zscaler™

WHITE PAPER

Software Developer Solution Guide

TLS and Certificate Integration

Version 1 | September 2024

Contents

Contents

The following acronyms and their associated definitions will be used throughout this document.

Acronym	Definition
AD	Active Directory
AD FS	Active Directory Federation Services
CA	Certificate Authority
CLI	Command Line Interface
DNS	Domain Name Service
FQDN	Fully Qualified Domain Name
GRE	Generic Routing Encapsulation
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaC	Infrastructure as Code
IdP	Identity Provider
IP	Internet Protocol
OS	Operating System
PKI	Public Key Infrastructure
SAML	Security Assertion Markup Language
SCIM	System for Cross-Domain Identity Management
SP	Service Provider
SSH	Secure Shell
SSL	Secure Socket Layer (superseded by TLS)
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VDI	Virtual Desktop Infrastructure
VM	Virtual Machine
WLAN	Wireless Local Area Network
ZCA	Zscaler Central Authority
ZDX	Zscaler Digital Experience
ZIA	Zscaler Internet Access
ZPA	Zscaler Private Access
ZTE	Zero Trust Exchange

Introduction

This Solution Guide provides guidance that will help implement Zscaler Internet Access (ZIA) with TLS interception, reducing friction, lower security risk(s) and degradation to the Developer's user experience and workflow. This solution guide is intended for use by software development teams, software developer managers, and Zscaler security professionals that support the enterprise environment.

The document focuses on changes to a developer's environment allowing TLS interception leveraging Zscaler Internet Access. Subsequent editions of this solution guide will include common ZIA configurations and guidance for Zscaler Private Access (ZPA) to secure developer traffic from endpoint devices to server based targets.

This Solution Guide will discuss the following topics:

- How Zscaler Internet Access provides security to Software developer use cases
- Zscaler TLS Interception and Public Key Infrastructure (PKI) Certificates
- TLS interception and configurations to the developer's tools
- Zscaler configurations and troubleshooting guidance

Initial Configuration guidance when deploying Zscaler Internet Access (ZIA) TLS Interception can be found in this document: ZIA SSL Inspection Leading Practices Guide

Problem Statement for Software Developers

This Solution Guide focuses on two common problem cases:

1. Applications or software packages that do not use the default system certificate store or do not trust Zscaler's certificates. This causes unexpected errors or behaviors that otherwise would not appear.
2. Applications that perform certificate chain checks (Certificate Pinning or mTLS) not allowing Zscaler TLS interception to inspect the traffic.

Solution Architecture

The following architecture design principles must be maintained for the developer cohort to be productive when using Zscaler. It is assumed the Developer's local computer will have the Zscaler Client Connector installed.

- Internet access is required for the Developer's local computer
- Internet access is required for the server/application that builds or hosts the application created by the developer.
- The developer is required to make connections to the servers to transfer data, upload code and review, test, administer, and troubleshoot the server.
- The developer is expected to inspect and troubleshoot the inter-server communications.

In this document we will be exploring application connections through ZIA.

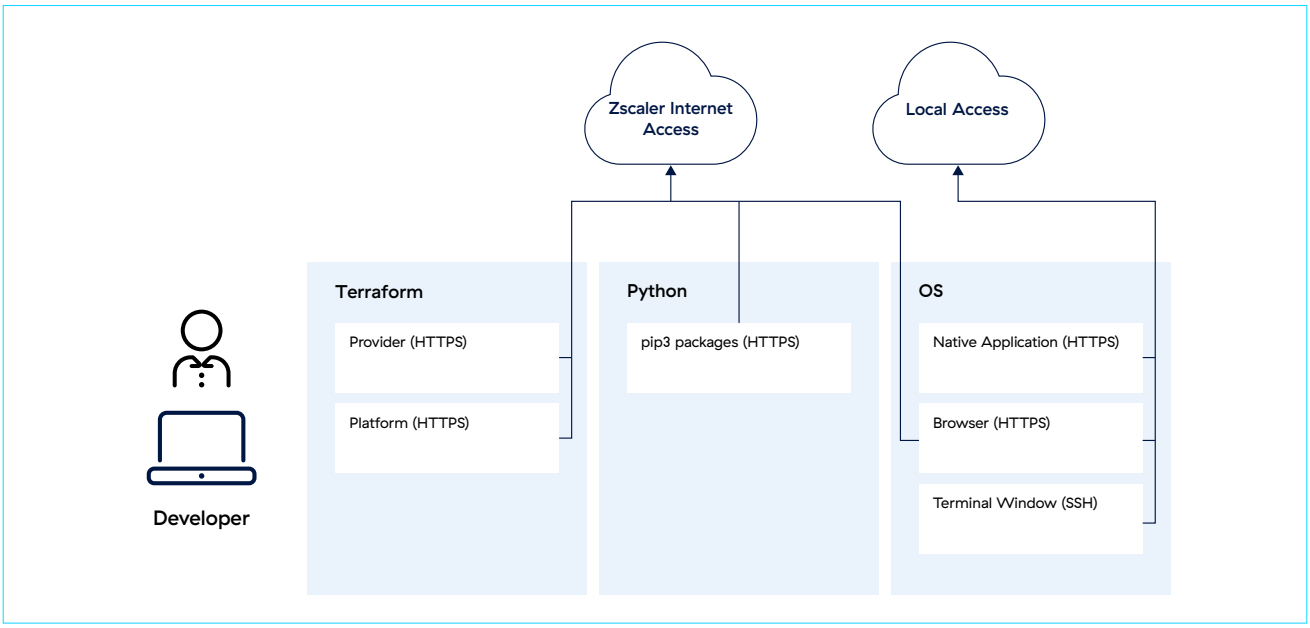


Figure 1: High level Developer Workflow

TLS Inspection Architecture

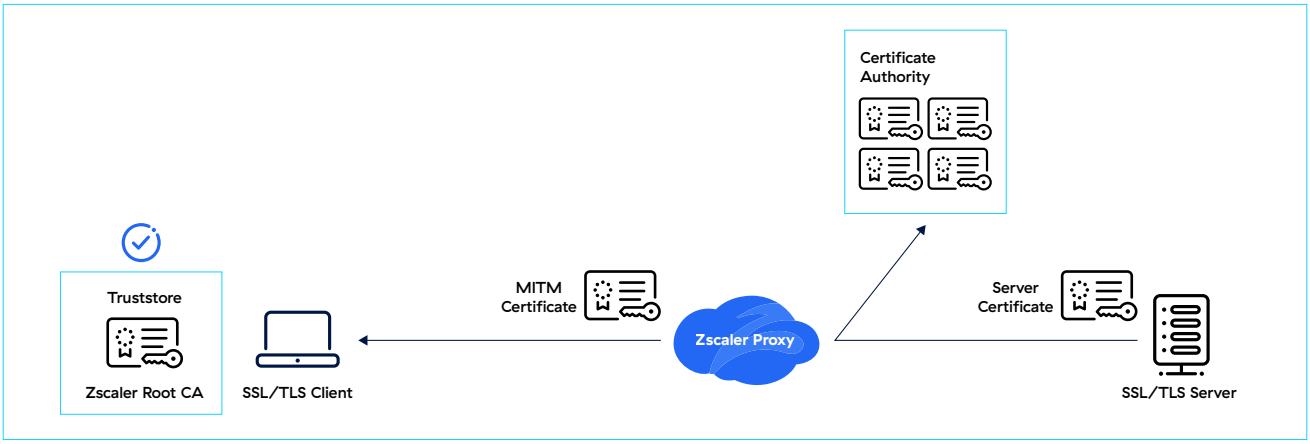
Details about TLS are outside the scope of this document: in general terms when establishing a secure TLS session from a client device to a server both will exchange secure messages, called a handshake. During this exchange, the server sends a certificate to the client endpoint device to prove its identity. The client endpoint verifies this certificate against a pre-installed list of certificate authorities, included within the web browser or operating system's trust store, and that the current date is a value between the "issued" and "expired" dates of the certificate. Before the network layer determines if the TLS handshake was successful several other steps usually are done.

Certificate validity is checked with either the Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP). In both cases, the client can contact the certificate authority (or its delegate) and verify if a given certificate has been revoked by the certificate authority. The result is success or failure for a given certificate. More information on PKI environments can be found here. If a certificate fails the verification, the TLS handshake is completed but no further data is sent and a warning message is shown in the Web browser or other client application.

If an application is using "certificate pinning" it will perform an additional check to validate that a specific hardcoded certificate or its public key is present in the TLS connection. This ensures that the app only accepts that pre-approved certificate, preventing Zscaler Internet Access from working for this application.

Zscaler intercepts all requests made by the client and uses this to make SSL interception decisions. When a session is selected for interception Zscaler Internet Access (ZIA) operates as a MiTM (Man in The Middle) proxy which intercepts the encrypted traffic, decrypts the traffic, inspects and applies controls, then re-encrypts the session before sending it to its destination. For ZIA to decrypt encrypted traffic, the Zscaler service dynamically generates, signs and issues a certificate on behalf of the server and thus

requires the client endpoint to trust this certificate issued for Zscaler as root Certificate Authority (CA). This is achieved by installing a root CA certificate as a trusted root CA in the Developer's local machine certificate trust store. Operating systems use a system-wide trust store to keep the root CA certificates. Linux is managed by OpenSSL, MacOS utilizes Keychain and in Windows, the system trust store is managed by CertMgr. In the Software Configuration Reference and Troubleshooting Connection Problems sections, there are some examples for Windows and Mac on tools that are commonly used by developers.



Solution

Please use the Order of policy enforcement documentation to understand how ZIA evaluates a given request, if it has been determined that there is a problem at the TLS layer then proceed to determine if there is a missing certificate trust configuration or a TLS network error. A quick test is to turn off ZIA, once off, if the application works as intended then proceed with this section. Troubleshooting can be found in Troubleshooting Connection Problems.

The most important steps happen between the developer and ZIA administrator in the form of a report and a confirmation that it is resolved or further analysis is required. This feedback loop starts with the developer reporting that a specific domain needs to be bypassed. The loop is either finished when the ZIA manager gets confirmation that the application is working or looping if further investigation is needed. We provide examples of how this communication can be done using the configurations and tools provided within Zscaler.

The 3 common solutions for developers when they encounter an application error:

- Specific ZIA SSL Inspection Policy to bypass the domain. This is applied to the Developer's user or group only.
- Install SSL Interception intermediate certificates into developer application/OS certificate store.
- Install or configure the application's specific packages to make use of system trust store or certificate bundle.

When there is a reported problem with a single application the most common scenario is Public Key Pinning (PKP) or Certificate Pinning, where an application will verify the server certificate in order to make the connection. When this happens and ZIA is enabled in the environment the application will not work and a URL bypass is needed.

ZIA SSL Interception Configuration

This section explores how to enable developers when using SSL inspection. When applying SSL inspection bypass policies keep in mind the following principles:

- A single policy can be applied to up to 4 users. To create scale, utilizing groups allows an administrator to apply policies to many users.
- Understanding the needs of each developer group allows policies to be applied to more than one developer team in a single group.
- Each group will have at least one URL category for both SSL Interception and URL Filtering.
- Consider applying a URL category for all groups and a unique URL category for each group
- Lastly keep in mind that a bypass for domain github.com would bypass all repositories, but if used as a URL filter i.e. github.com/github-copilot, it can be used to block as long as the github domain is being inspected

Please refer to the following table for SSL Inspection policies:

Order	Name	URL Category	Criteria	Action	Description
1	SSL Pinning Bypass for Apple	ByPass-Apple	Group: Developers	Do Not Inspect, Evaluate Other Policies	Example to enable Mac OS X Apple Store to work
2	Smart Isolation One Click Rule	Suspicious Domains		Inspect	Smart Isolate Single Click Rule automatically created upon enabling Smart Isolation from Browser Control.
3	Github_Exemptions	Developer GitHub Requests	Group: Developers	Inspect, Evaluate Other Policies	Github Specific URL filtering actions
4	Private CA	Private CA Sites	Group: Developers	Inspect Allow Untrusted Server Certificate	
5	Zscaler Recommended Exemptions	Recommended SSL Exemptions		Do Not Inspect Bypass Other Policies	
6	Office 365 One Click			Do Not Inspect, Bypass Other Policies	Disabled due to rule name in logging
7	Existing rules				
Default Rule	Default SSL Inspection Rule			Do Not Inspect, Evaluate Other Policies	Default

When creating URL Categories please refer to URL Format Guidelines. Keep in mind that TLS inspection can only operate on the information available in the SNI header (such as hostname or domain) but NOT the full URL. Without TLS inspection visibility into the full URL of an HTTPS session is not possible.

The same URL category can be used for SSL Inspection and for URL Filtering, the first feature uses domain names while the second feature uses URLs, and as stated previously URL filtering only applies to SSL intercepted traffic.

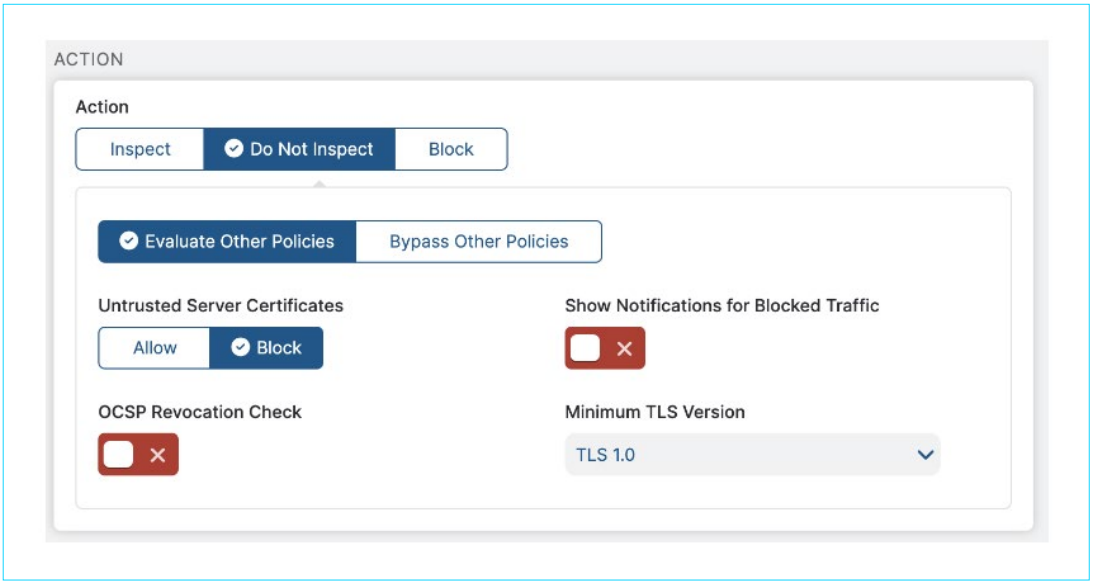
The following URL Categories are all User-Defined and are provided as guidance when enabling developers.

Order	Name	URL Category
Bypass-Apple	amp-api-edge.apps.apple.com amp-api.apps.apple.com api.apple-cloudkit.com app-site-association.cdn-apple.com appldnld.apple.com apps.mzstatic.com bag-cdn-lb.itunes-apple.com.akadns.net bag.itunes.apple.com configuration.apple.com dit.whatsapp.net downloaddispatch.itunes.apple.com entitlements.itunes.apple.com gateway.icloud.com gdmf.apple.com gg.apple.com gs-loc.apple.com gs.apple.com gsa.apple.com h3.media.apple.map.fastly.net humb.apple.com identity.ess.apple.com ig.apple.com init.itunes.apple.com itunes.apple.com lcdn-locator.apple.com lcdn-registration.apple.com mesu.apple.com metrics.icloud.com mmg.whatsapp.net ns.itunes.apple.com oscdn.apple.com p63-acsegateway.icloud.com p63-fmip.icloud.com playgrounds-assets-cdn.apple.com	Apple specific URLs that have public key pinning (PKP) and will be bypassed

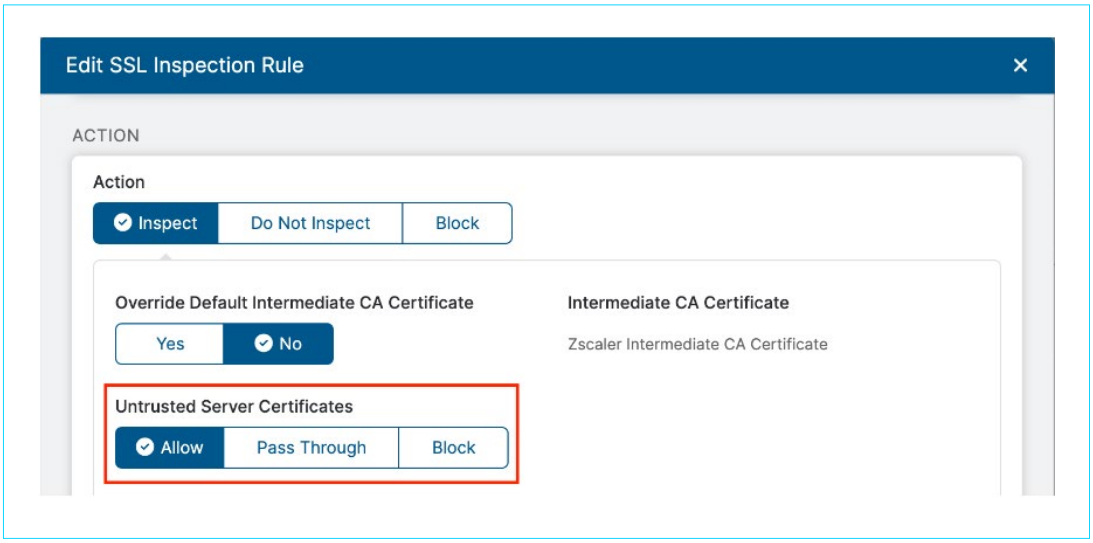
Order	Name	URL Category
Bypass–Apple	ppq.apple.com profile.ess.apple.com securemetrics.apple.com serverstatus.apple.com setup.icloud.com skl.apple.com suconfig.apple.com swcdn.apple.com swdist.apple.com swscan.apple.com token.safebrowsing.apple updates–http.cdn–apple.com updates.cdn–apple.com xp–cdn.apple.com xp.apple.com	
Developer GitHub Requests	github.com/microsoft/vcpkg.git github.com/composer codeload.github.com api.github.com/repos/Behat	URL requested by developers, assign to only developers and apply only to URL filtering
Github Auto Pilot URLs	github.com/github–copilot github.com/features/copilot/	Blocking copilot for everyone via URL filtering
Private CA Sites	testing.httpbin.org	Sites with invalid or private certificates

Each group of users would have two SSL Interception policies, both are defined by URL Categories that are part of the feedback loop that is discussed in this document.

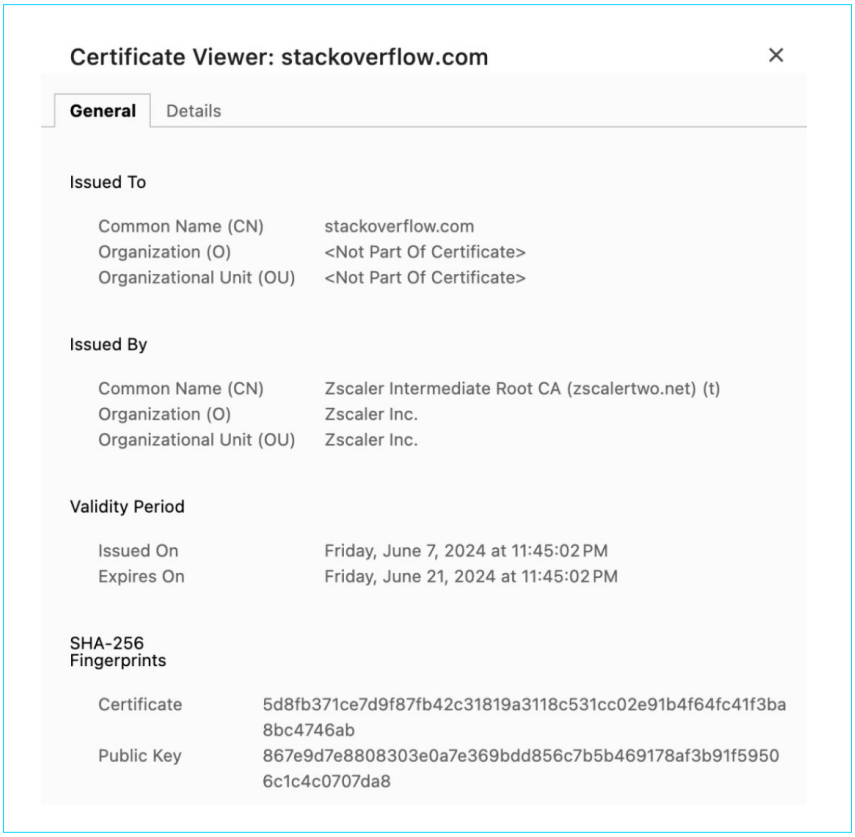
1. On a specific group this policy will allow bypassing TLS interception to certain specific domains and the second is to allow connections to sites with certificate errors. This is controlled within the SSL Interception Action:



2. On the same specific group the policy will connect to a site that has a certificate from a private CA, invalid certificate or an expired certificate. This is controlled within the SSL Interception Action:



When connecting to sites protected by the Zero Trust Exchange the following image shows how a site displays the provided Zscaler certificate:



In Appendix: PKI and Certificate Authorities we explain how a custom intermediate certificate can be requested and installed in ZIA.

Zscaler Client Connector Tunnel Version

Zscaler on the Client Connector offers two versions of the Z-Tunnel. Traffic is forwarded through the tunnel so that the ZIA Public Service Edge can apply the appropriate security and access policies. Discuss with your administrator which ZTunnel strikes the correct balance between connectivity and security. The following shows some of the advantages of each ZTunnel type:

Z-Tunnel 1.0
Z-Tunnel 1.0 forwards traffic to the Zscaler cloud via CONNECT requests, much like a traditional proxy. Version 1.0 sends all proxy-aware traffic or port 80/443 traffic to the Zscaler service.

For the developer use case all internet websites commonly use port 443 and any custom deployed software on the local machine would make use of higher (+1024) port numbers. This allows the developer to browse securely via ZIA but still deploy software locally and make requests to other machines in the local network.

