a Hewlett Packard Enterprise company

ClearPass Onboard

Using Active Directory Certificate Services

Open Source Code

Certain Aruba products include Open Source software code developed by third parties, including software code subject to the GNU General Public License (GPL), GNU Lesser General Public License (LGPL), or other Open Source Licenses. The Open Source code used can be found at this site:

http://www.arubanetworks.com/open_source

Revision History

Date	Author	Revision Description
4/7/14	Bob Filer	v1.0. Initial Revision
4/15/14	Bob Filer	v1.1. Minor edits (typos & grammar) plus new sections on IIS Authentication Providers and cURL troubleshooting.
5/25/16	Bob Filer	v1.2. Removed the requirement to join ClearPass to an Active Directory domain.

Table of Contents

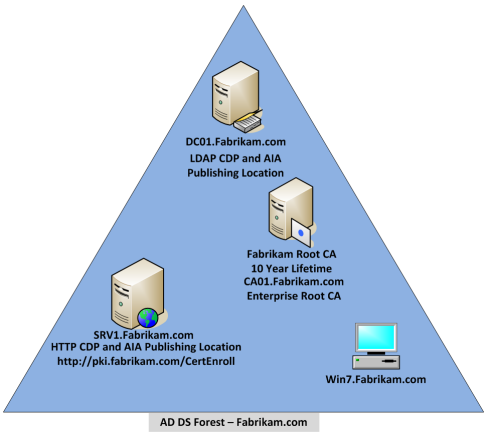
Introduction

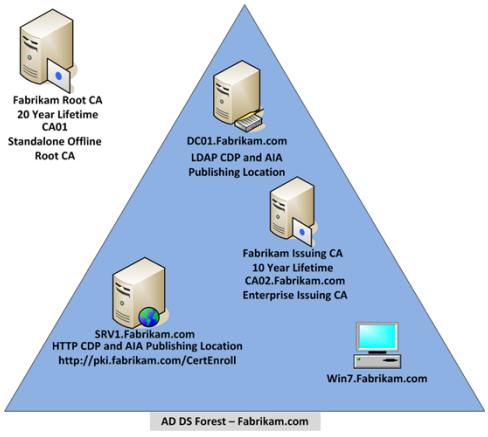
This document explains the use of Microsoft Active Directory Certificate Services (ADCS) to sign Onboard device TLS certificates, versus using the ClearPass Onboard Certificate Authority (CA) to do the signing. The use of ADCS provides for centralized management of TLS certificates including expiration, revocation, and deletion through ADCS.

This feature has been designed to provide an easy integration of ClearPass Onboard into an existing Public Key Infrastructure (PKI) deployment based on Active Directory Certificate Services.

PKI Architecture

Microsoft ADCS supports numerous deployment models including single-, two-, and three-tiered PKI hierarchies. The diagrams below represent a couple of example deployments of ADCS. The diagram on the left represents what Microsoft calls a two-tiered architecture with a single online root CA and separate publishing (or issuing) servers. The diagram on the right is a three-tiered PKI architecture consisting of an offline root CA, intermediate (or subordinate) CAs, and numerous publishing servers. No particular architecture is correct for a given organization – it will depend upon numerous factors such as the size of the organization, number of deployed devices, geographic dispersion, etc. However, ClearPass integration with ADCS is possible in all Microsoft PKI infrastructures as long as the guidelines described later in this document are followed.





NOTE Numerous web resources are available that explain the various ADCS models and how to configure them.

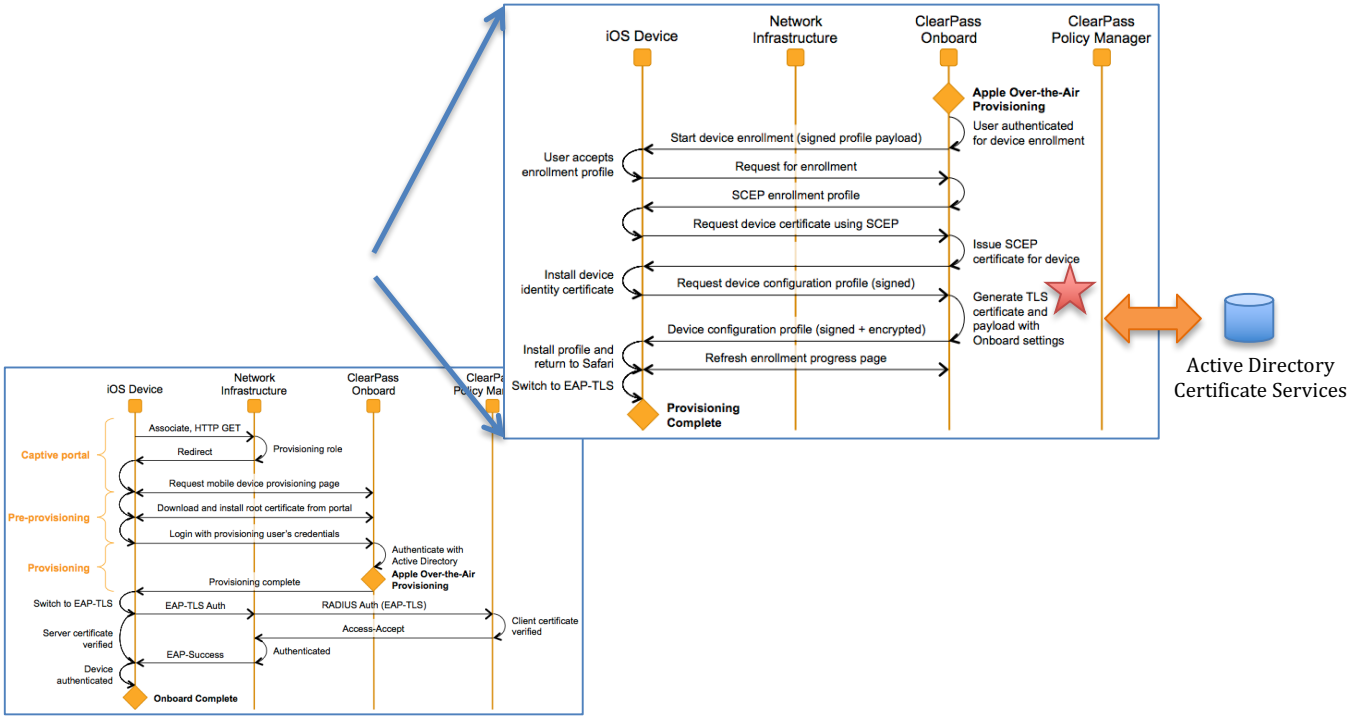
The configuration and deployment of an ADCS PKI infrastructure is beyond the scope of this document, however the section “ADCS Requirements” describes the minimum requirements of an ADCS installation in order to take advantage of the ADCS integration feature in ClearPass Onboard.

Onboard Workflow

ClearPass Onboard makes use of the ADCS web enrollment to sign the device TLS certificate(s). This is in contrast to other solutions that make use of technologies such as SCEP. What this means, as explained in detail later, is that you will have to enable certificate web enrollment for any user that wants to onboard their device(s).

The following diagrams show the Onboard workflow for iOS devices. The workflow for other operating systems is similar, but there is no over-the-air provisioning. Instead, the Aruba QuickConnect client app is required to complete all the steps in the box on the right. The key step to note in these diagrams is at the point of TLS certificate generation (designated with a star below).

In the typical Onboard case, a Certificate Signing Request (CSR) would be generated and subsequently signed by the Onboard Certificate Authority (CA). When integrating with ADCS, the CSR is still generated by Onboard, but the CSR is instead signed by ADCS. To accomplish this signing, ClearPass will sign into the ADCS web enrollment page (https://ADCS_Web_Enrollment_Server/certsrv/certfnsh.asp) using the Active Directory credential of the user that is attempting to onboard their device. ClearPass will POST the CSR to this web page and then retrieve the resulting signed certificate (certnew.cer).



