Building a "Rich Client" or "Internet App" with Runtime Revolution

This is a description of my approach to building a "rich client" or "Internet app" (Internet application) with Revolution. Revolution is a Rapid Application Development (RAD) tool which uses an English-like scripting language and allows cross-platform development and deployment of interactive applications.

New in August 2003: The English Windows PC version of the Lab has been integrated with the Internet! Other versions will be converted in the future. Go to Home page and then to the English download link.
My early prototype can be downloaded at www.reactorlab.net/erc/.
Related projects can be found at Fourth World Media's RevNet, Tactile Media's Tactile Media Player, and the Himalayan Academy's Cyber Study Hall. RevNet and the Tactile Media Player require the Revolution development environment.

The goal is a standalone application on a client computer that accesses stacks and media files which are posted on web servers.

In this project, local copies of stacks are saved to the client's disk for use off line.

Whenever the project is on line and new versions of any of the stacks or files are available on the server, they are automatically downloaded, written over the older copies on the client disk, and opened while the application remains open.

The standalone application file contains only a few lines of "bootstrap" script in addition to the Revolution runtime engine.

Thus, every stack and file in the project can be updated to new versions while the application is open, except for the Revolution runtime engine.

If local copies for off-line use are not required, another option would be to install Revolution as a "helper application" in a standard web browser. Then, when a user clicks on a link in a web page that is linked to a Revolution stack, the stack is downloaded and opened. I found that installing helper applications in web browsers was not an easy task and doing so required a different procedure in different browsers.

I suspect that many users would not be able to do this successfully. Thus, I would choose to use the method described here even if local copies for off-line user were not required.

For a discussion of the advantages of "Internet apps" relative to web browsers, see Fourth World Media's article "Beyond the Browser".

Engine

The "engine" stack will become the standalone application file. This stack has minimal script - essentially a bootstrap script - with all the rest of the script for the project contained in other stacks. This is done so that all of the scripts in the project can be updated to new versions while the application is open.

The engine stack can optionally have one or more substacks that enable the engine to contact the home server in the event that it becomes separated from its support stacks and files. This discussion assumes that the engine stack has these "default" stacks.

I have called this the "engine stack" because it becomes merged in the standalone application file with the Revolution engine that executes stacks and scripts, even though my engine stack and the Revolution engine are not the same thing.

On startup, the script in the engine stack sets the value of a global variable that contains the file path to the folder in which the engine stack resides (by getting "the long name of this stack" and then parsing it).

All other files in the project will be contained in subfolder "support" of this main folder.

The engine standalone file and the support folder and its contents must remain together on the client disk in the same folder.

The name of the folder containing the engine standalone file and the support folder can be changed whenever the project is not open.

The name of the support folder and the names of its contents must not be changed by the user.

Of course, all names of folders, stacks, and developer-written handlers are arbitrary, as are the folder hierarchies shown.

Then the engine checks to see whether the main support stack "/support/scripts/comm_scripts.mc" is present.

If it is, the engine stack sends message "bootstep01" to that stack.

If the main support stack is not present, then the engine accesses its default substack copy of this stack.

These two steps - set a global with the path to main folder and send a message to the main support stack - are the only things that the engine's script does.

The handler "bootstep01" in the main support stack (or default copy) puts the stack in use and then checks to see whether the main url library stack "/support/scripts/liburl.mc" is present on disk.

If it is, then the stack is put in use.

If not, the default substack copy in the engine stack is put in use. Stack liburl.mc is part of the Revolution distribution.

Next the handler checks to see whether the main interface or "directory" stack is present on disk at "/support/scripts/directory.mc".

If it is, then that "directory" stack is opened.

If not, a simple "connect" stack is generated by script (or could be a substack of the engine stack).

The "connect" stack just has a button that contacts the home server and causes the full set of support files on disk to be downloaded, saved to disk, and the new directory stack opened.

Directory
When the directory stack opens, it displays a list of any modules with local copies present on the client's disk.

When the user clicks to go on line, the directory first downloads an "updater" stack from the home server and opens it.

The updater stack checks to see whether the versions of the support files on the server are more recent versions than on the client.

Currently, lists of stacks and version information are contained in plain text files.

The server has a copy and the local client has a copy (/support/scripts/script_list.txt).

Any files which are older, or not present, on the client are updated.

An alternative to having this information in a text file would be to execute a script that gets the names of files in a folder and also gets their modification dates.

Updating of a file occurs by the updater stack first downloading and then over-writing the old copy of the file on the client's disk.

Then the updater activates the new copy. For example, for stacks that are not open but "in use", this activation occurs by "stop using stack...", then by "delete stack...", then by "start using stack…"

Note that, for stacks in use, the old copy of the stack must be deleted from RAM, even though it may not be open.

That is, for a stack that is in use but not open, stop using the stack, delete the stack (from RAM), and then start using the new copy.

This is true for a stack in use even if it is open and its destroyStack property set to true and then closed.

For an open stack that is also in use: stop using the stack, close the stack, delete the stack (from RAM), then finally open the new copy and put it in use.

The updater should also check if the old version of the stack is in the front or back scripts, and activate the new copy as necessary.

An old stack in a front or back script must removed from the front or back scripts, and then the stack must be deleted from RAM before the new copy of the stack is inserted into the front or back scripts.

After all of the support files, including the directory, have been updated if necessary to the latest version, the updater stack closes itself. The updater stack is not saved to the client disk.

In this manner, every disk file in the project on the client can be updated with the project open, except for the engine standalone application file.

If the engine standalone application file needs to be updated, the updater stack can download the new copy into a separate folder, and then instruct the user to quit the application, replace the old copy of the standalone file with the new copy, and then restart the application.

Since the engine stack contains a very minimal script, it would only need to be updated when a new Revolution engine version is made available that contains new features that you want to use in the project.

Modules

The design of the directory stack interface is completely arbitrary and can be changed whenever desired by simply posting the new version on the home server and updating the stack's entry in /support/scripts/ script_list.txt.

In the prototype project, the directory has two main elements: an "option" menu button that allows the user to select a server, and a field that lists stacks, which I call "lab modules", that are available on the selected server.

When the user is off line and selects a server, the directory checks to see if there is a local copy of a text file with a list of modules present on the server, "/support/labs/serverName/module_list.txt".

If so, the list is displayed. If not, a notice is displayed in the field telling the user they must go on line to see the list.

