

## **An Approach to Preferences**

By Timothy Aaron Penner, Technical Services Team Member, 4D Inc.

Technical Note 09-24

## Table of Contents

---

Table of Contents .....	2
Abstract .....	3
Introduction.....	3
What Are Preferences? .....	3
Why Use Preferences? .....	3
How Do Preferences Work? .....	4
Explanation of the implementation used here .....	4
XML Structure .....	4
Client/Server Environment Optimizations .....	5
Using the demo database .....	5
Saving and Loading Preferences .....	6
Loading Preferences from Disk into Memory.....	6
Saving Preferences to Disk .....	6
All about Preference Themes .....	6
Creating Preference Themes .....	7
Deleting Preference Themes .....	7
Getting a List of All Preference Theme Names .....	8
Getting, Setting, and Deleting Preferences .....	8
Get All Preferences for a Given Theme Name .....	8
Get All Preference Names for a Given Theme Name .....	8
Get the Value of a Given Preference .....	9
Set the Value of a Given Preference.....	9
Deleting Preferences .....	9
Preferences Viewer GUI .....	10
Displaying the Preferences Viewer GUI.....	10
Using the Preferences GUI.....	11
Adding and Deleting Themes and Preferences .....	12
Usage in a Client/Server Environment.....	12
Conclusion.....	12

## Abstract

---

This Technical Note describes an approach for handling user preferences saved in XML format with theme support. Included with this Technical Note is a component that can be used to store and retrieve preferences, and the source to the component.

## Introduction

---

With the growing complexity of applications and the user configurable options of said applications, one may be asking themselves how to handle storing and retrieving their users' preferences. This Technical Note describes an approach of tackling this task. Included with this Technical Note is a component that can be used to store and retrieve preferences, and the source to the component.

## What Are Preferences?

---

Preferences are runtime parameters that may differ from installation to installation but nonetheless play a role in the operation or appearance of a program. Preferences can be any number of things including the following:

- Color for a particular UI item
- Font face, color, size
- Sort orders (ascending or descending)
- Location of a folder used later in the operation of the program

## Why Use Preferences?

---

Preferences can be used as a way of remembering things. Preferences can also be used a way of setting values that will later be reused for the proper runtime of your application. Perhaps your application requires the user to set a location to be used for images, or a WebFolder location to be used for serving HTML files? All of these are good candidates for using a preferences file.

Using an external file (XML in this case) as opposed to using the internal data engine of 4D gives us a few major advantages. Since it is merely a file located on disk, the preferences are able to be easily copied from one installation to another. They can also be edited by hand using any available text editor.

One may even be able to devise a single database that changes completely based on a few standard preference files; so that simply swapping out the preference file allows the program to behave or appear in a completely different fashion.

## How Do Preferences Work?

---

Preferences work by checking the value of a given preference and then taking further action based upon that value. Typical places for preferences to be stored in are: an external file, internal to the application, or on the web.

Here are a few examples of how preferences could work:

Example 1 – You may have a preference in your database to enable extra warnings or information; perhaps the preference is saved as “VerboseMode”. The developer can check the value of the “VerboseMode” preference and then either show or hide extra information based upon its value.

Example 2 – You may have a method or set of methods in your database that frequently save files or reports. You could have a preference that controls how the save dialog works; perhaps you have a preference saved as “DefaultSavePath” and another as “PromptForSave”. The developer could check those values and then act accordingly. For example, the DefaultSavePath preference could be used in code as the default path displayed for a save dialog, and if the “PromptForSave” is set to False you could skip the prompting all together.

*Note: A default option/action should be coded into the application in case the preference is not found.*

## Explanation of the implementation used here

---

In the sample database included with this Technical Note, preferences are stored in an XML file and loaded into memory using 4D’s built in DOM commands. The preferences are then queried based on element id’s using more of the built in DOM commands. Adding, modifying, and deleting preferences and themes all use the DOM commands. All of the methods in the database are essentially wrappers to these DOM commands.

