DSAMsg unit

This unit provides "Don't Show Again" dialog and form services. Included is a form class that you can descend your own forms from, routines for showing standarad TForm descendants, and replacement rountines for the MessageDlg function. DSA services allow the user to specify whether or not they want to see the dialog or form in the future with only minimal effort on the application programmer's part.

The dialog has a check box positioned at the bottom left corner which the user can check to specify that he does not wish to see it again. If checked, calling the function again will not display the dialog, it will simply return a default value immediately.

Procedures to get and set the state of the dialog are also provided so that you can programmatically re enable a dialog that has been hidden by the user.

Components
TDSAForm

Routines
DSAClear
DSAFormClear
DSAFormGetState
DSAFormSetState
DSAGetState
DSAIdentsClear
DSAIdentsGetState
DSAIdentsMessageDlg
DSAIdentsSetState
DSAIdentsShowModal
DSAMessageDlg
DSASetState
DSAShowModal
Register

Constants
DefaultFilename
DontShowMsgText
DSA_CHECKBOX_NAME
RegRootKey
UseRegistry

TDSAForm Component

Properties Methods

Unit
DSAMsg

Description
The TDSAForm class is a TForm descendant that you can base your forms on to easily provide "Don't Show Again" functionalitiy. Several properties have been added to provide complete control over where the information on the displayable state of the dialog is stored, but you will often find that simply leaving them blank (which uses default values based on your application) will be sufficient.

The DSA_CheckBox property is key as it defines the TCheckbox component on the form to be used to indicate whether or not the user wishes to see the form in the future or not. You must assign a TCheckbox to it, or the DSA state can not be saved.

The two key methods that you need to be aware of are DSAShow and DSAShowModal. Because the Show and ShowModal methods of TForm are not virtual, they can not be overriden by descendants. Therefore, I had to provide completely new equivalents of these functions. One nice side effect of this is that if you want to treat the form as DSA in some situations but not others, you would simply call the old methods and not have to fool with making sure it was displayable first.

Design time support of the new properties is available for Delphi 3. Previous versions of Delphi and C++Builder 1.0 do NOT support design time access of TForm descendants. Sorry, just be happy Borland added it to Delphi 3 (and I presume C++ B 3.0). Unlike a normal component, TForm descendant classes must also have a package installed for the registration process to work. See the installation notes in DSAMsg.Txt for complete installation instructions.

If you are not using Delphi 3, you can still have your forms descend from TDSAForm, you simply won't have design time access to the properties. In that case you will have to set the property values in code, most likely in the form's OnCreate event handler or just after calling the form's Create constructor if operating outside of the form's code.

Properties

Run-time only
Key properties

DSA_CheckBox | DSA_Filename | DSA_Showable
DSA_DefaultResult | DSA_ID | DSA_UseRegistry

Methods

Key methods

DSAClear | DSAShow | DSAShowModal

DSA_CheckBox property

Applies to
TDSAForm

Declaration
```
property DSA_CheckBox: TCheckBox;
```

Description
The DSA_CheckBox property identifies the TCheckBox component on the form that should be treated as the "Don't Show Again" checkbox. If this property is blank (NIL), DSA functionality is disabled for the form (i.e. DSAShowModal and DSAShow will always show the form).

Simply assign any existing TCheckBox component on the form to this property and when the form is destroyed, this checkbox will be used to indicate the displayable state of the form for future calls to DSAShow and DSAShowModal.

If the box is checked, that indicates that it should not be shown in the future. The checkbox's caption should be worded accordingly.

DSA_DefaultResult property

Applies to
TDSAForm

Declaration
```
property DSA_DefaultResult: integer;
```

Description
The DSA_DefaultResult property is used to specify what value the DSAShowModal method should return if the user has elected not to display the form.

DSA_Filename property

Applies to
TDSAForm

Declaration
```
property DSA_Filename: string;
```

Description
DSA_Filename is the INI file name (DSA_UseRegistry = FALSE) or Registry path (Win32 only, DSA_UseRegistry = TRUE) that is used in conjunction with the DSA_ID property to store the displayable state of the form.

