

```
/*
*
-----
* main.c
* CocoaMail XFcn
*
* Created by Bruce Martin on 4/21/06.
* Copyright Martin Solution 2006. All rights reserved.
*
-----
*/
// load\ external\ \"$(PROJECT_DIR)/$(BUILD_DIR)/$(CONFIGURATION)/$(PROJECT_NAME).
bundle\"

// alert\ $(PROJECT_NAME)\(\"?\")\ explain\ $(PROJECT_NAME)\(\"!\")

/*
*
-----
* Includes
*
-----
*/

#include "commonExt_460.h"
#include "InternalsToolbox_460.h"
#import <Message/NSMailDelivery.h>
#import <Cocoa/Cocoa.h>
#import <AppKit/NSAttributedString.h>
#import <Foundation/NSDictionary.h>
/*
*
-----
* Defines
*
-----
*/

#define kMinNumParams 4 // Fill in your minimum number of parameters
here
#define kMaxNumParams 7 // Fill in your maximum number of parameters
here
```

```
#define kVersionNumber "1.0" // Fill in your current version number here
#define kSyntaxMsg
    "CocoaMail(<theCmd>,<theMessage>,<theSubject>,<to>,[<from>][theHeaders][,<theAttachm
entFilePath>])\n theCmd: This is one of the following commands; \t text This command wil
l send a plain text email without attachment.\n \t textWithAttach As it says this is a
text email with an attachment.\n \t HTML This is an HTML email. \n \t HTMLWithAttach An
HTML email with an attachment."
#define kCopyrightMsg "CocoaMail XFcن by Bruce Martin \2512006 Martin Solution"

/*
*
-----
* Prototypes
*
-----
*/

int main (XCmdPtr);
void Fail (char *);
Boolean FalseAlarm (void);
void SetResult (char *);
void doMyFunction (void);
void CFSetResult (CFStringRef message);
CFStringRef ParamToCFString (Handle theParam);
CFStringRef simpleSendASCIIEmail (CFStringRef theEmailDest,CFStringRef
theSubject,CFStringRef theBody);
CFStringRef sendEmail (NSAttributedString *theMsg,NSDictionary
*messageHeaders,NSString *messageFormat,NSString *deliveryProtocol);
void fillDict (NSMutableDictionary *theDict,Handle theParam);
char getItemDelimiter (void);
char getLineDelimiter (void);

/*
*
-----
* Static and Global Variables
*
-----
*/

XCmdPtr par; // Public pointer to parameter block
Boolean fatalError; // Had any fatal errors yet?
```

```
/*
 *
-----
 * Script Entry Point
 *
-----
 */

int main(XCmdPtr xcmdPtr)
{
    par = xcmdPtr;
    fatalError = false;
    par->returnValue = 0;

    if (!FalseAlarm())                // Deal with kSyntaxMsg, author, & bad
paramCounts
        doMyFunction();
    return(0);
}

/*
 *
-----
 * Put Your Code Here
 *
-----
 */

void doMyFunction(void)
{
    CFStringRef theCmd;
    CFStringRef theMsg;
    CFStringRef theSub;
    CFStringRef theDest;
    CFStringRef theSender;

    theCmd = ParamToCFString(par->params[0]);
    theMsg= ParamToCFString(par->params[1]);

    if(CFStringCompare(theCmd,CFSTR("text"),kCFCompareCaseInsensitive) ==
kCFCompareEqualTo){
        theSub= ParamToCFString(par->params[2]);
```

```
theDest= ParamToCFString(par->params[3]);
theSender=ParamToCFString(par->params[4]);
CFSetResult(simpleSendASCIIEmail(theDest,theSub,theMsg));
}else if(CFStringCompare(theCmd,CFSTR("textWithAttach"),kCFCompareCaseInsensitiv
e) == kCFCompareEqualTo){
    NSString *textMsg = [NSString stringWithFormat:@"%@", theMsg];

    //*****Fix it to use this so I can attach
files*****
    //attributedStringWithAttachment:
    NSAttributedString *tMsg = [[NSAttributedString alloc]
initWithString:textMsg]autorelease];
    if(par->params[6]){
        CFStringRef filePath = ParamToCFString(par->params[6]);
        CFURLRef theAttachmentPath = CFURLCreateWithFileSystemPath
(NULL,filePath,kCFURLHFSPStyle,false);

        tMsg = appendString:(NSAttributedString *)
attributedStringWithAttachment:(NSTextAttachment *)
//NSDictionary tHeaders = [NSDictionary dictionaryWithObjectsAndKeys:@"value
1", @"key 1", @"value 2", @"key 2",nil];
        NSMutableDictionary *tHeaders = [NSMutableDictionary dictionary];
        // *****//
        //get the path, convert it to POSIX-style (with CFURL, for example)
        //make an NSFileWrapper for that
        //and an NSTextAttachment from that
        // and use that with attributedStringWithAttachment
        //*****//
