plist Class for REALbasic

Table of Contents

The plist class is designed to allow you to easily maintain an Apple plist-standard preferences file for your applications.

New to Version 2.11!

- **Move** and **Copy** can now move and copy items from one plist to another.
- The **saveFile** property of plist is now global. See the **saveFile** property for more information.

Installation

To use plist in your application, drag both plist and plistDict onto your project.

Initialization

Use the new constructor to initialize plist (i.e. prefs=new plist). You must pass at least one parameter to the constructor. This is a folder item that contains the path to the preferences file you wish to open. The second, optional, parameter is a folder item that points to a template file. This template file would be your application's default preferences. This file would be initially read from if the preferences file does not exist.
It is highly recommended that you use the Property List Editor to build a default preferences file for your application. While you could build one from scratch within plist, it could be a daunting task if you have a complex preferences file.

Examples:
This example initializes plist without a template file.

```
f=PreferencesFolder.child("myapp.plist")
prefs=new plist(f)
```

This example initializes plist with a template file.

```
prefFile=PreferencesFolder("myapp.plist")
templateFile=getFolderItme("myapp.plist")
prefs=new plist(prefFile,templateFile)
```

When you initialize plist, the pretences in the file you specify are automatically loaded.

plist and plistDict

As you have already noticed, there are two classes used in plist. The plist class handles loading, saving, and error reporting while the plistDict class contains the data of the preferences file and all the methods to manipulate the data.

How to Get and Set Data

plist uses a parent-child relationship similar to the FolderItem class. A plist file consists of a root dictionary. The root dictionary can contain other dictionaries. These dictionaries are children of the root dictionary. The children dictionaries may also contain dictionaries and so on. You, of course, will need to know the structure of your preferences file. Most preferences files are simple and may not contain any children. Some, like the Finder's preferences file, are quite complex.

All the Get methods have an optional default parameter. This allows you to set a default value without having to call a Set method. For example, `prefs.root.GetBoolean(“ShowAll”,true)` would set ShowAll to true if it doesn’t exist and return true.

A very simple example of getting or setting a Boolean value would be:

```
Bool=prefs.root.GetBoolean("ShowAll")
prefs.root.SetBoolean(“ShowAll”,true)
```

Next is a more complex example of getting or setting data in a child of the root:

```
Width=prefs.root.child("WindowPosition").GetInteger("width")
prefs,root.child(“WindowPosition”).SetInteger(“width”,self.width)
```

Dealing With Arrays

Arrays are treated as children. Each element of the array can be anything. It can even be another dictionary or array. Getting and setting data from an array varies depending on where you need to get the data. A few examples should clarify this.

Getting Data from an Element

If the data type of the element is not an array or dictionary, treat the element like a child.

For example, to get an integer from the second element of an array called “myArray”, do this:

```
app.prefs.root.child(“myArray”).GetInteger(“2”)
```

If, however, the element is an array or dictionary, you must use the Index method. If, for example, the third element of the array is a dictionary and in the dictionary is a key called “myData”, you would retrieve it like this:

```
text=prefs.root.child(“myArray”).index(3).GetString(“myData”)
```

If the element was an array, you might use something like this:

```
text=prefs.root.child(“myArray”).index(3).GetString(“2”)
```

Nesting Limits of Arrays and Dictionaries

There are no nesting limits. They can go as deep as you like.

Looping Through a Dictionary

You may loop through a dictionary (i.e. child) one item at a time. Doing so allows you to step through the structure without having to know the structure in advance. This is handy if your application is parsing a plist you did not create. To loop through a dictionary, use the MoveFirst, MoveNext, MoveLast, and CurrentKey methods. You will also need to check the eof property. Just to be safe, use MoveFirst at the beginning of the loop. For example:

```
Dim key as string

prefs.root.MoveFirst
While not prefs.root.eof
	key=prefs.root.CurrentKey
	MsgBox "key="+key+" type="+prefs.root.GetType(key)
	Prefs.root.MoveNext
wend
```

If you wanted to look at the structure of a plist that had several dictionaries, you could call a method recursively. For example:

```
Sub ShowStructure(dict as plistDict)
Dim key,type as string

dictt.MoveFirst
While not prefs.root.eof
	key=dict.CurrentKey
	type=dict.GetType(key)
	MsgBox "key="+key+" type="+type
	If type="dict" or type="array" then
		ShowStructure(dict.child(key))
	end
	Dict.MoveNext
Wend
End sub
```

plist Properties

The properties of the plist class are:

Error

Type: Boolean

If set to true, then an error has occurred. See What Can Cause An Error? for more information.

