PacketCable™/CableHome™ CertBuilder User Guide

(Last update: 16 April, 2012)

IPfonix, Inc.
330 WCR 16½
Longmont
CO 80504-9467
USA

Tel: +1 303 682 2412
Fax: +1 781 240 0527

info@ipfonix.com

http://www.ipfonix.com

Table of Contents

1. Intended Audience

This document is designed to assist anyone wishing to generate X.509 digital certificates for use in a PacketCable and/or a CableHome network.

2. Supported certificates

The IPfonix, Inc. PacketCable/CableHome CertBuilder can generate the following certificates, all of which comply with the stated requirements in the latest release of the PacketCable security specification (which may be downloaded free of charge from www.packetcable.com) and the CableHome specification (which may be downloaded free of charge from www.cablehome.com):

- Border Proxy
- Cable Modem Termination System (CMTS)
- CableLabs Manufacturer Root
- CableLabs Service Provider Root
- Call Management Server (CMS)
- DF (Delivery Function)
- KDC (Key Distribution Center)
- Local System
- Manufacturer
- Media Gateway (MG)
- Media Gateway Controller (MGC)
- Media Player (MP)
- Media Player Controller (MPC)
- MTA Device
- MTA Manufacturer
- MTA Root (Multimedia Terminal Adapter Root)
- Provisioning Server
- PS Element (Portal Service Element)
- Record Keeping Server (RKS)
- Service Provider
- Signaling Gateway (SG)

3. Software Requirements

CertBuilder is currently available only for Windows-based systems. If you are interested in a version for Linux, please let us know and we will do our best to accommodate you.

4. Overview

CertBuilder has been designed to simply the automated generation of X.509 digital certificates for PacketCable and CableHome networks. It is a command-line program (no GUI!) whose execution and output is controlled by command-line parameters. While this is not as pretty as using a GUI, it is much easier to generate large numbers of certificates in a controlled manner by using a command-line interface, since this allows CertBuilder to be used within a script.

Most PacketCable/CableHome certificates may be generated from general-purpose certificate building programs. CertBuilder, however, also generates compliant certificates that are otherwise difficult to create (the KDC certificate in particular is somewhat specialized), and it also automatically checks each certificate against the requirements in the PacketCable and CableHome specifications after it has been created, to ensure that the certificate is truly compliant.

CertBuilder can be used either with pre-existing RSA keys, or it can generate RSA key pairs on the fly, saving them to disk so that they can then be used in other programs if necessary.

5. Installing the Software

The software comprises a single executable file, CERTBUILDER.EXE. Just place in a directory, and you are ready to go. (However, you had better read the rest of this User Guide first, since you may need to create some other files before CertBuilder will generate the particular kind(s) of certificate(s) that you are interested in.

6. Running the Software

You execute the CertBuilder program by moving to the directory in which the program is located, and then executing CERTBUILDER.EXE with a number of parameters.

Most parameters are of the form:

`-<cert><option>`

where `<cert>` determines the certificate type and `<option>` defines the option whose value is being set. Many parameters also require a value, which is always the value of the next field on the command line.

To make this clearer, here is a simple request to generate an MTA Root certificate:

```
CERTBUILDER -mr -mr# 99999
```

In this example:

1. The first parameter, `-mr`, instructs CertBuilder to construct an MTA Root certificate. Since “mr” is the abbreviation CertBuilder uses for an MTA Root certificate, this particular parameter contains only the `<cert>` part, without an `<option>` part.

2. The second parameter, `-mr#`, contains a `<cert>` part of "mr", which tells CertBuilder that this option is to be applied to an MTA Root certificate. The `<option>` part of the parameter is "#", which tells CertBuilder that the next value on the command line is to be interpreted as a serial number for the certificate.
3. The third parameter, 99999, is the value associated with the parameter `-mr#`. This tells CertBuilder that when it generates the MTA Root certificate, the certificate is to have the serial number 99999.

6.1. Global parameters

There is one optional global parameter, which is used to control the format in which any generated keys are written. By default, generated keys are written in the IPfonix, Inc. proprietary format described in section 8.1.1. However, if the parameter `-pkcs8` is present on the command line, then any generated private keys are also written in PKCS#8 format, in a file with the extension `.pkcs8`.

7. Values of `<cert>`

The permitted values of the `<cert>` portion of a parameter are given in Table 1 (next page).

Many of the certificates have two entries in this table, since the Local System certificate is optional and many certificates may be signed by either the Local System certificate or the Service Provider certificate[1].