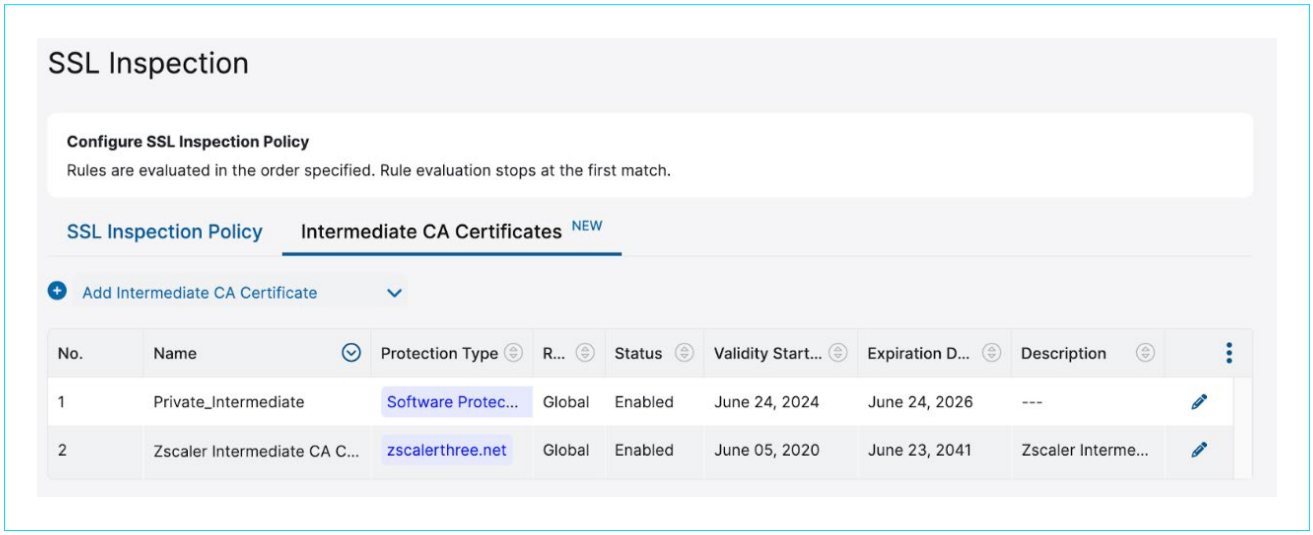
Z-Tunnel 2.0
Z-Tunnel 2.0 has a tunneling architecture that uses DTLS or TLS to send packets to the Zscaler service. Z-Tunnel 2.0 is capable of forwarding to Zscaler all ports and protocols where the forwarding policy will determine if a given request is for the local network or for a website on the internet via Zscaler.

Zscaler Client Connector Automated Certificate Installation

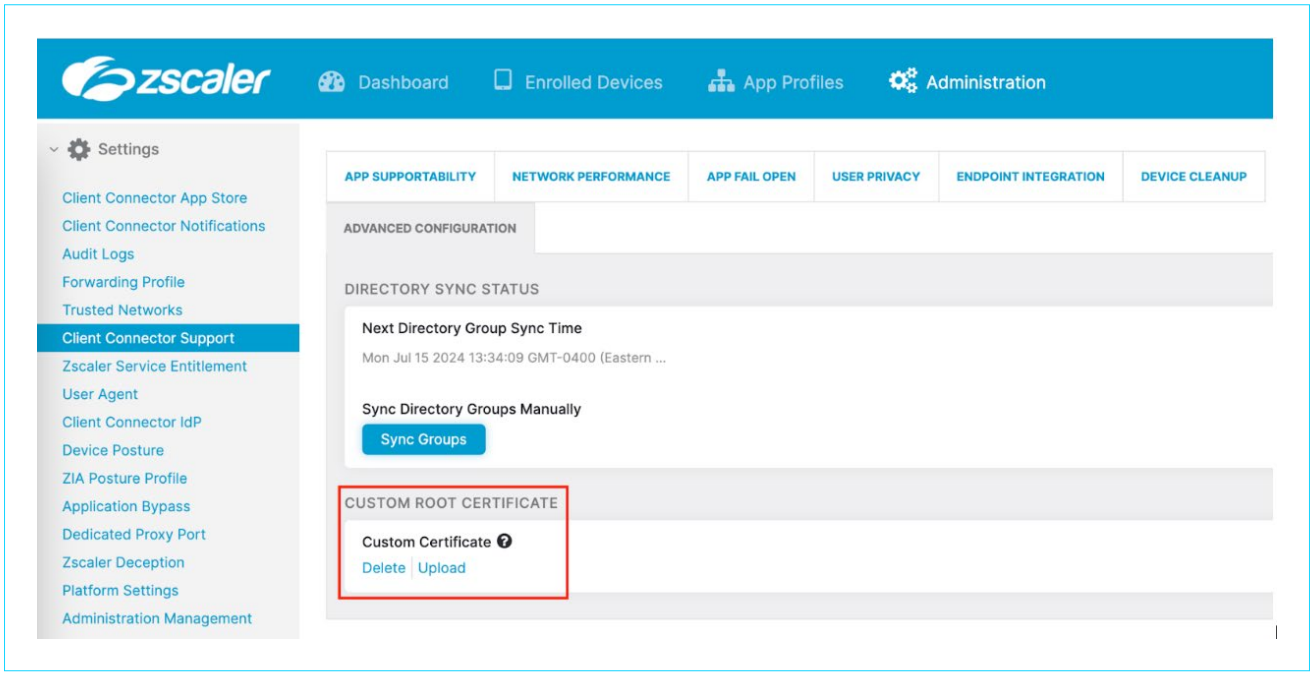
This section will showcase how some configurations will affect the developer local machine's Zscaler Client Connector. By default the Zscaler Client Connector will not install the intermediate certificate for SSL Interception, so follow the following steps to accomplish this.

When the developer uses either Zscaler provided certificates or their own custom Certificate Authority Root certificates, this allows the installation of the certificate to the system certificate store in Windows and Mac.

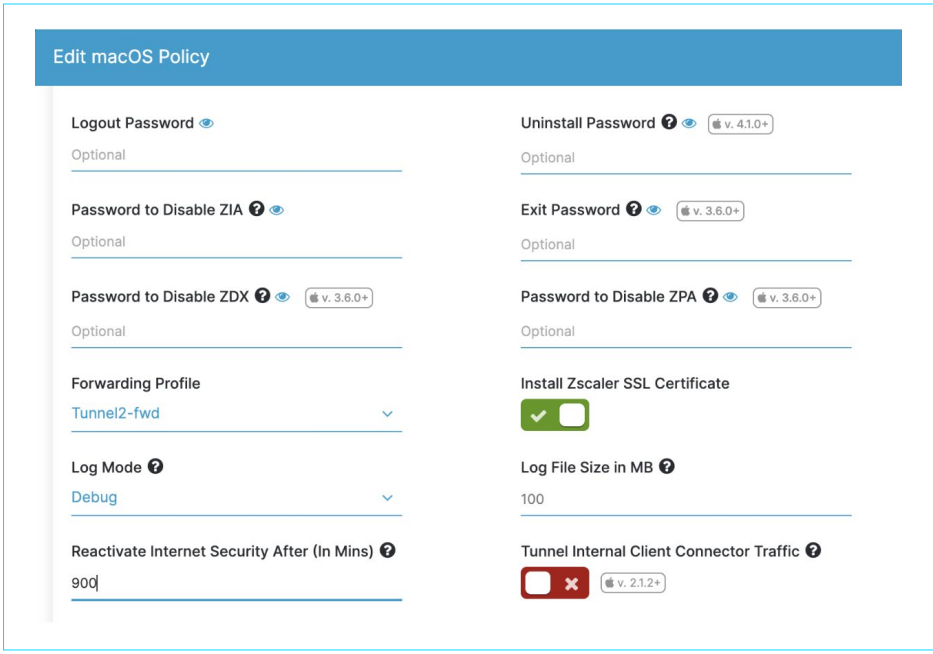
From the Zscaler Client Connector the App Profile controls if the Intermediate certificate is installed, by default no certificate is installed in the client machine. Download the intermediate certificate from ZIA:



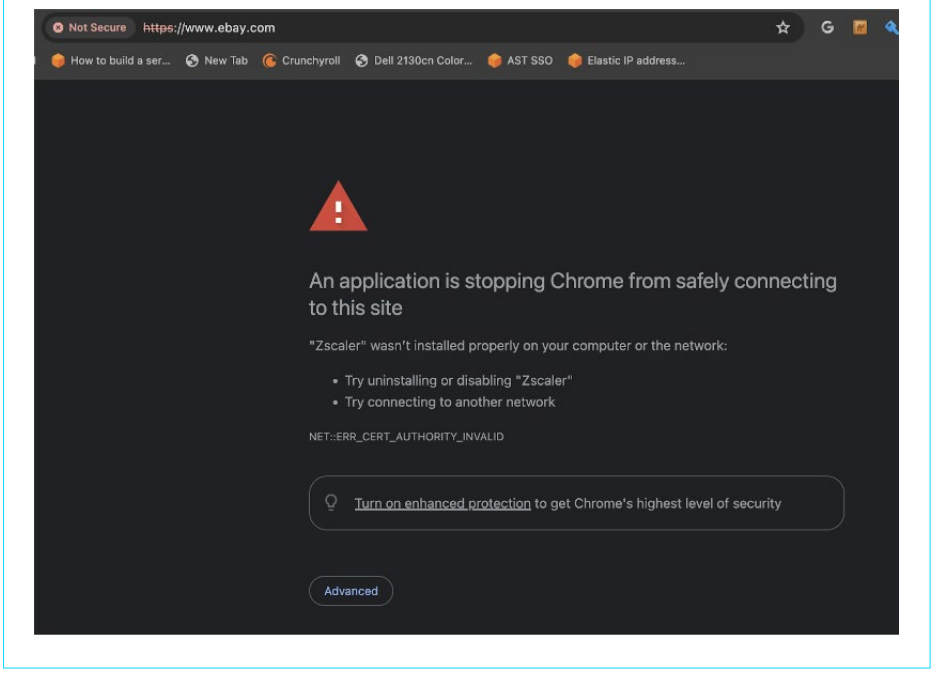
Regardless of which certificate is used for SSL Interception, the uploaded certificate in this section will be uploaded to the Client Portal App Profile:



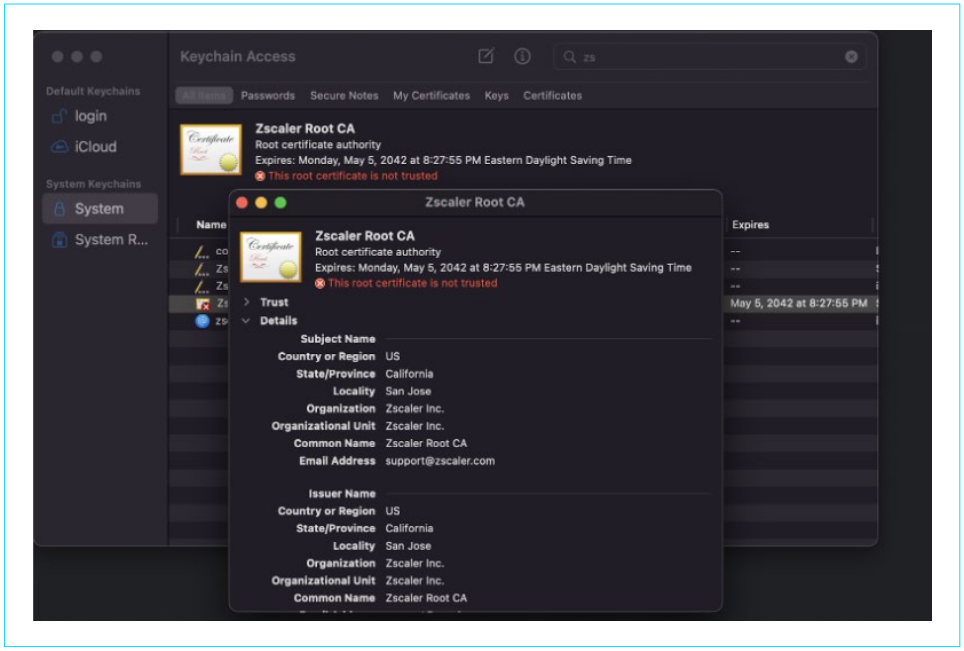
Proceed to the App Profile to enable the certificate installation from the Client Connector automatically, an App Update and possible browser restarts will be necessary. This is configured in App Profiles, Select Windows or Mac, and add a new policy. You will find an option to enable “Install Zscaler SSL Certificate”, this will install the certificate mentioned above in the developer machine.



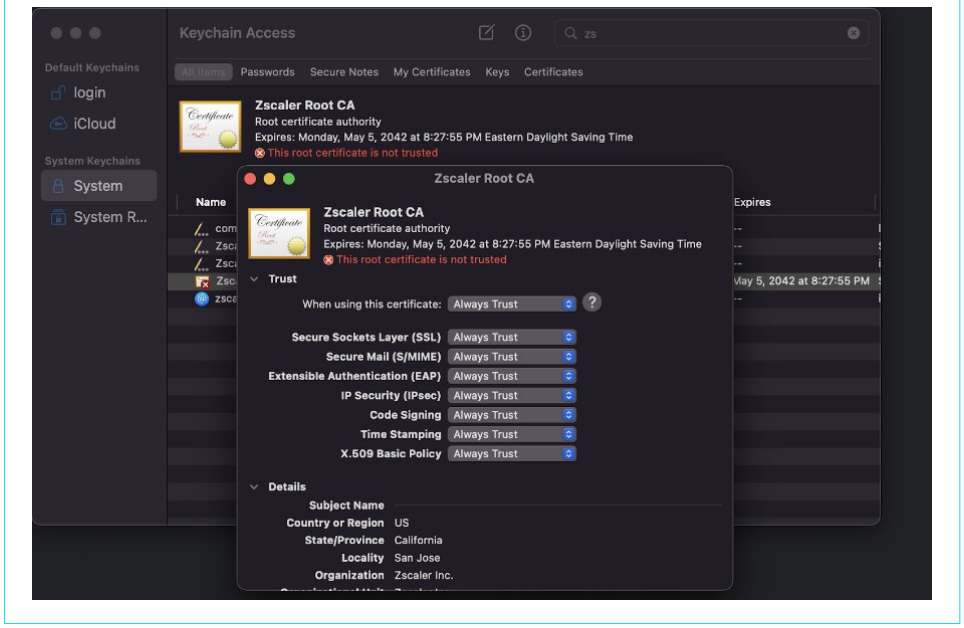
In Mac OS X (if there is not MDM solution in place) it is necessary also to trust the installed certificate, otherwise the following error will be seen:



Follow this step to Trust the installed certificate, in KeyChain Access:



Open Trust within the certificate and change it to Always trust:



In Microsoft Windows there is no need to make any changes.

DNS Control

Zscaler service typically includes some type of DNS modification or inspection. A full discussion of this topic is outside the scope of this document. It is important to have a discussion between the network/security team that manages Zscaler and the developer teams to understand if this will impact common developer activities. As an example, Internet bound DNS requests are often inspected, so a newly registered domain may trigger an action that a developer may not expect (possibly dropped or modified). In another example internal zones may be handled differently to enable Zero Trust Network Access (ZTNA) through the Zscaler service (internal IP may be modified in the DNS response). Have the conversation up front to be certain that the developer community understands the architecture and the implications.

Browser Isolation

This feature allows the developer to browse URL categories that are considered risky, yet there is valuable content that can be found on those sites.

A developer needs to be aware when a site is being rendered by Browser Isolation. Here is a an example for how a HTTP request looks like without browser isolation:

```
% curl https://openai.com/chatgpt/
<!DOCTYPE html><html lang="en-US"><head><title>Just a moment...</title><meta http-
equiv="Content-Type" content="text/html; charset=UTF-8"><meta http-equiv="X-UA-Compatible"
content="IE=Edge"><meta name="robots" content="noindex,nofollow"><meta name="viewport"
content="width=device-width,initial-scale=1">
```

And also for reference here is an example of the same web site with browser isolation:

```
% curl https://openai.com/chatgpt/
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3c.org/
TR/1999/REC-html401-19991224/loose.dtd">
<html>
<head>
<meta name="description" content="Zscaler makes the internet safe for businesses by
protecting their employees from malware, viruses, and other security threats.">
```

Zscaler Browser Isolation renders the risky web page on an air gapped computer that isolates it from the developer computer, yet it does allow certain configurable interaction options that are relevant for a developer use case:

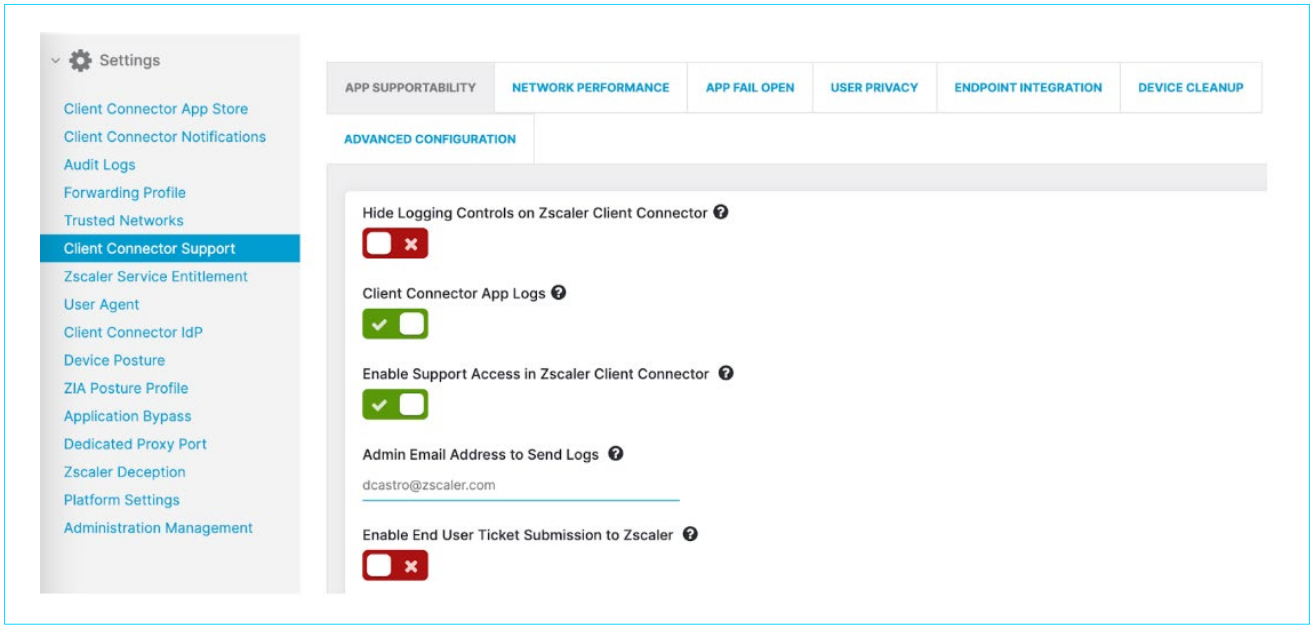
- Copy / Paste
  This feature can be turned on/off and will allow the developer to copy and paste text between the browser isolation and his local machine

- File Transfer
  This configuration allows the developer to upload files or download files (or both) to the isolated browser session running within Zscaler.
- URL Obfuscation
  The URL that was browsed to can be obfuscated from the developer, or can be shown plain text, this matters as this URL might be shared between developers.

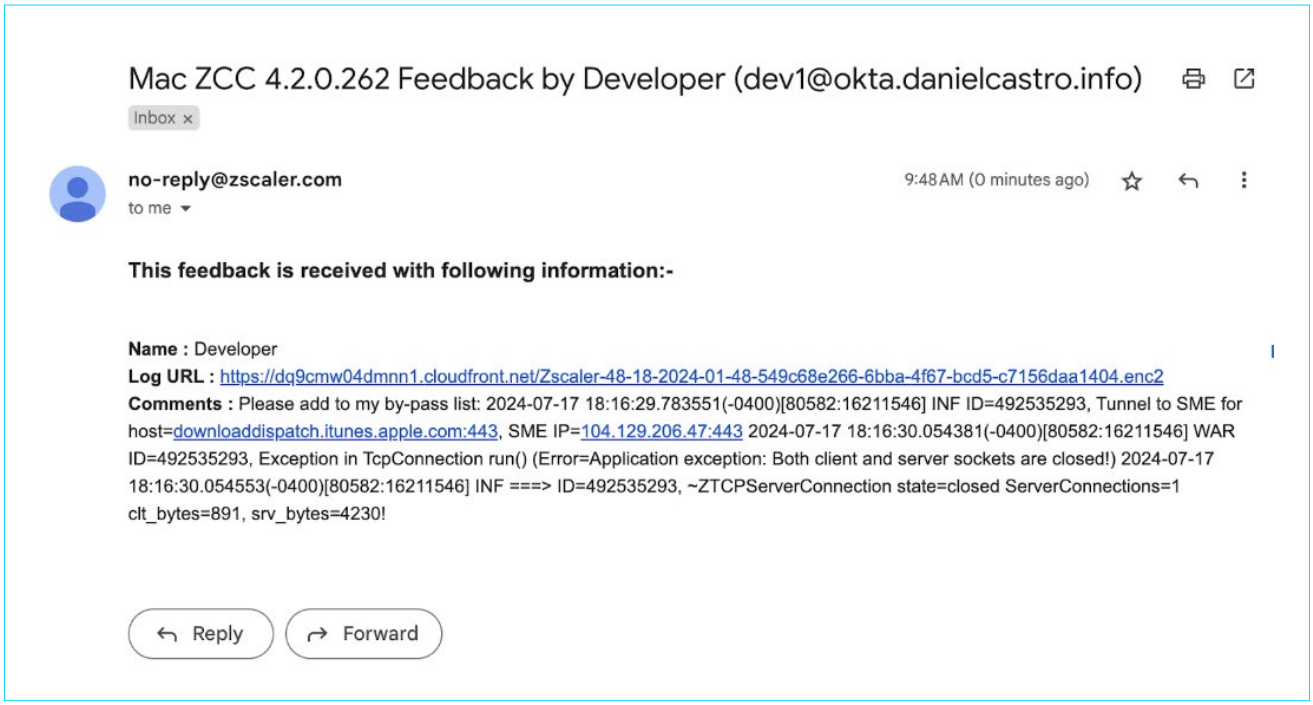
Enable feedback from User to Administrator

The Zscaler Client Connector has the ability to send an email with the relevant data that will allow the ZIA Administrator to work with the developer to create bypass rules in a short time frame.

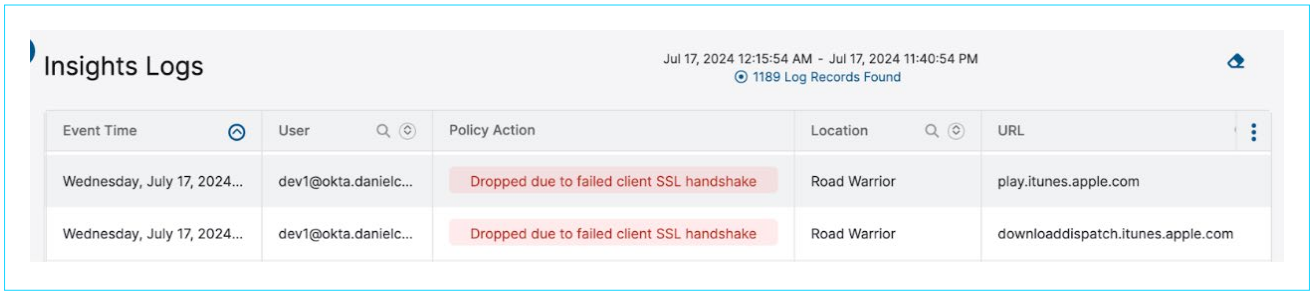
First the Administrator for Zscaler Client Connector needs to turn on “Enable Support Access in Zscaler Client Connector” and specify an email address that will receive the communication



Once a user provides feedback via the feature it will be received in the configured email address:



But more importantly it contains, when, who and what URL failed, this can be corroborated in the Web Insights Logs.



When looking for a particular entry in Web Insight logs it is easier to download the CSV file and find relevant entries using grep or a similar search application when dealing with very large files.

The administrator can now add the URL to the user/group bypass list and allow the app to run normally.

Software Configuration Reference

Some applications use their own application trust store instead of using the default system store. When this is the case, the application is not able to validate the TLS interception certificate. A "ca-bundle" file commonly refers to a file that contains a list of root certificates and normally replaces the system certificate store for a set of applications. We provide example steps to build a "ca-bundle" file with either the Zscaler root certificate or custom root certificate also containing Mozilla common system root CAs.