If this value is blank, the value of the DefaultFilename global is used.

DSA_ID property

Applies to
TDSAForm

Declaration
```
property DSA_ID: string;
```

Description
DSA_ID is the INI section name (DSA_UseRegistry = FALSE) or Registry path subkey (Win32 only, DSA_UseRegistry = TRUE) that is used in conjunction with the DSA_Filename property to store the displayable state of the form.

If this value is blank, the value of the ClassName property is used.

DSA_Showable property

Applies to
TDSAForm

Declaration
```
property DSA_Showable: boolean;
```

Description
The DSA_Showable property is used to check or set the displayable state of the form.

Normally, there is no way to re enable a form once the user has turned it off since it isn't displayed any longer. However, it is wise to include the ability to turn these back on in case the user disabled it by accident, or changes its mind. This functionality is often found in a configuration/settings dialog.

Setting this value to FALSE is equivalent to calling the DSAClear method.

DSA_UseRegistry property

Applies to
TDSAForm

Declaration
```
property DSA_UseRegistry: boolean;
```

Description
The DSA_UseRegistry property is used to indicate if the displayable state of the form should be stored in the registry or an INI file. The actual location in the registry or INI file is controlled by the DSA_Filename and DSA_ID properties. This property merely indicates how those values should be interpretted.

This property is not avaible under Delphi 1.

DSAClear method

Applies To
TDSAForm

Declaration
```
procedure DSAClear;
```

Description
Use the DSAClear method to reset form's displayable state. That is, if the user has elected not to show the form, you can reset it so that it will show up again.

Normally, there is no way to re enable a form once the user has turned it off since it isn't displayed any longer. However, it is wise to include the ability to turn these back on in case the user disabled it by accident, or changes its mind. This functionality is often found in a configuration/settings dialog.

DSAShow method

Applies To
TDSAForm

Declaration
```
procedure DSAShow;
```

Description
Shows the form modelessly, or not if the user has requested that it not be shown. In that case, the window is simply closed. The instance of the form is not freed unless you have set CloseAction to caFree in then OnClose event handler.

Note that this function MUST be used in place of the normal Show method because Show is not virtual (i.e. it can't be overriden to modify behavior). One nice side effect of this is that if you want to treat the form as DSA in some situations but not others, you would simply call the Show method and not have to fool with making sure it was displayable first.

DSAShowModal method

Applies To
TDSAForm

Declaration
```
function DSAShowModal: Integer;
```

Description
Shows the form modally, or not if the user has requested that it not be shown. In that case, the window is simply closed and the value of DSA_DefaultResult is returned. The instance of the form is not automatically freed, and must be treated as you would any normal TForm.

Note that this function MUST be used in place of the normal ShowModal method because ShowModal is not virtual (i.e. it can't be overriden to modify behavior). One nice side effect of this is that if you want to treat the form as DSA in some situations but not others, you would simply call the ShowModal method and not have to fool with making sure it was displayable first.

Register routine

Unit
DSAMsg

Declaration
```
procedure Register;
```

Description
Registers the class for use in the Delphi 3 IDE. This applies ONLY to Delphi 3. Previous versions of Delphi and C++Builder 1.0 do NOT support design time access of TForm descendants. Sorry, just be happy Borland added it to Delphi 3 (and I presume C++B 3.0). Unlike a normal component, TForm descendant classes must also have a package installed for the registration process to work. See the installation notes in DSAMsg.Txt for complete installation instructions.

DefaultFilename constant

Example

Unit
DSAMsg

Declaration
```
DefaultFilename: string = '';
```

Description
This writeable constant (also known as a static variable) allows you to control the default registry key or INI filename to use when storing dialog display state information.

This value is used by DSAMessageDlg, and also by DSAIdentsMessageDlg when the Filename parameter has been left blank.

If you are compiling for Win32, the default is 'Software\your_app_title\DSADialogs' where 'your_app title' is the value returned by Application.Title.

If you are compiling for Win16, the default is an INI file with the same name as your executable, and in the same directory.