When the user is on line and selects a server, the directory downloads from the server a text file with a list of modules and their version numbers.

If there is a local copy of this text file on the client disk, the directory adds any new modules to the local text file. If there is not a local copy, the text file from the server is written to the client.

Modules that have local copies on the client are denoted as having a local copy both in the module list text file and in the field in the directory.

When a user clicks on a module that is displayed in the directory, the directory checks to see if there is a local copy on the client.

If there is a local copy and the user is off line, or if the user is on line and the local copy is the most recent version, the directory opens the local copy.

If there is not a local copy and the user is off line, the user is informed that they must go on line to access the module.

If there is a local copy and the user is on line but the local copy is not the most recent version, then directory downloads the most recent copy of the module, saves it to disk, updates the module list text file on the client, and then opens the module.

When the user is on line and a module stack is downloaded and opened, the module stack may then download other files from the server.

The module stack and its support files are saved to a separate folder inside the server's folder: /support/labs/serverName/module_folder/.

Scripting details

Downloading stacks

The Revolution commands I use are "load url" to download a stack into RAM, and then "go url" to open the stack.

The load command is "non blocking" in the sense that the rest of the lines in a script will execute after the load command has been issued and while the load process is continuing.

For example, other script lines can update a progress indicator during the download.

For downloading stacks and other files from the server, I modified scripts from the "download stack", a substack in the Revolution distribution's mctools.mc stack.

In the Revolution development environment issue the command: go stack "download stack".

Then open that stack's script to see the handlers. Alternatively, you can just use a copy of this stack in your project.

In addition to "load" and "go url", the script of the download stack also checks the status of the non blocking load process and updates a progress indicator.

You can also inspect Runtime Revolution scripts for downloading stacks. See the script of the button revCommon of stack revLibrary in the Revolution development stacks.

Downloading other files

Currently, I simply use the Revolution command "get url" to download other files from the server.

The "get" command is "blocking" in the sense that script execution stops until the get process is completed.

If you want to download a large file that is not to be used immediately, or if you want to update a progress indicator during the download process, you can use the non blocking load command instead.

Files that have been downloaded into a variable can be written to disk with the "put" command.

The put command will create a new file or overwrite an existing file.

An example of getting a file from the server and writing it to the client's disk:

```
get url "http://myserver.com/myfiles/myphoto.jpg"
put it into url "binfile:C:/WINDOWS/Desktop/myphoto.jpg"
```

Note the binary file designation "binfile:" that is used with files that do not contain plain text. The plain text file designation is "file:".

Posting information to a CGI script on the server

The "post" command can be used to send information to a Common Gateway Interface (CGI) script on the server for processing.

For example:

```
set the httpHeaders to "Content-type: text/plain" & cr
put cr & the time && "hello from stack" && the short name
of this stack into tMsg
post tMsg to url "http://myserver.com/cgi-bin/mycgi.mt"
```

The file "mycgi.mt" is a Revolution script file in the cgi-bin directory of server that is executed by a copy of the Revolution engine in the cgi-bin directory.

Other CGI languages such as Perl can also be used - but why bother when you know Revolution.

The "post" process requires that the CGI script return a response to the client. The response can include such information as results computed from input information in the message received, data retrieved from a database on the client, or a message saying that the message has been received and recorded to a log file on the server.

Security

Whenever executable scripts are downloaded from the web, the potential exists for a rogue script to cause damage to the client's disk files or to upload private information.

This risk can be reduced by restricting access of the project to "trusted" servers, as is done in this project.

One option is to set the secureMode property to true. This prevents access to the disk file system or other resources of the client.

Once the secureMode is set to true, it stays true for the rest of the time the Revolution application remains open.

This protects the client's disk files but it also prevents saving copies of files to the client disk for use off line.

Another possibility is to add a "certificate of trust" to a stack that certifies that it can be trusted.

Such a certificate might consist of a result of the md5Digest() function stored as a custom property of the stack.

Set the lockMessages property to true, open the stack, check for a valid certificate and, if the certificate is valid, then send "open messages" or close and reopen stack.

Stack “erc engine”

Stack Script:

```
-- by Richard K. Herz <rich@reactorlab.net>

global gPathToEngine, gPathToEngineFolder

on startUp
  # on first start up of Rev engine when a standalone
  bootStep01
  pass startUp
end startUp

# when double-click bare stack on Win desktop
# no cursor is available until after end startUp

on bootStep01

    -- save the path to the standalone
    -- the script below does not work on Mac OS X
    -- a different method must be used on Mac OS X

    put the long name of this stack into gPathToEngine
    delete char 1 to 7 of gPathToEngine # stack "
    delete the last char of gPathToEngine # "
    put gPathToEngine into gPathToEngineFolder
    set the itemdelimiter to "/"
    delete the last item of gPathToEngineFolder

    put gPathToEngineFolder & "/support/scripts/comm_scripts.rev" into tPath
    if there is a stack tPath then
        start using stack tPath
        bootStep02
    else
        ## xxx "default" substacks have been removed from this version
        #    start using stack "default comm scripts"
        #    bootStep02
    end if
end bootStep01

Button: “bootStep01”
on mouseUp
  bootStep01 # in stack script
end mouseUp
```

Stack “Comm scripts”