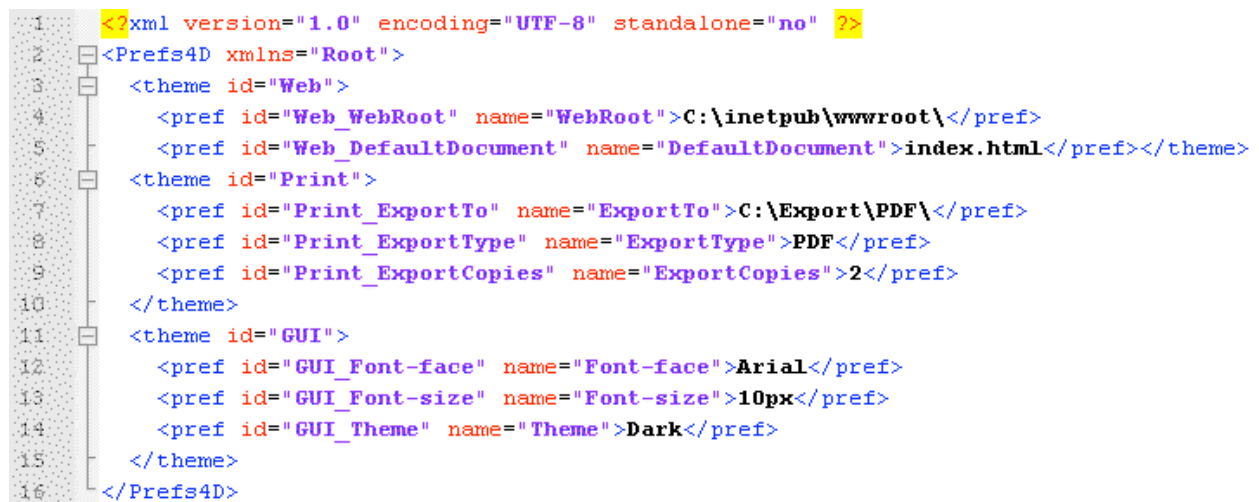
### XML Structure

The structure of the XML document is rather simple:

We start off with a root element named Prefs4D. From there, we have “theme” elements whose theme name is used in the element ID. Each “theme” element can have any number of “pref” child elements; the “pref” elements have two attributes and a value. The “pref” ID attribute is comprised of the theme name and the preference name concatenated together with an underscore. The name attribute is simply the name given for the preference and the value of the element is the value of the preference.

*Note: Using both the theme name and the preference name as the ID for the “pref” element allows you to use the same preference name in multiple themes.*

The following image displays a sample preferences XML file:



```
1 <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
2 <Prefs4D xmlns="Root">
3   <theme id="Web">
4     <pref id="Web_WebRoot" name="WebRoot">C:\inetpub\wwwroot\</pref>
5     <pref id="Web_DefaultDocument" name="DefaultDocument">index.html</pref></theme>
6   <theme id="Print">
7     <pref id="Print_ExportTo" name="ExportTo">C:\Export\PDF\</pref>
8     <pref id="Print_ExportType" name="ExportType">PDF</pref>
9     <pref id="Print_ExportCopies" name="ExportCopies">2</pref>
10  </theme>
11  <theme id="GUI">
12    <pref id="GUI_Font-face" name="Font-face">Arial</pref>
13    <pref id="GUI_Font-size" name="Font-size">10px</pref>
14    <pref id="GUI_Theme" name="Theme">Dark</pref>
15  </theme>
16 </Prefs4D>
```

In the screenshot above we have three themes;

- Web
- Print
- GUI

Each theme has multiple preferences and you can see how the “pref” ID is a concatenation of the theme name and the preference name. For example the **ExportTo** preference in the **Print** theme using a “pref” ID of **Print\_ExportTo**.

## Client/Server Environment Optimizations

The project methods have been set to execute on server (where necessary) using the Execute on Server method attribute; this has no effect in a single-user environment but allows the usage of this approach in both Client/Server as well as single-user environments.

Essentially the preferences are only stored in the memory on the server. When preferences are created, deleted, or queried the memory on the server is accessed.

## Using the demo database

---

The demo database comes in two flavors; a component that can be dropped into your own projects, and the source database to that component. This section covers the syntax and usage of the operations most commonly used in both.

Most of the command parameters are either Pointers or String literals. Pointers are used for return values and String Literals are used for referencing the preference names, themes, and values.

## Saving and Loading Preferences

Before preferences can be used they must be loaded from disk into memory; if changes to the preferences have been made during the runtime of the program they should be saved back to disk. This section discusses how to load the preferences from disk into memory, and also covers how to export the preferences from memory back out to disk.

