Preferences Utilities Reference

Core Foundation

2007-10-31

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Preferences Utilities Reference

Framework:	CoreFoundation/CoreFoundation.h
Companion guide	Preferences Programming Topics for Core Foundation
Declared in	CFPreferences.h

Overview

Core Foundation provides a simple, standard way to manage user (and application) preferences. Core Foundation stores preferences as key-value pairs that are assigned a scope using a combination of user name, application ID, and host (computer) names. This makes it possible to save and retrieve preferences that apply to different classes of users. Core Foundation preferences is useful to all applications that support user preferences. Note that modification of some preferences domains (those not belonging to the "Current User") requires Admin privileges—see *Authorization Services Programming Guide* for information on how to gain suitable privileges.

Unlike some other Core Foundation types, CFPreferences is not toll-free bridged to its corresponding Cocoa Foundation framework class (NSUserDefaults).

Functions by Task

Several functions return a preference value as a Core Foundation property list object. You can use the function `CFGetTypeID` to determine the value's type. For more information about property lists, see *Property List Programming Topics for Core Foundation*.

Getting Preference Values

`CFPreferencesCopyAppValue` (page 9)
Obtains a preference value for the specified key and application.

`CFPreferencesCopyKeyList` (page 10)
Constructs and returns the list of all keys set in the specified domain.

`CFPreferencesCopyMultiple` (page 10)
Returns a dictionary containing preference values for multiple keys.

`CFPreferencesCopyValue` (page 11)
Returns a preference value for a given domain.

`CFPreferencesGetAppBooleanValue` (page 12)
Convenience function that directly obtains a boolean preference value for the specified key.

`CFPreferencesGetAppIntegerValue` (page 13)
Convenience function that directly obtains an integer preference value for the specified key.

Setting Preference Values

`CFPreferencesSetAppValue` (page 14)
Adds, modifies, or removes a preference.

`CFPreferencesSetMultiple` (page 15)
Convenience function that allows you to set and remove multiple preference values.

`CFPreferencesSetValue` (page 15)
Adds, modifies, or removes a preference value for the specified domain.

Synchronizing Preferences

`CFPreferencesAppSynchronize` (page 7)
Writes to permanent storage all pending changes to the preference data for the application, and reads the latest preference data from permanent storage.

`CFPreferencesSynchronize` (page 16)
For the specified domain, writes all pending changes to preference data to permanent storage, and reads latest preference data from permanent storage.

Adding and Removing Suite Preferences

`CFPreferencesAddSuitePreferencesToApp` (page 6)
Adds suite preferences to an application's preference search chain.

`CFPreferencesRemoveSuitePreferencesFromApp` (page 13)
Removes suite preferences from an application's search chain.

Miscellaneous Functions

`CFPreferencesAppValueIsForced` (page 8)
Determines whether or not a given key has been imposed on the user.

`CFPreferencesCopyApplicationList` (page 8)
Constructs and returns the list of all applications that have preferences in the scope of the specified user and host.

Functions

CFPreferencesAddSuitePreferencesToApp

Adds suite preferences to an application's preference search chain.

```
void CFPreferencesAddSuitePreferencesToApp (
   CFStringRef applicationID,
   CFStringRef suiteID
);
```

Parameters

applicationID

The ID of the application to which to add suite preferences, typically `kCFPreferencesCurrentApplication` (page 18). Do not pass `NULL` or `kCFPreferencesAnyApplication` (page 18). Takes the form of a Java package name, `com.foosoft`.

suiteID

The ID of the application suite preferences to add. Takes the form of a Java package name, `com.foosoft`.

Discussion

Suite preferences allow you to maintain a set of preferences that are common to all applications in the suite. When a suite is added to an application's search chain, all of the domains pertaining to that suite are inserted into the chain. Suite preferences are added between the "Current Application" domains and the "Any Application" domains. If you add multiple suite preferences to one application, the order of the suites in the search chain is non-deterministic. You can override a suite preference for a given application by defining the same preference key in the application specific preferences.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFPreferences.h`

CFPreferencesAppSynchronize

Writes to permanent storage all pending changes to the preference data for the application, and reads the latest preference data from permanent storage.

```
Boolean CFPreferencesAppSynchronize (
   CFStringRef applicationID
);
```

Parameters

applicationID

The ID of the application whose preferences to write to storage, typically `kCFPreferencesCurrentApplication` (page 18). Do not pass `NULL` or `kCFPreferencesAnyApplication` (page 18). Takes the form of a Java package name, `com.foosoft`.

