EAP-TLS

Applying client certificates to 84-Series handsets

Customers that opt to deploy EAP-TLS and use their own certificates for the client side authentication will need to follow this process to get the handset configured properly for the WLAN environment. This is only necessary when you are not using the phone's built in device certificate and installing the Spectralink PKI certificates into the customer's RADIUS environment to trust the device certificates.

Background

Server Hello

When doing any EAP method the phone will receive a Server Hello from the network which is generated by the RADIUS server performing authentication. This Server Hello contains the relevant certificates for the chain of authority specific to the RADIUS server. For example, if the customer has a Root CA and an Intermediate CA then this Server Hello should contain, in the following order, the RADIUS server's certificate; the Intermediate CA certificate; followed by the Root CA certificate. All three of these certificates will be embedded into the Server Hello message. Having the entire chain of authority is critical for allowing the phone to validate the server side certificates. It should also be noted that the RADIUS server will use what are known as CRL (Certificate Revocation Lists) whenever it is validating certificates. These lists are how your certificate infrastructure tells participants whether a certificate is valid or not and when it is expected to expire or if it was revoked early.

On the phone, you would load only the Root CA certificate as it will be able to validate the entire chain that the RADIUS server sent if the Root CA certificate it has matches the one the Server Hello includes. Once the phone can validate the Server Hello is valid we can move on to the Client Hello.

Troubleshooting

But first, let's consider what it means if the phone cannot validate the Server Hello certificates. The phone would return an error message that would likely read "Unknown CA". This should be an indication that you likely don't have the right Root CA certificate loaded onto the phone. Or, the RADIUS server is not sending the entire chain of authority in its Server Hello. Obviously if you know you've got the right Root CA certificate then the next logical place to look is going to be at the RADIUS server setup. You will typically need to add the entire chain of authority for the RADIUS server to function but in some older platforms you only needed to load the server's certificate. You had to tell the RADIUS server specifically to send the entire chain in its Server Hello. So, check you RADIUS server documentation for details on how this is supposed to be accomplished.

Client Hello

Installations that are using EAP-TLS, or some of its variants, the client will send a Client Hello after validating the Server Hello contents. When this happens, the client will be sending allow its own credentials in the form of a certificate. The client doesn't typically need to send the entire chain of authority like the RADIUS server does since the phone is not the source of authority. If you think of it from the perspective of a lock and key; the lock knows whether the key will work because it has all the necessary pins in place to fit the key that is inserted. The key wouldn't have any idea whether the lock it is trying to access matches its ridge pattern.

The client certificate that is sent to the RADIUS server for validate is generated via a process call CSR (Certificate Signing Request). We'll cover CSR generation later. The certificate gets applied to the phone via configuration files and includes a private key, public certificate combination. The two pieces together are what allows the phone to open the lock and validate the key the server sends in its attempt to validate the Client Hello. The RADIUS server will send a string that uses a hash generated using the public certificate that was signed by the CA. The phone has the private key that can decode the string that gets sent which is how the phone is able to confirm the server is permitted to make a connection.

Assuming the Client Hello validation process completes the phone and WLAN controller will exchange encryption keys inside the established SSL tunnel that will be used in all communications to come.

Troubleshooting

If you encounter problems at this step it may be difficult to tell. This is because of the multiple points of failure that can occur in the handshake process between the client and RADIUS server. It's quite possible that the failure happened in the Server Hello validation mentioned above. But, if you're sure that part is working your best bet will be to look at your WLAN controller and RADIUS server logs to see why they say the connection is failing. It could be that the RADIUS server doesn't have the proper chain of authority for the phone's client certificate. This might happen if you're using the phone's built in device certificate. Or, if the environment you're deploying into uses multiple Certificate Authority chains for different applications. In many large enterprise environments, this will be quite common. It's entirely possible that you would have a completely different intermediate CA for your WLAN than you would for your wired network. Or even for voice versus data clients.

The best tool to troubleshoot at this point will be a wireless capture kit. If you can sniff the packets between the phone and AP you will be able to see the Server Hello and Client Hello exchange happening in clear text over the air. This is extremely helpful as you can extract the certificates being used directly from the packet capture to compare them to what you're applying to the phone. Or, better yet, just extract the Root CA certificate from the Server Hello and apply that to the phone to ensure the Server Hello validation works. Then you can focus on the Client Hello validation. Troubleshooting the client side of the validate will require the RADIUS and WLAN logs.

