

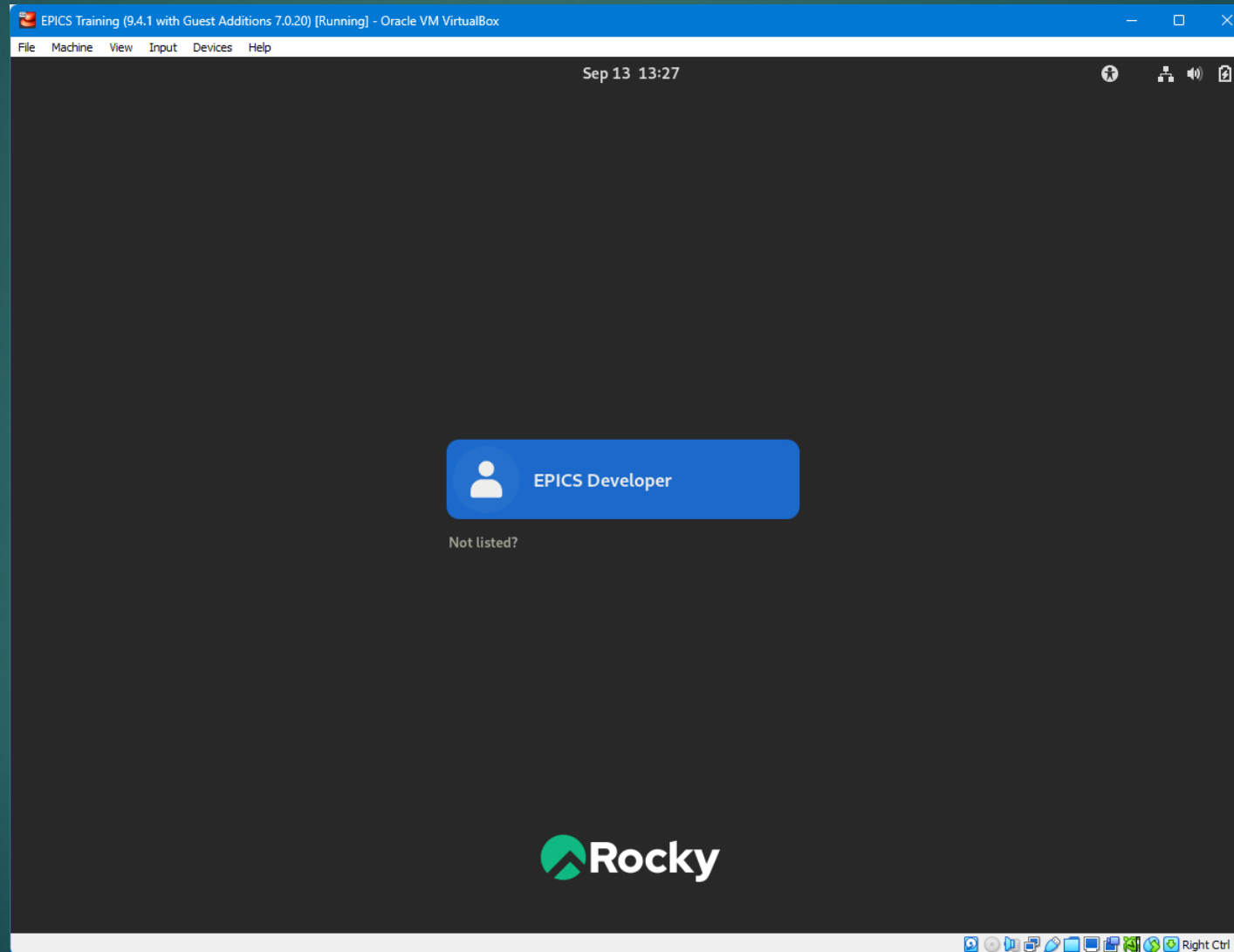
OPC UA Security

DEMO / HANDS-ON INSTRUCTIONS

Ralph Lange

Log Onto the Training-VM

2



Start the Demo Server

- ▶ Open a new terminal
- ▶ Unpack the Unified Automation demo server
- ▶ Run it
 - ▶ Use CTRL-C and restart it to simulate an orderly server disconnect and reconnect
 - ▶ Use CTRL-Z and "fg" to simulate network hang-up

