBuffer	A pointer to a buffer to contain the returned record.
iBufferSize	The size of the buffer passed.
pDataDetails	For each field, it provides the length of the data and any special flags. See the structure definition in the LCOE.H header file. Refer to notes for further details.
iFields	The maximum number of fields to retrieve in the buffer.

Return Values

TRUE is returned if the session can receive the selected record.

FALSE is returned if no more records are found or an error occurs. To distinguish between an error and no record found, use the LceGetStatus function to check that no error exists.

Notes

The LceReceiveNextX function is used with a number of select processing functions but only after a call to LceRequestSelect or LceRequestSelectWhere and if you are using the *RECEIVEIMMED option.

The general pattern of these functions is as follows:

- LceSetSelectOptions with option *RECEIVEIMMED specified.
- LceRequestSelect or LceRequestSelectWhere selects records on the host. (The LceRequestSelectWhere function allows you to attach a Where clause
to your selection criteria.)

• LceReceiveNextX reads the records transferred from the host to the PC. (Unless you have another session open, once a *RECEIVEIMMED select has been initiated, then all the records must be selected via the LceReceiveNextX function before any other function which results in communication with the Host (e.g. LceFetch) can be performed.

If at any time the remaining records for a Select call are not needed, they can be discarded by executing an LceDeleteSelect function.

If no buffer is specified, when a call to LceReceiveNextX is performed, all the values in the field information will be updated with the values for the current record. These values can be obtained in the same way you would retrieve the field values after a call to LceGetSelect, i.e. using the LceGetFieldValueX function.

If a buffer is specified, the buffer will be updated with the field values in the order specified on the call to LceRequestSelect. This buffer is for use in your client program and it cannot be accessed by other LANSA Open functions, i.e. LceGetFieldValueX/LceGetFieldValueU. The buffer provides a more efficient method of accessing the data. Using the buffer can improve performance greatly.

The size of each value returned is stored in fieldLen in the pDataDetails parameter. You can use fieldLen to move from field to field in the buffer. This size includes the 0x00 terminator between the values.

ulFlags in the pDataDetails parameter identifies the returned value as *SQLNULL. If the field has a value of *SQLNULL, pulFlags will return 1 otherwise it will return 0.

To ensure the buffer is large enough to store the data for all requested fields, use the maximum field length when determining the amount of memory to allocate for the buffer. This can be obtained using LceGetFieldType. However, the maximum may not always be required. Since RDMLX fields are not padded, a smaller buffer may be sufficient. An error will be displayed if the buffer is not large enough to store all the requested fields.

For backward compatibility the size of each value returned for an RDML field will continue to be defined by the following rules:

ALPHA FIELD LENGTH+1 SIGNED FIELD LENGTH+4

PACKED FIELD LENGTH+4

This size includes the 0x00 terminator between the values. If the value is actually shorter than the length specified then it will be padded with 0x00. This padding occurs after processing the current setting for alpha or numeric pad/strip.

• This function may only be executed once a session has been opened.

Related Functions

6.64 LceSetSelectOptions

- 6.50 LceRequestSelect
- 6.51 LceRequestSelectWhere
- 6.29 LceGetFieldValueX
- 6.18 LceGetDefaultValueU
- 6.11 LceDeleteSelect
- 6.27 LceGetFieldType

6.48.1 VB Example

Public Type FIELD_DETAIL fieldLen As Long flags As Long End Type

Public Const TOTAL_FIELDS = 2

Sub SetSkillsList() ' reads all skill descriptions and populates the skill list On Error Resume Next

Dim iRet As Integer Dim i As Integer Dim sBuff As String Dim sTemp As String Dim iPos As Long Dim iLovalPos As Long Dim fldData(TOTAL_FIELDS) As FIELD_DETAIL

' remove any previous query results
iRet = LceDeleteSelect(iSession, "SKLTAB")

' Request select

' Set Select to IMMEDIATE MODE is recommended to increase speed

iRet = LceSetSelectOptions(iSession, "*RECEIVEIMMED")

' Clear previous selection sBuff = String(FIELD_DATA_SIZE, Chr(0)) lstSkills.Clear ReDim SkillKey(0) i = 0 ' reset counter

'Read all records selected While LceReceiveNextX(iSession, sBuff, FIELD_DATA_SIZE, fldData(0), T(ReDim Preserve SkillKey(i + 1) ' resize array key

```
iPos = 1
    If (fldData(i).flags = 1) Then
       ' Display error as key field should not be Null
    Else
       sTemp = Trim(Mid(sBuff, iPos, fldData(0).fieldLen))
       iLovalPos = InStr(1, sTemp, Chr(0), vbTextCompare)
       If iLovalPos > 0 Then
         sTemp = Trim(Left(sTemp, iLovalPos - 1))
       End If
       SkillKey(i) = sTemp
       iPos = iPos + fldData(0).fieldLen
       If (fldData(i).flags = 1) Then
         ' Value is SQLNULL
         sTemp = "No description provided"
       Else
         sTemp = Trim(Mid(sBuff, iPos, fldData(1).fieldLen))
         iLovalPos = InStr(1, sTemp, Chr(0), vbTextCompare)
         If iLovalPos > 0 Then
            sTemp = Trim(Left(sTemp, iLovalPos - 1))
         End If
       End If
       lstSkills.AddItem sTemp
    End If
    i = i + 1 ' increment index
  Wend
  ' select first record
  If lstSkills.ListCount > 0 Then lstSkills.ListIndex = 0
End Sub
```

6.48.2 C Examples

C Example 1

BOOL fOK; BOOL fNext; int iSession; int iCount; long lFlags;

```
fOK = LceSetSelectOptions(iSession, "*RECEIVEIMMED");
fOK = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", "", FA
fNext = TRUE;
iCount = 0;
while (fNext)
{
    /* process records as they are received */
    fNext = LceReceiveNextX(iSession, "", 0, NULL, 0);
    if (fNext)
    {
      fOK = LceGetFieldValueX(iSession, "SURNAME", strSurname, &lFlags
      fOK = LceGetFieldValueX(iSession, "SALARY", strSalary, &lFlags);
    }
    /* count number of records received */
    iCount = iCount + 1;
}
```

C Example 2

#define TOTAL_FIELDS 2

BOOLfOK;intiSession;intiCount;longlFlags;FIELD_DETAILS fldDetails[TOTAL_FIELDS];charszBuff[256];

```
fOK = LceSetSelectOptions(iSession, "*RECEIVEIMMED");
fOK = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", "", FA
iCount = 0;
while (LceReceiveNextX(iSession, sBuff, 256, &fldDetails, TOTAL_FIELDS))
{
 if (fldDetails[0].fldLength = 1)
 {
   /* Display error as a Surname must exist */
  }
 else
  {
   strcpy( szSurname, sBuff );
   strcpy( szSalary, sBuff + fldDetails[0].fldLength );
  }
 /* count number of records received */
 iCount = iCount + 1;
}
```

6.49 LceReceiveSelect

Performs the data transfer, from the host to the PC, from a file of one or more records which have been identified by an LceRequestSelect or an LceRequestSelectWhere function.

BOOL	LceReceiveSelect	(int	iSession,
		char far *	strFileName)

Parameters

iSession The session identifier as returned by LceGetSessionId.

strFileName The name of the file. This name must correspond to the file name used in the preceding LceRequestSelect or LceRequestSelectWhere function.

Return Values

TRUE is returned if the session has received the selected records.

FALSE is returned if an error occurs.

Notes

This function is used with a number of other select processing functions. The general pattern of these functions is as follows:

- LceSetSelectOptions sets the selection search characteristics. The *RECEIVEIMMED option cannot be used with this function.
- LceRequestSelect or LceRequestSelectWhere selects records on the host.
- LceReceiveSelect transfers the records from the host to the PC.
- LceGetRecordCount determines how many records were transferred.
- LceGetSelect retrieves a single record from the group of records transferred.
- This function can only be executed after a session has been opened and following an LceRequestSelect function.

Related Functions

6.64 LceSetSelectOptions

- 6.50 LceRequestSelect
- 6.51 LceRequestSelectWhere

6.37 LceGetRecordCount6.38 LceGetSelect6.53 LceSelect6.11 LceDeleteSelect

6.49.1 VB Example

Dim iRet As Integer, iRecords As Integer, iCount As Integer, _ sBuff As String, lFlags As Long

iRet = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25")

iRet = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", _
"", FALSE)

iRet = LceReceiveSelect(iSession, "PSLMST")

```
iRet = LceGetRecordCount(iSession, "PSLMST", iRecords)
```

sBuff = space(256, chr(0))

For iCount=1 to iRecords

```
iRet = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME")
iRet = LceGetFieldValueX(iSession, "SURNAME", sBuff, lFlags)
lstNames.AddItem sTrim (sBuff)
```

lstNames.ItemData (lstNames.NewIndex) = iCount

Next

6.49.2 C Example

BOOL fOK; int iSession; int iRecords;

```
fOK = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25");
fOK = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", "", FA
fOK = LceReceiveSelect(iSession, "PSLMST");
fOK = LceGetRecordCount(iSession, "PSLMST", &iRecords);
for (iCount=0; iCount < iRecords; i++)
{
    fOK = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME, SALARY");
}
```

6.50 LceRequestSelect

This function is used to identify one or more records matching the search criteria, from a file on a host.

BOOL	LceRequestSelect	(int	iSession,
		char far *	strFieldList,
		char far *	strFileName,
		char far *	strKeyList,
		BOOL	fGeneric)

Parameters

iSession	The session identifier returned by LceGetSessionId.	
strFieldList	A list of fields to be retrieved from the file. Each field listed must have been declared using an LceUseField function.	
strFileName	The name of the file to be used. This file must have been declared using an LceUseFile function	
strKeyList	A list of the key field names in the sequence in which they are used in the file. These fields must have been declared using the LceUseField function. The values for the keys are set using the LceSetFieldValue/LceSetFieldValueU function. Refer to Notes for key and generic search considerations. Key fields must not be specified if the *BLOCKBYKEY option was used in the preceding LceSetSelectOptions.	
fGeneric	 Refer to notes for further information. True = apply a generic search. 	

• False = do not apply a generic search.

Return Values

TRUE is returned if the session has selected records.

FALSE is returned if an error occurs.

Notes

Key and Generic search considerations:

- You must know the specific key fields to the file. The key fields listed can be a subset of the actual keys to the file, but the sequence in which the keys are listed must match the sequence of the keys in the file. This is very important. The sequence of the keys must match the file. If you need the keys in a different sequence you must use a logical file.
- The fGeneric flag determines whether the keys must match exactly or whether a generic key search should apply. If a generic search is used on a file with more than one key field, then the search will apply only to the last non-blank key field specified. Any blank key field will be considered generic.
- The fGeneric flag is ignored for fields of type Date, Time, DateTime, Integer, and Float.
- You do not have to use the complete key list. You could use the first two fields in a file with three key fields. If you use a subset of the key fields, you are implicitly using a generic search on the unspecified keys, i.e. match key1 and key2 while key3 can have any value. In this case, you do not use the fGeneric search. If you did use the generic search in this situation, you will get a generic search based on key2 and key3.

This LceRequestSelect function is used with a number of other select processing functions. The general pattern of these functions is as follows:

- LceSetSelectOptions sets the selection search characteristics.
- LceRequestSelect selects records on the host.
- LceReceiveSelect transfers the records from the host to the PC.
- LceGetRecordCount determines how many records were transferred.
- LceGetSelect retrieves a single record from the group of records transferred.
- This function may only be executed after a session has been opened.

Tip

For a better performance when retrieving records, use the LceRequestSelect and LceReceiveNextX functions with the select option set to *RECEIVEIMMED.

Related Functions

6.64 LceSetSelectOptions6.50 LceRequestSelect6.51 LceRequestSelectWhere

6.48 LceReceiveNextX6.37 LceGetRecordCount6.38 LceGetSelect6.53 LceSelect6.11 LceDeleteSelect

6.50.1 VB Example

Dim iRet As Integer, iRecords As Integer, iCount As Integer, _ sBuff As String, lFlags As Long

iRet = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25")

iRet = LceReceiveSelect(iSession, "PSLMST")

```
iRet = LceGetRecordCount(iSession, "PSLMST", iRecords)
```

sBuff = space(256, chr(0))

For iCount=1 to iRecords

```
iRet = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME")
iRet = LceGetFieldValueX(iSession, "SURNAME", sBuff, lFlags)
lstNames.AddItem sTrim (sBuff)
```

lstNames.ItemData (lstNames.NewIndex) = iCount

Next

6.50.2 C Example

```
BOOL fOK;
int iSession;
int iRecords;
fOK = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25");
fOK = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", "", FA
fOK = LceReceiveSelect(iSession, "PSLMST");
fOK = LceGetRecordCount(iSession, "PSLMST", &iRecords);
for (iCount=0; iCount < iRecords; i++)
{
    fOK = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME, SALARY
}
```

6.51 LceRequestSelectWhere

Identifies one or more records matching the search criteria in a file on the host. No data is transferred from the host.