stack script

```
# by Richard K. Herz <herz@ucsd.edu>  May 2001
# revised October 2002
# revised February 2003
# revised May 2003

local stime, cancelled # from Rev download stack
local localGUIname
global gLoggerURL # set in Directory stack and used below
global gPathToEngine, gPathToEngineFolder # set on Rev startUp in engine

on bootStep02
  set cursor to watch
  # need to put this script into the backScripts (with "into back")
  # or it won't get any messages sent by engine stack connect button
  insert the script of me into back
  set the directory to gPathToEngineFolder
  startLibUrl
  initEngine
  set the loc of stack "erc engine" to -200,-200
  put gPathToEngineFolder & "/support/scripts/erc_directory.rev" into tPath
  if there is a stack tPath then
    # setting vis to false doesn't seem to work when engine is standalone
    go stack tPath
  else
    makeEngineGUI
    toplevel stack localGUIname
  end if
end bootStep02

on startLibUrl
  put gPathToEngineFolder & "/support/scripts/liburl.rev" into tPath
  if there is a stack tPath then
    start using stack tPath
  else
    ## xxx default substacks have been removed in this version
    #    set the itemDelimiter to "/"
    #    put "default liburl," into tPath
    #    put the last item of gPathToEngine after tPath
    #    set the stackfiles of me to tPath
    #    start using stack "default liburl"
  end if
end startLibUrl
```

```
on makeEngineGUI
  # make a new stack for this, since can't set script in engine if scripts are locked
  set the rect of the templateStack to 238,282,566,397
  put "engineGUI" into localGUIname # specific "engineGUI" used in directory
openstack
  set the name of the templateStack to localGUIname
  set the label of the templateStack to "ERC Web Courses connector"
  set the resizable of the templateStack to false
  set the destroyStack of the templateStack to true
  set the destroyWindow of the templateStack to true
  create stack
  go stack localGUIname
  set the rect of the templateField to 16,6,313,68
  set the name of the templateField to "status" # xxx see displayStatus handler
  set the opaque of the templateField to "false"
  set the showBorder of the templateField to "false"
  set the lockText of the templateField to "true"
  create field
  put "Prototype by Richard K. Herz <herz@ucsd.edu>." into field "status"
  put cr & "Click connect button..." after field "status"
  set the loc of the templateButton to 157,91
  set the name of the templateButton to "connect"
  set the traversalOn of the templateButton to false
  create button
  # "me" is the stack that this script is in
  # "this stack" is the engine stack currently
  get the cEngineConnectButtonScript of me
  set the script of button "connect" to it
  get the cEngineStackScript of me
  set the script of stack localGUIname to it
  set the tool to browse
end makeEngineGUI

# xxx alternative is just to open a GUI stack
# maybe make it as a substack of this stack? - but wierd place
# and then would have to make it a substack of the default of this stack
# and don't know if you can have subs of subs?

# xxx want to be able to make a GUI stack with normal dev tools
# rather than script it as in makeEngineGUI
# maybe use development GUI to make a stack
# then use my xml script to write a description to xml
# and store as a custom prop of this stack (and default)
# then use a stack creation script to read it and generate a GUI stack
```

```
on initEngine

  global gHomeURLs
  global gHomeURLDefault

  # if there is a button "secure mode" then
  # set the hilite of button "secure mode" to false
  # end if

  if there is a button "info" then
    if the platform is "WinOS" then
      set the textFont of button "info" to "courier new"
    else
      set the textFont of button "info" to "courier"
    end if
  end if

  displayStatus empty

  # QuickTime 5.0.2 has a bug that affects transition effects
  if the platform is "MacOS" and the qtVersion is "5.0.2" then set the dontUseQT to true

  # will be specified in Directory, used below in loadGoFinish
  # set to empty here in case it ever gets accessed but isn't assigned yet
  put empty into gLoggerURL

  # gHomeURLDefault is URL checked if all others don't download an mc stack
  # put "http://mechanics.ucsd.edu/research/herz/web_24/wc_directory.rev" into
gHomeURLDefault

  put "http://reactorlab.net/ERC_courses/erc_directory.rev" into gHomeURLDefault

  # specify list of URLs to mc stacks on servers; more than one in case preferred
one is down

  # here I specify this list in engine
  # below, commented out, is an option to check a disk text file with the list
  # probably want that text file to be on locked volume to prevent "hacking"

  put "http://reactorlab.net/ERC_courses/erc_directory.rev" into line 1 of gHomeURLs
  put "http://mechanics.ucsd.edu/research/herz/web_24/wc_directory.rev" into line 2 of
gHomeURLs
  put "http://courses.ucsd.edu/rherz/web_courses/wc_directory.rev" into line 3 of
gHomeURLs

  ## --- option to check disk file for list of URLs ---
```

```
  # put the filename of this stack into tPathTOapp
  # set the itemDelimiter to "/"
  # delete the last item of tPathTOapp  # delete:  this stack's name
  # set the itemDelimiter to ","  # reset to default delimiter
  #
  # put tPathTOapp & "/program_folder" into tPathTOscripts
  # put tPathTOscripts & "/startup.txt" into tPathTOhomeURLs
  #
  # if there is a file tPathTOhomeURLs then
  # put url tPathTOhomeURLs into gHomeURLs
  # else
  # put gHomeURLDefault into gHomeURLs
  # end if
  #
  # # delete comments
  # put the number of lines of gHomeURLs into tLines
  # repeat with i = 1 to tLines
  # if offset("#", line i of gHomeURLs) > 0 then
  # put the number of characters of line i of gHomeURLs into tEndC
  # delete character offset("#", line i of gHomeURLs) to tEndC of line i of
gHomeURLs
  # end if
  # end repeat
  # repeat with i = tLines down to 1
  # get stripper(line i of gHomeURLs)
  # if it is empty then
  # delete line i of gHomeURLs
  # end if
  # end repeat
  # if gHomeURLs is empty then
  # put gHomeURLDefault into gHomeURLs
  # end if

end initEngine

# ------------------

on phoneHome

  global gHomeURLs
  global gHomeURLDefault

  # xxx need to set up a repeat through gHomeURLs
  # xxx and a way to know a directory has been received to get out of repeat

  loadGoStart gHomeURLDefault
```

```
end phoneHome

# ------------------

# handlers below are from MC 2.4 download stack in mctools.mc
# edited and names changed by R. Herz

on mouseUp
  if the short name of the target is "Cancel"
  then put true into cancelled
end mouseUp

# ----------------

on loadGoStart href

  put false into cancelled
  load href
  displayStatus "contacting"
  send "loadGoFinish" && quote & href & quote to me in 1 second
  put the long seconds into stime
end loadGoStart

# ----------------

on loadGoFinish href

  local stat, nk

  if cancelled then
    unload url href
    exit to MetaCard
  end if
  put urlStatus(href) into stat
  displayStatus stat

  if stat contains "error" or stat contains "not found" then
    answer "Error downloading URL" && href  & cr \
       & "Check network connection and proxy setup in Preferences."
    unload url href
    displayStatus empty
    exit to MetaCard
