

```
//
// FILENAME: Router.m
// SUMMARY: Implementation of Router, which adds various services to NXApp
// SUPERCLASS: Application(Router)
// PROTOCOLS: <MailListener>
// INTERFACE: None
// AUTHOR: Rohit Khare
// COPYRIGHT: ©1993,94 California Institute of Technology, eText Project
//
// Implementation Comments
// The savePanel accessory and ID routines are interesting.
//
// HISTORY
// 10/22/94: Added GLOBAL/Mailer support as per Greg Anderson's post.
// 09/27/94: Revamped for eText5
// 07/06/94: Added kernel-wide mail-messaging support.
// 01/10/94: Added sharedText
// 01/05/94: Created.
//
// Imported Interfaces
//
// #import "Router.h"

@implementation Application (Router)
//
// Public Services
//
- (long) uniqueID {
    static BOOL seeded=NO;

    // Strategy: We have 24 sigbits of epoch in the one year.
    // Epoch-count is not unique, since the odds of epoch-collision is
    // a function of users and activity rate. By shoehorning into a 32bit
    // int, we are forced to support, at best, 2^8 transactions per second,
    // where a transaction is any call to uniqueID anywhere in the world!
    // Thus, we must migrate to 64bits...
    // The pattern is:
    // |....epoch(32)....|...pid(8) |...counter(8)...|...millisec(16)...|

    // However, this scheme is foolish, since it wastes an extreme number of
    // bits for the sake of monotonicity. Instead, we will rely on BSD's random
    // call, for which we offer a large amount of custom state, namely, the
    // user's encrypted login password, pid, and millisecond.

    if (!seeded) {
        char state[17];
        struct timeval t;
```

```

    strncpy(state+2, NXUserName(), 14);
    gettimeofday(&t, NULL);
    state[0] = t.tv_usec & MAXINT;
    state[16] = 0;
    initstate(t.tv_sec, crypt(state, "Rk")+2, 8);
    seeded = YES;
}
return random();
}
- (const char *) versionStr
{return (const char *)NXUniqueString("0.95");}

- (const char *)date {
    static char        buffer[64];
    struct tm          *timeptr;
    time_t              timer;

    time(&timer);
    timeptr = localtime(&timer);
    strftime(buffer, sizeof(buffer), "%c", timeptr);
    return buffer;
}
- sendMailTo:(const char *)to cc:(const char *)cc subject:(const char *)subject
    body:(const char *)body deliver:(BOOL)deliver {
    Speaker            *speaker;
    port_t              portSend = PORT_NULL;
    int                 window = 0;
    BOOL                success = NO;

    // Stock a new Compose window in Mail.app and optionally deliver it.
    // Any char* args may be NULL.
    // By Hugh Secker-Walker, based on code by Simson Garfinkel.
    if ((speaker = [[Speaker alloc] init]) == nil) return nil;
    // make sure Mail.app is launched
    portSend = NXPortFromName("MailSendDemo", 0);
    if (portSend==PORT_NULL) {
        const char *mailer = NXGetDefaultValue("GLOBAL", "Mailer");
        port_t portmail = NXPortFromName(mailer ? mailer : "Mail", 0);
        port_deallocate(task_self(), portmail);
    }
    portSend = NXPortFromName("MailSendDemo", 0);
    if (portSend==PORT_NULL) return nil;
    // make speaker use the port
    [speaker setSendPort:portSend];
    // open a new compose window in Mail, get window number
    if ([speaker selectorRPC:"openSend:" paramTypes:"I", &window])

```


@end

/*

Yes there is a simple answer, and I'll volunteer the scant lines of code required to do it properly. The following code snippet looks to see if any running app is offering a "MailSendDemo" port. If not, it looks at the user's defaults database for an app designated as the preferred mailer. (You can specify that by typing "dwrite GLOBAL Mailer MyPreferredMailer" in a Terminal window.) Failing that, it drops back to Mail.app.

*/