In each subsection you will find the required steps to use the generated ca-bundle to the application specific trust store. The general guidance is to use environment variables as this is the most broad approach to enabling certificate bundles. In some cases, individual configurations are employed for the specific tool or libraries.

Zscaler online documentation has the initial list of applications: Adding Custom Certificates to Applications. In the following subsections you will find applications that are not on that list or more details are provided on a particular application.

In order to build the ca-bundle you can leverage Mozilla Firefox CA Bundle (cacert.pem can be downloaded here) and add the Zscaler root certificate in PEM format at the end of the file. The ca-bundle filename extension must match what format is required by each application, this is commonly "pem" or "crt". If your administrator follows this guide your root Zscaler certificate will be installed already in your machine's certificate store and can be exported (for example, as ZscalerRootCertificate-2048-SHA256.crt).

- For Mac you would use Keychain Access, right click on the certificate in question (Zscaler or custom), & choose export from the menu.
- For Windows you would open the Certificate Manager MMC snap-in, select System Certificates and Right-click the certificate, select All Tasks, and then select Export. Finish the Export Wizard and use Base 64 Encoded Format, if this option is grayed out then select PKCS#12 DER format but this will need to be converted to PEM format.

Merge the two files using a text editor or with the following command in Mac or Linux:

```
cat cacert.pem ZscalerRootCertificate-2048-SHA256.crt > ca-bundle.pem
```

3.1 AWS-CLI v1 and v2

This section describes the SSL configuration of the two AWS CLI versions.

The Zscaler certificate error will be seen similar to this:

```
SSL validation failed for https://sts.amazonaws.com/ [SSL: CERTIFICATE_VERIFY_FAILED]
certificate verify failed: unable to get local issuer certificate (_ssl.c:1145)
```

```
SSL validation failed for https://ec2.us-west-2.amazonaws.com/ [SSL: CERTIFICATE_VERIFY_
FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1145)
```

AWS – CLI v1 and v2		
Tested Version	**Documentation**	**Notes**
aws-cli/2.15.33 Python/3.11.8	Version 1: https://docs.aws.amazon.com/cli/v1/userguide/cli-configure-envvars.html Version 2: https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-options.html	Apple specific URLs that have public key pinning (PKP) and will be bypassed
Configuration Details		
Command Line Argument		
`--ca-bundle <string>`		
Environment Variable		
`export AWS_CA_BUNDLE=<Path to Certificate>/ca-bundle.pem`		
Configuration File		

`% aws configure`
`When the above is run the following changes are done in the configuration file`
`ca_bundle="file.pem"`
`Specifies a CA certificate bundle (a file with the .pem extension) that is used to verify SSL certificates. Can be overridden by the AWS_CA_BUNDLE environment variable or the --ca-bundle command line option.`		

3.2 Curl

This section describes the SSL configuration for the curl application.

Curl		
Tested Version	**Documentation**	**Notes**
8.4.0 with libreSSL 3.3.6	https://curl.se/docs/sslcerts.html	On Mac OS X makes use of the system certificate store On Debian Linux distribution curl uses ca-certificates, see more information on 3.14 APT
Configuration Details		
Command Line Argument		
`--with-ca-file=<folder> or --with-ca-bundle=<file> OR --cacert <file> or --capath <folder>`		
Environment Variable		
`[System.Environment]::SetEnvironmentVariable("CURL_CA_BUNDLE", "<Path to Certificate>\ca-bundle.pem", "Machine")`		

3.3 Docker

This section describes the SSL configuration of various Docker components.

Docker		
Docker Image	**Docker Daemon**	**Notes**
Tested software version: Docker 26.1 Alpine latest (3.20.1) Ubuntu latest (24.04), Windows Server Core (2022) Official Documentation Link: https://docs.docker.com/build/building/best-practices https://docs.docker.com/reference/dockerfile/#copy https://docs.docker.com/reference/dockerfile/#run	Tested software version: 26.1 Official Documentation Link: https://docs.docker.com/engine/security/certificates	Docker daemon also accesses the certificate store of the operating system by default, so that a separate configuration of Docker is not required.

Configuration Details

Ubuntu Linux

```
FROM ubuntu:latest
# Don't ask any command line questions
ENV DEBIAN_FRONTEND=noninteractive
# Copy the certificate to the docker image file system in a trusted location. On Ubuntu the file needs to have the extension .crt otherwise it will no be accepted
COPY ca-bundle.crt /usr/local/share/ca-certificates/
# Create the ssl directory because it does not exists at this point
RUN mkdir -p /etc/ssl/certs
# Append the ca certificate to the main certificate file manually because otherwise it is probably not possible to reach the internet
RUN cat '/usr/local/share/ca-certificates/ca-bundle.crt' >> /etc/ssl/certs/ca-certificates.crt
# Update package list
RUN apt-get update
# Install certificate management
RUN apt-get install -y ca-certificates
# Update the certificates
RUN update-ca-certificates
# More commands…
```

Alpine Linux

```
FROM alpine:latest
# Copy the certificate to the docker image file system in a trusted location
COPY ‘ca-bundle.pem’ /usr/local/share/ca-certificates/
# Create the directory because it does not exists at this point
RUN mkdir -p /etc/ssl/certs
# Append the ca certificate to the main certificate file manually because otherwise it can be impossible to reach the internet
RUN cat ‘/usr/local/share/ca-certificates/ca-bundle.pem’ >> /etc/ssl/certs/ca-certificates.crt
# Install ca-certificates package
RUN apk add --no-cache ca-certificates
# Update CA certificates
RUN update-ca-certificates
# More commands…
```

Windows Server Core 2022

```
FROM mcr.microsoft.com/windows/servercore:ltsc2022
# Copy the certificate to the docker image file system
COPY ca-bundle.pem ./
# Import the certificate into the trust store of the computer
RUN powershell -Command Import-Certificate -FilePath ca-bundle.pem -CertStoreLocation Cert:\LocalMachine\Root\
# More commands…
```

3.4 Git

This section describes the SSL configuration for the git application.

Git		
Tested Version	**Documentation**	**Notes**
2.38.1	https://git-scm.com/docs/git-config	Git can be configured per individual repository or for all repositories. On Mac OS X git uses the system certificate store

Configuration Details

Command Line Argument

```
# one repository
git config --global http.”https://repos.sample.com”.sslCAInfo /path_to_file/cert.pem <string>
# all repositories
git config --global http.sslCAInfo /path_to_file/ca-bundle.pem
```

Windows System Store

```
git config --system http.sslbackend schannel
```

3.5 NPM

This section describes the SSL configuration of various node package manager application.

NPM		
Tested Version	**Documentation**	**Notes**
10.2.3	https://nodejs.org/api/cli.html#cli_node_extra_ca_certs_file	curl on Mac OS X makes use of the system certificate store
Configuration Details		
Command Line Argument		
`--with-ca-file=<folder> or --with-ca-bundle=<file>`		
Environment Variable Windows		
`[System.Environment]::SetEnvironmentVariable("NODE_EXTRA_CA_CERTS", "<Path to Certificate>\ca-bundle.pem", "Machine")`		
Environment Variable Mac		
`launchctl setenv NODE_EXTRA_CA_CERTS <Path to Certificate>/ca-bundle.pem`		
Environment Variable Linux		
`echo "export NODE_EXTRA_CA_CERTS"=<Path to Certificate>/ca-bundle.pem" >> $HOME/.bashrc`		

3.6 Oracle Java

This section describes the SSL configuration of various Java components.

Java		
Tested Version	**Documentation**	**Notes**
Java(TM) SE Runtime Environment (build 1.8.0_411-b09)	https://docs.oracle.com/en/java/javase/12/tools/keytool.html	Oracle Java uses its own certificate store, which is normally stored in the cacerts file in the Java Runtime installation directory. To manage the certificate store, Java provides the command line tool keytool, which is also located in the Java installation directory. The default password for the Java certificate store is changeit and is required for changes to the certificate store.
Configuration Details		
Command Line Argument		
`keytool -importcert -file "<Path to Certificate>/ca-bundle.pem" -alias mycert -keystore "<Java Runtime Install Path>/lib/security/cacerts" -storepass changeit`		

Note: The ca-bundle.pem file must have system trusted certificates stored along with Zscaler CA certificate or customer custom CA.

3.7 Python

This section describes the SSL configuration of various Python components and libraries. Use the following table to modify python in your local system.

Python 3.4.1 or later		
Tested Version	**Documentation**	**Notes**
3.14.OaO	https://www.python-httpx.org/environment_variables/#ssl_cert_file	Python 3.4.1 and earlier do not do certificate validation by default. If you are using virtual environments then use export on the terminal instead of launchtcl
Configuration Details		
Environment Variable Windows		
`[System.Environment]::SetEnvironmentVariable("SSL_CERT_FILE", "<Path to Certificate>\ca-bundle.pem", "Machine")`		
Environment Variable Linux		
`echo "export SSL_CERT_FILE=<Path to Certificate>/ca-bundle.pem" >> $HOME/.bashrc`		
Environment Variable Mac		
`launchctl setenv SSL_CERT_FILE <Path to Certificate>/ca-bundle.pem`		

If the developer would rather include the bundle in the python code follow the following guidance:

To verify if indeed the environment is being routed through Zscaler ZIA, the following Python code below can be used to verify the peer certificate:

```
from socket import socket
import ssl
import M2Crypto
import OpenSSL
# M2Crypto

cert = ssl.get_server_certificate(('www.google.com', 443))
x509 = M2Crypto.X509.load_cert_string(cert)
print x509.get_subject().as_text()
x509 = OpenSSL.crypto.load_certificate(OpenSSL.crypto.FILETYPE_PEM, cert)
print x509.get_subject().get_components()
```

The following code will return the ssl_cert_dir CA certificate store path:

```
python -c "import ssl; print(ssl.get_default_verify_paths())"
```

If behind ZIA you will get the following error:

```
File "~/.pyenv/versions/3.14-dev/lib/python3.14/urllib/request.py", line 1322, in do_open
raise URLError(err)
urllib.error.URLError: <urlopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify
failed: Missing Authority Key Identifier (_ssl.c:1020)>
```

In order to manually specify the ca bundle within the python code there are multiple ways to do it but depending on the package needed this will vary. In the following example no additional packages are installed:

```
import certifi
import ssl
from urllib.request import urlopen
ctx = ssl.create_default_context(ssl.Purpose.SERVER_AUTH) ctx.load_verify_
locations(cafile=certifi.where()) # modify to your location
response = urlopen("https://httpbin.org", context=ctx)
```

When distributing the software include the ca-bundle with the python files and point cafile to a relative location in relation to the python file.

3.8 Python PIP / Conda

This section describes the SSL configuration for PIP command.

Pip		
Tested Version	**Documentation**	**Notes**
pip 24.0 with python 3.14	https://pip.pypa.io/en/latest/topics/https-certificates/	By default, pip will not use the system certificate store but, instead, uses a bundled CA certificate store from certifi. Version 22.2+ use system trust store with --use-feature=truststore. This will replace the bundled packaged certificates for verifying HTTPS certificates. Requires Python 3.10+.
Configuration Details		
Command Line Argument		
`python -m pip install SomePackage --use-feature=truststore`		

3.9 Python urlib3 library

This section describes the SSL configuration for the urllib3 python library and packages.

Tested software version: 2.2.1

Official Documentation Link: https://urllib3.readthedocs.io/en/2.2.1/user-guide.html#certificate-verification

If certificate validation is needed, starting on version 1.25 you can use the PoolManager configuration option `ca_certs`

```
import certifi
import urllib3
http = urllib3.PoolManager(
cert_reqs='CERT_REQUIRED',
ca_certs=certifi.where()   # Specify your location
)
```

3.10 Python requests library

This section describes the SSL configuration for the request python library. Requests verifies SSL certificates for HTTPS requests, just like a web browser. By default, SSL verification is enabled, and Requests will throw a SSLError if it's unable to verify the certificate.

There are two main methods to enable a certificate bundle for requests, when making the request using the verify option. Or via environment variables.

```
import requests
cafile = 'cacert.pem' # adjust to your location
r = requests.get(url, verify=cafile)
```

Python Requests		
Tested Version	**Documentation**	**Notes**
Python 3.14 with Requests 2.32.3	https://requests.readthedocs.io/en/latest/	This list of trusted CAs can also be specified through the REQUESTS_CA_BUNDLE environment variable. If REQUESTS_CA_BUNDLE is not set, CURL_CA_BUNDLE will be used as fallback.
Configuration Details		
Environment Variable Windows		
`[System.Environment]::SetEnvironmentVariable("REQUESTS_CA_BUNDLE", "<Path to Certificate>\ca-bundle.pem", "Machine")`		
Environment Variable Linux		
`echo "export REQUESTS_CA_BUNDLE=<Path to Certificate>/ca-bundle.pem" >> $HOME/.bashrc`		
Environment Variable Mac		
`echo "export REQUESTS_CA_BUNDLE=<Path to Certificate>/ca-bundle.pem"`		

3.11 Xcode Simulator

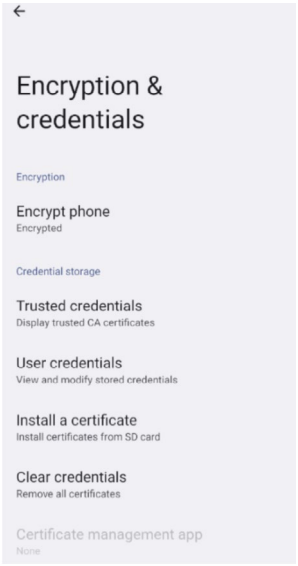
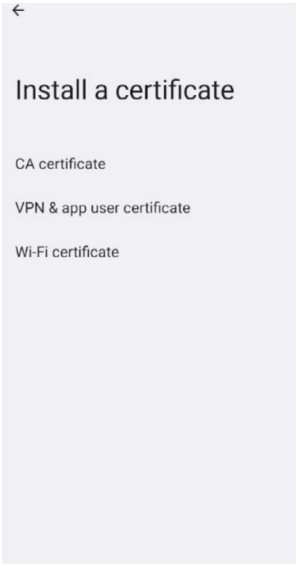
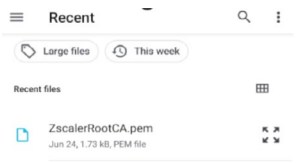
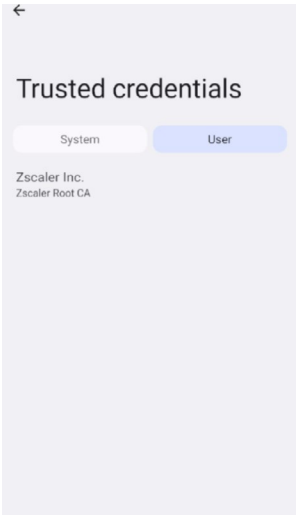
The following steps need to be taken in order to install Zscaler Root Certificate to Xcode simulator.

Xcode Simulator		
Tested Version	**Documentation**	**Notes**
xcrun version 68. Xcode 11.4	https://support.apple.com/en-us/HT204132 https://developer.apple.com/documentation/xcode/installing-additional-simulator-runtimes	Use this guide to troubleshoot connection problems: Identifying the Source of Blocked Connections
Configuration Details		
Environment Variable Windows		
`xcrun simctl keychain booted add-root-cert <Zscaler Root Certificate path in pem format>`		

3.12 Android Emulator

The following steps were tested in Koala (Build #AI-241.18034.62.2411.12071903, built on July 10, 2024), in order to install Zscaler Root Certificate to Android Emulator.

You can drag and drop the Zscaler Root Certificate onto the Android Emulator and it will be stored in the files.

Step 1	Step 2	Step 3	Step 4
Open Settings > Security > More Security Settings > Encryption & Credentials.	Click on Install a Certificate > CA Certificate to install Zscaler Root Certificate from files.	Select the certificate from files. And on the warning click on "Install Anyway".	A notification "CA certificate installed" at the bottom of the emulator screen and the Zscaler Root certificate should be installed. Verify the certificate is installed under Settings > Security > More Security Settings > Encryption & Credentials > Trusted Credentials > User.
			

3.13 APT

The command-line tool `apt-get` is the most popular package management tool used in the Debian-based Linux operating system. Starting on apt 1.5 HTTPS is natively supported, yet most sources are configured on HTTP by default.

The following procedure was tested on Ubuntu 20.04 LTS, official documentation on update-ca-certificates can be found here.

If you encounter an error like this:

```
Err:5 https://us-west-1.ec2.archive.ubuntu.com/ubuntu focal Release

  Certificate verification failed: The certificate is NOT trusted. The certificate
issuer is unknown.  Could not handshake: Error in the certificate verification. [IP:
54.241.183.81 443]
```

Depending on the distribution and version the package ca-certificates will enable a centralized certificate store for the Linux distribution located at /etc/ssl/certs. The package comes with CLI commands to interact with the store.

Make a folder to contain the Zscaler root certificate:

```
mkdir /usr/share/ca-certificates/zscaler
```

Then copy the certificate as an individual file within that folder and use the extension crt.

Then modify the ca-certificates configuration file to include the certificate. At the last line include:

```
zscaler/Zscaler_root.crt
```

Then issue the command:

```
update-ca-certificates
```

The output of the command will look like this:

```
ubuntu@ip-192-168-99-133:/etc/ssl/certs$ sudo update-ca-certificates
Updating certificates in /etc/ssl/certs...
rehash: warning: skipping ca-certificates.crt,it does not contain exactly one certificate
or CRL
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
```

This update CA store will enable ZIA for more applications than just APT.

3.14 YUM

YUM is a software package management utility used in many popular Linux distributions, including Fedora, CentOS and Amazon Linux 2.

The following procedure was tested in Amazon Linux 2.0.20240719.0

If you encounter an error like this:

```
[ec2-user@ip-192-168-99-237 tls]$ sudo yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Could not retrieve mirrorlist https://amazonlinux-2-repos-us-west-1.s3.dualstack.us-
west-1.amazonaws.com/2/core/latest/x86_64/mirror.list error was
14: curl#60 - "SSL certificate problem: unable to get local issuer certificate"
```

Place the Zscaler root certificate in this location:

```
/etc/pki/ca-trust/source/anchors/zscaler.pem
```

And run the following command:

```
sudo update-ca-trust extract
```

You can now update your system via YUM with the following command:

```
sudo yum update
```

Troubleshooting Connection Problems

This section describes ways to diagnose connection problems when ZIA is activated on the Zscaler Client Connector. Connection problems with individual applications often manifest themselves in the fact that the affected applications do not function correctly, no longer respond when starting or during execution or do not start at all. The problems only occur if the network traffic is routed via ZIA.

For the ZIA Administrator the easiest way to diagnose connection problems is via the Analytics function in the ZIA Admin Portal. To check the logs in the Admin Portal, navigate to Analytics / Web Insights / Logs, Analytics / Firewall Insights / Logs and Analytics / DNS Insights / Logs. Filter the log file using the username and timestamp of the blocked traffic and look for "not allowed" connections.

If a diagnosis via the ZIA Admin Portal is not possible, an analysis can also be carried out via the Zscaler Client Connector or via network diagnosis tools such as Wireshark. The procedure for Mac OS X and Windows is described in the following sections.

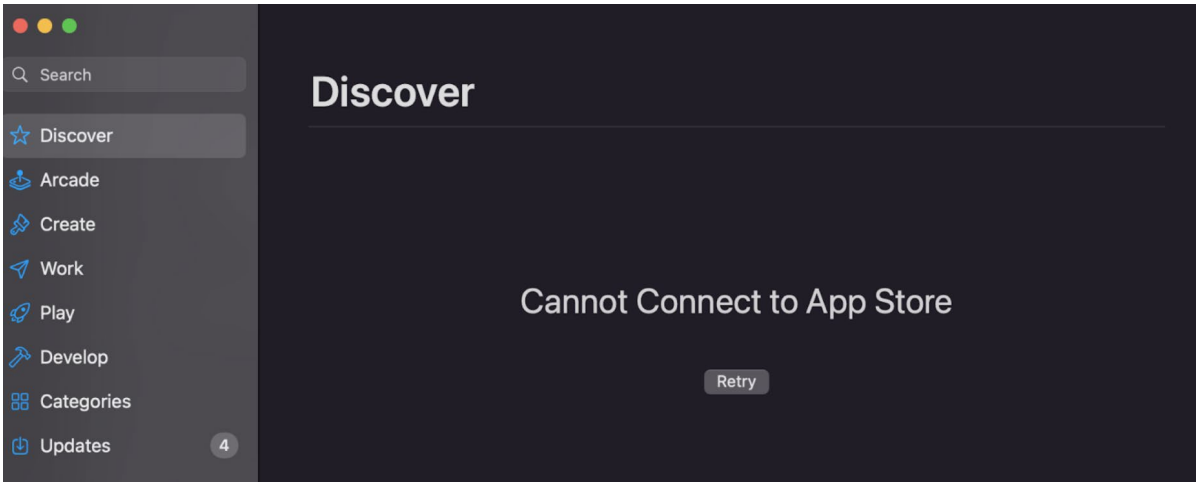
As of version 4.3 you can find local logs for the Zscaler Client Connector in the following locations:

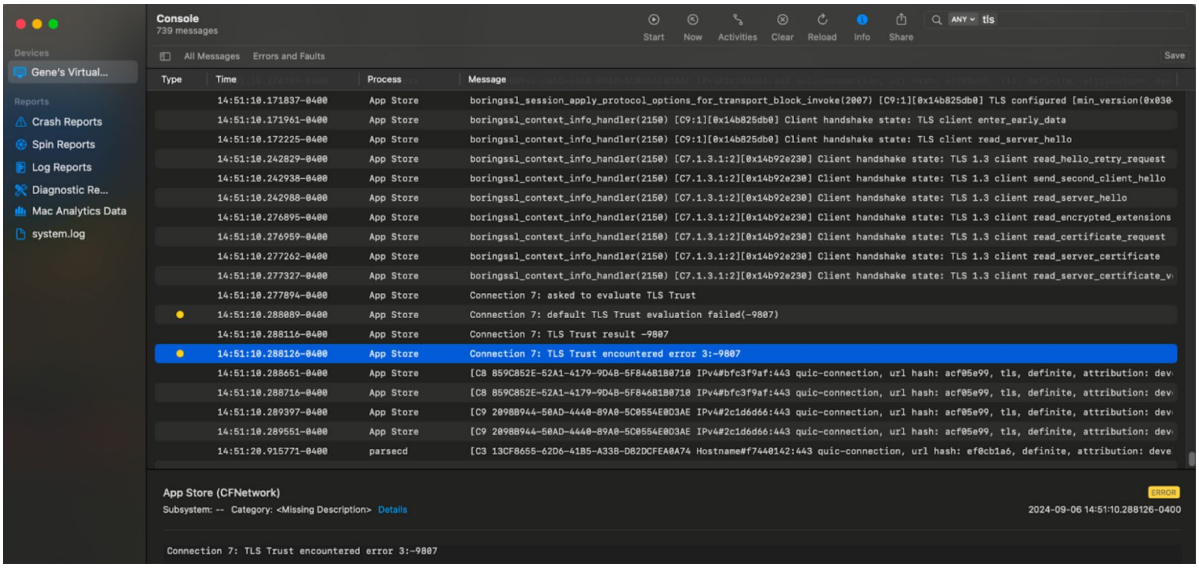
OS	Log Location	Details
macOS	/Library/Application Support/Zscaler/	Common Logs
	~/Library/Application Support/com.zscaler.zscaler/	User specific logs
Windows	C:\ProgramData\Zscaler %ALLUSERSPROFILE%\Zscaler	**Common Logs** ZSAService ZSAUpdater ZSATunnel (MT)
	C:\ProgramData\Zscaler\log-[User SID] %ALLUSERSPROFILE%\Zscaler\log-[User-SID]	**User Specific Logs** ZSATrayManager ZSATrayHelper ZSATunnel (User) ZSAUpm DBs
	C:\Users\[User Folder]\AppData\Local\Zscaler %LOCALAPPDATA%\Zscaler	**User AppData Logs** ZSATray ZSATrayHelper

Mac OS X

The application that is having trouble could be grouped in applications that open and don't show the correct or incomplete content, or applications that fail to open or quit unexpectedly. Regardless, use the Zscaler Client Connector Packet Capture with ZIA disabled as a comparison when the application works as expected.