CSR Generation

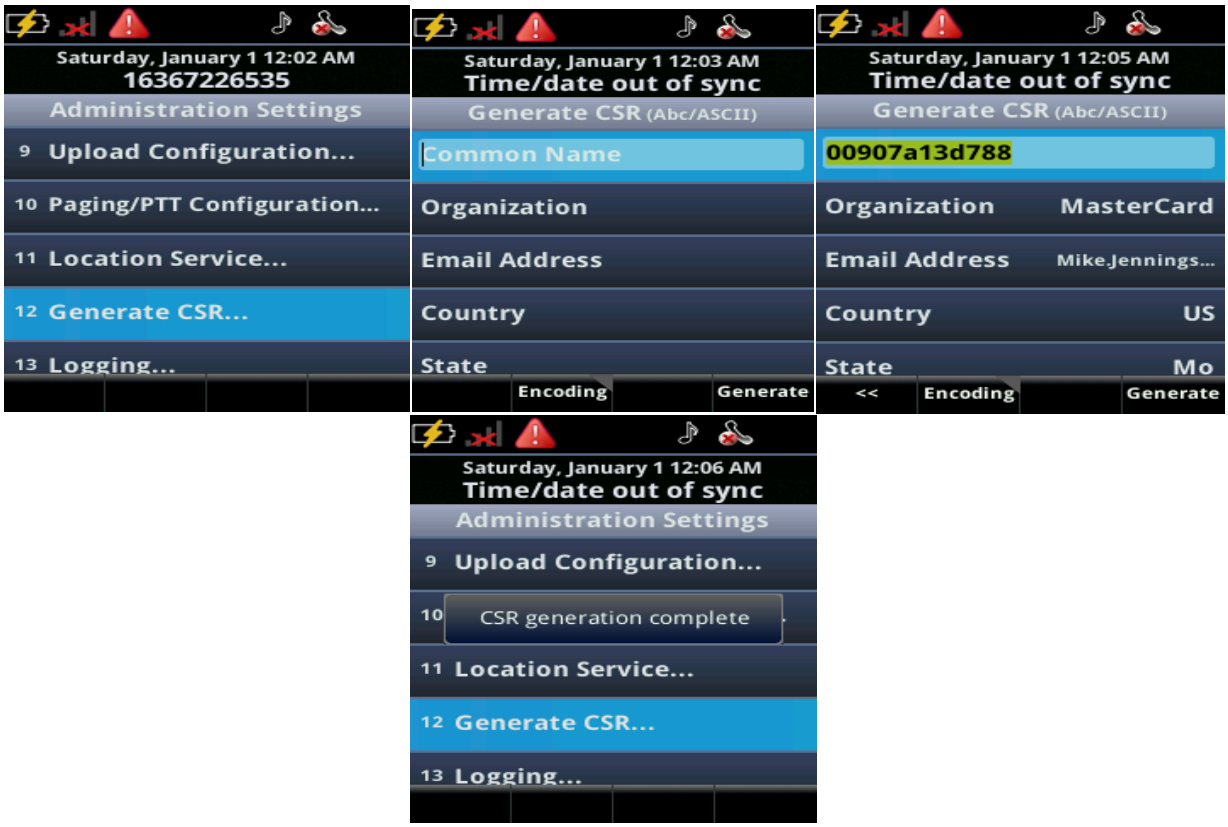
There are a couple of different ways to perform CSR generation for clients. With the 84-Series handsets you can generate a CSR directly from the handset. The only caveat though is that it must be done via the handset keypad. That's the only way! The other option would be use an external service, like your Windows environment, to generate multiple CSR's and sign them all at once. This is often more efficient but might make things a little more complicated too. Don't worry, I'll explain what that means later.

Any CSR contains two components. One is the CSR itself, which is what the Certificate Authority (CA) will use to create a valid certificate for the client. The other component is the private key. The private key is the client's equivalent of password that only it knows. For some tools that can generate a CSR there may also be a password that is tied to the private key but that's not something we need to worry about for our purposes.

Because the handset only allows CSR generation from the handset menus you will need to have access to each handset. The phone doesn't need to be configured at all yet so if it's fresh out of the box that

will be fine. You'll need to have either a QNC tool or a PC with a USB cable. If you go the PC route you'll also need to have an FTP server setup on the PC that is listening to the USB over Ethernet IP address. By default, this should be 169.254.1.1. The phone has a built in DHCP server that will provide the PC that IP address as soon as the driver installs upon connecting the USB cable between the phone and PC.

Once you've connected the phone you can start the CSR process on the phone. From the menus go to Settings -> Advanced Settings -> Administration Settings -> Generate CSR. The following series of images show the view of the phone's screens.