Example

This example will cause all DSAMessageDlg calls, and DSAIdentsMessageDlg with the Filename parameter blank to store values in the registry under the given key (must be running under Win32).

```
UseRegistry := TRUE;
RegistryRootKey := HKEY_LOCAL_MACHINE;
DefaultFilename := 'Software\MyApp\DSADialogs';
```

This example will cause all DSAMessageDlg calls, and DSAIdentsMessageDlg with the Filename parameter blank to store values in an INI file.

```
{$IFDEF WIN32}
UseRegistry := FALSE;
{$ENDIF}
DefaultFilename := 'MyApp.ini';
```

DontShowMsgText constant

Example

Unit

DSAMsg

Declaration

```
DontShowMsgText: string = '&Don''t show this message again';
```

Description

This writeable constant (also known as a static variable) allows you to control the text that appears next to the check box in DSAMessageDlg and DSAIdentsMessageDlg generated dialogs. By default, this value is "&Don't show this message again". If you do not like this, or if using a foreign language, you can change the text by assigning your own value.

Example

This example will cause the the text in all DSAMessageDlg to read "Don't show this stinking box any more!'.

```
  DontShowMsgText := 'Don''t show this stinking box any more!';
```

DSA_CHECKBOX_NAME constant

Unit
DSAMsg

Declaration
```
DSA_CHECKBOX_NAME = '__DSA_CheckBox';
```

Description
This constant is the string that is assigned to the Name property of the checkbox that is created by the various DSA dialog functions. It is included in case you ever needed to find the checkbox component. You can simply search for it using the FindComponent method, passing this value as the parameter.

Example

This call will re-enable a DSAMessageDlg dialog that has the text 'All temporary files have been deleted.'.

```
DSAClear('All temporary files have been deleted.');
```

DSAClear routine

Example

Unit
DSAMsg

Declaration
```
procedure DSAClear(const Msg: string);
```

Description
Use the DSAClear procedure to reset a DSAMessageDlg dialog that has been disabled by the user. Simply pass the same text in the Msg parameter as you do in the Msg parameter of the DSAMessageDlg function.

Normally, there is no way to re-enable a dialog once the user has turned it off since it isn't displayed any longer. However, it is wise to include the ability to turn these back on in case the user disabled it by accident, or changes its mind. This functionality is often found in a configuration/settings dialog.

Example

This call will re-enable a DSAShowModal form that is of the type TMyDSAForm. Notice that you do not need an instance of the class, just type name.

```
DSAFormClear(TMyDSAForm);
```

DSAFormClear routine

Example

Unit
DSAMsg

Declaration
```
procedure DSAFormClear(const AFormClass: TFormClass);
```

Description
Use the DSAFormClear procedure to reset a DSAShowModal form that has been disabled by the user. Simply pass the class type of the form that was passed to the DSAShowModal function.

Normally, there is no way to re enable a dialog once the user has turned it off since it isn't displayed any longer. However, it is wise to include the ability to turn these back on in case the user disabled it by accident, or changes its mind. This functionality is often found in a configuration/settings dialog.

Example

This example illustrates how to check the displayable state of a class named TMyDSAForm and reset it to be displayable if it is not.

```
if not DSAFormGetState(TMyDSAForm) then
  DSAFormSetState(TMyDSAForm, TRUE);
```

DSAFormGetState routine

Example

Unit
DSAMsg

Declaration
```
function DSAFormGetState(const AFormClass: TFormClass): boolean;
```

Description
This routine allows you to get the displayable state a form that is used with the DSAShowModal or DSAIdentsShowModal function.

Simply pass the form class to the function and if the form has been disabled by the user, this function returns FALSE; otherwise, TRUE is returned.

This function is useful when allowing users to restore DSA dialogs that they have hidden.