BOOL	LceRequestSelectWhere	(int	iSession,
		char *	strFieldList,
		char *	strFileName,
		char *	strKeyList,
		char *	strConditions,
		BOOL	fGeneric)

Parameters

iSession	The session identifier.
----------	-------------------------

strFieldList	A list of field names. All fields must have been declared using an LceUseField function.		
	Field (i.e. column) names may be specified in upper or lower case.		
	Field names do NOT begin with "#" (hash/pound) signs.		
	This list may refer to any fields from the specified file. Fields may be real or virtual.		
strFileName	The name of the file. This file must have been declared using an LceUseFile function		
strKeyList	A list of key field names. These key fields must have been declared using the LceUseField function. The values for the keys are set using the LceSetFieldValue/ LceSetFieldValueU function.		
	Key fields must not be specified if the *BLOCKBYKEY option was used in the preceding LceSetSelectOptions.		
	Refer to the notes before using these key fields.		

strConditions The 'Where' clause used for the select. Refer to the notes for

more information, including how to correctly use DateTime field values.

fGeneric Refer to the notes for more information. True = apply a generic search. False = do not apply a generic search.

Return Values

TRUE is returned if the session can select records.

FALSE is returned if an error occurs.

Notes

The Where clause, specified in strConditions, must conform to the rules defined for Expression format and definition in Specifying Conditions and Expressions in the *LANSA Technical Reference* Guide.

- If the Where expression uses only real fields, the expression is evaluated before any virtual field derivation logic is executed. This means that if the expression is not true, the record/row is skipped before incurring the virtual field derivation overhead.
- If the Where expression includes one or more virtual fields, the expression is not evaluated until all virtual logic has been executed.
- Bracketing of multiple expression components is mandatory. For example: (SALARY *LT 30000) *AND (DEPTMENT *EQ 'ADM')
- Single simple expression components should not be bracketed. For example: SALARY *LT 30000
- The decimal point used must match the decimal point used for the value in the host file.
- Numeric expression calculations and comparisons are carried out at (30,9) precision i.e. 21 integer digits and 9 decimal digits of precision. Calculations or comparisons requiring a precision above this level may produce unexpected results.
- Alphanumeric literal values must be enclosed in quotes and cannot exceed 256 characters.
- The entire Where expression cannot exceed 2048 characters.
- DateTime field values must be converted from local time to UTC before being used in a where clause. Use LceLocalDateTimeToServer to do the

adjustment.

• Unicode field values must be converted to a formatted hex byte stream before being used in a where clause. Use LceUnicodeToHex to convert a Unicode string to a hex byte stream.

Key and Generic search considerations:

- You must know the specific key fields to the file. The key fields listed can be a subset of the actual keys to the file, but the sequence in which the keys are listed must match the sequence of the keys in the file. This is very important. The sequence of the keys must match the file. If you need the keys in a different sequence you must use a logical file.
- The fGeneric flag determines whether the keys must match exactly or whether a generic key search should apply. If a generic search is used on a file with more than one key field, then the search will apply only to the last non-blank key field specified. Any blank key field will be considered generic.
- The fGeneric flag is ignored for fields of type Date, Time, DateTime, Integer, and Float.
- You do not have to use the complete key list. You could use the first two fields in a file with three key fields. If you use a subset of the key fields, you are implicitly using a generic search on the unspecified keys, i.e. match key1 and key2 while key3 can have any value. In this case, you do not use the fGeneric search. If you did use the generic search in this situation, you will get a generic search based on key2 and key3.

The LceRequestSelectWhere function is used with a number of other select processing functions. If the *RECEIVEIMMED option is **not** used with LceSetSelectOptions, the general pattern of these functions is as follows :

- LceSetSelectOptions sets the selection search characteristics.
- LceRequestSelectWhere selects records on the host.
- LceReceiveSelect transfers the records from the host to the PC.
- LceGetRecordCount determines how many records were transferred.
- LceGetSelect retrieves a single record from the group of records transferred.

If the *RECEIVEIMMED option is used, then the functions will be as follows:

- LceSetSelectOptions sets the selection search characteristics.
- LceRequestSelectWhere selects records on the host.
- LceReceiveNextX reads the next record transferred.

• This function can only be executed once a session has been opened.

Tip

Performance Considerations

For better performance, whenever possible use an LceRequestSelect instead of this LceRequestSelectWhere function.

If your host is an IBM i and when LceRequestSelectWhere must be used, consider using the *BLOCKBYRRN and *RECEIVEIMMED select options in the preceding LceSetSelectOptions for best performance and lowest impact on the IBM i server. Review the LceSetSelectOptions for more details.

For example:

```
LceSetFieldValue(iSessionId, "DEPTMENT", "ADM");
LceRequestSelectWhere(iSessionId,
"EMPNO,SURNAME,SALARY"
"PSLMST",
"DEPTMENT",
"SALARY *GT 30000",
FALSE);
```

will normally be more efficient than:

```
LceRequestSelectWhere(iSessionId,

"EMPNO,SURNAME,SALARY"

"PSLMST",

"",

"(SALARY *GT 30000) AND (DEPTMENT = 'ADM')",

FALSE);
```

Constantly consider the viability of what you are doing.

For example, a select operation reading 10,000 rows to select 100, frequently used from multiple PC clients, is not a viable operation on any host system. Consider setting up a key/logical view to better support this type of operation.

The concept of what is viable varies according to

- the power and loading of the host
- the speed and loading of the communications sub-system moving data between the host and the PC.
- the frequency with which the application is used and the importance of the

result it produces.

Remember that the LANSA Open functions are primarily designed to act as high performance "back ends" to programmer designed and implemented applications, not to be generic end user query tools by themselves.

Related Functions

6.64 LceSetSelectOptions
6.45 LceLocalDateTimeToServer
6.67 LceUnicodeToHex
6.49 LceReceiveSelect
6.48 LceReceiveNextX
6.37 LceGetRecordCount
6.38 LceGetSelect
6.53 LceSelect
6.11 LceDeleteSelect

6.51.1 VB Example

```
Public Type FIELD_DETAIL
  fieldLen As Long
  flags As Long
End Type
Sub SetEmployeesList()
  Dim i As Integer
  Dim fldData(3) As FIELD DETAIL
  ' clear list
  lstEmployees.Clear
  If cmbSection.ListIndex = -1 Then
Exit Sub ' no department do nothing
  End If
  ' Select data
  iRet = LceSetSelectOptions(iSession, "*RECEIVEIMMED")
  ' set keys value department/section
  If iRet = LceTrue Then
   iRet = LceSetFieldValue(iSession, "DEPTMENT",
         DeptKey(cmbDepartment.ListIndex))
  If iRet = LceTrue Then
   iRet = LceSetFieldValue(iSession, "SECTION", _
         SectKey(cmbSection.ListIndex))
  ' Request data
  If iRet = LceFalse then exit sub ' Failed
  If txtSurname = "" Then ' no surname search
   iRet = LceRequestSelect(iSession, "EMPNO, SURNAME, GIVENAME",
       "PSLMST1", "DEPTMENT, SECTION", False)
  Else
   iRet= LceRequestSelectWhere(iSession, _
     "EMPNO,SURNAME,GIVENAME", _
     "PSLMST1", "DEPTMENT, SECTION", _
     "DEPTMENT *EQ " &
     DeptKey(cmbDepartment.ListIndex))
```

End If

```
If iRet = LceTrue Then
    sBuff = String(FIELD_DATA_SIZE * 3, Chr(0)) ' plenty of space
    i = 0
    ReDim EmpKey(0)
    ' Read all records
    While (LceReceiveNextX(iSession, sBuff, FIELD_DATA_SIZE * 3, fldData
        = LceTrue)
      ReDim Preserve EmpKey(i + 1)
      ' Only RDML fields retrieved so can assume starting positions are in fixed
      lstEmployees.AddItem (sTrim(Mid(sBuff, 28, 20)) _
              & " " & sTrim(Mid(sBuff, 7, 20))) ' Full Name
      EmpKey(i) = sTrim(Left(sBuff, 5)) ' EmpNo
      i = i + 1
    Wend
    ' Select first on list
    If lstEmployees.ListCount > 0 Then lstEmployees.ListIndex = 0
 else
    ' failed
    ...
  End If
End Sub
```

6.51.2 C Example

BOOL fOK; int iSession; int iRecords;

```
fOK = LceSetSelectOptions(iSession, "*BLOCKBYRRN");
fOK = LceRequestSelectWhere(iSession, "EMPNO,SURNAME,SALARY","PSI
"","(SALARY *GT 50000) AND (SALARY *LT 100000)",FALSE);
fOK = LceReceiveSelect(iSession, "PSLMST");
fOK = LceGetRecordCount(iSession, "PSLMST", &iRecords);
for (iCount=0; iCount < iRecords; i++)
{
    fOK = LceGetSelect(iSession, "PSLMST", iCount, "EMPNO,SURNAME,S
```

```
}
```

6.52 LceRollBackWork

This function is used to rollback the current transaction to the previous commit point for a transaction or set of transactions on the host.

An LceSetCommitmentOn function must have been executed before LceRollBackWork can be used.



Parameters

iSession The session identifier as returned by LceGetSessionId.

Return Values

TRUE is returned if the session has performed the function.

FALSE is returned if an error occurs.

Notes

LceRollBackWork can be called after one or more inserts, updates or deletes have been performed.

Commitment control allows you to recreate events from a specific point which would otherwise be lost due to a system or application failure.

Commitment control only applies to files for which it has been specified on the host. It is your responsibility to ensure that commitment control has been properly configured on the host. Refer to the appropriate database manuals and Commitment Control in the LANSA Application Design Guide and Using Commitment Control in the LANSA for iSeries User Guide for more details about commitment control and commitment control processing.

Related Functions

6.54 LceSetCommitmentOff6.55 LceSetCommitmentOn6.9 LceCommitWork

6.52.1 VB Example

```
iRet = LceSetCommitmentOn(iSession)
iRet = LceOpenSession(iSession) ' enable commit mode
...
if (bCommit) then
    iRet = LceCommitWork (iSession)
else
    iRet = LceRollBackWork (iSession)
end if
```

6.52.2 C Example

BOOL fOK; int iSession;

iSession = LceGetSessionId fOK = LceSetCommitmentOn(iSession); fOK = LceOpenSession(iSession);

/* perform file updates */
fOK = LceRollBackWork(iSession);

6.53 LceSelect

Select and retrieve specified fields from the file using the key list specified.

BOOL	LceSelect	(int	iSession,
		char far *	strFieldList,
		char far *	strFileName,
		char far *	strKeyList,
		BOOL	fGeneric,
		int far *	piRecordCount)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strFieldList	A list of fields to be retrieved from the file. Each of these fields must have been declared using an LceUseField function.
strFileName	The name of the file. This must have been declared using an LceUseFile function.
strKeyList	A list of key field names. These key fields must have been declared using the LceUseField function. The values for the keys are set using the LceSetFieldValue/ LceSetFieldValueU function. Refer to the notes for key and generic search considerations.
fGeneric	Refer to the notes for further information. True = apply a generic search. False = do not apply a generic search.

piRecordCount The number of records returned by the select.

Return Values

TRUE is returned if the session has selected records.

FALSE is returned if an error occurs.

Notes

Portability Considerations

An error will occur if you attempt to use a generic key which is numeric if you are using a Windows, Linux or other non-IBM i host system.

The characteristic of this LceSelect search can be specified using the LceSetSelectOptions function.

Note: You cannot use the *RECEIVEIMMED option with LceSelect.

This function is a simplified version of these functions:

- LceRequestSelect
- LceReceiveSelect
- LceGetRecordCount.

Instead of calling these three functions, all the records are returned after the one LceSelect call.

In some circumstances the use of LceSelect may not be the best option. If the Host is slow or there are many records to return, then it may be better to use the LceRequestSelect, LceReceiveSelect and LceGetRecordCount functions, as LceRequestSelect returns control to the caller while the host prepares the data. While this is happening, your application can perform other tasks locally.

Key and Generic search considerations:

- You must know the specific key fields to the file. The key fields listed can be a subset of the actual keys to the file, but the sequence in which the keys are listed must match the sequence of the keys in the file. This is very important. The sequence of the keys must match the file. If you need the keys in a different sequence you must use a logical file.
- The fGeneric flag determines whether the keys must match exactly or whether a generic key search should apply. If a generic search is used on a file with more than one key field, then the search will apply only to the last non-blank key field specified. Any blank key field will be considered generic.
- You do not have to use the complete key list. You could use the first two fields in a file with three key fields. If you use a subset of the key fields, you are implicitly using a generic search on the unspecified keys, i.e. match key1 and key2 while key3 can have any value. In this case, you do not use the fGeneric search. If you did use the generic search in this situation, you will get a generic search based on key2 and key3.

• This function can only be executed once a session has been opened.