Return Value

`true` if synchronization was successful, otherwise `false`.

Discussion

Calling the function `CFPreferencesSetAppValue` (page 14) is not in itself sufficient for storing preferences. The `CFPreferencesAppSynchronize` function writes to permanent storage all pending preference changes for the application. Typically you would call this function after multiple calls to `CFPreferencesSetAppValue` (page 14). Conversely, preference data is cached after it is first read. Changes made externally are not automatically incorporated. The `CFPreferencesAppSynchronize` function reads the latest preferences from permanent storage.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ComboBoxPrefs
DTSCarbonShell
HID Config Save
QTCarbonShell
RecentItems

Declared In
`CFPreferences.h`

CFPreferencesAppValueIsForced

Determines whether or not a given key has been imposed on the user.

```
Boolean CFPreferencesAppValueIsForced (
   CFStringRef key,
   CFStringRef applicationID
);
```

Parameters

key
The key you are querying.

applicationID
The application's ID, typically `kCFPreferencesCurrentApplication` (page 18). Do not pass `NULL` or `kCFPreferencesAnyApplication` (page 18). Takes the form of a Java package name, `com.foosoft`.

Return Value
`true` if value of the key cannot be changed by the user, otherwise `false`.

Discussion
In cases where machines and/or users are under some kind of management, you should use this function to determine whether or not to disable UI elements corresponding to those preference keys.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`CFPreferences.h`

CFPreferencesCopyApplicationList

Constructs and returns the list of all applications that have preferences in the scope of the specified user and host.

```
CFArrayRef CFPreferencesCopyApplicationList (
   CFStringRef userName,
   CFStringRef hostName
);
```

Parameters

userName

`kCFPreferencesCurrentUser` (page 18) to search the current-user domain, otherwise `kCFPreferencesAnyUser` (page 18) to search the any-user domain.

hostName

`kCFPreferencesCurrentHost` (page 18) to search the current-host domain, otherwise `kCFPreferencesAnyHost` (page 18) to search the any-host domain.

Return Value
The list of application IDs. Ownership follows the Create Rule.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CFPrefsDumper

Declared In
`CFPreferences.h`

CFPreferencesCopyAppValue

Obtains a preference value for the specified key and application.

```
CFPropertyListRef CFPreferencesCopyAppValue (
   CFStringRef key,
   CFStringRef applicationID
);
```

Parameters

key

The preference key whose value to obtain.

applicationID

The identifier of the application whose preferences to search, typically `kCFPreferencesCurrentApplication` (page 18). Do not pass `NULL` or `kCFPreferencesAnyApplication` (page 18). Takes the form of a Java package name, `com.foosoft`.

Return Value
The preference data for the specified key and application. If no value was located, returns `NULL`. Ownership follows the Create Rule.

Discussion
Note that values returned from this function are immutable, even if you have recently set the value using a mutable object.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CFPreferences
DockBrowser
GrabBag
QTCarbonShell
RecentItems

Declared In
`CFPreferences.h`

CFPreferencesCopyKeyList

Constructs and returns the list of all keys set in the specified domain.

```
CFArrayRef CFPreferencesCopyKeyList (
   CFStringRef applicationID,
   CFStringRef userName,
   CFStringRef hostName
);
```

Parameters

applicationID
The ID of the application whose preferences to search. Takes the form of a Java package name, `com.foosoft`.

userName
`kCFPreferencesCurrentUser` (page 18) to search the current-user domain, otherwise `kCFPreferencesAnyUser` (page 18) to search the any-user domain.

hostName
`kCFPreferencesCurrentHost` (page 18) to search the current-host domain, otherwise `kCFPreferencesAnyHost` (page 18) to search the any-host domain.