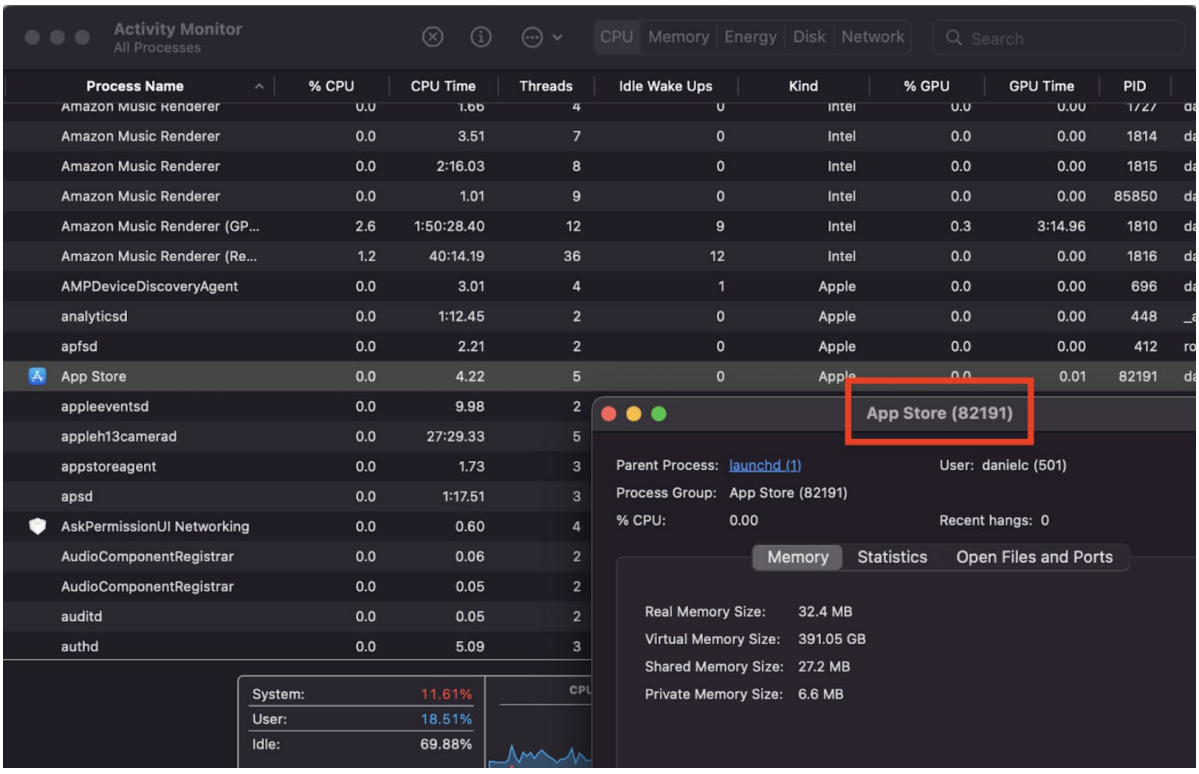
1. As an example we will use the Apple App Store. This application has certificate pinning, so it will not work with default settings with ZIA, in fact it will open and look like this:





The easiest way to troubleshoot an application in Mac is to use several built-in tools and the ability of the Zscaler Client Connector to do a Packet Capture.

2. If the application opens but does not work, use “Application Monitor” to get the PID of the application.

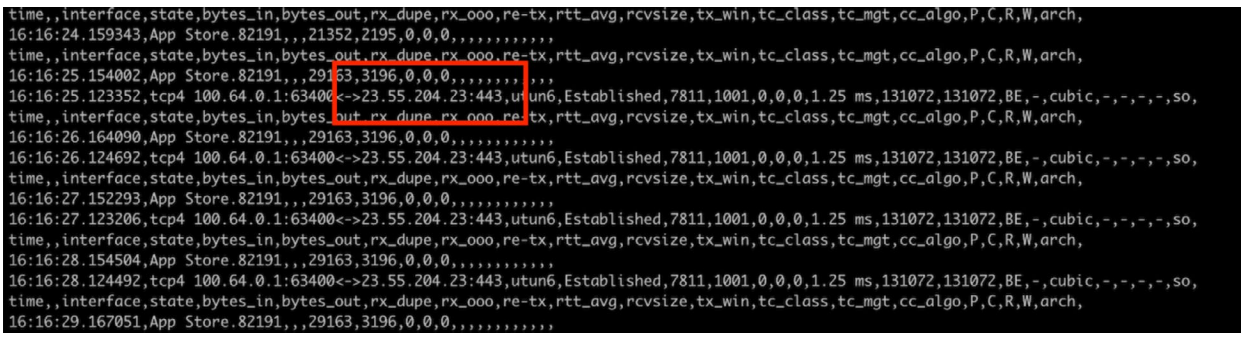


3. With the PID in hand we can now use nettop to track the connection done by the application. Use the following command:

   `nettop -L 0 -p <PID>`

Add a “–n” if you want to try to see IP addresses instead of names.

This will open a log file on screen that shows any TCP connections that the application tries. Most often you will be able to see the names or destination IP address, in this case 23.55.204.23. A reverse lookup for the IP points to akamaitechnologies.com, a known CDN provider.



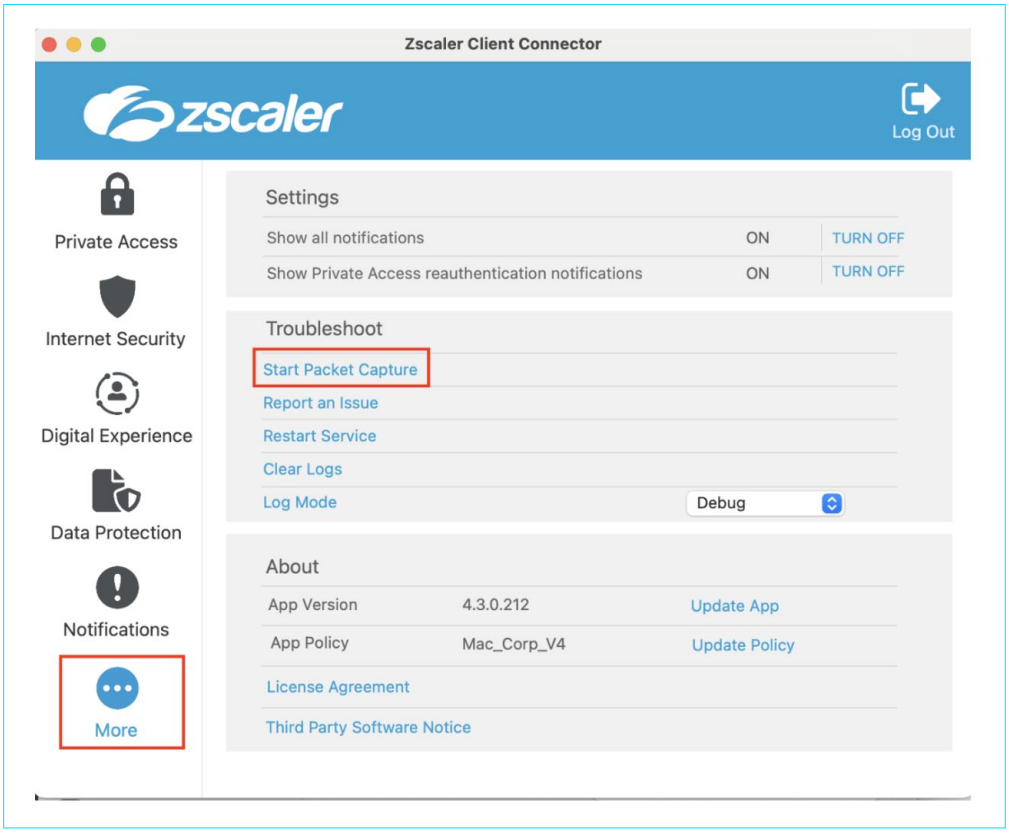
In the Appendix: Wireshark Troubleshooting Example we demonstrate how the developer can use Wireshark and other utilities to capture the traffic and see the DNS queries and possibly the TCP reset on the connection. In Appendix: Wireshark Troubleshooting TLS Example we present an alternate troubleshooting method that focuses on the TLS handshake.

Also with some of the Zscaler Client Connector logs it corroborates that indeed there is a connection problem with the application:

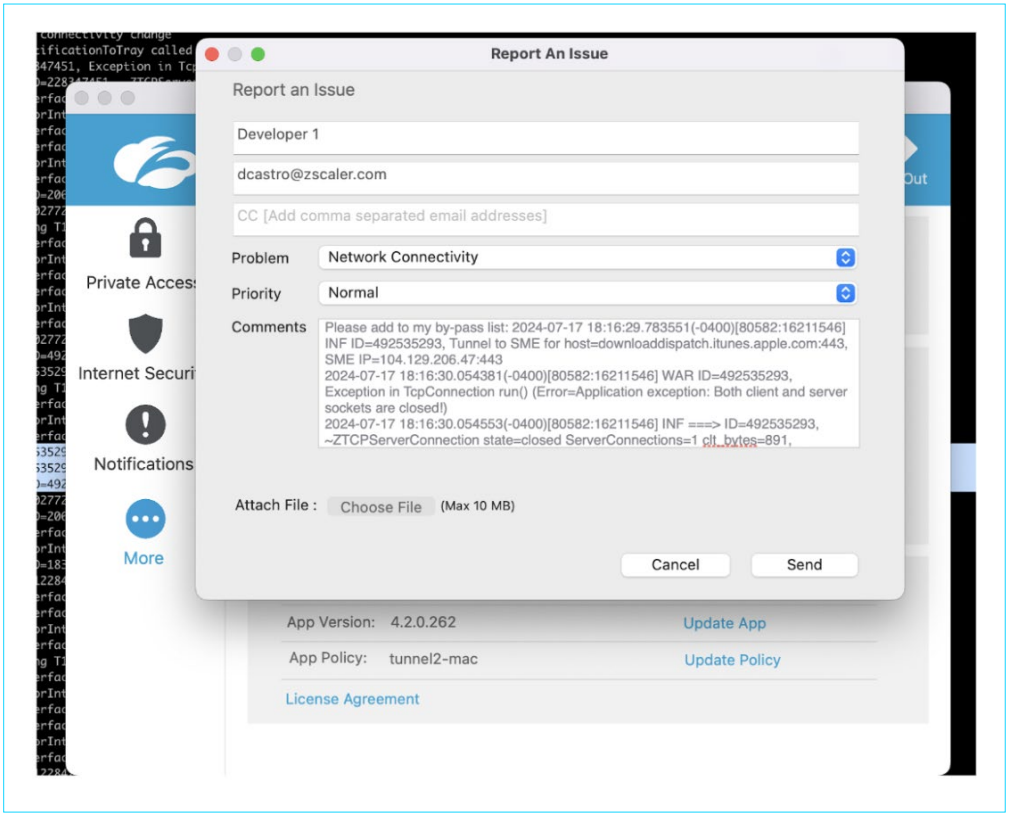
```
2024-07-17 18:16:29.783551(-0400)[80582:16211546] INF ID=492535293, Tunnel to SME for host=downloaddispatch.itunes.apple.com:443, SME IP=104.129.206.47:443

2024-07-17 18:16:30.054381(-0400)[80582:16211546] WAR ID=492535293, Exception in TcpConnection run() (Error=Application exception: Both client and server sockets are closed!
```

4. Now communicate with the ZIA Administrator or follow the procedure described in the section Feedback from User to Administrator to send a message to the administrators using the Zscaler Client Connector.

Report the issue using the Zscaler Client Connector More option:



With that information the Administrator can investigate the issue on the ZIA logs. In the form specify that there was a Network Connectivity issue and a copy of the Zscaler Client Connector log entry that shows the URL for the connection:

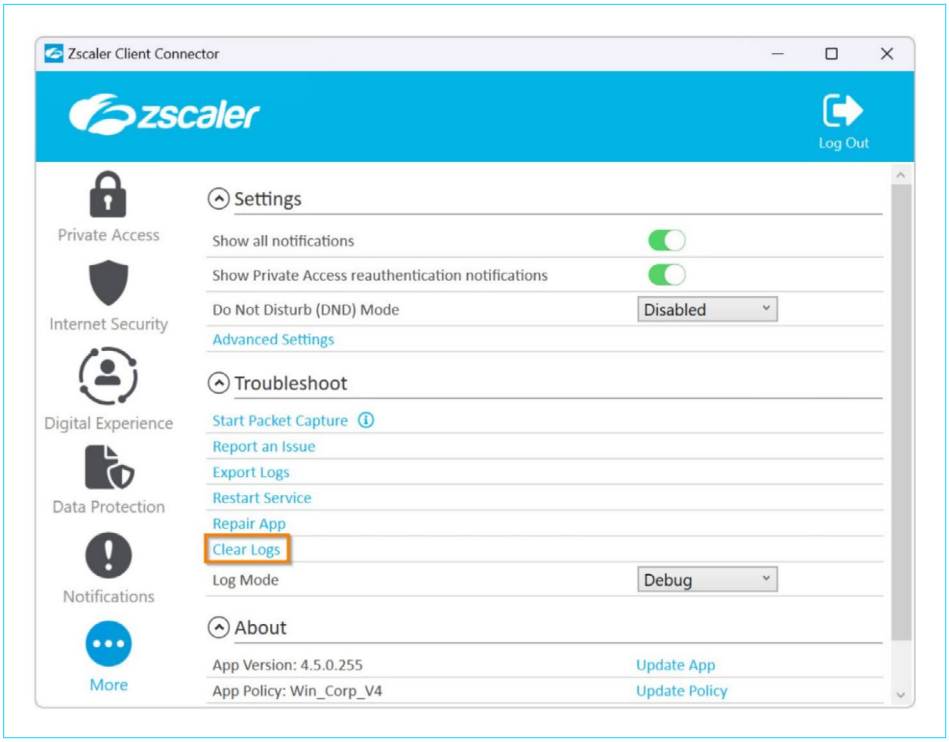


The developer has now provided enough information for the Administrator to act on this and possibly resolve the case.

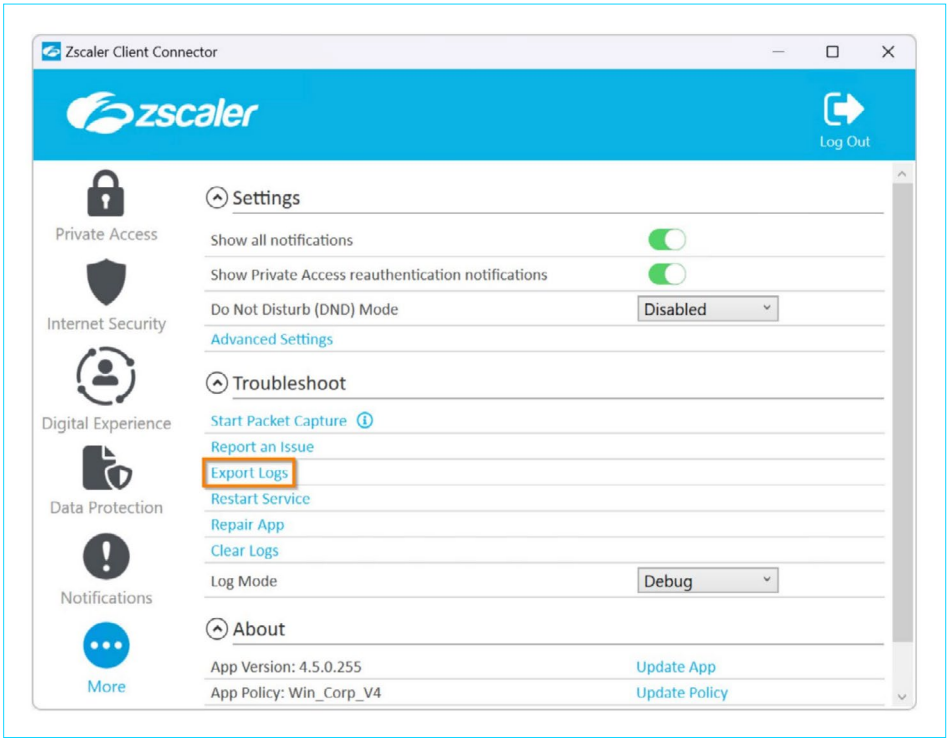
Windows

The Zscaler Client Connector is well suited for diagnostics on Windows systems, as it automatically creates the necessary logs and can also record data traffic for analysis in Wireshark if required.

To keep the size of the logs to be analyzed small, the **Clear Logs** option can be selected in the Zscaler Client Connector before reproducing the problem:



The **Start Packet Capture** option is then selected to obtain the Wireshark logs as well as the logs, the problem is reproduced and the recording of the network traffic is ended via **Stop Packet Capture.** The logs can then be exported in the form of a ZIP file via the **Export Logs** option in the Zscaler Client Connector:

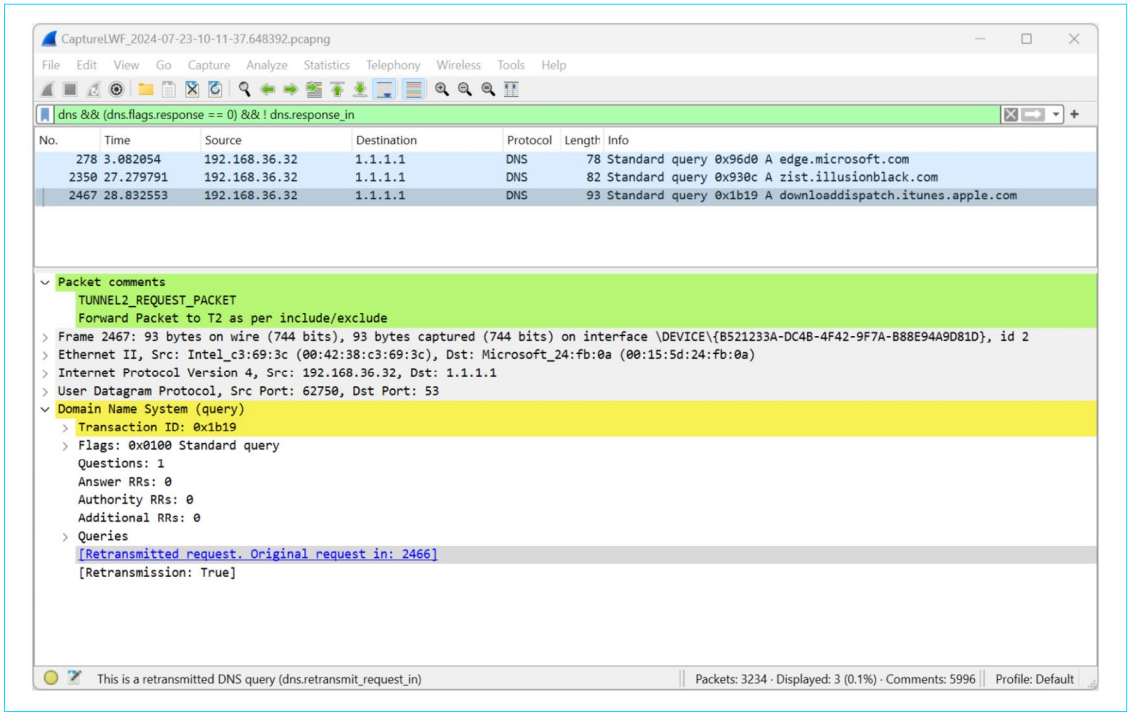


The ZIP file contains many logs. For the analysis of non-permitted host names and IP addresses, the logs beginning with the designation **ZSATunnel*** and the explicitly created Wireshark logs with the designation **CaptureLWF*** are of interest.

To select possible problem candidates, the network record is opened with Wireshark and searched for unanswered DNS queries. If the Zscaler Zero Trust Exchange blocks a connection at DNS level, this is shown in the logs by the fact that a DNS query is sent but no response is returned. The following Wireshark filter detects such requests:

```
dns && (dns.flags.response == 0) && ! dns.response_in
```

The following queries are displayed as a result:

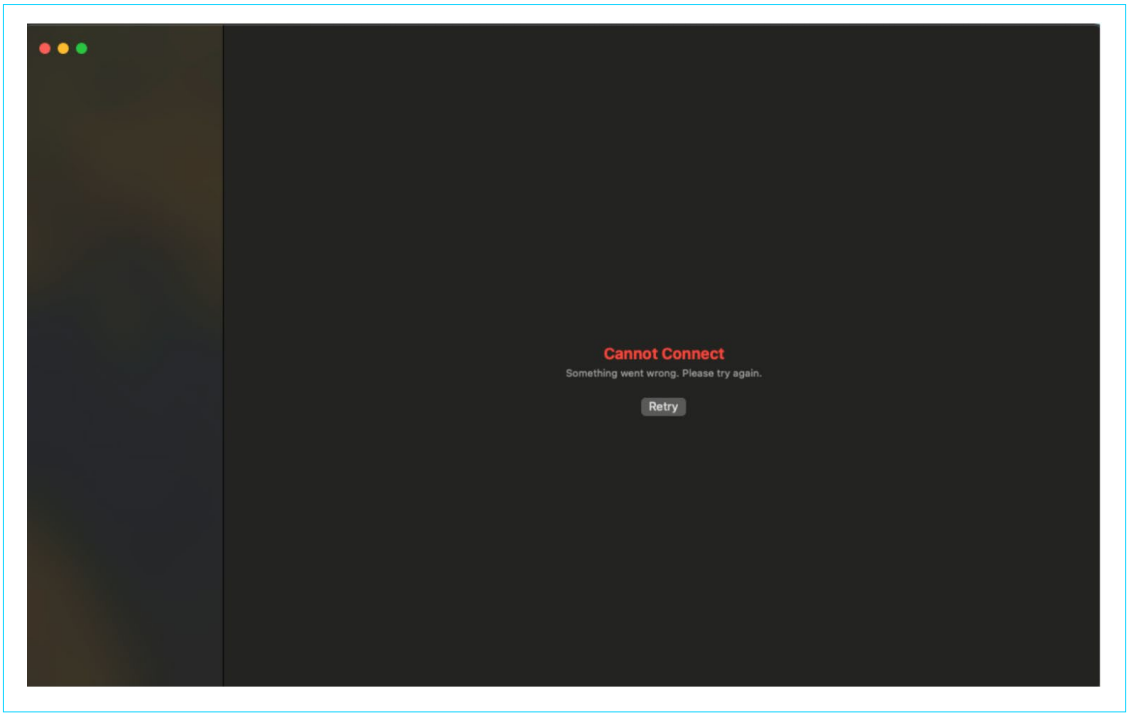


There are other reasons why a DNS request may not be answered, but the three hostnames listed – edge.microsoft.com, zist.illusionblack.com and downloaddispatch.itunes.apple.com – are candidates for further investigation.

