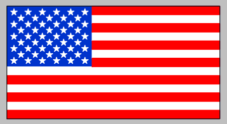
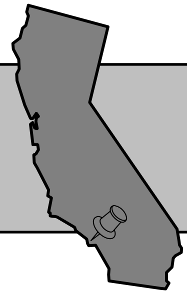


DECwindows Session Internals



From VMS Boot to the Session Manager

DECUS U.S. Chapter Fall 1991 Symposium

Session UI044 - Wednesday 6PM - A7 Convention Center

Presented By

John "Fast-Eddie" McMahon

TGV, Incorporated

603 Mission Street

Santa Cruz, California 95060

800-848-3440

MCMAHON@TGV.COM

DECwindows Session Internals

Here there be dragons

> or <

To boldly go where no one has gone before

- That is, unless you work for Digital

DECwindows Session Internals

Administrative Trivia

- The original outline for this talk consisted of a "cradle to grave" analysis of DECwindows Session Internals. Unfortunately, due to time constraints, we will only cover from System Startup through the execution of the session manager.
- A significant amount of this talk deals with Unsupported, Undocumented, Uneverything features of VMS DECwindows. Use what you learn at your own peril.
- All the information for this talk was retrieved from the VMS Listings CD-ROM (which you can buy from Digital).
- This session covers DECwindows Version 2 (VMS 5.4). It does not cover DECwindows Motif!

DECwindows Session Internals

Outline

Startup Internals

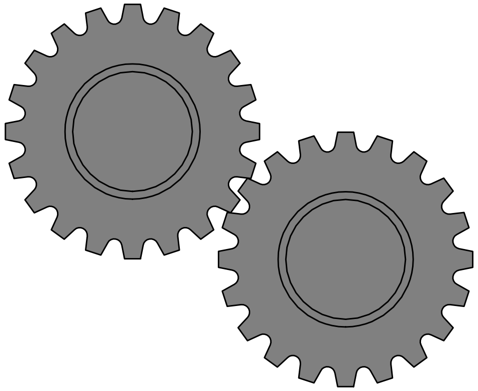
- From system boot to first invocation of DECwindows LOGINOUT
  - Startup command procedures
  - DECW$STARTLOGIN.EXE
    - Invoking LOGINOUT in an DECwindows environment
- The role of the Workstation device (WSAnnnn:)

DECwindows Session Internals

Outline (Continued)

Session Creation Internals

- Entering your password and doing real work
  - LOGINOUT.EXE
  - DECW$LOGINOUT extensions to LOGINOUT
    - LOGO processing
    - Server initialization
    - Window Manager Startup
    - Session Manager Startup
- Questions and (hopefully) Answers



Startup Internals

DECwindows Startup Command Procedure Flow

- SYS$MANAGER:DECW$STARTUP.COM calls
  - SYS$MANAGER:DECW$LOGICALS.COM
  - SYS$MANAGER:DECW$DEVICE.COM
  - SYS$MANAGER:DECW$STARTSERVER.COM calls
    - SYS$MANAGER:DECW$LOGICALS.COM
    - SYS$MANAGER:DECW$RGB.COM
  - SYS$MANAGER:DECW$CHECK_PARAMS.COM
  - SYS$MANAGER:DECW$STARTXTERMINAL.COM
  - SYS$MANAGER:DECW$STARTLIBS.COM
  - SYS$MANAGER:DECW$STARTAPPS.COM calls
    - SYS$SYSTEM:DECW$STARTLOGIN.EXE

Startup Internals

SYS$MANAGER:DECW$STARTUP.COM

- Initialization procedure for the DECwindows environment
  - Creates DECwindows specific logical name table (DECW$LOGICAL_NAMES)
    - SYS$MANAGER:DECW$LOGICALS.COM
  - Initializes the VAXstation environment
    - SYS$MANAGER:DECW$DEVICE.COM
  - Starts the DECwindows server on workstations
    - SYS$MANAGER:DECW$STARTSERVER.COM
  - Checks SYSGEN parameters
    - SYS$MANAGER:DECW$CHECK_PARAMS.COM

Startup Internals

SYS$MANAGER:DECW$STARTUP.COM

- Installs the DECwindows transport (and other) images
- Installs the Display information pseudodevice (WSA0:)
- Starts the Xwindows terminal software
  - SYS$MANAGER:DECW$STARTXTERMINAL.COM
- Installs the DECwindows libraries
  - SYS$MANAGER:DECW$STARTLIBS.COM
- Installs the DECwindows applications
  - SYS$MANAGER:DECW$STARTAPPS.COM