Return Value
The list of keys. Ownership follows the Create Rule.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CFPreferences.h`

CFPreferencesCopyMultiple

Returns a dictionary containing preference values for multiple keys.

```
CFDictionaryRef CFPreferencesCopyMultiple (
   CFArrayRef keysToFetch,
   CFStringRef applicationID,
   CFStringRef userName,
   CFStringRef hostName
);
```

Parameters

keysToFetch

An array of preference keys the values of which to obtain.

applicationID

The ID of the application whose preferences are searched. Takes the form of a Java package name, such as `com.foosoft`.

userName

`kCFPreferencesCurrentUser` (page 18) to search the current-user domain, otherwise `kCFPreferencesAnyUser` (page 18) to search the any-user domain.

hostName

`kCFPreferencesCurrentHost` (page 18) to search the current-host domain, otherwise `kCFPreferencesAnyHost` (page 18) to search the any-host domain.

Return Value

A dictionary containing the preference values for the keys specified by *keysToFetch* for the specified domain. If no values were located, returns an empty dictionary. Ownership follows the Create Rule.

Discussion

Note that values returned from this function are immutable, even if you have recently set the value using a mutable object.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CFPrefsDumper

Declared In

`CFPreferences.h`

CFPreferencesCopyValue

Returns a preference value for a given domain.

```
CFPropertyListRef CFPreferencesCopyValue (
   CFStringRef key,
   CFStringRef applicationID,
   CFStringRef userName,
   CFStringRef hostName
);
```

Parameters

key

Preferences key for the value to obtain.

applicationID

The ID of the application whose preferences are searched. Takes the form of a Java package name, such as `com.foosoft`.

userName

`kCFPreferencesCurrentUser` (page 18) if to search the current-user domain, otherwise `kCFPreferencesAnyUser` (page 18) to search the any-user domain.

hostName

`kCFPreferencesCurrentHost` (page 18) if to search the current-host domain, otherwise `kCFPreferencesAnyHost` (page 18) to search the any-host domain.

Return Value

The preference data for the specified domain. If the no value was located, returns `NULL`. Ownership follows the Create Rule.

Discussion

This function is the primitive get mechanism for the higher level preference function `CFPreferencesCopyAppValue` (page 9) Unlike the high-level function, `CFPreferencesCopyValue` (page 11) searches only the exact domain specified. Do not use this function directly unless you have a need. All arguments must be non-`NULL`. Do not use arbitrary user and host names, instead pass the pre-defined domain qualifier constants.

Note that values returned from this function are immutable, even if you have recently set the value using a mutable object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFPreferences.h`

CFPreferencesGetAppBooleanValue

Convenience function that directly obtains a boolean preference value for the specified key.

```
Boolean CFPreferencesGetAppBooleanValue (
   CFStringRef key,
   CFStringRef applicationID,
   Boolean *keyExistsAndHasValidFormat
);
```

Parameters

key

The preference key whose value to obtain. The key must specify a preference whose value is of type `Boolean`.

applicationID

The identifier of the application whose preferences are searched, typically `kCFPreferencesCurrentApplication` (page 18). Do not pass `NULL` or `kCFPreferencesAnyApplication` (page 18). Takes the form of a Java package name, such as `com.foosoft`.

keyExistsAndHasValidFormat

On return, `true` if the preference value for the specified key was located and found to be of type `Boolean`, otherwise `false`.

Return Value
The preference data for the specified key and application, or if no value was located, `false`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DTSCarbonShell
QTCarbonShell
RecentItems

Declared In
`CFPreferences.h`

CFPreferencesGetAppIntegerValue

Convenience function that directly obtains an integer preference value for the specified key.

```
CFIndex CFPreferencesGetAppIntegerValue (
   CFStringRef key,
   CFStringRef applicationID,
   Boolean *keyExistsAndHasValidFormat
);
```

Parameters

key
The preference key whose value you wish to obtain. The key must specify a preference whose value is of type `int`.

applicationID
The identifier of the application whose preferences you wish to search, typically `kCFPreferencesCurrentApplication` (page 18). Do not pass `NULL` or `kCFPreferencesAnyApplication` (page 18). Takes the form of a Java package name, `com.foosoft`.

keyExistsAndHasValidFormat
On return, indicates whether the preference value for the specified key was located and found to be of type `int`.