Related Functions

6.64 LceSetSelectOptions

6.50 LceRequestSelect

6.49 LceReceiveSelect

- 6.51 LceRequestSelectWhere
- 6.37 LceGetRecordCount
- 6.38 LceGetSelect
- 6.11 LceDeleteSelect

6.53.1 VB Example

Dim iRet As Integer, iRecords As Integer, iCount As Integer, _ sBuff As String, lFlags As Long

iRet = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25")
iRet = LceSelect(iSession, "SURNAME, SALARY", "PSLMST", "", _
 LceFalse, iRecords)

sBuff = space(256, chr(0))

For iCount=1 to iRecords

iRet = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME")
iRet = LceGetFieldValueX(iSession, "SURNAME", sBuff, lFlags)
lstNames.AddItem sTrim (sBuff)
lstNames.ItemData (lstNames.NewIndex) = iCount

Next

6.53.2 C Example

```
BOOL fOK;
int iSession;
int iRecordCount;
int i;
long lFlags;
fOK = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25");
fOK = LceSelect(iSession, "SURNAME, SALARY", "PSLMST", "",
FALSE, &iRecordCount);
for (i=1; i <= iRecordCount; i++)
{
    fOK = LceGetSelect(iSession, "PSLMST", i, "SURNAME, SALARY");
    fOK = LceGetFieldValueX(iSession, "SURNAME", (char far *) strSurname
}
```

6.54 LceSetCommitmentOff

Turns off commitment control on the host system.

BOOL LceSetCommitmentOff (int iSession)

Parameters

iSession The session identifier as returned by LceGetSessionId.

Return Values

TRUE is returned if commitment control is successfully turned off. FALSE is returned if an error occurs.

Notes

Once this function has been executed, the LceCommitWork or LceRollBackWork cannot be used.

You must end this session before you can turn commitment control on again.

- This function must be executed before the LceOpenSession function.
- This function performs no communications to the host.

Related Functions

6.55 LceSetCommitmentOn6.9 LceCommitWork6.52 LceRollBackWork

6.54.1 VB Example

' set commit mode iSession = LceGetSessionId(): If bCommit = True Then iRet = LceSetCommitmentOn(iSession) Else iRet = LceSetCommitmentOff(iSession) End If iRet = LceOpenSession(iSession):

6.54.2 C Example

BOOL fOK; int iSession;

iSession = LceGetSessionId(); fOK = LceSetCommitmentOff(iSession); fOK = LceOpenSession(iSession);

6.55 LceSetCommitmentOn

Turns on commitment control on the host system.

```
LceSetCommitmentOn (int iSession)
```

Parameters

iSession The session identifier as returned by LceGetSessionId.

Return Values

TRUE is returned if commitment control has been successfully turned on. FALSE is returned if an error occurs.

Notes

Commitment control allows you to recreate events from a specific point which would otherwise be lost due to a system or application failure. Host commitment control only applies to files for which it has been specified. It is your responsibility to ensure that commitment control has been properly configured on the host. Refer to the appropriate IBM manuals and Commitment Control in the LANSA Application Design Guide and Using Commitment Control in the LANSA for iSeries User Guide for more details about commitment control and commitment control processing.

This function must be executed before you can use the LceCommitWork or LceRollBackWork functions.

You must end this session if you wish to switch commitment control off.

- This function must be executed before the LceOpenSession function.
- This function performs no communications to the host.

Related Functions

6.54 LceSetCommitmentOff6.9 LceCommitWork6.52 LceRollBackWork

6.55.1 VB Example

' set commit mode iSession = LceGetSessionId(): If bCommit = True Then iRet = LceSetCommitmentOn(iSession) Else iRet = LceSetCommitmentOff(iSession) End If iRet = LceOpenSession(iSession):
6.55.2 C Example

BOOL fOK; int iSession;

iSession = LceGetSessionId();

fOK = LceSetCommitmentOn(iSession);

fOK = LceOpenSession(iSession);

6.56 LceSetCommsCursor

The Windows hourglass cursor is used with most Windows products to indicate that communication between the client and server is in progress. This function allows you to vary the type of cursor or to have no cursor at all.



Parameters

iCursorType The Cursor type to use. Refer to the notes for the cursor types available.

Return Values

TRUE is returned if the function is successful.

FALSE is returned if an incorrect cursor type was specified.

Notes

The available cursor types are:

LCE_CURSOR_DEFAULT

The standard LANSA Open cursor. This is LANSA Open's default communications cursor.

LCE_CURSOR_HOURGLASS

The standard hourglass cursor which could be used when LANSA Open is used with a product, such as a spreadsheet, which may use this cursor.

LCE_CURSOR_NONE

No cursor will be set when communications is in progress. If the calling application does not set the cursor itself, then the standard Windows pointer will remain displayed.

• This function performs no communications to the host.

Tip

Make sure that the most frequently used cursor type is set in the System Registry as described in Communication and Session Parameters in the Installing LANSA on Windows Guide.

6.56.1 VB Example

iRet = LceSetCommsCursor(LCE_CURSOR_HOURGLASS)

6.56.2 C Example

C Example 1

LceSetCommsCursor(LCE_CURSOR_NONE);

C Example 2

iCursorDef = 1 iCursorGlass = 2 iCursorNone = 3

CALL LceSetCommsCursor(iCursorDef)

6.57 LceSetCommsType

The Communications Type is detected automatically. This function is provided for backward compatibility only. It performs no actions and will return immediately with a return value of TRUE.

BOOL LceSetCommsType (int iCommsType)

6.58 LceSetFieldValue

Sets the field value in the field information area. This function is very important. It is used to assign values to fields in an information area, so that they may be used in other functions, such as LceFetch.

BOOL	LceSetFieldValue	(int	iSession,
		char far *	strFieldName,
		char far *	strValue)

Parameters

iSession	The session identifier as returned by LceGetSessionId.
strFieldName	The name of the field as used in the Repository. This field must have been declared using an LceUseField function.
strValue	The string which contains the value. The length of this parameter should not exceed RDMLX_FIELD_DATA_SIZE + 1 as defined in the LCOE.H header file. A NULL pointer indicates the value is *SQLNULL.

Return Values

TRUE is returned if the session has set the field value.

FALSE is returned if an error occurs.

Notes

This concept may be difficult to understand, but you must remember that you are accessing data on a server and not locally. You do not have the files and their field characteristics stored locally. The file and field definitions are not compiled into your applications. As a developer, you may know that the employee number "EMPNO" is an integer, but the application has no way of knowing this until it connects to the server. Consequently, the intermediate step of setting field values and getting field values is required.

The LceSetFieldValue function uses string parameters. Even if the field you are setting is a numeric, it must be assigned a string value. You must convert the numeric value to a string before using it in this function.

An error will be generated if you attempt to assign *SQLNULL to a field which

does not have the ASQN input attribute. Use LceGetFieldAttributes to determine if a field can be assigned *SQLNULL.

If the field is a Unicode type (NChar or NVarchar), the value will be automatically converted to Unicode and the LceSetFieldValueU function called.

- This function may be executed at any time after the session is opened.
- This function performs no communications to the host.

Related Functions

LceSetFieldValueU 6.29 LceGetFieldValueX 6.18 LceGetDefaultValueU

6.58.1 VB Example

```
Private Sub txtCode_Change()
```

'whenever txtCode changes, set field value.

```
' This is safer than using lost focus, as the field can
```

' change without getting focus

Dim iRet As Integer

If txtCode <> "" Then

```
iRet = LceSetFieldValue(iSession, "SKILCODE", txtCode)
```

Else

```
' "" is not a null terminated string
```

```
iRet = LceSetFieldValue(iSession, "SKILCODE", vbNullString)
End If
```

End

6.58.2 C Examples

C Example 1

BOOL fOK; int iSession;

fOK = LceSetFieldValue(iSession, "EMPNO", "12345");

C Example 2

BOOL fOK; int iSession;

```
fOK = LceSetFieldValue(iSession, "EMPNO", NULL);
```

C Example 3

```
strcpy(strField, "EMPNO");
if !LceSetFieldValue(iSession, strField, "12345")
{
     /* return error */
}
```

Note: You are not assigning strField = "12345". You are assigning the field "EMPNO", which happens to be stored in the variable strField, to the value of "12345".

C Example 4

```
iValue = 12345;
sprintf(strValue, "%ld", iValue);
strcpy(strKey, "EMPNO");
if !LceSetFieldValue(iSession, strKey , strValue)
{
    /* return error */
}
```

Note: The field "EMPNO" may be a numeric value in a file; however, the field must be assigned a string value. If the value entered by a user was an integer, for

example iValue, it must be converted to a string before it is used in the LceSetFieldValue function.

6.59 LceSetFieldValueU

Sets the field value in the field information area. This function should be used for Unicode fields. Unicode fields are field types NChar and NVarchar. This function is very important. It is used to assign values to fields in an information area, so that they may be used in other functions, such as LceFetch.

BOOL	LceSetFieldValue	(int	iSession,
		char far *	strFieldName,
		wchar_t far *	strValue)

Parameters

strFieldName The name of the field as used in the Repository. This field must have been declared using an LceUseField function.

strValue The string which contains the value. The length of this parameter should not exceed RDMLX_FIELD_DATA_SIZE + 2 as defined in the LCOE.H header file. A NULL pointer indicates the value is *SQLNULL.

Return Values

TRUE is returned if the session has set the field value.

FALSE is returned if an error occurs.

Notes

See notes for LceSetFieldValue.

If this function is used with non-Unicode fields, it will automatically convert the value to an ascii value and call LceSetFieldValue.

- This function may be executed at any time after the session is opened.
- This function performs no communications to the host.

Related Functions

6.28 LceGetFieldValue6.29 LceGetFieldValueX6.18 LceGetDefaultValueU

6.59.1 VB Example

```
Private Sub txtCode_Change()
```

'whenever txtCode changes, set field value.

```
' This is safer than using lost focus, as the field can
```

' change without getting focus

Dim iRet As Integer

```
If txtCode <> "" Then
```

```
iRet = LceSetFieldValueU(iSession, "TOWN", txtCode)
```

Else

```
' "" is not a null terminated string
```

```
iRet = LceSetFieldValueU(iSession, "TOWN", vbNullString)
End If
```

End

6.59.2 C Examples

C Example 1

BOOL fOK; int iSession;

fOK = LceSetFieldValueU(iSession, "TOWN", "宮代町");

C Example 2

BOOL fOK; int iSession;

```
fOK = LceSetFieldValueU(iSession, "TOWN", NULL);
```

C Example 3

```
strcpy(strField, "TOWN");
if !LceSetFieldValueU(iSession, strField, "Henty")
{
     /* return error */
}
```

Note: You are not assigning strField = "12345". You are assigning the field "EMPNO", which happens to be stored in the variable strField, to the value of "12345".

6.60 LceSetHostType

The Host Type is detected automatically. This function is provided for backward compatibility only. It performs no actions and will return immediately with a return value of TRUE. If your application needs to know the host type, use the new function LceGetSystemAttribute.

BOOL LceSetHostType (int iHostType)

Related Functions

6.41 LceGetSystemAttribute

6.61 LceSetIBMiSignon

Sets the password of the user profile on the IBM i host.

BOOL	LceSetIBMiSignon	(char far *	strServer,	
		int	iMapperPort,	
		BOOL	fSSLRequired	
		char far *	strUserid,	
		char far *	strPassword,	
		char far *	strNewPassword,	
		BOOL	fEncryptPassword,	
		char far *	strReturnCode)	

Parameters

strServer	The IBM i server name. The length should not exceed SYSTEM_LENGTH as defined in LCOE.h.
iMapperPort	Defaults to 449 if passed as 0.
fSSLRequired	This parameter provides the ability to use SSL to communicate with the IBM i server. True = Use SSL False = Do not use SSL.
strUserid	The User Identifier. The length should not exceed USERID_LENGTH as defined in the LCOE.h header file.
strPassword	The password. The length should not exceed PASSWORD_LONG_LENGTH as defined in the LCOE.H header file
strNewPassword	The new password. The length should not exceed PASSWORD_LONG_LENGTH as defined in the LCOE.H header file.
fEncryptPassword	This parameter provides the ability to encrypt the

	password in communications with the IBM i server. True = Encrypt the password. False = Do not encrypt the password
strReturnCode	The buffer location for the return code. The size should be at least RETURN_CODE_LENGTH + 1.
	The return codes returned:
	OK - Signon is OK
	CE - Communications error
	LE - Local Encryption error
	NR - User name does not exist
	SE - Server error
	WP - Wrong Password
	UD - User name is disabled
	LP – Password is too long
	NE – New Password Error

Return Values

TRUE indicates that request completed without error.

FALSE indicates an error occurred.

Notes

Because of the way that the IBM i operating system handles user names and short passwords (Password level 0 or 1) with the US English (CCSID 037) characters '@', '#' and '\$', this facility will only work with such user names and short passwords if the IBM i is operating in US English (CCSID 037).

The current implementation of SSL used for this facility ensures that encryption is negotiated and used for communication between the client and the IBM i server. It does not verify that the IBM i server is that specified on the security certificate that has been downloaded.

The interplay between SSL Required and Encrypt Password is interesting. If SSL is available and SSL Required is TRUE, then strictly speaking password encryption is not needed because the entire communication stream is encrypted, so Encrypt Password could be specified as FALSE. If SSL Required is FALSE, then we recommend that Encrypt Password be specified as TRUE.