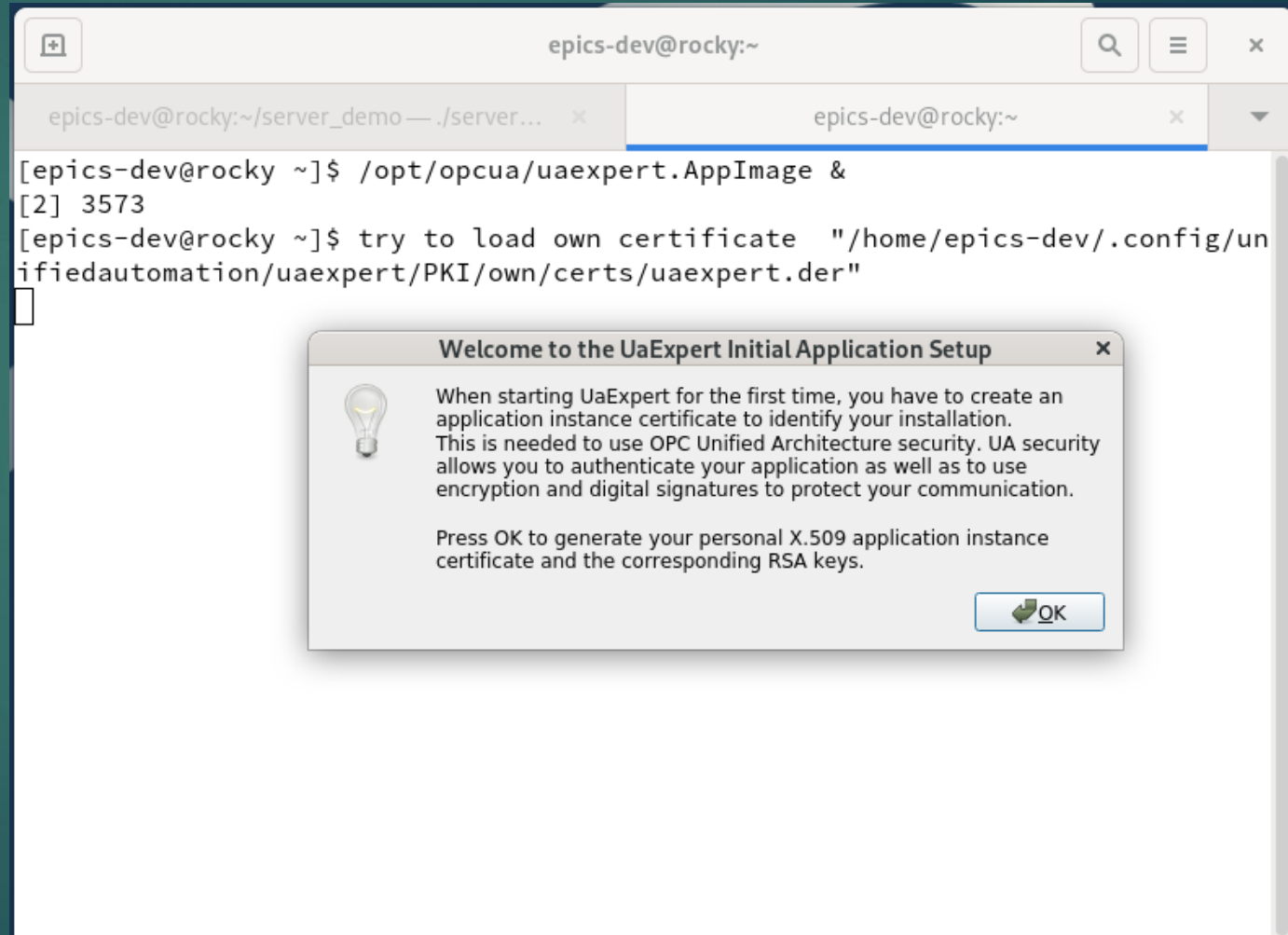
```
epics-dev@rocky:~/server_demo — ./server_cpp_demo -c config/ServerConfig.xml
[epics-dev@rocky ~]$ tar -xzf /opt/opcua/server_demo.tar.gz
[epics-dev@rocky ~]$ cd server_demo
[epics-dev@rocky server_demo]$ ./server_cpp_demo -c config/ServerConfig.xml
*****
Server opened endpoints for following URLs:
    opc.tcp://rocky:48010
*****
Press CTRL-C to shutdown server
*****

*****
The OPC UA server SDK is running in demo mode.
Communication will be stopped in 60 minutes.
*****
█
```

Start the UaExpert Browser

- ▶ The first start requires you to create a self-signed certificate.