ADCS Requirements

There are a few requirements of the ADCS environment that must be met in order for ClearPass Onboard to function correctly.

Components Required

The following Windows Server Roles and Role Services must be installed in the PKI hierarchy. Note, however, they do not have to be installed on the same server.

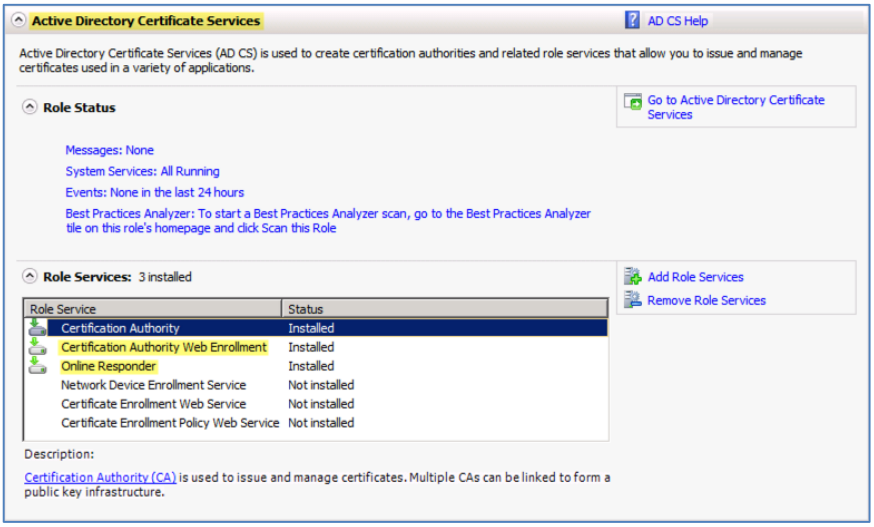
- Active Directory Certificate Services
  - Certification Authority
  - Certification Authority Web Enrollment
  - Online Responder
- Active Directory Domain Services
- Web Server (IIS)

Web Enrollment

The ADCS web enrollment interface (typically http://ADCS_Web_Enrollment_Server/certsrv) must be available for access from ClearPass, and users must have permission to request their own certificates. When ClearPass requests that ADCS sign the TLS CSR, it will log into the web enrollment interface as the user, POST the Certificate Signing Request (CSR), and download the signed TLS certificate.

Depending upon the Microsoft PKI deployment model chosen, the Web Enrollment Server and the Certificate Services Server may not be the same server. There is no requirement that the ADCS server run any web services at all. However, a server running Microsoft IIS with the Certification Authority Web Enrollment and Online Responder role services must be available in the environment, appropriately connected to ADCS. The ClearPass integration with ADCS only works through the Web Enrollment interface.

NOTE Certification Authority Web Enrollment and Online Responder role services belong to the Active Directory Certificate Services role, not the Web Server (IIS) role. However, the IIS role is required to install those role services.



NOTE ClearPass signs into the web enrollment interface as the user requesting the certificate. No ClearPass service account is used. Depending upon the ADCS environment – whether users can request their own certificates – this might pose a problem.

Certificate Templates

The ClearPass Onboard configuration permits you to specify the ADCS Certificate Template used to create the requested TLS certificate. By default, ADCS includes a template named *User* that can be used for this purpose. The *User* template, however, is limited by the certificate fields it supports. Specifically, the *User* template does not allow the setting of the Mobile Device Provisioning Service (MDPS) attributes in the Subject Alt Name (these are still stored locally in the Onboard database). This limitation doesn't prevent the use of the TLS certificates for authentication, but may affect authorization using the certificates depending upon configuration.

In addition, the default *User* template does not permit setting an arbitrary Subject within the certificate. Instead, the Subject is set based information from Active Directory including the email address. This may or may not be a problem in your environment.

In order to create a certificate that supports the MDPS attributes and arbitrary subject, a custom certificate template is required. A custom certificate template can also be used to control other attributes such as certificate lifetime and key strength.

NOTE Using a custom certificate template requires enabling the capability to create certificates with arbitrary subjects. This opens a potential security hole if the certificate web enrollment interface is generally available to users. Users that can create certificates with arbitrary subjects have the potential to create certificates that can impersonate other users. It's suggested that access to the web enrollment interface only be permitted to ClearPass, using network- or host-based firewalling or certificate policies.

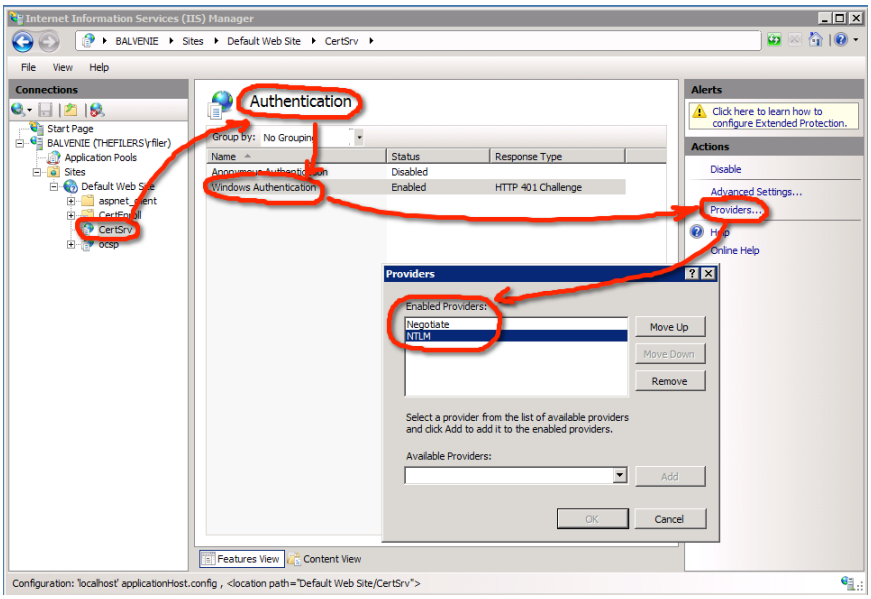
Internet Information Server (IIS) Authentication

When requesting a certificate, ClearPass connects to the ADCS web enrollment interface (typically http://ADCS_Web_Enrollment_Server/certsrv) to make the request. This page resides on an IIS server, which may or may not also be running the Certificate Authority. ClearPass will log into this web page with the credential of the user requesting the certificate, using the NTLM authentication method. This is important, because depending upon the configuration of your Active Directory domain, you may have to make modification to the authentication provider order for the /certsrv web page.

NOTE You will only need to make this change if you experience the failure condition described in the Appendix section “Testing with cURL”.

The screen capture below shows Internet Information Services Manager and the /certsrv web page Authentication settings. The default list of Providers has “Negotiate” (indicating Kerberos authentication) first and NTLM second. As stated, ClearPass uses NTLM authentication, not Kerberos. However, in most scenarios the default order is acceptable because ClearPass and IIS will properly skip the Kerberos authentication and move onto NTLM – the process will succeed. Depending upon your AD domain settings, however, this initial Kerberos authentication may fail and NTLM will not be attempted resulting in a failed authentication attempt. This will result in no user certificate being issued.

To overcome this Kerberos authentication failure, you can simply change the Provider order by moving NTLM to the top of the list.



NOTE Consult your server and/or security team for the possible ramifications of making this change.