Return Value
The preference data for the specified key and application. If no value was located, `0` is returned.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CFPreferences.h`

CFPreferencesRemoveSuitePreferencesFromApp

Removes suite preferences from an application's search chain.

```
void CFPreferencesRemoveSuitePreferencesFromApp (
   CFStringRef applicationID,
   CFStringRef suiteID
);
```

Parameters

applicationID

The ID of the application from which to remove suite preferences, typically `kCFPreferencesCurrentApplication` (page 18). Do not pass `NULL` or `kCFPreferencesAnyApplication` (page 18). Takes the form of a Java package name, `com.foosoft`.

suiteID

The ID of the application suite preferences to remove. Takes the form of a Java package name, `com.foosoft`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CFPreferences.h`

CFPreferencesSetAppValue

Adds, modifies, or removes a preference.

```
void CFPreferencesSetAppValue (
   CFStringRef key,
   CFPropertyListRef value,
   CFStringRef applicationID
);
```

Parameters

key

The preference key whose value you wish to set.

value

The value to set for the specified *key* and application. Pass `NULL` to remove the specified key from the application's preferences.

applicationID

The ID of the application whose preferences you wish to create or modify, typically `kCFPreferencesCurrentApplication` (page 18). Do not pass `NULL` or `kCFPreferencesAnyApplication` (page 18). Takes the form of a Java package name, `com.foosoft`.

Discussion
New preference values are stored in the standard application preference location, `~/Library/Preferences/`. When called with `kCFPreferencesCurrentApplication` (page 18), modifications are performed in the preference domain "Current User, Current Application, Any Host." If you need to create preferences in some other domain, use the low-level function `CFPreferencesSetValue` (page 15).

You must call the `CFPreferencesAppSynchronize` (page 7) function in order for your changes to be saved to permanent storage.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DockBrowser
DTSCarbonShell
GrabBag
QTCarbonShell
RecentItems

Declared In
`CFPreferences.h`

CFPreferencesSetMultiple

Convenience function that allows you to set and remove multiple preference values.

```
void CFPreferencesSetMultiple (
   CFDictionaryRef keysToSet,
   CFArrayRef keysToRemove,
   CFStringRef applicationID,
   CFStringRef userName,
   CFStringRef hostName
);
```

Parameters

keysToSet
A dictionary containing the key/value pairs for the preferences to set.

keysToRemove
An array containing a list of keys to remove.

applicationID
The ID of the application whose preferences you wish to modify. Takes the form of a Java package name, `com.foosoft`.

userName
`kCFPreferencesCurrentUser` (page 18) to modify the current user's preferences, otherwise `kCFPreferencesAnyUser` (page 18) to modify the preferences of all users.

hostName
`kCFPreferencesCurrentHost` (page 18) to modify the preferences of the current host, otherwise `kCFPreferencesAnyHost` (page 18) to modify the preferences of all hosts.

Discussion
Behavior is undefined if a key is in both *keysToSet* and *keysToRemove*

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CFPreferences.h`

CFPreferencesSetValue

Adds, modifies, or removes a preference value for the specified domain.

```
void CFPreferencesSetValue (
   CFStringRef key,
   CFPropertyListRef value,
   CFStringRef applicationID,
   CFStringRef userName,
   CFStringRef hostName
);
```

Parameters

key

Preferences key for the value you wish to set.

value

The value to set for *key* and application. Pass `NULL` to remove *key* from the domain.

applicationID

The ID of the application whose preferences you wish to modify. Takes the form of a Java package name, `com.foosoft`.

userName

`kCFPreferencesCurrentUser` (page 18) to modify the current user's preferences, otherwise `kCFPreferencesAnyUser` (page 18) to modify the preferences of all users.

hostName

`kCFPreferencesCurrentHost` (page 18) to modify the preferences of the current host, otherwise `kCFPreferencesAnyHost` (page 18) to modify the preferences of all hosts.