When filling in the form the Common Name will be the phone's MAC address. This is the only field that has any real importance but be sure to fill in each of the fields anyway. When you've completed all press the Generate soft key and wait a few moments for the "CSR generation complete" message to display. At this point the phone uploads two files. The first is the CSR which will have be named after the phone's MAC address. For example, 00907a13d788.csr. The other file is the private key file which will be named 00907a13d788-private.pem, with the MAC address being your phone's MAC address.

Just hang onto the private key file, you'll need that later when you build the phone's configuration file. For now, you need just need to use the CSR to generate the certificate for the phone. The specifics of how that's done are a bit outside the scope of this document. But it's important to know that the usage template for the certificate must include Client Authentication. Without that the certificate will be unable to be properly used. Once you've got your certificate generated make sure it's in a base64 format because we're going to need to paste it into a configuration file.

Troubleshooting

There's not much to troubleshoot in this step. The most likely issue will be with the phone uploading its CSR and Private Key files to the PC or QNC tool. If you're using the QNC tool, make sure you've got a USB drive attached to the QNC. The QNC will automatically put files onto the flash drive so you can take them off the drive and generate the certificate and use the private key in the configuration file setup. If you're using a PC to do this, you'll want to make sure your FTP server is listening to the right IP address. You'll also want to make sure the IP address assigned to the interface is correct. You may want to statically configure the IP address too once the phone is attached the driver loads. If you need the USB driver for the phone then download it from the Spectralink support website. Beyond that, if you're having trouble getting the files to the PC, look at the FTP server logs on your PC to see what's being sent or if there are any errors. You may need to verify that you've got the right username and password account setup on the FTP server for the phone. The phone's default username and password will be administrator/admin123 with most newer software releases. If it's got old software, you're going to need to update the phone before going too far anyway.

Configuration File Setup

The configuration file setup will be relatively basic in the end. Where it gets potentially confusing is with the structure of the configuration files. This is due to each phone needing a unique configuration file to allow you to apply the unique client certificate to each phone. Normally you would use a single configuration file for all phones to do the initial configuration. But this time each phone needs to be unique so you're going to need a unique file per phone.

There are some tricks we can use to make this a little easier and we can still leverage a global configuration file to handle shared parameters. In this section, I'll walk you through the setup and the parameters you'll need to get each phone setup with its unique certificate. We do assume here that you have a basic understanding of the XML configuration file structure used by the phone. If you're confused or otherwise unsure of how to do any of this then refer to the Deployment Guide for details on the basics.

Let's start by detailing the various parameters that you're going to need use to get the certificates loaded onto the phone. The other parameters you'll need are generic to any installation so we won't focus on those.

```
device.sec.TLS.customDeviceCert1.publicCert=""
device.sec.TLS.customDeviceCert1.privateKey=""
device.sec.TLS.customDeviceCert1.set="1"
device.sec.TLS.customCaCert1=""
device.sec.TLS.customCaCert1.set="1"
device.sec.TLS.customCaCert2=""
device.sec.TLS.customCaCert2.set="1"
device.sec.TLS.profile.CaCertList1="All"
device.sec.TLS.profile.CaCertList1.set="1"
device.sec.TLS.dot1x.strictCertCommonNameValidation="0"
device.sec.TLS.dot1x.strictCertCommonNameValidation.set="1"
device.sec.TLS.profileSelection.dot1x="PlatformProfile1"
```

```
device.sec.TLS.profileSelection.dot1x.set="1"
device.sec.TLS.profile.deviceCert1="Platform1"
device.sec.TLS.profile.deviceCert1.set="1"
```

We'll start out with the following three parameters that will go into the phone's *MACAddress*-config.cfg file.

```
device.sec.TLS.customDeviceCert1.publicCert=""
device.sec.TLS.customDeviceCert1.privateKey=""
device.sec.TLS.customDeviceCert1.set="1"
```

MACAddress-config.cfg File

device.sec.TLS.customDeviceCert1.publicCert

This parameter is only used when deploying with a device certificate that is generated inside the customer network. This means that you've either generated a CSR directly from the phone, as we described above, or you've used another service in your network to generate client certificates. The contents of this parameter will be the base64 contents of the certificate that is signed by the certificate authority in the customer's network. Make note of what we said there, the certificate must be in base64 format; otherwise it will not be possible to copy and paste the certificate into the configuration file.