  end if

  if stat contains "cached" then
```

```
  set the itemDelimiter to "."
  if the last item of href is "gz" then
    #gzipped stack
    displayStatus "decompressing"
    put decompress(url href) into temp
    if word 1 of temp is "#!/bin/sh" then
      displayStatus empty
      go temp
      unload url href
      put empty into temp
      exit to MetaCard
    else
      answer "URL" && href  && "is not a stack."
      unload url href
      put empty into temp
      displayStatus empty
      exit to MetaCard
    end if
  else

    if word 1 of url href is "#!/bin/sh" then
      displayStatus empty
      go url href
      unload url href
      exit to MetaCard
    else
      answer "URL" && href  && "is not a stack."
      displayStatus empty
      unload url href
      exit to MetaCard
    end if

  end if
end if

if item 1 of stat is "loading" then
  put item 2 of stat div 1024 into nk
  displayStatus nk & "K/" & item 3 of stat div 1024 & "K  "\
      & nk * 10 div (the long seconds - stime) / 10 & "K/sec"
end if

if stat is empty then
  displayStatus empty
  exit to MetaCard
end if
```

```
  if stat is "requested" then
    send "loadGoFinish" && quote & href & quote to me in 1 second
  else
    send "loadGoFinish" && quote & href & quote to me in 500 milliseconds
  end if

end loadGoFinish

# ----------------

on displayStatus tStatus
  if there is a field "status" of the topStack then
    put tStatus into field "status" of the topStack
  else
    if there is a field "status" of me then
      put tStatus into field "status" of me
    end if
  end if
end displayStatus

# ----------------

on unloadAll

  repeat for each line i in the cachedURLs
    unload url i
  end repeat

end unloadAll

# ------------------------

function stripper tVar

  # strips spaces before and after, and returns after string

  # on one server (Win NT server) I am getting an ASCII 13
  # which isn't recognized as return or cr
  # this doesn't happen on other servers (Sun Unix server)

  repeat while charTOnum(the last char of tVar) is 13
    #    answer "delete an ASCII 13 (return)"
    delete the last character of tVar
  end repeat
```

```
  repeat while the last character of tVar is return
    delete the last character of tVar
  end repeat

  repeat while the last character of tVaris is space
    delete the last character of tVar
  end repeat

  repeat while the first character of tVar is space
    delete the first character of tVar
  end repeat

  return tVar

end stripper

# ----------------------------

# from Runtime Revolution 2.0.1
# from button revCommon of stack revLibrary

on revGoURL pWhich
  global gREVWebBrowser
  revSetWindowsShellCommand
  put revRunningWindowsNT() into tNT
  if "file://" is not in pWhich then replace "file:/" with "file:///" in pWhich--so that local
URLs as used in Transcript can be used
  if the platform is "Win32" then
    if char 1 to 7 of pWhich is "mailto:" then
      put queryRegistry("HKEY_CLASSES_ROOT\mailto\shell\open\command\") into
tMailApp
      replace quote & "%1" & quote with pWhich in tMailApp
      replace "%1" with pWhich in tMailApp

      --older versions use %l ("pecent L")
      replace quote & "%l" & quote with pWhich in tMailApp
      replace "%l" with pWhich in tMailApp

      -- for the new and wonderful XP
      replace "%ProgramFiles%" with $ProgramFiles in tMailApp

open process tMailApp for neither # original

      # launch tMailApp
      # if the result is not empty then
        # answer "Sorry, can't open mail application." & cr & "Please do it manually."
```

```
      # end if

    else
      put word 1 of
queryRegistry("hkey_local_machine\software\classes\http\shell\open\command\") into
tWebBrowserPath
      if not tNT then
        get shell(tWebBrowserPath && quote & pWhich & quote)
      else
        set the hideconsolewindows to false
        open process (tWebBrowserPath && quote & pWhich & quote) for neither
      end if
    end if
  else if the platform is "MacOS" then
    if "appleScript" is not in the alternateLanguages then
      return "Error: AppleScript not installed"
    end if
    do ("open location" && quote & pWhich & quote) as appleScript
  else
    if gREVWebBrowser is empty
    then launch "mozilla" && quote & pWhich & quote
    else launch gREVWebBrowser && quote & pWhich & quote
  end if
end revGoURL

on revMail pTo, pCC, pSubject, pBody
  local tURL
  put empty into tURL
  if the paramcount is 1 then
    put "mailto:" & pTo into tURL
  else
    if pCC is not empty then put true into tCC
    if pSubject is not empty then put true into tSubject
    if pBody is not empty then put true into tBody

    put "mailto:" & pTo into tURL
    if tCC then
      if (tURL contains "?Subject=") or (tURL contains "?Body=") then
        put "&" & "CC=" & pCC after tURL
      else
        put "?" & "CC=" & pCC after tURL
      end if
    end if
    if tSubject then
      if (tURL contains "?CC=") or (tURL contains "?Body=") then
        put "&" & "Subject=" & pSubject after tURL
```

```
      else
        put "?" & "Subject=" & pSubject after tURL
      end if
    end if
    if tBody then
      if (tURL contains "?CC=") or (tURL contains "?Subject=") then
        put "&" & "Body=" & pBody after tURL
      else
        put "?" & "Body=" & pBody after tURL
      end if
    end if
  end if
  revGoURL tURL
end revMail

function revRunningWindowsNT
  if the platform is not "Win32" then return false
  if word 1 of the systemVersion is "Windows"
  then return false
  else return true
end revRunningWindowsNT

on revSetWindowsShellCommand
  if the platform is not "Win32" then exit revSetWindowsShellCommand
  set the hideConsoleWindows to true
  if $COMSPEC is not empty then set the shellCommand to $COMSPEC
  else
    --just in case some windows versions don't use $COMSPEC
    if revRunningWindowsNT() then set the shellCommand to "cmd.exe"
    else set the shellCommand to "command.com"
  end if
end revSetWindowsShellCommand

# ----------------------------

# launchInternetApp by Richmond Mathewson, 2003
# posted at www.runrev.com, developers, user contributions

on launchInternetApp tAddress
  switch the platform
  case "MacOS"
    send tAddress to program "Finder" with "GURLGURL"
    if the result is not empty then
      answer error "Sorry, unable to open your e-mail or web browser. Please check your
internet settings"
    end if
```

```
    break
  case "Win32"
    if tAddress contains "www" then
      put word 1 to -2 of \
          queryRegistry("hkey_local_machine\software\classes\http\shell\open\command\")
into tBrowserPath
      launch tAddress with tBrowserPath
      -- open process tBrowserPath && tURL -- this works too
      if the result is not empty then
        answer error "Sorry, unable to open Web browser. Please check your internet
settings"
      end if
    else
      if tAddress contains "mailto"  then
        put word 1 to -2 of
queryRegistry("hkey_local_machine\software\classes\mailto\shell\open\command\") \
            into tBrowserPath
        launch tAddress with tBrowserPath
        if the result is not empty then
          answer error "Sorry, unable to open e-mail application."
        end if
      end if
    end if
    break
  end switch
end launchInternetApp
```

