macOS Server

Service Migration Guide v1.3

March 2021

Contents

Introduction

macOS Server is changing to focus more on management of computers, devices, and storage on your network. As a result, some changes are coming in how Server works. Beginning in the spring of 2018, several services will be hidden on new installations of an update to macOS Server. Then in the fall of 2018, new installations and upgrades of macOS Server will require you to migrate most services to other software.

This guide is designed to assist those administrators comfortable with installing and maintaining open source projects to migrate their service data to the underlying open source project that was previously bundled with macOS Server. Others may want to migrate to cloud-based solutions from third-party vendors. For more information, see the Apple Support article, Prepare for changes to macOS Server at: https://support.apple.com/HT208312.

NOTE: This document will be updated with additional service migration tasks. For updates to this document, see the macOS Server Overview page in Server Help at: https://help.apple.com/serverapp/mac/5.6/#/apdAC95F5C4-27AA-4378-8BB9-95A79A64AB5D

Before you begin

Back up your Mac

You should back up the Mac on which macOS Server is installed. This includes the /Library/Server/ folder and its contents.

Migration prerequisite

The tasks in each chapter are designed for services that are already configured and turned on. If you haven't turned on a specific service, for example, VPN, the VPN migration tasks won't work.

Password types

The following services require users whose password type is RECOVERABLE:

- VPN
- RADIUS

If macOS Server is installed and configured, you can use it to create additional users whose password type can be set to RECOVERABLE.

NOTE: If Server app is removed, you will no longer be able to create users whose password type is RECOVERABLE.

Xcode

Depending on the service you are migrating, you should probably have Xcode and the Xcode command-line tools installed. To install the Xcode command-line tools, open /Applications/Utilities/Terminal, type `xcode-select --install`, then press Return.

Property lists

Each service should have a launchd property list (.plist) file so that the service starts up after the computer restarts. Some services, such as the Mail service, have more than one .plist file.

Open source software

As you migrate services, you'll install open source software and third-party tools designed to make this process easier. It is important that you read all terms and conditions before you accept any license agreements.

IMPORTANT: Please note that information provided in this document about open source projects, third-party vendors, products not manufactured by Apple, or independent websites not controlled or tested by Apple, is provided without recommendation or endorsement. Apple assumes no responsibility with regard to the selection, performance, or use of such open source projects, third-party vendors, non-Apple products or third-party websites that may be mentioned in this document. Apple makes no representations or warranties regarding the foregoing or with respect to product accuracy or reliability or compatibility with macOS Server. Risks are inherent in the use of the Internet as well as in such projects, vendors, products or websites. Contact the project owner or vendor for additional information. Other company and product names may be trademarks of their respective owners.

DNS

Overview

DNS is implemented via BIND, which is an open source product. The DNS server implementation shipped with macOS Server is BIND 9.9.7-P3 from ISC. This version of `named` (the nameserver daemon in BIND) performs the following tasks:

- Looks for its configuration files in /Library/Server/named/
- Logs to a file in /Library/Logs/named.log by default

Before you begin

Turn off the DNS service in the Server app.

Post-migration result

After migration you'll have:

- BIND as your DNS service
- The identical configuration as the macOS Server DNS service
- A launchd job that starts the service after computer restarts

Switch from macOS Server to open source

You can manually install and configure the open source BIND9 so that it can run on a server that has macOS Server installed with existing DNS service data. The steps follow here.

Download and build BIND9

1. Install Xcode. For more information, see the Introduction.
2. Go to https://www.isc.org/downloads/, and select the desired version of BIND.

NOTE: The steps below use bind-9.9.9-P6. Your version may be different.

3. Download the selected version and one or more signature (.asc) files. Run the following commands:
   A. `tar xzf bind-9.9.9-P6.tar.gz`
   B. `cd ./bind-9.9.9-P6`
   C. `./configure --infodir="/usr/share/info" --sysconfdir="/etc" --localstatedir="/var" --enable-atomic="no" --with-openssl=no --with-gssapi=yes --enable-symtable=none --with-libxml2=no`
   D. `make`
4. To test the build, run the following commands:
   A. `sudo ./bin/tests/system/ifconfig.sh up`
   B. `make test`
   C. `sudo ./bin/tests/system/ifconfig.sh down`

Install BIND9

1. Run the following command:
   `sudo make install`
2. Verify that `man named` finds the man page properly.

Create a launchd .plist file for the BIND9 service

1. Create a text file in /Library/LaunchDaemons/, and name it org.isc.named.plist.
2. Add the following content and save the file:

```
<?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>Disabled</key>
             <true/>
             <key>EnableTransactions</key>
             <true/>
             <key>Label</key>
             <string>org.isc.named</string>
             <key>KeepAlive</key>
             <true/>
             <key>ProcessType</key>
             <string>Standard</string>
             <key>ProgramArguments</key>
             <array>
             <string>/usr/local/sbin/named</string>
             <string>-f</string>
             <string>-c</string>
             <string>/Library/Server/named/named.conf</string>
             </array>
          </dict>
     </plist>
```

3. Set file ownership to root:wheel.
   `sudo chown root:wheel /Library/LaunchDaemons/org.isc.named.plist`
4. Load and verify the job.
   `sudo launchctl load -w /Library/LaunchDaemons/org.isc.named.plist`
   `launchctl print system/org.isc.named`

Ongoing management

Settings can be changed after BIND is configured by editing the /Library/Server/named/named.conf file. After changes are made, you can have the service reread the configuration file by executing the command `sudo killall -HUP named.`

DHCP

Overview

The macOS Server DHCP service uses the bootpd daemon to provide DHCP services. The bootpd daemon is also one of the daemons required to provide NetInstall services. The configuration for DHCP and NetInstall reside in the same configuration file, /etc/bootpd.plist. Refer to the bootpd(8) man page for more information on the full capabilities of the bootpd daemon as well as information on how to update your DHCP configuration in /etc/bootpd.plist to add or remove network interfaces and associated network address ranges.

Before you begin

Turn off the DHCP service in the Server app.

Post-migration result

After migration you'll have:

- bootpd as your DHCP service
- The identical configuration as the macOS Server DHCP service
- A launchd job that starts the service after computer restarts

Switch from macOS Server to macOS for DHCP service

1. List all network interfaces on your system and identify the potential network interfaces on which you want to support DHCP. List the network interfaces by executing the command:

   ```
   ifconfig –a
   ```

2. Take note of each interface name, such as en1, associated with the IPv4 address of each network interface on which you want to support DHCP. Typically, you will only support DHCP on a single network interface.
3. Edit /etc/bootpd.plist to add entries to enable DHCP on your desired network interfaces.
   A. In the top level <dict>, add "`<key>dhcp_enabled</key>`"
   B. The value for this key should be an array of strings where each string value is the interface name, such as en1, of an interface for which you want to enable DHCP. For example, if you want to enable DHCP on a single network interface, en1, then add the following to the top level <dict>:

      ```
      <key>dhcp_enabled</key>
      <array>
         <string>en1</string>
      </array>
      ```

4. Save your changes to /etc/bootpd.plist

5. Load and verify the DHCP job:
   ```
   sudo launchctl load -w /System/Library/LaunchDaemons/
   bootps.plist
   launchctl print system/com.apple.bootpd
   ```

Ongoing management

Additional changes to your DHCP configuration can be made by editing the "Subnets" dictionary in /etc/bootpd.plist. See the bootpd(8) man page for more information on what can be configured for DHCP. After changes are made, you can have the service reread its configuration file by executing the command `sudo killall -HUP bootpd`.

VPN

Overview

The macOS Server VPN service uses the vpnd daemon in macOS to provide L2TP IPSEC VPN services. View the vpnd man page for more information. The configuration file is `/Library/Preferences/SystemConfiguration/com.apple.RemoteAccessServers.plist`, and its format is defined in the vpnd man page.

Before you begin

Turn off the VPN service in the Server app.

