Warning: This object is only partially tested as of this release.

MiscRegistration

Inherits From: Object

Declared In: <misckit/MiscRegistration.h>

Class Description

This object handles the user interface for a simple registration scheme. It brings up a modal panel to handle registrations and handles keeping track of the license key entered by the user. Any object in the application may query the MiscRegistration object to see if the application is correctly registered by sending a ±**registered** message. The programmer should override the ±**keyOK** method to insert a license key checking algorithm into the object; the default implementation is to accept any registration key given to the MiscRegistration object.

The license keys entered by users are stored in two places. First, an attempt is made to store the key in the file ªkeyº inside the app wrapper. This will probably only work for the user installing the application, possibly only for root on a network installation. Second, the key is stored in the user's defaults database. This allows the key to be remembered even if the key file cannot be written. Also, if the user updates the application and in the process deletes an old key file, this allows a new key file to be automatically generated for the application the next time the user runs the application. (They have to run

the application after installation for this to happen, though.)

To create a proper interface for this object, you should create a panel with two TextFields in it, one for the user to type in the registration key (the *regNumText* instance variable) and the other an uneditable field where the serial number of the application may be displayed (connected to the *regText* instance variable). The *registerPanel* instance variable should be connected to the panel itself. The panel should have two buttons, one to abort the registration process (tied to the ± **cancelRegistration:** method) and another, with a return arrow, to complete the registration (tied to the ±**registerApp:** method). That is all you need to do, and the MiscKit comples complete with a pre-connected panel in the Register.nib sample interface.

The logic behind this object is to provide a simple framework for a rudimentary registration scheme, but to allow the developer to implement the actual encoding in their own unique way. This saves the time required to implement the interface, but does not compromise the security of the registration keys or the exportability of this object from the USA.

Several of the strings used in the Registration... Panel user interface are stored in an NXStringTable. If an NXStringTable is not connected up in the ªRegister.nibº file, then the controller's string table is used instead. (The controller is a MiscInfoController which is using this MiscRegistration instance.) The table is expected to contain the following keys:

NotRegistered	Đ	ªNot registered.º This is the license key written to the key file of a non-registered app.
Unreg	Đ	ªThis Copy Is Unregisteredº to be placed in the serial number text fields of an unregistered application, in white letters.
Reg	Đ	ªCopy #%s is registeredº to be placed in the serial number text fields; the serial number is inserted wherever you place the ª%sº. This text appears in dark gray.
OK	Đ	ªOKº for alert panel buttons.
CantReg	Đ	Text for an alert panel saying that the registration key entered could not be saved to a file inside the app wrapper. (You may wish to note that the key *is* still saved in the user's default database, so this is only a problem for multi-user installations.)
BadRegNum	Đ	Text for an alert panel saying that the registration key entered was invalid.

Instance Variables

id **strings**;
id **controller**;
id **registerPanel**;
id **regText**;
id **regNumText**;

strings	NXStringTable with the strings used in the Registration... panel interface.
controller	The MiscInfoController which is using this object for handling registrations.
registerPanel	The Register... panel where the user types in a registration number/key.
regText	The TextField in the Register... panel showing the current registered serial number, if applicable.
regNumText	The TextField in the Register... panel where the user types the actual registration number/key.

Method Types

Initialization	- init + initialize - setController:
Target/Action Methods	- cancelRegistration: - registerApp: - registration:

Obtaining and Displaying Serial Numbers	- fillRegistrationText: - serialNumber
Handling Registration Keys	- keyOK - readKey - readKeyFromFile - registerPanel - registered - registrationKey - writeKey - writeKeyToFile

Class Methods

initialize
+ **initialize**

Initialize the MiscRegistration class. This sets up a defaults vector for the default registration key, in case a registration key doesn't actually exist yet for the application. Returns **self**.

Instance Methods

cancelRegistration:
- **cancelRegistration:***sender*

Cancels the modal session for the Registration... panel and removes it from the screen. Returns **self**.

See also: **±registerApp:** and ±**registration:**

fillRegistrationText:

- **fillRegistrationText:***textField*

Fills in the text in *textField* with the serial number of the application. If the application is registered, the text's color will be dark gray. If the application is not registered yet, the text will be white. Returns **self**.

init

- **init**

Initializes this instance of the MiscRegistration class and reads in the registration key, checking both the ªkeyº file and the user's defaults database. Returns **self**.

keyOK

- (BOOL)**keyOK**

This method should be overridden to analyze the registration key and return either a YES (if the key is valid) or a NO (if the key is not valid). If the key contains a serial number encoded in it (and it should), you should then set the recognized serial number by setting the string value of the *_serialNum* instance variable (a MiscString instance). The license key entered by the user should be obtained with the ± **registrationKey** method. If you wish to determine if the application is properly registered, you should call the ±**registered** method and *not* ±**keyOK**. The ±**keyOK** method is meant only to be overridden by the programmer, not called.

See also: ±**registered** and ±**registrationKey**

readKey

- **readKey**

Reads in the current registration key. If reading the key in from the file ªkeyº inside the app wrapper fails, then the key is loaded from the user's defaults database, if it exists. Returns **self**.

See also: ±readKeyFromFile

readKeyFromFile
- **readKeyFromFile**

Reads the registration key in from the file ªkeyº inside the app wrapper. Returns **self** if successful or *nil* otherwise.

See also: ±readKey

registerApp:
- **registerApp:***sender*

Called to terminate the modal session for the Register... panel and register the application with the newly entered key, if any. Returns **self**.

See also: ±cancelRegistration:, ±**registration:** and ±**writeKey**

registerPanel
- **registerPanel**

Returns the Register... panel, loading it first if necessary. The panel should be in the file ªRegister.nibº in the app wrapper inside an appropriate .lproj directory.

See also: ±registration:

registered

- (BOOL)**registered**

Returns YES is the application has been successfully registered and NO if not. This method can be overridden if the ±**keyOK** mechanism is not enough. For example, although ±**keyOK** checks for validity of a key, this method might check to see if there are already too many copies of the application running on the network using the same key.

See also: ±**keyOK** and ±**registrationKey**

registration:

- **registration:***sender*

Loads, if necessary, the Register... panel and begins a modal session for it. Returns **self**.

See also: ±**registerPanel**

registrationKey

- (const char *)**registrationKey**

Returns a pointer to the string entered by the user as a registration key.

See also: ±**keyOK**, ±**registered**, and ±**serialNumber**

serialNumber

- (const char *)**serialNumber**

Returns a pointer to the string entered by the user as a registration key.

See also: **±keyOK** and ± **registrationKey**

setController:

- **setController:***sender*

Sets the controller, a MiscInfoController instance. This is usually done when the object is created by the MiscInfoController.

writeKey

- **writeKey**

Writes the current license key to the user's defaults database, in the app's name, and also to the file ªkeyº inside the app wrapper. The copy in the defaults database will allow the license key to be preserved if the user updates the application to a new version and blows away the ªkeyº file indavertently in the upgrade. This method is called automatically when the Register... panel is given a new license key. Returns **self**.

See also: **±writeKeyToFile**

writeKeyToFile

- **writeKeyToFile**

Writes the license key to the file ªkeyº inside the application's app wrapper. Returns **self**. If unable to write the key file, an alert panel warning the user of this situation is presented and *nil* is returned.

See also: **±writeKey**