Example

This example illustrates how to check the displayable state of a class named TMyDSAForm and reset it to be displayable if it is not.

```
if not DSAFormGetState(TMyDSAForm) then
  DSAFormSetState(TMyDSAForm, TRUE);!!! ENTER DESCRIPTION OF EXAMPLE HERE
```

DSAFormSetState routine

Example

Unit
DSAMsg

Declaration
```
procedure DSAFormSetState(const AFormClass: TFormClass; Value: boolean);
```

Description
This routine allows you to set the displayable state a form that is used with the DSAShowModal or DSAIdentsShowModal function.

Simply pass the form class to the function and a boolean value indicating whether or not the form should be displayed.

This function is useful when allowing users to override DSA dialogs settings, in a configuration dialog for instance.

Example

This example illustrates how to check the displayable state of a DSAMessageDlg that was displayed with the text "Are you sure you want to crash the system?" and reset it to be displayable if it is not.

```
if not DSAGetState('Are you sure you want to crash the system?') then
  DSASetState('Are you sure you want to crash the system?', TRUE);
```

DSAGetState routine

Example

Unit
DSAMsg

Declaration
```
function DSAGetState(Msg: string): boolean;
```

Description
This routine allows you to get the displayable state for a dialog that is used with the DSAMessageDlg function.

Simply pass the same message string to the function that is passed to the DSAMessageDlg function and if the dialog has been disabled by the user, this function returns FALSE; otherwise, TRUE is returned.

This function is useful when allowing users to restore DSA dialogs that they have hidden.

Example

This call will clear the state of a dialog stored in the 'Software\MyApp\WarningDialogs\NoNet' key (UseRegistry set to TRUE and running under Win32).

```
DSAIdentsClear('Software\MyApp\WarningDialogs', 'NoNet');
```

This call will clear the state of a dialog stored in the 'MyApp.INI' file, in the 'NoNet' section.

```
DSAIdentsClear('MyApp.ini', 'NoNet');
```

DSAIdentsClear routine

Example

Unit
DSAMsg

Declaration
```
procedure DSAIdentsClear(Filename, ID: string);
```

Description
Use the DSAIdentsClear procedure to reset a DSAIdentsMessageDlg dialog that has been disabled by the user. Simply pass the same values in the Filename and ID parameters as you do in the Filename and ID parameters of the DSAIdentsMessageDlg function.

Normally, there is no way to re-enable a dialog once the user has turned it off since it isn't displayed any longer. However, it is wise to include the ability to turn these back on in case the user disabled it by accident, or changes its mind. This functionality is often found in a configuration/settings dialog.

Example

This call will check the state of a dialog stored in the 'Software\MyApp\WarningDialogs\NoNet' key (UseRegistry set to TRUE and running under Win32) and restore it if needed.

```
if not DSAIdentsGetState('Software\MyApp\WarningDialogs', 'NoNet') then
  DSAIdentsSetState('Software\MyApp\WarningDialogs', 'NoNet', TRUE);
```

This call will clear the state of a dialog stored in the 'MyApp.INI' file, in the 'NoNet' section.

```
if not DSAIdentsGetState('MyApp.ini', 'NoNet') then
  DSAIdentsSetState('MyApp.ini', 'NoNet', TRUE);
```

DSAIdentsGetState routine

Example

Unit
DSAMsg

Declaration
```
function DSAIdentsGetState(Filename, ID: string): boolean;
```

Description
This routine allows you to get the displayable state for a dialog that is used with the DSAIdentsMessageDlg function.

Simply pass the same Filename and ID parameters to the function that are passed to the DSAIdentsMessageDlg function and if the dialog has been disabled by the user, this function returns FALSE; otherwise, TRUE is returned.

This function is useful when allowing users to restore DSA dialogs that they have hidden.

