PSetting unit

The PSetting unit contains the TPFormSettings and TPAppSettings components.

Components

TPSetting
TPSettings
TPFormSettings
TPAppSettings

Exceptions

EPSettingException
EPSettingsException
EPFormSettingsException
EPAppSettingsException

Types

TPAppSaveLocType
TPEvMRUClick
TPEvUserRestore
TPEvUserSave
TPFormSaveOpt
TPFormSaveOpts
TPMRUDisplayType
TPRootKeyType
TPSaveLocationType

EPSettingException exception

Unit

PSetting

Description

EPSettingException is raised by TPSetting objects. The most common cause of an EPSettingException is a data conversion failure. See AsBoolean, AsDateTime, AsFloat, AsInteger, AsString, AsVariant for more details about data conversion.

EPSettingsException exception

Unit

PSetting

Description

EPSettingsException is raised by TPSettings objects.

EPFormSettingsException exception

Unit

PSetting

Description

EPFormSettingsException is raised by TPFormSettings objects.   This exception may be caused by errors in storing and retrieving settings.

EPAppSettingsException exception

Unit

PSetting

Description

EPAppSettingsException is raised by TPAppSettings objects.   This exception may be caused by errors in storing and retrieving settings.

TPSetting component

Properties	Methods

Unit

PSetting

Description

The TPSetting component is the base component used by both TPFormSettings and TPAppSettings. The TPSetting components are maintained automatically. To interact directly with a TPSetting component, you must specify a name for the setting. This is done in the component editor. Generic settings are named automatically. Form-related or property-related settings are only named if you have entered a name in the External Name field on the component editor.

TPSetting components are similar to TField components and provide similar access properties. By setting the value of a TPSetting which is a property-related setting, you also update the property.

There is no design-time interaction with TPSetting components except through the TPFormSettings and TPAppSettings component editor.

Properties

▶ Run-time only
🔑 Key properties

▶ 🔑	AsBoolean
▶ 🔑	AsDateTime
▶ 🔑	AsFloat
▶ 🔑	AsInteger
▶ 🔑	AsString
▶ 🔑	AsVariant
▶ 🔑	BoundProperty
▶ 🔑	Encrypted
▶ 🔑	Name
▶ 🔑	RelAppSetting
▶ 🔑	UniqueName

Methods

Key methods

- GetAsStrings
- SetAsStrings

AsBoolean property

See also Example

Applies to

TPSetting component

Declaration

```
property AsBoolean: Boolean;
```

Description

Run-time only

Sets or returns the value of the TPSetting as a Boolean.

If the current value of the TPSetting cannot be converted to a boolean, an exception will be raised.

See also

AsDateTime
AsFloat
AsInteger
AsString
AsVariant

GetAsStrings
SetAsStrings

AsBoolean property example

```
var
      tempBool : Boolean;
begin
      PFormSettings['Some Setting'].AsBoolean := True;

      tempBool := PFormSettings['Some Setting'].AsBoolean;
end;
```

AsDateTime property

See also Example

Applies to

TPSetting component

Declaration

```
property AsDateTime: TDateTime;
```

Description

Run-time only

Sets or returns the value of the TPSetting as a TDateTime.

If the current value of the TPSetting cannot be converted to a TDateTime, an exception will be raised.

See also

AsBoolean
AsFloat
AsInteger
AsString
AsVariant

GetAsStrings
SetAsStrings

AsDateTime property example

```
var
      tempVal : TDateTime;
begin
      PFormSettings['Some Setting'].AsDateTime:= Now;

      tempVal := PFormSettings['Some Setting'].AsDateTime;
end;
```

AsFloat property

See also Example

Applies to

TPSetting component

Declaration

```
property AsFloat: Extended;
```

Description

Run-time only

Sets or returns the value of the TPSetting as an Extended.

If the current value of the TPSetting cannot be converted to an extended floating-point value, an exception will be raised.