Post-migration result

After migration you'll have:

- vpnd as your VPN service
- The identical configuration as the macOS Server VPN service
- A launchd job that starts the service after computer restarts

Switch from macOS Server to macOS for VPN service

1. Create a text file in /Library/LaunchDaemons/ and name it vpn.ppp.l2tp.plist.
2. Add the following content and save the file:

```
<?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>Disabled</key>
             <true/>
             <key>EnableTransactions</key>
             <true/>
             <key>Label</key>
             <string>vpn.ppp.l2tp</string>
             <key>KeepAlive</key>
             <true/>
             <key>Program</key>
             <string>/usr/sbin/vpnd</string>
             <key>ProgramArguments</key>
             <array>
             <string>vpnd</string>
             <string>-x</string>
             <string>-i</string>
             <string>com.apple.ppp.l2tp</string>
             </array>
             <key>EnableTransactions</key>
             <false/>
```

```
        <key>EnablePressuredExit</key>
        <false/>
    </dict>
</plist>
```

3. Set file ownership to root:wheel.
   `sudo chown root:wheel /Library/LaunchDaemons/vpn.ppp.l2tp.plist`
4. Load and verify the job.
   `sudo launchctl load -w /Library/LaunchDaemons/vpn.ppp.l2tp.plist`
   `launchctl print system/vpn.ppp.l2tp`

Ongoing management

Settings can be changed after vpnd is configured by editing the /Library/Preferences/SystemConfiguration/com.apple.RemoteAccessServers.plist file. Refer to the vpnd (5) man page for details on the configuration format. After changes are made, you can have the service reread the configuration file by executing the command `sudo killall -HUP vpnd`.

RADIUS

Overview

RADIUS (Remote Authentication Dial-In User Service) is a networking protocol that provides AAA (authentication, authorization, and accounting) for network services. The RADIUS service included in macOS Server is based on the open source FreeRADIUS project version 2.2.9. It provides RADIUS services to Apple AirPort products, allowing integration with Open Directory users and groups. Converting to the open source version allows continued authentication for all current users with password type RECOVERABLE. New users won't be able to use FreeRADIUS.

Before you begin

Turn off the RADIUS service in the Server app.

Post-migration result

After migration you'll have:

- FreeRADIUS as your RADIUS service
- The identical configuration as the macOS Server RADIUS service
- A launchd job that starts the service after computer restarts

Switch from macOS Server to open source

Build FreeRADIUS

FreeRADIUS requires talloc, available from samba.org

NOTE: The steps below use talloc 2.1.0. Your version may be different.

1. Run the following commands:

   A. `curl -LO http://www.samba.org/ftp/talloc/talloc-2.1.0.tar.gz`
   B. `tar zxvf talloc-2.1.0.tar.gz`
   C. `cd talloc-2.1.0`
   D. `./configure --without-gettext`
   E. `make`
   F. `sudo make install`

FreeRADIUS requires OpenSSL, available from openssl.org

NOTE: The steps below use openssl 1.1.0e. Your version may be different.

1. Run the following commands:
   A. `curl -LO https://www.openssl.org/source/openssl-1.1.0e.tar.gz`
   B. `tar zxvf openssl-1.1.0e.tar.gz`
   C. `cd openssl-1.1.0e`
   D. `./config`
   E. `make`
   F. `make test`
   G. `sudo make install`

FreeRADIUS

1. Download the current stable version of FreeRADIUS from freeradius.org. and unpack the archive.

NOTE: The steps below use FreeRADIUS 3.0.21. Your version may be different.

2. Run the following commands:
   A. `tar zxvf freeradius-server-3.0.21.tar.gz`
   B. `cd freeradius-server-3.0.21`
   C. `./configure --enable-developer=yes --localstatedir=/var --with-openssl-lib-dir=/usr/local/lib`
   D. `make`
   E. `sudo make install`

Configure FreeRADIUS

Once FreeRADIUS is built, you must configure it.

1. Verify that the package is functional using the following command:
   `sudo /usr/local/sbin/radiusd -X`
2. Wait until the line "Ready to process requests" is printed, then press Control-C to stop the service.
3. Verify that the opendirectory module is present using the following command:
   `sudo ls /usr/local/etc/raddb/mods-available/`
   You should see *opendirectory* in the output.
4. Run the following command:
   `ls /usr/local/lib/rlm_opendirectory.*`
   You should see the following in the output:
   - `rlm_opendirectory.a`
   - `rlm_opendirectory.dylib`
   - `rlm_opendirectory.la`

Configure sqlite

1. Run the following commands:
   A. `cd /usr/local/etc/raddb/`
   B. `sudo -s`
   C. `cd mods-enabled`
   D. `ln -s ../mods-available/sql sql`
   E. `cd ..`
6. In the mods-enabled/sql file, make the following changes:

A. Locate "driver =" and set it as follows:
`driver = “rlm_sql_sqlite”`
B. Locate “filename" in the file, uncomment (if necessary), then set it to:
`filename = “/var/db/radius/freeradius.db”`
C. Uncomment the following line:
`read_clients = yes`

7. Run the following commands:
   A. `sudo mkdir /var/db/radius`
   B. `sudo /usr/local/sbin/radiusd –XC`
8. Verify that the initial database is created in /var/db/radius/freeradius.db.
9. Export the existing AirPort info by running the following command:
`sudo sqlite3 –csv /Library/Server/radius/raddb/sqlite_radius_client_database ‘select * from nas;’ > /tmp/clients.csv`
10. Import into the new database by running the following commands:
    A. `sudo sqlite3 /var/db/radius/freeradius.db`
    B. `sqlite> .separator “,”`
    C. `sqlite> .import /tmp/clients.csv nas`
    D. `sqlite> .q`
11. Test the configuration by running the following command:
`sudo /usr/local/sbin/radiusd –XC`
12. Verify that you see a line starting with:
"rlm_sql (sql) Adding client <IP address of AirPort Base Station> ….”

Configure certificates

1. Run the following command:
`cd /usr/local/etc/raddb/`
2. Move the existing certs directory aside by running the following commands:
   A. `sudo mv certs certs.orig`
   B. `sudo mkdir certs`
3. Export the private key for the default identity certificate.
   A. Open /Applications/Utilities/Keychain Access.
   B. Select the System Keychain in the All Items category
   C. Look for the MACHINE_IDENTITY identity preference.
   D. If there is no MACHINE_IDENTITY preference, find the identity certificate used to secure the webserver.
   E. Select the identity certificate indicated above, in the My Certificates Category.
   F. The identity certificate should be the one named with the FQDN name of your server.
   G. Export the identity to a .p12 file (choose a passphrase), for example, /usr/local/etc/raddb/certs/Certificates.p12.
4. `cd /usr/local/etc/raddb/certs`
5. Create the private key pem file by running the following command:
`sudo openssl pkcs12 –in Certificates.p12 –out server.private.pem –nocerts`

NOTE: Provide the passphrase you used for the export above, and choose a passphrase for the private keyfile.

6. Remove the file by running the following command:
`sudo rm Certificates.p12`

7. Look in /Library/Server/radius/raddb/eap.conf file for the following settings:

A. In the *tls* section:
- certificates_file
- CA_file

8. Copy the files pointed to by the settings above into: /usr/local/etc/raddb/certs/
9. `cd /usr/local/etc/raddb` and edit mods-enabled/eap
10. In the eap section:
`set default_eap_type = ttls`
11. In the tls-config tls-common section:

A. `Set private_key_password = <the passphrase you used above [the openssl cmd]>`
B. `Set private_key_file = /usr/local/etc/raddb/certs/server.private.pem`
C. `Set certificates_file = /usr/local/etc/raddb/certs/ <certificates_file from eap.conf>`
D. `Set CA_file = /usr/local/etc/raddb/certs/ <CA_file from eap.conf>`
E. `Uncomment random_file = /dev/urandom`
F. `Uncomment fragment_size = 1024`
G. `Uncomment include_length = yes`
H. `Set crl_check = no`

12. In the cache section:
`Set enable = no`
13. In the ttls section:
`Set default_eap_type = mschapv2`
14. Run the following openssl command:
`sudo openssl dhparam –out certs/dh 2048`
15. Test the configuration again by running the following command:
`sudo /usr/local/sbin/radiusd –XC`