Example

This example shows a warning dialog, unless the user has indicated that it should not be shown. In that case, the value mrYes is immediately returned. The display state information will be saved to a registry key named 'Software\MyApp\WarningDialogs\NoNet' (UseRegistry set to TRUE and running under Win32).

```
  DSAMessageDlg('The network is unavailble. Cancel the operation?', mtWarning,
[mbYes, mbNo], 0, 'Software\MyApp\WarningDialogs', 'NoNet', mrOK);
```

This example shows is the same, except storage is to an INI file (UseRegistry must by FALSE if running under Win32).

```
  DSAMessageDlg('The network is unavailble. Cancel the operation?', mtWarning,
[mbYes, mbNo], 0, 'MyApp.ini', 'NoNet', mrOK);
```

DSAIdentsMessageDlg routine

Example

Unit
DSAMsg

Declaration

```
function DSAIdentsMessageDlg(const Msg: string; AType: TMsgDlgType; AButtons:
TMsgDlgButtons; HelpCtx: Longint; Filename, ID: string; DefaultResult: word):
Word;
```

Description
A MessageDlg replacement function. This function will display a dialog that is identical to the one that MessageDlg will display, except it will also include a check box in the bottom left corner of the dialog. If the user checks it before closing the dialog, the dialog will not be displayed in the future when this function is called.

The text that appears next to the check box is "Don't show this message again", but this can be changed by using the DontShowMsgText global variable.

If the user elects not to display the dialog in the future, this function stores a value in an INI file or the registry (Win32 only) to identify this fact. Where this value is stored is controlled by the Filename and ID parameters, along with three global variables defined in the DSAMsg unit: UseRegistry, RegRootKey, and DefaultFilename. These values can be changed in your program if you so desire. If you leave the Filename parameter blank, the value in DefaultFilename will be used. If this is also blank, or if ID is blank, an exception will be raised.

If you need to re-enable a dialog that has been disabled, you can use the DSAIdentsClear function, passing it the same Filename and ID parameters as you pass to this function. The dialog will then be displayed when this function is called.

The message box displays the value of the Msg parameter. Use the AType parameter to indicate the purpose of the dialog. Use the AButtons parameter to indicate what buttons should appear in the message box. Use the HelpCtx parameter to specify the context ID for the help topic that should appear when the user clicks the help button or presses F1 while the dialog is displayed. Filename is the INI file name or Registry path (Win32 only) that is used in conjunction with the ID identifier to store the displayable state of the dialog. Use the DefaultResult value to specify what value to return if the user has elected not to display the dialog.

DSAIdentsMessageDlg returns the value of the button the user selected, or the value of the DefaultResult parameter if the dialog was not displayed. These are the possible return values if DefaultResult is not used:

mrNone	mrAbort	mrYes
mrOk	mrRetry	mrNo
mrCancel	mrIgnore	mrAll

Example

This code will check the state of a dialog stored in the 'Software\MyApp\WarningDialogs\NoNet' key (UseRegistry set to TRUE and running under Win32) and restore it if needed.

```
if not DSAIdentsGetState('Software\MyApp\WarningDialogs', 'NoNet') then
  DSAIdentsSetState('Software\MyApp\WarningDialogs', 'NoNet', TRUE);
```

This code will check the state of a dialog stored in the 'MyApp.INI' file, in the 'NoNet' section and restore it if needed.

```
if not DSAIdentsGetState('MyApp.ini', 'NoNet') then
  DSAIdentsSetState('MyApp.ini', 'NoNet', TRUE);
```

DSAIdentsSetState routine

Example

Unit
DSAMsg

Declaration
```
procedure DSAIdentsSetState(Filename, ID: string; Value: boolean);
```

Description
This routine allows you to set the displayable state for a dialog that is used with the DSAIdentsMessageDlg function.

Simply pass the same Filename and ID parameters to the function that are passed to the DSAIdentsMessageDlg function and and a boolean value indicating whether or not the dialog should be displayed by DSAIdentsMessageDlg.

This function is useful when allowing users to set preferences for showing or hiding DSA dialogs from a central location (say a configuration dialog that lists all DSA dialogs).