What you'll have is something that looks like the following:

```
-----BEGIN CERTIFICATE-----
MIIHTCCBgWgAwIBAgITGwAEr0hC8vOR81VgzAAAAASvSDANBgkqhkiG9w0BAQsF
ADCBrjETMBEGCgmSJomT8ixkARkWA29yZzEaMBgGCgmSJomT8ixkARkWCm1hc3R1
cmNhcmQxCzAJBgNVBAYTAlVTMR0wGwYDVQQKExRNYXN0ZXJDYXJkIFdvcmxkd21k
ZTEkMCIGA1UECxMbR2xvYmFsIEluZm9ybWF0aW9uIFN1Y3VyaXR5MSkwJwYDVQQD
EyBNYXN0ZXJDYXJkIFBSRCBDb3Jwb3JhdGUgU3ViIENBMTAeFw0xNzExMDgxNzM0
/aeJ3/o5NXIAkwAvBLLuIP7WddnDWfyZASX6nuyDhnGDxEh0hcqHkZi5wznOUk2N
DMU+vk38MD1VqJh20hgFfWbTEBVX1PrHXi/YumLnzhfTEkGiatOg4/Am85w/SOH7
p/TkEZkXP0Wca55ijdX/acgejahDLL504jn/iHabJbrnDOQBs6uJzGQyyW31ncCH
c5sN0/UKbtMIpuTcBn+cDARm4UDYzzZxIqyLp6WQgJQqZPRxzVFaPp5GtN4nerRo
X4nJHvQ8VVbzj9F7obPhuUBdtKZL6vW6/Agw0/YIMiK//gTRKbWyxTgJZzaYCDnR
sQUnkC/TZxhlInI3hDgjPpA=
-----END CERTIFICATE-----
```

This has been shortened to fit the space as the real certificate is very long; it's a 2048-byte certificate. Which also means that your certificate length will vary. To apply this parameter, you will copy and paste the entire certificate, including the "-----BEGIN CERTIFICATE-----"and the "-----END CERTIFICATE-----"bits at the beginning and end of the certificate file. You'll end up with something that looks a bit like this in your configuration file:

```
device.sec.TLS.customDeviceCert1.publicCert="-----BEGIN CERTIFICATE-----
MIIIHTCCBgWgAwIBAgITGwAEr0hC8vOR8lVgzAAAAASvSDANBgkqhkiG9w0BAQsF
c5sN0/UKbtMIpuTcBn+cDARm4UDYzzZxIqyLp6WQgJQqZPRxzVFaPp5GtN4nerRo
```

X4nJHvQ8VVbzj9F7obPhuUBdtKZL6vW6/Agw0/YIMiK//gTRKbWyxTgJZzaYCDnR
sQUnkC/TZxhlInI3hDgjPpA=
-----END CERTIFICATE-----"

device.sec.TLS.customDeviceCert1.privateKey

Now we need to apply the private key to the configuration file. Your private key is generated along with your CSR as we discussed earlier. This file will already be in a base64 format when it is output and will have a file extension of ".pem", which is a good indication that it is already base64. Must like the public certificate above, you'll be pasting the private key directly into the configuration file. Your key will look something like this:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAx2kNuvsDc5V/wIKKjgrzFgVYX0TDrzghvJ0ZO5Vgl2K9wQQg
41Boxmf2NU0i79sJaE+VBgiS6XbFhC5PYOOz7EBDtdbTA+nZfUZoBv87pGyi5vGR
1G3yDF8UMnT2mExbgiLJ9mmMD+ywXf9WTAAg1ZKSKvpfqEzPQJnHg+2BeZ8LyPQG
Z7VET5I03Ve8vCTJCDWGkDMZXSbMKPxnPyyXIczIur4Q4LW9+6HCkJRmVukONdkd
hHL4v10IdVDtn8MatlvYiQGD1Yz+kD/o8q87U+YN6g39tb0mfwAodwn3+27/PvPr
8bcn3YWBWl9+cRA06Qc8aJQPPvEbT+Br3PiM5WaWiKhNonWzdriSa3nHQFacG3ov
i7OSO/NGAoEKgo0V0sKgfnsivVm+Dsnxy6ThTuRDCd9ZKuYOR4IDcfMPLQotqG6k
n5U6UwKBgQCZU2jPDJQBy8nG5PB9N5Ci/bX6FzeHQxSRYGP/2MJWEyNvBusi9bjI
5R2TnetOam7HRUR1x5r7aPKlgPxV3sIgUReL80VWedzFEltRXxrfRYQdfmYpN8v8
Iv+ky+D11x4p4B5PblYvInXGQ3bTNs6mrNS4VV+92p4nRB20cxk+jg==
-----END RSA PRIVATE KEY-----
```

Just like the public certificate, you'll be including the header and footer into the parameter. Copy and paste the entire key contents directly into parameter. This will look like the following:

device.sec.TLS.customDeviceCert1.privateKey="-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAx2kNuvsDc5V/wIKKjgrzFgVYX0TDrzghvJ0ZO5Vgl2K9wQQg
41Boxmf2NU0i79sJaE+VBgiS6XbFhC5PYOOz7EBDtdbTA+nZfUZoBv87pGyi5vGR
5R2TnetOam7HRUR1x5r7aPKlgPxV3sIgUReL80VWedzFEltRXxrfRYQdfmYpN8v8
Iv+ky+D11x4p4B5PblYvInXGQ3bTNs6mrNS4VV+92p4nRB20cxk+jg==
-----END RSA PRIVATE KEY-----"

This has been shortened like the public certificate above to anonymize the contents and to make them fit better into this document.

device.sec.TLS.customDeviceCert1.set

This parameter is an easy one as it's just necessary to tell the phone to parse the contents of the prior to parameters and apply them to the phone's configuration. This parameter is binary, 1 or 0, where 1 is enabled and 0 is disabled. You'll need to set this parameter to a value of 1 to allow the phone to handle the associated parameters.

device.sec.TLS.profile.deviceCert1

For this parameter, we will be specifying the certificate store the phone will use when performing Client Hello operations. This is very important to set if you are using certificates other than the built-in device certificate in the phone. If you leave this parameter to default it will only ever send the built-in device certificate when performing Client Hello messages. Since we've loaded the client certificate into the first client certificate slot in the phone we will be using "Platform1" as the value for this parameter. That will look like the following in your configuration file:

```
device.sec.TLS.profile.deviceCert1="Platform1"
```

device.sec.TLS.profile.deviceCert1.set

The last parameter for client authentication that we will add to the phone specific configuration file the ".set" for the profile selection parameter. As with the prior ".set" parameter, just set this to a value of "1" and let's move on.

MACAddress-config.cfg Complete

Now that we've gone through the various parameters for the phone specific configuration file let's show you what the result will look like all together.

```
device.sec.TLS.customDeviceCert1.publicCert="-----BEGIN CERTIFICATE-----
MIIIHTCCBgWgAwIBAgITGwAEr0hC8vOR8lVgzAAAAASvSDANBgkqhkiG9w0BAQsF
c5sN0/UKbtMIpuTcBn+cDARm4UDYzzZxIqyLp6WQgJQqZPRxzVFaPp5GtN4nerRo
X4nJHvQ8VVbzj9F7obPhuUBdtKZL6vW6/Agw0/YIMiK//gTRKbWyxTgJZzaYCDnR
sQUnkC/TZxhlInI3hDgjPpA=
-----END CERTIFICATE-----""
device.sec.TLS.customDeviceCert1.privateKey="-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAx2kNuvsDc5V/wIKKjgrzFgVYX0TDrzghvJ0ZO5Vgl2K9wQQg
41Boxmf2NU0i79sJaE+VBgiS6XbFhC5PYOOz7EBDtdbTA+nZfUZoBv87pGyi5vGR
5R2TnetOam7HRUR1x5r7aPKlgPxV3sIgUReL80VWedzFEltRXxrfRYQdfmYpN8v8
Iv+ky+D11x4p4B5PblYvInXGQ3bTNs6mrNS4VV+92p4nRB20cxk+jg==
-----END RSA PRIVATE KEY-----""
device.sec.TLS.customDeviceCert1.set="1"
device.sec.TLS.profile.deviceCert1="Platform1"
device.sec.TLS.profile.deviceCert1.set="1"
```

Obviously there's more to this file than just these parameters but this is what we're covering in this document.

Wireless.cfg File

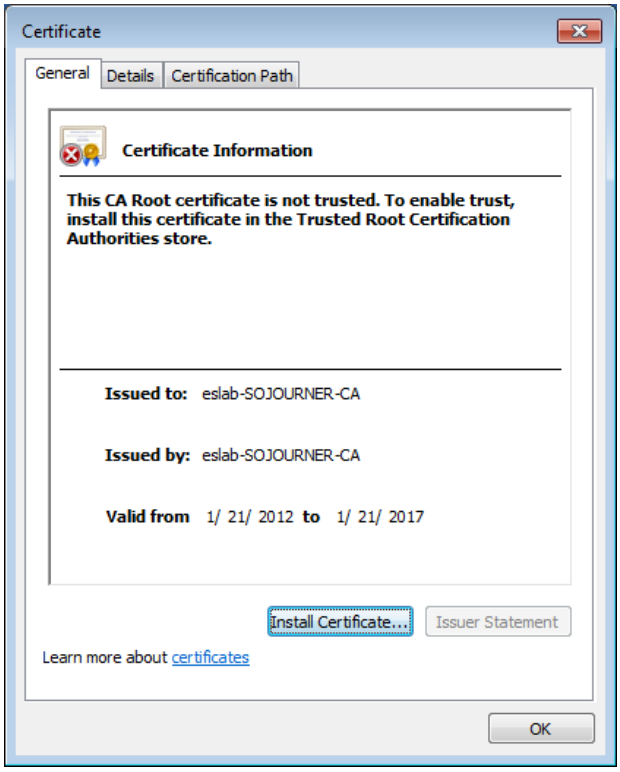
The wireless.cfg configuration file your initial provisioning file that contains all the global configuration settings for the phones. This means you will include your SSID, wireless security type details, radio settings and so on. Detailed information on this file can be found in the 84-Series Deployment Guide. For our purposes, we will only be focusing on the WPA2-Enterprise EAP-TLS required parameters.

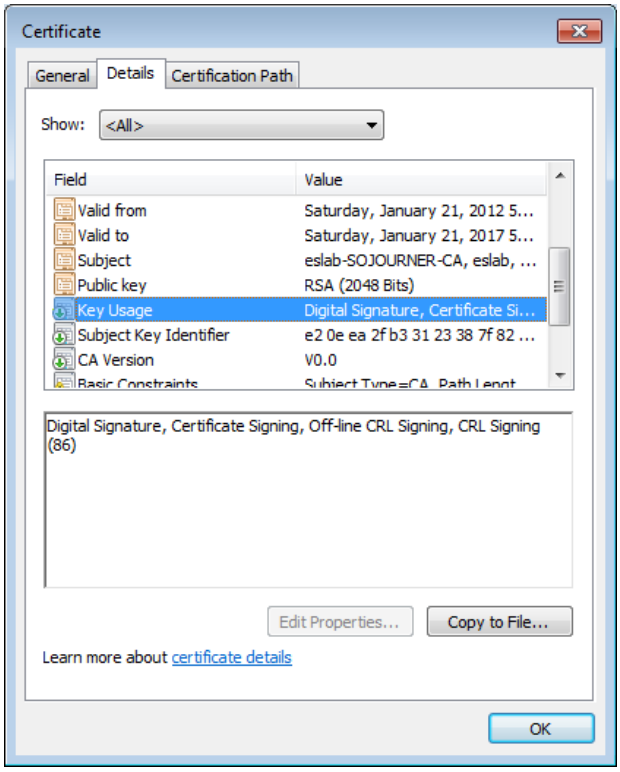
device.sec.TLS.customCaCert1

The first parameter defines the certificate that will be used for Server Hello validation in the first phase of the EAP-TLS handshake. This parameter should be the Root CA Certificate for the customer's

environment. It must be the same Root CA that has authority over the certificate signing chain that signed the RADIUS server's certificate. When the RADIUS server sends its Server Hello to the phone it will/should include this Root CA certificate. Since the phone only needs to validate that the server's certificate is really signed by who it says signed it, we only need the Root CA certificate to be installed on the phone.

Much like the public certificate and private key parameter above, the customer certificate parameter uses the contents of the Root CA certificate in base64 format. This will allow you to copy and paste the certificate directly into the configuration file for the phones. If you look at the Root CA certificate in Windows by double-clicking on the file, it will open in the Windows certificate snap-in. You'll know you've got the Root CA certificate if you see that it was "Issued to" and "Issued by" the same server. Root certificates are essentially self-signed certificates but they have the difference of special attributes being set that allow them to be used to sign other certificates, among other things. Here's an example of what a Root certificate might look like if you opened it in Windows:





Note that the Issued to and Issued by names are the same. If you click on the Details tab you can see a lot more details about the certificate. Scroll down through the list and find Key Usage and highlight that line. Then you'll see the different things that certificate be used for. The Certificate Signing is obviously the key one for our purposes and helps identify the certificate as a Root or Intermediate certificate.

The following is what this parameter will look like when placed into the wireless.cfg configuration file:

device.sec.TLS.customCaCert1="-----BEGIN CERTIFICATE-----
MIIIHzCCBgegAwIBAgIRAKZhU6cgfQIYbi6oS39GsuIwDQYJKoZIhvcNAQELBQAw
gbkxEzARBgoJkiaJk/IsZAEZFgNvcmcxGjAYBgoJkiaJk/IsZAEZFgptYXN0ZXJj