Startup Internals

SYS$MANAGER:DECW$STARTUP.COM

- Potentially useful internals
  - Defining the logical DECW$IGNORE_DECWINDOWS to any value will cause DECwindows to abort startup
  - DECwindows requires BYPASS, PRMGBL, SYSNAM, DETACH, PSWAPM, ALTPRI, NETMBX, TMPMBX, PRMMBX, SYSPRV, CMKRNL, PFNMAP, SYSGBL, WORLD, OPER AND LOG_IO privileges to execute
  - DECW$STARTUP.COM typically resubmits itself as a subprocess (SPAWN/NOWAIT) and exits. The process name "DECW$STARTUP" is used. This behavior can be cancelled by defining the logical DECW$IGNORE_SUBPROCESS to any value

Startup Internals

SYS$MANAGER:DECW$STARTUP.COM

- Useful internals (continued)
  - Defining the logical name DECW$IGNORE_WORKSTATION to any value will cause DECwindows to not execute any steps involving VAXstation initialization
    - Server will not be initialized
    - DECW$DEVICE is skipped
    - DECW$CHECK_PARAMS is skipped
    - DECW$STARTLOGIN.EXE will not be executed in DECW$STARTAPPS.COM

Startup Internals

SYS$MANAGER:DECW$STARTUP.COM

- Useful internals (continued)
  - The logical name DECW$LICENSE_CHECK is created prior to DECwindows startup. This logical name contains a Boolean value indicating whether or not the workstation is properly licensed. If FALSE, the effect is similar to setting DECW$IGNORE_WORKSTATION

Startup Internals

SYS$MANAGER:DECW$STARTUP.COM

- Useful internals (continued)
  - The logical name DECW$IGNORE_AUTOGEN can be defined to any value if you wish to skip the AUTOGEN checking done when a server is started
  - DECW$STARTUP accepts 5 parameters (P1-P5)
    - P1 = "" (Execute all steps of DECwindows startup)
  - When P1 = RESTART
    - Restart the DECwindows server
    - Reinstall the DECwindows libraries (DECW$STARTLIBS.COM)
    - Reinstall the DECwindows applications (DECW$STARTAPPS.COM)
    - P2-5 are passed to STARTSERVER
    - P2 is passed to STARTLIBS and STARTAPPS

Startup Internals

SYS$MANAGER:DECW$STARTUP.COM

- When P1 = SERVER/APPS/LIBS/XTERMINAL
  - Execute that step of DECwindows startup
  - P2 is passed to the appropriate command procedure

Startup Internals

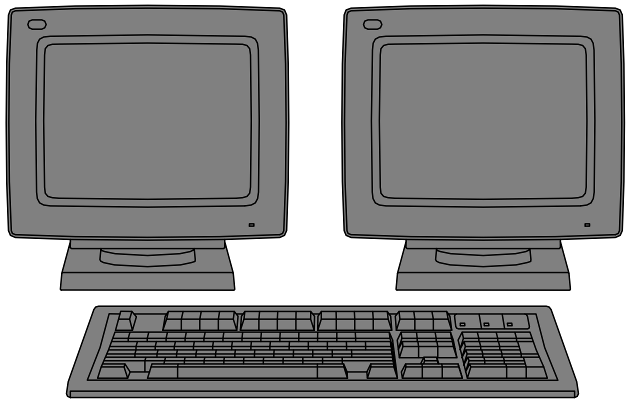
SYS$MANAGER:DECW$DEVICE.COM

- VAXstation initialization
  - Identifies type of CPU
  - SYSGEN AUTOCONFIGUREs output graphics drivers
    - GA, GB, GC, GD, GE, GF drivers
  - Defines global symbols
    - Keyboard, Pointer/Mouse device
    - Screen size
    - Color support
    - Monitor/Server density per inch
    - Fonts
    - Keymaps

Startup Internals

SYS$MANAGER:DECW$DEVICE.COM

- Basic support for "dual-head" VAXstations
  - Capability to have two screens (one Color, one B/W) on the same workstation
  - Requires two monitors and a special cable
  - Color screen hardware already has capability to handle two heads
- Loads DECwindows device drivers
  - IKDRIVER (Keyboard)
  - IMDRIVER (Mouse)
  - INDRIVER (Common)