See also

AsBoolean
AsDateTime
AsInteger
AsString
AsVariant

GetAsStrings
SetAsStrings

AsFloat property example

```
var
      tempVal : Extended;
begin
      PFormSettings['Some Setting'].AsFloat:= 1.2;

      tempVal := PFormSettings['Some Setting'].AsFloat;
end;
```

AsInteger property

See also Example

Applies to

TPSetting component

Declaration

```
property AsInteger: LongInt;
```

Description

Run-time only

Sets or returns the value of the TPSetting as a LongInt.

If the current value of the TPSetting cannot be converted to a LongInt, an exception will be raised.

See also

AsBoolean
AsDateTime
AsFloat
AsString
AsVariant

GetAsStrings
SetAsStrings

AsInteger property example

```
var
      tempVal : LongInt;
begin
      PFormSettings['Some Setting'].AsInteger:= 102;

      tempVal := PFormSettings['Some Setting'].AsInteger;
end;
```

AsString property

See also Example

Applies to

TPSetting component

Declaration

```
property AsString: String;
```

Description

Run-time only

Sets or returns the value of the TPSetting as a String.

See also

AsBoolean
AsDateTime
AsFloat
AsInteger
AsVariant

GetAsStrings
SetAsStrings

AsString property example

```
var
      tempVal : String;
begin
      PFormSettings['Some Setting'].AsString:= 'Some String Value';

      tempVal := PFormSettings['Some Setting'].AsString;
end;
```

AsVariant property

See also Example

Applies to

TPSetting component

Declaration

```
property AsVariant: Variant;
```

Description

Run-time only

Sets or returns the value of the TPSetting as a Variant.

See also

AsBoolean
AsDateTime
AsFloat
AsInteger
AsString

GetAsStrings
SetAsStrings

AsVariant property example

```
var
      tempVal : Variant;
begin
      PFormSettings['Some Setting'].AsString:= 'Some Variant Value';

      tempVal := PFormSettings['Some Setting'].AsVariant;
end;
```

BoundProperty property

See also Example

Applies to

TPSetting component

Declaration

```
property BoundProperty: String;
```

Description

Read-only

Returns the name of the property to which this TPSetting is bound.   Some TPSettings will have no BoundProperty.

This can be useful when writing custom save and restore methods for your application.

See also

BoundProperty
Encrypted
Name
RelAppSetting
UniqueName

BoundProperty property example

```
var
    tempSetting;
begin
    tempSetting := TPFormSettings['Some Setting'];

    MessageDlg( Format( 'The name of the property for this setting is "%s"', [tempSetting.BoundProperty] ),
                        mtInformation, [mbOK], 0 );
end;
```

Encrypted property

See also Example

Applies to

TPSetting component

Declaration

```
property Encrypted: Boolean;
```

Description

Read-only

Returns True if this TPSetting will be stored encrypted.

NOTE: If you are writing a custom save and restore method for your application, you will need to implement the encryption scheme yourself. You can use this property to check if the data should be stored encrypted.

See also

BoundProperty
Encrypted
Name
RelAppSetting
UniqueName

Encrypted property example

```
var
      tempSetting;
begin
      tempSetting := TPFormSettings['Some Setting'];

      if ( tempSetting.Encrypted = True ) then
      begin
            MessageDlg( Format( 'The setting, "%s", should be stored encrypted', [tempSetting.Name] ),
                                    mtInformation, [mbOK], 0 );
      end;
end;
```

Name property

See also Example

Applies to

TPSetting component

Declaration

```
property Name: String;
```

Description

Read-only

Returns the name of the TPSetting. Some TPSettings will have no Name.

This information will be useful if you are writing a custom save and restore method for your application.