efk2Xo3tjpR1XPvXUL8dWLeu4cFatTCSLa5leNZGoc6S90FobUM3fyEJ9OJyO9X+
66zEIRivYa+ird6x7RvZ3dACRpaFW0+WZaWxg9B/g8UhrsKXJg4LSPlHnTb1J4r2
QHT2XJ3HuRvLSVNk66dnIbHGbA==
-----END CERTIFICATE-----"

This has also been shortened to fit into the document and to anonymize the contents of the certificate. We should also note that the phone has pre-loaded a list of all the major public certificate authorities. So if you're using certificates in your environment that have been issued through a public certificate authority you will likely not need to load any additional certificates into the phone for it to be able to validate your server certificate.

device.sec.TLS.customCaCert1.set

This parameter is another easy one as it's just necessary to tell the phone to parse the contents of the prior to parameters and apply them to the phone's configuration. This parameter is binary, 1 or 0, where 1 is enabled and 0 is disabled. You'll need to set this parameter to a value of 1 to allow the phone to handle the associated parameters. Set it to 1 and let's move on.

device.sec.TLS.dot1x.strictCertCommonNameValidation

Now we'll look at how the phone uses the certificates it has stored. When a certificate is validated the phone will compare many factors about the certificate. One of those is the Common Name of the certificate. Every certificate has a Common Name. For certificates issued to hosts it will usually be the hostname of the machine. For client certificates, it is usually the username of the person it is issued to or for devices, the device name or MAC address depending on how authentication will be handled. Common Name Validation is a process where the phone compares the Common Name stored in the certificate to the DNS hostname of the system that provided the certificate. This is only a problem when a certificate is issued to a host but the host sends its IP address instead of its hostname in communications. The most common scenario this happens in is with SIP call servers. They are often setup with only an IP address initially and certificates that get generated may not reflect that information as they may be created later.

We recommend that you set Common Name Validate to be disabled. It's really another security layer that can be used to help prevent man-in-the-middle type attacks. But in our case, it really doesn't add value to the process and can complicate things quite a bit when it comes to troubleshooting. Therefore, let's set this parameter to a value of 0.

device.sec.TLS.dot1x.strictCertCommonNameValidation.set

This parameter is yet another easy one as it's just necessary to tell the phone to parse the contents of the prior to parameters and apply them to the phone's configuration. This parameter is binary, 1 or 0, where 1 is enabled and 0 is disabled. You'll need to set this parameter to a value of 1 to allow the phone to handle the associated parameters.

device.sec.TLS.profileSelection.dot1x

The phone uses different profiles to determine which certificates or groups of certificates to use in certain applications. The default value for this parameter is to use Platform Profile 1 but I recommend setting this parameter to a value of "All" to ensure that every certificate in the phone's store gets checked when doing any dot1x authentication. Then if you have multiple certificates that you need to use to complete authentication the phone can leverage any of them.

device.sec.TLS.profileSelection.dot1x.set

And then we have one last ".set" parameter to apply. Each device parameter in the phone typically has a related ".set" parameter. This does vary a little bit but is generally true. If you're not sure whether you need a ".set" parameter you can check the Admin Guide or just add one. If the phone doesn't need it then it will just ignore it anyway. But in this case, we do need it so go ahead and set this to "1" and let's move on.

Wireless.cfg Complete

Let's go ahead and show you what this will all look like now that we've covered the details above. Don't forget that this is just specific to the EAP-TLS configuration and doesn't include all the other stuff that is part of the typical wireless.cfg configuration file.