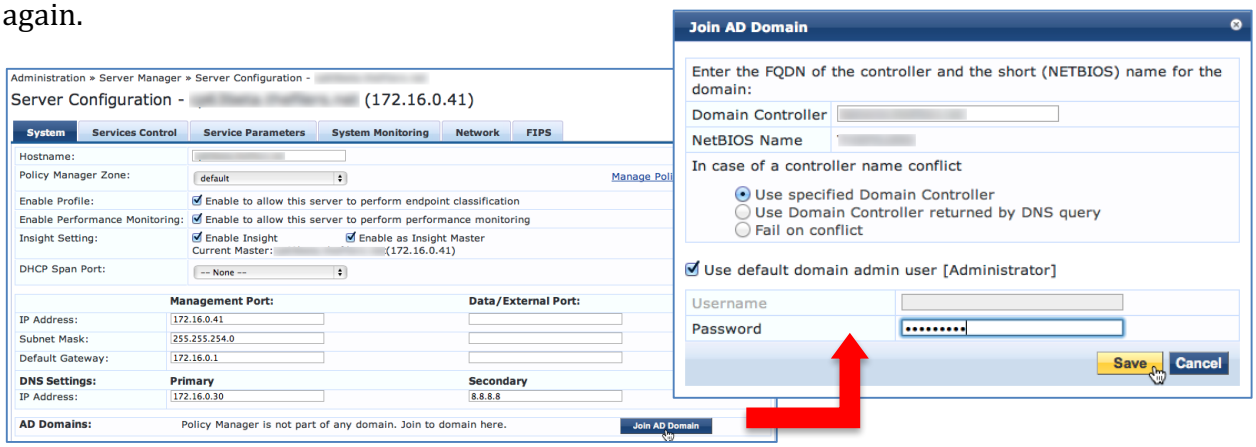
ClearPass Configuration

The following ClearPass configuration steps are not all the steps required to completely configure Onboarding, but are only descriptions of the changes required to the Onboard configuration to support Active Directory Certificate Services signing of your Onboard TLS certificates. For a complete description of Onboard configuration, see the ClearPass Administrator's Guide.

Join CPPM to the Active Directory Domain (OPTIONAL)

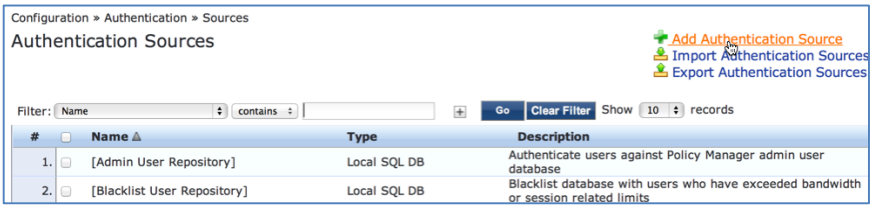
This step is OPTIONAL: Your network configuration and authentication requirements will dictate whether or not you need to join CPPM to the AD domain (such as the need to support PEAP/MSCHAPv2). Joining CPPM to the domain is NOT required for the certificates to be properly signed. ClearPass can be joined to multiple domains to support complex PKI environments.

Navigate to **Administration > Server Manager > Server Configuration** and click on the **Server Name**. Click **Join AD Domain**. Complete the form and click **Save**, then click **Save** again.

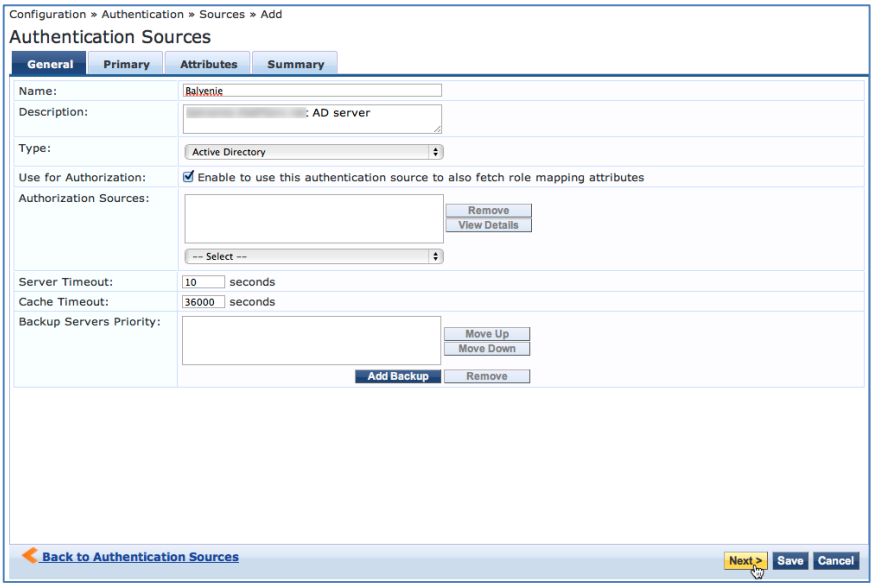


Create an AD Authentication Source

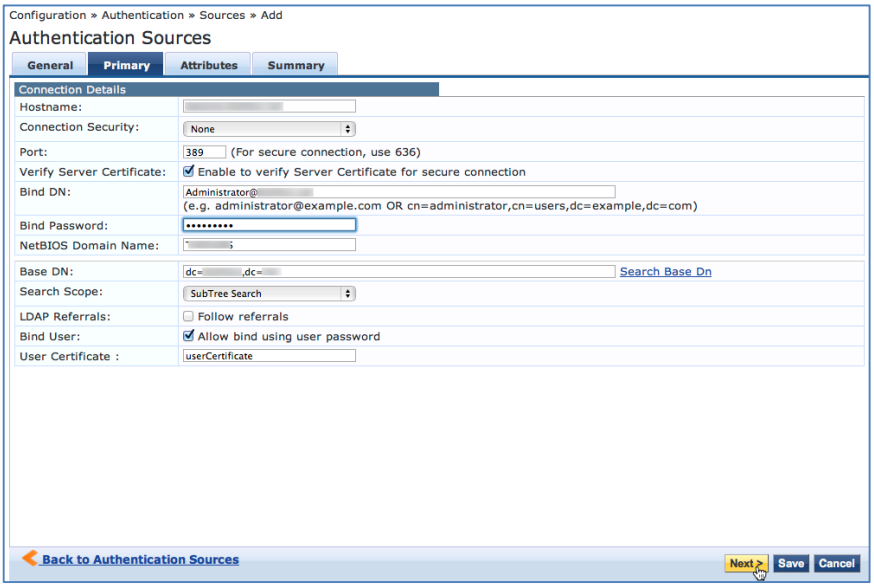
During the certificate signing process, CPPM will log into the ADCS server via the /certsrv web form with the username and password used to onboard. You should configure an AD authentication source to validate the user credentials prior to submitting to ADCS. Go to **Configuration > Authentication > Sources** and click **Add Authentication Source**.



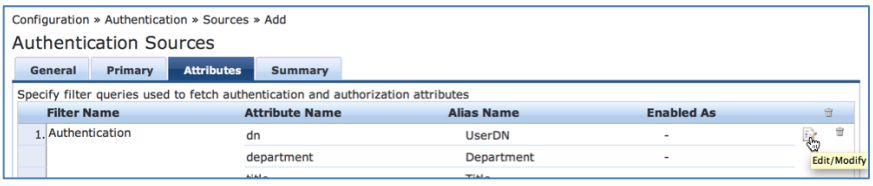
On the **General tab**, enter a **Name** for the source, select the **Type** as *Active Directory*, then click **Next**.



On the **Primary tab**, enter the AD **Hostname**, the **Bind DN** and **Bind Password**, and modify other parameters as needed for your environment. Click **Next**.