Set up users

1. Run the following commands:

A. `cd /usr/local/etc/raddb`
B. `cd mods–enabled`
C. `sudo ln –s ../mods–available/opendirectory opendirectory`
D. `cd ..`

2. Edit sites-enabled/default:

A. In the authorize section, comment out filter_username.
B. After files, insert the following lines:
- `#`
- `# Mac OS X OpenDirectory`
- `opendirectory`

C. Comment out -ldap
D. In the authenticate section after digest, insert the following lines:
- `#`
- `# Mac OS X OpenDirectory`
- `Auth–Type opendirectory {`
- `opendirectory`

```
}
```

E. In the accounting section, comment out unix.

3. Test the configuration once again:
   ```
   sudo /usr/local/sbin/radiusd –XC
   ```

Create a launchd .plist file for the RADIUS service

1. Create a text file in /Library/LaunchDaemons/ and name it org.freeradius.radiusd.plist.
2. Add the following content and save the file:

```
<?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>Disabled</key>
             <true/>
             <key>EnableTransactions</key>
             <true/>
             <key>Label</key>
             <string>org.freeradius.radiusd</string>
             <key>KeepAlive</key>
             <true/>
             <key>ProcessType</key>
             <string>Standard</string>
             <key>ProgramArguments</key>
             <array>
             <string>/usr/local/sbin/radiusd</string>
             <string>-sf</string>
             </array>
          </dict>
     </plist>
```

3. Set file ownership to root:wheel.
   ```
   sudo chown root:wheel /Library/LaunchDaemons/
   org.freeradius.radiusd.plist
   ```
4. Load and verify the job.
   ```
   sudo launchctl load –w /Library/LaunchDaemons/
   org.freeradius.radiusd.plist
   launchctl print system/org.freeradius.radiusd.plist
   ```

AirPort Base Station management

macOS Server allowed you to configure an AirPort Base Station. You can now use the AirPort Utility to manage your base stations. For more information, see AirPort Utility Help at:
https://support.apple.com/guide/aputility/welcome/mac.

Ongoing management

Adding a new base station: The freeRadius documentation refers to base stations (or access points) as clients. Follow the instructions in the Getting Started guide at: http://wiki.freeradius.org/guide/Getting%20Started#adding-a-client.

Adding a user to freeRadius: Add new users of the radius service through the Server.app. This will require that the user's password be stored in a less secure manner.

NetInstall

Overview

The NetInstall service hosts images so they can be installed on other computers. The NetInstall service relies on several components:

- BOOTP service: Responds to boot requests made by NetInstall clients with information about available NetInstall images
- TFTP service: Sends files to NetInstall clients which are required for booting into a NetInstall image
- HTTP or NFS service: Delivers the actual disk image data of the NetInstall image

For further information on configuring NetInstall, see the bootpd(8) man page.

Before you begin

Turn off the NetInstall service in the Server app.

NetInstall migration requires the NetInstall service to be disabled. Depending on the type of NetInstall images you are serving, you may require the use of the HTTP server. If you require the use of the HTTP server for NetInstall then you must also disable the Profile Manager service.

Post-migration result

After migration you'll have:

- The identical configuration as the macOS Server NetInstall service
- A launchd job that starts the required services after computer restarts

Switch from macOS Server to macOS for NetInstall

1. Enable the com.apple.bootps launchd job, which provides services for BOOTP by running the following command:
   `sudo launchctl load –w /System/Library/LaunchDaemons/bootps.plist`
2. Enable TFTP service, which provides access to files in the /private/tftpboot directory by running the following command:
   `sudo launchctl load –w /System/Library/LaunchDaemons/tftp.plist`
3. NetInstall images are stored in /Library/NetBoot/NetBootSP0/ on any volume configured to store images in macOS Server. If you configured multiple volumes, you will have a /Library/NetBoot/NetBootSP0 directory on each volume. For ease of ongoing management, you should choose a single volume for storing Images and consolidate all your existing .nbi bundles to that directory.

4. Determine which protocol is needed to serve your NetInstall images. The protocol may be NFS, HTTP, or both. You can consult the "NBImageInfo.plist" inside a .nbi bundle to see whether a given image was configured to be served via HTTP or NFS. In the NBImageInfo.plist, you will see either:

```
<key>Type</key>
<string>NFS</string>
```

or

```
<key>Type</key>
<string>HTTP</string>
```

If you have NFS images, follow steps 5 through 6:

5. Run the following command:
   `sudo launchctl load -w /System/Library/LaunchDaemons/com.apple.nfsd.plist`
6. For NFS, you must add an entry like "/Library/NetBoot/NetBootSP0 -ro -maproot=root" to /etc/exports. For volumes other than the boot volume, the full path must be included. So, for a volume named "Data" mounted at /Volumes/Data, the entry would be "/Volumes/Data/Library/NetBoot/NetBootSP0 -ro -maproot=root"

If you have HTTP images, follow steps 7 through 8:

7. Run the following commands:
   A. `sudo launchctl load -w /System/Library/LaunchDaemons/org.apache.httpd.plist`
   B. `sudo mkdir /Library/WebServer/Documents/NetBoot`
8. Create a symlink for your /Library/NetBoot/NetBootSP0 directory, in /Library/WebServer/Documents/NetBoot/ by running the following command (replacing <VolumeName> with the appropriate volume name):
   `sudo ln -s /Volumes/<VolumeName>/Library/NetBoot/NetBootSP0 /Library/WebServer/Documents/NetBoot/NetBootSP0`

Ongoing management

If you want to add a NetInstall image

1. Copy the NetInstall image .nbi bundle into your /Library/NetBoot/NetBootSP0 directory.
2. Inspect the NBImageInfo.plist to determine whether the image is configured to be served via HTTP or NFS.
3. If needed, change the key type to your preferred protocol. If the value for <key>Type</key> is missing from the NBImageInfo.plist, edit the file and add one of the following depending on your choice of protocol:

```
<key>Type</key>
<string>NFS</string>
```

or

```
<key>Type</key>
<string>HTTP</string>
```

If you want to remove a NetInstall image

1. Delete the .nbi bundle for the image from the /Library/NetBoot/NetBootSP0/ folder

If you want to disable the NetInstall service

1. The bootpd daemon is used by the NetInstall and DHCP services. If you are not using bootpd to provide DHCP services then you can disable the daemon by running the following command:
   `sudo launchctl unload –w /System/Library/LaunchDaemons/bootps.plist`
2. If you are serving DHCP content and only want to disable NetInstall, edit the /etc/bootpd.plist file and replace "<key>netboot_enabled</key>" with "<key>netboot_disabled</key>". Send a HUP signal to bootpd to have it reread its configuration by running the following command:
   `sudo killall –HUP bootpd`
3. Unload the tftpd daemon by running the following command:
   `sudo launchctl unload –w /System/Library/LaunchDaemons/tftp.plist`
4. If you were serving NetInstall images using NFS you can disable that service by running the following command:
   `sudo launchctl unload –w /System/Library/LaunchDaemons/com.apple.nfsd.plist`
5. If you were serving NetInstall images using HTTP you can disable that service by running the following command:
   `sudo launchctl unload –w /System/Library/LaunchDaemons/org.apache.httpd.plist`

Websites

Overview

Websites uses the Apache httpd service. The Apache httpd server is also used by several other HTTP-based services in macOS Server, including Wiki, WebDAV, and Profile Manager. The chapters about migrating those services describe the relevant Apache changes for them; this chapter's description about Apache changes is specific to the Websites service.

Before you begin

Turn off the Websites service in the Server app. The Profile Manager service must also be turned off and remain off in order to run the migrated HTTP service on the original macOS Server system.

Post-migration result

After migration you'll have a fully functional web service configuration running on macOS, serving the content that was previously served under macOS Server, from the same set of virtual hosts (sites) with the following changes:

- The virtual hosts won't be behind a reverse proxy; they'll be listening on standard HTTP ports unless configured otherwise.
- The SSL certificates will reside in the System.keychain, not in the file system.
- The SSL processing will be managed by Apple's mod_secure_transport plug-in for Apache, not by mod_ssl.
- The recoverable hashes needed to support HTTP Digest authentication are not typically available in macOS without macOS Server. So any Digest-based authentication realms (where access to web content requires authenticated membership in an Open Directory group) are converted to Basic authentication, which is only secure with SSL.