        fillDict(tHeaders,par->params[5]);
        //[dict setObject:@"foo" forKey:@"bar"];

        CFSetResult(sendEmail(tMsg,tHeaders,@"NSMIMEFormat",nil));
        //CFSetResult(CFSTR("This feature is not implemented yet!"));
    }else if(CFStringCompare(theCmd,CFSTR("HTML"),kCFCompareCaseInsensitive) ==
kCFCompareEqualTo){
        //*****Change to use
initWithHTML*****
        unsigned long theLen = GetHandleSize(par->params[1]);
        NSData *HTMLData = [NSData dataWithBytes:*par->params[1] length:theLen];
        NSAttributedString *tMsg = [[NSAttributedString alloc] initWithHTML:HTMLDat
a documentAttributes:NULL]autorelease];
        //NSDictionary tHeaders = [NSDictionary dictionaryWithObjectsAndKeys:@"value
1", @"key 1", @"value 2", @"key 2",nil];
        NSMutableDictionary *tHeaders = [NSMutableDictionary dictionary];

        fillDict(tHeaders,par->params[5]);
        //[dict setObject:@"foo" forKey:@"bar"];
        CFSetResult(sendEmail(tMsg,tHeaders,@"NSMIMEFormat",nil));
        //CFSetResult(CFSTR("This feature is not implemented yet!"));
```

```
}else if(CFStringCompare(theCmd,CFSTR("HTMLWithAttach"),kCFCompareCaseInsensitive) == kCFCompareEqualTo){
    //*****Change to use
initWithHTML*****
    unsigned long theLen = GetHandleSize(par->params[1]);
    NSData *HTMLData = [NSData dataWithBytes:*par->params[1] length:theLen];
    //NSAttributedString *tMsg = [[[NSAttributedString alloc] initWithHTML:HTMLData documentAttributes:NULL]autorelease];

    NSAttributedString *tMsg = [[[NSAttributedString alloc] initWithHTML:HTMLData documentAttributes:NULL]autorelease];
    //NSDictionary tHeaders = [NSDictionary dictionaryWithObjectsAndKeys:@"value 1", @"key 1", @"value 2", @"key 2",nil];
    NSMutableDictionary *tHeaders = [NSMutableDictionary dictionary];

    fillDict(tHeaders,par->params[5]);
    //[dict setObject:@"foo" forKey:@"bar"];
    CFSetResult(sendEmail(tMsg,tHeaders,@"NSMIMEMailFormat",nil));

    //CFSetResult(CFSTR("This feature is not implemented yet!"));
}else if(CFStringCompare(theCmd,CFSTR("RTF"),kCFCompareCaseInsensitive) == kCFCompareEqualTo){

    unsigned long theLen = GetHandleSize(par->params[1]);
    NSData *rtfData = [NSData dataWithBytes:*par->params[1] length:theLen];
    NSAttributedString *tMsg = [[[NSAttributedString alloc] initWithRTF:rtfData documentAttributes:NULL]autorelease];
    //NSDictionary tHeaders = [NSDictionary dictionaryWithObjectsAndKeys:@"value 1", @"key 1", @"value 2", @"key 2",nil];
    NSMutableDictionary *tHeaders = [NSMutableDictionary dictionary];

    fillDict(tHeaders,par->params[5]);
    //[dict setObject:@"foo" forKey:@"bar"];
    CFSetResult(sendEmail(tMsg,tHeaders,@"NSMIMEMailFormat",nil));
    //CFSetResult(CFSTR("This feature is not implemented yet!"));
}else{
    CFSetResult(CFSTR("That command is not recognized!"));
}

}
CFStringRef sendEmail(NSAttributedString *theMsg,NSDictionary *messageHeaders,NSString *messageFormat,NSString *deliveryProtocol){
//dataWithBytes:length:
//- (id)initWithRTF:(NSData *)rtfData documentAttributes:(NSDictionary **)docAttributes

    //[NSDictionary dictionaryWithObjectsAndKeys:@"MyProgram",@"User-Agent",@"foo",@"SomeOtherHeader",nil];
    //[NSMutableDictionary dictionary] and then fill it with [dict setObject:someValue
```

```
forKey:someHeaderName]; for every header
    if([NSMailDelivery deliverMessage:(NSAttributedString *)theMsg headers:(NSDictionary *)messageHeaders format:(NSString *)messageFormat protocol:deliveryProtocol]){
        return CFSTR("Email Sent");
    }else{
        CFShow(CFSTR("Error: \nEmail Not Sent"));
        return CFSTR("Error: \rEmail Not Sent");
    }
}
```

```
char getItemDelimiter(void)
{
    Handle    h = nil;
    char      c = 0;

    if ((h = EvalExpr(par, "\pthe itemDel")) != nil) {
        c = **h;
        DisposeHandle(h);
    }
    return(c);
}
```

```
char getLineDelimiter(void)
{
    Handle    h = nil;
    char      c = 0;

    if ((h = EvalExpr(par, "\pthe lineDel")) != nil) {
        c = **h;