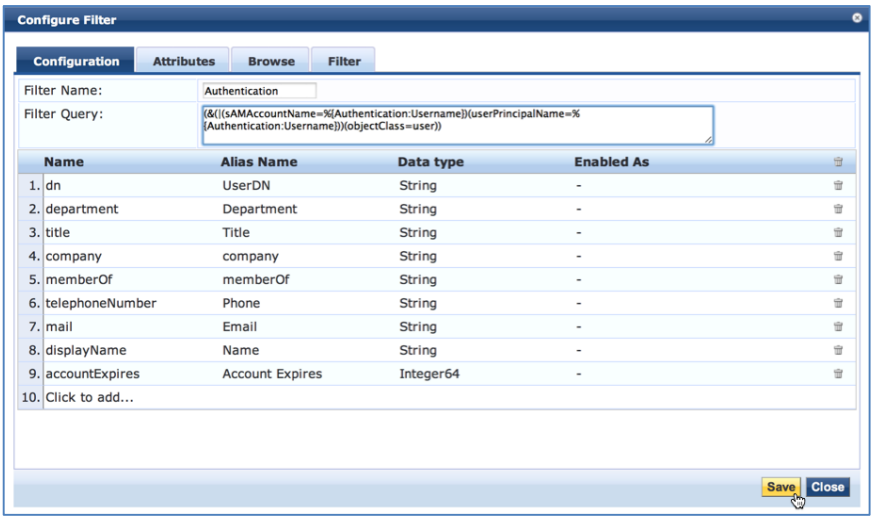


On the **Attributes tab**, you need to modify the **Filter Query** for the filter *Authentication*. Click on the **Edit** icon ( )for the filter *Authentication*.

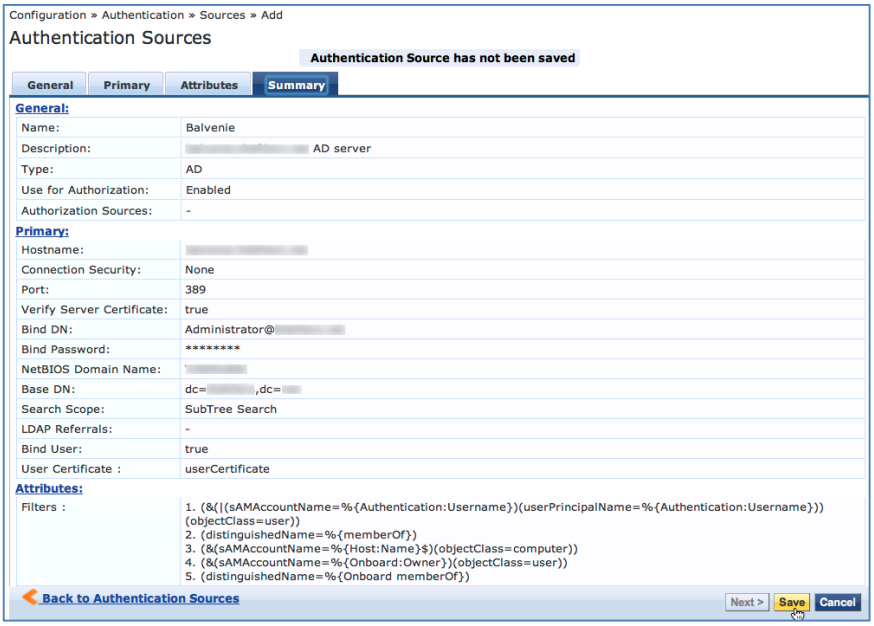


Modify the **Filter Query** so that it looks like the text below. You can simply copy this text and paste it into the **Filter Query** block. The modification to the **Filter Query** permits the user to sign in with either just their username (sAMAccountName) or their email address (userPrincipalName). Click **Save** to save the **Filter Query** changes.

```
(&(|(sAMAccountName=%{Authentication:Username})(userPrincipalName=%{Aut
hentication:Username}))(objectClass=user))
```



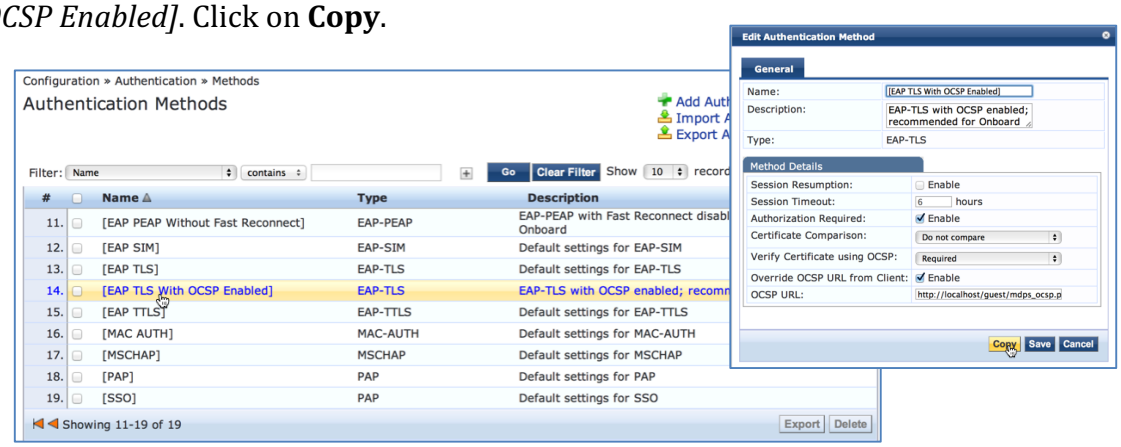
Click **Next** to go to the **Summary tab**, then click **Save**.



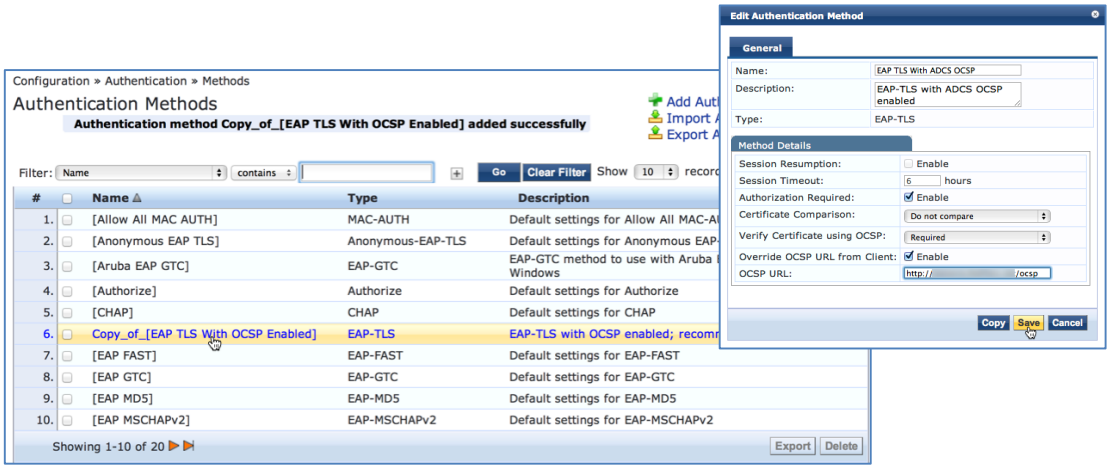
Create a New Authentication Method

You need to create a new authentication method that will query the OCSP responder on the ADCS server, instead of the local OCSP responder. To do this, begin by making a copy of the

authentication method *[EAP TLS with OCSP Enabled]* and rename it to *EAP TLS with ADCS OCSP*. To begin, go to **Configuration > Authentication > Methods** and click on *[EAP TLS with OCSP Enabled]*. Click on **Copy**.



Click on the newly copied authentication method. Change the **Name** to *EAP TLS with ADCS OCSP*, and modify the **OCSP URL** to match the actual ADCS OCSP URL (typically http://ADCS_Web_Enrollment_Server/ocsp) and click **Save**.