See also

BoundProperty
Encrypted
Name
RelAppSetting
UniqueName

Name property example

```
var
      tempSetting;
begin
      tempSetting := TPFormSettings['Some Setting'];

      { In this case, the Name will by 'Some Setting' }
      MessageDlg( Format( 'The name of this setting is "%s".', [tempSetting.Name] ),
                              mtInformation, [mbOK], 0 );

end;
```

RelAppSetting property

See also Example

Applies to

TPSetting component

Declaration

```
property RelAppSetting: String;
```

Description

Read-only

Returns the related application setting for this setting.   This information applies only to property-related settings and is set using the TPFormSettings component editor.

If you are writing a custom save and restore method for your application, you can check this value to determine if the setting should be stored by the form or the application.

See also

BoundProperty
Encrypted
Name
RelAppSetting
UniqueName

RelAppSetting property example

```
var
      tempSetting;
begin
      tempSetting := TPFormSettings['Some Setting'];

      if ( tempSetting.RelAppSetting <> '' ) then
      begin
            MessageDlg( Format( 'This setting is related to the application setting, "%s".', [tempSetting.RelAppSetting] ),
                                    mtInformation, [mbOK], 0 );
      end;
end;
```

UniqueName property

See also   Example

Applies to

TPSetting component

Declaration

```
property UniqueName: String;
```

Description

Read-only

Returns a unique name of the TPSetting.   It is simply the concatenation of the BoundProperty and Name properties.   It is not necessarily guaranteed to be unique under all circumstances, but normally will be.

This method is especially useful when writing a custom save and restore method for your application.

See also

BoundProperty
Encrypted
Name
RelAppSetting
UniqueName

UniqueName property example

```
var
      tempSetting;
begin
      tempSetting := TPFormSettings['Some Setting'];

      MessageDlg( Format( 'The unique name for this setting is "%s".', [tempSetting.UniqueName] ),
                                    mtInformation, [mbOK], 0 );
end;
```

GetAsStrings method

See also Example

Applies to

TPSetting component

Declaration

```
procedure GetAsStrings(aStrings: TStrings);
```

Description

Populates the parameter aStrings with the value of the TPSetting. If the value is not a TStrings object, the aStrings object will receive a single string representing the value.

See also

AsBoolean
AsDateTime
AsFloat
AsInteger
AsString
AsVariant

SetAsStrings

GetAsStrings method example

```
begin
	{ copies the items in the ListBox to the TPSetting named "List" and the copies the strings to the Memo }
	PFormSettings['List'].SetAsStrings( ListBox.Items );
	PFormSettings['List'].GetAsStrings( Memo.Lines );
end;
```

SetAsStrings method

See also Example

Applies to

TPSetting component

Declaration

```
procedure SetAsStrings(aStrings: TStrings);
```

Description

Sets the value of the TPSetting to the strings, aStrings.

See also

AsBoolean
AsDateTime
AsFloat
AsInteger
AsString
AsVariant

GetAsStrings

SetAsStrings method example

```
begin
	{ copies the items in the ListBox to the TPSetting named "List" and the copies the strings to the Memo }
	PFormSettings['List'].SetAsStrings( ListBox.Items );
	PFormSettings['List'].GetAsStrings( Memo.Lines );
end;
```

TPSettings component

Properties Methods

Unit

PSetting

Description

TPSettings is the abstract base class for the TPFormSettings and TPAppSettings components. TPSettings maintains the list of TPSetting components and provides a mechanism for accessing them. TPSettings also maintains several of the properties and methods shared between the TPFormSettings and TPAppSettings components.

Properties

Run-time only
Key properties

CompanyName

OnUserRestore
OnUserSave
RegistryRootKey

Settings

SoftwareName
SoftwareVersion

Methods

🔑 Key methods

🔑 AddSetting

CompanyName property

Applies to

TPSettings component

Declaration

```
property CompanyName: String;
```

Description

This is used to create the base key for the Registry.   If the settings are not going to be saved in the Registry, this value has no meaning.

When this value is set to a TPAppSettings component, any TPFormSettings components which are attached to the TPAppSettings component will automatically be updated.

OnUserRestore property

See also Example

Applies to

TPSettings component

Declaration

```
property OnUserRestore: TPEvUserRestore;
```

Description

This event is triggered when you have selected User-Defined for the storage location in either the TPFormSettings or TPAppSettings component and the component settings are being restored. The event is triggered once for each setting. You can use this to create your own save and restore method such as storing settings in a database or in an initialization file.

See also

OnUserSave

OnUserRestore property example

```
procedure TForm1.MyUserRestore( Sender : TObject; Setting : TPSetting );
begin
    { The RestoreSetting method is a user-defined method }
    Setting.AsString := RestoreSetting( Setting.UniqueName );
end;
```

OnUserSave property

See also   Example

Applies to

TPSettings component

Declaration

```
property OnUserSave: TPEvUserSave;
```

Description

This event is triggered when you have selected User-Defined for the storage location in either the TPFormSettings or TPAppSettings component and the component settings are being saved. The event is triggered once for each setting. You can use this to create your own save and restore method such as storing settings in a database or in an initialization file.

See also

OnUserRestore

OnUserSave property example

```
procedure TForm1.MyUserRestore( Sender : TObject; Setting : TPSetting );
begin
    { The SaveSetting method is a user-defined method }
    SaveSetting( Setting.UniqueName, Setting.AsString );
end;
```

RegistryRootKey property

Applies to

TPSettings component

Declaration

```
property RegistryRootKey: TPRootKeyType;
```

Description

This specifies the root key when the save location is the Registry.

Settings property

Example

Applies to

TPSettings component

Declaration

```
property Settings: TPSetting;
```

Description

Run-time only
Read-only

Returns a TPSetting by name.

There is no direct access to TPSetting components which do not have a name.

Settings property example

```
var
    tempSetting : TPSetting;
begin
    tempSetting := PFormSettings['Some Setting'];

    { Use setting here }

end;
```

SoftwareName property

Applies to

TPSettings component

Declaration

```
property SoftwareName: String;
```

Description

This is used to create the base key for the Registry.   If the settings are not going to be saved in the Registry, this value has no meaning.

When this value is set to a TPAppSettings component, any TPFormSettings components which are attached to the TPAppSettings component will automatically be updated.

SoftwareVersion property

Applies to

TPSettings component

Declaration

```
property SoftwareVersion: String;
```

Description

This is used to create the base key for the Registry.   If the settings are not going to be saved in the Registry, this value has no meaning.

When this value is set to a TPAppSettings component, any TPFormSettings components which are attached to the TPAppSettings component will automatically be updated.

AddSetting method

Example

Applies to

TPSettings component

Declaration

```
procedure AddSetting(aName: String; storeEncrypted: Boolean);
```

Description

This method allows you to add settings at run time.   Be careful, however, not to use this mechanism when the Save Location is either the Registry or User-Defined since the values for all the settings will be restored before you can create a new setting using this method.   Therefore, the only time that this method should be used is when the Save Location is a Stream or a Storage file. Then you can add new settings before calling RestoreFromStream or RestoreFromStorage.

AddSetting method example

```
begin
    { create a new setting }
    PFormSetttings.AddSetting( 'Password', True );
    PFormSettings.RestoreFromStream( fGlobalStream );

    if ( PFormSettings['Password'] = 'FooBar' ) then
    begin
        DoSomething;
    end;
end;
```

TPFormSettings component

See also Properties Methods

Unit

PSetting

Description

The TPFormSettings component is a descendent of the TPSettings component. The TPFormSettings component is intended to be used for saving a restoring form-related settings.

The TPFormSettings component may only be dropped on TForm descendents. It is designed to automatically save basic form information such as Left, Top, Width, Height, and WindowState based on the type of its parent form. If the parent form is a dialog box, for example, only the Left and Top will be stored. You can easily change the information that is stored by setting the appropriate properties.

TPFormSettings also makes it very easy to maintain a Most-Recently-Used files list (MRU list) in your application. You can specify the length of the list (in items), the view properties of the items, and the TMenuItem from which the MRU list descends.