### Loading Preferences from Disk into Memory

Use the **pref\_LOAD\_prefs\_into\_mem** method to load the preferences from disk into memory. This method requires no parameters:

```
pref_LOAD_prefs_into_mem
```

This attempts to open the preferences xml file located next to the structure and parse it into memory as an interprocess blob named `<>pref_Prefs4D`. The method looks for a file named "**pref\_" + *structure name* + ".xml"** so if your structure is named test.4DB, the preference file would be named pref\_test.4DB.xml. If the file cannot be found it is created automatically.

### Saving Preferences to Disk

Use the **pref\_EXPORT\_dom** method to export the DOM from memory out to a file on disk. This method requires no parameters.

```
pref_EXPORT_dom
```

The above line of code exports the DOM from memory out to a file on disk comprised of "**pref\_" + *structure name* + ".xml"** so if your structure is named test.4DB the preference file would be named pref\_test.4DB.xml

*Note: This method must be issued in order to save the changes from memory out to disk. The only exception to that is that when using the GUI you will be prompted to save upon closing the form (when changes have been made).*

## All about Preference Themes

Themes are a way of grouping preferences together. One advantage to grouping preferences into themes is that it allows you to have the same preference name in multiple different themes. Here are the methods in the Theme set of commands:

- Pref\_CREATE\_theme for creating a preference theme.
- Pref\_DELETE\_theme for deleting a preference theme.
- Pref\_GET\_themes for getting a list of all preference theme names.

Each of these commands is discussed in the following sections.

## Creating Preference Themes

Use the **pref\_CREATE\_theme** method to create a preference theme. This method requires 1 parameter, the name of the theme.

```
pref_CREATE_theme("test theme")
```

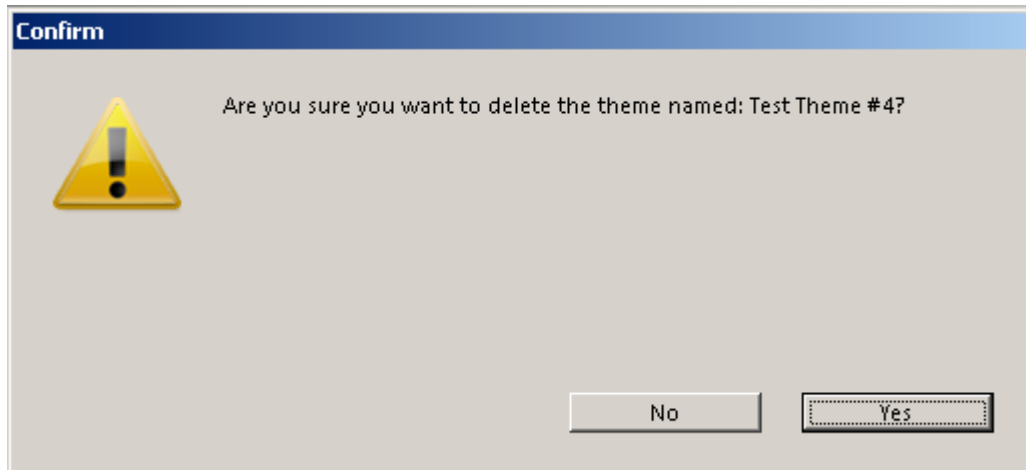
The above snippet of code attempts to create a preferences theme named "test theme". If the theme already exists, a reference to that theme will be returned and no action will be taken.

## Deleting Preference Themes

Use the **pref\_DELETE\_theme** method to delete a theme and all of its preferences. This method requires one parameter but accepts two; the required parameter is the name of the theme to delete. The optional second parameter is a Boolean value that controls whether to skip the warning or not. By default, the user is warned before deleting a theme.

```
pref_DELETE_theme("Test Theme #4")
```

The above snippet of code attempts to delete the theme named "Test Theme #4"; a confirmation dialog is displayed to the user;



If the user confirms, the theme is deleted. If the user cancels, the theme is not deleted. The OK variable is passed back in \$0.

```
pref_DELETE_theme("Test Theme #4";True)
```

The above snippet of code attempts to delete the theme named "Test Theme #4". The second parameter, True, tells the method to skip the warning and delete the theme. The OK variable is passed back in \$0

*Note: Deleting a theme automatically exports the DOM to disk and then reloads it using the `pref_EXPORT_dom` and `pref_LOAD_prefs_into_mem` methods.*

## Getting a List of All Preference Theme Names

Use the **pref\_GET\_themes** method to get a list of all themes found in the preferences file. This method requires one parameter, a pointer to a text array used for holding the theme names.

```
pref_GET_themes (->themes_at)
```

The above snippet of code gets all themes found and places them into the text array named `themes_at`.

## Getting, Setting, and Deleting Preferences

There are multiple commands for getting and setting preferences:

- `Pref_GET_AllPrefs` for getting all preferences (names and values) for a given theme name.
- `Pref_GET_prefNames` to get all preference names from a given theme name.
- `Pref_GET_prefValue` to get the value of the given preference.
- `Pref_SET_pref` to set the value of a preference.
- `Pref_DELETE_pref` to delete a preference.

Each of these commands is discussed in the following sections.

### Get All Preferences for a Given Theme Name

Use the **pref\_GET\_AllPrefs** method to get a matched pair of arrays containing all preference names, and all preference values from the given theme name. This method requires three parameters; the first parameter is the name of the theme as a string, the second parameter is a pointer to an array to hold the preference names, the third parameter is a pointer to an array to hold the preference values.

```
pref_GET_AllPrefs ("Web";->prefs_at;->values_at)
```

The above snippet of code returns all preferences from the theme named "Web"; the matched arrays `prefs_at` and `values_at` are used for the preference names and values respectively.

*Note: The arrays used in this method are matched so if you sort them, make sure to use the **MULTI SORT ARRAY** command in order to keep them synchronized.*

### Get All Preference Names for a Given Theme Name

Use the **pref\_GET\_prefNames** method to get all of the preference names for a given theme name. This method requires two parameters; the first parameter is the theme name as a string, the second parameter is a pointer to a text array to hold all of the preference names.



```
pref_GET_prefNames("Web";->WebPrefs_at)
```

The above snippet of code gets all of the preference names from the theme named "Web" and places them into the text array named WebPrefs\_at.

## Get the Value of a Given Preference

Use the `pref_GET_prefValue` method to get the value of a given preference. This method requires two parameters; the first parameter is the theme name as a string, the second parameter is the preference name as a string. The preference value is returned in \$0

```
myVar_t:=pref_GET_prefValue("Web";"WebRoot")
```

The above snippet of code gets the value of the preference named "WebRoot" from the theme named "Web" and places it in the variable named "myVar\_t".

*Note: The value of the preferences are always returned as text.*

## Set the Value of a Given Preference

Use the `pref_SET_pref` method to set the value of a given preference. This method requires three parameters; the first parameter is the theme name as a string, the second parameter is the preference name as a string, and the third parameter is the value of the preference as a string.

```
pref_SET_pref("Web";"WebRoot";":MacHD:Library:WebServer:Documents:")
```

The above snippet of code sets the value of the preference named "WebRoot" in the theme named "Web" to the value of ":MacHD:Library:WebServer:Documents:".

A Windows version of the above snippet of code may look like this:

```
Pref_SET_pref("Web";"WebRoot";"C:\\inetpub\\wwwroot\\")
```

The above snippet of code sets the value of the preference named "WebRoot" in the theme named "Web" to the value of "C:\\inetpub\\wwwroot\\"

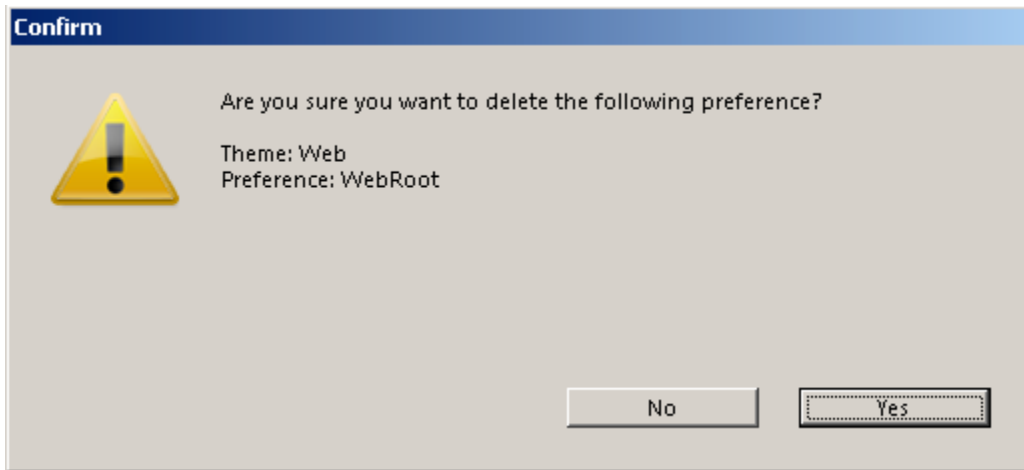
*Note: The values of the preferences are always returned as text.*

## Deleting Preferences

Use the `pref_DELETE_pref` method to delete a preference from the DOM. This method requires two parameters but accepts three. The first parameter is the name of the theme as a string, the second parameter is the name of the preference as a string, and the optional third parameter is a Boolean value controlling whether or not to skip the warning.

```
pref_DELETE_pref("Web";"WebRoot")
```

The above snippet of code attempts to delete the preference named "WebRoot" from the theme named "Web" and a confirmation dialog is displayed to the user;



If the user confirms, the preference is deleted. If the user cancels, the preference is not deleted. The OK variable is passed back in \$0.

## Preferences Viewer GUI

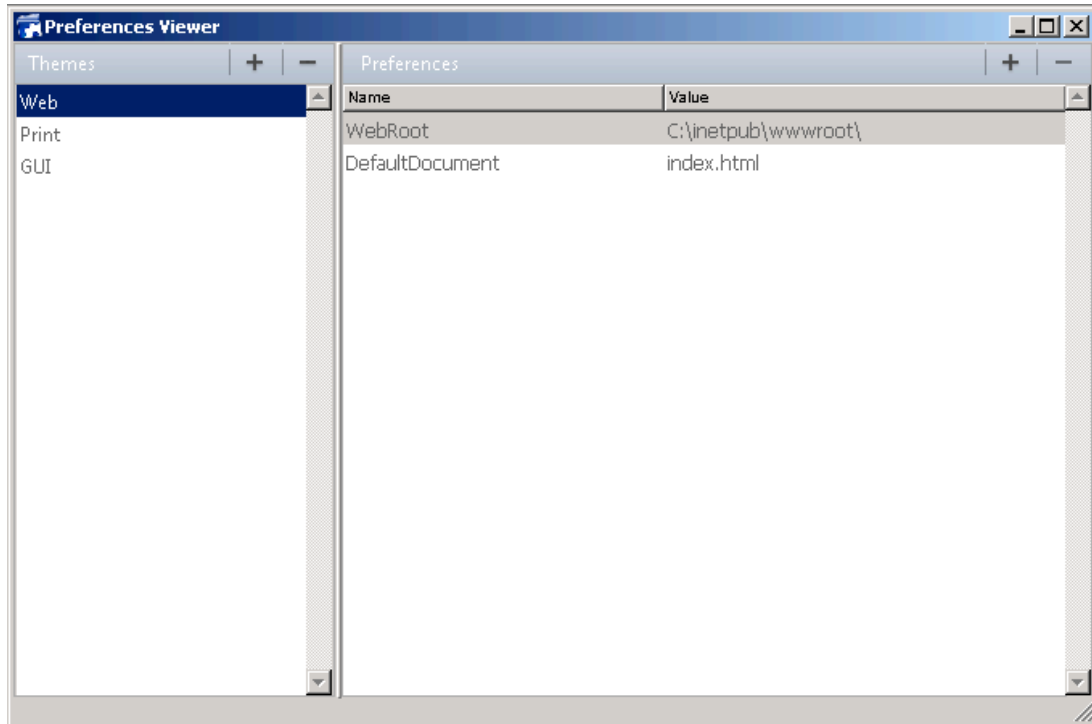
In addition to the methods listed above, there is also a GUI for viewing and modifying the Preferences. This section discusses how to show and use the GUI.