Stack “erc directory”

stack script:

```
-- by Richard K. Herz <rich@reactorlab.net>

# Rules for module main stacks:
# Set destroyWindow and destroyStack properties to true.
# Set name of stack (space or _ word separator) to same as
# stack's filename ( _ word separator plus .rev extension).
# Label of stack can differ from label used in module_list.txt on server.
# Can NOT have $ in stack file name or label in module_list.txt
# Name of directory stack is used specifically below and must have
# either _ or space as used in stack name, not either one as in module stacks.

global gPathToEngineFolder # set on startUp of Rev
global gLabURL # assigned in directory stack
```

```
global gConnectStatus # assigned in directory stack
global gModuleSupportPath # assigned by module
global gModules # assigned in directory stack when getting module list
global gLoggerURL #  assigned below in openStack
global gHistory # assigned below in openStack

on openStack

   if the short name of this stack is "erc directory" then # xxx specific name
      # OK to proceed
   else
      # since this stack will be in use, will get other stack's openStack
      pass openStack
      exit to metacard # exit to Rev engine
   end if

   set cursor to watch

   # start using this stack to handle messages from modules
   # this line needs to be above sending mouseUp to button "server"
   if the short name of me is not in the stacksinuse then
      start using me
   end if

   -- specify the path to the CGI script on the server to log activity
   -- put "http://reactorlab.net/cgi-bin/logit.pl" into gLoggerURL
   put empty into gLoggerURL

   set the cFirstConnectMade of this stack to "false" # used in button "on line"

   if "engineGUI" is in the windows then
      # this directory was just downloaded by the bare engine bootstrap
      # xxx currently updater will end up updating this stack just downloaded
      # xxx because there is no support/scripts/script_list.txt file with
      # xxx this stack and current server version listed
      close stack "engineGUI"
      send mouseUp to button "on line" of card 1 of me
   else
      send mouseUp to button "off line" of card 1 of me
   end if
   set the label of button "server" of card 1 of me to "San Diego"
   send mouseUp to button "server" of card 1 of me

   put the time & " enter " & the short name of this stack into gHistory

end openStack
```

```
# ----------------------

on closeStackRequest

  # this stack is in use so it will get all closeStackRequests
  # check to make sure sending stack is this stack before processing

  if the short name of this stack is "erc directory" then # xxx specific name

    # need to quit if close this window because have engine stack window off
screen
    # and no way to close it unless quit here

    if the environment is "development" then
      answer "this would quit standalone after asking user"
      pass closeStackRequest
      exit to metacard # exit to Rev engine
    end if

    answer "Closing this window will quit ERC Web Courses." & cr &  "Do you want to
quit?" with "Yes" or "No"
    if it is "no" then
      exit closeStackRequest
    end if

    set cursor to watch

    if gLoggerURL is not empty then
      set cursor to watch
      set the httpheaders to "Content-type: application/x-www-form-urlencoded" & return
      put cr & the time & " exit " & the short name of this stack after gHistory
      if gConnectStatus is true then
        post gHistory to url gLoggerURL
      end if
    end if

    quit

  end if

  pass closeStackRequest

end closeStackRequest

# ----------------------
```

```
on displayStatus tStatus
  if there is a field "status" then
    put tStatus into field "status"
  end if
end displayStatus

# ----------------------

on initModuleFiles tFileList

  put the long name of this stack into tStackPath
  delete char 1 to 7 of tStackPath # delete stack "
  set the itemdelimiter to "/"
  delete the last item of tStackPath

  if tStackPath is empty then
    initModuleFromServer tFileList
  else
    initModuleFromClient tFileList
  end if

end initModuleFiles

# -------------------------

on initModuleFromServer tFileList

set cursor to watch

  # stack was just downloaded from web

  # go to card 1 in case user closed stack on another card
  # also may get error when trying to show/hide field "splash"
  go card 1

  # need to verify buttons and fields, since all modules don't have them
  if there is a field "splash" then
    show field "splash"
    put cr & "Please wait a moment while component files are downloaded..." into field
"splash"
  end if
  if there is a button "Run" then
    set the disabled of button "Run" to true
  end if
  # these will be reset at very bottom of this handler
```

```
  # need to create a new folder and download support files into it

  put gPathToEngineFolder into tSupportFolderPath

  put "/support" after tSupportFolderPath
  if there is a directory tSupportFolderPath then
    # support directory exists
  else
    create directory tSupportFolderPath
  end if

  put tSupportFolderPath & "/labs" into  tLabsFolderPath
  if there is a directory tLabsFolderPath then
    # labs directory exists
  else
    create directory tLabsFolderPath
  end if

  # want to put lab in a server directory in case other servers have a different
  # lab of same name (which means you want servers to also have different
names!)

  # the stack name in next line is specific to current name of directory stack
  # have to specify specific stack or will look for button of lab module that's
opening
  put tLabsFolderPath & "/" & the label of button "server" of stack "erc directory" into
tServerFolderPath
  # xxx specific button and stack name in line above

  if there is a directory tServerFolderPath then
    # server directory exists
  else
    create directory tServerFolderPath
  end if

  # name of lab stack must be same as name of stack file listed in module_list.txt
  # label of lab stack can be anything
  # generate stack file name from stack name

  put the short name of this stack into tModuleFileName
  replace space with "_" in tModuleFileName
  put tModuleFileName into tModuleFolderName
  put ".rev" after tModuleFileName
  put tServerFolderPath & "/" & tModuleFolderName into tModuleFolderPath
```

```
  if there is a directory tModuleFolderPath then
    # lab directory exists
  else
    create directory tModuleFolderPath
  end if

  # save stack to disk
  set the directory to tModuleFolderPath
  put tModuleFolderPath into gModuleSupportPath # for use by module to locate its
files
  set the filename of this stack to tModuleFileName
  save this stack

  # note lab as local in client module list
  put "file:" & tServerFolderPath & "/module_list.txt" into tServerModuleListPath
  put url tServerModuleListPath into tClientList
  # now add local notation and version number to local module list
  put lineoffset(tModuleFileName, tClientList) into tLine
  set the itemdelimiter to "$"
  put " (local) " into item 4 of line tLine of tClientList
  put empty into item 5 of line tLine of tClientList # just downloaded module so not
"old"

  # update version number in client module list
  put line 1 of gModules into tServerURL # all lines have necessary info
  set the itemdelimiter to "/"
  delete the last item of tServerURL
  put stripper(tServerURL) into tServerURL
  put tServerURL & "/module_list.txt" into tServerListURL
  get url tServerListURL
  put it into tServerList
  put line tLine of tServerList into tServerList
  set the itemdelimiter to "$"
  put item 3 of tServerList into item 3 of line tLine of tClientList

  # write updated client module list to disk
  put tClientList into url tServerModuleListPath

  # mark module listing in Directory as having a local copy
  put the label of this stack into tStackLabel # this stack is the module main stack

  # xxx specific stack name in lines below, e.g., for "erc directory"

  put the cClickLine of field "list" of stack "erc directory" into tLine # set when user
clicked to get lab
  put line tLine of field "list" of stack "erc directory" into tModLine
```

```
  put stripper(tModLine) into tModLine # remove any returns
  if the last word of tModLine is "(old)" then delete the last word of tModLine
  if the last word of tModLine is "(local)" then
    # leave it
  else
    put " (local)" after tModLine
  end if
  put tModLine into line tLine of field "list" of stack "erc directory"

  # xxx above marking of module_list.txt and directory list assume everything
below goes OK

  # get each file in tFileList from server and write it to local current directory

  put gLabURL into tSupportURL
  put the number of characters of tSupportURL into tchar
  delete character (tchar-2) to tchar of tSupportURL # delete ".rev" at end of module
stack name
  put "_support/" after tSupportURL

  if tFileList is not empty then
    repeat with i = 1 to the number of lines in tFileList
      put tSupportURL & line i of tFileList into tempFileURL
      put "binfile:" & line i of tFileList into tempFilePath

      get url tempFileURL
      put it into url tempFilePath

    end repeat
  end if

  if there is a field "splash" then
    hide field "splash"
  end if
  if there is a button "Run" then
    set the enabled of button "Run" to true
  end if

end initModuleFromServer

# ---------------------

on initModuleFromClient tFileList

  set cursor to watch
```

```
  # a local copy of the stack is running
  # still need to tell local copy of stack where its support files are

  # xxx for some reason I keep getting these popping up!
  if there is a field "splash" then
    hide field "splash"
  end if
  if there is a button "Run" then
    set the enabled of button "Run" to true
  end if

  put the long name of this stack into tModuleFolderPath
  delete char 1 to 7 of tModuleFolderPath
  set the itemdelimiter to "/"
  delete the last item of tModuleFolderPath

  # check to make sure required folder and files are present
  if there is a directory tModuleFolderPath then
    # directory exists, check to see if all files needed are there
    # xxx for now assume everything is there
  else
    # directory missing so module can't run
    # xxx how to handle this?
  end if

  put tModuleFolderPath into gModuleSupportPath

end initModuleFromClient

# ------------------------

function stripper tVar

  # on courses.ucsd.edu (Win NT server) I am getting an ASCII 13
  # which isn't recognized as return or cr
  # this doesn't happen on mechanics.ucsd.edu (Sun Unix server)

  repeat while chartonum(the last char of tVar) is 13
    #    answer "delete an ASCII 13 (return)"
    delete the last character of tVar
  end repeat

  repeat while the last character of tVar is return
    delete the last character of tVar
  end repeat
```

```
  repeat while the last character of tVar is cr
    delete the last character of tVar
  end repeat

  repeat while the last character of tVar is space
    delete the last character of tVar
  end repeat

  repeat while the first character of tVar is space
    delete the first character of tVar
  end repeat

  return tVar

end stripper

# ----------------------------

# from Runtime Revolution 2.0.1
# from button revCommon of stack revLibrary

on revGoURL pWhich
  global gREVWebBrowser
  revSetWindowsShellCommand
  put revRunningWindowsNT() into tNT
  if "file://" is not in pWhich then replace "file:/" with "file:///" in pWhich--so that local
URLs as used in Transcript can be used
  if the platform is "Win32" then
    if char 1 to 7 of pWhich is "mailto:" then
      put queryregistry("HKEY_CLASSES_ROOT\mailto\shell\open\command\") into
tMailApp
      replace quote & "%1" & quote with pWhich in tMailApp
      replace "%1" with pWhich in tMailApp

      --older versions use %l ("pecent L")
      replace quote & "%l" & quote with pWhich in tMailApp
      replace "%l" with pWhich in tMailApp

      -- for the new and wonderful XP
      replace "%ProgramFiles%" with $ProgramFiles in tMailApp

      open process tMailApp for neither
    else
      put word 1 of
queryregistry("hkey_local_machine\software\classes\http\shell\open\command\") into
tWebBrowserPath
```

```
      if not tNT then
        get shell(tWebBrowserPath && quote & pWhich & quote)
      else
        set the hideconsolewindows to false
        open process (tWebBrowserPath && quote & pWhich & quote) for neither
      end if
    end if
  else if the platform is "MacOS" then
    if "appleScript" is not in the alternatelanguages then
      return "Error: AppleScript not installed"
    end if
    do ("open location" && quote & pWhich & quote) as appleScript
  else
    if gREVWebBrowser is empty
    then launch "mozilla" && quote & pWhich & quote
    else launch gREVWebBrowser && quote & pWhich & quote
  end if
end revGoURL

on revMail pTo, pCC, pSubject, pBody
  local tURL
  put empty into tURL
  if the paramcount is 1 then
    put "mailto:" & pTo into tURL
  else
    if pCC is not empty then put true into tCC
    if pSubject is not empty then put true into tSubject
    if pBody is not empty then put true into tBody

    put "mailto:" & pTo into tURL
    if tCC then
      if (tURL contains "?Subject=") or (tURL contains "?Body=") then
        put "&" & "CC=" & pCC after tURL
      else
        put "?" & "CC=" & pCC after tURL
      end if
    end if
    if tSubject then
      if (tURL contains "?CC=") or (tURL contains "?Body=") then
        put "&" & "Subject=" & pSubject after tURL
      else
        put "?" & "Subject=" & pSubject after tURL
      end if
    end if
    if tBody then
      if (tURL contains "?CC=") or (tURL contains "?Subject=") then
```

```
      put "&" & "Body=" & pBody after tURL
    else
      put "?" & "Body=" & pBody after tURL
    end if
  end if
 end if
 revGoURL tURL
end revMail

function revRunningWindowsNT
 if the platform is not "Win32" then return false
 if word 1 of the systemversion is "Windows"
 then return false
 else return true
end revRunningWindowsNT

on revSetWindowsShellCommand
 if the platform is not "Win32" then exit revSetWindowsShellCommand
 set the hideconsolewindows to true
 if $COMSPEC is not empty then set the shellcommand to $COMSPEC
 else
  --just in case some windows versions don't use $COMSPEC
  if revRunningWindowsNT() then set the shellcommand to "cmd.exe"
  else set the shellcommand to "command.com"
 end if
end revSetWindowsShellCommand

# ----------------------------
```