Such certificates may be built using either of the forms `-xxx` or `-xxxls`. For example, in the case of a KDC certificate, the form `-kdc` tells CertBuilder to construct a KDC certificate signed by a Service Provider certificate, and the form `-kdcls` tells CertBuilder to construct a KDC certificate signed by a Local System certificate. All other parameters must always be of the form `-xxx<option>` (*i.e.*, there is, for example, no `-kdcls#` or `-mgcpri` parameter).

Note that the security specification uses the generic term "PacketCable Server Certificate" for certificates for the following devices:

- Border Proxy
- Cable Modem Termination System
- Call Management Server
- Media Gateway
- Media Gateway Controller
- Media Player
- Provisioning Server
- Record Keeping Server

[1] Throughout this document we use a convenient shorthand. When we say that certificate X "is signed by" certificate Y, it is to be understood to mean that certificate X is actually signed by the private key that corresponds to the public key encapsulated in certificate Y.

- Signaling Gateway

Table 1

Certificate	<cert>
Border Proxy (signed by Local System)	bpls
Border Proxy (signed by Service Provider)	bp
Cable Modem Termination System (signed by Local System)	cmtsls
Cable Modem Termination System (signed by Service Provider)	cmts
CableLabs Manufacturer Root	cmr
CableLabs Service Provider Root	cspr
Call Management Server (signed by Local System)	cmsls
Call Management Server (signed by Service Provider)	cms
Delivery Function (signed by Local System)	dfls
Delivery Function (signed by Service Provider)	df
KDC (signed by Local System)	kdcls
KDC (signed by Service Provider)	kdc
Local System	ls
Manufacturer	m
Media Gateway (signed by Local System)	mgls
Media Gateway (signed by Service Provider)	mg
Media Gateway Controller (signed by Local System)	mgcls
Media Gateway Controller (signed by Service Provider)	mgc
Media Player (signed by Local System)	mpls
Media Player (signed by Service Provider)	mp
Media Player Controller (signed by Local System)	mpcls
Media Player Controller (signed by Service Provider)	mp
MTA Device	md
MTA Manufacturer	mm
MTA Root	mr
Provisioning Server (signed by Local System)	provsrvrls
Provisioning Server (signed by Service Provider)	provsrvr
PS Element	pse
Record Keeping Server (signed by Local System)	rksls
Record Keeping Server (signed by Service Provider)	rks
Service Provider	sp
Signaling Gateway (signed by Local System)	sgls
Signaling Gateway (signed by Service Provider)	sg

8. Values of <option>

Table 2 shows the currently supported values of <option>, including the values of <cert> with which each particular option may be combined.

Table 2

<option>	Meaning	May be combined with <cert>	Meaning of following parameter	Mandatory or Optional	Comments
#	Create certificate with this serial number	all	Serial number	O	If not present, the certificate will be generated with the arbitrary serial number "12345".
namefile	Use this file to set the subject name	mm, md, m, pse, sp, ls, kdc, df, any PacketCable Server	Name of file that contains subject name	M	
pri	Use a pre-generated private key	all	Name of file that contains private key	O	If not present, CertBuilder will generate a suitable key pair. See section 7.1 for details regarding the supported formats
realm	Sets the realm name	kdc	Name of realm	M	
size	Sets the key size	md, pse, ls, kdc	Key length in bits	O	If the private key is not read from a pre-existing file, this sets the size of the keys generated by CertBuilder
subjectaltname	Add a subjectaltname extension	cms, cmts, mgc	Value of subjectaltname	M	This option does not apply to the KDC certificate, whose subjectaltname extension is automatically generated

8.1.pri

If using the pri value of <option>, a private key must be provided. The key format must be in one of the following formats:

- PKCS#8
- PKCS#1
- the simple proprietary format also understood by the IPfonix, Inc. range of KDCs (and documented below)

CertBuilder attempts to read the provided file in binary PKCS#8 format; if that fails, it will make another attempt, this time interpreting it as a file in binary PKCS#1 format. If that also fails, it will make a final attempt to read the file as a proprietary-format key.

8.1.1.Proprietary Format for Private Keys

This is a simple ASCII hex format that allows operators to create a reasonably user-friendly file that can easily be parsed by a human.