ErrorMessage

Type: String
If an error has occurred, this will contain the error message. See What Can Cause An Error? for more information.

foundDict

Type: plistDict
Contains the dictionary of the last successful Find.

foundKey

Type: String
Contains of key of the last successful Find.

foundValue

Type: String
Contains the value of the last successful Find.

foundType

Type: String
Contains the type of the last successful Find.

Root

Type: plistDict
This points to the root of the preferences file. When using any method of the **plistDict** class, you must use root! For example,
value=prefs.root.GetInteger("WindowHeight")

saveFile

Type: FolderItem
This points to where the file will be saved and the Save method is called. As of version 2.11, this property is now global. This allows you to easily change where the plist will be saved. For example, you are writing an application where the user can create something and then save his/her work. You've decided that a plist would be a good storage solution. So, as the user is working on his/her project, you could use a temporary plist to store the work. Then, when the user saves it, just set the saveFile property to the FolderItem returned from the Open File Dialog box and then call the plist Save method.

plist Methods

The methods of the plist class are:

ClearSearch

Input: Nothing
Output: Nothing
Clears the search flags used for FindNext. Normally, you will not need to call this. It is automatically called when you use Find or if the previous text searched for doesn’t match the text being searched for in FindNext.

Cleanup

Input: Nothing
Output: Nothing
If the plist file was binary, the Cleanup method will delete any temporary files created.
This method is also called if you set the Clean parameter in the Save method.
Very Important: Only call this method when you quit your application. Calling it earlier may cause unexpected results and crashes.

Find

Input: searchText as String
Output: Boolean
Finds the first occurrence of the search string (searchText) in the plist. If found, true will be returned and the following properties in plist will be populated:

- foundDict – This is the dictionary in which the item was found. This is a pointer to that dictionary. Therefore it is a plistDict object.
- foundKey – The name of the key in which the item was found.
- foundValue – The entire value in which the item was found.
- foundType – The data type in which the item was found.

Example:

```
if prefs.Find(“Smith”) then
        MsgBox “key=”+prefs.foundKey
        MsgBox “value=”+prefs.foundValue
        MsgBox “type=”+prefs.foundType
        MsgBox “Full Name=”+prefs.foundDict.GetString(“firstName”)+”
“+prefs.foundDict.GetString(“lastName”)
End
```

The above example searches for the word “Smith” and returns the key, value, and type. It also uses the foundDict property to display the first and last name of the person found.

FindNext

Input: searchText as String
Output: Boolean
Finds the next occurrence of the search string (searchText) in the plist. If a

search has not been done on this text before, it will find the first occurrence. See Find for more details.

Plist

Input: prefFile as FolderItem,[templateFile as FolderItem]
Output: Object
This is the constructor class. See Initialization above for more details.

Save

Input: [clean as Boolean]
Output: Nothing
Saves the preferences file to disk. If Clean is set to true (the default is false and it is optional), then any temporary files will be deleted. This applies only to saving binary plists. **Very Important**: Only set Clean to true when exiting your program! Doing so in the middle somewhere may cause unexpected results.

Load

Input: prefFile as FolderItem,[templateFile as FolderItem]
Output: Nothing
Loads a preferences file. You do not need to call this after you initialize plist – it is called automatically. Use this method if you wish to re-load the file.

plistDict Properties

Listed below are some properties of plistDict you may find useful.

Name

Type: String
The name of the child (the root is called "root")

Parent

Type: plistDict
A pointer to the parent of the child.

Values

Type: Dictionary
The key-value pairs of the child.

Types

Type: Dictionary
The types of the elements in the child.

EOF

Type: Boolean
End-of-File. Check this property if you are using the MoveNext method.

plistDict Methods

Listed below are all of the methods of plistDict.