        DisposeHandle(h);
    }
    return(c);
}
```

```
CFStringRef simpleSendASCIIEmail(CFStringRef theEmailDest,CFStringRef
theSubject,CFStringRef theBody)
{
```

```
    // NSString *theNSEmailDest = (NSString*)theEmailDest;
    // NSString *theNSSubject = (NSString*)theSubject;
    // NSString *theNSBody = (NSString*)theBody;

    if ( [NSMailDelivery deliverMessage:(NSString*)theBody subject:(NSString*)theSubject
to:(NSString*)theEmailDest] ){
        // CFShow(CFSTR("Email Sent"));
        return CFSTR("Email Sent");
    }else{
        // CFShow(CFSTR("Email Not Sent"));
    }
}
```

```
        return CFSTR("Email Not Sent");
    }
}

void fillDict(NSMutableDictionary *theDict, Handle theParam)
{
    // prefill theDict with To,From,Subject headers
    NSString *theSub = [NSString stringWithFormat:@"%@",
ParamToCFString(par->params[2])];
    NSString *theDest = [NSString stringWithFormat:@"%@",
ParamToCFString(par->params[3])];
    NSString *theSender = [NSString stringWithFormat:@"%@",
ParamToCFString(par->params[4])];
    [theDict setObject:theDest forKey:@"To"];
    [theDict setObject:theSender forKey:@"From"];
    [theDict setObject:theSub forKey:@"Subject"];
    if(par->paramCount >= 6){
        NSString *theHeaders= [NSString stringWithFormat:@"%@",
ParamToCFString(theParam)];

        //theHeaders=ParamToCFString(theParam);
        NSArray *listItems = [theHeaders componentsSeparatedByString:[NSString
stringWithFormat:@"%c",getLineDelimiter()]];

        NSEnumerator *enumerator = [listItems objectEnumerator];
        NSString *aLine;

        while((aLine = [enumerator nextObject])){
            /* code to act on each element as it is returned */
            NSArray *theElement = [aLine componentsSeparatedByString:[NSString
stringWithFormat:@"%c",getItemDelimiter()]];
            NSEnumerator *elementItem = [theElement objectEnumerator];
            NSString *theKey = [elementItem nextObject];
            NSString *theValue = [elementItem nextObject];
            [theDict setObject:theValue forKey:theKey];
        }
    }
}

/*
*
*
* Boilerplate Utilities
*

```

```
*/

/***** Fail() *****/
// Load up par->returnValue with "false" and an error message
void Fail(char *message)
{
    Handle h;
    fatalError = true;
    if (par->returnValue) DisposeHandle(par->returnValue);
    if (!*message) message = kSyntaxMsg;
    par->returnValue = (PtrToHand("false\r", &h, 6) ||
        PtrAndHand(message, h, strlen(message)+1)) ? nil : h;
}

/***** ParamToCFString() *****/
// Convert a param to a CFStringRef

CFStringRef ParamToCFString(Handle theParam)
{
    CFStringRef theString;
    theString=CFStringCreateWithBytes(kCFAllocatorDefault,(const UInt8*)theParam,
    GetHandleSize(theParam)-1, kCFStringEncodingMacRoman,false);
    return theString;
}

/***** SetResult() *****/
// Load up par->returnValue with a string
void SetResult(char *message)
{
    PtrToHand(message, &par->returnValue, strlen(message) + 1);
}

void CFSetResult(CFStringRef message)
{
    char theMsg[CFStringGetLength(message)];
    if(CFStringGetCString(message,theMsg,CFStringGetLength(message)+1,kCFStringEncodingM
acRoman))
    {
        SetResult(theMsg);
    }else{
        SetResult("Error!\rThere was an error converting a CFString to a C string.");
    }
}

/***** FalseAlarm() *****/
// Handle requests for kSyntaxMsg or kCopyrightMsg & bad paramCounts
Boolean FalseAlarm(void)
```

```
{
Boolean result = false;
short pCnt = par->paramCount;

if(pCnt == 1) {
    if ((result = !strcmp((char*)*par->params[0], "!")))
        SetResult(kCopyrightMsg);
    else if ((result = !strcmp ((char*)*par->params[0], "?")))
        SetResult(kSyntaxMsg);
    else if ((result = !strcmp ((char*)*par->params[0], "??")))
        SetResult(kVersionNumber);
    else if ((result = !strcmp ((char*)*par->params[0], "???")))
        SetResult(__DATE__ " " __TIME__);
}
if (!result && (result = ((pCnt < kMinNumParams) || (pCnt > kMaxNumParams))))
    Fail(kSyntaxMsg);

return(result);
}
```