Just enter any data.

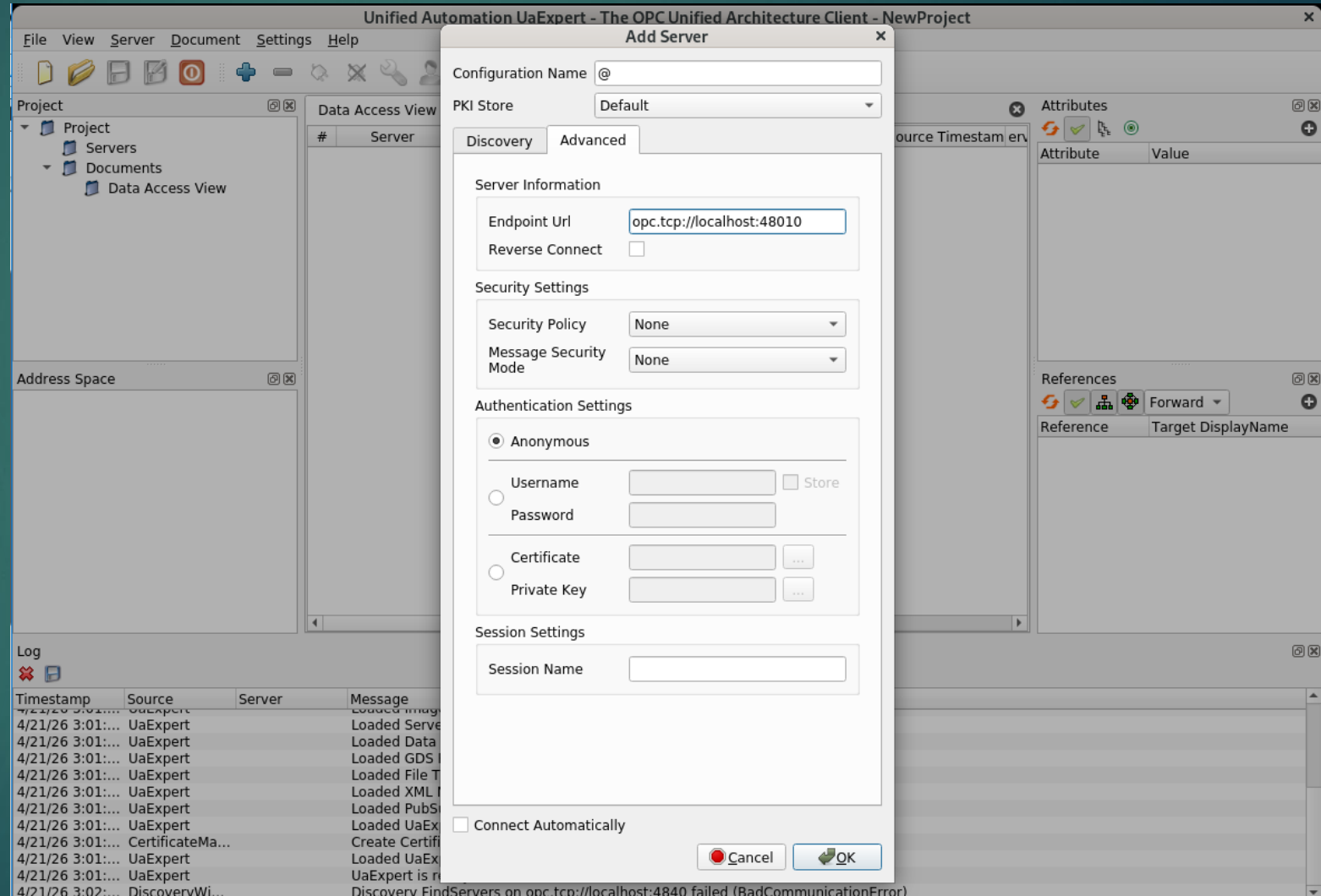


UaExpert: Add the Demo Server

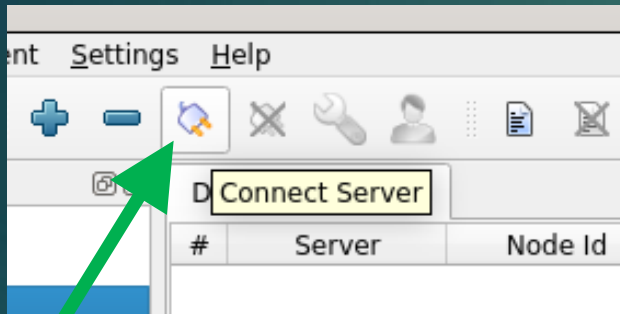
5

- ▶ Select Server, then Add...
- ▶ Use the *Advanced* tab and enter the endpoint URL