Card id 1002
card script:

```
global gPathToEngineFolder # set on engine startUp
global gConnectStatus # assigned in directory stack
global gModules
global gLabURL
global gHistory

on mouseUp

 # setting cursor to watch doesn't work because
 # gets reset immediately after loadGoStart
```

```
  # xxx without this, when ask to go on line below and user does
  # then would get server list and get first lab but then put
  # (local) before first lab - mouse click was going somewhere
  # it shouldn't have
  put me into temp
  # string "Work on line to get" set in button "server" of this stack
  if temp contains "Work on line to get" then
    exit mouseUp
  end if

  get the clickLine
  put word 2 of it into tLine
  get line tLine of me
  if it is empty then exit mouseUp
  put it into tLab
  select line tLine of me
  wait 3 ticks
  select empty

  set the cClickLine of me to tLine # used in initModuleFiles in this stack script

  # xxx note space so finds start or whole word
  # xxx but lab name could be "the local temperature" or "the old forest"
  # xxx and get spurious hit
  if line tLine of me contains "(local)" then
    if line tLine of me contains "(old)" then
      # old local copy
      if gConnectStatus is "true" then
        getLabOnServer tLine
      else
        getLabOnClient tLine
      end if
    else
      # local current copy or "local only"
      getLabOnClient tLine
    end if
  else
    # no local copy
    if gConnectStatus is "true" then
      getLabOnServer tLine
    else
      answer "Only available on line. Connect?" with "no" or "yes"
      if it is "yes" then
        send mouseUp to button "on line"
        # xxx cClickLine gets set to 0 by button "server" which is activated by button
"on line"
```

```
      # xxx set the cClickLine of me to tLine
      getLabOnServer tLine
     else
      exit mouseUp
     end if
   end if
  end if

end mouseUp

# ---------------------

on getLabOnServer tLine
  put line tLine of gModules into gLabURL
  put cr & the time && gLabURL after gHistory
  loadGoStart gLabURL
end getLabOnServer

# --------------------

on getLabOnClient tLine

  # now check for local labs

  put gPathToEngineFolder into tModulesFolderPath

  put "/support" after tModulesFolderPath
  if there is a directory tModulesFolderPath then
    # support directory exists
  else
    create directory tModulesFolderPath
  end if

  put "/labs" after tModulesFolderPath
  if there is a directory tModulesFolderPath then
    # labs directory exists
  else
    create directory tModulesFolderPath
  end if

  put the label of button "server" into tServerName
  put "/" & tServerName after tModulesFolderPath

  if there is a directory tModulesFolderPath then
    # server directory exists
  else
```

```
    # create directory tModulesFolderPath
    put "go on line to get server list" into field "list"
    exit getLabOnClient
  end if

  put tModulesFolderPath into tModuleListPath
  put "/module_list.txt" after tModuleListPath
  put "file:" & tModuleListPath into tModuleListURL

  if there is a file tModuleListPath then

    # check server lab list for local labs and versions
    put url tModuleListURL into tModules
    set the itemDelimiter to "$"
    put item 2 of line tLine of tModules into tLabFileName
    put stripper(tLabFileName) into tLabFileName
    set the itemDelimiter to "."
    put item 1 of tLabFileName into tLabName

    put "/" & tLabName after tModulesFolderPath

    if there is a directory tModulesFolderPath then

      # lab directory exists
      put tModulesFolderPath into tLabStackPath
      put "/" & tLabFileName after tLabStackPath

      if there is a stack tLabStackPath then
        go stack tLabStackPath
      else
        answer "no local lab file found"
        exit getLabOnClient
      end if

    else
      answer "no local lab folder found"
      exit getLabOnClient
    end if

  else
    answer "local lab file list not found"
    exit getLabOnClient
  end if

end getLabOnClient
```