AbsolutePath

Input: Nothing
Output: String
Returns a colon-delimited string showing the path from root to the array or dictionary. While this method may not be useful for normal use, it may come in handy if you are looping through a plist you don't know the structure of.

Examples:
path=prefs.root.child("myChild").AbsolutePath – This would be returned as ":root:myChild"

AddArray

Input: name as String
Output: Nothing
Adds an array to a child.

Examples:
prefs.root.AddArray("myArray") – Adds an array to the root.
prefs.root.child("myChild").AddArray("myArray") – Adds an array to a child called "myChild".
prefs.root.child("myRootArray").index(3).AddArray("myArray") – Adds an array to the index of another array. In this case, the array is called "myRootArray" and it is being added to the third element in the array.

AddChild

Input: name as String
Output: Nothing
Adds a child (i.e. dictionary). If the name you pass already exists, the error property will be set.

Examples:
prefs.root.AddChild("newChild") – Adds a child to the root.
prefs.root.child("firstChild").AddChild("anotherChild") – Adds a child to another child called "firstChild"

AppendArray

Input: Nothing
Output: Nothing
Appends an array to an array.

Example:
prefs.root.child("myArray").AppendArray – Appends an array to an array called "myArray"

AppendBoolean

Input: value as Boolean
Output: Nothing
Appends a Boolean value to an array.

AppendColor

Input: value as Color
Output: Nothing
Appends a string value with a hex representation of the color passed. For example, white would be stored as FFFFFF.

AppendData

Input: value as String
Output: Nothing
Appends data to an array.

AppendDate

Input: value as Date
Output: Nothing
Appends a date value to an array.

AppendDict

Input: Nothing
Output: Nothing
Appends a dictionary (i.e. child) to an array.

AppendDouble

Input: value as Double
Output: Nothing
Appends a double value to an array.

AppendInteger

Input: value as Integer
Output: Nothing
Appends an integer value to an array.

AppendList

Input: items() as string, startIndex as integer, endIndex as integer
Output: Nothing
Appends a list of items (array) to an array. The startIndex and endIndex parameters allow you to control which items in the array you want saved to the plist.

Examples:
prefs.root.child("myArray").AppendList(items,1,UBound(items)) – Adds all elements in the array starting at 1.
prefs.root.child("myArray").ApopendList(items,1,2) – Adds only the first and second elements to the array.

AppendListbox

Input: list as Listbox
Output: Nothing
Appends a list box's items to an array.

AppendPopup

Input: list as Listbox
Output: Nothing
Appends a popup menu's items to an array.

AppendString

Input: value as String
Output: Nothing
Appends a string value to an array.

AppendWindow

Input: win as Window
Output: Nothing
Appends a window's title, position, and size to an array.

Child

Input: name as String
Output: plistDict
Returns the plistDict object of the name of the child passed.

Examples:
dim d as plistDict
d=prefs.root.child("myDictionary") – Assigns the child "myDictionary" to the d property.
prefs.root.child("myDictionary").SetString("name","Bob") – Sets the key "name" to the value of "Bob" in the root's child called "myDictionary".

Copy

Input: key as String,dest as plistDict
Output: Nothing
Copies an element from one dictionary or array to another dictionary or array. If the destination is an array, the entry is appended to it.

Example:
prefs.root.Copy("myPage",prefs.root.child("pages")) – This will copy "myPage" to the destination dictionary.

Count

Input: Nothing
Output: Integer
Returns the number of entries in the child. You can also use this to get the number of elements in an array.

Examples:
c=prefs.root.child("myDictionary").Count – Returns the number of entries in the child "myDictionary"
c=prefs.root.child("myArray").Count – Returns the number of elements in the array.

CurrentKey

Input: Nothing
Output: Nothing
Returns the name of the current key. Use this in conjunction with MoveFirst, MoveLast, and MoveNext.

Exists

Input: key as String
Output: Boolean
Determines if a key exists.

GetBoolean

Input: key as String,[default as Boolean]
Output: Boolean
Returns the Boolean value of the key passed.