Migrate websites

Edit the Websites configuration files. See the diff examples in the next section for details.

1. Copy files from /Library/Server/WebData/Sites to /Library/WebServer/Sites.
2. Locate and remove any default.html.* symlinks to localized pages.
3. Copy the main Apache configuration file /Library/Server/Web/Config/apache2/httpd_server_app.conf to /etc/apache2/.
4. Copy the active Apache configuration files for all sites from /Library/Server/Web/Config/apache2/sites/*.conf to /etc/apache2/sites/.
5. Change DocumentRoot and Directory paths from /Library/Server/WebData/Sites to /Library/WebServer/Sites.
6. Change the VirtualHost and Listen directives to reflect actual addresses and ports rather than those used behind the service proxy. For example:
   - Change <VirtualHost 127.0.0.1:34543> to <VirtualHost *:443>

- And in virtual_host_global.conf, change Listen 127.0.0.1:34543 to Listen *:443>.

7. Delete the mod_ssl directives and enable the the mod_secure_transport directives for affected sites.
8. Remove IfDefine WEBSERVICE_ON blocks.
9. Convert each authentication realm (typically a Directory block) from OD-based digest auth to OD-based basic auth:
   A. Change "Require group ..." directives to "Require od_group ..".
   B. Add AuthBasicProvider od_group directives.
10. Change "Require all denied" to "Require no-user".
11. Disable any Include directives.

Diff file examples for the Websites service

You can identify the changed lines as follows:

- Changes with a "-" (indicate lines or text that should be replaced or removed)
- Changes with a "+" (indicate lines or text that should be edited or added)
- Other changes appear in **bold**

Example of diff file for the following files:

- /Library/Server/Web/Config/apache2/sites/0000_127.0.0.1_34543_.conf
- /etc/apache2/sites/0000_127.0.0.1_34543_.conf

```
diff –ubr /Library/Server/Web/Config/apache2/sites /etc/
apache2/sites
diff –ubr /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34543_.conf /etc/apache2/sites/
0000_127.0.0.1_34543_.conf
--- /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34543_.conf 2018-02-18 09:41:31.000000000 -0700
+++ /etc/apache2/sites/0000_127.0.0.1_34543_.conf 2018-02-18
09:44:33.000000000 -0700
@@ -1,42 +1,26 @@
-<VirtualHost 127.0.0.1:34543>
+<VirtualHost *:443>
ServerName default
ServerAdmin admin@example.com
- DocumentRoot "/Library/Server/Web/Data/Sites/Default"
+ DocumentRoot "/Library/WebServer/Sites/Default"
DirectoryIndex index.html index.php default.html
#CustomLog |/usr/sbin/rotatelogs combinedvhost
#ErrorLog |/usr/sbin/rotatelogs
CustomLog |/usr/sbin/rotatelogs combinedvhost
ErrorLog |/usr/sbin/rotatelogs
- <IfModule mod_ssl.c>
- SSLEngine Off
- SSLCipherSuite "HIGH:MEDIUM:!MD5:!RC4:!3DES"
- SSLProtocol –all +TLSv1 +TLSv1.1 +TLSv1.2
- SSLProxyEngine Off
- SSLCertificateFile "/etc/certificates/
al5.apple.com.E565FC5DFDF57DD6284D5700A4540EC78625C27F.cert.pem
"
```

```
- SSLCertificateKeyFile "/etc/certificates/
al5.apple.com.E565FC5DFDF57DD6284D5700A4540EC78625C27F.key.pem"
- SSLCertificateChainFile "/etc/certificates/
al5.apple.com.E565FC5DFDF57DD6284D5700A4540EC78625C27F.chain.pe
m"
- SSLProxyProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
- SSLProxyCheckPeerCN off
- SSLProxyCheckPeerName off
- </IfModule>
<IfModule mod_secure_transport.c>
- MSTEngine Off
+ MSTEngine On
MSTCipherSuite HIGH, MEDIUM
MSTProtocolRange TLSv1.2 TLSv1.2
MSTProxyEngine On
MSTIdentity
SHA-256:583e635c1cf9042535f822bdb22e4d136a09205280c375ba2724c12
c658f3482:"al5.apple.com"
MSTProxyProtocolRange TLSv1.2 TLSv1.2
</IfModule>
- <Directory "/Library/Server/Web/Data/Sites/Default">
+ <Directory "/Library/WebServer/Sites/Default">
Options All -Indexes -ExecCGI -Includes +MultiViews
AllowOverride None
<IfModule mod_dav.c>
DAV Off
</IfModule>
- <IfDefine !WEBSERVICE_ON>
- Require all denied
- ErrorDocument 403 /customerror/websitesoff403.html
- </IfDefine>
</Directory>
- Include /Library/Server/Web/Config/apache2/
httpd_webdavsharing.conf
+ #Include /Library/Server/Web/Config/apache2/
httpd_webdavsharing.conf
</VirtualHost>
Only in /Library/Server/Web/Config/apache2/sites:
0000_127.0.0.1_34543_.conf.default
Only in /Library/Server/Web/Config/apache2/sites:
0000_127.0.0.1_34543_.conf.orig
Only in /Library/Server/Web/Config/apache2/sites:
0000_127.0.0.1_34543_.conf.prev
```

Example of diff file for the following files:

- /Library/Server/Web/Config/apache2/sites/
  0000_127.0.0.1_34543_multipass.example.com.conf
- /etc/apache2/sites/0000_127.0.0.1_34543_multipass.example.com.conf

```
diff -ubr /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34543_multipass.example.com.conf /etc/apache2/
sites/0000_127.0.0.1_34543_multipass.example.com.conf
--- /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34543_multipass.example.com.conf 2018-02-18
09:41:31.000000000 -0700
```

```
+++ /etc/apache2/sites/
0000_127.0.0.1_34543_multipass.example.com.conf 2018-02-18
09:44:33.000000000 -0700
@@ -1,51 +1,32 @@
-<VirtualHost 127.0.0.1:34543>
+<VirtualHost *:443>
ServerName https://multipass.example.com:443
ServerAdmin admin@example.com
- DocumentRoot "/Library/Server/Web/Data/Sites/
multipass.example.com"
+ DocumentRoot "/Library/WebServer/Sites/multipass.example.com"
DirectoryIndex index.html index.php default.html
CustomLog /var/log/apache2/access_log combinedvhost
ErrorLog /var/log/apache2/error_log
- <IfModule mod_ssl.c>
- SSLEngine Off
- SSLCipherSuite "HIGH:MEDIUM:!MD5:!RC4:!3DES"
- SSLProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
- SSLProxyEngine Off
- SSLCertificateFile "/etc/certificates/
multipass.example.com.C4613458D36646E957667EC08A2FEA945838D483.
cert.pem"
- SSLCertificateKeyFile "/etc/certificates/
multipass.example.com.C4613458D36646E957667EC08A2FEA945838D483.
key.pem"
- SSLCertificateChainFile "/etc/certificates/
multipass.example.com.C4613458D36646E957667EC08A2FEA945838D483.
chain.pem"
- SSLProxyProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
- SSLProxyCheckPeerCN off
- SSLProxyCheckPeerName off
- </IfModule>
<IfModule mod_secure_transport.c>
- MSTEngine Off
+ MSTEngine On
MSTCipherSuite HIGH, MEDIUM
MSTProtocolRange TLSv1.2 TLSv1.2
MSTProxyEngine On
MSTIdentity
SHA-256:dbb1dd7eee6f9749d6476d9ef0a0838442210de0bc4a6b82cd4b5e0
718640b25:"multipass.example.com"
MSTProxyProtocolRange TLSv1.2 TLSv1.2
</IfModule>
- <Directory "/Library/Server/Web/Data/Sites/
multipass.example.com">
+ <Directory "/Library/WebServer/Sites/multipass.example.com">
Options All -Indexes -ExecCGI -Includes +MultiViews
AllowOverride None
<IfModule mod_dav.c>
DAV Off
</IfModule>
- <IfDefine !WEBSERVICE_ON>
- Require all denied
- ErrorDocument 403 /customerror/websitesoff403.html
- </IfDefine>
```

```
- AuthType Digest
+ AuthType Basic
+ AuthBasicProvider od_apple
AuthName "Realm ID 45004451"
<Limit PUT DELETE PROPPATCH PROPFIND MKCOL COPY MOVE LOCK
UNLOCK>
- Require no-user
+ Require all denied
</Limit>
<Limit GET HEAD OPTIONS CONNECT POST>
- Require group admin
+ Require od_group admin
</Limit>
- <IfDefine !WEBSERVICE_ON>
- Require all denied
- ErrorDocument 403 /customerror/websitesoff403.html
- </IfDefine>
</Directory>
</VirtualHost>
Only in /Library/Server/Web/Config/apache2/sites:
0000_127.0.0.1_34543_multipass.example.com.conf.prev
```