The reasons for Return Code CE -Communications error can include:

- a misspelling in the IBM i Server name;
- the IBM i Server name not being locatable by your DNS;
- a firewall between the local computer and the IBM i server;
- the IBM i server being offline;
- TCP/IP not being started on the IBM i server;
- TCP/IP host servers not being started on the IBM i server;
- SSL Required TRUE and the SSL TCP/IP host servers not being started on the IBM i server;
- SSL Required FALSE and the non-SSL TCP/IP host servers not being started on the IBM i server.

If Return Codes 'SE - Server Error' or 'NE - New Password Error' is returned, a review of the joblog for the QZSOSIGN job on the IBM i server should show the reason in detail.

Related Functions

6.12 LceDisplayErrors6.34 LceGetIBMiSignon

6.61.1 VB Example

Declare Function LceSetIBMiSignon Lib "LCOEW32" (_ ByVal sServer As String, _ ByVal iMapperPort As Integer, _ ByVal bSslRequired As Byte, _ ByVal sUserId As String, _ ByVal sPassword As String, _ ByVal sNewPassword As String, _ ByVal bEncryptPassword As Byte, _ ByVal sReturnCode As String _) As Integer

Sub LopenTest1()

Dim sServer As String Dim iMapperPort As Integer Dim bSslRequired As Byte Dim sUserId As String Dim sPassword As String Dim sNewPassword As String Dim bEncryptPassword As Byte Dim sReturnCode As String Dim i As Integer

```
sServer = "myserver"

iMapperPort = 0

bSslRequired = False

sUserId = "myuser"

sPassword = "mypasswd"

sNewPassword = "mynewpassword"

bEncryptPassword = True
```

```
sReturnCode = String(3, Chr(0))
sExpiryDate = String(20, Chr(0))
```

Call LceSetIBMiSignon(sServer, iMapperPort, bSslRequired, sUserId, sPassword, sNewPassword, bEncryptPassword, sReturnCode)

Sheet1.Columns("B").ColumnWidth = 20 Sheet1.Cells(4, "B") = "Return Code" Sheet1.Cells(4, "C") = sReturnCode

End Sub

6.61.2 C Example

```
char strServer[ SYSTEM_LENGTH + 1 ] = "myserver";
int iMapperPort = 0;
BOOL fSSLRequired = false;
char strUserid[ USERID_LENGTH + 1 ] = "myuser";
char strPassword[ PASSWORD_LONG_LENGTH + 1 ] = "mypasswd";
char strNewPassword[ PASSWORD_LONG_LENGTH + 1 ] =
"mynewpassword";
BOOL fEncryptPassword = true;
char strReturnCode[ RETURN_CODE_LENGTH + 1 ];
BOOL rc;
```

rc = LceSetIBMiSignon(strServer, iMapperPort, fSSLRequired, strUserid, strPassword, strNewPassword, fEncryptPassword, strReturnCode);

```
if (rc)
{
 if ( strcmp( strReturnCode, "OK" ) == 0 )
  {
   printf( "Signon details successfully changed." );
  }
  else
  ł
   printf( "The request to the server has failed with return code: %s\n",
strReturnCode );
  }
}
else
{
 printf( "The function failed to run successfully\n" );
}
```

6.61.3 C# Example

```
namespace LansaOpenNative
{
 class LansaOpen
  {
   [DllImport( "lcoew32.dll", CharSet = CharSet.Ansi )]
   public static extern int LceSetIBMiSignon(
     StringBuilder
                                    server,
                               mapperPort,
     int
     int
                               sslRequired,
     StringBuilder
                                    userId,
     StringBuilder
                                    password,
                                    newPassword,
     StringBuilder
                               encryptPassword,
     int
     StringBuilder
                                    returnCode );
 }
}
private void LOpenSample()
 String strServer = "myserver";
 int iMapperPort = 0;
 bool fSSLRequired = false;
 String strUserid = "myuser";
 String strPassword = "mypasswd";
 String strNewPassword = "mynewpassword";
 bool fEncryptPassword = true;
 StringBuilder strReturnCode = new StringBuilder( 3 );
 bool rc;
 rc = LansaOpen.LceSetIBMiSignon(
   new StringBuilder( strServer ),
   iMapperPort,
   (fSSLRequired?1:0),
   new StringBuilder( strUserid ),
   new StringBuilder( strPassword ),
```

```
new StringBuilder( strNewPassword ),
   (fEncryptPassword?1:0),
   strReturnCode );
 if (rc)
 {
   if ( strReturnCode.ToString() == "OK" )
   {
     txtStatusField.Text = "Signon details successfully changed";
   }
   else
   {
     txtStatusField.Text = String.Format( "The request to the server has failed
with return code: {0}", strReturnCode );
   }
  }
 else
  {
   txtStatusField.Text = "The function failed to run successfully";
  }
}
```

6.62 LceSetKerberosOff

This function indicates that the connection to the server should no longer use Windows Credentials (Kerberos / Single Signon / SSO).

BOOL LceSetKerberosOff (int iSession)

Parameters

iSession The session identifier as returned by LceSetKerberosOff.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

If you use this function, you must also use the LceSetKerberosOn function.

• This function must be executed after LceGetSessionId and before LceOpenSession.

Related Functions

6.63 LceSetKerberosOn

6.63 LceSetKerberosOn

This function indicates that the connection to the server will be made using Windows Credentials (Kerberos / Single Signon / SSO) rather than the User Id. This means that the user's Windows profile and password is used to sign on to the server. The server must have been configured for Single Sign On, and the user enrolled, before this can be done.

BOOL LceSetKerberosOn (int iSession)

Parameters

iSession The session identifier as returned by LceSetKerberosOn.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

If you use this function, you must also use the LceSetKerberosOff function.

• This function must be executed after LceGetSessionId and before LceOpenSession.

Kerberos works without further configuration directly to a server with no access outside that server, say to SQL Server or a file share.

If access outside of that 1 server is required – so called "multi-hop" – then this is what is supported:

1.Trust whole computer to *any* services – We have tested and proved this is working

2.Trust a specific domain user to *any* services – We have tested and proved this is working (this requires setting up listener properly to run as a specific user, see the attached document. This should be verified first using lcoecho)

If your environment does not allow one of these configurations then multi-hop cannot be used.

Related Functions

6.62 LceSetKerberosOff

6.64 LceSetSelectOptions

Specifies the options to be used for the next **SELECT** or **FETCH** function. This function sets the characteristic of the search when an LceRequestSelect or LceSelect function is used. It is also used for record locking when an LceFetch function is used.

BOOL	LceSetSelectOptions	(int	iSession,
		char far *	strOptions)

Parameters

iSession The session identifier as returned by LceGetSessionId.

strOptions A comma-separated list of **SELECT** or **FETCH** options. The options used will revert to their default values after the next LceRequestSelect or LceFetch function. Refer to the Notes for details.

strOptions

Valid **SELECT** options:

*BACKWARDS	Records are retrieved in reverse order.
*STARTKEY	This key is used as the starting position for the select.
*MAXRECnn	Select a maximum of nn records.
*BLOCKBYKEY[nnnn]	Block by key. Only valid on IBM i hosts.
*BLOCKBYRRN[nnnn]	Block by relative record number. Only valid on IBM i hosts.
*RECEIVEIMMED	Receive immediately.

Valid **FETCH** option:

*LOCK Locks the record to prevent access by another application.

If no options are specified, or if one is missed, the default for that option is used. After the next Select request, these defaults are re-instated.

The default options for **SELECT** are:

Forward record retrieval.

No *STARTKEY option.

No record selection limit.

No record locking.

No blocked I/O performed.

The records will not be returned until they've all been read at the host.

The default option for **FETCH**:

No record locking.

Return Values

TRUE is returned if valid options are set. FALSE is returned if an error occurs.

Notes

Portability Considerations	The options * BLOCKBYKEY and * BLOCKBYRRN and * LOCK are only valid on IBM i hosts. They are ignored on other platforms.
	For RDMLX files, *BLOCKBYKEY and *BLOCKBYRRN are always ignored.
	For RDML files, *BLOCKBYKEY and *BLOCKBYRRN are valid for IBM i hosts, but ignored on other platforms.
	The option *LOCK is only valid on IBM i hosts. It is ignored on other platforms. For RDMLX files, if the OAM cannot support locking because it is using SQL, an error will be returned from the LceFetch call.

***BACKWARDS**

Records are retrieved in reverse order. This option indicates that rows are to be processed in reverse order to that which would normally be used. Backwards processing by sequential, full or partial key is supported.

This option must not be used with either of the *BLOCKBY options.

*STARTKEY

The key is used as the starting position for the first read operation. The first record read will be the first one which has a key equal to or greater than the key values specified. All subsequent records are then processed with no regard to the keys.

*MAXRECnn

This option is used to indicate the maximum number of records which are to be returned. It can be used with all other select options.

The nn parameter must be specified and it must be greater than zero, otherwise no records will be returned.

*BLOCKBYKEY[nnnn]

Block by key.

This is a select option that will improve application performance, depending on whether the data is in the same physical order as the primary key of the file. *BLOCKBYKEY is generally less efficient than *BLOCKBYRRN unless the IBM i physical file has been recently reorganized. (Reorganization ensures that the data is in the same order as the file's primary key.)

This option cannot be specified with the *BLOCKBYRRN option. i.e. You cannot specify both options.

Important Note: This option is only valid on IBM i hosts. It is ignored on other platforms.

Note the following about using this option:

- It may only be used when an entire file is to be processed from end to end.
- The "nnnn" portion of the option is optional. This is the blocking factor that the host based I/O module should use when reading data during the select operation. If used, it must be an integer in the range 1 to 9999. Generally the "nnnn" option should be reserved for use in high speed file transfer operations. When the "nnnn" option is not specified, the OS/400 operating system will automatically calculate an appropriate value.
- Before using this option, the associated I/O module on the host must have been recompiled some time after the system definition data area, DC@OSVEROP, was set to contain options *IOMXSERVER and *IOMBLOCKBYKEY. Failure to do this will cause your application to fail. Refer to Version Dependancy Data Area DC@OSVEROP in the LANSA for iSeries User Guide for more details about data area DC@OSVEROP.
- This option may be used with the *RECEIVEIMMED option.

- Rows are returned in the order of the primary key of the physical file from which data is being selected. The host's relative record numbers are not available and have no defined value when used with this option.
- The following LceRequestSelect or LceRequestSelectWhere functions may only refer to a physical file. References to logical files are not supported and may cause unpredictable results and/or errors.
- The key fields must not be specified on the following LceRequestSelect or LceRequestSelectWhere functions.
- This option only persists until the next Select request. After the Select request is executed, this option is destroyed and the defaults are reinstated.

*BLOCKBYRRN[nnnn]

Block by relative record number.

*BLOCKBYRRN is generally the most efficient type of IBM i sequential database file access. Used correctly and with an appropriate blocking factor, it can improve sequential access time by a factor of 10 or more. This improvement may not be significant, however, if communications between the IBM i and your PC are slow.

This option cannot be used with the *BLOCKBYKEY option. i.e. You cannot specify both options.

Important Note: This option is only valid on IBM i hosts. It is are ignored on other platforms.

Note the following about using this option:

- It may only be used when the entire file is to be processed from end to end.
- The "nnnn" portion of the option is optional. This is the blocking factor that the IBM i based I/O module should use when reading data during the select operation. If used, it must be an integer in the range 1 to 9999. Generally the "nnnn" option should be reserved for use in high speed file transfer operations. When the "nnnn" option is not specified, the OS/400 operating system will automatically calculate an appropriate value.
- Rows are returned in the order of their relative record number (i.e. the order the rows were created or organized into) from the physical file. The IBM i's relative record numbers are not available for use with this option.
- Before using this option, the associated I/O module on the IBM i must have been recompiled some time after the system definition data area, DC@OSVEROP, was set to contain options *IOMXSERVER and *IOMBLOCKBYKEY. Failure to do this will cause application failure.

Refer to Version Dependancy Data Area DC@OSVEROP in the LANSA for *iSeries User Guide* for more details about data area DC@OSVEROP.

- *BLOCKBYRRN can be used with the *RECEIVEIMMED option.
- This option only persists until the next Select request. After the Select request is executed, this option is destroyed and the defaults are reinstated.

***RECEIVEIMMED**

Receive immediately, to indicate that the next Select operation is to be performed in "immediate" mode. This option may substantially improve the time taken to select rows, particularly where a large number are involved. Note the following about using this option:

- This option only persists until the next Select request. After the Select request is executed, this option is destroyed and the system's defaults are reinstated.
- This option can only be used with the LceRequestSelect and LceRequestSelectWhere functions not with LceSelect.
- Once a "Select" has been issued under this option, ALL selected records must be received by using LceReceiveNextX until LceReceiveNextX indicates that there are no more rows to receive. An LceDeleteSelect can be issued to purge any remaining records.
- This option cannot be used with the LceSelect function or the LceReceiveSelect, LceGetSelect and LceGetRecordCount functions. Doing so will lead to unpredictable results and/or application failure.
- While repeated LceReceiveNextX operations are receiving ALL selected records, no other I/O operation at all (e.g. LceFetch) is permitted with the same Session Id, regardless of whether it involves the same file or not.
- This function can only be executed once a session has been opened.