GetCheckbox

Input: box as Checkbox
Output: Nothing
Sets a checkbox's value depending on what is in the plist. The key of the value is the name of the checkbox.

GetColor

Input: key as String,[default as Color]
Output: Color
Returns the Color value of the key passed.

GetData

Input: key as String,[default as String]
Output: String
Returns the String value of the key passed.

GetDate

Input: key as String,[default as Date]
Output: Date
Returns the Date value of the key passed.

GetDouble (defunct)

Input: key as String,[default as Double]
Output: Double
Returns the Boolean value of the key passed. Use GetReal instead as this function is only provided for backward compatibility.

GetEditField

Input: field as EditField
Output: Nothing
Populates an EditField's text property with the value in the plist. The key of the value is the name of the EditField.

GetList

Input: key as String,items() as string
Output: Nothing
Populates the array you pass with items from the array in the plist. Please note that, unlike most of the other Get methods, this one does not return a value. Instead, it populates the array that is passed.

GetListbox

Input: list as Listbox, setDefault as Boolean
Output: Nothing
This will populate the list box passed with the items in the plist. If the setDefault parameter is set to true, then the default value stored in the plist will be selected in the list box. See Notes About Listboxes and Popup Menus for more information.

GetPopup

Input: list as PopupMenu, setDefault as Boolean
Output: Nothing
This will populate the popup menu passed with the items in the plist. If the setDefault parameter is set to true, then the default value stored in the plist will be selected in the popup. See Notes About Listboxes and Popup Menus for more information.

GetRadio

Input: radio as RadioButton
Output: Nothing
Sets a RadioButton's value to the value in the plist. The key of the value is the name of the RadioButton.

GetReal

Input: key as String, [default as double]
Output: Double
Returns the Double value of the key passed. Use this instead of GetDouble.

GetInteger

Input: key as String,[default as Integer]
Output: Integer
Returns the Integer value of the key passed.

GetStaticText

Input: txt as StaticText
Output: Nothing
Sets the caption property of the StaticText to the value in the plist. The key of the value is the name of the StaticText field.

GetString

Input: key as String,[default as String]
Output: String
Returns the String value of the key passed.

GetType

Input: key as String
Output: String
Returns the data type of the key. The possible values are:

- Boolean
- Date
- Real
- Integer
- String

- Dict
- Array

GetValue

Input: key as String, [default as String]
Output: String
Returns the value as a string no matter what type it is.

GetWindow

Input: key as String, win as Window
Output: Nothing
Sets a window’s title, position, and size. If the key you pass doesn’t exist, then the properties of the Window object you pass will be used as the default.

Index

Input: index as Integer
Output: plistDict
Used with arrays. Use this for multi-dimensional arrays or if an element in an array contains a dictionary or another array.

Example:
s=prefs.root.child(“myArray”).index(1).GetString(“mySite”) – Returns the value of “mySite” which is located in a dictionary of the first element of the “myArray” array.

Move

Input: key as String,dest as plistDict
Output: Nothing
Moves an element from one dictionary or array to another. After moving the element, the original one is deleted. . If the destination is an array, the entry is appended to it.

Example:
prefs.root.Move(“myElement”,prefs.root.child(“myDict”))

MoveFirst

Input: Nothing
Output: Nothing
Moves to the first item in the child.

MoveLast

Input: Nothing
Output: Nothing
Moves to the last item in the child.

MoveNext

Input: Nothing
Output: Nothing
Moves to the next item in the child. Use CurrentKey to find out what the key name is.

Rename

Input: key as string,newName as string
Output: Nothing
Renames an element to the new name. Any element can be renamed including dictionaries and arrays.

Remove

Input: ToRemove as Variant
Output: Nothing
Removes an element from a dictionary or an array. To remove an element from an array, pass an integer. To remove an element from a dictionary, pass the name as a string.

SetBoolean

Input: key as String, value as Boolean
Output: Nothing
Sets a value as Boolean.

SetCheckbox

Input: box as Checkbox
Output: Nothing
Sets a Boolean with the name of the key being the name of the Checkbox.

SetColor

Input: key as String, value as Color
Output: Nothing
This method converts the data of the color data type to a hex value and stores it as a string.