Example

This example illustrates how to display an existing form named TMyCustomForm with DSA functionality, storing the displayable state in an INI file.

```
  var
    MyForm: TMyCustomForm;
  begin
    MyForm := TMyCustomForm.Create(Application);
    try
      if DSAIdentsShowModal(MyForm, 'MyApp.ini', 'DSADialogs', mrYes) = mrYes
then
      begin
        { user selected the Yes button, or dialog wasn't supposed to display }
      end;
    finally
      MyForm.Free;
    end;
  end;
```

DSAIdentsShowModal routine

Example

Unit
DSAMsg

Declaration

```
function DSAIdentsShowModal(const AForm: TForm; Filename, ID: string;
DefaultResult: word): Word;
```

Description
A TForm.ShowModal replacement function. This function will display the form passed in the AForm parameter using the form class' ShowModal function. However, before ShowModal is called, the function will add a check box in the bottom left corner of the form. If the user checks it before closing the form, the form will not be displayed in the future when this function is called.

The text that appears next to the check box is "Don't show this message again", but this can be changed by using the DontShowMsgText global variable.

If the user elects not to display the form in the future, this function stores a value in an INI file or the registry (Win32 only) to identify this fact. Where this value is stored is controlled by the Filename and ID parameters, along with three global variables defined in the DSAMsg unit: UseRegistry, RegRootKey, and DefaultFilename. These values can be changed in your program if you so desire. If you leave the Filename parameter blank, the value in DefaultFilename will be used. If this is also blank, or if ID is blank, an exception will be raised.

If you need to re-enable a form that has been disabled, you can use the DSAIdentsClear function, passing it the same Filename and ID parameters as you passed to this function. The form will then be displayed when this function is called.

AForm is an instance of the form you want to display already created. Filename is the INI file name or Registry path (Win32 only) that is used in conjunction with the ID identifier to store the displayable state of the dialog. DefaultResult value is used to specify what value to return if the user has elected not to display the form.

DSAIdentsShowModal returns the same value that TForm.ShowModal returns, except if the form is not displayed. In that case, the value of the DefaultResult parameter is returned.

Example

This example shows a simple message dialog, unless the user has indicated that it should not be shown. In that case, the value mrOk is immediately returned.

```
  DSAMessageDlg('All temporary files have been deleted.', mtInformation,
[mbOK], 0, mrOK);
```

DSAMessageDlg routine

Example

Unit
DSAMsg

Declaration
```
function DSAMessageDlg(const Msg: string; AType: TMsgDlgType; AButtons:
TMsgDlgButtons; HelpCtx: Longint; DefaultResult: word): Word;
```

Description
A MessageDlg replacement function. This function will display a dialog that is identical to the one that MessageDlg will display, except it will also include a check box in the bottom left corner of the dialog. If the user checks it before closing the dialog, the dialog will not be displayed in the future when this function is called.

The text that appears next to the check box is "Don't show this message again", but this can be changed by using the DontShowMsgText global variable.

If the user elects not to display the dialog in the future, this function stores a value in an INI file or the registry (Win32 only) to identify this fact. Where this value is stored is controlled by three global variables defined in the DSAMsg unit: UseRegistry, RegRootKey, and DefaultFilename. These values can be changed in your program if you so desire. A unique identifier based on the Msg parameter will also be used. If you need more precise control over the storage location, you should use the DSAIdentsMessageDlg.

If you need to re-enable a dialog that has been disabled, you can use the DSAClear function, passing it the same Msg parameter as you pass to this function. The dialog will then be displayed when this function is called.

The message box displays the value of the Msg parameter. Use the AType parameter to indicate the purpose of the dialog. Use the AButtons parameter to indicate what buttons should appear in the message box. Use the HelpCtx parameter to specify the context ID for the help topic that should appear when the user clicks the help button or presses F1 while the dialog is displayed. Use the DefaultResult value to specify what value to return if the user has elected not to display the dialog.

DSAMessageDlg returns the value of the button the user selected, or the value of the DefaultResult parameter if the dialog was not displayed. These are the possible return values if DefaultResult is not used:

mrNone	mrAbort	mrYes
mrOk	mrRetry	mrNo
mrCancel	mrIgnore	mrAll

Example

This example illustrates how to check the displayable state of a DSAMessageDlg that was displayed with the text "Are you sure you want to crash the system?" and reset it to be displayable if it is not.

```
if not DSAGetState('Are you sure you want to crash the system?') then
  DSASetState('Are you sure you want to crash the system?', TRUE);
```

DSASetState routine

Example

Unit
DSAMsg

Declaration
```
procedure DSASetState(Msg: string; Value: boolean);
```

Description
This routine allows you to set the displayable state for a dialog that is used with the DSAMessageDlg function.