Button script “server”

```
global gPathToEngineFolder
global gConnectStatus
global gModules
global gLoggerURL
global gHistory

on mouseUp

  # xxx do not set the cClickLine of this button here

  set cursor to watch
  put the label of me into tServerName
  switch tServerName
  case "San Diego"
    put "http://reactorlab.net/ERC_courses/San_Diego_server/" into tServerURL
    break
  case "Arizona"
    put "http://reactorlab.net/ERC_courses/Arizona_server/" into tServerURL
    break
  end switch

  if gConnectStatus is "true" then
    # get info from server
    getLabListFromServer tServerName, tServerURL
  else
    # get info from client
    getLabListFromClient tServerName, tServerURL
  end if

end mouseUp

# ----------------------

on getLabListFromClient tServerName, tServerURL

set cursor to watch

  put empty into field 1
  put empty into gModules

  # now check for local labs

  put gPathToEngineFolder into tModulesFolderPath
```

```
put "/support" after tModulesFolderPath
if there is a directory tModulesFolderPath then
  # support directory exists
else
  create directory tModulesFolderPath
end if

put "/labs" after tModulesFolderPath
if there is a directory tModulesFolderPath then
  # labs directory exists
else
  create directory tModulesFolderPath
end if

put "/" & tServerName after tModulesFolderPath

if there is a directory tModulesFolderPath then
  # server directory exists
else
  # create directory tModulesFolderPath
  put "Work on line to get server list." into field "list"
  exit getLabListFromClient
end if

put tModulesFolderPath into tModuleListPath
put "/module_list.txt" after tModuleListPath
put "file:" & tModuleListPath into tModuleListURL
if there is a file tModuleListPath then
  # check server lab list for local labs and versions
  put url tModuleListURL into tModules
  # format of each line currently (module name for display)/(file name of module)
  set the itemDelimiter to "$"
  repeat with i = 1 to the number of lines in tModules
    put stripper(line i of tModules) into temp
    if temp is empty then
      exit repeat
    else
      put stripper(item 1 of line i of tModules) into tlab
      put stripper(item 4 of line i of tModules) into tlocal
      put stripper(item 5 of line i of tModules) into tage
      put tlab && tlocal && tage into line i of field "list"
      put tServerURL & stripper(item 2 of line i of tModules) into line i of gModules
    end if
  end repeat
else
```

```
    # no file module_list.txt
    put "go on line to get server list" into field "list"
    exit getLabListFromClient
  end if

end getLabListFromClient

# ----------------------

on getLabListFromServer tServerName, tServerURL

  set cursor to watch

  put empty into field 1
  put empty into gModules

  put tServerURL & "module_list.txt" into labURL
  get url labURL

  put it into tModules

  put wordOffset("error", tModules) into tErrorCheck
  if tModules contains "<html>" and tErrorCheck > 0 then
    put word (tErrorCheck + 1) of tModules into tErrorNumber
    put offset("<", tErrorNumber) into tStarter
    if tStarter > 0 then
      put the number of characters in tErrorNumber into tEnder
      delete char tStarter to tEnder of tErrorNumber
    end if
    put "Error " & tErrorNumber into line 1 of field "list"
    if tErrorNumber is "404" then
      put "File not found" into line 2 of field "list"
    end if
    exit getLabListFromServer
  end if

  # format of each line currently (module name for display)/(file name of module)
  set the itemDelimiter to "$"
  repeat with i = 1 to the number of lines in tModules
    put stripper(item 1 of line i of tModules) into temp
    if temp is empty then exit repeat
    else
      put temp into line i of field "list"
      put tServerURL & stripper(item 2 of line i of tModules) into line i of gModules
    end if
  end repeat
```

```
  # now check for local labs

  put gPathToEngineFolder into tModulesFolderPath

  put "/support" after tModulesFolderPath
  if there is a directory tModulesFolderPath then
    # support directory exists
  else
    create directory tModulesFolderPath
  end if

  put "/labs" after tModulesFolderPath
  if there is a directory tModulesFolderPath then
    # labs directory exists
  else
    create directory tModulesFolderPath
  end if

  put "/" & tServerName  after tModulesFolderPath

  if there is a directory tModulesFolderPath then
    # server directory exists
  else
    create directory tModulesFolderPath
  end if

  put tModulesFolderPath into tModuleListPath
  put "/module_list.txt" after tModuleListPath
  put "file:" & tModuleListPath into tModuleListURL
  if there is a file tModuleListPath then
    # check server lab list for local labs and versions
    put url tModuleListURL into tLocalLabList
    set the itemDelimiter to "$"
    repeat with i = 1 to the number of lines in tLocalLabList
      if stripper(item 4 of line i of tLocalLabList) contains "(local)" then
        put stripper(item 2 of line i of tLocalLabList) into tLocalLab
        put stripper(item 3 of line i of tLocalLabList) into tLocalVersion
        put lineOffset(tLocalLab, tModules) into tline
        if tline = 0 then
          # local version but not on server anymore
          put the number of lines of tModules into tnlines
          # need to have notation in line below as (local) (only) not (local only)
          put stripper(item 1 of line i of tLocalLabList) & " (local) (only)" into line tnlines+1 of
field "list"
          put line i of tLocalLabList into line tnlines+1 of tModules
```

```
        else
          put " $ (local)" after line tline of tModules
          put " (local)" after line i of field "list"
          put stripper(item 3 of line tline of tModules) into tServerVersion
          if tLocalVersion < tServerVersion then
            put " $ (old)" after line tline of tModules
            put " (old)" after line i of field "list"
            put space & tLocalVersion into item 3 of line tline of tModules # for local list
          end if
        end if
      end if
    end repeat
  else
    # no lab list for this server, assume no local labs
  end if

  put tModules into url tModuleListURL

  put cr & the time && tServerURL after gHistory

end getLabListFromServer



BUTTON: “Quit”
global gConnectStatus
global gLoggerURL
global gHistory

on mouseUp

   set cursor to watch

    if gLoggerURL is not empty then
      set cursor to watch
      set the httpHeaders to "Content-type: application/x-www-form-urlencoded" & return
      put cr & the time & " exit " & the short name of this stack after gHistory
      if gConnectStatus is true then
        post gHistory to url gLoggerURL
      end if
    end if

  quit

end mouseUp
```