opc.tcp://localhost:48010



UaExpert: Connect



▶ Connect by clicking on the *Connect Server* icon

Validating the certificate of server 'UaServerCpp@rocky' returned an error:

BadCertificateUntrusted

Certificate Chain

Name	Trust Status
UaServerCpp@rocky	Untrusted

Certificate Details

Errors

Error	ok [BadCertificateUntrusted]
Subject	
Common Name	UaServerCpp@rocky
Organization	Organization
OrganizationUnit	Unit
Locality	LocationName
State	
Country	DE
DomainComponent	rocky
Issuer	
Common Name	UaServerCpp@rocky
Organization	Organization
OrganizationUnit	Unit
Locality	LocationName
State	
Country	DE
DomainComponent	rocky
Validity	
Valid From	Tue Apr 21 14:00:57 2026
Valid To	Sun Apr 20 14:00:57 2031
Info	
Serial Number	69E78319
Signature Algorithm	RSA-SHA256
Cipher Strength	RSA (2048 bit)
Thumbprint (SHA1)	7CFD273A21F63111E1D92DBC4B13CD76A984C57B

Trust Server Certificate

Accept the server certificate temporarily for this session

▶ Trust the unknown self-signed server certificate

▶ and *Continue*

UaExpert: Familiarize Yourself

7

Browse

Monitor

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	@	NS3 String ...	Boolean	false	Boolean	3:24:39.308 ...	3:24:39.308 ...	Good
2	@	NS3 String ...	Byte	80	Byte	3:24:39.639 ...	3:24:39.639 ...	Good

Attributes

Attribute	Value
NodeId	ns=3;s=Demo.Dynamic.Scalar.Byte
NamespaceIndex	3
IdentifierType	String
Identifier	Demo.Dynamic.Scalar.Byte
NodeClass	Variable
BrowseName	3, "Byte"
DisplayName	"" , "Byte"
Description	BadAttributeIdInvalid (0x80350000)
Value	
SourceTimestamp	4/21/26 3:12:48.936 PM
SourcePicoSeconds	0
ServerTimestamp	4/21/26 3:12:48.957 PM
ServerPicoSeconds	0
StatusCode	Good (0x00000000)
Value	22
DataType	Byte
NamespaceIndex	0

References

Reference	Target DisplayName
HasTypeDefi...	BaseDataVariableType

Log

Timestamp	Source	Server	Message
4/21/26 3:12:...	DA Plugin	@	Item [NS3 String Demo.Dynamic.Scalar.Boolean]: SamplingInterval=250, QueueSize=1, DiscardOldest=1, ClientHandle=1
4/21/26 3:12:...	DA Plugin	@	CreateMonitoredItems succeeded [ret = Good]
4/21/26 3:12:...	DA Plugin	@	Item [NS3 String Demo.Dynamic.Scalar.Boolean] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=1 [ret = Good]
4/21/26 3:12:...	Attribute Plugin	@	Read attributes of node 'NS3 String Demo.Dynamic.Scalar.Byte' succeeded [ret = Good].
4/21/26 3:12:...	Reference Pl...	@	Browse succeeded.
4/21/26 3:12:...	AddressSpac...	@	QascAddressSpaceModel::mimeData
4/21/26 3:12:...	DA Plugin	@	QascDaModel::dropMimeData
4/21/26 3:12:...	DA Plugin	@	Found existing subscription for ServerId 0
4/21/26 3:12:...	DA Plugin	@	Item [NS3 String Demo.Dynamic.Scalar.Byte]: SamplingInterval=250, QueueSize=1, DiscardOldest=1, ClientHandle=3
4/21/26 3:12:...	DA Plugin	@	CreateMonitoredItems succeeded [ret = Good]
4/21/26 3:12:...	DA Plugin	@	Item [NS3 String Demo.Dynamic.Scalar.Byte] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=2 [ret = Good]

