Credential Manager

Table of contents

Overview

Credential Manager is a Delphi component that wraps the Value List Editor component to simplify the following:

- The management of a library of credentials using an ordinary .ini file.
- The selection of a desired set of credentials that may then be employed by an application.
- A method that merges the selected credentials into another TStrings object that may then be used for whatever purposes deemed desirable.

Although the component was originally intended for use with dataase credentials, in reality it is nothing more than a TStrings manager. Therefore, it may be used for any set of TStrings as required by an application.

Requirements

This project was developed using RAD Studio 10.1 Berlin Update 2. It will probably work with other recent versions of RAD Studio but this may require some adjustments in the project parameters to achieve.

Credential Manager is a VCL component that works for both 32-bit and 64-bit Windows applications. Other platforms are not supported.

Installation

Complete the following steps.

- Download the source code. You can either download and unzip the code or use any of a number of tools to clone the repository. Github Repository: credmgr
- Launch RAD Studio and open the CredentialManager Project Group
- Use the Build Groups pane of the Project Manager to select the Installation Active Group. Make sure all project boxes are checked and then Build the Group.
- Install the design time package.
- Create a test .ini file in the appropriate directory
- Use the example code to test the component and installation
- For general use, make sure the two .dfm files are placed in a suitable directory that can be used by other Projects and Project Groups.

Utilization

Using the component consists of the following:

- Drop the TGetCredentials component on a form or datamodule.
- Configure the TGetCredentials properties that determine the location and name of the .ini file.
- Create the .ini file in the correct location.
- Populate the .ini file with a section named *model (with the asterisk) that will be used to create new entries in the library.
- Program the TGetCredentials component methods as needed to invoke the Credential Manager and utilize the credential set returned.

Installation

The source code and other library components for TGetCredentials can be obtained from GitHub.

Credential Manager at GitHub

Downloading the Code

You can download the project by using either of the following methods:

- Download the .zip file that contains the project

- Copy the project URL and use any of the available GIT tools to clone the project. (GitHub provides a desktop utility that may be used. Other tools such as Atlassian's Source Tree may also be employed.)

Delphi Libraries

For the most part, normal Delphi library setup is appropriate. Some differences should be noted. If required, change the library specifications to suit your own requirements. Following are some things to keep in mind:

Project Options

The project options for CredMgr240.bpl specify a unit output directory of $(BDSCOMMONDIR)\DCU\$(Platform) for All configurations - All platforms. This is largely a matter of convenience because in most cases the .dcu files will be required for successful utilization. This makes them easier to find.

Manual File Copy

The component was developed using a directory named $(BDSCOMMONDIR)\DFMFMX. After the component was developed and debugged, the .dfm files were manually copied to this directory. This again is a matter of convenience, and groups all such files (from this and other components) in a single directory that is easy to access.

For more information about this see Runtime Packages.

Global Options

Choose Tools==>Options from the main menu bar. Then select Delphi Options==>Library. This will display the global directory panel. Choose 32-bit Windows or 64-Windows from the Selected Platform dropdown. (Both must be configured.)

32-bit Windows

The Package output directory is customarily $(BDSCOMMONDIR)\Bpl.
The DCP output direcory is customarily $(BDSCOMMONDIR)\Dcp.
The Library path should include both of these directories.

Additionally, if an application that uses the component is compiled without using Runtime Packages, the Library path should include $(BDSCOMMONDIR)\DCU\$(Platform) and $(BDSCOMMONDIR)\DFMFMX. For more information about this see Runtime Packages.

64-bit Windows

The Package output directory is customarily $(BDSCOMMONDIR)\Bpl\$(Platform).
The DCP output direcory is customarily $(BDSCOMMONDIR)\Dcp\$(Platform).
The Library path should include both of these directories.

Additionally, if an application that uses the component is compiled without using Runtime Packages, the Library path should include $(BDSCOMMONDIR)\DCU\$(Platform) and $(BDSCOMMONDIR)\DFMFMX. For more information about this see Runtime Packages.

Compilation and Installaltion

Compilation and Installation is most easily accomplished by using the Build Groups pane of the Project manager. Open the Build Groups pane, choose Installation from the Active Group dropdown if it is not already showing, and then click the Build button.

