Q: I've read the release notes about the NetInfoKit (NIKit). Is there additional information about these objects and how to use them?

A: The only documentation that is currently available are the on-line Release Notes on 2.0 and whatever you can glean from the .h files. As you saw from those notes, it provides you with a rough overview of the objects available in the NIKit, but doesn't provide you with any reference material. Through this document, we hope to shed some light on the NIKit objects, how to use them, and for what purpose.

Feedback for these objects would be greatly appreciated by the folks in engineering. Please send your feedback to bug_next@next.com.

NIDomain object
The first object, the NIDomain object, was created so that the other objects could have something through which to communicate to the NIKit database. The panel objects (like NIOpenPanel) use NIDomain to set and retrieve data from the NetInfo database. This object is the most likely one to change in future releases. You probably won't use NIDomain directly in your application, since the panel objects provide a higher level interface to it.

NIDomainPanel Object
The NIDomainPanel object is essentially a browser. It browses through the domain hierarchy, and displays it (much like the file system browser). To use the NIDomainPanel you should follow these steps:

- **new**
  returns an id for the new instance of the NIDomainPanel.
- **runModal**
  this runs a modal loop which exits when the user clicks ok or cancel
- **exitFlags**
  returns whether the user clicked ok or cancel
- **domain**
  returns the name of the domain that the user has selected.

The other methods are primarily there in case you wish to override them and provide some special functionality. Most useful are **cancel**, **ok** and **domain**.

NIOpenPanel Object
The next object, NIOpenPanel is a sub class of NIDomainPanel. It is used to display and select a particular directory from a list of its peers inside of any NetInfo domain in the hierarchy. It's analogous to the OpenPanel for filesystems. NIOpenPanel consists of the domain browser (which it inherits from the NIDomainPanel), buttons and a scrolling list of directories. You create an NIOpenPanel the same way in which you create a NIDomainPanel: using **new** and **runModal**. The other really interesting methods in NIOpenPanel are the **directory** and **setDirectoryPath** methods which you can use to programmatically browse to the directory path that you want from inside of the domains, or ask the NIOpenPanel where it currently is browsed to.

NISavePanel Object
NISavePanel is a subclass of NIOpenPanel, and is used to assist in saving changes to an open

directory in a particular domain. NISavePanel is very much like NIOpenPanel, but with some additional functionality. It provides three different **runModal** routines. **runModal** is the one which you will use most frequently. It will start such that the browser is displaying your current domain. **runModalWithString** runs a modal loop, and starts up such that the browser is displaying the domain which is passed in through "initialValue". **runModalwithUneditableString** is the same except for the fact that the directory is uneditable. It is used this way inside the NetManager and UserManager where the name of the directory is a fundamental property of the machine or user being edited and cannot be changed on the fly while performing a Save To.

NILoginPanel Object

And finally the NILoginPanel. This is probably the object for which you will have the most use. The login panel is used to verify (authenticate) a user's identification. You can either rely on the user id and password that are stored in the NI database, or you can use your own. There are three **runModal** methods in this object as well:

- (BOOL)runModal:sender inDomain:(void *) domainID;

This runs the login dialog modally with the default user "root".

- (BOOL)runModal:sender withUser:(char *)userName inDomain:(void *)domainID withString:(const char *)whatWarning allowChange:(BOOL)disableUser;

This runs the login dialog modally with the user as specified. You can change the message displayed on the panel by using the withString argument. Also, this method allows the user to

edit the default user name.

- (BOOL)runModalwithValidation:sender withUser:(const char *)userName inDomain:(void *) domainID withString:(const char *)whatWarning allowChange:(BOOL)enableUser;

This runs the login dialog modally as the previous method. However, this one requires validation.

With this modal method you can provide your own password validation scheme. You use this

in conjunction with the NILoginPanelDelegate's **authenticateUser** method.

At this point there is no example code from which you can learn to use these objects wisely (or at all). For examples about how these NIKit objects look and for an idea of how they can be put to good use, you can look at the system administration tools in /NextAdmin. Most of them use the NIKit. Future changes in the NIKit will be made to better support these applications.

QA697

Valid for 2.0, 3.0