```
device.sec.TLS.customCaCert1="-----BEGIN CERTIFICATE-----
MIIIHzCCBgegAwIBAgIRAKZhU6cgfQIYbi6oS39GsuIwDQYJKoZIhvcNAQELBQAw
gbkxEzARBgoJkiaJk/IsZAEZFgNvcmcxGjAYBgoJkiaJk/IsZAEZFgptYXN0ZXJj
66zEIRivYa+ird6x7RvZ3dACRpaFW0+WZaWxg9B/g8UhrsKXJg4LSPlHnTb1J4r2
QHT2XJ3HuRvLSVNk66dnIbHGbA==
-----END CERTIFICATE-----"
device.sec.TLS.customCaCert1.set="1"
device.sec.TLS.profile.CaCertList1="All"
device.sec.TLS.profile.CaCertList1.set="1"
device.sec.TLS.dot1x.strictCertCommonNameValidation="0"
device.sec.TLS.dot1x.strictCertCommonNameValidation.set="1"
device.sec.TLS.profileSelection.dot1x="PlatformProfile1"
device.sec.TLS.profileSelection.dot1x.set="1"
```

You can obtain all the template files from any software package from the Spectralink Support website. All software releases include a folder called Config that has the various templates available to you.

Provisioning Tips

Earlier we mentioned a trick that you can use to limit the number of configuration files required for each phone. Typically, you would create a MACAddress.cfg file for each phone which is the master configuration files for the phone. However, since the phone has some macros built into it let us leverage that functionality to make this more streamlined.

The macro we will leverage in the master configuration file is as follows:

[MACADDRESS]

This value, entered into the configuration file just as it is shown above will cause the phone to automatically put its own MAC address in place wherever it finds this value. This means that we can use this in the master configuration file as a kind of wildcard value. Here's that might look like if we use an all-zeros (0000000000000.cfg) configuration file:

```
<?xml version="1.0" standalone="yes"?>
<!-- Default Master SIP Configuration File-->
<APPLICATION
```

```
 APP_FILE_PATH="sip.ld"
 CONFIG_FILES="[MACADDRESS]-config.cfg, site.cfg, wireless.cfg"
 MISC_FILES=""
 LOG_FILE_DIRECTORY=""
 OVERRIDES_DIRECTORY=""
 CONTACTS_DIRECTORY=""
 LICENSE_DIRECTORY="">
</APPLICATION>
```

The CONFIG_FILES parameter is essentially the list of files the phone needs to download; an index of sorts. By telling the phone to download a file using this macro we can ensure that each phone gets its own unique configuration file along with the other global values in the other files listed.

Summary

This document covers, at a high level, the concepts around EAP-TLS and the various parameters and procedures necessary to load certificates onto the 84-Series handsets. EAP-TLS is a more and more common security type being used in Enterprises everywhere. While it has a very high management overhead when using certificates generated by the enterprise certificate authority; it also provides the greatest level of security over other more common EAP types like PEAP and EAP-FAST.

The troubleshooting tips included in this document are by no means comprehensive and should be used as a starting point for investigating issues in your own environment. It's quite valuable to have on hand a wireless packet capture adapter(s). At Spectralink we use AirPCap adapters most often but there are many alternatives available that are more cost effective for some customers. Wireless capture adapters can make a problem that seemed insurmountable suddenly become obvious and easy to resolve. At the very least, it will give you a must more comprehensive view of your environment and what's going on under the hood.

Some final thoughts, certificates often seem scary or complicated but just remember that in the end it's all about chain of authority. Think of any hierarchy, your own company for example, where you have someone that is in charge and has the authority to give someone else similar rights and authority. You have a CEO that has VP's that report to them and Directors that report to the VP's; Managers report the Directors and individual contributors (clients) report to Managers. Regardless of how many levels there are this is still just a chain of authority. Certificates aren't really any different to this analogy with Root CA's the equivalent of the CEO and Intermediate or Subs the equivalent of the VP's, Directors and Managers. In the end, your individual contributors are your clients just like they will be in your enterprise IT systems. So, don't overcomplicate certificates.

Oh, and let's not forget about certificate usage. That doesn't muddy the waters though if you think of usage as your departments in the company. Your VP of Operations might have a different role than your VP of Sales but they still both get their authority from the CEO. Their functional activities are different but how their authority is granted is still the same.

If you're still confused then ask your local IT team for insight into how your own certificate environment is structured to help you better understand. And remember, it's only complicated if you make it that way.

For more information about Spectralink products and product deployment, please visit the Spectralink Support Portal for the 84-Series Handset at: http://support.spectralink.com/products/wi-fi/spectralink-84-series-wireless-telephone

And be sure to let us know what you think about our documentation so we can continue to improve how to serve you!

Copyright Notice

Notice

Spectralink Corporation has prepared this document for use by Spectralink personnel and customers. The drawings and specifications contained herein are the property of Spectralink and shall be neither reproduced in whole or in part without the prior written approval of Spectralink, nor be implied to grant any license to make, use, or sell equipment manufactured in accordance herewith.

Spectralink reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult Spectralink to determine whether any such changes have been made.

NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE, OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY SPECTRALINK FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF SPECTRALINK WHATSOEVER.

Warranty

The *Product Warranty and Software License and Warranty* and other support documents are available at http://support.spectralink.com.

Contact Information

US Location
+1 800-775-5330

Spectralink Corporation
2560 55th Street
Boulder, CO 80301
USA

info@spectralink.com

Denmark Location
+45 7560 2850

Spectralink Europe ApS
Bygholm Soepark 21 E Stuen
8700 Horsens
Denmark

infoemea@spectralink.com

UK Location
+44 (0) 20 3284 1536

Spectralink Europe UK
329 Bracknell, Doncastle Road
Bracknell, Berkshire, RG12 8PE
United Kingdom

infoemea@spectralink.com