Startup Internals

SYS$MANAGER:DECW$DEVICE.COM

- Potentially useful internals
  - DECwindows support for "new" VAX systems turns up here, sometimes before the processor is released

Startup Internals

SYS$MANAGER:STARTSERVER.COM

- Getting the DECwindows server running
  - Create server-specific logical name table (DECW$SERVERn_TABLE)
    - SYS$MANAGER:DECW$LOGICALS.COM
  - Create named color logical name table
    - Mapping between color names and RGB values for each color
    - SYS$MANAGER:DECW$RGB.COM
  - Define logicals
    - Fonts, Server devices and capabilities, Keymaps, Available transports,Display Postscript extension
  - Execute the server as a detached process

Startup Internals

SYS$MANAGER:STARTAPPS.COM

- Responsible for installing applications
- Starts DECwindows LOGINOUT application on workstations
  - Issues SET DISPLAY command to create a WS device using transport LOCAL with logical name pointer "DECW$STARTUP_DEVICE". This is later equated to "DECW$DISPLAY" for use by DECW$STARTLOGIN.EXE
  - The logical "DECW$DEFAULT_TRANSPORT" can be used to establish a different default transport for the WS device
  - Executes SYS$SYSTEM:DECW$STARTLOGIN.EXE

Startup Internals

SYS$SYSTEM:DECW$STARTLOGIN.EXE

- Triggers DECwindows LOGIN application by sending message to the Job Controller
  - Requires CMKRNL
- Source code is on the VMS Listings kit
  - Disk 2, Directory [V54.DECW$APPS.LOGINEXT.LIS], File START_LOGIN.LIS, Language: MACRO-32

Startup Internals

SYS$SYSTEM:DECW$STARTLOGIN.EXE (Program Flow)

→ Try to translate DECW$DISPLAY in the JOB logical name table

→ If found then goto 20$

→ Try to translate DECW$DISPLAY in the PROCESS logical name table

→ If error (e.g. DECW$DISPLAY is not defined) then print error and exit

Startup Internals

SYS$SYSTEM:DECW$STARTLOGIN.EXE

- → 20$:
- → Assign a channel to DECW$DISPLAY device (e.g. WSA49:)
- → If error (e.g. Device doesn't exist) then print error and exit
- → Issue a $GETDVI call on the device
  - — Fetch DEVCLASS, DEVTYPE and DEVNAM
- → If error then print error and exit
- → If the device class is not DC$_WORKSTATION then goto 30$
- → If the device type is not DT$_DECW_PSEUDO then goto 30$
- → Issue a $QIO call on the device to get the WS device flags

Startup Internals

SYS$SYSTEM:DECW$STARTLOGIN.EXE

→ If error then print error and exit

→ If the NORESTART flag is not set on the device then goto 40$

→ Issue a $QIO call on the device to clear the permanent bit on the device

→ If error then print error and exit

→ 30$:

→ goto 90$

Startup Internals

SYS$SYSTEM:DECW$STARTLOGIN.EXE

→ 40$:

→ Call the routine "get_wkst_ucb" to get the Unit Control Block of the WS device

→ If error then print error and exit

→ Call the routine "sndmsg". This sends a message to the job controller to start LOGINOUT using the WS device as input

→ 90$:

→ Exit

The Role of the Workstation Device

A important but hidden part of DECwindows

- It is a "Pseudo-Device". It does not drive hardware
  - Device WSAnnnn: (uses WSDRIVER)
  - Device class DC$_WORKSTATION
  - Device type DT$_DECW_PSEUDO
- Stores information about where X clients should direct their output to
  - Transport Name, Node Name, Server Number, Screen Number

The Role of the Workstation Device

A important but hidden part of DECwindows

- User typically manipulates a WS device through use the SET DISPLAY command
  - The logical name DECW$DISPLAY always points to a WS device
  - There is an undocumented DCL and $QIO interface to WSDRIVER. See the session notes for more details
- When a Workstation device is used as SYS$INPUT/OUTPUT/ERROR to LOGINOUT a DECwindows session is started

Session Creation Internals

SYS$SYSTEM:LOGINOUT.EXE

- Application for handling all functions required to log in to/out of VMS
- Routines for Interactive, Batch, Network, Other class logins
- DECwindows logins are handled as a special class of Interactive logins
- Calls the sharable library SYS$LIBRARY:DECW$LOGINOUT.EXE to access DECwindows specific login code

Session Creation Internals

SYS$SYSTEM:LOGINOUT.EXE

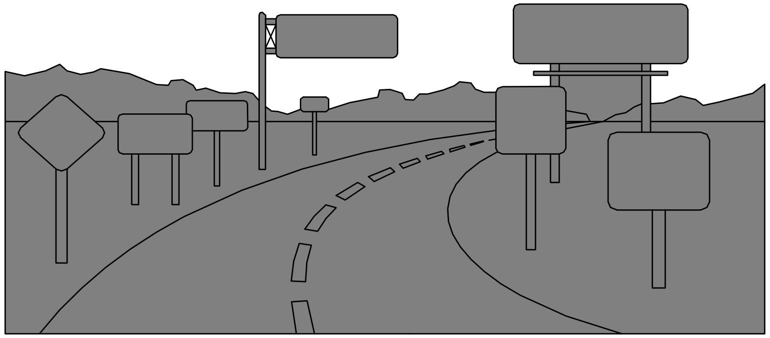
- Source code is on the VMS Listings kit
  - Disk 1, Directory [V54.LOGIN.LIS], Languages: BLISS and MACRO

Session Creation Internals

SYS$SYSTEM:LOGINOUT.EXE

There is a signpost up ahead...

"We will be concentrating on the parts of LOGINOUT that influence DECwindows session creation. LOGINOUT is an amazing module, it could take hours to explain all of it’s secrets."



Session Creation Internals

LOGINOUT (Program Flow by routine/module/facility)

Start/Login/Loginout

$\downarrow$

Login/Login/Loginout

- This happens to be the easy part

Session Creation Internals

LOGINOUT (Program Flow Con't)

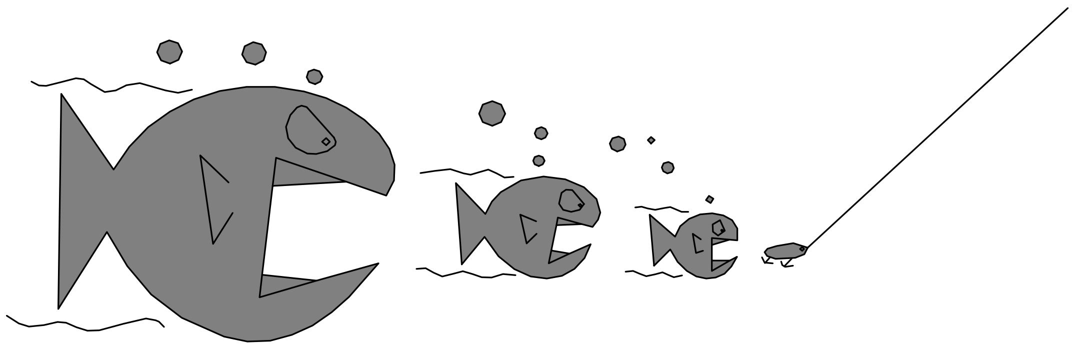
Login/Login/Loginout

- ⇨ Check_For_Decwindows/Login/Loginout
  - Note: A white arrow indicates the routine is directly called by "Login"
- ➡ Load_Decwindows_Routines/Decwindows/Loginout
  - Note: A black arrow indicates the routine is called by the routine above it (1 call below "Login")
- ⇨ Decw$Setup_Operator/Decw$Setio/Decw$Loginout
- ⇨ Open_Output/Fileio/Loginout
- ⇨ Decw$Init/Decw$Interact/Decw$Loginout
- ➡ Decw$Start_Logo/Decw$Logo/Decw$Loginout

- Create_Logo/Decw$Logo/Decw$Loginout
  - Note: A long arrow indicates the routine is called by the one directly above (2 calls below "Login")
- Init_Interactive/Interact/Loginout
- Interactive_Validation/Interact/Loginout
- Decw$Interactive_Validation/Decw$Interact/Decw$Loginout
- Δ Decw$Create_Dialog/Decw$Createdi/Decw$Loginout
  - Note: A delta symbol indicates the routine is called by the one directly above (3 calls below "Login")
- Decw$Stop_Logo/Decw$Logo/Decw$Loginout
- Init_User/Inituser/Loginout
- Init_Cli/Inituser/Loginout
- Decw$Setup_Sys$Input/Decw$Setio/Decw$Loginout

- Decw$Setup_Sys$Output/Decw$Setio/Decw$Loginout
- Decw$Init_Workstation/Decw$Initworkstation/Decw$Loginout
- Utilities module routines for getting username/authorized node
- Utilities module routines for setting the access lists
- Init_Property/Initpropery/Decw$Loginout
- Start Window Manager routines

End Of Loginout/Session Manager Startup



Session Creation Internals

LOGINOUT Routine "Start"

- Located in module "Login", Language BLISS
- Image execution starts here
- Issue a $GETDVI on SYS$INPUT
  - Fetch DEVCLASS, DEVCHAR, DEVCHAR2, DEVDEPEND, DEVDEPEND2, DEVNAME, TT_PHYDEVNAM, TT_ACCPORNAM
- Test for device class equal to DC$_WORKSTATION

Session Creation Internals

LOGINOUT Routine "Start" (continued)

- If true
  - Set ws_device = true, devdepend2 = 0
  - Setting the ws_device variable will cause "Check_For_DECwindows" to invoke the code for setting up the Session Manager environment
  - Call "Login"
  - Image execution ends here

Session Creation Internals

LOGINOUT Routine "Login"

- Located in module "Login", Language: BLISS
- Translate (in LNM$PROCESS) SYS$INPUT, SYS$OUTPUT, SYS$ERROR
- Call "check_for_decwindows"
- We will be back here soon...

Session Creation Internals

LOGINOUT Routine: "check_for_decwindows"

- Located in module "Login", Language: BLISS
- Test for ws_device equal to true and Process type equal to Interactive
- If true
  - Try loading the DECwindows LOGINOUT extensions (call "load_DECwindows_routines")
  - Successful? Then set DECwindows = true, decw$login = true
  - Create DECW$DISPLAY logical name for later use in LNM$PROCESS (LNM$JOB doesn’t exist yet)

Session Creation Internals

LOGINOUT Routine: "check_for_decwindows"

- If true (con’t)
  - Copy WS device name into variable decw$workstation
  - Note: This code starts the setup for the invocation of the Login window and Session Manager later on
- Test for SYS$ERROR equated to a Workstation device (DC$_WORKSTATION)
- If true
  - Note: DECterm sessions invoke LOGINOUT with SYS$INPUT/OUTPUT equated to a pseudo-device and SYS$ERROR pointing to a WS device
  - Set decw$login = true

Session Creation Internals

LOGINOUT Routine: "check_for_decwindows"

- If true (con’t)
  - Note: DECwindows was not set True in this block. The DECwindows variable is specific to Session Manager startup
  - Copy WS device name into variable decw$workstation
  - Redefine SYS$ERROR to point to SYS$INPUT
  - Set the job type to LOCAL, regardless of how it was originally set
  - Return (back to "Login")

Session Creation Internals

LOGINOUT Routine
"load_decwindows_routines"

- Located in module: "Decwindows", Language: BLISS
- Dynamically loads the following routines from DECW$LOGINOUT using LIB$FIND_IMAGE_SYMBOL
  - Decw$Init, Decw$Stop_Logo, Decw$Setup_Sys$Input, Decw$Setup_Sys$Output, Decw$Setup_Operator, Decw$Init_Workstation, Decw$Flag_Set_Password, Decw$Flag_Exp_Password, Decw$Putmsg, Decw$Exit, Decw$Init_Cli And Decw$Interactive_Validation

Session Creation Internals

LOGINOUT Routine: "Login" (Continued)

- Return from "check_for_decwindows"
- Test for interactive job, then test for DECwindows flag
- if DECwindows flag then
  - Call DECW$SETUP_OPERATOR
    - Tests for existance of logical DECW$LOGIN_LOG in table DECW$LOGICAL_NAMES
    - If it exists set SYS$OUTPUT to point to it and set logging_output = 1
  - If logging_output then Call OPEN_OUTPUT
    - Open the output file, if failure open NLA0: for output
  - Call DECW$INIT
- Once again... we will be back after these subroutines

Session Creation Internals

Excuse me... would the person in the 13th row please refrain from snoring, you are keeping everyone awake!

Session Creation Internals

DECW$LOGINOUT.EXE Sharable Library

- All source code is on the VMS Listings kit
  - Disk 2, Directory [V54.DECW$APPS.LOGINEXT.LIS]
- Modules written in C or Macro

Session Creation Internals

DECW$LOGINOUT Routine: "DECW$SETUP_OPERATOR"

- Located in Module: DECW$SETIO
- Useful Internal:
  - Output logical for DECwindows LOGINOUT "DECW$LOGIN_LOG"
  - Define in DECW$LOGICAL_NAMES table
  - Prevents messages to OPA0:
  - Allows you to see errors that cause LOGINOUT to fail (e.g. Unable to open display) in a file

Session Creation Internals

DECW$LOGINOUT Routine: "DECW$INIT"

- Located in Module: DECW$INTERACT, Language: C
- Tests for definition of logical "DECW$LOGIN_MULTIPLE" in LNM$SYSTEM
  - If not defined, opens display and counts the number of active windows. If greater than zero then DECW$EXIT is called and LOGINOUT ultimately exits
  - According to the comments, you can thank the US Air Force for this "feature"

Session Creation Internals

DECW$LOGINOUT Routine: "DECW$INIT"

- Initializes DECwindows toolkit
  - Resource file DECW$SYSTEM_DEFAULTS:DECW$LOGIN.DAT
  - UID File DECW$SYSTEM_DEFAULTS:DECW$LOGIN.UID
    - Source UIL is in the listings kit
- Creates error message dialog box
- Calls DECW$START_LOGO
- Returns information on the LOGO process, so it can be killed off later if needed

Session Creation Internals

Useful internal hook in DECW$INIT

- Logical name DECW$LOGIN_MULTIPLE
  - Ability to run LOGINOUT on an X terminal/workstation that already has an active window manager (or other windows) on it
  - Can cause some weird problems, most notably the possibility of multiple LOGINOUT windows "fighting" over the same display. DECW$LOGINOUT checks for multiple LOGINOUT windows by merely counting windows. A better method would be to read the window names in the hirearchy looking for similar widget names.
  - DECwindows LOGINOUT will silently fail if a window exists on the server and this logical is not defined.

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Start_Logo"

- Located in Module "Decw$Logo", Language: C
- Test for definition of DECW$LOGIN_BACKGROUND in LNM$SYSTEM
- If it doesn’t exist, call CREATE_LOGO to bring up the familiar DIGITAL logo

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Start_Logo"

- If it exists, kick off a detached process with the following characteristics
  - SYS$INPUT = Translation of DECW$LOGIN_BACKGROUND (command procedure)
  - SYS$OUTPUT = The workstation device (WSAnnnn:)
  - Privs = NETMBX, TMPMBX

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Start_Logo"

- If (con’t)
  - Detached Process, No UAF or PASSWORD bits set
  - Executing LOGINOUT.EXE (This creates a CLI so the command procedure can run)
  - Process Name LOGO
  - Base Priority 4
- Most errors cause the regular DIGITAL logo to be displayed

Session Creation Internals

Issues when writing your own LOGO program

- Any X program can be a LOGO program. Be careful with programs which may allow some sort of input/output from the mouse/keyboard. Remember that the LOGO is operational before the user logs in. Simple non-interactive drawing programs are best for LOGOs

Session Creation Internals

Issues when writing your own LOGO program

- If you run multiple LOGINOUT sessions on your node (Do you have a few X Terminals running on this system ?) be advised that a simple command procedure that just executes a RUN command will not work
  - Remember that LOGOs are invoked with a "fixed" process name under the SYSTEM account
  - First LOGO process that is running will "capture" the process name and prevent further LOGO processes from being created
  - Solution: Use a command procedure that changes the process name to something else before starting the LOGO program

Session Creation Internals

LOGINOUT.EXE Routine: "Login" (Con't)

- Return from "DECW$INIT"
- Set account name to "NULL<login>"
- Call "init_interactive" in module INTERACT
  - Set username to "<login>"
  - Set job type to LOCAL
  - System password processing is skipped
  - Auto login processing is skipped

Session Creation Internals

LOGINOUT.EXE Routine: "Login" (Con't)

- "interactive_validation" in module INTERACT
  - Called from "init_interactive"
  - Disables normal LOGINOUT auditing routines in favor of DECwindows extensions
  - Calls DECW$INTERACTIVE_VALIDATION
  - Sets username to proper value
  - May returns errors related to bad password, etc.
  - Special test for captive accounts and DECwindows (this is disallowed)
- More details in our next exciting episode...

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Interactive_Validation"

- Located in Module: "DECW$INTERACT.LIS", Language: C
- Calls "decw$create_dialog" in DECW$CREATEDI to create the LOGIN box if needed
  - Special callback for the return key
  - Event handler to prevent login box being occluded by other windows
  - Box is always centered on the screen
  - No echo in password widget

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Interactive_Validation"

- Calls back to LOGINOUT routines to determine if password/username are valid

Session Creation Internals

LOGINOUT in Routine: "Login" (Yet more stuff)

- Having survived validation, the LOGIN process continues...
- The LOGO process/widget is removed by calling DECW$STOP_LOGO
- The user context is initialized (Routine: "init_user")

Session Creation Internals

LOGINOUT Routine: "Login"

- The CLI is initialized (Routine: "init_cli")
  - If decwindows is set (session mgr activation) then
    - SYS$SYLOGIN will not be executed
    - LOGIN.COM (or equivalent) will not be activated
    - SYS$MANAGER:DECW$SYLOGIN.COM will be executed
    - DECW$LOGIN.COM in SYS$SYLOGIN will be executed (subject to /NOCOMMAND)
    - SYS$MANAGER:DECW$STARTSM.COM will be executed (session manager)
  - Otherwise, the "normal" SYLOGIN/LOGIN sequence occurs

Session Creation Internals

LOGINOUT Routine: "Login"

- DECW$SETUP_SYS$INPUT in DECW$SETIO is called
  - Sets input to SYS$MANAGER:DECW$STARTSM.COM
- DECW$SETUP_SYS$OUTPUT in DECW$SETIO is called
  - Sets up Session Manager output file
  - Checks for logical name DECW$SM_CREATE_LOG in DECW$LOGICAL_NAMES
  - If defined to FALSE, no session manager output file
  - Otherwise Session Manager SYS$OUTPUT equated to default:[DEFAULT]DECW$SM.LOG

Session Creation Internals

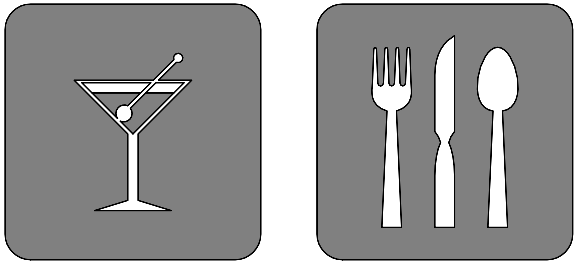
LOGINOUT Routine: "Login"

- If a file was opened for LOGINOUT errors (DECW$LOGIN_LOG) it is closed at this point
- If decw$login is set then DECW$DISPLAY is created in LNM$JOB (exec mode) to point to decw$workstation
  - These variables were loaded in the "check_for_decwindows" routine a LONG time ago
  - Placing DECW$DISPLAY in JOB and PROCESS allows a user to modify the PROCESS to suit whatever needs for remote displaying are required without harming the JOB logical name, which is used to restart the LOGIN application during session shutdown

Session Creation Internals

LOGINOUT Routine: "Login"

- DECW$INIT_WORKSTATION is called
  - Window manager startup
- Almost (but not quite) done with "Login"
- Is it time for dinner yet ?



Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- Located in Module: "Decw$Initworkstation"
- Fetches the username
- Calculate the "authorized node" using the following algorithm (buried in various routines in the UTILITIES module)
  - Fetch the nodename from the DECW$DISPLAY device, either 0:: (if DECW$DISPLAY is undefined) or NODENAME:: is returned. Presumably NODENAME:: is returned for TCP/IP nodes as well (e.g. MILL.YOYODYNE.COM::)

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- Authorized node (con’t)
  - If 0:: return this as the "authorized node"
  - Find out the local nodename by translating SYS$NODE
  - If that fails, local nodename is retrieved by calling $GETSYI
  - If that fails, local nodename is "Unknown::"
  - If local nodename is "Unknown::" then return 0::
  - If the DECW$DISPLAY nodename equals local nodename then return 0::
  - If we get this far, return the local system nodename

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- Calculate the transport in use. If DECW$DISPLAY is missing for some reason, use "LOCAL"
- If TRANSPORT is LOCAL, check for enough global pages/sections. If not enough then display an warning message
- Set the cursor on all root windows to the "Wait" cursor
- Unmanage/Unmap all windows on the screen. This is for neatness as the Window Manager starts and the LOGO program/Login application shut down

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- Enable Security on the Server ("add_current_user in UTILITIES")
- Add username (or "?" for TCPIP transport), nodename, transport to the trusted access list ("change_trusted_list" in UTILITIES)
  - This is the list that allows a user to change the access list
  - Changing the trusted list on the server involves prefixing the transport with "_" instead of the usual XAddHost call
  - DECNET and LOCAL transports are equated here. If one is selected, the othere is added by default

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- Build a security structure for the access list ("add_current_user")
  - This is the list that allows a user to open a window on the server
  - DECNET and LOCAL transports are equated here. If one is selected, the other is added by default
- The danger here is when trying to run LOGINOUT displaying on a non-VMS server. DEC uses a special "family" for it’s security records, most servers do not understand the records

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- Once the Trusted list has been set, the regular access list is CLEARED! The regular access records calculated above are then added to the regular access list
- We finally leave "add_current_user" and return to "decw$init_workstation"
- Resource database is cleared

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- "init_property" in module INITPROPERTY is called
  - Resource files are loaded (from module SMSHARE)
    - Color systems: DECW$SM_COLOR.DAT in SYS$LOGIN/DECW$SYSTEM_DEFAULTS
    - B/W systems: DECW$SM_BW.DAT in SYS$LOGIN/DECW$SYSTEM_DEFAULTS
    - Greyscale systems: DECW$SM_GRAY.DAT in SYS$LOGIN/DECW$SYSTEM_DEFAULTS
    - All systems: DECW$SM_GENERAL.DAT in SYS$LOGIN/DECW$SYSTEM_DEFAULTS and SYS$LOGIN:DECW$XDEFAULTS.DAT

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- Note: SYS$LOGIN is used here, not DECW$USER_DEFAULTS. This prevents you from moving these files elsewhere.
- Info about the window manager is pulled from the recently loaded resource database. If it’s "System Default", the logical name DECW$WINMGREXE in DECW$LOGICAL_NAMES is translated to find the executable name

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- The window manager is started (STARTWM module) on each screen
- Useful Window Manager logical names in LNM$SYSTEM
  - DECW$WINMGR_OUTPUT
    - Window Manager output (Defaults to NL:)
  - DECW$WINMGR_PGFLQUOTA (Default: 5000)
  - DECW$WINMGR_BYTLM (10000)
  - DECW$WINMGR_WSEXTENT (4000)
  - DECW$WINMGR_FILLM (92)

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- Useful WINMGR logical names in LNM$SYSTEM (More!)
  - DECW$WINMGR_DIOLM (100)
  - DECW$WINMGR_ENQLM (600)
  - DECW$WINMGR_JTQUOTA (-1)
  - DECW$WINMGR_WSDEFAULT (512)
  - DECW$WINMGR_WSQUOTA (650)
  - DECW$WINMGR_BIOLM (60)

Session Creation Internals

DECW$LOGINOUT Routine: "Decw$Init_Workstation"

- Useful WINMGR logical names in LNM$SYSTEM (More!)
  - DECW$WINMGR_ASTLM (100)
  - DECW$WINMGR_TQELM (-1)
  - DECW$WINMGR_PSWAPM (Swapping Enabled)
- WINMGR Process name DECW$WM_<WSA device number>
- $CREPRC call made with NETMBX, TMPMBX only
- The server is flushed
- We return to LOGIN

Session Creation Internals

LOGINOUT Routine: "Login"

- LOGINOUT hands control over to the CLI
  - DECW$STARTSM.COM is executed by the CLI
  - Logical name DECW$SM_WSQUOTA can be used to adjust Session Manager WSQUOTA, it defaults to the SYSGEN parameter PQL_WSQUOTA
- After several hundred lines of code and maybe 30 seconds of real time the Session Manager starts up
  - DECW$SYLOGIN.COM and DECW$LOGIN.COM are executed
  - User's "Autostart" applications fire up

DECwindows Session Internals

Questions and Answers

Thank you very much for coming!