BUTTON “New Button”

```
on mouseUp
  revMail "herz@ucsd.edu",, "ERC Web Courses"
end mouseUp
```

BUTTON “ercMenuLogo.jpg

```
# on mouseUp
  # launchInternetApp "http://www.erc.arizona.edu/"
# end mouseUp
```

BUTTON “on line”

```
on mouseUp

  global gConnectStatus
  global gHomeURLDefault

  # set hilites here so can just send mouseUp to these buttons
  set the hilite of me to true
  set the hilite of button "off line" to false
  put "true" into gConnectStatus

  if the cFirstConnectMade of this stack is "false" then
    # get an update stack from server on first connect
    # to check for updates of files in the support/scripts folder
    set the cFirstConnectMade of this stack to "true"
    put gHomeURLDefault into tUpdateStackURL
    set the itemDelimiter to "/"
    if the last item of tUpdateStackURL contains ".rev" then
      delete the last item of tUpdateStackURL
    end if
    put "/wc_updater.rev" after tUpdateStackURL
    put cr & the time && tUpdateStackURL after gHistory
    # send the loadGoStart because want this handler to end since my replace this
stack
    # need quote literals around tUpdateStackURL below or gets clipped at : in
http:
    # xxx specific name
    put the stacksInUse into tStacks # need to check if using default comm scripts
    put lineOffset("comm scripts", tStacks) into tline
    put line tline of tstacks into tCommStack
```

```
    send loadGoStart && quote & tUpdateStackURL & quote to stack tCommStack in 10
ticks # so this script will end
  else
    send mouseUp to button "server"
  end if
end mouseUp
```

BUTTON “off line”

```
on mouseUp
  global gConnectStatus
  # set hilites here so can just send mouseUp to these buttons
  set the hilite of me to true
  set the hilite of button "on line" to false
  put "false" into gConnectStatus
end mouseUp
```

BUTTON “get info”

```
on mouseUp
  go stack "directory info"
end mouseUp
```