The LceSetSelectOptions function is used with a number of other select processing functions. The general pattern of these functions is as follows if the *RECEIVEIMMED option is **not** used:

- LceSetSelectOptions sets the selection search characteristics.
- LceRequestSelect selects records on the host.
- LceReceiveSelect transfers the records from the host to the PC.
- LceGetRecordCount determines how many records were transferred.
- LceGetSelect retrieves a single record from the group of records transferred.

If the *RECEIVEIMMED option is used, then the functions will be as follows:

- LceSetSelectOptions sets the selection search characteristics.
- LceRequestSelect selects records on the host.
- LceReceiveNextX reads the next records sent and stored on the PC.

Related Functions

6.15 LceFetch
6.50 LceRequestSelect
6.49 LceReceiveSelect
6.51 LceRequestSelectWhere
6.48 LceReceiveNextX
6.37 LceGetRecordCount
6.38 LceGetSelect
6.53 LceSelect
6.11 LceDeleteSelect

6.64.1 VB Example

iRet = LceDeleteSelect(iSession, "SKLTAB")
' Set Select to IMMEDIATE MODE is recommended for reasons of speed
iRet = LceSetSelectOptions(iSession, "*RECEIVEIMMED")

Refer to the Visual Basic sample programs for further examples of using this function.

6.64.2 C Examples

C Example 1

```
Set characteristics for a select search:
BOOL fOK;
int iSession;
fOK = LceSetSelectOptions(iSession, "*BACKWARDS, *MAXREC25");
fOK = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", "", FA
fOK = LceReceiveSelect(iSession, "PSLMST");
fOK = LceGetRecordCount(iSession, "PSLMST", &iRecords);
for (iCount=0; iCount < iRecords; i++)
{
    fOK = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME, SALARY
}
```

C Example 2

Lock a record during a fetch: BOOL fOK; int iSession;

fOK = LceSetSelectOptions(iSession, "*LOCK");

fOK = LceSetFieldValue(iSession, "EMPNO", "12345"); fOK = LceFetch(iSession, "SURNAME, GIVENAME", "PSLMST", "EMPNO")

C Example 3

Use *RECEIVEIMMED to Select Records:

BOOL fOK; int iSession; long lFlags;

fOK = LceSetSelectOptions(iSession, "*RECEIVEIMMED");

fOK = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", "", FA

fOK = LceReceiveNextX(iSession,"",0,NULL,0);

```
fOK = LceGetFieldValueX(iSession, "SURNAME", strSurname, lFlags);
fOK = LceGetFieldValueX(iSession, "SALARY", strSalary, lFlags);
```

C Example 4

```
Use blocking by key:
BOOL fOK;
int iSession;
fOK = LceSetSelectOptions(iSession, "*BLOCKBYKEY");
```

```
fOK = LceRequestSelect(iSession, "SURNAME, SALARY", "PSLMST", "", FA
fOK = LceReceiveSelect(iSession, "PSLMST");
fOK = LceGetRecordCount(iSession, "PSLMST", &iRecords);
for (iCount=0; iCount < iRecords; i++)
{
    fOK = LceGetSelect(iSession, "PSLMST", iCount, "SURNAME, SALARY
}
```

6.65 LceSetSessionOption

This function enables you to switch options to:

- vary the format in which data is returned to you (and to set and reset the option during the session).
- specify the time of file opening.

The default values are reinstated when the session is closed.

This function will be extended to support more options as they are required.

BOOL	LceSetSessionOption	(int	iSession,
		int	iOption,
		char far *	strValue)

Parameters

iSession The session identifier.

iOption The option to be set. Available options are listed in the Notes.

strValue The value of the option. Refer to the Notes for the available values.

Return Values

TRUE is returned if the option was set successfully.

FALSE is returned if an invalid option or value was specified.

Notes

The default values required for these options should be set in the Windows System Registry. If these defaults are not set, LANSA Open sets those described with each option.

Refer to Configuration File in the PC Configuration File section of this guide for further information about setting the default values.

LCE_OPT_NUMERIC_DATA

This option can be set and reset at any time during a session.

- "PAD" causes LANSA Open to return all numeric values with leading zeros. This can be useful if using PACKED data to store dates.
- "STRIP" causes LANSA Open to return all numeric values with the leading zeros stripped.

LANSA Open's default: Numeric values will be returned stripped of all leading zeros and only containing a leading sign if negative.

LCE_OPT_ALPHA_DATA

This option can be set and reset at any time during a session.

- "PAD" causes LANSA Open to return all alpha values with the trailing spaces up to their correct length.
- "STRIP" causes LANSA Open to return all alpha values with the trailing spaces stripped.

LANSA Open's default: Alpha values are returned stripped of all trailing spaces.

LCE_OPT_EDIT_DATA

This option can be set and reset at any time during a session.

This option allows you to specify the way in which data is returned to you by LceGetFieldValueX.

- "Y" causes LANSA Open to format the data before it is returned. If you have recorded an Edit Code for the field definition in the Repository, then the data will be returned in the format associated with that edit code. For example, a date could be returned as mm/dd/yy instead of as straight numeric. See under Creating a New Field Definition for *Input Options when Creating a New Field* in the *LANSA for iSeries User Guide* for more information about Edit Codes.
- "N" causes LANSA Open to return the data unedited.

LANSA Open's default: Data returned is not reformatted, regardless of the field's Edit Code.

LCE_OPT_OPEN_FILE_ON_DEMAND

This option must be specified before the session is opened

- "Y" causes LANSA Open to open files only when they are used. This saves considerable time when many files are used in a session, however, if a file without an I/O module is accessed during the course of processing, unacceptable processing delays may occur.
- "N" causes LANSA Open to open all the files to ensure that they all have I/O modules, before returning from the call to LceOpenSession. If a file without an I/O module is detected, LceOpenSession will fail.

LANSA Open's default: All the files are opened at the beginning of a session, to ensure that all the I/O modules exist.
6.65.1 VB Example

' Set Numeric padding If iRet = LceTrue Then iRet = LceSetSessionOption(iSession, ______ LCE_OPT_NUMERIC_DATA, "STRIP") ' set alphanumeric padding If iRet = LceTrue Then iRet = LceSetSessionOption(iSession, ______ LCE_OPT_ALPHA_DATA, "PAD") ' set edit characters retrieved with data If iRet = LceTrue Then iRet = LceSetSessionOption(iSession, ______ LCE_OPT_EDIT_DATA, "Y") ' set open files only on demand If iRet = LceTrue Then iRet = LceSetSessionOption(iSession, ______ LCE_OPT_OPEN_FILE_ON_DEMAND, "Y")

6.65.2 C Example

BOOL fOK; int iSession;

fOK = LceSetSessionOption(iSession, LCE_OPT_NUMERIC_DATA, "STRIP")
fOK = LceSetSessionOption(iSession, LCE_OPT_ALPHA_DATA, "PAD");
fOK = LceSetSessionOption(iSession, LCE_OPT_EDIT_DATA, "Y");
fOK = LceSetSessionOption(iSession, LCE_OPT_OPEN_FILE_ON_DEMAND)

6.66 LceSubmit

LceSubmit is used to submit a LANSA function/process or an IBM i program, such as a report program, to run in batch mode. If you submit a program that expect user interaction, the job will fail.

There is no communication between the called program and your calling application once the call has been made. A return value, indicating whether the job has been successfully submitted is passed back to your application. Your application does not have to wait for the program to complete.

BOOL	LceSubmit	(int	iSession,
		char far *	strProcess,
		char far *	strFunction
		char far *	strProgram,
		char far *	strExchangeList,
		char far *	strJobName,
		char far *	strJobDesc,
		char far *	strJobQueue,
		char far *	strOutputQueue)

Parameters

iSession	The session identifier as returned by LceGetSessionId.		
strProcess	The name of the LANSA process.		
strFunction	The name of the LANSA function.		
	Or		
strProgram			
Suriogram	The name of the IBM i program.		
0	A list of field names.		
0	1 0		

strJobQueue The job queue. The default value is *JOBD.

strOutputQueue The output queue. The default value is *JOBD.

Return Values

TRUE is returned if the session can execute the process/function or program. FALSE is returned if an error occurs.

Notes

Portability Considerations	The format which uses strProgram is only supported if your host system is an IBM i. An error will be returned by the host if you make a call with strProgram to any other platform.
	When calling a program on the IBM i, the exchange list may only contain fields of type A, P, and S. If exchanging numeric fields, the maximum length is 30 and the maximum decimals is 9.

If you are not familiar with the concept of job name, job description, job queue and output queue, contact your host Administrator.

Parameters are passed to the called program or function using the list of field names in strExchangeList. Each field used in the exchange must have been declared using an LceUseField function. The value for the field is assigned using the LceSetFieldValue/LCeSetFieldValueU function. The LANSA process and function to be executed do not have to be declared previously by an Lce function call.

• This function can only be executed after a session has been opened.

Tip

If you wish to execute an IBM i program that either passes or receives parameters, use the Lce3GLCall function.

LceExecute400 function can also be used to submit IBM i jobs, but its use is much more complicated and it is best suited to IBM i commands.

Related Functions

6.44 LceLansaCall6.14 LceExecute4006.3 Lce3GLCall

6.66.1 VB Example

```
Private Sub cmdPrintAll_Click()

' set department to blank to print all departments

iRet = LceSetFieldValue(iSession, "DEPTMENT", Chr(0))

If iRet = LceTrue Then

iRet = LceSubmit(iSession, "PSLSYS", "EMPLIST", "", _

"DEPTMENT", "", "", "")

End If

End Sub
```

6.66.2 C Example

BOOL fOK; int iSession;

fOK = LceSubmit(iSession, "PROCREP", "SALES01", "", "MONTH", "", "",

6.67 LceUnicodeToHex

This function converts a Unicode string into a formatted stream of hex bytes to be used in the where clause.

int	LceUnicodeToHex	(wchar_t far *	strString,
		char far *	strHexString,
		int	iHexStringSize)

Parameters

strString	Unicode string to be converted. Should be null terminated.	
strHexString	The generated stream of hex bytes.	
iHexStringSize	The size of the strHexString buffer. The size of the buffer will normally be ((4 * number of characters in strString) + 5).	

Return Values

If strHexString is NULL, then this function will return the size of the converted string (see Notes).

If strHexString is not NULL, the following is returned:

TRUE is returned if every character in the Unicode string has been converted without errors.

FALSE is returned if an error occurs or the buffer is not large enough to convert the entire string.

Notes

This function should be used when including a Unicode value in the where clause in LceRequestSelectWhere.

To determine the size of the buffer required to store the converted string you may call this function and pass NULL for the strHexString parameter. The function will not perform the conversion but will return the size of the buffer required for the conversion.

Related Functions

6.51 LceRequestSelectWhere

6.67.1 VB Example

Dim i As Integer Dim hexString As String Dim whereClause As String

iRet = LceSetSelectOptions(iSession, "*RECEIVEIMMED")
If iRet = LceFalse then exit sub ' Failed

hexString = String(200, Chr(0))
iRet = LceUnicodeToHex(name, hexString, 200)
If iRet = LceFalse then exit sub ' Failed

whereClause = "NAME = " + hexString

6.67.2 C Example

BOOL fOK; int iSession; char* strHexString; char strWhere[256]; int bufferSize;

fOK = LceSetSelectOptions(iSession, "*RECEIVEIMMED"); bufferSize = LceUnicodeToHex(strString, NULL, 0); strHexString = malloc(bufferSize) fOK = LceUnicodeToHex(strString, strHexString, bufferSize); sprintf(strWhere, "NAME = %s", strHexString); fOK = LceRequestSelectWhere(iSession, "EMPNO,SALARY","WORKERS", "",strWhere,FALSE); free(strHexString);

6.68 LceUpdate

This function updates an existing record in a file. The list of fields to be updated is specified in a list.

BOOL	LceUpdate	(int	iSession,
		char far *	strFieldList,
		char far *	strFileName,
		char far *	strKeyList,
		BOOL	fValidate)

Parameters

- iSession The session identifier as returned by LceGetSessionId.
- strFieldList A list of fields to be updated. These fields must have been declared using an LceUseField function. Only the fields specified in this list will be updated. The value for each of the fields in the list must be set using the LceSetFieldValue/LceSetFieldValueU function.
- strFileName The name of the file. This file must have been declared using an LceUseFile function.
- strKeyList A list of key field names. The fields must have been declared using an LceUseField function. The values for the keys are set using the LceSetFieldValue/LceSetFieldValueU function. Refer to the Notes for the benefits of leaving this key list blank.
- fValidate This parameter provides a " check but do not update" facility. True = Perform validate only.

False = Perform update immediately.

Return Values

TRUE is returned if the session has updated or can update the record. FALSE is returned if an error occurs.