Example of diff file for the following files:

- /Library/Server/Web/Config/apache2/sites/0000_127.0.0.1_34580_.conf
- /etc/apache2/sites/0000_127.0.0.1_34580_.conf

```
diff -ubr /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34580_.conf /etc/apache2/sites/
0000_127.0.0.1_34580_.conf
--- /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34580_.conf 2018-02-18 09:41:31.000000000 -0700
+++ /etc/apache2/sites/0000_127.0.0.1_34580_.conf 2018-02-18
09:44:33.000000000 -0700
@@ -1,19 +1,10 @@
-<VirtualHost 127.0.0.1:34580>
+<VirtualHost *:80>
ServerName default
ServerAdmin admin@example.com
- DocumentRoot "/Library/Server/Web/Data/Sites/Default"
+ DocumentRoot "/Library/WebServer/Sites/Default"
DirectoryIndex index.html index.php default.html
CustomLog "/var/log/apache2/access_log" combinedvhost
ErrorLog "/var/log/apache2/error_log"
- <IfModule mod_ssl.c>
- SSLEngine Off
- SSLCipherSuite "HIGH:MEDIUM:!MD5:!RC4:!3DES"
- SSLProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
- SSLProxyEngine Off
- SSLProxyProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
- SSLProxyCheckPeerCN off
- SSLProxyCheckPeerName off
- </IfModule>
<IfModule mod_secure_transport.c>
MSTEngine Off
MSTCipherSuite HIGH, MEDIUM
```

```
@@ -21,20 +12,16 @@
MSTProxyEngine On
MSTProxyProtocolRange TLSv1.2 TLSv1.2
</IfModule>
- <Directory "/Library/Server/Web/Data/Sites/Default">
+ <Directory "/Library/WebServer/Sites/Default">
Options All -Indexes -ExecCGI -Includes +MultiViews
AllowOverride None
<IfModule mod_dav.c>
DAV Off
</IfModule>
- <IfDefine !WEBSERVICE_ON>
- Require all denied
- ErrorDocument 403 /customerror/websitesoff403.html
- </IfDefine>
</Directory>
RewriteEngine On
RewriteCond %{HTTP_HOST} !(^localhost|^127.0.0.1|^::1)
RewriteCond %{REQUEST_URI} !^/netboot/ [NC]
RewriteRule .* https://%{SERVER_NAME}%{REQUEST_URI} [R]
- Include /Library/Server/Web/Config/apache2/
httpd_webdavsharing.conf
+ #Include /Library/Server/Web/Config/apache2/
httpd_webdavsharing.conf
</VirtualHost>
Only in /Library/Server/Web/Config/apache2/sites:
0000_127.0.0.1_34580_.conf.default
Only in /Library/Server/Web/Config/apache2/sites:
0000_127.0.0.1_34580_.conf.orig
Only in /Library/Server/Web/Config/apache2/sites:
0000_127.0.0.1_34580_.conf.prev
```

Example of diff file for the following files:

- /Library/Server/Web/Config/apache2/sites/
  0000_127.0.0.1_34580_tworealms.nossl.example.com.conf
- /etc/apache2/sites/
  0000_127.0.0.1_34580_tworealms.nossl.example.com.conf

```
diff -ubr /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34580_tworealms.nossl.example.com.conf /etc/
apache2/sites/
0000_127.0.0.1_34580_tworealms.nossl.example.com.conf
--- /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34580_tworealms.nossl.example.com.conf
2018-02-18 09:41:31.000000000 -0700
+++ /etc/apache2/sites/
0000_127.0.0.1_34580_tworealms.nossl.example.com.conf
2018-02-18 09:44:33.000000000 -0700
@@ -1,19 +1,10 @@
-<VirtualHost 127.0.0.1:34580>
+<VirtualHost *:80>
ServerName http://tworealms.nossl.example.com:80
ServerAdmin admin@example.com
- DocumentRoot "/Library/Server/Web/Data/Sites/
tworealms.nossl.example.com"
```

```
+ DocumentRoot "/Library/WebServer/Sites/
tworealms.nossl.example.com"
DirectoryIndex index.html index.php default.html
CustomLog /var/log/apache2/access_log combinedvhost
ErrorLog /var/log/apache2/error_log
- <IfModule mod_ssl.c>
- SSLEngine Off
- SSLCipherSuite "HIGH:MEDIUM:!MD5:!RC4:!3DES"
- SSLProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
- SSLProxyEngine Off
- SSLProxyProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
- SSLProxyCheckPeerCN off
- SSLProxyCheckPeerName off
- </IfModule>
<IfModule mod_secure_transport.c>
MSTEngine Off
MSTCipherSuite HIGH, MEDIUM
@@ -21,43 +12,33 @@
MSTProxyEngine On
MSTProxyProtocolRange TLSv1.2 TLSv1.2
</IfModule>
- <Directory "/Library/Server/Web/Data/Sites/
tworealms.nossl.example.com">
+ <Directory "/Library/WebServer/Sites/
tworealms.nossl.example.com">
Options All -Indexes -ExecCGI -Includes +MultiViews
AllowOverride None
<IfModule mod_dav.c>
DAV Off
</IfModule>
- <IfDefine !WEBSERVICE_ON>
- Require all denied
- ErrorDocument 403 /customerror/websitesoff403.html
- </IfDefine>
</Directory>
- <Directory "/Library/Server/Web/Data/Sites/
tworealms.nossl.example.com/red-group-folder">
- AuthType Digest
+ <Directory "/Library/WebServer/Sites/
tworealms.nossl.example.com/red-group-folder">
+ AuthType Basic
+ AuthBasicProvider od_apple
AuthName "Realm ID 92177280"
<Limit PUT DELETE PROPPATCH PROPFIND MKCOL COPY MOVE LOCK
UNLOCK>
- Require no-user
+ Require all denied
</Limit>
<Limit GET HEAD OPTIONS CONNECT POST>
- Require group "Red Site Admin"
+ Require od_group "Red Site Admin"
</Limit>
- <IfDefine !WEBSERVICE_ON>
- Require all denied
- ErrorDocument 403 /customerror/websitesoff403.html
```

```
- </IfDefine>
</Directory>
- <Directory "/Library/Server/Web/Data/Sites/
tworealms.nossl.example.com/blue-group-folder">
- AuthType Digest
+ <Directory "/Library/WebServer/Sites/
tworealms.nossl.example.com/blue-group-folder">
+ AuthType Basic
+ AuthBasicProvider od_apple
AuthName "Realm ID 84185326"
<Limit PUT DELETE PROPPATCH PROPFIND MKCOL COPY MOVE LOCK
UNLOCK>
- Require no-user
+ Require all denied
</Limit>
<Limit GET HEAD OPTIONS CONNECT POST>
- Require group Blue-Donuts
+ Require od_group Blue-Donuts
</Limit>
- <IfDefine !WEBSERVICE_ON>
- Require all denied
- ErrorDocument 403 /customerror/websitesoff403.html
- </IfDefine>
</Directory>
</VirtualHost>
Only in /Library/Server/Web/Config/apache2/sites:
0000_127.0.0.1_34580_tworealms.nossl.example.com.conf.prev
```