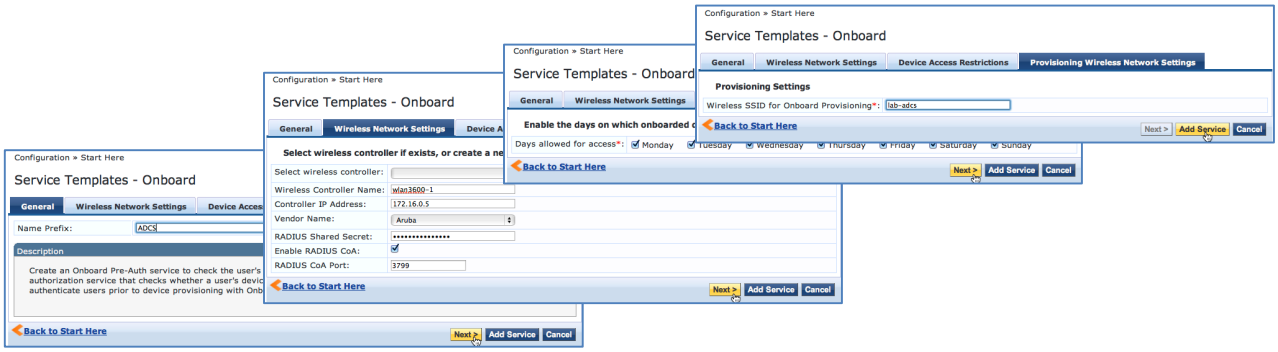


Create New Services

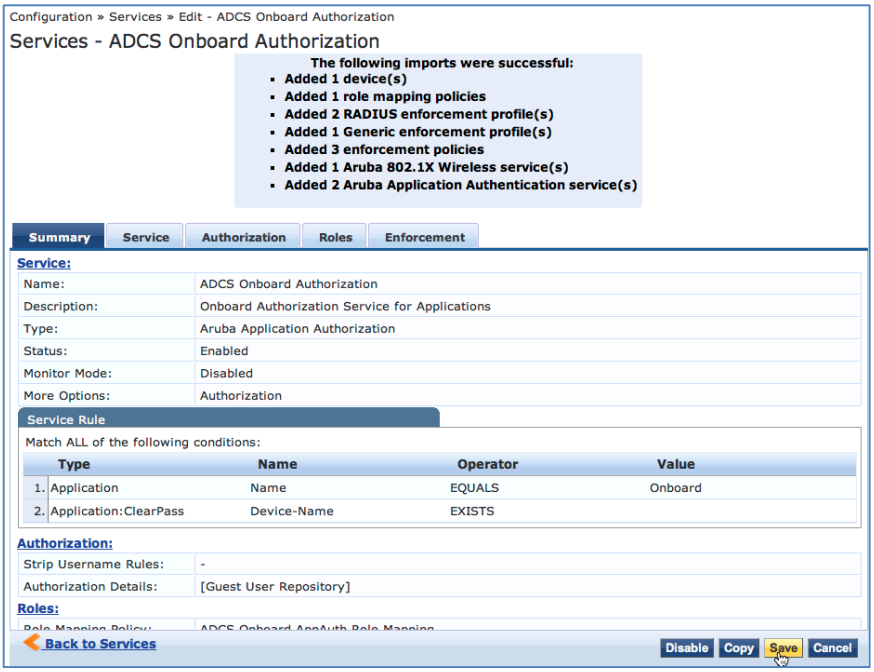
Create new services to support Onboarding of devices. To do so, use the Service Template *Onboard*. This will create numerous configuration elements, including:

1. Services: Onboard Provisioning, Onboard Authorization, Onboard Pre-Auth
2. Role Mapping: Onboard AppAuth Role Mapping
3. Enforcement Policies: Onboard AppAuth Policy, Onboard Pre-Auth Policy, Onboard Provisioning Policy
4. Enforcement Profiles: Onboard Post-Provisioning, Onboard Pre-Provisioning, Onboard Session Timeout

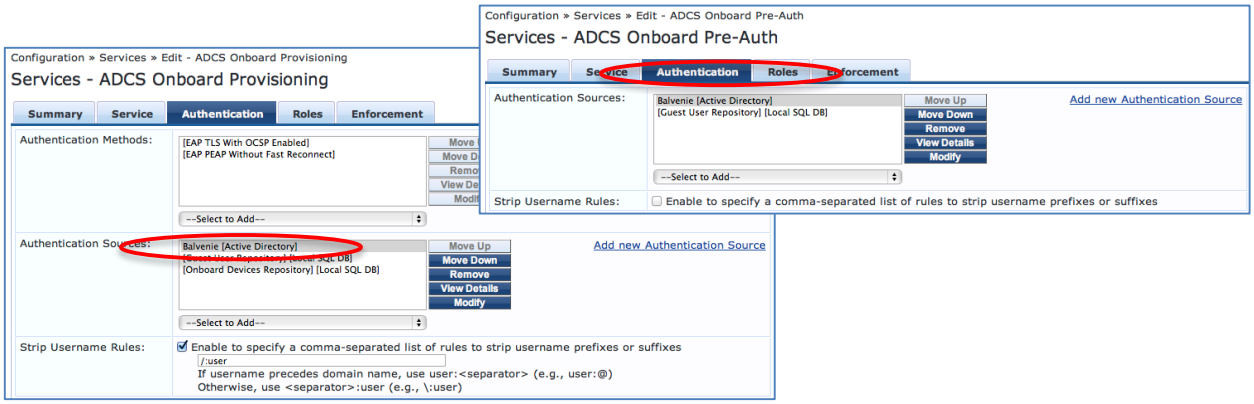
NOTE These instructions regarding the use of Service Templates apply to CPPM version 6.3 and above. The Service Templates in versions prior to 6.3 operate in a slightly different manner.



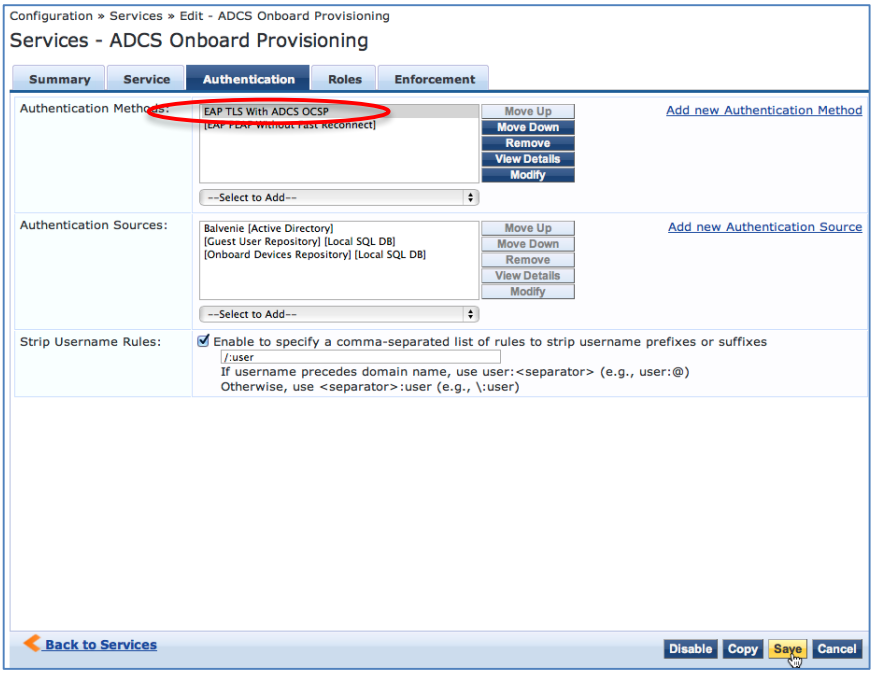
Once you reach the **Summary tab**, click **Save**.



Next you need to modify the services *Onboard Provisioning* and *Onboard Pre-Auth* to add your Active Directory server as an authentication source. Go to **Configuration > Services** and click on each of the above services in turn. For each service, go to the **Authentication tab**, add your AD server to the list of **Authentication Sources**, and move the AD server to the top of the list. Click **Save** after modifying each service.