TPFormSettings also makes it easy to save and property of the parent form or any property of any component on the parent form. Simply double-click the component at design time to select the properties visually.

TPFormSettings components can be related to TPAppSettings components with the AppSettingsCtrl property. If you have multiple forms in your application, each form will have its own TPFormSettings related to a single application-wide TPAppSettings component. With very few lines of code, similar properties on different forms can be synchronized automatically through the TPAppSettings component.

See also

TPAppSettings component

Properties

▶ Run-time only
🔑 Key properties
🔑 AppSettingsCtrl
- 🔑 FormSaveOpt
- 🔑 MRUDisplay
- 🔑 MRULength
- 🔑 MRUMaxItemWidth
- 🔑 MRUMenuItem
- 🔑 OnMRUClick
- 🔑 SaveLocation

From TPSettings (ancestor component)

- 🔑 CompanyName
- 🔑 OnUserRestore

🔑 OnUserSave
- 🔑 RegistryRootKey

🔑
🔑 Settings
- 🔑 SoftwareName
- 🔑 SoftwareVersion

Methods

Key methods

- MRUAdd
- RestoreFromStorage
- RestoreFromStream
- SaveToStorage
- SaveToStream
- UpdateToAppSettings

From TPSettings (ancestor component)

- AddSetting

About the TPFormSettings component

See also TPFormSettings reference

Purpose

The TPFormSettings component is designed to simplify the requirements for maintaining end-user-defined application settings and other form and application information.   This information can be maintained in the Registry, in a file (Stream or OLE Structured Storage), or in any programmer-defined location such as a database or initialization file.

The TPFormSettings can be used simply to store form size and location, or it can be used to save Most-Recently-Used file lists, property values, internal settings, along with form size, location, and state information.

Along with TPAppSettings, the TPFormSettings component makes possible the persistent storage requirements of today's complicated applications without a large investment in time or learning.   All the details of using the Registry, and maintaining the information (RTTI) are encapsulated in the components.

See also

TPAppSettings component

AppSettingsCtrl property

Applies to

TPFormSettings component

Declaration

```
property AppSettingsCtrl: TPAppSettings;
```

Description

Specifies the TPAppSettings to which this TPFormSettings component is related.

FormSaveOpt property

Example

Applies to

TPFormSettings component

Declaration

```
property FormSaveOpt: TPFormSaveOpt;
```

Description

Specifies which of the basic form properites (Left, Top, Width, Height, WindowState) to maintain.

FormSaveOpt property example

```
PFormSettings.FormSaveOpt := [pfsLeft, pfsTop];
```

MRUDisplay property

Example

Applies to

TPFormSettings component

Declaration

```
property MRUDisplay: TPMRUDisplayType;
```

Description

Specifies how each MRU item will appear in the menu.

MRUDisplay property example

PFormSettings.MRUDisplayType := `pmdFileOnly;`

MRULength property

Example

Applies to

TPFormSettings component

Declaration

```
property MRULength: Integer;
```

Description

Specifies the length of the MRU list in number of items.   The valid range is 0 - 9.

MRULength property example

```
PFormSettings.MRULength := 5;
```

MRUMaxItemWidth property

See also		Example

Applies to

TPFormSettings component

Declaration

```
property MRUMaxItemWidth: Integer;
```

Description

Specifies the maximum width (in pixels) of a MRU item appearing in the menu.   This value is only meaningful if the MRUDisplay property is set to pmdAbbPath.   The valid range is 100-300.

See also

MRUDisplay property

MRUMaxItemWidth property example

```
PFormSettings.MRUMaxItemWidth := 250;
```

MRUMenuItem property

Example

Applies to

TPFormSettings component

Declaration

```
property MRUMenuItem: TMenuItem;
```

Description

Specifies the menu item to which the MRU list will be appended.   If the menu item is not empy, a separator will be placed before the items.