Example of diff file for the following files:

- /Library/Server/Web/Config/apache2/sites/virtual_host_global.conf
- /etc/apache2/sites/virtual_host_global.conf

```
diff -ubr /Library/Server/Web/Config/apache2/sites/
virtual_host_global.conf /etc/apache2/sites/
virtual_host_global.conf
--- /Library/Server/Web/Config/apache2/sites/
virtual_host_global.conf 2018-02-18 09:41:31.000000000 -0700
+++ /etc/apache2/sites/virtual_host_global.conf 2018-02-18
09:44:33.000000000 -0700
@@ -4,5 +4,5 @@
# based on the VirtualHost directive inside
# each site configuration file found in this directory.
#
-Listen 127.0.0.1:34580
-Listen 127.0.0.1:34543
+Listen *:80
+Listen *:443
Only in /Library/Server/Web/Config/apache2/sites:
virtual_host_global.conf.default
Only in /Library/Server/Web/Config/apache2/sites:
virtual_host_global.conf.orig
Only in /Library/Server/Web/Config/apache2/sites:
virtual_host_global.conf.prev
```

Convert the httpd_server_app.conf file

Revise the httpd_server_app.conf file. See the diff examples in the next section for details.

1. Remove references to SERVER_INSTALL_PATH_PREFIX, webobjects, getsslpassphrase.
2. Remove "IfDefine WEBSERVICE_ON" blocks; replace with "Include /etc/apache2/sites/*.conf'.
3. Remove the PHP7 block, since these directives are available in the ./other/php7.conf subdirectory.
4. Remove the webdavmap line if present.
5. Change any customerror directives to refer to /usr/share/httpd/customerror.
6. Revise the IncludeOptional directive to refer to /etc/apache2/other/*.conf.
7. Change /Library/Server/Web/Data/CGI-Executables to /Library/WebServer/CGI-Executables".
8. Revise the LoadModule directives:
9. Disable mod_ssl, mod_auth_digest_apple.
10. Enable mod_secure_transport, mod_authnz_od_apple.
11. Revise the SSLPassPhraseDialog section, (which is inactive with mod_ssl disabled) to eliminate the reference to getslpassphrase.

Example diff file for httpd_server_app.conf

You can identify the changed lines as follows:

- Changes with a "-" (indicate lines or text that should be replaced or removed)
- Changes with a "+" (indicate lines or text that should be edited or added)
- Other changes appear in **bold**

Example of diff file for the following files:

- /Library/Server/Web/Config/apache2/httpd_server_app.conf
- /etc/apache2/httpd_server_app.conf

```
--- /Library/Server/Web/Config/apache2/httpd_server_app.conf
2018-02-18 05:17:36.000000000 -0700
+++ /etc/apache2/httpd_server_app.conf 2018-02-18
08:00:54.000000000 -0700
@@ -184,15 +184,13 @@
#Server-specific modules
# SERVER_INSTALL_PATH_PREFIX should be set as Environment
variable in launchd.plist
-#LoadModule apple_userdir_module $
{SERVER_INSTALL_PATH_PREFIX}/usr/libexec/apache2/
mod_userdir_apple.so
-#LoadModule bonjour_module ${SERVER_INSTALL_PATH_PREFIX}/usr/
libexec/apache2/mod_bonjour.so
-LoadModule auth_digest_apple_module $
{SERVER_INSTALL_PATH_PREFIX}/usr/libexec/apache2/
mod_auth_digest_apple.so
```

```
-LoadModule apple_auth_module ${SERVER_INSTALL_PATH_PREFIX}/
usr/libexec/apache2/mod_auth_apple.so
-#LoadModule spnego_auth_module ${SERVER_INSTALL_PATH_PREFIX}/
usr/libexec/apache2/mod_spnego_apple.so
-#LoadModule apple_digest_module ${SERVER_INSTALL_PATH_PREFIX}/
usr/libexec/apache2/mod_digest_apple.so
-#LoadModule wsgi_module ${SERVER_INSTALL_PATH_PREFIX}/usr/
libexec/apache2/mod_wsgi.so
-#LoadModule xsendfile_module ${SERVER_INSTALL_PATH_PREFIX}/
usr/libexec/apache2/mod_xsendfile.so
-#LoadModule secure_transport_module $
{SERVER_INSTALL_PATH_PREFIX}/usr/libexec/apache2/
mod_secure_transport.so
+#LoadModule apple_userdir_module /usr/libexec/apache2/
mod_userdir_apple.so
+#LoadModule bonjour_module /usr/libexec/apache2/mod_bonjour.so
+#LoadModule spnego_auth_module /usr/libexec/apache2/
mod_spnego_apple.so
+#LoadModule apple_digest_module /usr/libexec/apache2/
mod_digest_apple.so
+#LoadModule wsgi_module /usr/libexec/apache2/mod_wsgi.so
+#LoadModule xsendfile_module /usr/libexec/apache2/
mod_xsendfile.so
+#LoadModule secure_transport_module /usr/libexec/apache2/
mod_secure_transport.so
# If you wish httpd to run as a different user or group, you
must run httpd as root initially and it will switch.
@@ -243,13 +241,13 @@
RequestHeader set X_FORWARDED_PROTO 'http' env=!https
RequestHeader unset Proxy early
</IfModule>
-<Directory ${SERVER_INSTALL_PATH_PREFIX}/usr/share/web/
customerror>
+<Directory /usr/share/httpd/customerror>
AllowOverride None
Options MultiViews FollowSymlinks
Require all granted
Header Set Cache-Control no-cache
</Directory>
-Alias /customerror ${SERVER_INSTALL_PATH_PREFIX}/usr/share/
web/customerror
+Alias /customerror /usr/share/httpd/customerror
#
# The following lines prevent .htaccess and .htpasswd files
from being
@@ -347,14 +345,13 @@
# client. The same rules about trailing "/" apply to
ScriptAlias
# directives as to Alias.
#
- ScriptAliasMatch ^/cgi-bin/((?!(?i:webobjects)).*$) "/
Library/Server/Web/Data/CGI-Executables/$1"
</IfModule>
#
```

```
# "/Library/WebServer/CGI-Executables" should be changed to
whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
-<Directory "/Library/Server/Web/Data/CGI-Executables">
+<Directory "/Library/WebServer/CGI-Executables">
AllowOverride None
Options None
Require all granted
@@ -550,13 +547,13 @@
# Include /private/etc/apache2/extra/httpd-dav.conf
# Certain generated config files, such as migrated forward
proxy configs from SnowLeopard
-IncludeOptional /Library/Server/Web/Config/apache2/other/
*.conf
+IncludeOptional /etc/apache2/other/*.conf
# Secure (SSL/TLS) connections
<IfModule mod_ssl.c>
SSLProtocol -all +TLSv1.2
SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
- SSLPassPhraseDialog exec:/Library/Server/Web/Config/apache2/
getsslpassphrase
+ SSLPassPhraseDialog builtin
SSLSessionCache shmcb:/var/run/ssl_scache(512000)
SSLSessionCacheTimeout 300
SSLRandomSeed startup builtin
@@ -573,26 +570,11 @@
AddType application/x-pkcs7-crl crl
</IfModule>
-<IfModule php7_module>
- AddType application/x-httpd-php .php
- AddType application/x-httpd-php-source .phps
- <IfModule dir_module>
- DirectoryIndex index.html index.php
- </IfModule>
-</IfModule>
<IfModule remoteip_module>
RemoteIPHeader X-Forwarded-For
</IfModule>
RewriteEngine On
-RewriteMap webdavmap prg:/Applications/Server.app/Contents/
ServerRoot/usr/libexec/webdavsharing/webdavsharing_mapper
-<IfDefine WEBSERVICE_ON>
- Include /Library/Server/Web/Config/apache2/sites/*.conf
-</IfDefine>
-<IfDefine !WEBSERVICE_ON>
- Include /Library/Server/Web/Config/apache2/sites/
virtual_host_global.conf
- Include /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34580_.conf
- Include /Library/Server/Web/Config/apache2/sites/
0000_127.0.0.1_34543_.conf
-</IfDefine>
\ No newline at end of file
+Include /etc/apache2/sites/*.conf
```

Wiki

Overview

You can migrate your wikis using the following methods:

- Export wiki content to WordPress
- Export wiki content to read-only browsable HTML pages, and use them as a reference
- Export wiki content to JSON
- Devise your own scheme for migration to another wiki system, using one or more of the above formats.