Each element of the private key is stored on a separate line, and each element is represented by a series of hexadecimal characters. The elements, in order, are:

modulus
public exponent
exponent
prime_0
prime_1
prime_exponent_0
prime_exponent_1
coefficient

So, for example, a private-key file in proprietary format might look like this:

```
a6 b0 5a c0 2a 6e 8d 36 3b 40 cf 86 c5 09 ef  ...  ee c7 53
01 00 01
6e 54 de cc c1 89 49 58 f3 21 73 fb bd a2 54  ...  e2 31 11
dd 77 92 67 58 20 df 24 da db 9f 1c 59 1e d9  ...  57 4a ab
c0 ae 2b 8d 19 05 00 2e 21 f1 1e 89 87 c2 88  ...  b5 75 f9
0b c0 ca be 3d 49 11 4e 8d 66 d6 5c d4 c5 f4  ...  53 61 9d
07 38 7f b9 51 ee b6 0b 04 8a 9c b2 5a bc 17  ...  a2 c3 41
75 0e f5 16 9f 6e 33 1a 44 f0 1f 8d 5a 22 18  ...  c8 c2 23
```

where the public exponent is the value 0x010001 and the value of the second prime is 0xc0ae2b...75f9. The ellipses represent hexadecimal pairs for which there is insufficient room on the page.

8.1.2.PKCS#8 format for Private Keys

Files that contain a PKCS#8 format private key should consist simply of a valid ASN.1 BER encoding of a PrivateKeyInfo object.

8.2.namefile

The namefile `<option>` identifies a file that contains the Relative Distinguished Name that is the subject of the certificate. The format of a namefile is simple: each line contains an ASCII listing of a parameter, a space, and the value of the parameter. For example, a namefile for an MTA Device certificate would look something like this:

```
C US
```

```
O IPfonix Pretend Telephony
OU PacketCable
CN 12:34:56:78:9A:BC
```

The order of the lines is respected by CertBuilder when it generates the certificate's subject name. In the above example, the subject will contain the following fields, in order: C, O, OU, CN. Values may contain spaces (as in the "O" field in the example).

An namefile for a KDC would typically look something like this for a system running without a Local System certificate:

```
C US
O Really Amazing Telephone Company
OU CableLabs Key Distribution Center
CN kdc1.ratco.com
```

A similar namefile for a KDC operating in a system running with a Local System certificate would look something like this:

```
C US
O Really Amazing Telephone Company
OU RATCo Denver
OU CableLabs Key Distribution Center
CN kdc1.ratco.com
```

9. Filenames

CertBuilder makes certain assumptions about the names of particular files, both files used as input, and those used for output. Table 3 lists the names of various files that hold certificates:

Table 3

Certificate	Filename
CableLabs Manufacturer Root	CableLabs_Manufacturer_Root.cer
CableLabs Service Provider Root	CableLabs_Service_Provider_Root.cer
KDC	KDC.cer
Local System	Local_System.cer
Manufacturer	Manufacturer.cer
MTA Device	MTA_Device.cer
MTA Manufacturer	MTA_Manufacturer.cer
MTA Root	MTA_Root.cer
PS Element	PS_Element.cer
Service Provider	Service_Provider.cer

Table 4 and lists the names of the various key files that may be output or may be used as input to sign a generated certificate.

Table 4

Certificate	Private key filename
Border Proxy	Border_Proxy_private_key
Cable Modem Termination System	Cable_Modem_Termination_System_private_key
CableLabs Manufacturer Root	CableLabs_Manufacturer_Root_private_key
CableLabs Service Provider Root	CableLabs_Service_Provider_Root_private_key
Call Management Server	Call_Management_Server_private_key
Delivery Function	Delivery_Function_private_key
KDC	KDC_private_key
Local System	Local_System_private_key
Manufacturer	Manufacturer_private_key
Media Gateway	Media_Gateway_private_key
Media Gateway Controller	Media_Gateway_Controller_private_key
Media Player	Media_Player_private_key
Media Player Controller	Media_Player_Controller_private_key
MTA Device	MTA_Device_private_key
MTA Manufacturer	MTA_Manufacturer_private_key
MTA Root	MTA_Root_private_key
Provisioning Server	Provisioning_Server_private_key
PS Element	PS_Element_private_key
Record Keeping Server	Record_Keeping_Server_private_key
Service Provider	Service_Provider_private_key
Signaling Gateway	Signaling_Gateway_private_key

Certificate	Public Key filename
Border Proxy	Border_Proxy_public_key
Cable Modem Termination System	Cable_Modem_Termination_System_public_key
CableLabs Manufacturer Root	CableLabs_Manufacturer_Root_public_key
CableLabs Service Provider Root	CableLabs_Service_Provider_Root_public_key
Call Management Server	Call_Management_Server_public_key
Delivery Function	Delivery_Function_public_key
KDC	KDC_public_key
Local System	Local_System_public_key
Manufacturer	Manufacturer_public_key
Media Gateway	Media_Gateway_public_key
Media Gateway Controller	Media_Gateway_Controller_public_key
Media Player	Media_Player_public_key
Media Player Controller	Media_Player_Controller_public_key

MTA Device	`MTA_Device_public_key`
MTA Manufacturer	`MTA_Manufacturer_public_key`
MTA Root	`MTA_Root_public_key`
Provisioning Server	`Provisioning_Server_public_key`
PS Element	`PS_Element_public_key`
Record Keeping Server	`Record_Keeping_Server_public_key`
Service Provider	`Service_Provider_public_key`
Signaling Gateway	`Signaling_Gateway_public_key`

So, for example, if CertBuilder generates an MTA Root certificate, the generated certificate will always be placed in a file called `MTA_Root.cer`.