Discussion

This function is the primitive set mechanism for the higher level preference function `CFPreferencesSetAppValue` (page 14). Only the exact domain specified is modified. Do not use this function directly unless you have a specific need. All arguments except `value` must be non-`NULL`. Do not use arbitrary user and host names, instead pass the pre-defined constants.

You must call the `CFPreferencesSynchronize` (page 16) function in order for your changes to be saved to permanent storage. Note that you can only save preferences for "Any User" if you have Admin privileges.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CFPrefTopScores

Declared In

`CFPreferences.h`

CFPreferencesSynchronize

For the specified domain, writes all pending changes to preference data to permanent storage, and reads latest preference data from permanent storage.

```
Boolean CFPreferencesSynchronize (
   CFStringRef applicationID,
   CFStringRef userName,
   CFStringRef hostName
);
```

Parameters

applicationID

The ID of the application whose preferences you wish to modify. Takes the form of a Java package name, `com.foosoft`.

userName

`kCFPreferencesCurrentUser` (page 18) to modify the current user's preferences, otherwise `kCFPreferencesAnyUser` (page 18) to modify the preferences of all users.

hostName

`kCFPreferencesCurrentHost` (page 18) to search the current-host domain, otherwise `kCFPreferencesAnyHost` (page 18) to search the any-host domain.

Return Value

`true` if synchronization was successful, `false` if an error occurred.

Discussion

This function is the primitive synchronize mechanism for the higher level preference function `CFPreferencesAppSynchronize` (page 7); it writes updated preferences to permanent storage, and reads the latest preferences from permanent storage. Only the exact domain specified is modified. Note that to modify "Any User" preferences requires Admin privileges—see *Authorization Services Programming Guide*.

Do not use this function directly unless you have a specific need. All arguments must be non- `NULL`. Do not use arbitrary user and host names, instead pass the pre-defined constants.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CFPrefTopScores

Declared In

`CFPreferences.h`

Constants

Application, Host, and User Keys

Keys used to specify the common preference domains.

```
const CFStringRef kCFPreferencesAnyApplication;
const CFStringRef kCFPreferencesAnyHost;
const CFStringRef kCFPreferencesAnyUser;
const CFStringRef kCFPreferencesCurrentApplication;
const CFStringRef kCFPreferencesCurrentHost;
const CFStringRef kCFPreferencesCurrentUser;
```

Constants

`kCFPreferencesAnyApplication`

Indicates a preference that applies to any application.

Available in Mac OS X v10.0 and later.

Declared in `CFPreferences.h`.

`kCFPreferencesAnyHost`

Indicates a preference that applies to any host.

This domain is currently unsupported.

Available in Mac OS X v10.0 and later.

Declared in `CFPreferences.h`.

`kCFPreferencesAnyUser`

Indicates a preference that applies to any user.

This domain is currently unsupported.

Available in Mac OS X v10.0 and later.

Declared in `CFPreferences.h`.

`kCFPreferencesCurrentApplication`

Indicates a preference that applies only to the current application.

Available in Mac OS X v10.0 and later.

Declared in `CFPreferences.h`.

`kCFPreferencesCurrentHost`

Indicates a preference that applies only to the current host.

Available in Mac OS X v10.0 and later.

Declared in `CFPreferences.h`.

`kCFPreferencesCurrentUser`

Indicates a preference that applies only to the current user.

Available in Mac OS X v10.0 and later.

Declared in `CFPreferences.h`.

Document Revision History

This table describes the changes to *Preferences Utilities Reference*.

Date	Notes
2007-10-31	Noted unsupported domains.
2007-02-21	Corrected description of return value for CFPreferencesCopyMultiple.
2005-12-06	Made minor text changes to conform to reference consistency guidelines.
2005-08-11	Cosmetic changes to conform to documentation guidelines.
2004-08-31	Added note about immutability of values returned from preferences.
	Added note about need for Admin privileges to modify “Any User” preferences.
	Corrected function result description for `CFPreferencesAppValueIsForced` (page 8).
2003-11-11	Clarification of use of `CFPreferencesSynchronize` (page 16).
2003-01-01	First version of this document.

Document Revision History

Index

A

C

K