The WordPress system and its import/export format allows migration of wiki content, but without significant metadata and Access Control Lists (ACLs). Use the basic export-to-WXR (WordPress Extended RSS) tool to migrate macOS Server wikis.

NOTE: Major elements of the macOS Server Wiki structure, user model, and semantics—that is, separate wikis with separate ACLs, pages and files associated with specific wikis, user pages and blogs, page histories, directory-based users and groups, themes, avatars, custom banners – won't fully map to any current open source project.

Before you begin

Make sure the Wiki service is turned *on* if you choose to migrate to WordPress. Otherwise, you should turn off the Wiki service.

Post-migration result

After migration you'll have your Wiki data migrated to WordPress or to read-only browsable HTML pages

Manually migrate wikis to WordPress

1. Download and install WordPress. This typically includes separate installation of a database server. The import process will take most information from the WXR files, but will download attachments directly from the Wiki.
2. Make sure the macOS Server installation running Wiki is configured with a valid SSL certificate. WordPress won't import from servers with self-signed certificates.
3. Export the wikis to the file system in WXR format. To export all of the wikis, run the following command:
   `sudo wikiadmin export –all –format wxr –path /tmp/ExportedXMLFiles`
4. The files are only readable by user _teamsserver. Make the files accessible to the others by running the following command:
   `sudo chmod –R 755 /tmp/ExportedXMLFiles`

5. Install and activate the WordPress Importer plugin, log in as an administrative user, then go to the Dashboard.
6. Go to Tools, select Import, then select the exported WXR file.

NOTE: One import step is required for each of the exported WXR files. For each user that created content in macOS Server Wiki, the migration process will present a UI that allows mapping of that user to an existing WordPress user, or creation of a new WordPress user.

7. Manually edit each page then move it from Draft to Published status.
8. If necessary, add access control.

Export a browsable wiki HTML snapshot to file system

You can export the latest revision of all pages in all or selected wikis to the file system, in a location and format where they can be browsed with Safari.

1. Before exporting wikis, plan a replacement for Wikis Access Control Lists, which are not exported. Apache can be configured to restrict access to exported Wiki content, using file-based authentication realms. Detailed guidance on this process is not available in this document, but in general it includes:
   - Devising an authorization scheme to determine which groups have access to which Wikis, based on the pre-export ACLs from the macOS Server Wiki.
   - Configuring mod_authnz_od_basic to require authenticated membership in specific Open Directory groups.
   - Editing desktop Apache's config file to implement the authorization scheme, by adding Directory blocks and Require od_group and Limit directives.
2. Prepare a location for export by running the following commands:
   A. `sudo mkdir /Library/WebServer/Documents/ExportedWikis`
   B. `sudo chown _teamsserver /Library/WebServer/Documents/ExportedWikis`
3. Export the wikis to the file system as .html pages. To export all of the wikis, run the following command:
   `wikiadmin export –all –format pages –path /Library/WebServer/Documents/ExportedWikis`
4. The files are only readable by user _teamsserver. To make them accessible to the others, run the following command:
   `sudo chown –R admin:_www /Library/WebServer/Documents/ExportedWikis`
5. To browse the exported pages locally, by opening the generated index file, without going through the web server by running the following commands:
   A. `cd /Library/WebServer/Documents/ExportedWikis`
   B. `open index.html`
6. To browse the exported pages remotely, through the web server, enter this URL in Safari:
   https://(hostname):8080/ExportedWikis

NOTE: When macOS Server is promoted, the “Desktop” version of apache runs on port 8080. If macOS Server isn’t promoted, then it runs on port 80.

7. To make the exported pages browsable via a WebDAV client from an iOS device, make the ExportedWikis folder a WFS share point by running the following command:
   ```
   sudo wfsctl share /Library/WebServer/Documents/ExportedWikis
   ```
8. To browse WebDAV share points from iOS, for example from the WebDAV Nav app:
   A. Add a server and configure the host name https://(hostname) and an account with Open Directory credentials.
   B. Connect to the server and navigate to the ExportedWikis share point.
   C. Touch the index.html file.

After you finish

If you chose to migrate to WordPress, after successfully importing the wiki data you can turn off the Wiki service in macOS Server.

Calendar and Contacts

Overview

macOS Server includes a Calendar service that implements the CalDAV protocol to support a client/server calendaring and scheduling solution for enterprise and personal calendaring. macOS Server also includes a Contacts service that implements the CardDAV protocol to support a client/server contacts (Address book) solution for enterprise and personal contacts management. Both of these services use the same:

- Code
- Database to store their data
- Directory system to provide users and groups information

The server included in macOS Server is based on the open source CalendarServer project created by Apple. The code that runs in macOS Server is the exact same code available on the open source site. You must download, build, and run it on macOS or another UNIX- or linux-based system. The open source project includes mailing lists to support administrators and developers.

The data and configuration information for the Calendar and Contacts services used by macOS Server are, by default, stored in the "/Library/Server/Calendar and Contacts" directory. When transitioning to the open source CalendarServer, that same directory can be used as-is to allow the new service to use the same data as the old one.

The default configuration for the Calendar and Contacts service used by macOS Server is stored in the Server.app bundle. However, the exact same configuration file is available in the open source CalendarServer project and can be used to setup the new server in the same way as the old—though changes will be needed to configure an alternative directory service and disable the APNs service.

Before you begin

Turn off both the Calendar and Contacts services.

Post-migration result

After migration you'll have one of the following:

- The CalendarServer open source project with all your calendars and contacts imported
- Exported calendar and contact data for import into a third-party solution

Open source replacement options

The open source package can be used as a direct replacement for the Calendar and Contacts services in macOS Server, with the following caveats:

Apple Push Notification Service (APNs)

The Calendar and Contacts service in macOS Server makes use of the APNs to send push notifications to calendar or contacts clients when changes occur on the server. APNs requires special Transport Layer Security (TLS) certificates in order for the server to authenticate to the Apple-hosted service. Those certificates are generated by macOS Server. There is currently no alternative means of generating the required APNs certificates, so push notifications can't be used when macOS Server is no longer available.

Postgres database

macOS Server includes a database that is used for storing the Calendar and Contacts service data. The open source CalendarServer project also expects a postgres service to be available for use. A setup script in the open source project is able to download and build a postgres server. Any existing macOS Server data can then be used with a new database server.

Fast caching

The Calendar and Contacts services used by macOS Server depend on a memcached server to provide an in-memory, fast cache of frequently used data. A memcached server is included as part of macOS Server. The open source CalendarServer also relies on `memcached`, which can be built and installed as part of the open source package.

Scheduling invitations

The Calendar service used by macOS Server has the (optional) ability to send or receive scheduling invitations on behalf of calendar users via email. When enabled, that makes use of the Mail service provided by macOS Server. To use that same feature with the open source CalendarServer, an IMAP service (for incoming email) and an SMTP service (for outgoing email) must be separately provided and configured for use. These mail services can be hosted on the same computer as CalendarServer, or elsewhere.

Automatic startup and shutdown

The startup and shutdown of the Calendar and Contacts services is managed via macOS Server. The open source CalendarServer can be managed via macOS's launchd daemon/agent management system. On other UNIX or linux systems, you can use alternative daemon/agent management tools.

Settings files

macOS Server manages the configuration of the Calendar and Contacts services itself. The configuration is specified in a series of property list (.plist) files. The open source CalendarServer is configured by editing those .plist files directly or via calendarserver_config.

Location and resource management

macOS Server also provides a user interface for managing locations and resources to allow them to be scheduled and managed by calendar users. The open source CalendarServer includes a command-line tool that can be used to manage locations and resources in the same way that macOS Server does.