MRUMenuItem property example

```
PFormSettings.MRUMenuItem := File1; { attaches the MRU list to the menu item named "File1" }
```

OnMRUClick property

Example

Applies to

TPFormSettings component

Declaration

```
property OnMRUClick: TPEvMRUClick;
```

Description

This event is triggered whenever one of the MRU menu items is clicked.

OnMRUClick property example

```
procedure TForm1.MyMRUClick(Sender: TObject; MenuItem: TMenuItem; FileName: String; MRUIndex:
Integer);
begin
    OpenFile( FileName );
end;
```

SaveLocation property

Applies to

TPFormSettings component

Declaration

```
property SaveLocation: TPSaveLocationType;
```

Description

Specifies where to store the settings information.   Selecting pslRegistry will cause all the form settings to be stored in the Registry.   Selecting pslStream requires you to call the SaveToStream and RestoreFromStream methods at shutdown and startup, respectively.   Selecting pslStorage is similar to pslStream.   Selecting pslUser will cause the TPFormSettings component to trigger OnUserRestore and OnUserSave events for each setting.

This property may not be changed at run time.

MRUAdd method

Example

Applies to

TPFormSettings component

Declaration

```
procedure MRUAdd(aPathName: String);
```

Description

This method adds a new MRU filename to the MRU list.

Note: The SaveToStorage and SaveToStream methods can also update the MRU list.

MRUAdd method example

```
PFormSettings.MRUAdd( 'c:\windows\win.ini' );
```

RestoreFromStorage method

See also Example

Applies to

TPFormSettings component

Declaration

```
procedure RestoreFromStorage(aStorage: IStorage);
```

Description

This method is called to restore all the settings information from an OLE Structured Storage file. The caller must provide a valid, open IStorage pointer. This method should be used to retrieve previously saved settings information from a Storage file.

See also

SaveToStorage method

RestoreFromStorage method example

```
PFormSettings.RestoreFromStorage( myStorage );
```

RestoreFromStream method

See also Example

Applies to

TPFormSettings component

Declaration

```
procedure RestoreFromStream(aStream: TStream);
```

Description

This method is used to restore all settings information from a TStream. The caller must provide a valid TStream object as the parameter to the method. The TPFormSettings component will begin reading from the current stream position and will not seek for the information in the TStream. Therefore, you must use caution when saving to and restoring from TStreams.

See also

SaveToStream method

RestoreFromStream method example

```
var
      aStream : TFileStream;
begin
      aStream := TFileStream.Create( 'c:\filename.dat', fmOpenRead );
      RestoreOtherInformation( aStream );
      PFormSettings.RestoreFromStream( aStream );
      aStream.Free;
end;
```

SaveToStorage method

See also Example

Applies to

TPFormSettings component

Declaration

```
procedure SaveToStorage(aStorage: IStorage; aPathName: String);
```

Description

This method is used to store all settings information to an OLE Structured Storage file. The caller must provide a valid IStorage pointer.

If aPathName is not empty, it is added to the MainForm's MRU list. This is a shortcut method for adding files to the MRU list.

See also

RestoreFromStorage method

SaveToStorage method example

```
PFormSettings.SaveToStorage( myStorage );
```

SaveToStream method

See also Example

Applies to

TPFormSettings component

Declaration

```
procedure SaveToStream(aStream: TStream; aPathname: String);
```

Description

This method is used to save all settings information to a TStream. The caller must provide a valid TStream object. Use caution when saving and restoring information in TStreams. TPFormSettings will begin writing to the TStream at the current location. It will not seek to the beginning or end of the TStream.

If aPathName is not empty, it is added to the MainForm's MRU list. This is a shortcut method for adding files to the MRU list.

