DON'T USE THE M WORD

Tania Dastres and Marcus Ransom
RMIT University

Hashtag : #xw14
Please leave comments on this talk at auc.edu.au/xworld/sessions
Don’t use the M word

Tania Dastres
MacWorks Technical Lead
and
Marcus Ransom
Lead Apple Technician
Don’t use the M word

- RMIT University
- Old IT vs new IT. It’s ALL about the user
- How are we moving towards the new?
- What next?
RMIT University

- Founded in 1887
- Australia’s largest tertiary institution
- 82,000 students
RMIT University

- Campuses in Melbourne CBD, Bundoora and Brunswick
- 2 international campuses in Vietnam
- Research office in Barcelona
RMIT University

- Over 15,000 computers
- At least 2500 Mac OS X*
Using a Mac at RMIT
where we have come from

• Individual college based IT departments
• Mixed teams providing support across platforms
• Labs owned and maintained by colleges/schools
• Some knowledge sharing between colleges
Labs

- Nearly 1300 machines in over 60 Labs
- Monolithic images
- Individual customisation for different spaces
- Network logins
- Administration via ARD
- Moved to Munki and Deploy Studio in 2011
Staff Machines

- Over 1200 machines (exact numbers uncertain)
- Monolithic or no images
- No centralised management
- Local user accounts
- Mix of purchased and leased
- Poor asset tracking
2012

• Centralised ITS
• Client Computing

2013

• Advanced Technologies - Apple Team
Apple Team

- Third level support for Mac OS X and iOS
- Supporting Service Desk and Field Services with level 1 & 2 tasks
- Deployments outsourced
- Project support with experienced Apple technical knowledge
- Casper Suite used to manage 1200 lab machines in 2014
Why change?
What worked and what didn’t

• No management = minimal restrictions
• Excellent specialised and localised support
• Poor skill levels in some areas
• No way of automating updates to staff
• We had NO idea how many machines we had
Managed Operating Environment

that’s an M word right at the start isn’t it?
Don’t use the M word
Old IT
Old style Macintosh management

• Monolithic image - make any changes - do it all again

• Manage configuration and preferences, software updates

• Golden Triangle/Directory Services/MCX/network home directories

• Restricted access to admin privileges

• Goal of consistency
manage ALL the things
The perfect storm
The storm builds

- yearly OS Updates
- installESD
- iCloud integration
- deprecation of MCX
- configuration profiles
- move from MIT to Hemdahl Kerberos
- rewrite of dscl
- document autosave and versions
- iLife app adoption
- client OS Virtualisation
- internet recovery
- recovery HD
Can you see a pattern?

- Free Upgrade
- Mac App Store for standard users
- VPP and DEP
- iWork app adoption
- plist caching
What is going to change next??

- Apple ID for local password
- iCloud Drive
- OS X Beta Program
- watch this space
The New IT

It's all about the User
“You’ve got to start with the customer experience and work back toward the technology - not the other way around”
We are all users

• How would we like our machines set up and administered?
• What would annoy us if someone imposed it on our machines?
• Users are just trying to do their job
Getting buy in from users

• Promote the augmented services

• Don’t focus on the restrictions

• Give them something they have been asking for

• Lead by example
New style IT management

- Design based on needs, not consistency with other platforms or historical policies
- Embracing differences rather than enforcing consistency
- Educate other departments on the requirements of the Mac OS X platform
- Manage once, not always
- Thin provisioning, modular deployment & rapid adoption
- Self service
MacWorks
What is MacWorks?

- Core Configuration
- Core Software/Applications
- Seamless Printing
- Wireless Device Authentication
- Hardware Lifecycle Improvements
- Software License Metering
- Mac Imaging
- Patch Management
- Asset Reporting
- Install & Update Software
- OS Updates
- User Initiated Maintenance & Troubleshoot
- Remote Assistance
- Knowledge Base
- Upskill of ITS Support Teams
## Basic standard configuration

Staff machines are provided with only basic software installed. Users can add anything else they require through self service.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Office</td>
<td>Adobe Flash Player</td>
</tr>
<tr>
<td>iLife + iWork</td>
<td>Java Web Plugin</td>
</tr>
<tr>
<td>Google Chrome</td>
<td>KeyAccess</td>
</tr>
<tr>
<td>Citrix Receiver</td>
<td>Ricoh Drivers</td>
</tr>
<tr>
<td>Casper Self Service</td>
<td>McAfee</td>
</tr>
<tr>
<td>Fetch</td>
<td></td>
</tr>
<tr>
<td>VLC</td>
<td></td>
</tr>
</tbody>
</table>
What configuration DO we perform?