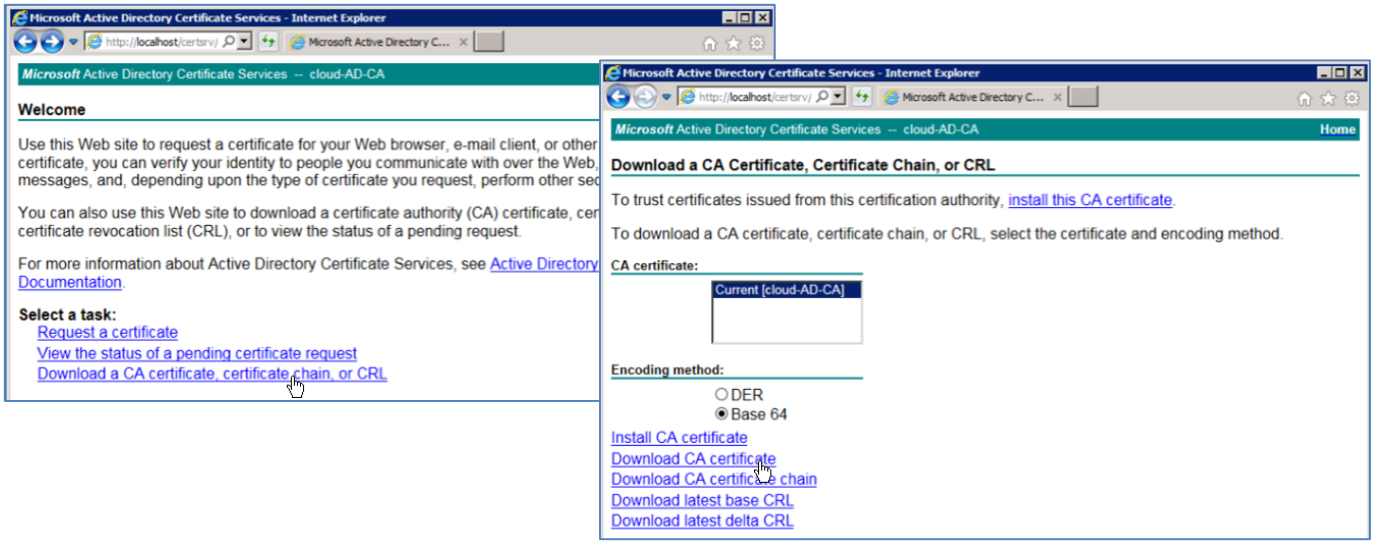


You need to further modify the service *Onboard Provisioning* to remove the **Authentication Method** *[EAP TLS with OCSP Enabled]* and add the new **Authentication Method** *EAP TLS with ADCS OCSP* created above. To do so, go to **Configuration > Services**, click on the *Onboard Provisioning* service, then on the Authentication tab. Remove the Authentication Method **Authentication Method** *[EAP TLS with OCSP Enabled]*, then add *EAP TLS with ADCS OCSP*. Place this method at the top of the list.

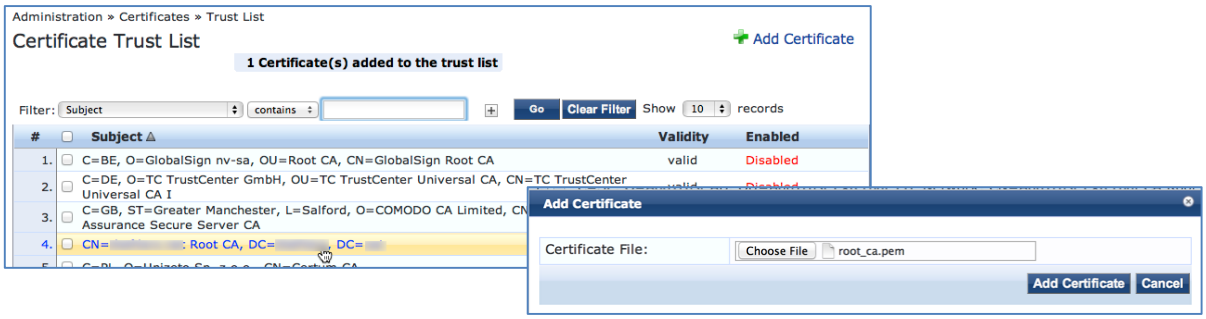


Modify Certificate Trust Lists

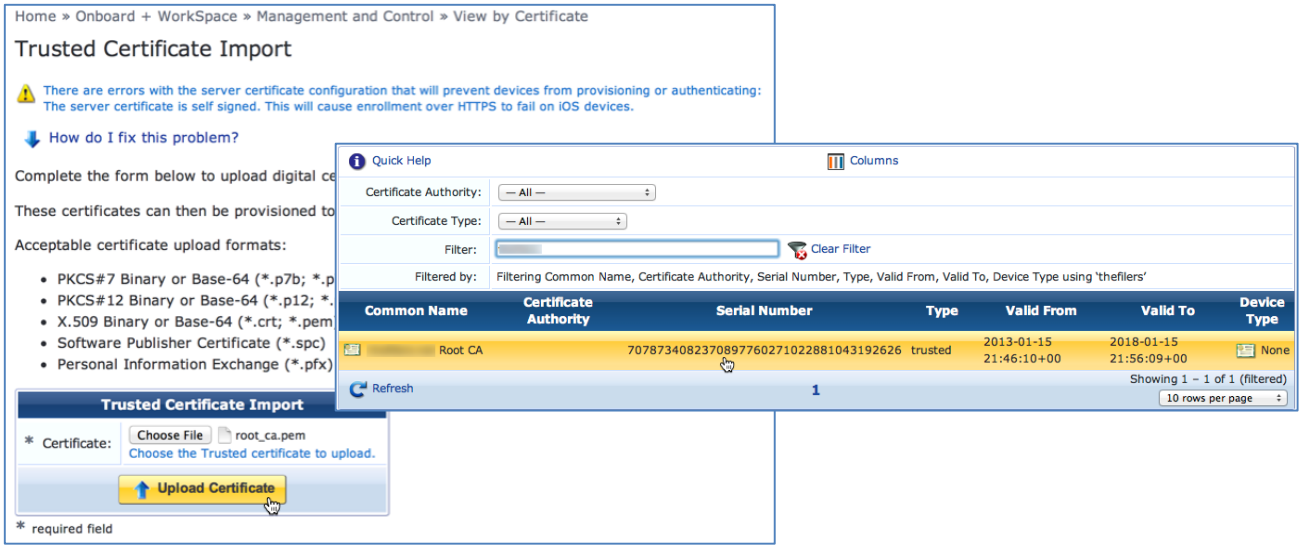
You need to add the ADCS root certificate to both the CPPM Certificate Trust List and to the Onboard Certificate Store. To begin, you need to first download the root certificate from the Active Directory Certificate Server. You can obtain the root certificate by browsing to the ADCS Web Enrollment Server, to the URL http://ADCS_Web_Enrollment_Server/certsrv. Click on the **Download a CA certificate, certificate chain, or CRL** link, select *Base 64* as the **Encoding Method**, then click on the **Download CA certificate** link.



To add the root certificate to CPPM, go to **Administration > Certificates > Trust List** and click **Add Certificate**. Select the certificate that you downloaded from ADCS, click **Add Certificate**.