SetData

Input: key as String, value as Color
Output: Nothing
Sets a data value as a string.

SetDate

Input: key as String, value as Date
Output: Nothing
Sets a date value.

SetDouble (defunct)

Input: key as String, value as Double
Output: Nothing
Sets a double value. Use SetReal instead. This method is provided for backward compatibility.

SetEditField

Input: field as EditField
Output: Nothing
Sets a String with the name of the key being the name of the EditField.

SetInteger

Input: key as String, value as Integer
Output: Nothing
Sets an integer value.

SetList

Input: key as String, items() as String, startIndex as Integer, endIndex as Integer
Output: Nothing
This method is an easy way to create an array from an array. The startIndex and endIndex parameters allow you to control which items in the array will be stored.

SetListbox

Input: list as Listbox
Output: Nothing
This method stores the items in the list box and the selected value, if any. The name of the key is the name of the list box. See Notes About Listboxes and Popup Menus for important information.

SetPopup

Input: list as PopupMenu
Output: Nothing
This method stores the items in the list box and the selected value, if any. The name of the key is the name of the list box See Notes About Listboxes and Popup Menus for important information.

SetRadio

Input: radio as RadioButton
Output: Nothing
Sets a Boolean with the name of the key being the name of the RadioButton.

SetReal

Input: key as String, value as Double

Output: Nothing
Sets a double value. Use this instead of SetDouble.

SetStaticText

Input: txt as StaticText
Output: Nothing
Sets a String with the name of the key being the name of the StaticText field.

SetString

Input: key as String, value as String
Output: Nothing
Sets a string value.

SetWindow

Input: key as String, win as Window
Output: Nothing
This method actually creates another dictionary that contains the window's name, position, and size.

Notes About Listboxes and Popup Menus

You may notice that there is no key parameter for GetListbox, GetPopup, SetListbox, and SetPopup. This is because the key is the name of the list box or the popup menu. Plist can also handle arrays of list boxes and popup menus. If a list box or popup menu is part of an array, its key will have its index tacked onto the end. For example, if you have an array of list popup menus called favorites, The key of the first popup menu would be "favorites.0", the second "favorites.1" and so on.

What Can Cause An Error?

plist is designed to trap and report all sorts of errors. When an error occurs, the plist **Erro** property is set to true and the plist **ErrorMessage** property contains the message. If you wish to be notified via a message box when an error occurs, set the plist **debug** property to true.
To aid you in tracking down error, below is a list of all the possible errors and what can cause them.

Error	**Possible Cause**
Key does not exist	A key you are reference doesn't exist.
Illegal Type	You are trying to do a Get or Set on an element that is actually a dictionary or array.
Type Mismatch	You are trying to Set a different data type than what the element is. For example, if

	you did prefs.root.SetString("date",date) and the "date" element is actually a date, you would get this error.
Could not create date object	The date that was passed could not be parsed into a date format.
Child Exists	A child you are trying to add already exists.
Array Exists	An array you are trying to add already exists.
[key] is not an array	You are trying to append to something that is not an array.
Subscript out of range	You are trying to reference an element in an array that is beyond the actual number of elements.
Source cannot be a dictionary or an array	You are attempting to either **Move** or **Copy** a dictionary or an array.
Source does not exist	The source in **Move** or **Copy** does not exist.
Destination is nil	The destination in **Move** or **Copy** is nil.
Key [key] already exists	You are attempting to **Rename** an element to a key that already exists.

plist Example

Included with the plist are three sample projects.

- The plist Example project shows you how to use some of the methods in plist. It also shows you how to save a list of opened windows and then re-open them when the application is re-launched. When you click on the Save button, the plist will be displayed in the edit field below it.
- The From Scratch Example project shows you how to create a plist from scratch within your code.
- The Safari Search project shows you how to use the Find methods as well as some of the "all-in-one" methods such as SetCheckbox and SetRadio.

Hopefully, this documentation has provided you with enough information to make effective use of the plist and plistDict classes. The classes were tested against the Finder's plist (a rather complex plist). If you have any questions or come across any problems, feel free to send an email to macmage@maccrafters.com