Configurations that enhance rather than restrict

- Global print queue
- Preventing .ds_store
- Local admin for tech support
- Enabling click through at login
- Basic network and local settings
- Skip welcome screen in Safari
- VNC to currently logged in user
- Set Safari home page to RMIT
- Disable iCloud setup prompt
- Device wireless authentication
Transparency about restrictions

• Password protected screensaver timeout - 10 minute with 5 second grace

• Auto login disabled

• Enforced password policy - expiry, complexity and not recycled.
Active Directory

AD login on laptops posed several challenges

• External password resets
• Users seldom log off
• No password reminder at login window since 10.9 (or if FileVault is enabled)
• Introduction of Apple ID password reset
• Departmental shared drives
If we aren’t managing, what ARE we doing?

• Providing services
• Configurations that enhance rather than restrict
• Building a knowledge base
• Providing automated tech support
• Simplified network connectivity
• Self service delivery of software, updates and configuration
• Championing for services to become compatible
• Hidden control with visible customisation
Build communities
Deployment workflows - Staff

• Why image a machine if it comes with a perfectly good OS already?

• No more updating net boot images to suit new hardware / forked builds of OS

Workflow

• Deployment tech boots to recovery HD and runs a script.

• Tech runs some setup policies in self service

• User installs remaining software
Deployment workflows - Staff

Boot to recovery partition and run the following command in terminal

/Volumes/Casper/bootstrap.sh

Contents of Bootstrap.sh script run from USB

#!/bin/sh

# Install Bootstrap package to Macintosh HD
/usr/sbin/installer -package "${0%/*}/Bootstrap.pkg" -target "'/Volumes/Macintosh HD'"

/usr/bin/touch "'/Volumes/Macintosh HD/private/var/db/.AppleSetupDone"

# Restart
/sbin/reboot
Deployment workflows - Staff

Quickadd.plist

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>Label</key>
    <string>au.edu.rmit.quickadd</string>
    <key>LimitLoadToSessionType</key>
    <string>LoginWindow</string>
    <key>Program</key>
    <string>/Library/PrivilegedHelperTools/au.edu.rmit.quickadd.sh</string>
    <key>RunAtLoad</key>
    <true/>
  </dict>
</plist>
Deployment workflows - Staff

Quickadd script run from launchd

```
#!/bin/sh

# Get serial number
SERIAL_NUMBER=$(/usr/sbin/system_profiler SPHardwareDataType | /usr/bin/awk
'/Serial Number \(system\)/ { print $4 }')

# Set computer name
/usr/sbin/scutil --set ComputerName "$SERIAL_NUMBER"

# Install QuickAdd-Transition package
/usr/sbin/installer -package "/Library/PrivilegedHelperTools/QuickAdd-Transition.pkg" -target /
until [ $? -eq 0 ]; do
  /bin/sleep 30
  /usr/sbin/installer -package "/Library/PrivilegedHelperTools/QuickAdd-Transition.pkg" -target /
done

/bin/launchctl load -F -S LoginWindow "/Library/LaunchAgents/au.edu.rmit.bootstrap.plist"

# Cleanup
/bin/rm -r "/Library/PrivilegedHelperTools/QuickAdd-Transition.pkg"
/bin/rm "/Library/LaunchAgents/au.edu.rmit.quickadd.plist"
/bin/rm "$0"
```
Deployment workflows - Staff

Bootstrap.plist

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
.plist version="1.0">
<dict>
    <key>Label</key>
    <string>au.edu.rmit.bootstrap</string>
    <key>Disabled</key>
    <true/>
    <key>LimitLoadToSessionType</key>
    <string>LoginWindow</string>
    <key>Program</key>
    <string>/Library/PrivilegedHelperTools/au.edu.rmit.bootstrap.sh</string>
    <key>RunAtLoad</key>
    <true/>
</dict>
</plist>
Deployment workflows - Staff

Bootstrap policy trigger run from launchd:

```bash
#!/bin/sh
until [ -f /private/var/db/dslocal/nodes/Default/users/taa.plist ]; do
    /usr/sbin/jamf policy -trigger Bootstrap
    [ ! -f /private/var/db/dslocal/nodes/Default/users/taa.plist ] && /bin/sleep 30
done
# Cleanup
/bin/rm "/Library/LaunchAgents/au.edu.rmit.bootstrap.plist"
/bin/rm "$0"
```
Self Service
How are we trying to do this at RMIT?
Work smarter with the tools that we have at our disposal.

Build the kind of tools that users are going to want to use.

Improve the tools you create willingly and often.
Scripts are easy to develop and easy to deploy.

But how can we also make them user-friendly?
There are tools available that give your script a GUI as well as facilitate script-user interaction.

Platypus
CocoaDialog
Pashua

You don't need to be an expert developer or coder to use them!
When our users contact IT Service Desk they are usually asked a series of questions about their Mac.

To gather this information for the Mac can be time consuming and frustrating for both the customer and the IT support person.
How to create a script that displays a summary of this information in one easy to find place.