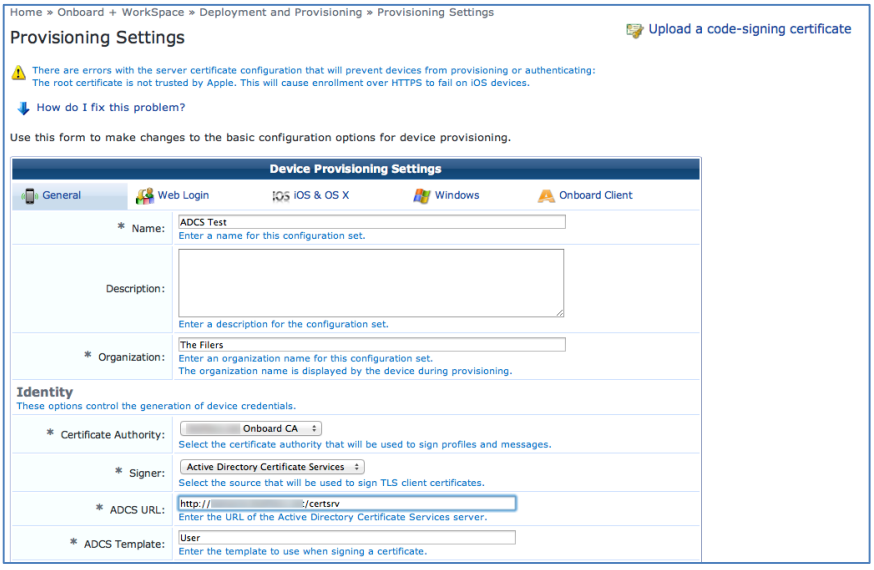
You also need to add the ADCS root certificate to the ClearPass Onboard certificate store. Within ClearPass Guest, go to **Onboard + WorkSpace > Management and Control > View by Certificate** and click **Upload a trusted certificate**. Select the root certificate and click **Upload Certificate**.



Configure Onboard

NOTE There are several methods to configure Onboard, and numerous settings that can be applied to onboarded devices. An explanation of those methods and options is beyond the scope of this paper.

The one significant change that you need to make to your Onboard configuration is to enable ADCS signing of the Onboard TLS certificates. Within ClearPass Guest go to **Onboard + WorkSpace > Deployment and Provisioning > Provisioning Settings > General tab**, and set the **Signer** as *Active Directory Certificate Services*. Enter the **ADCS URL**, which is typically http://ADCS_Web_Enrollment_Server/certsrv. Also enter the **ADCS Template**, which should either be *User* (if you're using the default certificate template), or the name of your custom template.



Click **Save** to save the provisioning settings page.

Limitations/Caveats

There are some limitations and caveats that you should be aware of when implementing ADCS signing of your TLS certificates. These have been noted previously in this document, but are collected here for reference.

- Though the TLS certificate is signed by ADCS, a copy of the certificate is maintained within ClearPass Onboard. This is to facilitate certificate expiration checks so that users can be notified when their certificate is about to expire. If configured correctly, the OCSP check will be performed against ADCS, which is where the certificate management should take place.
- You cannot revoke the TLS certificates from within Onboard, you must do so within ADCS. This centralizes management of the device certificates.
- You need to configure an ADCS URL for OCSP. As mentioned previously, all TLS certificate management should be done through ADCS. If using the default User certificate template, a manual override of the OCSP URL is required since the template doesn't include that attribute.
- Users must have permissions to use the ADCS web enrollment interface. This implies that they can manually create certificates outside of the Onboard workflow if they know the ADCS certificate enrollment URL. You can use a firewall to prevent access to this URL from all but ClearPass.
- You should also limit access to the ADCS web enrollment interface if you're using custom templates that allow setting of the certificate subject. Making the web interface generally available would allow creation of certificates with arbitrary subjects, meaning a user could request a certificate in the name of another user.
- All users must be in Active Directory. Users from other authentication sources such as Local or Guest cannot be used as they will not have the permissions to create certificates.
- ADCS is only used for the final TLS certificate. Certificates used during enrollment (temporary identity certificate, profile signing certificate) must still be managed by Onboard.

Appendix

Installing and Configuring Microsoft ADCS

The installation and configuration of Microsoft Active Directory Certificate Services is beyond the scope of this paper, however the following linked articles might be of use to you. Aruba takes no responsibility for the content of the articles.

- Designing and Implementing a PKI: Part I Design and Planning
- Designing and Implementing a PKI: Part II Implementation Phases and Certificate Authority Installation
- Designing and Implementing a PKI: Part III Certificate Templates
- Designing and Implementing a PKI: Part IV Configuring SSL for Web Enrollment and Enabling Key Archival
- Designing and Implementing a PKI: Part V Disaster Recovery
- OCSP
    - Implementing an OCSP responder: Part I Introducing OCSP
    - Implementing an OCSP responder: Part II Preparing Certificate Authorities
    - Implementing an OCSP responder: Part III Configuring OCSP for use with Enterprise CAs

MDPS SAN Fields

OID	Meaning	Example
1.3.6.1.4.1.14823.1.5.1.1	Device Type	iOS
1.3.6.1.4.1.14823.1.5.1.2	UDID	b837d9e7c2a9c9c5fe263746e765c3160b6e8f
1.3.6.1.4.1.14823.1.5.1.5	MAC Address	01:02:03:04:05:06
1.3.6.1.4.1.14823.1.5.1.6	Product Name	iPad2,5
1.3.6.1.4.1.14823.1.5.1.7	Product Version	10B329
1.3.6.1.4.1.14823.1.5.1.8	Username	fred

Testing with cURL

Using the open source command line utility cURL, you can test whether the user account you are using to onboard can successfully log into the Web Enrollment web page. This tool should be used from a system other than ClearPass, as you won't have access to the shell on ClearPass.

cURL should be run with the following options. These correspond exactly to the options that ClearPass uses to request the certificate.

- `-v`: verbose
- `-k`: insecure (ignores SSL certificate errors)
- `-u user[:password]`: the username to sign in with and optionally the password. If the password isn't supplied on the command line, it will be asked for.
- `--ntlm`: force NTLM authentication method
- `URL`: this should be https://Web_Enrollment_Certificate_Server/certsrv/. Make sure to include the trailing slash.

Success

In a successful request, you should look for a "200 OK" HTTP response, as well as Javascript from the /certsrv page and the <HTML> </HTML> tags.

```
> curl -v -k -u fred:password --ntlm https://balvenie.example.com/certsrv/
* About to connect() to balvenie.example.com port 443 (#0)
*   Trying 172.16.0.30... connected
* Connected to balvenie.example.com (172.16.0.30) port 443 (#0)
* successfully set certificate verify locations:
*   CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CApath: none
* SSLv3, TLS handshake, Client hello (1):
* SSLv3, TLS handshake, Server hello (2):
* SSLv3, TLS handshake, CERT (11):
* SSLv3, TLS handshake, Server finished (14):
* SSLv3, TLS handshake, Client key exchange (16):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSL connection using AES128-SHA
* Server certificate:
*        subject: CN=balvenie.example.com
*        start date: 2013-12-05 00:48:17 GMT
*        expire date: 2014-12-05 00:48:17 GMT
*        subjectAltName: balvenie.example.com matched
*        issuer: DC=net; DC=thefilers; CN=example.com Root CA
*        SSL certificate verify result: unable to get local issuer certificate (20),
continuing anyway.
* Server auth using NTLM with user 'fred'
> GET /certsrv/ HTTP/1.1
> Authorization: NTLM TlRMTVNTUAABAAAABoIIAAAAAAAAAAAAAAAAAAAAAAA=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 OpenSSL/1.0.1e
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: balvenie.example.com
> Accept: */*
>
< HTTP/1.1 401 Unauthorized
< Content-Type: text/html; charset=us-ascii
```

```
< Server: Microsoft-HTTPAPI/2.0
< WWW-Authenticate: NTLM
TlRMTVNTUAACAAAACQAJADgAAAAGgokCDACJuKfa8awAAAAAAAAAAAKYApgBBAAAABgGxHQAAAA9USEVGSUxFUlMCA
BIAVABIAEUARgBJAEwARQBSAFMAAQAQAEIAQQBMAFYARQBOAEkARQAEABoAdABoAGUAZgBpAGwAZQByAHMALgBuAG
UAdAADACwAYgBhAGwAdgBlAG4AaQBlAC4AdABoAGUAZgBpAGwAZQByAHMALgBuAGUAdAAFABoAdABoAGUAZgBpAGw
AZQByAHMALgBuAGUAdAAHAAgAHyvIGbtSzwEAAAAA
< Date: Mon, 07 Apr 2014 23:42:56 GMT
< Content-Length: 341
<
* Ignoring the response-body
* Connection #0 to host balvenie.example.com left intact
* Issue another request to this URL: 'https://balvenie.example.com/certsrv/'
* Re-using existing connection! (#0) with host balvenie.example.com
* Connected to balvenie.example.com (172.16.0.30) port 443 (#0)
* Server auth using NTLM with user 'fred'
> GET /certsrv/ HTTP/1.1
> Authorization: NTLM
TlRMTVNTUAADAAAAGAAYAEAAAAAYABgAWAAAAAAAAABwAAAABAAEAHAAAAAEAAQAdAAAAAAAAAAAAAAAABoKJAnc5k
zbN8tn0AAAAAAAAAAAAAAAAAAAAAAPN+wXf8WVDIKtUpPd8fYQDl78LULbJ83GZyZWRjcHBt
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 OpenSSL/1.0.1e
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: balvenie.example.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Cache-Control: private
< Content-Length: 3639
< Content-Type: text/html
< Server: Microsoft-IIS/7.5
< Set-Cookie: ASPSESSIONIDAGTCQBST=EPGBDJJBGNNCBHNEFCNJDMFB; secure; path=/
< Persistent-Auth: true
< X-Powered-By: ASP.NET
< Date: Mon, 07 Apr 2014 23:42:56 GMT
<


<Script Language="JavaScript">
        //---------------------------------------------------------------------
        // convert a (signed) number into a (unsigned) hex string
        function toHex(number) {
                var sRight=(number&0x0FFFFFFF).toString(16).toUpperCase();
                sRight="0000000".substring(0, 7-sRight.length)+sRight;
                return ((number>>28)&0x0000000F).toString(16).toUpperCase()+sRight;
        }

</Script>
<HTML>
...
</HTML>
```