All Projects should be selected. Order of compilation is important and the order in the Build Group should be preserved to ensure successful compilation.

After the Installation Build Group has been successfully compiled, the component should be installed by right-clicking the CredentialManager project, and choosing Install.

Changing the Tool Palette Name

The component is installed in the Tool Pallett group named "VyDev Utility." If it is desired to install the component in a different group, this can be achieved by editing the pCredentialManager.pas file *prior to building and installing the component.*

Open the pCredentialManager file and change the Tool Palette Page string from "VyDev Utility" to the desired name.

Using the Component

As with most Delphi components, utilization consists of the following:

- Drop the TGetCredentials component on a TForm or TDataModule.
- Configure the required properties. See Property Settings.
- Manually create the required .ini file. See The .ini File.
- Use the desired component methods in your application code. See Methods.

Property Settings

TGetCredentials has four unique properties, three of which are required. The values may be specified at design time or programmatically at run time. If an attempt to retrieve credentials is made at run time and one or more of the required values has not been provided, a message is displayed and the request will fail. (mrCancel will be returned from the dialog.)

For details about the .ini File location and full path name see .ini File location.

For details about the minimum required content for the .ini file see .ini File Minimum Contents.

Application

This is the application name. A single application might have more than one executable program. All programs in the same application should share the same application name.

The value entered must conform to the naming standards for file names.

CredentialSet

Credential set is the default name of the desired credential set. This corresponds to a section name in the .ini file. When the GetParameters method is invoked, the dialog will position itself to this name. If the name is not present, no error is thrown; the dialog simply presents a blank Value List Editor and the desired set can be selected from the dropdown in the customary manner.

The value of this property is modified each time the user elects to use the currently displayed credential set. Thus the current value of the property always reflects the most recently used credential set.

Note that if the user elects to cancel the credential set dialog, no change is made to this property and it remains as it was specified when the GetParameters method was invoked.

IniFileName

This is the .ini file name. A single application might have more than one distinct .ini file. All .ini files in the same application should share the same application name.

The value entered must conform to the naming standards for file names. Although not required, it is recommended that the file type .ini be used, i.e, MyParams.ini.

Vendor

This is the vendor name. A single vendor might have more than one application. All applications from the same vendor should share the same vendor name.

The value entered must conform to the naming standards for file names.

Methods

Only two methods are provided by Credential Manager.

GetParameters invokes the Value List Editor modal dialog that permits the user to manipulate and select a desired credential set. The dialog returns a TModalResult value indicating the user's selection.

MergeInto provides a convenient way to merge the values of the returned parameters into an existing TStrings instance. For example, the parameter list of a TFDConnection is a TStrings instance. MergeInto permits the easy transfer of Credential Manager returned values into the TFDConnection component's parameters.

GetParameters

```
function GetParameters: TModalResult;
```

Description

Displays the Value List Editor modal dialog that allows the user to manipulate credential sets, and either select a set for use or cancel the selection.

Return Values

Value	Meaning
`mrOK`	A credential set was selected. It has been returned in the Credentials property. The name of the selected credential set has been returned in the CredentialSet property.
`mrCancel`	The user has cancelled the request. No changes have been made to the Credentials property or the CredentialSet property.

MergeInto

```
procedure MergeInto(const AParamList: TStrings);
```

Description

The current values in the Credentials property of Credential Manager are merged into the AParamList parameter.

Name in AParamList matches Name in Credentials Property	Action Taken
Names Match	Value of corresponding Credentials item replaces Value of corresponding AParamList item.
Name in Credentials Property not found in AParamList	Name/Value pair from Credentials property is added to AParamList.
Name in AParamList not found in Credentials Property.	No action is taken.

The .ini File

The component uses a standard .ini file that can be edited using any standard text editor. The file has a number of sections, corresponding to the dialog's "Credential Set Name." Each section has a number of properties that are displayed in the Value List Editor when the set has been selected. The properties are of the customary form "Name=Value." All values are string values.

.ini File location

The .ini file location and hence the full path name is determined by a combination of operating system convention and the three required component properties, Vendor, Application and IniFileName.

The full path name is comprised of the concatenation of:

- **TPath.GetHomePath** (For Windows Vista or later, this will probably be `C:\Users\<username>\AppData\Roaming`
- The **Vendor** property value, for example `MySoftware`
- The **Application** property value, for example `CustomApp`
- The desired **IniFileName** property value (can be any valid file name) for example `DBLocs.ini`

The value of IniFileName does not necessarily require a file type of .ini, but this is recommended.

The full path and file name form a tree structure:

```
TPath.GetHomePath
|
----Vendor
    |
    ----Application
        |
        ----IniFileName
```

In this example, the complete .ini File file name would be:

```
C:\Users\<username>\AppData\Roaming\MySoftware\CustomApp\DBLocs.ini
```

.ini File Minimum Contents

The .ini file must contain at least one section that is named [*model]. This is used as the source text to populate additional credential sets when the **New** function is selected. The only way to initialize or modify the *model section is by using a text editor directly on the .ini file. Credential Manager will neither display its contents directly nor allow you to save directly to it. It never even appears in the credential set drop down.

Credential Manager creates a new credential set by copying the contents of the *model section to the newly created section. Thus, the contents of the *model section determine what name-value pairs are available in the section.

The following is an example of a *model section that might be used for Interbase connection parameters.

Example:

```
[*model]
User_Name=
Password=
RoleName=
Server=
Port=
Database=
```

Note that it is also possible to include values in the *model section. These may be thought of as default values and will appear in newly created credential sets. For example:

```
[*model]
User_Name=TestUser
```

```
Password=TestPassword
RoleName=
Server=localhost
Port=
Database=Developer
```

When a new credential set is created, the name/value pairs that have values will appear with the default values given by the *model section. They may be edited the same as any other value.

Runtime Packages

There are two general approaches to using .bpl components:

- Compile the main program with the *Link with runtime packages* option set to **True.** This will make the executable load the .bpl *at run time* and the .bpl will not be linked in with the main program. This is probably the more infrequent way most programmers will elect to use components.

  In order for this to work correctly, the necessary .bpl files must be able to be located within a path accessible to the running main program. This can be accomplished in a number of ways. They are mentioned here, but a complete discussion of each of them is beyond the scope of this document.
  - Probably the most common way is to put the .bpl files in the same directory as the program, but this tends to defeat the benefits of .bpl files.
  - Another way of achieving this is to put the .bpl files in the Common Program Files for the Publisher. This requires the determination of the proper path and the use of the Windows SHGetFolderPath and SetDLLDirectory APIs.
  - A third way would be to place the needed files in a directory that is in the PATH environment variable.
- Compile the main program with the *Link with runtime packages* option set to **False.** In this case, all executable code is linked into a single .exe file and no run time loading of .bpl files will take place.

  This method has the advantage of being simpler both in terms of development and deployment at the expense of larger executables. Note, however, that there are additional requirements for linking when using this method.

  Because the component has two units that define Delphi forms, you must make both the .dcu and the .dfm files available to the linker. This is the reason for the suggestions (mentioned in Delphi Libraries) that .dcu files from compilation and .dfm files from the original source be placed in easily accessible folders and that those two folders be a part of the global Search path.

  Remember, if you choose to set *Link with runtime poackages* to **False**, then you must ensure that the .dcu files and the .dfm files for the component are available when you compile and link your own program.

Code Example

A small test project named **ComponentTest.exe** is included in the project group. It can be used to test the installation of the component as well as seeing an example of its use. Before you can successfully execute the test program, you must first place a minimal .ini file in an appropriate location. A suitable file is included in the DATA folder that is a part of the Project Group. It can be copied to the necessary location and modified as desired. See The .ini File for additiional information.

Unless you modify the properties of the TGetCredentials1 component, the .ini file must be placed in

`C:\Users\<UserName>\AppData\Roaming\VyDevSoft`

where `<UserName>` is your logged on User Name.

- Most Windows installations will have a system drive of `C:`, but this is not necessarily true.
- Likewise, in certain very unusual circumstances, `<UserName>` will not be your User Name.

- Finally, if you want to use a different Vendor name from the "VyDevSoft" provided, you may do so by changing the component property value **Vendor** to the desired value, and placing the .ini file accordingly.

After you have placed the .ini file appropriately and changed the Vendor property, if desired, to correspond to the location of the .ini file you should be able to successfully execute the test program.

The test program has a single method executed by the corresponding button or menu item. The comments explain the reason for each statement.