Command line service management

The open source CalendarServer project includes a comprehensive set of command-line tools for managing the service, several of which are also used by macOS Server.

Web-based calendar client

macOS Server includes a Wiki service with a built-in web-based calendar client that supports accessing calendar data on macOS Server's Calendar service. The web-based calendar is part of the Wiki service and so isn't included in the open source CalendarServer project.

Wiki calendars

The Wiki service also supports calendars associated with wikis. Those calendars won't be accessible directly via the wiki. Apple added a command-line tool (`calendarserver_migrate_wiki`) that does convert wiki principals into resource principals. Because access to the lists of users who previously had access to each wiki isn't available, the command-line tool allows the administrator to specify a read-write delegate. After the migration, user visible names for the migrated wiki → resource principals can now be changed via calendarserver_manage_principals. No UID values need to change, and so the database doesn't require editing.

Backup of service data

macOS Server has the ability to backup service data via macOS's Time Machine service. The open source CalendarServer project doesn't have any automated backup. Instead, administrators must use existing backup strategies for the configuration and database files. There are tools available for the postgres database server to support live backup and replication strategies. See the postgres documentation at:
https://www.postgresql.org/docs/9.5/static/backup.html.

Switch from macOS Server to open source

You can install and configure the open source CalendarServer so that it can run on a server that has macOS Server installed with existing Calendar or Contacts service data. You can use either the manual instructions or a transition script. Both methods are described below.

IMPORTANT: The postgres version that is installed to replace the one bundled with macOS Server must match the version of the existing cluster, otherwise you must manually upgrade the cluster (consult the postgres documentation for guidance). If you have a cluster created by a postgres version earlier than the one installed by the "package" script (currently 9.5.3), you can work around this by manually installing the matching version of postgres before taking the steps below, which will cause the "package" script to use the already-installed version of postgres. You can find the version of the installed postgres software using the command: `pg_config –version`.

Manual instructions

1. Create a standard user named "calendarserver" by choosing System Preferences > Users and Groups, then run the following commands:

NOTE: By creating a new user, you don't need root to build everything, and the server can be configured while logged in as that user. You can still have the server listen on a privileged port such as 80 or 443 because launchd still spawns the master process as root.

2. `sudo –iu calendarserver`
3. `git clone https://github.com/apple/ccs-calendarserver.git`
4. `echo "source CalendarServer/environment.sh" > .profile`

5. `cd ccs-calendarserver`
6. `export USE_OPENSSL=1 # Don't use SecureTransport`
7. `./bin/package /Users/calendarserver/CalendarServer #`
8. `cd /Library/Server/`

NOTE: If you've moved the /Library/Server/ folder, use that path instead of the default one listed.

9. `sudo chown -R calendarserver:staff Calendar\ and\ Contacts`
10. `cd Calendar\ and\ Contacts/Data/Database.xpg/cluster.pg`
11. `mv postgresql.conf postgresql.conf.orig # move it aside because it refers to xpg`
12. `touch postgresql.conf # because it needs to exist`
13. `cd ~/CalendarServer`
14. `mkdir conf run logs`
15. `cd ~/ccs-calendarserver/contrib/conf`
16. `sudo cp org.calendarserver.plist /Library/LaunchAgents`
17. `cp calendarserver.plist ~/CalendarServer/conf`
18. Acquire your certificates, copy them to ~/CalendarServer/certs as:
    - calendarserver.cert.pem
    - calendarserver.key.pem
    - calendarserver.chain.pem

    NOTE: The private key must not be encrypted.
19. `sudo find /var/run/caldavd -user _calendar -exec chown calendarserver:staff {} \;`
20. `sudo launchctl load /Library/LaunchAgents/org.calendarserver.plist`

Instructions using the transition script:

1. Create a standard user named "calendarserver" by choosing System Preferences > Users and Groups.

NOTE: By creating a new user, this avoids requiring root to build everything, and the server can be configured while logged in as that user. You can still have the server listen on a privileged port such as 80 or 443 because launchd still spawns the master process as root.

2. Run the following commands:

   A. `sudo -iu calendarserver`
   B. `git clone https://github.com/apple/ccs-calendarserver.git`
   C. `sudo ~/ccs-calendarserver/contrib/calendarserver_conversion.py`
4. Acquire your certificates, copy them to ~/CalendarServer/certs as:
    - calendarserver.cert.pem
    - calendarserver.key.pem
    - calendarserver.chain.pem

    NOTE: The private key must not be encrypted.
5. `sudo launchctl load /Library/LaunchAgents/org.calendarserver.plist`

Moving macOS Server Calendar or Contacts service data to open source

This section outlines how to install and configure the open source CalendarServer on any supported OS and migrate data from a server that has macOS Server installed with existing Calendar or Contacts service data. The exact steps required will vary based on many factors that are beyond the scope of this section.

1. Use the "QuickStart" instructions on the main CalendarServer project page on Github to install, configure and run a calendar/contacts service with test users and data.
2. Stop the test service and remove any postgres and account data that was generated.
3. Transfer the postgres data from macOS Server to the new service and reset the permissions on all the data files. Use `pg_dump` and `pg_restore` when moving between incompatible systems.
4. Transfer the account information from macOS Server to the new directory service.
5. Start the new service.

Moving macOS Server Calendar or Contacts service data to some other service

This section outlines how to export existing Calendar or Contacts service data in order to import that data into another calendar or contacts solution. The exact steps required to import the data into the replacement solution will vary based on many factors that are beyond the scope of this section.

Calendar data

Importing calendar data that contains invitations (events with organizer and attendee properties) can trigger scheduling messages to be sent. This isn't ideal as attendees potentially receive duplicate invitations for events they already have and which haven't changed (and in addition those could be for events in the past that are no longer relevant).

It is possible that the new service provides a mechanism to migrate calendar data in such a way as to avoid sending out duplicate invitations. If so, then the `calendarserver_export` tool can be used to dump the data from each users' calendars to a file which can then be imported to the new service.

The `calendarserver_export` tool can be used for situations where users have not been doing any scheduling at all.

1. Run the following commands:
2. `sudo –iu calendarserver`
3. `. ~/CalendarServer/environment.sh`
4. `calendarserver_export –directory ~/exported_calendar_data –all –mailto`

A directory named ~/exported_calendar_data is created and contains an .ics file for each calendar.

Contacts data

User driven solution

The simplest approach is to setup the new service, have all users of the old service add the new service to their clients, then copy data over from the old service to the new one using their contacts app.

Administrator driven solution

Each users' data on the old service can be exported in the form of a file containing a set of vCards for each contact, and those can be imported directly into the new service.

1. Run the following commands:
2. `sudo –iu calendarserver`
3. `. ~/CalendarServer/environment.sh`
4. `calendarserver_export –directory ~/exported_contact_data –all –contacts`

A directory named ~/exported_contact_data will be created and it will contain a .vcf file for each address book.

FTP

Overview

FTP (File Transfer Protocol) allows files to be transferred between computers. Because FTP isn't secure, you should use SFTP (Secure File Transfer Protocol) instead.

Post-migration result

After migration you'll have one of the following:

- A secure FTP (SFTP) service managed by System Preferences
- SFTP enabled on macOS

Switch from FTP on macOS Server to SFTP/SSH on macOS for file transfer service

1. To enable SFTP and SSH, open System Preferences > Sharing and check the Remote Login box.
2. Next, select the users who should have access to transfer files to the Mac.

Software update

Overview

The Software Update service of macOS Server is being replaced by managed software updates from a supported mobile device management (MDM) solution and the content caching service in macOS High Sierra or later. For more information on configuring content caching, see the following Apple Support articles:

- **About content caching:** https://support.apple.com/guide/mac-help/about-content-caching-mchl9388ba1b/mac
- **Manage content caching:** https://support.apple.com/guide/mac-help/manage-content-caching-mchl3b6c3720/mac
- **Set up content cache clients, peers, or parents:** https://support.apple.com/guide/mac-help/set-up-content-cache-clients-peers-or-parents-mchl9b56e1cf/mac

Apple Inc.
One Apple Park Way
Cupertino, CA 95014
408-996-1010

March 2021 028-00434