Failure

If the request fails, it could be for several reasons.

1. It may be that ClearPass simply cannot reach the Web Enrollment Server. This should be obvious from error messages on ClearPass and can be tested through the ClearPass console using the `network ping` command.
2. It may be that ClearPass cannot access the Web Enrollment page on the Web Enrollment server. This is a little harder to troubleshoot because of the limited tools on ClearPass. However, check you firewall rules and logs to see that ClearPass is making a request but possibly being denied.
3. If you Require Extended Protection on the Windows Authentication service for the /certsrv web page, then you also get an error as shown in the following capture. The

error returned is a little misleading because it looks like there is a username/password mismatch but is actually caused by the Extended Protection feature which is incompatible with the version of NTLM authentication used by cURL.

```
> curl -v -k -u fred:password --ntlm https://balvenie.example.com/certsrv/
* About to connect() to balvenie.example.com port 443 (#0)
*   Trying 172.16.0.30... connected
* Connected to balvenie.example.com (172.16.0.30) port 443 (#0)
* successfully set certificate verify locations:
*   CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CApath: none
* SSLv3, TLS handshake, Client hello (1):
* SSLv3, TLS handshake, Server hello (2):
* SSLv3, TLS handshake, CERT (11):
* SSLv3, TLS handshake, Server finished (14):
* SSLv3, TLS handshake, Client key exchange (16):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSL connection using AES128-SHA
* Server certificate:
*        subject: CN=balvenie.example.com
*        start date: 2013-12-05 00:48:17 GMT
*        expire date: 2014-12-05 00:48:17 GMT
*        subjectAltName: balvenie.example.com matched
*        issuer: DC=net; DC=thefilers; CN=example.com Root CA
*        SSL certificate verify result: unable to get local issuer certificate (20),
continuing anyway.
* Server auth using NTLM with user 'fred'
> GET /certsrv/ HTTP/1.1
> Authorization: NTLM TlRMTVNTUAABAAAABoIIAAAAAAAAAAAAAAAAAAAAAAA=
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 OpenSSL/1.0.1e
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: balvenie.example.com
> Accept: */*
>
< HTTP/1.1 401 Unauthorized
< Content-Type: text/html; charset=us-ascii
< Server: Microsoft-HTTPAPI/2.0
< WWW-Authenticate: NTLM
TlRMTVNTUAACAAAACQAJADgAAAAGgokCxzX1LGkHpCIAAAAAAAAAAAKYApgBBAAAABgGxHQAAAA9USEVGSUxFUlMCA
BIAVABIAEUARgBJAEwARQBSAFMAAQAQAEIAQQBMAFYARQBOAEkARQAEABoAdABoAGUAZgBpAGwAZQByAHMALgBuAG
UAdAADACwAYgBhAGwAdgBlAG4AaQBlAC4AdABoAGUAZgBpAGwAZQByAHMALgBuAGUAdAAFABoAdABoAGUAZgBpAGw
AZQByAHMALgBuAGUAdAAHAAgA31eHJmRTzwEAAAAA
< Date: Tue, 08 Apr 2014 19:53:02 GMT
< Content-Length: 341
<
* Ignoring the response-body
* Connection #0 to host balvenie.example.com left intact
* Issue another request to this URL: 'https://balvenie.example.com/certsrv/'
* Re-using existing connection! (#0) with host balvenie.example.com
* Connected to balvenie.example.com (172.16.0.30) port 443 (#0)
* Server auth using NTLM with user 'fred'
> GET /certsrv/ HTTP/1.1
> Authorization: NTLM
TlRMTVNTUAADAAAAGAAYAEAAAAAYABgAWAAAAAAAAABwAAAABAAEAHAAAAAEAAQAdAAAAAAAAAAAAAAABoKJApsu9
/K5P/BBAAAAAAAAAAAAAAAAAAAAAAAtXDgnfyko11KipcYoDhQNC6TkaTtXYe2ZyZWRjcHBt
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 OpenSSL/1.0.1e
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: balvenie.example.com
> Accept: */*
>
< HTTP/1.1 401 Unauthorized
< Content-Type: text/html
< Server: Microsoft-IIS/7.5
```

```
* gss_init_sec_context() failed: : Credentials cache file '/tmp/krb5cc_0' not foundWWW-
Authenticate: Negotiate
* Authentication problem. Ignoring this.
< WWW-Authenticate: NTLM
< X-Powered-By: ASP.NET
< Date: Tue, 08 Apr 2014 19:53:02 GMT
< Content-Length: 1293
<
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<title>401 - Unauthorized: Access is denied due to invalid credentials.</title>
<style type="text/css">
<!--
body{margin:0;font-size:.7em;font-family:Verdana, Arial, Helvetica, sans-
serif;background:#EEEEEE;}
fieldset{padding:0 15px 10px 15px;}
h1{font-size:2.4em;margin:0;color:#FFF;}
h2{font-size:1.7em;margin:0;color:#CC0000;}
h3{font-size:1.2em;margin:10px 0 0 0;color:#000000;}
#header{width:96%;margin:0 0 0 0;padding:6px 2% 6px 2%;font-family:"trebuchet MS",
Verdana, sans-serif;color:#FFF;
background-color:#555555;}
#content{margin:0 0 0 2%;position:relative;}
.content-container{background:#FFF;width:96%;margin-
top:8px;padding:10px;position:relative;}
-->
</style>
</head>
<body>
<div id="header"><h1>Server Error</h1></div>
<div id="content">
 <div class="content-container"><fieldset>
  <h2>401 - Unauthorized: Access is denied due to invalid credentials.</h2>
  <h3>You do not have permission to view this directory or page using the credentials
that you supplied.</h3>
 </fieldset></div>
</div>
</body>
</html>
* Connection #0 to host balvenie.example.com left intact
* Closing connection #0
* SSLv3, TLS alert, Client hello (1):
```

The messages that will appear in the Application Log will look like this: