## **What It Is**

A way of handling non fatal errors that are technically errors SQL reports.

These include:

* Data validation/format errors e.g. incorrect phone number format
* No rows found/written
* Other procedural errors by the user

As well as any other message that needs to be sent to the user, also allows for more “human” error messages to be returned.

## **How It Works**

The custom error messages are stored in a folder called: Resources, in a file called: ErrorList.res.

This file has a simple format of:

Custom error number, custom error text.

E.g.:

*1000,Login Not Found*

*1001,User Not Found Or Password Incorrect*

*1003,Error Creating LOT Record!*

*1004,Error Reading User Password*

**Every** SQL Stored procedure that is called by the system has a special variable added to it for returning custom error values: @ErrorCustom int OUTPUT.

@ErrorCustom is set to a value in the ErrorList.re file.

The C# program first loads the ErrorList.res file into a *dictionary* using the error number as the key:

In the clsData modules is defined the dictionary:

*static Dictionary<int, string> dicSQLErrors = new Dictionary<int, string>();*

The function to load the the ErrorList.res file into the dictionary is also in the module:

*public static bool InitCustomErrorhandler(string strPath)*

*{*

*/\**

*Created 02/07/2025 By Roger Williams*

*Inits custom SQL error resource file path variabel*

*Then loads it into dictionary: dicSQLErrors*

*Checks if passed path is null or file does not exist*

*VAR*

*strpath - location of resource file*

*RETURNS*

*true if ok*

*\*/*

*StreamReader strmTemp = null;*

*string strTemp = "";*

*string strError = "";*

*string strMsg = "";*

*if (strPath.Length == 0 || !File.Exists(strPath))*

*{*

*return false;*

*}*

*CNST\_STR\_ERRORFILEPATH = strPath;*

*strmTemp = new StreamReader(CNST\_STR\_ERRORFILEPATH);*

*while (!strmTemp.EndOfStream)*

*{*

*strTemp = strmTemp.ReadLine();*

*//split into error number and error message*

*//purposely add,*

*strError = strTemp.Substring(0, strTemp.IndexOf(",") + 1);*

*strMsg = strTemp.Remove(0, strError.Length);*

*//remove it*

*strError = strError.Remove(strError.Length - 1);*

*dicSQLErrors.Add(Convert.ToInt32(strError), strMsg);*

*}*

*strmTemp.Close();*

*strmTemp.Dispose();*

*return true;*

*}*

In the Load event of the main form this function is called:

*//init custom sql error data*

*if (! Modules.clsData\_Utilities.InitCustomErrorhandler(Path.GetDirectoryName(Application.ExecutablePath) + @"\Resources\Errorlist.res"))*

*{*

*this.DialogResult = System.Windows.Forms.DialogResult.Cancel;*

*this.Close();*

*}*

*}*

Note: Due to not knowing *exactly* where the Exe is, uses Application.ExecutablePath to ascertain *exactly* where to find the Resource folder.

To display messages to the user all that is needed is to parse the error number returned by the SQL stored procedure with the dictionary:

*SQLConn.Open();*

*SQLCmd = SQLConn.CreateCommand();*

*SQLCmd.CommandText = "SP\_GetPassword";*

*SQLCmd.CommandType = CommandType.StoredProcedure;*

*SQLCmd.Parameters.Add("@User", SqlDbType.VarChar, 30).Value = strUser;*

*SQLCmd.Parameters.Add("@Password", SqlDbType.VarChar, 10).Direction = ParameterDirection.Output;*

*SQLCmd.Parameters.Add("@ErrorCustom", SqlDbType.Int).Direction = ParameterDirection.Output;*

*SQLCmd.ExecuteNonQuery();*

*if (Convert.ToInt32(SQLCmd.Parameters["@ErrorCustom"].Value) == 0)*

*{*

*return (SQLCmd.Parameters["@Password"].Value.ToString());*

*}*

*else*

*{*

*//show error to user*

***dicSQLErrors.TryGetValue(Convert.ToInt32(SQLCmd.Parameters["@ErrorCustom"].Value), out strTemp);***

***strTemp = SQLCmd.Parameters["@ErrorCustom"].Value.ToString() + "\n\n" + strTemp;***

***MessageBox.Show("Error: \n\n" + strTemp);***