Notes

When multiple users are working with a file, it is likely that two users will want to update the same record at the same time. LANSA Open will handle this situation automatically if you leave the strKeyList blank. In this case, if a user attempts to update a record which has already been changed by another user, LANSA Open issues an appropriate message and does not perform the update. Because LANSA Open does this check for you, record locking on the host is

unnecessary.

If you specify a key, then that record will be updated regardless of whether it has been changed since it was selected.

If you do specify keys, you do not have to use the complete key list. For example, you could use the first two fields in a file with three key fields. You must always use the keys in the correct order. If you use a subset of the key fields, you are implicitly using a "generic select" on the unspecified keys, i.e. match key1 and key2 while key3 can have any value. In this case, you are requesting that all records which match the partial key are updated. If you specify a unique key, you will perform an update of a single record.

• This function can only be executed after a session has been opened.

Tip

By creating the proper validation rules in the Repository, you can use LANSA to perform referential integrity checks for you. For example, the Repository's rules could specify that an order's details can only be changed if the order flag indicates that the order hasn't already been shipped. The relevant error message can also be stored in the Repository and displayed using LANSA Open's automated error message facility.

Related Functions

6.71 LceUseFile
6.58 LceSetFieldValue
6.59 LceSetFieldValueU
6.43 LceInsert
6.10 LceDelete
6.8 LceCheckFor

6.68.1 VB Example

```
Private Sub cmdSave_Click()
  ' saves department description changes
  ' no need to set key DEPTMENT as it is already set
  If txtDesc <> "" Then
     iRet = LceSetFieldValue(iSession, "DEPTDESC", txtDesc)
  Else
     "" is not allowed, sNull same as chr(0)
     iRet = LceSetFieldValue(iSession, "DEPTDESC", sNull)
  End If
  iRet = LceUpdate(iSession, "DEPTDESC", "DEPTAB", _____
      "DEPTMENT", LceFalse)
  If iRet = LceTrue Then
    ' set flag to cause refresh in all combos displaying
    ' departments
     bDepFileChanged = True
     Call RefreshDepList
  End If
End Sub
```

6.68.2 C Example

Update a record from a file. BOOL fOK; int iSession;

fOK = LceSetFieldValue(iSession, "SALARY", "25000");

fOK = LceSetFieldValue(iSession, "SURNAME", "Buckley");

fOK = LceSetFieldValue(iSession, "EMPNO", "12345");

fOK = LceUpdate(iSession, "SURNAME, SALARY", "PSLMST", "EMPNO", F.

6.68.3 Psuedocode Example

```
Check if the record can be updated.
strList = "SURNAME, SALARY"
strName = "PSLMST"
strKey = "EMPNO"
IF NOT LceSetFieldValue(iSession, strKey, "12345")
     /* return error */
ENDIF
IF NOT LceSetFieldValue(iSession, "SALARY", "25000")
     /* return error */
ENDIF
IF NOT LceSetFieldValue(iSession, "SURNAME", "Buckley")
     /* return error */
ENDIF
IF LceUpdate(iSession, strList, strName, strKey, TRUE)
     /* update of the record can be performed */
ELSE
     /* cannot update the record */
     /* display error messages */
```

ENDIF

6.69 LceUseExceptionalArguments

Instructs the session to use the exceptional arguments when opening a session. This function is optional and is only valid when used against LANSA hosts which support X_RUN exceptional arguments.



Parameters

iSession The session identifier as returned by LceGetSessionId.

strValue The list of exceptional arguments. The length of this parameter should not exceed EXCEPTIONAL_ARGS_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

Not all arguments are allowed in this argument list. Some arguments are set by calls to other LANSA Open functions (For example, the partition is set by calling LceUsePartition) and others are not allowed due to the context in which the LANSA request processor runs.

The following X_RUN arguments are supported in this function:

DBID=, DBII=, DBIT=, DBUS=, DBUT=, FXQF=, FXQM=, HSKC=, INIT=, ITHP=, ITRC=, ITRL=, ITRM=, ITRO=, ODBI=, PPTH=, PRTR=, PSWD=, RPTH=, TERM=, TPTH=, WPDF=, WPDS=, WPEN=, WPFD=, WPFO=, WPPD=.

Arguments not supported will be ignored by the host.

Names assigned to DBID and DBII must be enclosed in double quotes (") if the name contains spaces. For example (in C):

```
LceUseExceptionalArguments( iSession, "DBID=\"LX
Trunk\" DBUS=DBA PSWD=SQL" );
```

Related Functions

6.13 LceEndSession
6.46 LceOpenSession
6.72 LceUseFunction
6.73 LceUseLanguage
6.74 LceUsePartition
6.75 LceUsePassword
6.76 LceUsePriority
6.77 LceUseProcess
6.78 LceUseSystem

6.80 LceUseUserId

6.69.1 VB Example

iRet = (iSession, "DBUS=DBA PSWD=SQL")

6.69.2 C Example

BOOL fOK; int iSession; char szUser[11]; char szPassword[11];

iSession = LceGetSessionId(); fOK = LceUseSystem(iSession, "SYSTEM1");

GetUserInformation(szUser, szPassword); fOK = LceUseUserId(iSession, szUser); fOK = LceUsePassword(iSession, szPassword);

fOK = LceUsePartition(iSession, "DEM"); fOK = LceUseLanguage(iSession, "FRA"); fOK = LceUseProcess(iSession, "ABCDE"); fOK = LceUseExceptionalArguments(iSession, "DBUS=DBA PASWD=SQL");

••

fOK = LceOpenSession(iSession);

6.70 LceUseField

This function is used to declare which fields from the Repository (on the host) are to be used in the PC application. The LceUseField function indicates that the Repository field information will be required.

BOOL	LceUseField	(int	iSession,
		char far *	strFieldList)

Parameters

- iSession The session identifier as returned by LceGetSessionId.
- strFieldList The list of fields to be used. All fields in this list must be defined in the LANSA Repository. If the field is not defined, an error will result. The length of each field should not exceed FIELD_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

Any data dictionary field which is used in another function (e.g. LceFetch) must have been declared using this LceUseField function.

- This function must be executed before LceOpenSession function has been issued. If you need to execute this function after a session has been opened, you must end the session or open a new session.
- This function performs no communications to the host.

Related Functions

- 6.21 LceGetFieldDesc
- 6.23 LceGetFieldHeading
- 6.25 LceGetFieldIndicator
- 6.26 LceGetFieldLabel
- 6.27 LceGetFieldType
- 6.29 LceGetFieldValueX
- 6.30 LceGetFieldValueU

6.30 LceGetFieldValueU6.58 LceSetFieldValue6.59 LceSetFieldValueU6.46 LceOpenSession6.13 LceEndSession

6.70.1 VB Example

iRet = LceUseField(iSession, "SKILCODE,SKILDESC")

6.70.2 C Examples

C Example 1

Define fields using a list. BOOL fOK; int iSession;

fOK = LceUseField(iSession, "EMPNO, GIVENAME, SURNAME, ADDRESS1

C Example 2

Define fields individually.

fOK = LceUseField(iSession, "EMPNO");

fOK = LceUseField(iSession, "GIVENAME");

fOK = LceUseField(iSession, "SURNAME");

fOK = LceUseField(iSession, "ADDRESS1");

6.70.3 Psuedocode Example

6.71 LceUseFile

This function is used to declare which files on the host are going to be used in the application.



Parameters

iession The session identifier as returned by LceGetSessionId.

strFileList The list of files to be used. All files in this list must be defined in the LANSA Repository.

The length of each file should not exceed FILE_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

Portability	The references to LIBRARY and multi-member files in	
Considerations	the following description are applicable only when your host is an IBM i.	

The absolute specification for the file is

LIBRARY/FILENAME.MEMBER.BASEMEMBER

The library name on the IBM i host and the member name within a physical file are optional. In general, you should not be constructing applications which require these parameters.

The default values for library and member are:

LIBRARY *LIBL

MEMBER *FIRST

BASEMEMBER *FIRST

Any file which is used in another function (e.g. LceFetch) must have been

declared in an LceUseFile function.

If the file has not been defined to the LANSA Repository, an error will result. If you are using host files which were not created by LANSA, they must be loaded into the LANSA Repository on the host before you use this function. Building the LANSA Repository - Files in the *Introduction to LANSA for iSeries* Guide explains how to load existing IBM i files to LANSA.

- This function must be executed before LceOpenSession function has been issued. If you need to execute this function after a session has been opened, you must end the session or open a new session.
- This function performs no communications to the host.

Related Functions

6.32 LceGetFileDesc6.33 LceGetFileType6.46 LceOpenSession6.13 LceEndSession

6.71.1 VB Example

iRet = LceUseFile(iSession, "SKLTAB")

6.71.2 C Example

BOOL fOK; int iSession;

fOK = LceUseFile(iSession, "PSLMST, DEPTAB"); fOK = LceUseFile(iSession, "MYLIB/PSLMST");

6.71.3 Psuedocode Example

6.72 LceUseFunction

Instructs the session to use the function name specified for transaction stamping.

BOOL	LceUseFunction	(int	iSession,
		char far *	strValue)

Parameters

iSession The session identifier as returned by LceGetSessionId.

strValue The function's name. The length of this name should not exceed FUNCTION_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

When a record in a file is changed, LANSA can maintain a "stamped" audit trail. A number of attributes, including date and time, can be automatically "stamped" in specified fields in the file.

If the field stamping attribute has been set to "FUNC" (i.e. insert function name) in the Repository, the value in strValue will be used when stamping the function. If this function is not used, the default value "LCOE" is inserted.

For more information about LANSA's field stamping attributes, refer to the section: Input Options when Creating a New Field in the LANSA for iSeries User Guide.

- This function should be executed after the LceGetSessionId and before the LceOpenSession.
- This function performs no communications to the host.

Related Functions

6.77 LceUseProcess

6.72.1 VB Example

iRet = LceUseProcess (Isession, "STAMPP")
iRet = LceUseFunction(iSession, "STAMPF")

6.72.2 C Example

BOOL fOK; int iSession;

fOK = LceUseFunction(iSession, "ABCD");

6.73 LceUseLanguage

Instructs the session to use the specified language. This function is not used in a single language environment.



Parameters

iSession The session identifier as returned by LceGetSessionId.

strValue The language code as specified in the partition definition. If left blank, the partition default language (i.e. the national language) is assumed. strValue's length should not exceed LANGUAGE_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

LANSA uses partitions to separate different environments or application work areas. If a partition is defined as multilingual, you will use this function to specify the language to be used in that partition for this session. (For more information regarding multilingual processing, refer to the Partition Multilingual Attributes in the LANSA for iSeries User Guide.) Once a language is specified, details such as field descriptions, labels, etc. will appear in the language specified. File data is not affected.

Multilingual installations:

The actual language details must be recorded in the LANSA Repository on the host. If such details have not been entered for the fields or files being accessed, the equivalent details may instead be returned in the default language.

Single (national) language installations:

If the partition is single language and you execute this function, but do not leave the language code blank, unpredictable results will occur during the call to LceOpenSession.

• This function must be executed after LceGetSessionId and before

LceOpenSession.

• This function performs no communications to the host.

Related Functions

6.74 LceUsePartition

6.73.1 VB Example

iRet = LceUseLanguage(iSession, "ENG")

6.73.2 C Example

BOOL fOK; int iSession;

iSession = LceGetSessionId(); fOK = LceUsePartition(iSession, "DEM"); fOK = LceUseLanguage(iSession, "FRA");

fOK = LceOpenSession(iSession);
6.74 LceUsePartition

Instructs the session to use the specified partition. This function is optional, however, as most of your applications will require a specific partition, this function will generally be used. The default value is "SYS" or the system partition in LANSA.

BOOL	LceUsePartition	(int	iSession,
		char far *	strValue)

Parameters

iSession The session identifier as returned by LceGetSessionId.

strValue The partition name. The length of this parameter should not exceed PARTITION_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

LANSA uses partitions to separate different environments or application work areas. It is similar in concept to a directory on a PC and has a number of characteristics associated with it. The LceUsePartition function is used to specify the partition to be used.

A partition cannot be changed once the session is open. The session must be closed or a new session opened if you wish to use another partition. If you wish to use more than one partition, you must start more than one session. You will need to get another session ID.

- This function must be executed after LceGetSessionId and before LceOpenSession.
- This function performs no communications to the host.

Related Functions

6.13 LceEndSession6.46 LceOpenSession6.73 LceUseLanguage

6.74.1 VB Example

iRet = LceUsePartition(iSession, "DEM")

6.74.2 C Examples

C Example 1

BOOL fOK; int iSession;

iSession = LceGetSessionId(); fOK = LceUsePartition(iSession, "DEM"); fOK = LceUseLanguage(iSession, "FRA");

fOK = LceOpenSession(iSession);

C Example 2

Use two partitions. BOOL fOK; int iSession1; int iSession2;

iSession1 = LceGetSessionId(); fOK = LceUsePartition(iSession1, "DEM"); fOK = LceOpenSession(iSession1);

iSession2 = LceGetSessionId(); fOK = LceUsePartition(iSession2, "TRN"); fOK = LceOpenSession(iSession2);

6.75 LceUsePassword

Instructs the session to use the specified password. This function is used in conjunction with LceUseUserId.

BOOL	LceUsePassword	(int	iSession,
		char far *	strValue)

Parameters

iSession The session identifier as returned by LceGetSessionId.

strValue The password. The length should not exceed PASSWORD_LONG_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

This function is used to specify the host user password relating to the User ID specified using LceUseUserId.

If you are using a Windows version of LANSA Open with Client Access/400, this function is optional. The default value is the user profile and password used when the communication to the host was started by Client Access/400. Whilst Client Access/400 has the capability for LANSA Open to provide a password, other routers may not permit this, in which case calls to this function are ignored.

Be careful when using this function. Passwords on the host can be changed, so you should not use passwords in your applications unless they are entered by the user or are encrypted in a file which can be changed.

Portability Considerations An IBM i requires passwords to be uppercase, therefore the supplied password is converted to uppercase when the Host Type is specified or defaulted to EBCDIC. Windows, Linux or other non-IBM i platforms may have case sensitive passwords using upper and/or lower case characters. No conversion of the password is performed if the Host Type is specified as ASCII.

- This function must be executed after the LceGetSessionId and before the LceOpenSession.
- This function performs no communications to the host.

Related Functions

6.80 LceUseUserId 6.78 LceUseSystem

6.75.1 VB Example

iRet = LceUseUserId(iSession, sUserId)
iRet = LceUsePassword (iSession, sPassword)

6.75.2 C Example

BOOL fOK; int iSession; char szBuffer[PASSWORD_LONG_LENGTH + 1];

/* Call dialog to get password */
GetPassword (szBuffer);
iSession = LceGetSessionId();
fOK = LceUseUserId(iSession, "QPGMR");
fOK = LceUsePassword(iSession, szBuffer);
...
fOK = LceOpenSession(iSession);

6.76 LceUsePriority

Instructs the session to use the specified job priority. This functions sets the execution priority of the session and any jobs related to the session on the host system.



Parameters

iSession The session identifier as returned by LceGetSessionId.

iValue The priority number. The default value is 20.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

Portability Considerations

You should be aware of the implications of using this option. If you specify an unreasonable priority, you may severely impact the host system by monopolizing all of the system resources. On the host, you should never use a priority less than 10.

- This function should be executed after the LceGetSessionId and must be specified before the LceOpenSession is executed. The priority cannot be changed once the session is open.
- This function performs no communications to the host.
- This function is ignored on Linux hosts.

Related Functions

6.78 LceUseSystem

6.76.1 VB Example

iRet = LceUsePriority(iSession, 25)

6.76.2 C Example

BOOL fOK; int iSession;

iSession = LceGetSessionId(); fOK = LceUsePriority(iSession, 25); fOK = LceOpenSession(iSession);

6.77 LceUseProcess

Instructs the session to use the process name specified in strValue for transaction stamping.



Parameters

iSession The session identifier as returned by LceGetSessionId.

strValue The process name to be inserted with. The length should not exceed PROCESS_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

When a record in a file is changed, LANSA can maintain a "stamped" audit trail. A number of attributes, including date and time, can be automatically "stamped" in specified fields in the file.

If the field stamping attribute has been set to "PROC" (i.e. insert process name) in the Repository, the value in strValue will be used when stamping the function. If this function is not used, the default value "LANSA" is inserted.

For more information about LANSA's field stamping attributes, refer to the section: Input Options when Creating a New Field in the LANSA for iSeries User Guide.

- This function must execute after LceGetSessionId and before LceOpenSession.
- This function performs no communications to the host.

Related Functions

6.72 LceUseFunction

6.77.1 VB Example

iRet = LceUseProcess (Isession, "STAMPP")
iRet = LceUseFunction(iSession, "STAMPF")

6.77.2 C Example

BOOL fOK; int iSession;

fOK = LceUseProcess(iSession, "ABCDE");

6.78 LceUseSystem

Instructs the session which host system to use. This function is not necessary if you only have one host system.



Parameters

iSession The session identifier as returned by LceGetSessionId.

strValue The host system name. The length should not exceed SYSTEM_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

If you are using any Windows version of LANSA Open with Client Access/400, this function is optional. The default host system will be the one used when the communications to the host was started.

Refer to Configuration File for more details.

- This function must be executed after the LceGetSessionId and before the LceOpenSession. The host cannot be changed once the session is open.
- This function performs no communications to the host.

Related Functions

6.74 LceUsePartition6.80 LceUseUserId6.75 LceUsePassword

6.78.1 VB Example

iRet = LceUseSystem(iSession, sSystem)

6.78.2 C Example

BOOL fOK; int iSession;

iSession = LceGetSessionId(); fOK = LceUseSystem(iSession, "SYSTEM1"); fOK = LceOpenSession(iSession);

6.78.3 Psuedocode Example

6.79 LceUseTmpDrive2

Instructs the session to use the directory specified as TmpDrive2 in the PC configuration files, for the storage of local information.

BOOL LceUseTmpDrive2 (int iSession)

Parameters

iSession The session identifier as returned by LceGetSessionId.

Return Values

TRUE is returned if the session can perform the function.

FALSE is returned if an error occurs.

Notes

LANSA Open uses local (temporary) files on the PC to store information. The actual drives used are specified in the Registry file. Two temporary drives can be specified. Using this function indicates that the local information is to be stored in the second temporary drive.

Using this function out of sequence will cause your application to fail or produce unreliable results. For example, if you transfer data using an LceSelect function and then follow it with an LceUseTmpDrive2 function, your application will not find the selected data as it would have been stored on temporary drive 1. For this reason, frequent use of this function is not recommended.

- This function should immediately follow LceGetSessionId and be before LceOpenSession function. Once issued, the second temporary drive will be used for the remainder of the session.
- This function performs no communications to the host.

6.79.1 VB Example

iRet = LceUseTmpDrive2(iSession)

6.79.2 C Example

BOOL fOK; int iSession;

iSession = LceGetSessionId(); fOK = LceUseTmpDrive2(iSession); fOK = LceOpenSession(iSession);

6.80 LceUseUserId

This function specifies the host User Id to be used when accessing the host. LANSA programs and files are secured by User Id. Use this function when you want to specify a user other than the one used by default when starting some communications software to the host.



Parameters

iSession The session identifier as returned by LceGetSessionId.

strValue The User Identifier. The length should not exceed USERID_LONG_LENGTH as defined in the LCOE.H header file.

Return Values

TRUE is returned if the session can accept the value.

FALSE is returned if an error occurs.

Notes

If you use this function, you must also use the LceUsePassword function.

If you are using any Windows version of LANSA Open with Client Access/400, this function is optional. The default value is the user profile and password used when the communications to the host was started. Whilst Client Access/400 has the capability for LANSA Open to provide a User Id, other routers may not permit this, in which case calls to this function are ignored.

As User IDs can easily be changed on the host, have the User Id entered by the user or store them in a file which can easily be changed.

• This function must be executed after LceGetSessionId and before LceOpenSession.

Related Functions

6.75 LceUsePassword6.78 LceUseSystem

6.80.1 VB Example

iRet = LceUseUserId(iSession, sUserId)
iRet = LceUsePassword (iSession, Password)

6.80.2 C Example

BOOL fOK; int iSession; char szBuffer[PASSWORD_LONG_LENGTH + 1];

/* Call dialog to get password */
GetPasword (szBuffer);
iSession = LceGetSessionId();
fOK = LceUseUserId(iSession, "QPGMR");
fOK = LceUsePassword(iSession, szBuffer);
...

fOK = LceOpenSession(iSession);

6.81 LceVersion

This function returns the current LANSA Open version number and displays it in a dialog box on the screen.



Parameters

fDisplay A boolean to indicate whether to display the dialog box.

hwnd A valid window handle, usually the main window for the application.

If an invalid window handle is specified, the dialog box will not be displayed.

NULL indicates that the current active window is to be used.

Return Values

The return value is a long integer. The major version number is in the High Word (2 bytes). The minor version number is in the Low Word (2 bytes).

Notes

• This function can be executed at any time.

6.81.1 VB Example

' get Version dim l as long, dVer as double ' get version l = LceVersion(False, 0) ' extract major / minor version and set in m.nn format dVer = l \ &H10000 + ((l Mod &H10000) \ 100) / 100 lbLansaVer.caption = str(dVer)

6.81.2 C Example

Retrieve version information and do not display.

long lVersion; int iMajorRel; int iMinorRel;

lVersion = LceVersion(FALSE, NULL); iMajorRel = HIWORD(lVersion); iMinorRel = LOWORD(lVersion);

6.81.3 Psuedocode Example

Display version number in current active window. IF NOT LceVersion(TRUE, NULL) /* return error */ ENDIF

7. Troubleshooting

This chapter describes some of the communication errors you could encounter, how to discover the cause and how to fix the error.

Error Message Codes contains a list of error message codes and their meanings.

7.1 What to Do if You Have Communications Problems

7.1 What to Do if You Have Communications Problems

Communication problems can be caused by a number of factors, from the communications router to the application on the PC. While the actual configuration and maintenance of communications between client workstations and IBM i servers is beyond the scope of this guide, the information in this chapter may help you to determine the cause of your problem.

Start with these basic checks:

Can you get a 5250 terminal emulation session running under the communications router in order to sign on to the IBM i?

If you can't, then work with your communications router and its configuration options until you can. If you can't run a terminal emulation session then there is no way that you are going to get a session using LANSA Open to start and run.

- Check that the user profile you are using:
 - Is defined on the IBM i.
 - Is 8 characters or less in length.
 - Has the password that you are using.

Check that the user profile you are using is validly defined and is authorized to use the LANSA system and partition.

Do this simple test:

- Sign on at a 5250 terminal and immediately type the command LANSA PARTITION(ppp). Can the LANSA command be found? If not, you have a basic library list problem. The job description associated with the user profile does not include the LANSA program library (usually DC@PGMLIB) in its initial library list.
- Use the DSPUSRPRF command to view the Job Description associated with the user profile and note down the associated job description. Next use the DSPJOBD command to view the job description itself. Ensure it has an initial library list which includes the **LANSA program library** (usually DC@PGMLIB) and the LANSA Communications Library.
- Also check that the Job Description has LOG(4 00 *SECLVL) logging, if you are trying to solve a problem. This will ensure any OS/400 job that the user profile starts will produce an OS/400 job log. This job log will **almost**

always yield useful error information. If you request help from your LANSA vendor, always send example(s) of the job logs resulting from failed connection attempts.

Check that the partition and language code that you are using are valid.

- Make sure that the partition and language you have specified are valid.
- The partition you specify must exist in the target LANSA system on the IBM i.
- If the partition you specify is not multilingual then the language you specify must be NAT (National Language).
- If the partition you specify is multilingual then the language you specify must be a valid language defined within the specified partition.

Check that your LANSA system is correctly licensed.

Sign on to LANSA for iSeries as QSECOFR, or the LANSA system owner, and use the command LANSA REQUEST(LICENSE) to display your current licensing status.

• To use LANSA Open you need a License Type of LCE.

Check the communications subsystem on the IBM i is active.

- Make sure the subsystem QCMN is active.
- If it isn't, start it and try the connect operation again. To run communication sessions, QCMN must be active.

Check the communications router is active and correctly started on your workstation.

- Make sure that the User Id and Password have been specified.
- Make sure that the communications router you are using is started.
- Check the IBM i LU name you are specifying. Sign on to your IBM i and use the DSPNETA (Display Network Attributes) command to determine your local control point name (for example, SYDASDEV).

Check for a communication job on the IBM i

Does an OS/400 "workstation" job appear in the QCMN subsystem?

• Use the OS/400 command WRKSBSJOB QCMN, to display the active jobs in subsystem QCMN. Repeatedly use F5=Refresh, to refresh the list, as you start the communications connection from the application executing on the workstation.

Normally a job in IBM i subsystem QCMN has the same name as the associated PC workstation.

- If a job appears in the subsystem, immediately put a "5" (Work with) against it and then choose the option to display its spooled files. When it completes it should produce an IBM i job log (spool file QPJOBLOG). If it does not produce a job log then alter the job description associated with it (described previously) and attempt the operation again. Review the resulting job log backwards. Errors and escape message at the end of the job log will usually show the cause of the problem, whereas errors and escape message at the start of the job log may mislead you because they are often trapped and handled. (For example, "Program LCXP9000 not found" is not an error, it is simply an attempt to find a user exit program).
- If a job does not appear in the subsystem, use the DSPLOG command to display the IBM i system history log. Move to the end of the log and look for messages that relate to the PC workstation involved. Usually such messages will indicate the cause of (and often the solution to) basic IBM i communication configuration problems.
- If no job appears in the subsystem and there are no messages in the IBM i system history log, then you would appear to have such a fundamental connection problem between your PC and IBM i that the IBM i does not even become aware of the connection request. Proceed to review the configuration details of your communication router and use its error logging and tracing capabilities.

Check that the file or library you are trying to open exists

• Most of the error messages found in job logs are self explanatory. Trying to open a session that uses a file that does not exist on the IBM i would cause an Lce0023 return code 0x01 to be received by LANSA Open. In this case, the job log of such a job could look like this:

CPF4101 Message: File <file name> in library *LIBL not found or

inline data file missing. Cause : The file was not opened. The reason code is 03.

Possible reason codes and their meanings:

- 01 The library does not exist.
- 02 The file does not exist but the library does exist.
- 03 The file does not exist.

The reason for that failure can be clearly identified, however, there are other errors when the cause is not so straightforward. For example:

• MCH0601 Message: Space offset &2 is outside current limit for object &1.

This could be caused by:

- The space declared in data area DC@A10 not being big enough to hold the records selected. Increase the size of the value in DC@A10.
- MCH0802 Message: Total parameters passed does not match number required.

This could be caused by:

• Using the *BLOCKBYRRNnnn or the *RECEIVEIMMED select option, but with *IOMXSERVER, *IOMBLOCKBYKEY and *IOMBLOCKBYRRN missing from data area DC@OSVEROP's I/O module.

Add these keywords into data area DC@OSVEROP and recreate the I/O module.

• Other objects in the library list with same name as the I/O module.

• MCH3601 Message: Pointer not set for location referenced

This could be caused by:

• An RCLRSC command being issued directly from the application, or position 480 in data area DC@A01 being set to "Y".

Either don't use RCLRSC or change position 480 of the DC@01 data area to a value other than "Y".

General Protection Fault (GPF)

A GPF usually occurs when an application tries to access a part of the memory that is already allocated. It can also occur when running out of stack space.

If you request help from your LANSA vendor, be sure to include the job log with the problem's description.

Appendix A. Error Message Codes

These error codes are included:

- LANSA Open Error Codes
- Error Code 3 Internal Error Reason Codes
- Error Code 23 CPI-C Return Codes

LANSA Open Error Codes

Almost every LANSA Open function returns a True or False value. If a False value is returned, it will be defined by the error codes in this list.

When a function returns FALSE, LceGetStatus can be used to determine the code.

The object in error is included in the error message where applicable. This is represented by %d, %c or %100s in these messages.

Error Reason Code

- 0 The title: Error
- 1 The title: LANSA Open Validation Messages
- 2 The title: Cancel
- 3 Internal Error. Reason Code = %d.

Note: The Reason Codes (represented by %d) are listed in Internal Error Reason Codes.

- 4 An error was encountered while allocating memory.
- 5 Invalid parameter. The parameter cannot be a NULL pointer.
- 6 The field %100s has not been defined for the current session.
- 7 The file %100s has not been defined for the current session.
- 8 The priority %d is not valid. It must be a value in the range 5 to 80.
- 9 The heading number is not valid. It must be in the range 1 to 3.
- 10 The field %100s cannot be set to the value specified because it exceeds %d, the maximum length for the field.
- 11 No Select Options have been specified in the Options list.
- 12 The Select Option %.100s is not valid. The only valid Select Options are *STARTKEY, *BACKWARDS, *LOCK, *MAXREC.
- 13 The function has not been specified, or it exceeds the maximum length of %d.

- 14 No Fields were specified in the Field list.
- 15 The parameter %.100s is too long. It exceeds the maximum length of %d.
- 16 The combination of parameters is not valid. A program cannot be specified if a process and function are defined. If the process is defined the function must be defined also.
- 17 The temporary directory %.100s is not valid. Specify a directory that exists in the Configuration File.
- 18 The object %.100s is not defined in the current session.
- 19 The numeric %.100s is not in a valid format for the field specified. It should be in the format (%d,%d).
- 20 The Session ID %d is not valid. It must be a positive number less than the maximum number of Sessions, 20.
- 21 The Session ID %d is not open.
- 22 The Session ID %d is already open.
- A Communications Error has occurred for the action %.100s. The return code is %d.

Note: The return codes (represented by %d) are listed in and CPI-C Return Codes.

- A memory allocation error occurred while trying to initialize the library. No LANSA Open functions can be called.
- 25 The LANSA.INI file is not valid. No LANSA Open functions can be called.
- 26 The communications type is not valid. The session could not be opened.
- 27 The PC Support Router has not been started. No LANSA Open functions can be called.
- 28 Communications Manager has not been started. No LANSA Open functions can be called.
- 29 No previous LceRequestSelect was done for this file.

- 30 Error in OS400 command.
- 31 Error in called program or process/function.
- 32 Submit failed.
- 33 Commitment control not started.
- 34 Field(s) defined in LceUseField function not defined in the LANSA Repository.
- 35 File(s) defined in LceUseFile function are not known to the LANSA Repository.
- 36 Validation errors have occurred.
- 37 I/O errors have occurred.
- 38 Unknown error response code received.
- 39 The name %.100s is not a valid file name.
- 40 The library name in file %.100s is too long.
- 41 The file name in file %.100s is too long.
- 42 The member name in file %.100s is too long.
- 43 The record number %d is not valid.
- 44 Record not found.
- 45 The maximum number of concurrently Open Sessions, %d, has been reached. A new Session cannot be obtained.
- 46 The list passed contains an invalid comma or does not have one where it is necessary.
- 47 An internal file cannot be opened. This may be due to insufficient disk space on the temp drive or not enough file handles.
- 48 Cannot open the session because the system name, userid and password were not all specified.
- 49 An internal file could not be written to. Ensure that there is enough space on the drive which was nominated as the temporary drive or the current drive if one was not specified.

- 50 You are not licensed to use this product. Contact your LANSA vendor to obtain a LICENSE.
- 51 Your license is valid, however the maximum number of users are already using it.
- 52 Failed to load Dynamic Link Library Module, %.100s.
- 53 Failed to load Dynamic Link Library Procedure, %.100s.
- 54 The option or value specified is not valid.
- 55 A key list cannot be specified for LceRequestSelect or LceRequestSelectWhere when blocking options are used.
- 56 The conditions for LceRequestSelectWhere exceed the maximum length of %d.
- 57 Blocked I/O cannot be used with the *BACKWARDS option.
- 58 The Session ID %d must receive all the records via LceReceiveNextX before this function can be used.
- 59 The buffer passed to LceReceiveNextX is too small to contain the complete record. An incomplete record has been returned.
- 60 LceReceiveSelect cannot be used when the *RECEIVEIMMED option has been specified.
- 61 LceReceiveNextX cannot be called unless LceRequestSelect or LceRequestSelectWhere is called with the *RECEIVEIMMED option specified.
- 62 No more records to be returned by LceReceiveNextX.
- 63 The options *BLOCKBYRRN and *BLOCKBYKEY are not valid when specified together.
- 64 LceRequestSelect and LceRequestSelectWhere cannot be called for logical files when blocked I/O is being used.
- 65 Invalid Communications Cursor Type specified.
- 66 Invalid Communications Type specified.
- 67 Invalid Host Type specified.

- 68 The variable specified does not exist or a multilingual variable was requested in a non-multilingual partition.
- 69 Invalid Help Text Type Specified.
- 70 End Of Line string specified for Help Text is too long.
- 71 Error when accessing or creating a Local Data Dictionary file. Ensure that there is enough space on the drive which was nominated as the Local Data Dictionary drive or the current drive, if one was not specified.
- 72 Invalid Field list.
- 73 Host returned an error code on last operation.
- 74 Inconsistent Type, Length or Decimals was encountered for field %.100s .
- 75 Invalid 3GL parameter number combination. The first level number must be from 0 to 19. The sub-parameter must be from 0 to 255.
- 76 Invalid parameter Direction specified.
- 77 Invalid parameter Type specified.
- 78 Aggregate parameter Length for this first level number exceeds the maximum allowed OR numeric parameters length or decimals is invalid.
- 79 Parameter %d, %d has not been defined using Lce3GLDefineParameter.
- 80 Invalid attempt to set or get a parameter value which has an inappropriate direction value. Direction must be "P" or "B" to set the value, or "R" or "B" to get the value.
- 81 An error occurred executing the host 3GL program.
- 82 The sum of the length of the fields requested for use exceeds the host's maximum I/O buffer size of %d.
- 83 The operation requested has resulted in a communications transaction to the host which exceeds the hosts receive buffers maximum size of %d.

Error Code 3 - Internal Error Reason Codes

This error code is returned when the internal logic of the software detects a condition which should not occur in normal operation.

Specific cause information is not available, however, a Reason Code representing an internal macro is returned. This code may assist in determining the cause of the error.

Listed below are the Reason Codes and the associated internal macro names.

Reason Code	Associated Macro Name
1	LCE_INTERR_INVALID_DATA_TYPE
2	LCE_INTERR_DBCS_TARGET_TOO_SMALL
3	LCE_INTERR_TOO_MANY_DIGITS
4	LCE_INTERR_TOO_MANY_DECIMALS
5	LCE_INTERR_NUMERIC_TARGET_TOO_SMALL
6	LCE_INTERR_RECV_BUFF_TOO_SMALL
7	LCE_INTERR_RECV_NO_TERMINATOR
8	LCE_INTERR_RECV_INVALID_COMMAND
9	LCE_INTERR_RECV_FATAL_ERROR
10	LCE_INTERR_RECV_BAD_FORMAT
11	LCE_INTERR_RECV_INVALID_IDENTIFIER
12	LCE_INTERR_REVERSE_BUFF_TOO_SMALL
13	LCE_INTERR_COMMS_ALREADY_ALLOCATED
14	LCE_INTERR_UNKNOWN_ACTION
15	LCE_INTERR_COMMS_RECV_BAD_DATA
16	LCE_INTERR_BUFF_TOO_MANY_FILES
17	LCE_INTERR_BUFF_FILE_READ
18	LCE_INTERR_BUFF_FILE_ACCESS

LCE_INTERR_RECV_INVALID_FILE_TYPE
LCE_INTERR_DBCS_BAD_CLIENT_STRING
LCE_INTERR_DBCS_BAD_HOST_STRING
LCE_INTERR_PACKED_BUFF_TOO_SMALL
LCE_INTERR_SIGNED_BUFF_TOO_SMALL
LCE_INTERR_TABLE_BUFF_TOO_SMALL
LCE_INTERR_TP_END_SESSION
LCE_INTERR_INVALID_TABLE_INDEX
LCE_INTERR_INVALID_TABLE_RECORD
LCE_INTERR_EDIT_BAD_FORMAT
LCE_INTERR_VARIABLE_BAD_TYPE
LCE_INTERR_DEFAULT_BAD_TYPE
LCE_INTERR_UNKNOWN_RC_LCLCE
LCE_INTERR_PARBLOCK_ZERO_MEM

Error Code 23 - CPI-C Return Codes

This list includes the most common return codes issued with a communications error code (Lce0023) returned by LANSA Open when using the CPI-C communication method (i.e. MS Host Integration Server, Network Services for DOS).

Some of the return codes associated with the allocation of a conversation have the suffix RETRY or NO_RETRY in their name:

- **RETRY** means that the condition indicated by the return code may not be permanent and the program can try to allocate the conversation again. Whether or not the retry attempt succeeds depends on the duration of the condition.
- **NO_RETRY** indicates that the condition is probably permanent.

Please note that LANSA Open always returns the error codes in hexadecimal format. This documentation lists the error codes in decimal format, so you need to convert the returned code to decimal format before looking up the error in this table.

Dec Value	Description	Reason for error
6	SECURITY_NOT_VALID	Userid/password combination invalid (i.e. user profile has not been defined in the host or this is an incorrect password for the userid - note that passwords are case sensitive for Windows, Linux and non-IBM i hosts). No job could be started in the Host, so don't look for a job log.
17	DEALLOCATED_ABEND	 LANSA program library not in user's job description's initial library list.
		2. When I/O module was compiled, settings *IOMXSERVER, *IOMBLOCKBYKEY and *IOMBLOCKBYRRN were missing in DC@OSVEROP data area. (Refer to

		MCH0802 in the job log). 3. Other causes may be discovered in the IBM i Job Log or Windows Application Event Log.
18	DEALLOCATED_NORMAL	1. Partition doesn't exist or user is not authorized for the partition.
		2. File doesn't exist or the user is not authorized for the file.
		3. Language used is not defined in the partition.
		4. Other causes may be discovered in the IBM i Job Log or Windows Application Event Log.
19	PARAMETER_ ERROR	On the Host Integration Server, use SNA Server Admin. to add a LU6.2 partner relationship between the "Local LU" and "Partner LU" which has a MODE of "BLANK".
20	PRODUCT_ SPECIFIC_ERROR	As for Error 19 (PARAMETER_ERROR).
24	PROGRAM_ PARAMETER- CHECK	Most likely cause is that the userid or password is more than 8 characters. SNA requires userids and passwords to be a maximum of 8 characters.
26	RESOURCE_ FAILURE_NO_ RETRY	Most likely cause is hardware failure. Check the connections (cabling) between PC and host.
27	RESOURCE_ FAILURE_RETRY	PC has run out of memory. Confirm by opening a session with no declarations. Reduce the number of applications loaded when the system starts.