Simply pass the same message string to the function that is passed to the DSAMessageDlg function and a boolean value indicating whether or not the dialog should be displayed by DSAMessageDlg.

This function is useful when allowing users to set preferences for showing or hiding DSA dialogs from a central location (say a configuration dialog that lists all DSA dialogs).

Example

This example illustrates how to display an existing form named TMyCustomForm with DSA functionality, storing the displayable state in the default location.

```
var
  MyForm: TMyCustomForm;
begin
  MyForm := TMyCustomForm.Create(Application);
  try
    if DSAShowModal(MyForm, mrYes) = mrYes then
    begin
      { user selected the Yes button, or dialog wasn't supposed to display }
    end;
  finally
    MyForm.Free;
  end;
end;
```

DSAShowModal routine

Example

Unit
DSAMsg

Declaration
```
function DSAShowModal(const AForm: TForm; DefaultResult: word): Word;
```

Description
A TForm.ShowModal replacement function. This function will display the form passed in the AForm parameter using the form class' ShowModal function. However, before ShowModal is called, the function will add a check box in the bottom left corner of the form. If the user checks it before closing the form, the form will not be displayed in the future when this function is called.

The text that appears next to the check box is "Don't show this message again", but this can be changed by using the DontShowMsgText global variable.

If the user elects not to display the form in the future, this function stores a value in an INI file or the registry (Win32 only) to identify this fact. Where this value is stored is controlled by three global variables defined in the DSAMsg unit: UseRegistry, RegRootKey, and DefaultFilename. These values can be changed in your program if you so desire. A unique identifier based on the class name of the form will also be used. If you need more precise control over the storage location, you should use the DSAIdentsShowModal function.

If you need to re-enable a form that has been disabled, you can use the DSAFormClear function, passing it the class type of the form variable you passed to this function. The form will then be displayed when this function is called.

AForm is an instance of the form you want to display already created. The DefaultResult value is used to specify what value to return if the user has elected not to display the form.

DSAShowModal returns the same value that TForm.ShowModal returns, except if the form is not displayed. In that case, the value of the DefaultResult parameter is returned.

RegRootKey constant

Example

Unit
DSAMsg

Declaration
```
RegRootKey: HKey = HKEY_CURRENT_USER;
```

Description
This writeable constant (also known as a static variable) allows you to control which root registry key is used when storing DSAMessageDlg dialog display state information to the registry. By default, the HKEY_CURRENT_USER key is used, as that is the recommend key for applications to use However, you may assign any of the HKEY_* constants to this to change the root key.

This is only available under Win32. It does not exist in Delphi 1 since the registry is not the same as it is in Win32. Only INI files can be used in Delphi 1.

Example

This example will cause all DSAMessageDlg calls, and DSAIdentsMessageDlg with the Filename parameter blank to store values in the registry under the given key (must be running under Win32).

```
UseRegistry := TRUE;
RegistryRootKey := HKEY_LOCAL_MACHINE;
DefaultFilename := 'Software\MyApp\DSADialogs';
```

UseRegistry constant

Example

Unit
DSAMsg

Declaration
```
UseRegistry: boolean = TRUE;
```

Description
This writeable constant (also known as a static variable) allows you to control whether the DSAMessageDlg dialog display state storage uses the registry or an INI file. By default, the registry is used for Win32 (Windows 95 and Windows NT), but simply setting this to FALSE will cause values to saved in an INI file.

This is only available under Win32. It does not exist in Delphi 1 since the registry is not the same as it is in Win32. Only INI files can be used in Delphi 1.

Example

This example will cause all DSAMessageDlg calls, and DSAIdentsMessageDlg with the Filename parameter blank to store values in an INI file.

```
{$IFDEF WIN32}
UseRegistry := FALSE;
{$ENDIF}
DefaultFilename := 'MyApp.ini';
```