Appendix

Wireshark Troubleshooting TLS Example

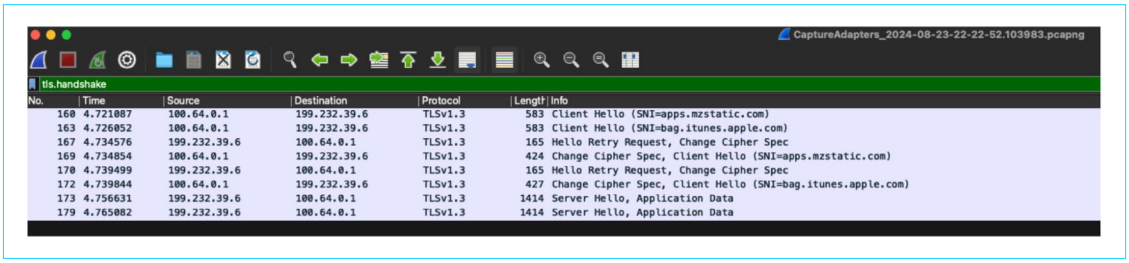
1. Start at the same spot for App Store broken:



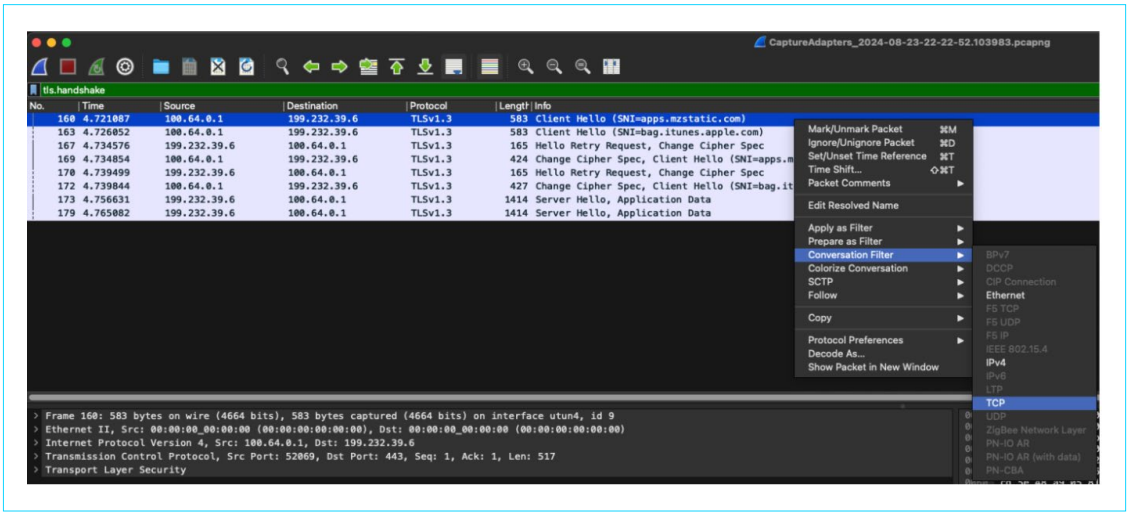
2. Open ZCC, Clear Logs, Start Packet Capture. Then hit the displayed “Retry” button from above. Stop Packet Capture, Export Logs.

3. Unzip Log bundle, search for .pcapng file with today’s date & open in Wireshark.

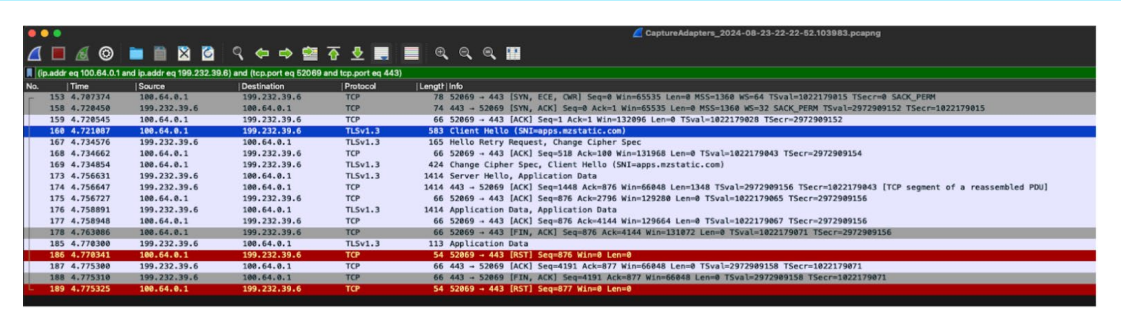
4. First look for TLS handshakes using the “tls.handshake” filter:



5. Then use the Conversation Filter option to view a likely suspect as displayed:

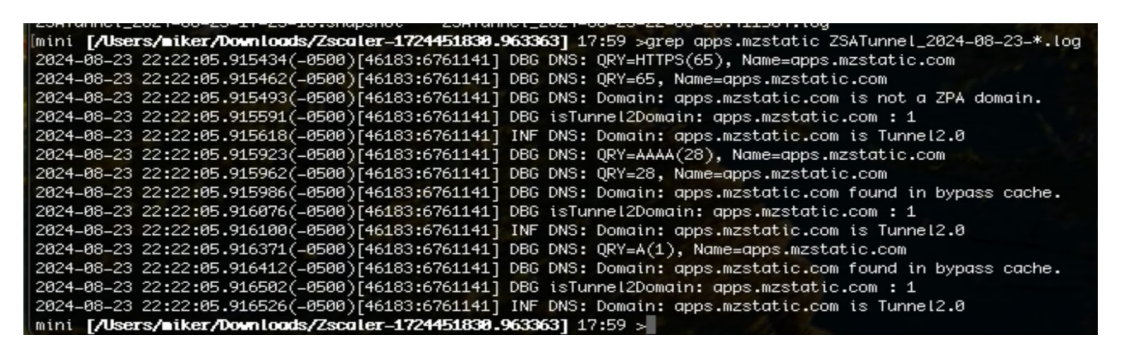


6. Here you can see that immediately after the server side transmits it's hello information in packets 173, 174, & 176. Immediately following the ACK of packet 176 (packet 177) the client side (App Store) sends a RST in packet 178. This is what it looks like when the application terminates a session without finding an acceptable certificate (certificate pinning).



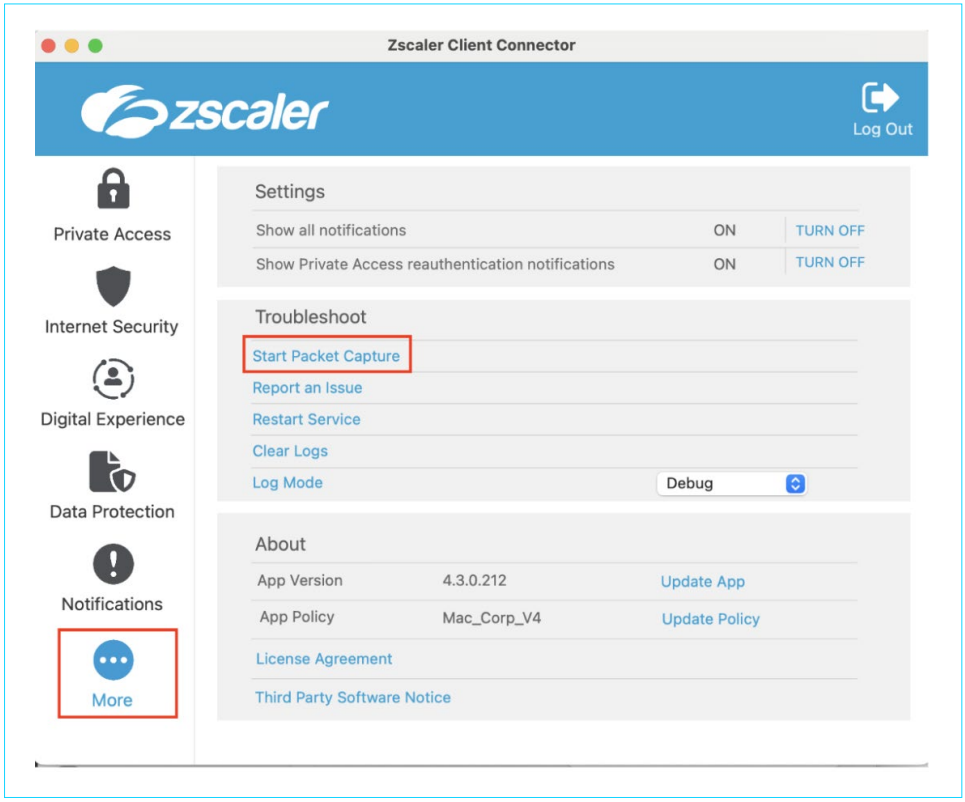
7. In this case both of these destinations (apps.mzstatic.com & bag.itunes.apple.com) display the same behavior shown using the mzstatic example above.

8. Note, the ZSATunnel logs here show almost nothing from a web perspective as the web packets are routed directly to the tunnel:



Wireshark Troubleshooting DNS Example

Start the Packet Capture in the Zscaler Client Connector:



In another terminal window, and while the Packet Capture is already running issue the following command to clear the DNS cache:

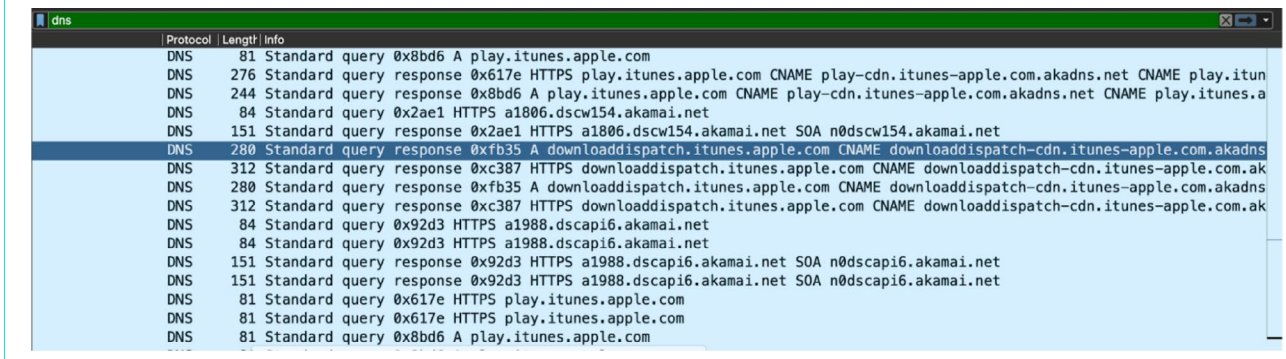
```
sudo dscacheutil -flushcache: sudo killall -HUP mDNSResponder
```

Refresh the application, open the application or any similar action that forces the application to use the network. In our case we hit the Retry button.

Proceed to stop the Packet Capture and Export the logs to the local file system.

Unzip the file and look for a file with filename extension **pcapng** and open this file in Wireshark. Once open use the “dns” filter and look for the relevant entry that points to the application you are trying to use.

Note: The application developer could have used an acronym, a short name or some random cdn as the connection that is failing, in this case the application provider is Apple and the IP is owned by Akamai Technologies. Therefore, look for a combination of Apple and Akamai in the domain name. The image below illustrates the DNS entry as a combined list of CNAME entries:



The DNS response is:

```
Standard query response 0xc387 HTTPS downloaddispatch.itunes.apple.com CNAME
downloaddispatch-cdn.itunes-apple.com.akadns.net CNAME downloaddispatch.itunes.apple.com.
edgesuite.net CNAME a1988.dscapi6.akamai.net SOA n0dscapi6.akamai.net
```

Following the next DNS request in the trace will resolve a1988.dscap16.akamai.com to 23.55.204.23 This Akamai IP falls within the ASN 35994 (https://bgp.he.net/AS35994) so is 23.46.150.50. Name verification indicates that all other names are within this ASN.

When exploring this DNS name on the exported logs from Zscaler Client Connector you will find this the Application Exception error for Client and Server connection close:

```
2024-07-17 18:16:29.783551(-0400)[80582:16211546] INF ID=492535293, Tunnel to SME for
host=downloaddispatch.itunes.apple.com:443, SME IP=104.129.206.47:443
2024-07-17 18:16:30.054381(-0400)[80582:16211546] WAR ID=492535293, Exception in
TcpConnection run() (Error=Application exception: Both client and server sockets
are closed!
2024-07-17 18:16:30.054553(-0400)[80582:16211546] INF ===> ID=492535293,
~ZTCPServerConnection state=closed ServerConnections=1 clt_bytes=891, srv_bytes=4230!
2024-07-17 18:16:29.783551(-0400)[80582:16211546] INF ID=492535293, Tunnel to SME for
host=downloaddispatch.itunes.apple.com:443, SME IP=104.129.206.47:443
2024-07-17 18:16:30.054381(-0400)[80582:16211546] WAR ID=492535293, Exception in
TcpConnection run() (Error=Application exception: Both client and server sockets are
closed!)
2024-07-17 18:16:30.054553(-0400)[80582:16211546] INF ===> ID=492535293,
~ZTCPServerConnection state=closed ServerConnections=1 clt_bytes=891, srv_bytes=4230!
```

So the first entry shows the URL for the connection and the string `Error=Application exception: Both client and server sockets are closed!` was exactly what we needed to make sure this DNS name requires a corresponding bypass rule to be created in ZIA.

PKI and Certificate Authorities

Self Signed Intermediate CA Certificates

In order for Zscaler to verify SSL traffic, the operating system from which an SSL connection is established to a target server on the Internet must trust the root and intermediate CA certificate used by Zscaler.

It is possible to create and use your own root and intermediate CA certificate or to use the one provided by Zscaler.

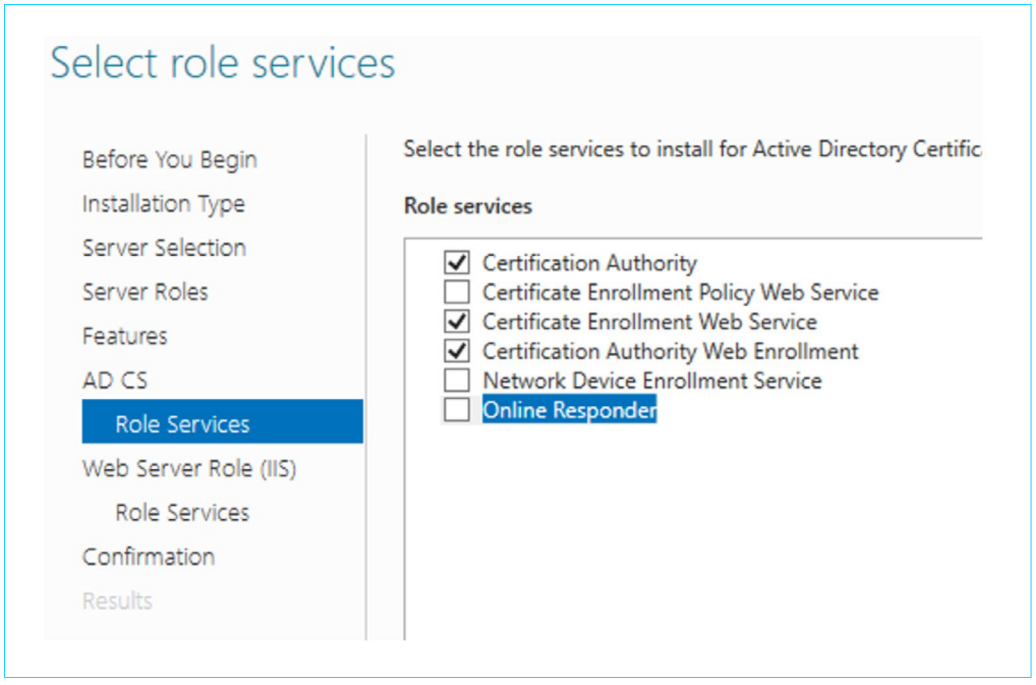
If such a certificate is placed in the SSL Inspection section of the ZIA Admin Portal, then this certificate is installed by the Zscaler Client Connector in the certificate store of the operating system so that all applications that access the certificate store of the operating system trust this certificate.

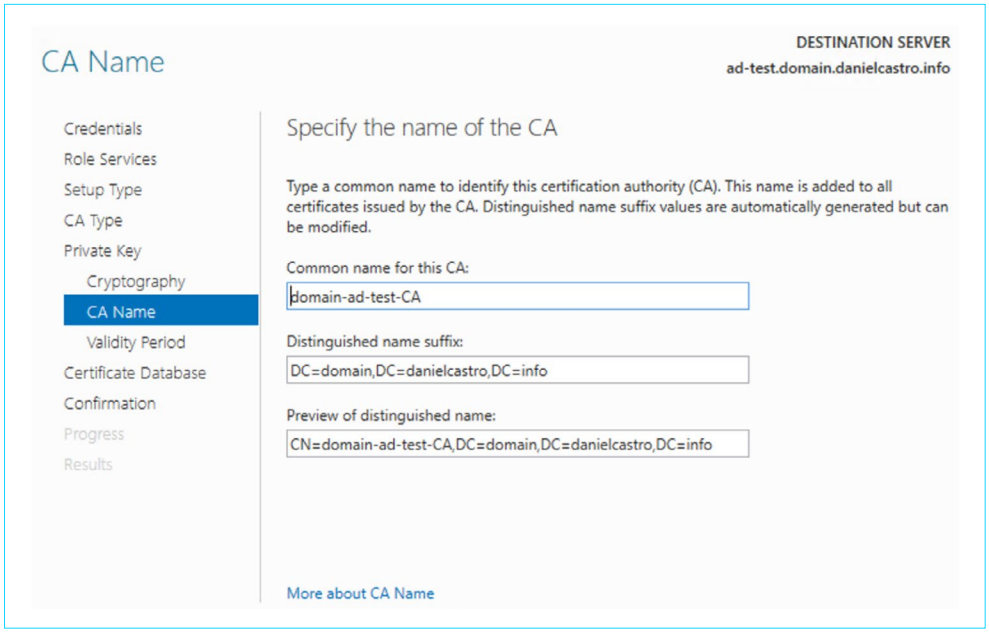
In managed environments, a certification authority usually already exists and a suitable certificate has already been stored in ZIA. If not, the following sections describe various options for creating a corresponding root and intermediate certification authority certificate that can be used in this case.

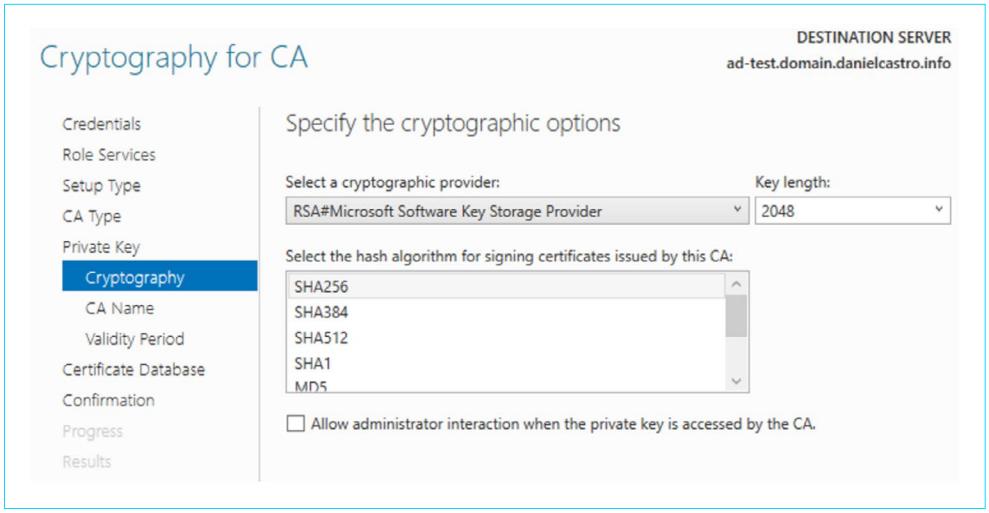
Windows Certificate Authority Installation Procedure

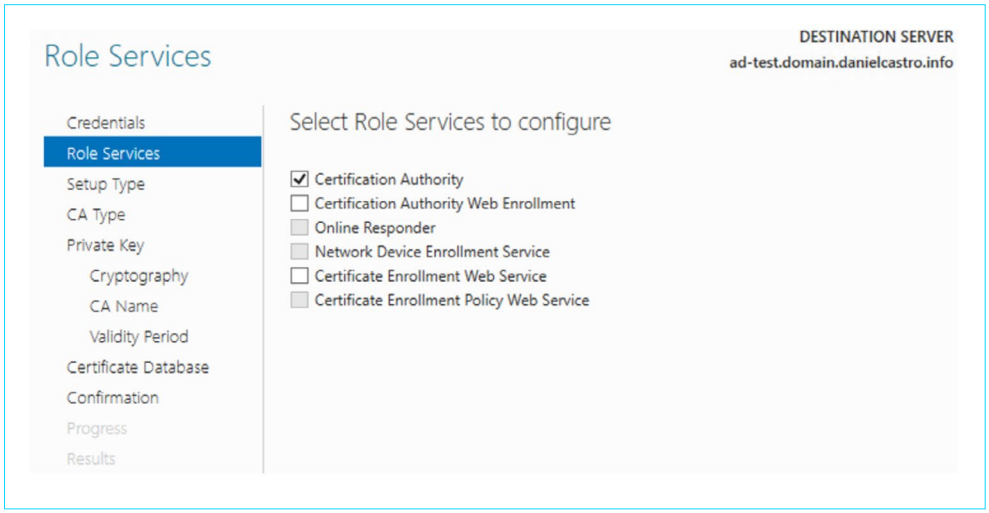
A Windows Certificate Authority can be deployed on a member server or domain controller of an Active Domain or on a standalone server. If the certificate authority is installed on a domain controller or member server of an Active Directory domain, the root certificate authority certificate is automatically distributed to all member servers and computers via group policy. If it is a standalone server, the root certification authority certificate must be published manually in the Active Directory domain.

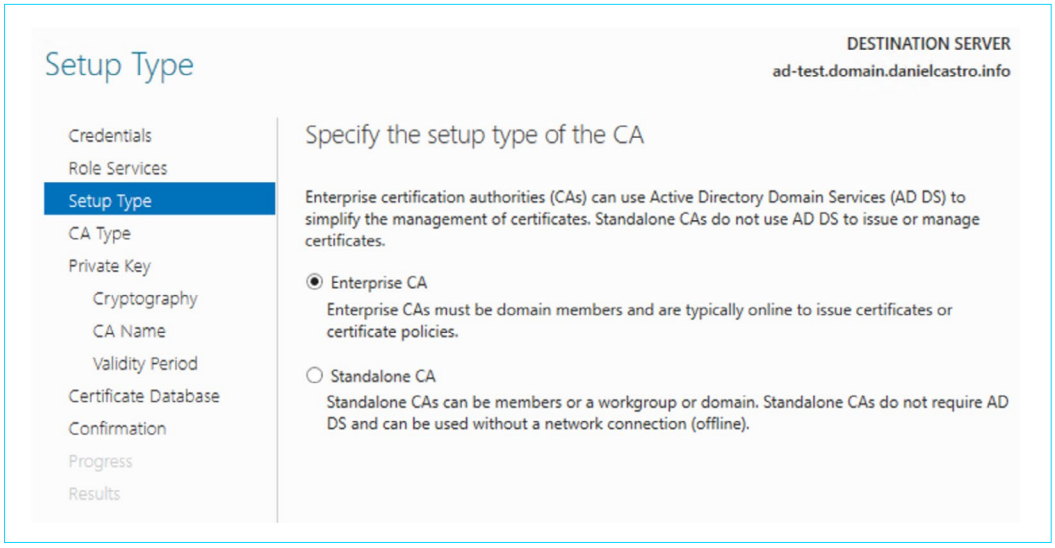
In the rest of this section we will show how how to configure AD and ADCS and then how to install an Enrolment Certificate in ZPA:

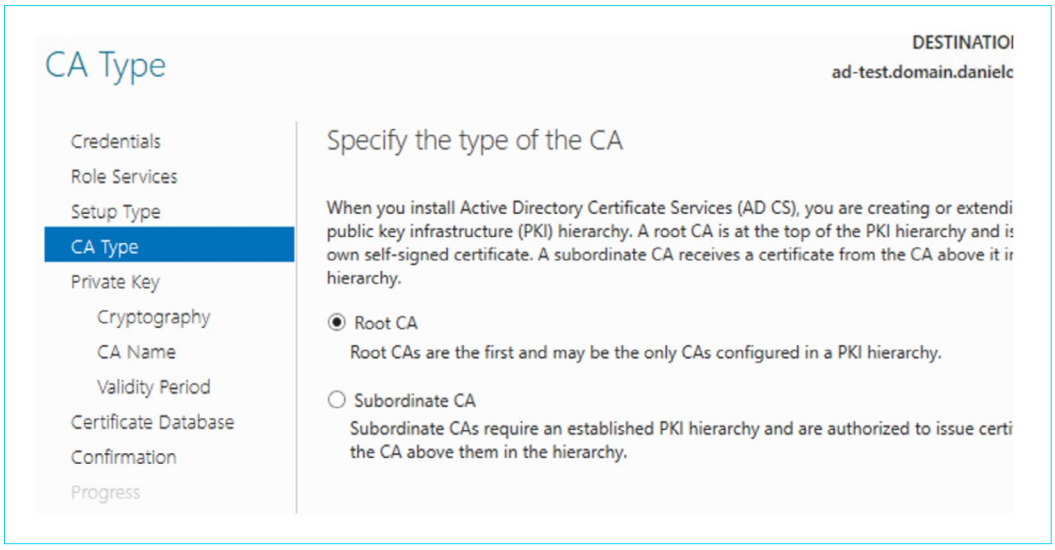


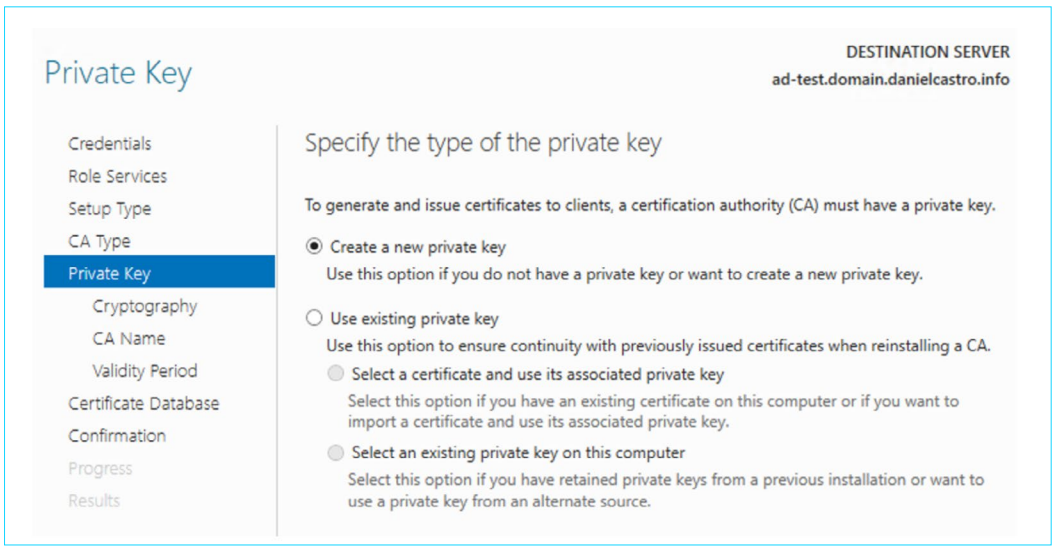


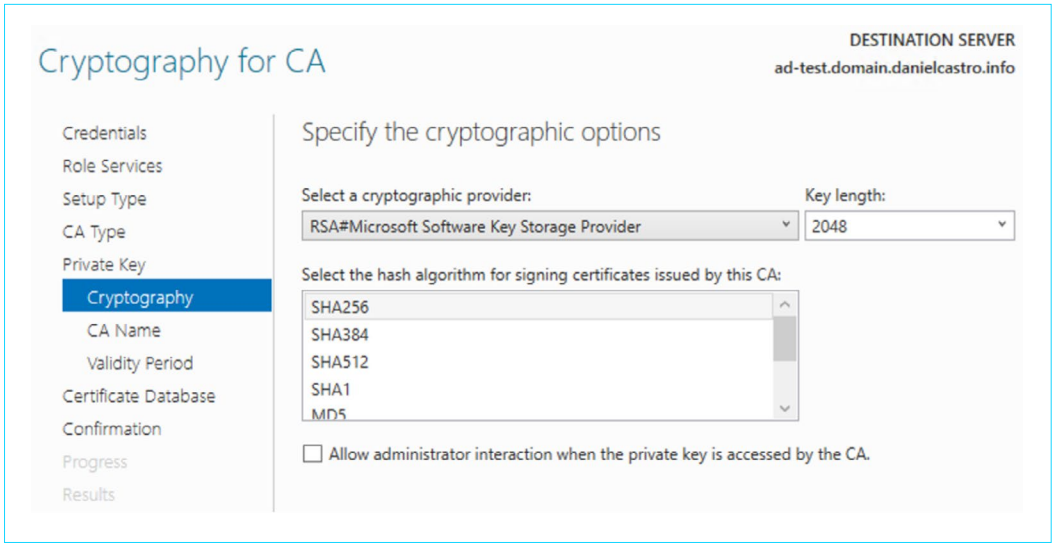


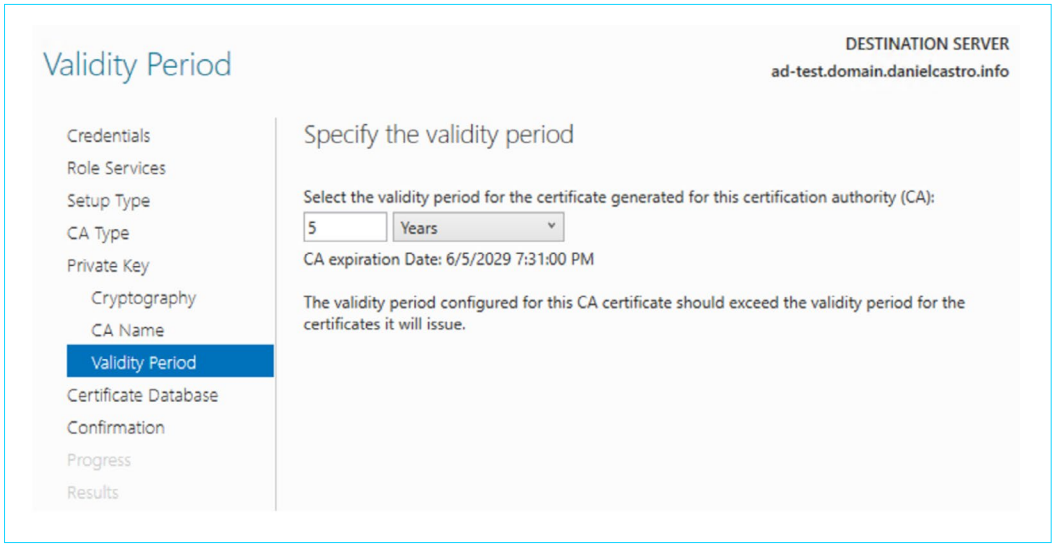


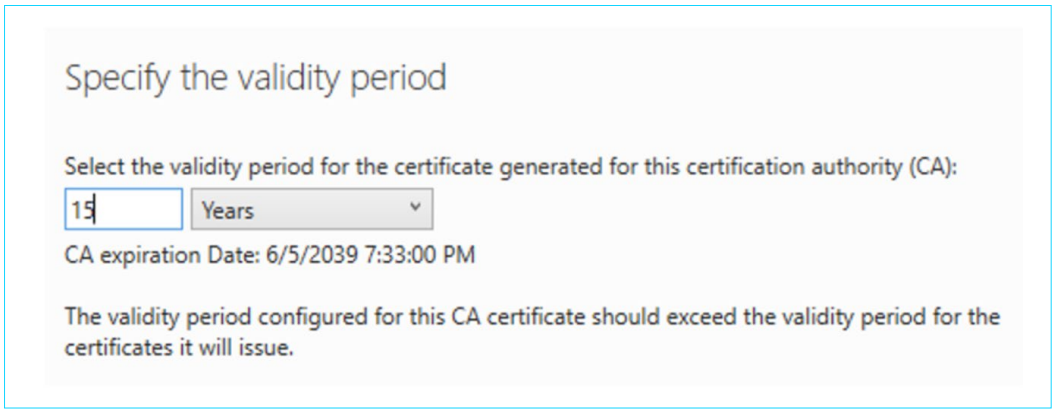


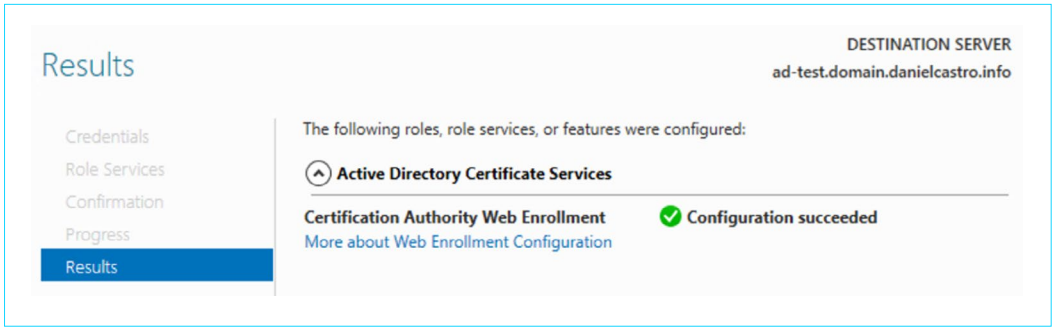


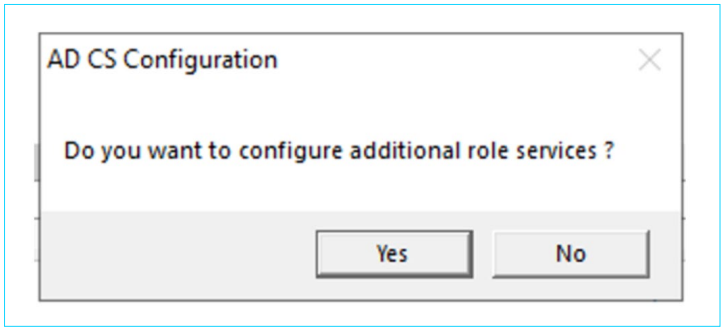


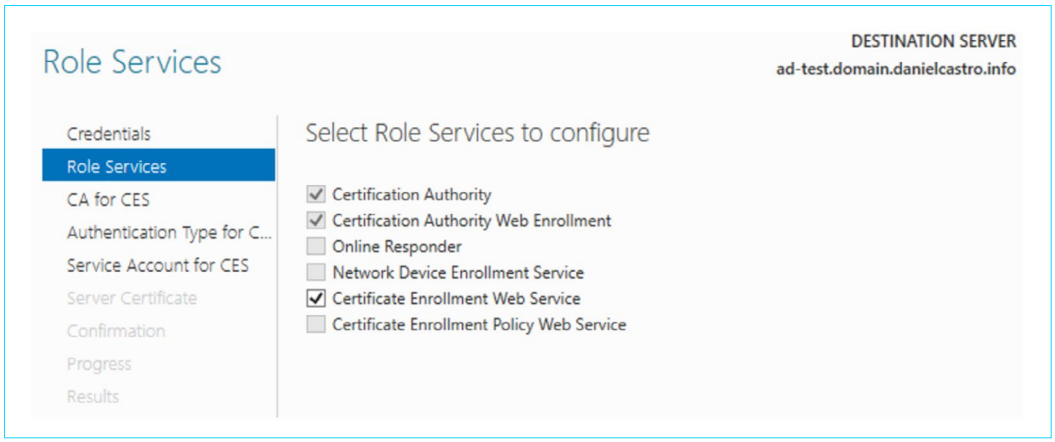


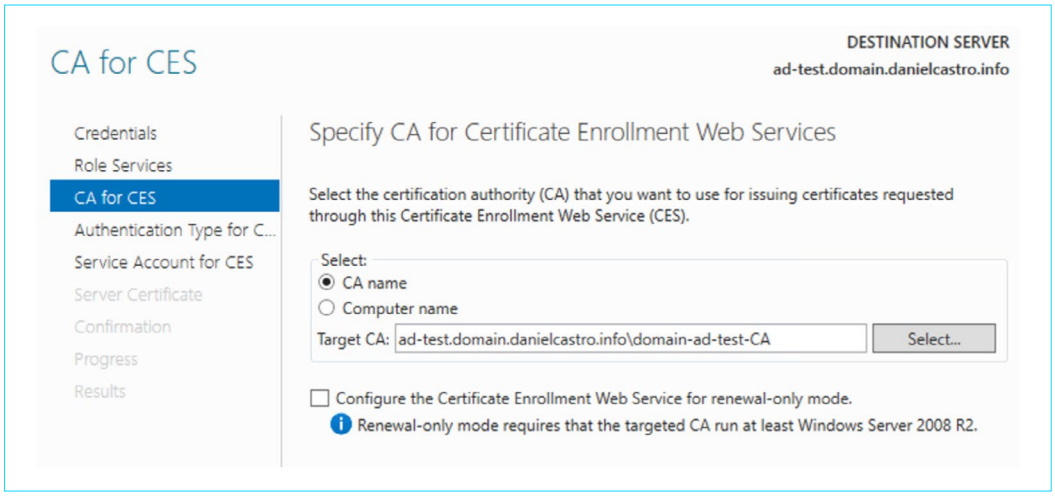


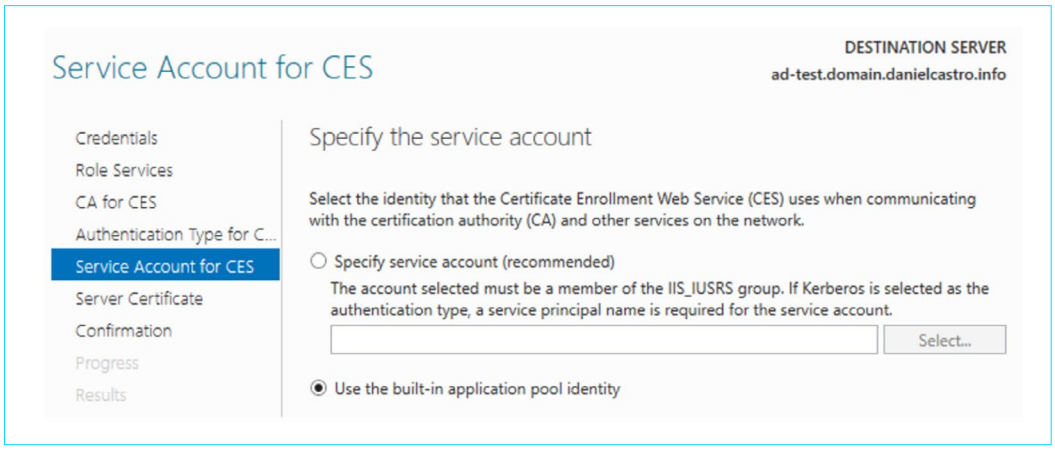


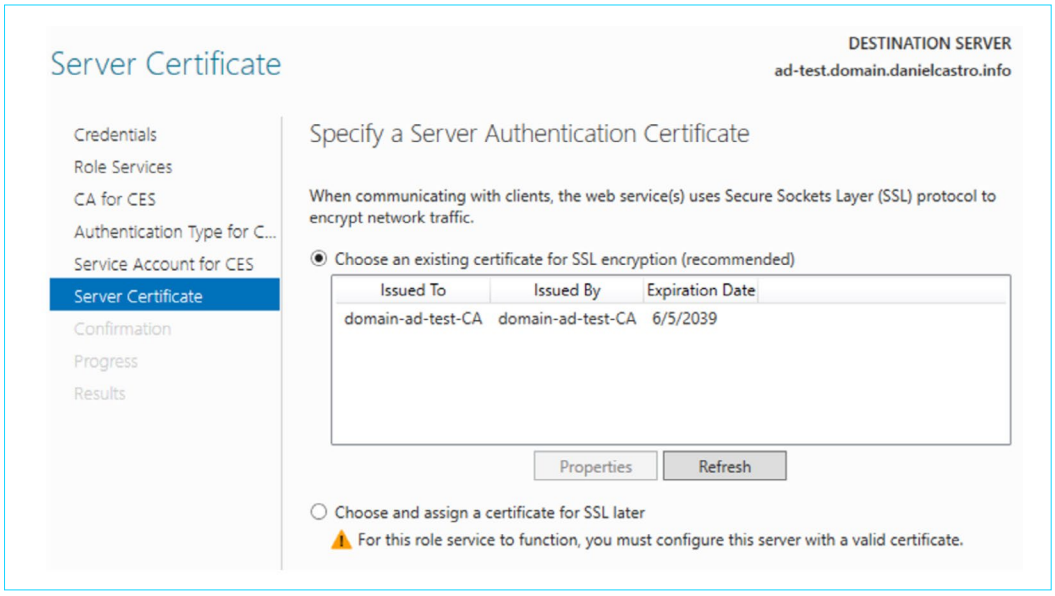


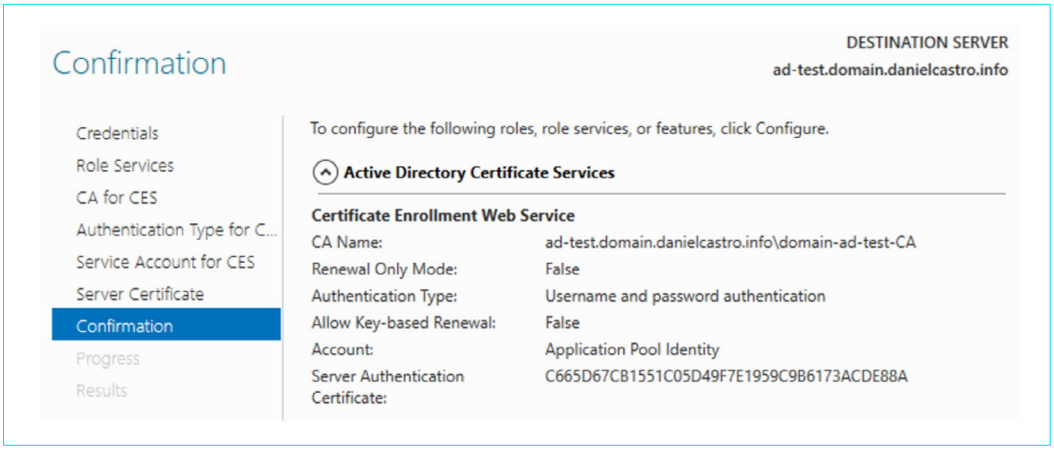




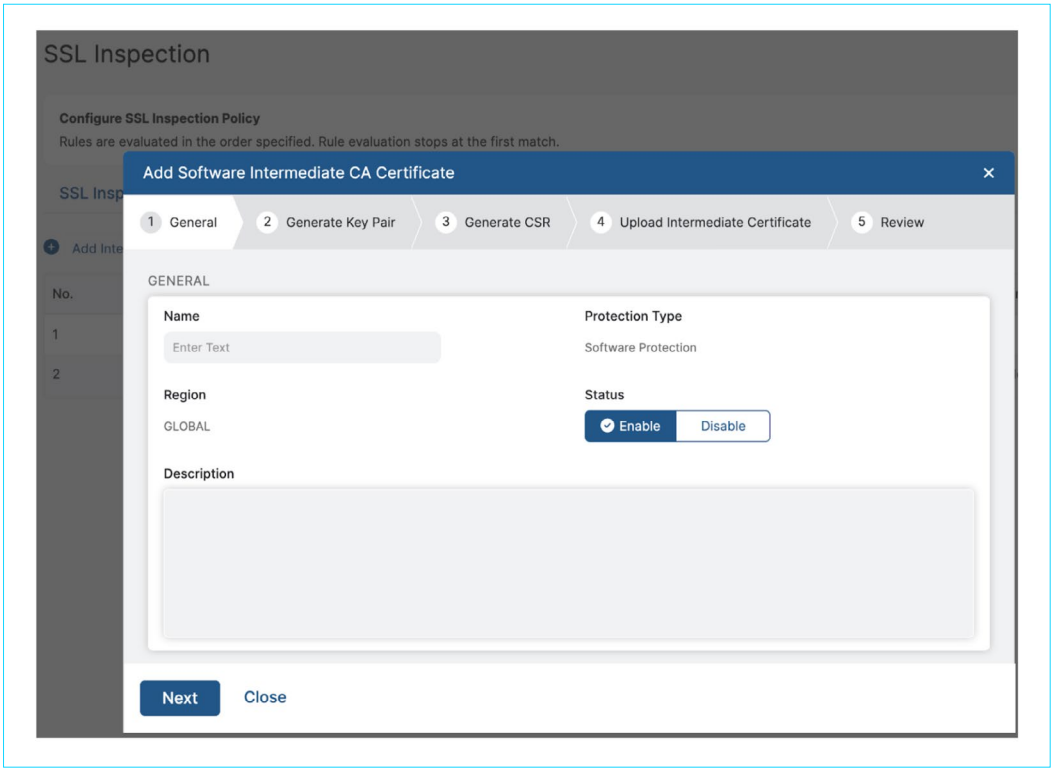








At this point ADCS has been successfully installed and we can proceed to open the browser and explore the CA services. But first head over to either ZIA or ZPA and request a new CSR:



Certificates without Windows Certificate Authority

In addition to creating an intermediate certification authority certificate as described in the previous section, it is also possible, for testing purposes, to create corresponding certificates on a Windows client computer. It should be noted that certificates created in this way are only available on the local computer. For use on other computers, both the root and intermediate certification authority certificates must be exported and imported on the other computers, for example via management software such as Intune or through group policies for Active Directory domain members.

Create a new Root CA Certificate with PowerShell

When ADCS is ready you can create a new root CA certificate \ with the following PowerShell cmdlet:

```
PS C:\> $rootCert = New-SelfSignedCertificate -CertStoreLocation
Cert:\CurrentUser\My -DnsName "RootCA" -TextExtension
@("2.5.29.19={text}CA=true") -KeyUsage CertSign,CrlSign,DigitalSignature
```

The root CA certificate (without the private key) can then be exported as a file using the following command:

```
PS C:\> [String]$rootCertPath = Join-Path -Path 'cert:\CurrentUser\My\'
-ChildPath "$($rootCert.Thumbprint)"
PS C:\> Export-Certificate -Cert $rootCertPath -FilePath 'RootCA.crt'
```

Create a new Intermediate CA Certificate with PowerShell

A new intermediate certification authority certificate, which is signed with the previously created root CA certificate, can then be created with the following command:

```
PS C:\> $intermediateCACert = New-SelfSignedCertificate -CertStoreLocation
Cert:\LocalMachine\My -DnsName "IntermediateCA" -TextExtension
@("2.5.29.19={text}CA=true") -KeyLength 2048 -KeyUsage
CertSign,CrlSign,DigitalSignature -Signer $rootCert
```

The Intermediate CA certificate (without the private key) can then be exported as a file using the following commands:

```
PS C:\> [String]$intermediateCertPath = Join-Path -Path 'cert:\LocalMachine\My\'
-ChildPath "$($intermediateCACert.Thumbprint)"
PS C:\> Export-Certificate -Cert $intermediateCertPath -FilePath
'IntermediateCA.crt'
```

OpenSSL Certificate Authority Creation

In this section we will show you how to create a root CA from which openssl can create additional certificates or can be uploaded to ZIA for TLS interception.

Create a configuration file for the root CA, we will call this rootCA_openssl.conf, modify the folder locations to match your configuration

```
[ ca ]
# `man ca`
default_ca = CA_default

[ CA_default ]
# Directory and file locations.
# The root key and root certificate.
private_key       = root.key
certificate       = root.pem
new_certs_dir             = /Users/danielcastro/Downloads
database          = index.txt
policy            = policy_strict
dir               = "~/CA"
certs             = $dir/certs
crl_dir           = $dir/crl
database          = $dir/index.txt
serial            = $dir/serial
RANDFILE          = $dir/.rand

[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName             = optional
stateOrProvinceName     = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

```
[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ req ]
distinguished_name       = req_distinguished_name
extensions               = v3_ca
req_extensions           = v3_ca

[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection
```

```
[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always

[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default         = CA
countryName_min             = 2
countryName_max             = 2
organizationName            = Zscaler Private CA
organizationName_default = Zscaler Inc.
0.organizationName                  = Zscaler Private
organizationalUnitName              = Pro Serv
commonName                          = ca.danielcastro.info
emailAddress                        = dcastro@zscaler.com
```

Define the serial number of the generated certificates with this command:

```
> echo 1000 > serial
```

Proceed to create the ca private key, and certificate with:

```
> openssl req -config rootCA_openssl.conf -key root.key -new -x509 -days 7300 -sha256
-extensions v3_ca -out root.pem
```

Now in the same folder you will have root.pem, root.key and a configuration file. With these three files you can create server certificates, intermediate certificates and many more.

PKI and Customer Provided Certificate Authority

To provide an Intermediate Certificate in ZIA and the organization uses their own enterprise Certificate Authority (CA) it is necessary to understand Public key infrastructure (PKI).

Public key infrastructure (PKI) is a system of processes, technologies, and policies that allows you to encrypt and/or sign data. With PKI, you can issue digital certificates that authenticate the identity of users, devices, or services. These certificates work for both public web pages and private internal services. In the enterprise this is usually a web service or software that is run as part of the enterprise infrastructure, this is normally called the Certificate Authority (CA).

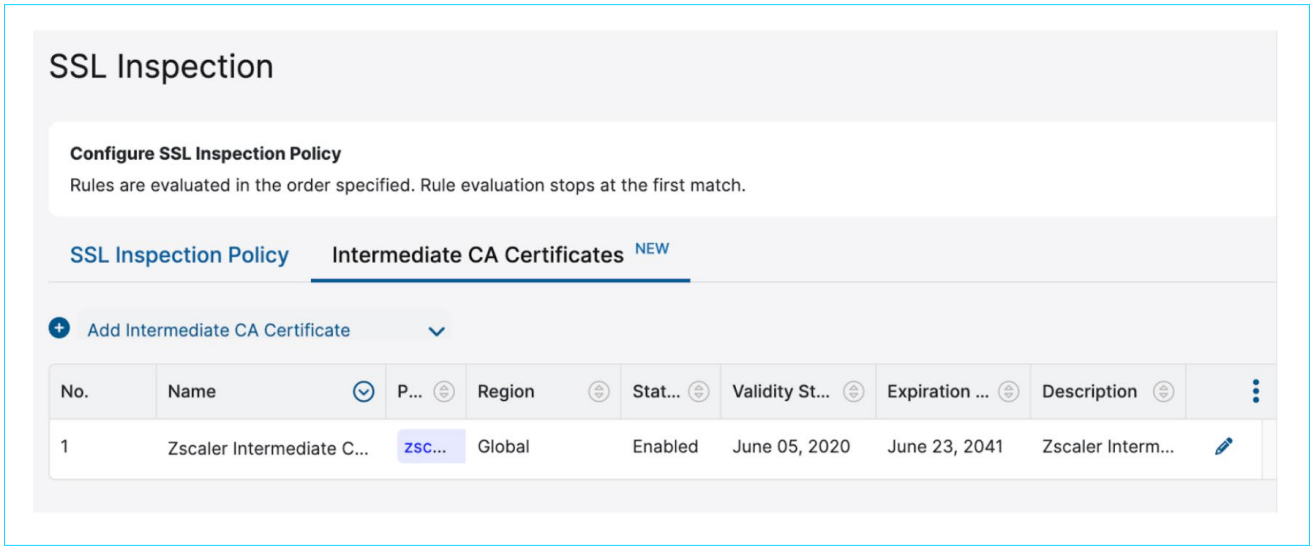
In the context of this whitepaper we will refer to the following objects and operations:

- Public Key is a number that is inside a certificate, when applied to data it will encrypt the data but the data can only be decrypted with the private key.
- Private Key is a file (inside there is a string that represents a number) that was created and will allow it to decrypt any data that is encrypted using the public key of the certificate. This file must be stored in a safe place.
- Certificate Sign Request (CSR) is a file that describes the desired certificate that is being requested.
- Signing is the process of taking in a CSR to the CA and asking for a certificate.
- Custom Intermediate Certificate is a certificate (used in ZIA and ZPA but cannot be the same) that allows operations as if it were a Certificate Authority, becoming a delegate of a higher tier Certificate Authority. ZIA uses this for TLS Interception and ZPA uses this for certificate enrollment.

A client in order to make a secure connection must have a trust relationship to the Certificate Authority that issued the certificate provided by the server. That inherited trust enables secure communications. Every device has a list of trusted certificate authorities (CA) that are provided by the manufacturer of the device. Installing an additional CA is usually documented by the manufacturer or software provider so that the device can trust any secure connections that make use of certificates under that CA. Furthermore, most individual software components provide a method by which insecure (untrusted) communication can be done, for example in python's urllib3 a specific configuration needs to be done when sending HTTP requests to a server with an untrusted certificate, but this configuration does not enable trust between the client and server.

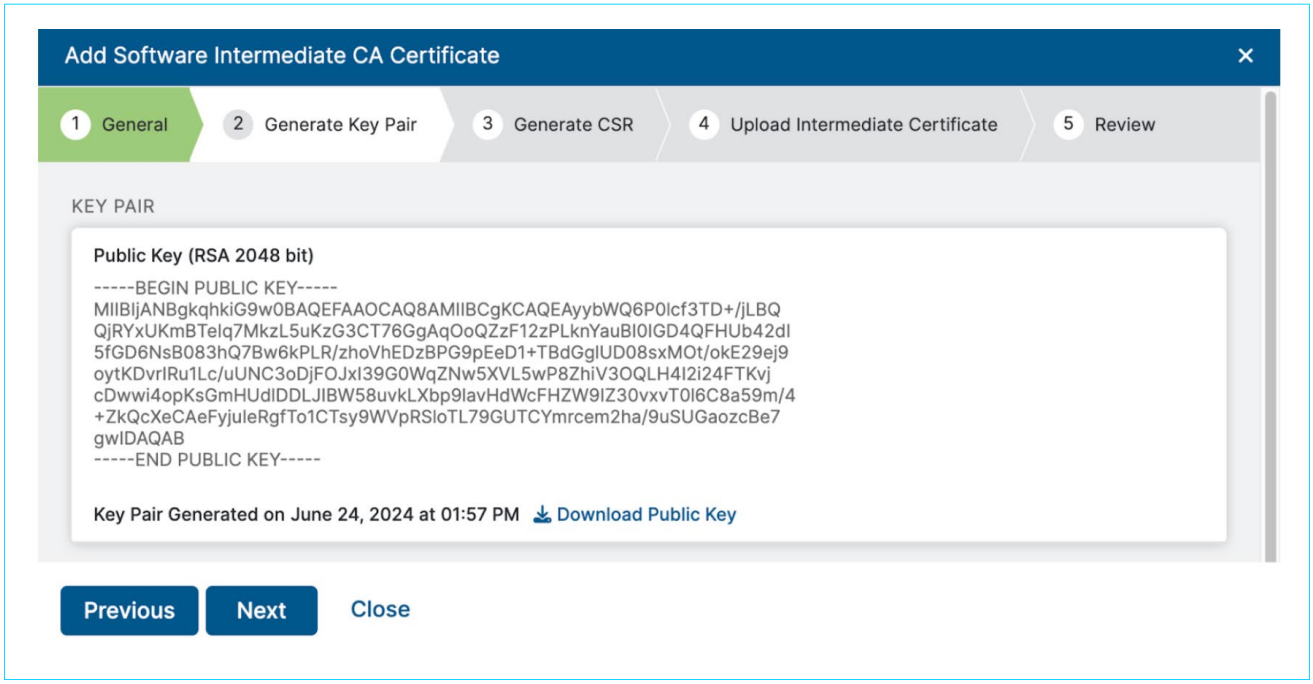
ZIA TLS Interception Certificate

When an organization wants to bring their own certificate to ZIA they first need to create a request, the procedure is as follows, start on ZIA's SSL Inspection – Intermediate CA Certificates and use the Add option:

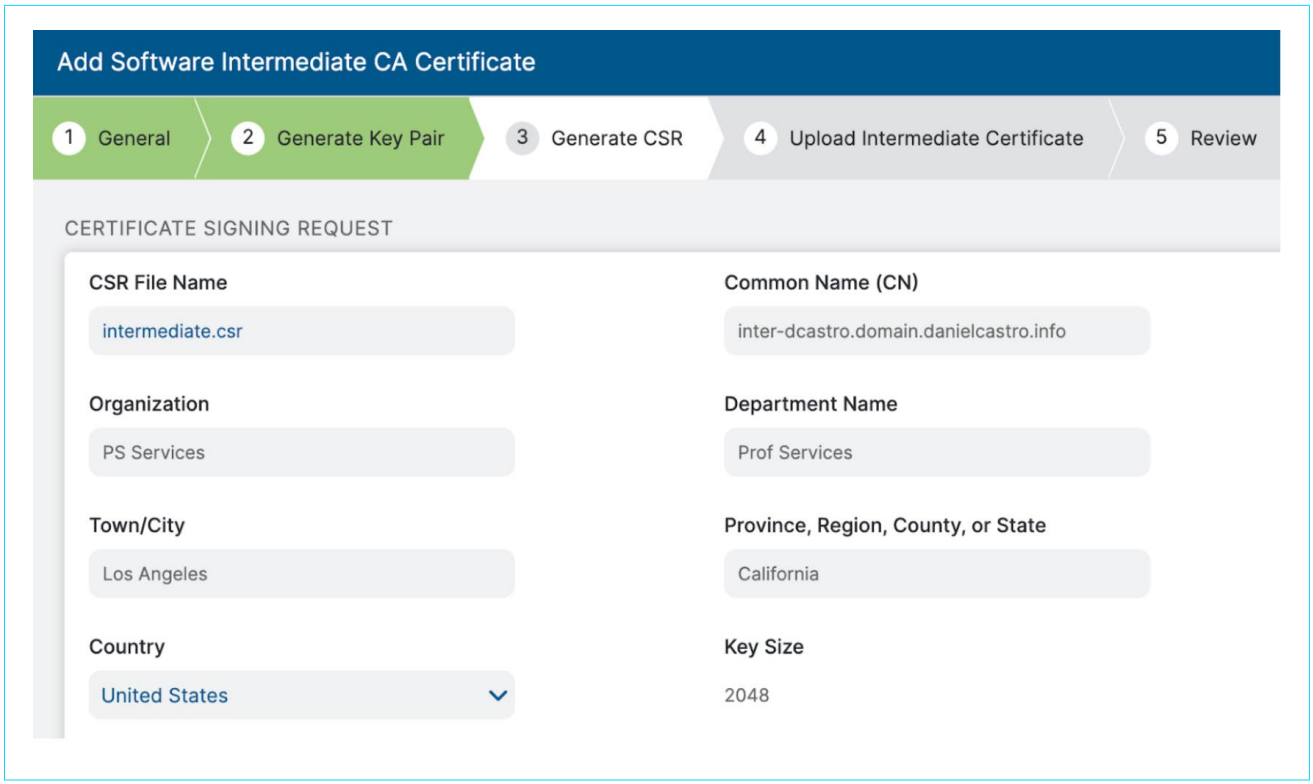


Proceed with the Private Key. This key is stored inside Zscaler and can not be exported.

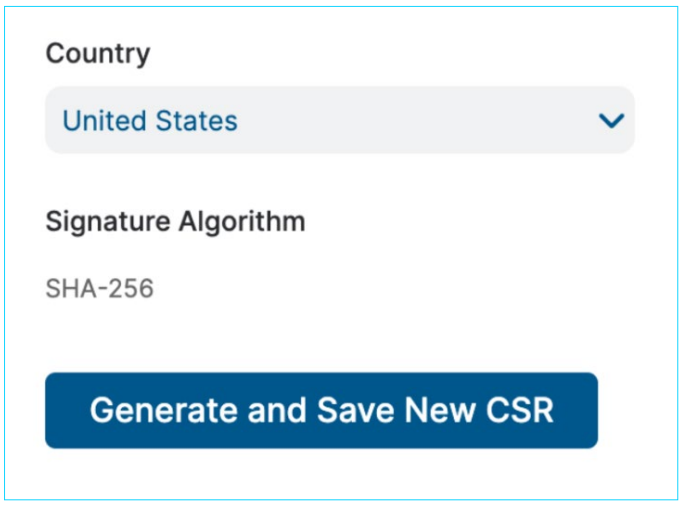
Proceed with the key creation:



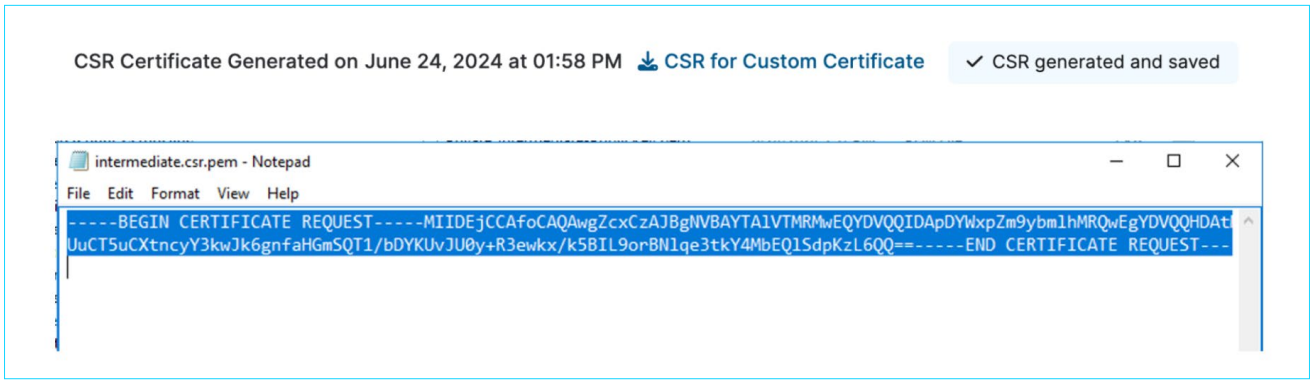
In the next step the CSR details must be filled in accordingly:



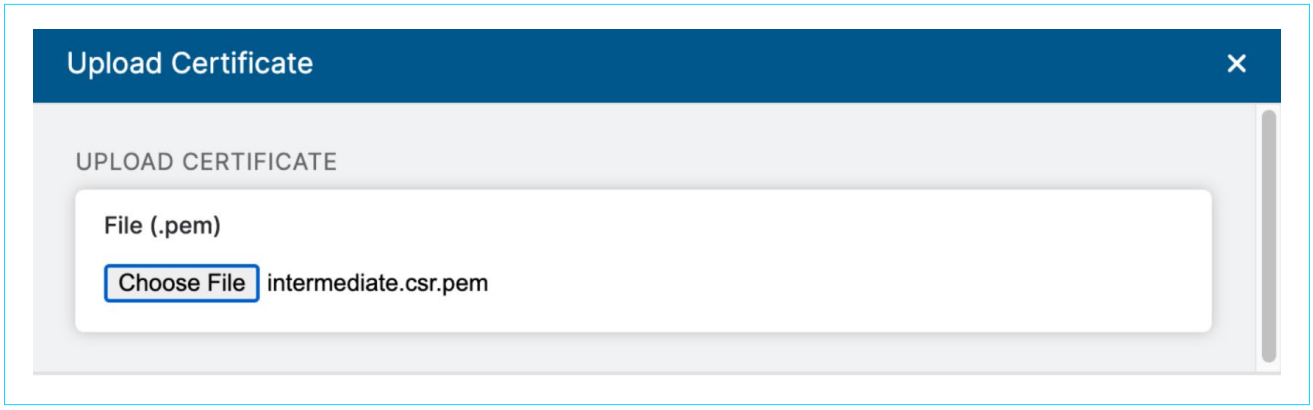
The unique details about your certificate need to be adjusted based on your organization. Finish with Generate and Save New CSR.



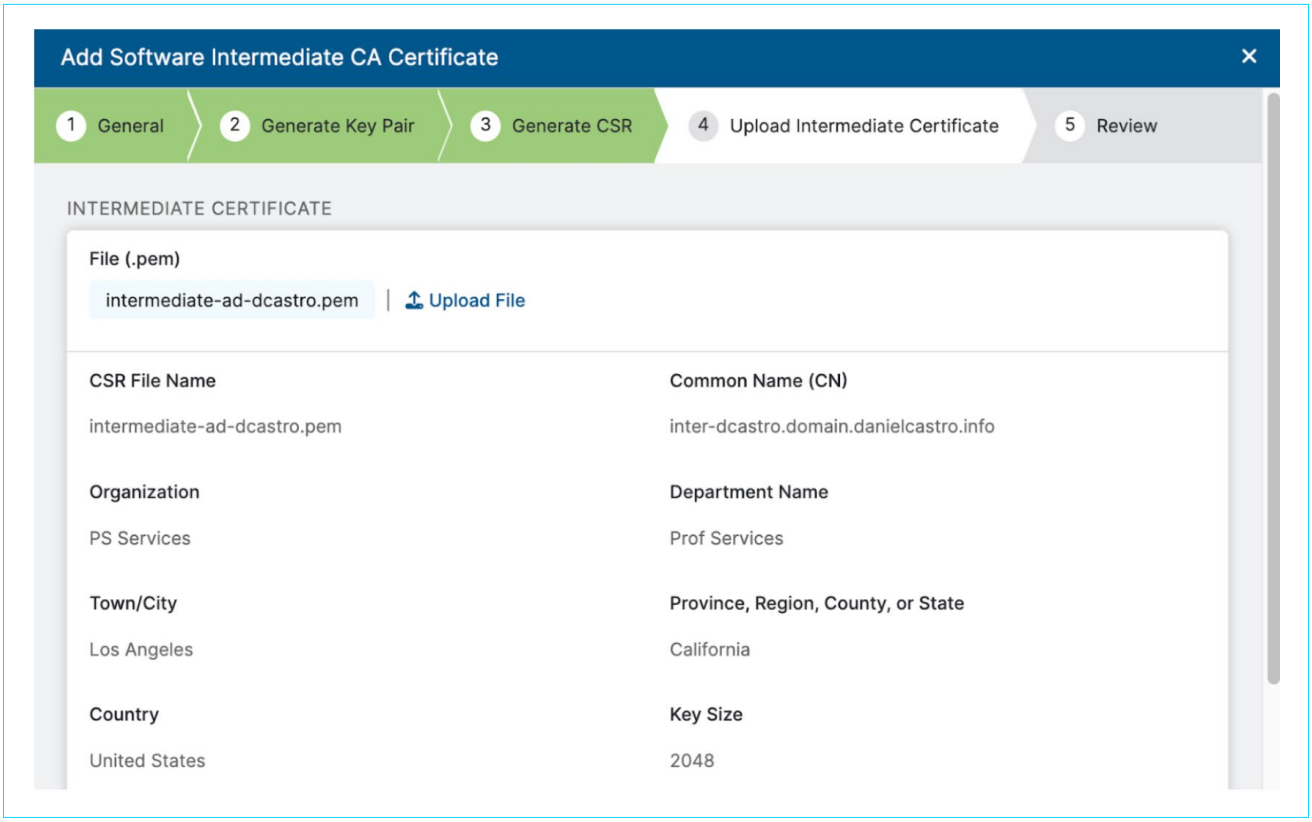
Proceed to download the file and open it in a text editor



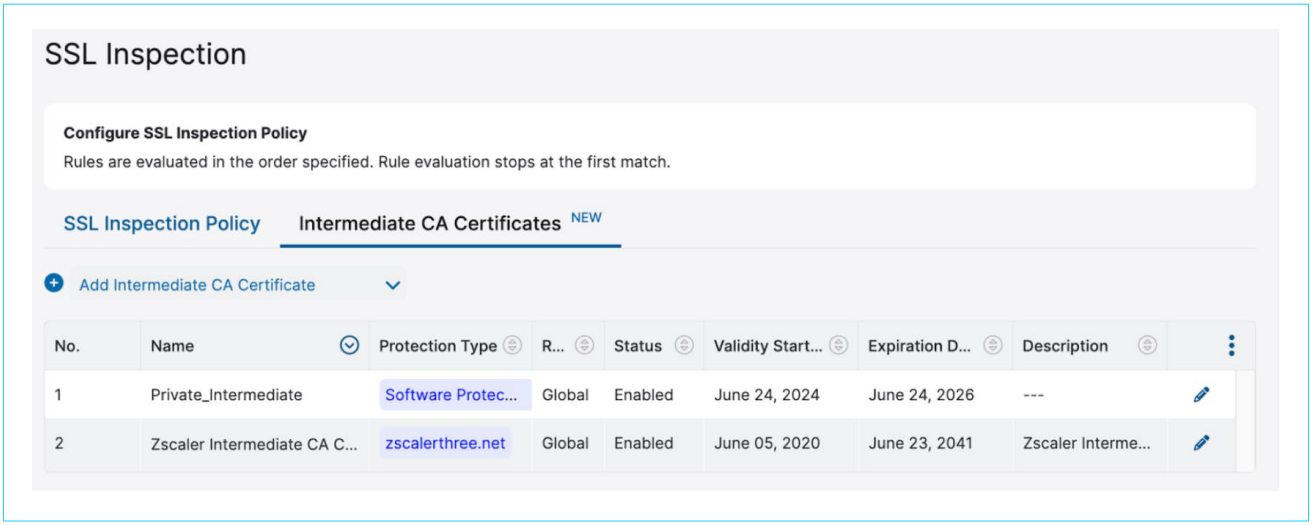
The CSR's text will be needed in the CA to request the certificate. The next section explains how to request the certificate from Active Directory Certificate Services (ADCS). Once you have the certificate make sure it is in PEM format and the filename ends with .pem and then proceed



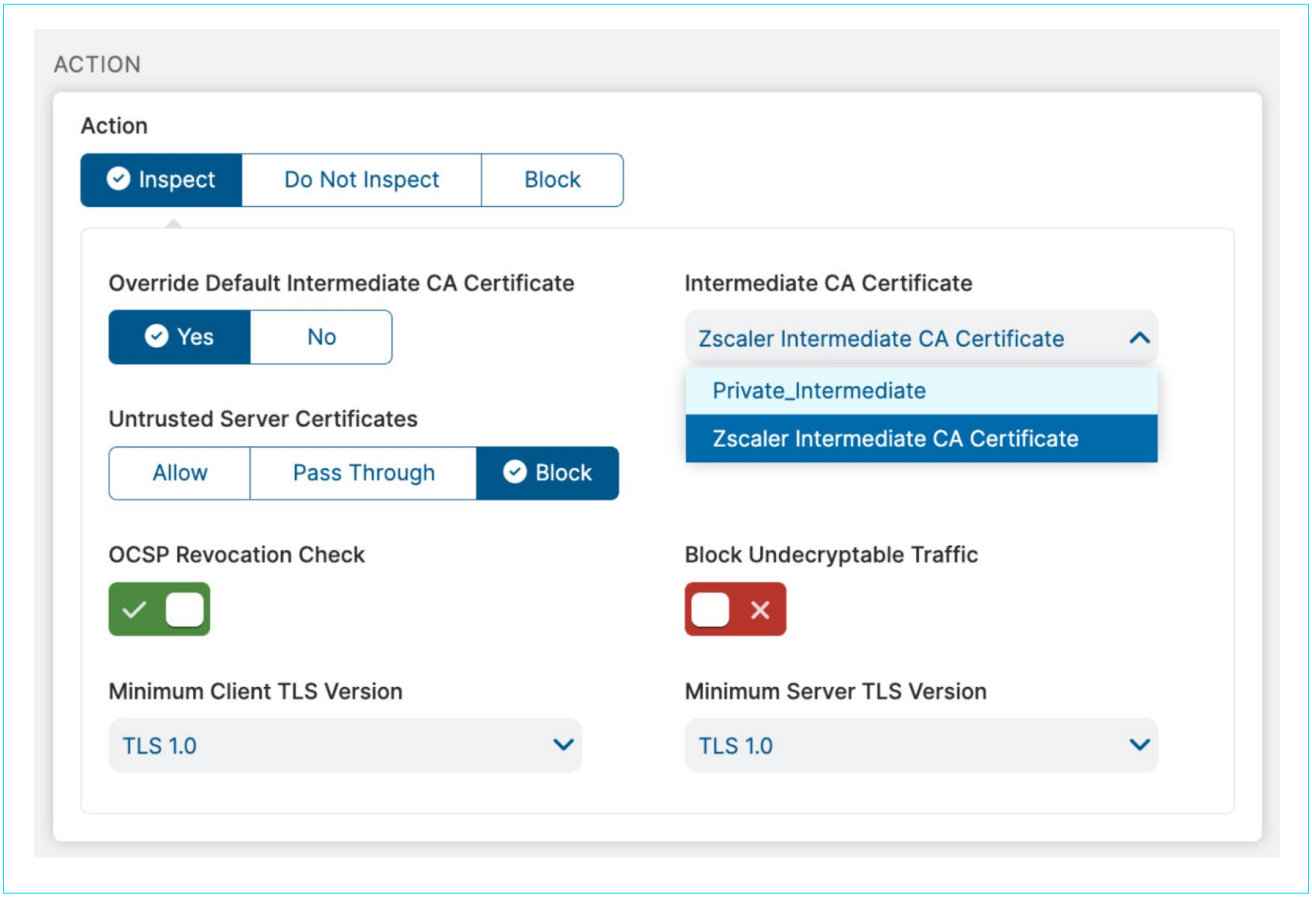
Upload and review



Once you finish the upload process you will see the certificate in the administration panel.



To make use of the certificate you must use it as a SSL inspection policy action. The following screenshot is from the “SSL Inspection Policy” in ZIA:



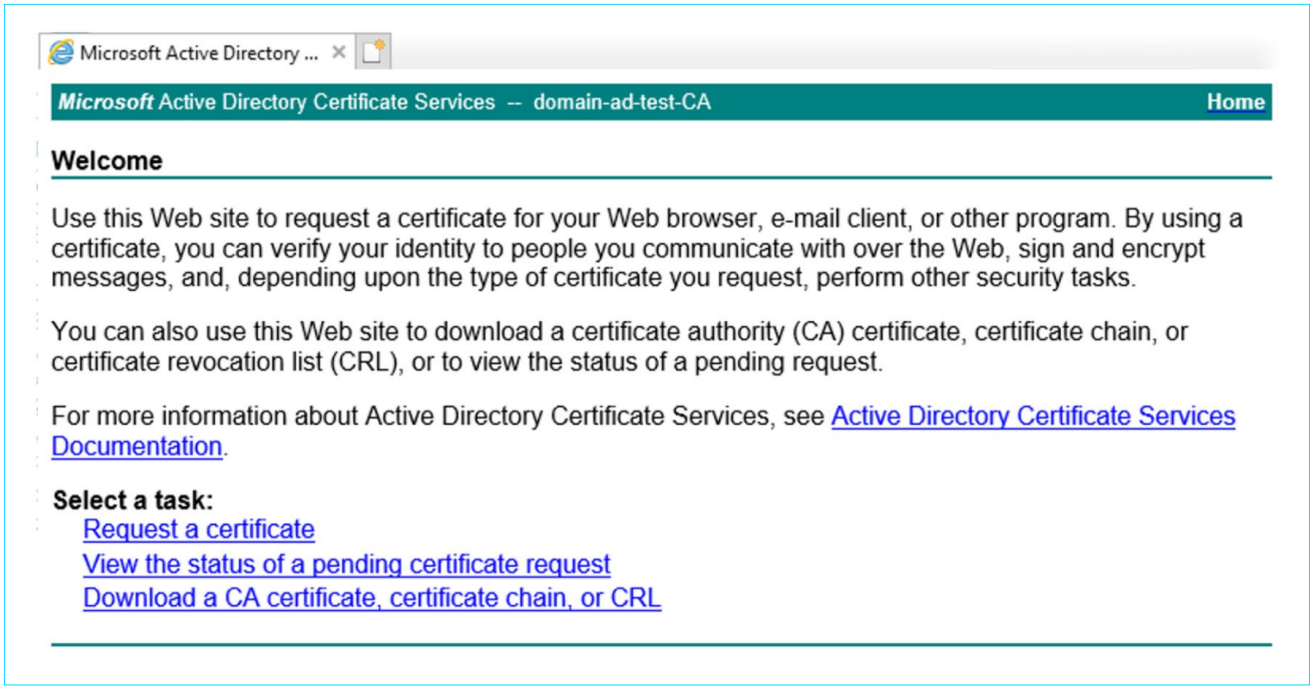
You can select now which Intermediate CA Certificate is used for SSL Interception.

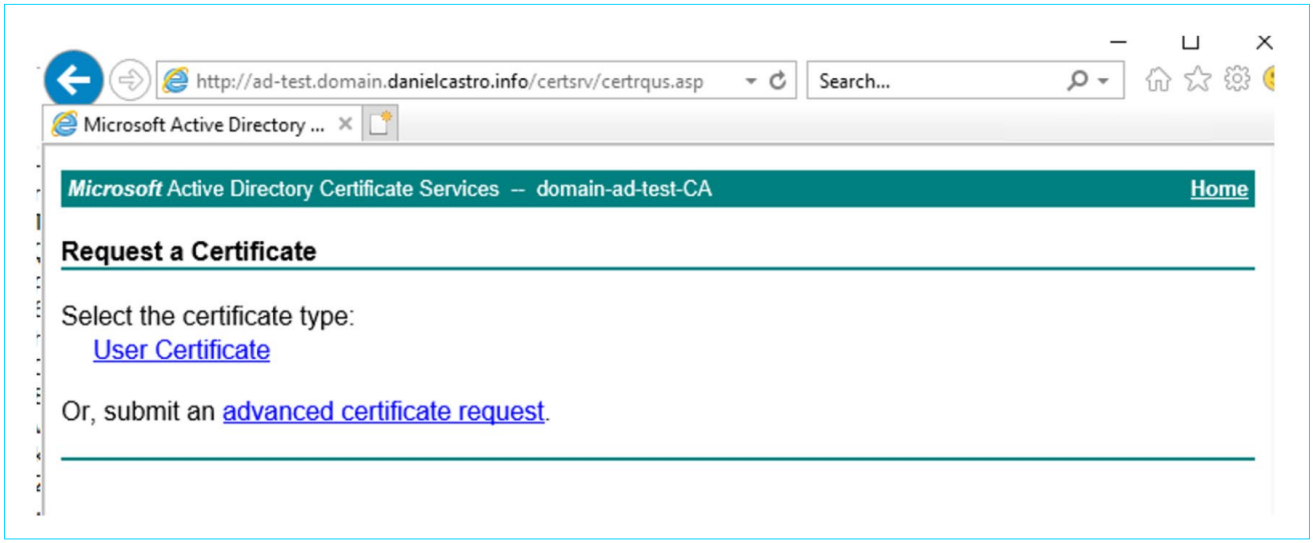
Create an Intermediate Certificate at Active Directory Certificate Services

Active Directory Certificate Services (ADCS) is a Windows Server role for issuing and managing public key infrastructure (PKI) certificates used in secure communication and authentication protocols. This is usually hosted on a MS Windows server and exposes a Web Server that allows anonymous and authenticated users to perform PKI operations on the Certificate Authority (CA). There can be multiple ADCS as part of a windows domain, where each ADCS operates over a part of the certificate chain. In other words there could be one ADCS for the root CA and multiple ADCS for subordinate intermediate CA that issue server certificates from those intermediate certificates.

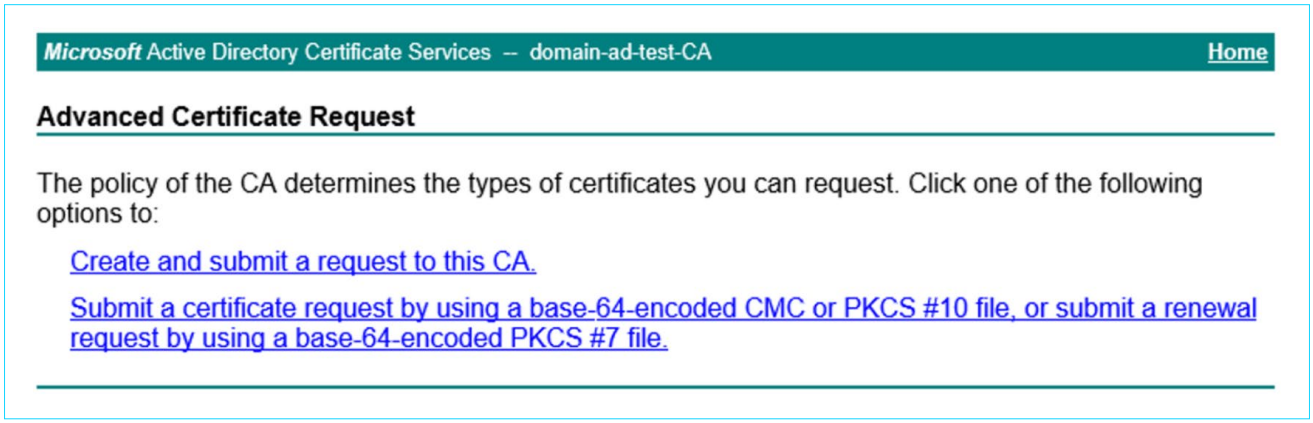
If you need instruction on how to install ADCS please see Windows Certificate Authority Installation Procedure. We will now show how to request an Intermediate Certificate. The steps for both ZIA and ZPA are very similar since the process starts by requesting a CSR from the service. Once you have the CSR file it can be used to sign a certificate at the CA.

Microsoft Certificate Enrollment Services allows the usage of a web form to submit the CSR and get the certificate.

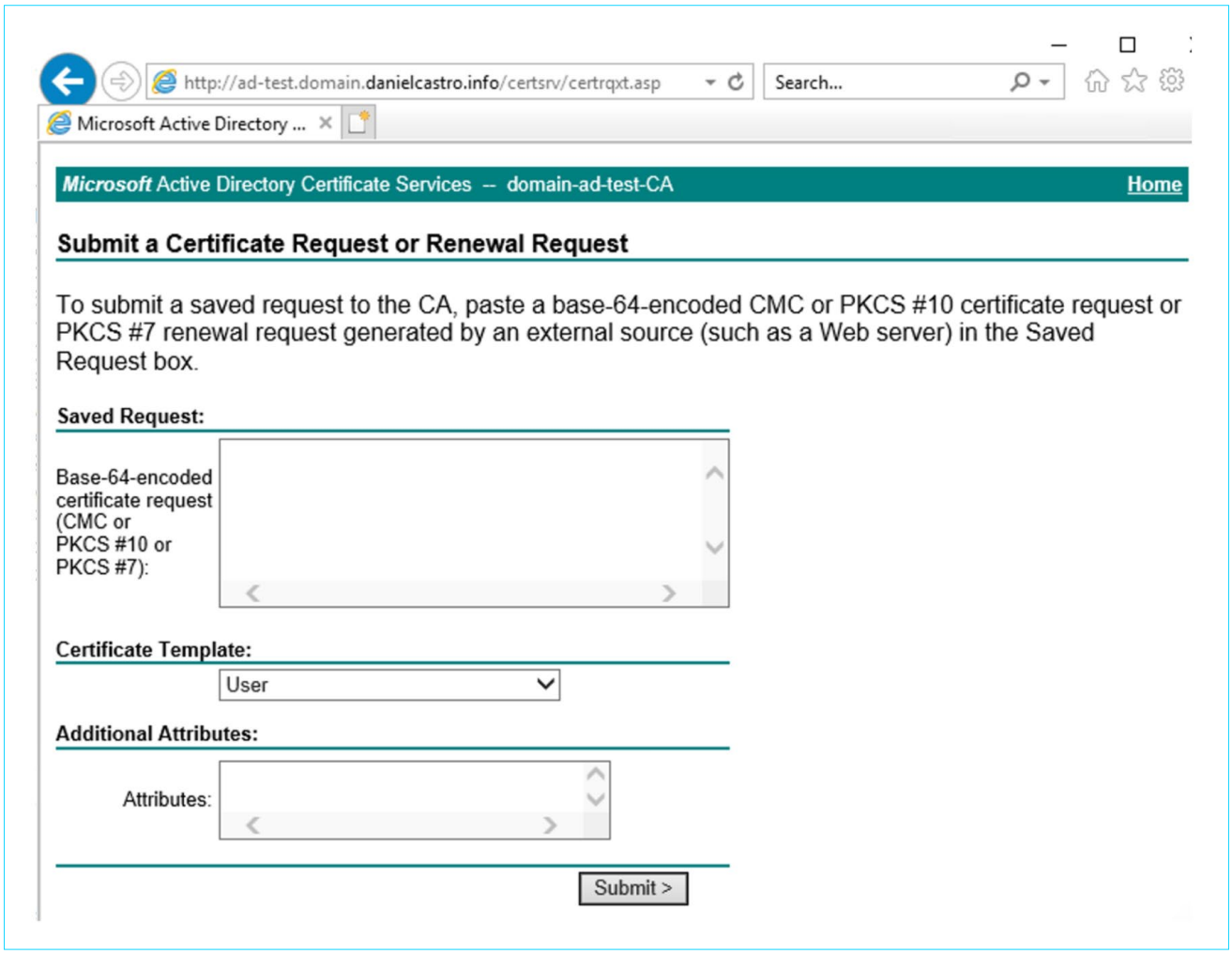




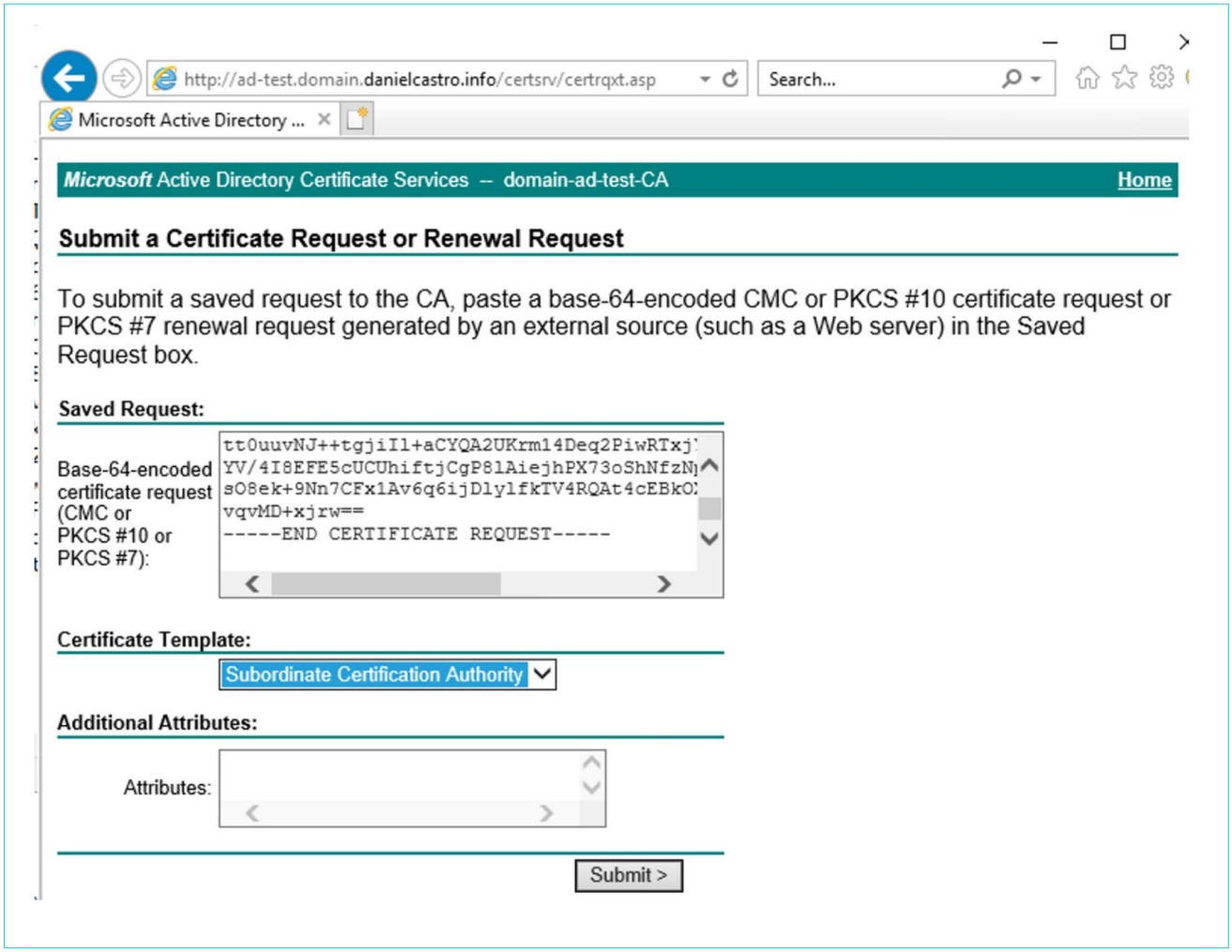
Select the advanced certificate request option.



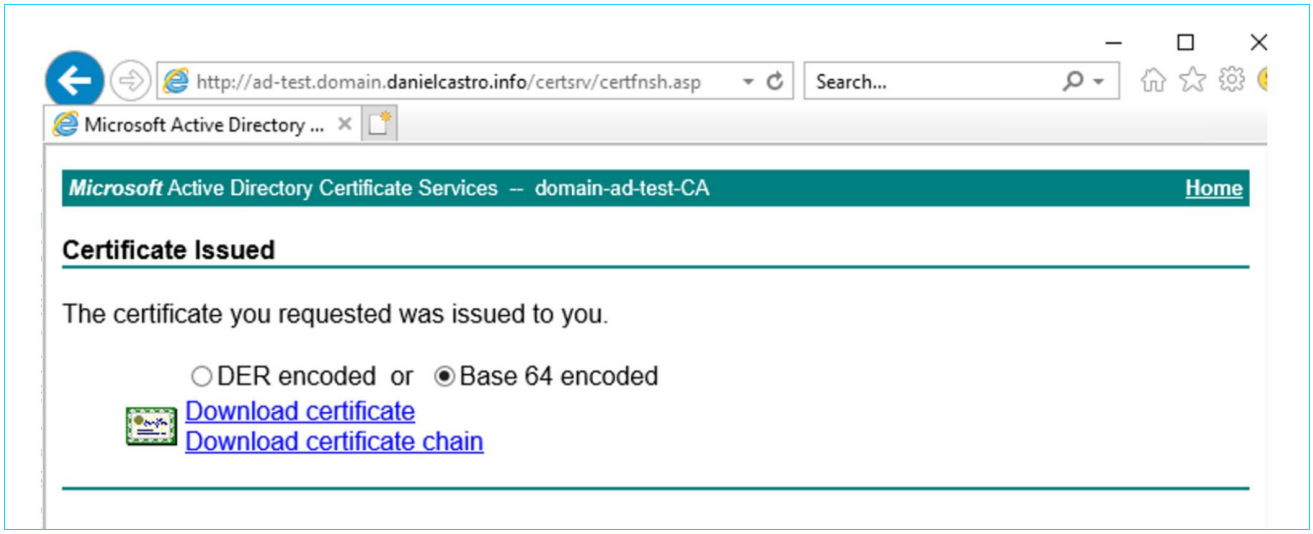
Use the second option to submit a certificate request.



Paste the contents of the CSR file into the “Saved Request:” section, change the “Certificate Template” to “Subordinate Certification Authority” and submit. It will look like this:



Once you submit the request, either the CA is configured to issue the certificate automatically or an administrator must approve the certificate and issue it. You then will login into Microsoft Certificate Enrollment Services and see your approved certificates and be able to download the issued certificate.



Be careful with the format you downloaded the certificate, as ZIA and ZPA only accept Base64 encoded PEM. If you see the certificate file name extension as p7b you used the wrong format, the correct filename ends with cer.

Create a Intermediate Certificate with OpenSSL

OpenSSL is a software library for applications that enables secure communications over computer networks. Among its features it allows the creation and verification of certificates and cryptographic keys. In this guide this will be used to create the root certificate that can be used to sign other certificates.

If you need instruction on how to create a CA using openssl please see OpenSSL Certificate Authority Creation. In the rest of this section we will show you how to create an intermediate certificate using openssl as CA.

In order to obtain an Intermediate certificate using openssl as CA you need to have defined the attributes in the configuration file. The CSR contains the "want" information for the future certificate, the configuration is the "possibilities" for the future certificate. When generating the certificate the CSR makes a request and the configuration determines if that is an attribute that can be used in the certificate.

```
openssl x509 -in from_zia.csr -out cert.pem -req -signkey root.key -days 1001 -config
rootCA_openssl.conf -key root.key -extensions v3_ca
```

The output of the command will be the certificate in Base64 encoded PEM, copy and paste the output into a file.

- `days` determine for how many days the certificate will be valid, it is paramount that the number of days is less than the days your root CA is valid, otherwise the resulting certificate will not be accepted by ZIA.
- `config` determines what and how openssl operates on the given command, the configuration file is used from creating the initial root certificate to any other certificate that the CA creates.
- `extensions` determine what attributes will be met for this particular certificate. The possible extensions are defined in the configuration file.