1. Retrieve information
   
   system_profiler SPHardwareDataType
   
   sw_vers -productVersion
   
   networksetup -listallnetworkservices

2. Make a clickable app
Platypus puts your script in an application bundle and creates the binary to execute it.

There are six output display options:

- None
- Progress Bar
- Text Window
- Status Menu
- Droplet
- Web View
Secure Bundled Script:

- Contents
  - Info.plist
  - MacOS
    - Mac Support Summary
- Resources
  - appIcon.icns
  - AppSettings.plist
  - MainMenu.nib

Without a Secure Bundled Script:

- Contents
  - Info.plist
  - MacOS
    - Mac Support Summary
- Resources
  - appIcon.icns
  - AppSettings.plist
  - MainMenu.nib
  - script

The result:
- Create
- Development Version (symlink to script & files)
- Optimize Application (strip nib file)
- Use XML property list format instead of binary
**Mac Support Summary**

![Mac Support Summary](image-url)
Platypus can let you do some other cool things

**Output Type: Status Menu**

Eg. A script that displays your Mac IP address.

**Output Type: Droplet**

Eg. A script that creates a payloadless package.
Problem

How to provide the benefits of AD without the user needing to log in to an AD-bound account?
Option #1: Pashua

Multiple GUI elements displayed in the one window

Separate display configuration file

Limited text formatting options (use "[return]" for a line break)

Pashua dock item appears by default

Some of Pashua’s 15 available GUI elements:
  Buttons
  Checkboxes
  Images
  Popup list…

For the full list - http://www.bluem.net/en/mac/pashua/
#!/bin/bash
BUNDLEPATH="Pashua.app/Contents/MacOS/Pashua"
PASHUAPATH="/usr/local/$BUNDLEPATH"
FIRST_CONF="first_conf"

pashua_run() {
    # In this case, $1 is first_conf
    pashua_configfile="$1"
    # Pashua does its magic, and returns the
    # resulting user input as one long string.
    result=`"$PASHUAPATH" $pashua_configfile | sed 's/ /;;;/g'`
    # pashua_run then parses this result
    # into variables with the same names as
    # the element name in the conf file
    }

    # User clicked the default OK button
    if [[ $db -eq 1 ]]
    then
        # AD username
        check_eNumber "$enumber"
        fi
Option #2: CocoaDialog

Lets your script display one dialog type after the other.
Fourteen dialog types to choose from
Icons can added to some dialog types only.
And they must be in .icns format.
And they can’t be resized or moved around.
Text cannot include a line break.
Dialog type includes a progress bar and bubbles.
CocoaDialog - the basics

CD_APP="/usr/local/CocoaDialog.app"
CD="$CD_APP/Contents/MacOS/CocoaDialog"

e_number_input=`$CD inputbox --title "RMIT Network Connector" "Enter your RMIT ID" \ 
  --text "e-number" \ 
  --button1 Select \ 
  --button2 Cancel`;

# User input gets submitted as a string so you need to extract out the values you want
declare -a input=($e_number_input);
button=${input[0]};
eNumber=${input[1]};

# User clicked button 1, the OK button
if [ $button -eq 1 ];
then
  # So now ask them for their password
  password_input=`$CD secure-inputbox "RMIT Network Connector" "RMIT ID Password" \ 
    --button1 OK \ 
    --button2 Cancel`;
fi
RMIT Network Connector. So what does it do?

# 1. Confirm that the Mac is on an RMIT network.
checkDatasource=`dscl /Active\ Directory/<Domain> -read Users/$1 | grep "Data source (/Active Directory/<Domain>) is not valid"`

# 2. Get the mount folders ready
if [[ ! -d ~/mount ]]
then
    mkdir ~/mount
    mkdir ~/mount/H
    mkdir ~/mount/K
fi
# 3. Make sure that the dock item Network Drives is there. If it's not, create it!

dock_item_exists=$(echo "$persistent_dock_items" | grep "file-label = Network Drives")
if [[ -z "$dock_item_exists" ]]
then
defaults write com.apple.dock persistent-others -array-add "<dict><key>tile-data</key><dict><key>file-data</key><dict><key>_CFURLString</key><string>/Users/$USER/mount</string><key>_CFURLStringType</key><integer>0</integer></dict><key>file-label</key><string>Network Drives</string><key>file-type</key><integer>18</integer></dict><key>tile-type</key><string>directory-tile</string></dict>"
killall Dock
fi

# 4. Use CocoaDialog to ask the user for their AD username and password.

# 5. Confirm that their username is a valid AD user.
error_check=$(dscl /Active\ Directory/<Domain> -read Users/$eNumber 2>&1 > /dev/null)

# 6. Generate the Kerberos ticket
kinit_result="$(echo "$password" | kinit --password-file="STDIN" "$eNumber"@<Domain> 2>&1 > /dev/null)"

# 7. Do an LDAP query to get the user's H drive address
home_dir=$(ldapsearch -LLL -x -H ldap://<Domain> -D "RMIT\"$eNumber" -b "ou=Accounts,dc=rmit,dc=internal" -w $password cn="$eNumber" | grep "homeDirectory" | sed 's/\"//g')
# 8. Unmount the drives in case they're already mounted
diskutil umount ~/mount/H 2>&1
diskutil umount ~/mount/K 2>&1

# 9. And mount the user's H and K drive
mount_smbfs "$home_dir" ~/mount/H
mount_smbfs //<K drive address> ~/mount/K

# 10. Finally, call the function that displays a completion message
completionMessage "Complete! Your H ($eNumber) and K (University) drives are now available from the Network Drives folder on your dock."

    # Generate a completion message for the user - in this case, using CocoaDialog
    function completionMessage()
    {
        message=`$CD msgbox --icon network --text "RMIT Network Connector" 
                --informative-text "$1" 
                --no-newline 
                --button1 "OK"`
    }
Problem

How to we record RMIT specific information in the JSS?
Mac ID Setup, in four easy steps.

1. Retrieve information

2. JSS API to write this information to the JSS Computer object.

```bash
curl -X PUT -H "Accept: application/xml" -H "Content-type: application/xml" -k -u "$API_USER":"$API_PW" -d "<computer><purchasing><purchasing_account>
$cc_confirmed</purchasing_account></purchasing></computer>" "{jssServer}/JSSResource/computers/udid/$udid"
```
User and Location: Username
If the usage type is Staff Mac the e number is requested.

User and Location: Department
Cost Centre Code has corresponding Department name in csv file script searches through that gets searched for in cc.csv file to look for a corresponding Department string, which gets written to User and Location: Department.

User and Location: Room

Hidden file

General: Asset Tag
4. Progress bar

5. Completion message.
Mac ID Setup

Install staff_Mac ID Setup Resources.pkg

Run Script MacWorks – Mac ID Setup

# Display the MacWorks logo
img.type = image
img.path = /tmp/macworks.gif
img.border = 0
Problem

How do we keep RMIT specific information up to date in the JSS?
Solution: Update Location Details, in five easy steps
1. JSS API
locationXML=$(curl -s -u "$API_USER":"$API_PW" "${jssServer}/JSSResource/computers/udid/$udid/subset/Location")
username=$( echo "$locationXML" | xpath /computer/location/username | sed -e 's/<username>//;s/<\/username>//'
location=$( echo "$locationXML" | xpath /computer/location/room | sed -e 's/<room>//;s/<\/room>//'

2. Create a temporary Pashua configuration file
settings_page=$(mktemp /tmp/settings_conf_XXXXXX)
chmod 755 "$settings_page"

...to display the retrieved values as text

    echo "username.type = text" >> "$settings_page"
    echo "username.label = Username: " >> "$settings_page"
    echo "username.text = $username" >> "$settings_page"

But display the location information (room) as a textfield element

    echo "location.type = textfield" >> "$settings_page"
    echo "location.label = Location (bxxx.yy.zzz):" >> "$settings_page"
    echo "location.default = $location" >> "$settings_page"
4. Progress bar
$CD$ progressbar --indeterminate --title "Updating Location" --text "Please wait..." < /tmp/hpipe &

5. Write the new location/room information back to the JSS
$curl$ -X PUT -H "Accept: application/xml" -H "Content-type: application/xml" -k -u "$API_USER"":"$API_PW" -d "<computer><location><room>$location</room></location></computer>" "${jssServer}/JSSResource/computers/udid/$udid"

6. Completion message.
Links to checkout

Platypus
http://sveinbjorn.org/platypus

Pashua
http://www.bluem.net/en/mac/pashua/

CocoaDialog
http://mstratman.github.io/cocoadialog/
What Next?
What Next?

Improve our custom tools

Add new tools for
- enterprise file vault
- repairs to AD binding and device wireless auth.
- autorun of RMIT Network Connector
- Mac Support Summary auto submits ticket and console logs
- develop a banner showing that their RMIT or local password is about to expire
- provide scoped policies to install specialist printers
- reset local account passwords via self service (with complexity guide)
- leverage the API to deal with individual software license keys (eg VMware Fusion)
- use the API to provide live information to users about software in specific labs
embrace the community

- AUC
- Illuminate.mx
- Sydney MacAdmins
- MacBrained.org

- MacEnterprise
- JamfNation
- IRC ##osx-server
- Twitter #macadmin
Create the community

• RMIT is hosting /dev/world/ in September this year in partnership with AUC
• January 2015?
Questions?

tania.dastres@rmit.edu.au
marcus.ransom@rmit.edu.au
twitter #xw14
Thank you