### Displaying the Preferences Viewer GUI

Use the **pref\_UI\_SHOW** method display the preferences viewer. This method requires no parameters.

```
pref_UI_SHOW
```

The above snippet of code displays the preferences viewer:

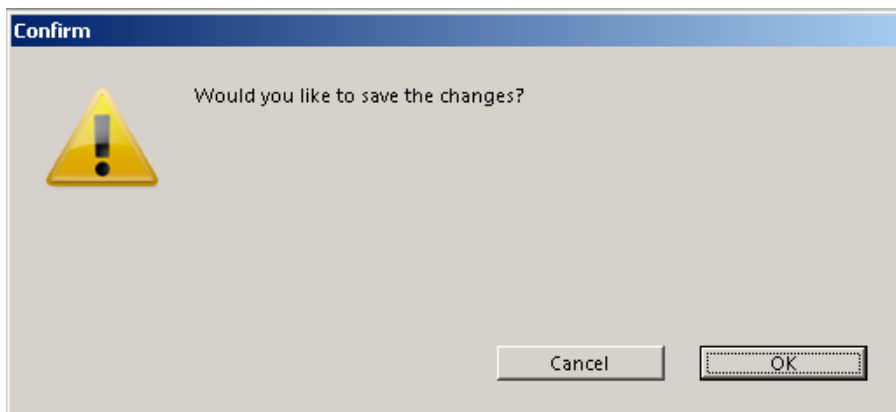


## Using the Preferences GUI

The Preferences Viewer automatically loads the preferences from disk into memory in the form's On Load form event. This can be disabled by commenting out the call to ***pref\_LOAD\_prefs\_into\_mem*** on line 25 in the form method for the pref\_Preferencesviewer form.


*Note: If the automatic loading of the preferences are disabled by commenting out the line of code above, make sure that the preferences are being loaded in the On Startup / On Server Startup database methods.*

Changes in this form are written to the DOM in memory so they are effective right away. However changes are not saved to disk until the form is closed. Upon closing the form the user is prompted with the following confirmation box:



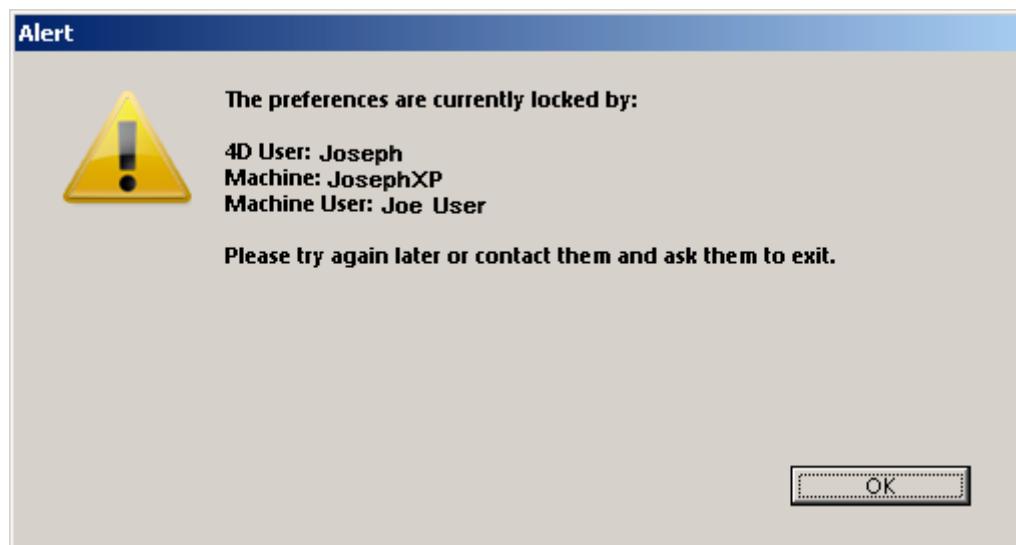
If the user clicks OK, the changes are saved to disk. If the user clicks cancel, the changes are reverted back by reloading the preferences file into memory.

## **Adding and Deleting Themes and Preferences**

Adding and Deleting Preferences and Themes are accomplished by using the  buttons. The states of the buttons are controlled by what is currently selected. If no theme is selected, the add preference button will not be clickable. Conversely if a theme is selected but a preference is not selected, the add preference button will be clickable but the delete preference button will not be clickable.

## **Usage in a Client/Server Environment**

A semaphore is created when the form is displayed, and cleared when the form is unloaded. This is used to prevent multiple users from editing the preferences at the same time. If the preferences are locked by another user, the following dialog is displayed:



Using this information, the user has the ability to contact the person who has the preferences locked in order to address it.

## **Conclusion**

---

This Technical Note described an approach of handling preferences in one's application. A sample database and component were presented that outlined the approach described in this Technical Note. This information should allow the 4D developer to use preferences in their own applications.