See also

RestoreFromStream method

SaveToStream method example

```
var
      aStream : TFileStream;
begin
      aStream := TFileStream.Create( 'c:\filename.dat', fmOpenWrite or fmCreate );
      SaveOtherInformation( aStream );
      PFormSettings.SaveToStream( aStream );
      aStream.Free;
end;
```

UpdateToAppSettings method

Example

Applies to

TPFormSettings component

Declaration

```
procedure UpdateToAppSettings;
```

Description

This method updates all of the application-related settings for the TPFormSettings component.   This method is most commonly used when implementing options dialog boxes.   After the options dialog is closed, this method is called which forces all of the application-related settings to be updated and propogated through the application.

UpdateToAppSettings method example

```
begin
    frmOptions.ShowModal; { show the options dialog }
    if ( frmOptions.ModalResult = mrOK ) then
    begin
        frmOptions.PFormSettings.UpdateToAppSettings;
    end;
end;
```

TPAppSettings component

See also Properties Methods

Unit

PSetting

Description

The TPAppSettings component is similar to the TPFormSettings component except that it does not maintain any property-related settings or form-related settings. All of the settings maintained by the TPAppSettings component are named settings. TPFormSettings components may bind themselves to a TPAppSettings component in order to maintain similar properties throughout an application.

TPAppSettings components may be dropped onto forms or data modules as needed. Typically only a single TPAppSettings component exists in an application and resides in a data module.

See also

TPFormSettings component

Properties

Run-time only
Key properties
- SaveLocation

From TPSettings (ancestor component)

- CompanyName
- OnUserRestore
- OnUserSave
- RegistryRootKey
- Settings
- SoftwareName
- SoftwareVersion

Methods

Key methods

UpdateToFormSettings

From TPSettings (ancestor component)

AddSetting

About the TPAppSettings component

See also TPAppSettings reference

Purpose

The TPAppSettings component is designed to maintain all application-wide settings such as passwords and options. All settings are accessed by name. The TPAppSettings component can also be used to synchronize similar settings throughout an application.

See also

TPFormSettings component

SaveLocation property

Applies to

TPAppSettings component

Declaration

```
property SaveLocation: TPAppSaveLocType;
```

Description

Specifies where to store the settings information.   Selecting pslUser will cause the TPFormSettings component to trigger OnUserRestore and OnUserSave events for each setting.

This property may not be changed at run time.

UpdateToFormSettings method

Example

Applies to

TPAppSettings component

Declaration

```
procedure UpdateToFormSettings;
```

Description

Informs all TPFormSettings components to update their appliation-bound settings based on the current values.

UpdateToFormSettings method example

```
PAppSettings.UpdateToFormSettings;
```

TPAppSaveLocType type

Unit

PSetting

Declaration

```
type TPAppSaveLocType = (palRegistry, palUser);
```

TPEvMRUClick type

Unit

PSetting

Declaration

```
type TPEvMRUClick = procedure(Sender: TObject; MenuItem: TMenuItem; FileName: String;
MRUIndex: Integer) of Object;
```

TPEvUserRestore type

Unit

PSetting

Declaration

```
type TPEvUserRestore = procedure(Sender: TObject; Setting: TPSetting) of Object;
```

TPEvUserSave type

Unit

PSetting

Declaration

```
type TPEvUserSave = procedure(Sender: TObject; Setting: TPSetting) of Object;
```

TPFormSaveOpt type

Unit

PSetting

Declaration

```
type TPFormSaveOpt = set of TPFormSaveOpts;
```

TPFormSaveOpts type

Unit

PSetting

Declaration

```
type TPFormSaveOpts = (pfsLeft, pfsTop, pfsWidth, pfsHeight, pfsState);
```

TPMRUDisplayType type

Unit

PSetting

Declaration

```
type TPMRUDisplayType = (pmdFileOnly, pmdFullPath, pmdAbbPath);
```

TPRootKeyType type

Unit

PSetting

Declaration

```
type TPRootKeyType = (prkCurrentUser, prkLocalMachine);
```

TPSaveLocationType type

Unit

PSetting

Declaration

```
type TPSaveLocationType = (pslRegistry, pslStream, pslStorage, pslUser);
```