As another example, if CertBuilder generates an MTA Manufacturer certificate, then it requires access to an MTA Root certificate and the private key that corresponds to that MTA Root certificate. It will look for these in the files `MTA_Root.cer` and `MTA_Root_private_key` respectively.

10.Validating certificates

CertBuilder will build certificates that match the information passed to it on the command line. Not all command lines and data files necessarily result in certificates that comply with the PacketCable and CableHome specifications; however, once a certificate has been generated, CertBuilder checks it for compliance with these specifications and reports if it finds that the certificate does not comply. *The most common cause of a non-compliant certificate is a namefile that does not comply with the requirements listed in the security specification.*

11.Creating invalid certificates

Sometimes, typically for the purpose of testing error-handling in devices, one desires to create certificates that are invalid in well-defined ways. There are several command-line options that may be used to create such certificates.

11.1. -badsignature

If the `-badsignature` parameter is included on the command line, the signature in the generated certificate will be invalid.

11.2. -expired

If the `-expired` parameter is included on the command line, the generated certificate will have an expiration date in the past. Because of the requirements of RFC 2459, the validity period of the certificate will also be reduced, so as to ensure that the start date is not prior to the year 2000.

11.3. -nosubjectaltname

Normally, KDC certificates are generated with a subjectAltName extension, as required by the specifications. However, if the `-nosubjectaltname` parameter is included on the command line, the generated certificate will *not* have a subjectAltName extension..

12.Examples

This section includes a number of examples of how to use CertBuilder to generate various kinds of certificates.

12.1.MTA Root

Although there is an official MTA Root certificate, it is expensive and slow to obtain a MTA Manufacturer certificate that has been signed by the official MTA Root certificate. Therefore, a vendor may wish to implement its own independent MTA hierarchy for the purpose of testing code prior to obtaining an MTA Manufacturer certificate signed by the official MTA Root.

```
CERTBUILDER –mr –mr# 1000
```

`-mr` tells CertBuilder to generate an MTA Root Certificate
`-mr# 1000` causes the serial number of the generated certificate to be 1000.

CertBuilder will automatically generate a key pair for this certificate, and put the public and private portions in the files `MTA_Root_public_key` and `MTA_Root_private_key` respectively.

12.2.MTA Device

Every MTA needs a unique MTA device certificate.

```
CERTBUILDER –md –mdnamefile MDDATA
```

`-md` tells CertBuilder to generate an MTA Device Certificate
`-mdnamefile MDDATA` tells CertBuilder to look in the file `MDDATA` for the subject name of the certificate. The contents of `MDDATA` will look something like the example in section 8.2; *i.e.*:

```
C US
O IPfonix Pretend Telephony
OU PacketCable
CN 12:34:56:78:9A:BC
```

The key pair will be generated automatically, and CertBuilder will expect to find a MTA Manufacturer certificate in `MTA_Manufacturer.cer`, and the corresponding private key in `MTA_Manufacturer_private_key`.

12.3.KDC

The KDC cert may be signed either by a Local System cert of a Service Provider cert. To build a KDC certificate signed by a Local System certificate:

```
CERTBUILDER -kdcls -kdc# 97531 -kdcnamefile KDCDATA
-kdcsize 1024 -kdcrealm IPFONIX.COM
```

`-kdcls` tells CertBuilder that the KDC certificate will be signed by a Local System certificate (which will be in `Local_System.cer`, with the corresponding private key in `Local_System_private_key`)
`-kdc# 97531` causes the generated certificate to have the serial number 97531
`-kdcnamefile KDCDATA` tells CertBuilder to look in the file `KDCDATA` for the subject name of the certificate. `KDCDATA` will look something like this:

```
C US
O IPfonix MSO
OU CableLabs Key Distribution Center
CN not.really.an.mso.ipfonix.com
```

`-kdcsize 1024` tells CertBuilder to generate a keypair of size 1024 bits
`-kdcrealm IPFONIX.COM` sets the realm of the generated certificate to be IPFONIX.COM.