Inspect

Build the IOC

Two Steps

1. Change directory into the application
`cd training/opcua-support`
2. Build the example IOC
`make`

```
[epics-dev@rocky ~]$ cd training/opcua-support
[epics-dev@rocky opcua-support]$ make
make -C ./configure install
make[1]: Entering directory '/home/epics-dev/training/opcua-support/configure'
perl -CSD /opt/epics/base-7.0.10/bin/linux-x86_64/makeMakefile.pl 0.linux-x86_64 ../..
mkdir -p 0.Common
make -C 0.linux-x86_64 -f ../Makefile TOP=../.. \
    T_A=linux-x86_64 install
make[2]: Entering directory '/home/epics-dev/training/opcua-support/configure/0.linux-x86_64'
perl -CSD /opt/epics/base-7.0.10/bin/linux-x86_64/convertRelease.pl checkRelease
EPICS/Release.pm: Undefined variable $(SUPPORT) used while checking module
    STREAM = /opt/epics/stream-2.8.26
make[2]: Leaving directory '/home/epics-dev/training/opcua-support/configure/0.linux-x86_64'
```

Run the IOC

- ▶ Change into the IOC's startup directory
- ▶ Run the IOC binary, giving the name of the startup file as argument
- ▶ Using a subshell avoids having to change directory all the time
(`cd iocBoot/iocUaDemoServer/; ../../bin/linux-x86_64/opcuaIoc st.cmd`)
- ▶ Always do a clean exit of the IOC, using **CTRL-D** or the **exit** command

OPC UA servers don't free their session resources when a connection breaks – if the same session is reconnected, it just continues

Run the IOC

10

```
[epics-dev@rocky opcua-support]$ ( cd iocBoot/iocUaDemoServer/; ../../bin/linux-x86_64/opcuaIoc st.cmd )
#!../../bin/linux-x86_64/opcuaIoc
## You may have to change opcuaIoc to something else
## everywhere it appears in this file
< envPaths
epicsEnvSet("IOC","iocUaDemoServer")
epicsEnvSet("TOP","/home/epics-dev/training/opcua-support")
epicsEnvSet("OPCUA","/opt/epics/opcua-0.11.2")
epicsEnvSet("ASYN","/opt/epics/asyn-4.44.2")
epicsEnvSet("CALC","/opt/epics/calc-3.7.5")
epicsEnvSet("P4P","/opt/epics/p4p-4.2.0")
epicsEnvSet("PVXS","/opt/epics/pvxs-1.5.1")
epicsEnvSet("SSCAN","/opt/epics/sscan-2.11.6")
epicsEnvSet("STREAM","/opt/epics/stream-2.8.26")
epicsEnvSet("EPICS_BASE","/opt/epics/base-7.0.10")
cd "/home/epics-dev/training/opcua-support"
## Register all support components
dbLoadDatabase "dbd/opcuaIoc.dbd"
opcuaIoc_registerRecordDeviceDriver pdbname
# Pretty minimal setup: one session with a 200ms subscription on top
opcuaSession OPC1 opc.tcp://localhost:48010
opcuaSubscription SUB1 OPC1 200
# Switch off security
opcuaOptions OPC1 sec-mode=None
```

Find out the IOC's URI

iocInit

Starting iocInit

#####

EPICS R7.0.10

Rev. 2026-04-13T11:16+0000

Rev. Date build date/time:

#####

OPC UA Client Device Support 0.11.2 (-); using Open62541 Client API v1.3.15

iocRun: All initialization complete

OPC UA: Autoconnecting sessions

OPC UA session OPC1: connected as 'Anonymous' with security level 0 (mode=None; policy=None)

OPC UA session OPC1: WARNING - this session uses *** NO SECURITY ***

epics> opcuaShowSecurity

Certificate store:

Server trusted certificates dir:

Server revocation list dir:

Issuer trusted certificates dir:

Issuer revocation list dir:

Rejected certificates are not saved.

ApplicationURI: urn:iocUaDemoServer@rocky:EPICS:IOC

No client certificate loaded.

Supported security policies: Aes128_Sha256_Rsa0aep Aes256_Sha256_RsaPss Basic128Rsa15 Basic256 Basic256Sha256 None

Prepare the **xca** Cert Manager

12

- ▶ Powerful free Certificate and Key Management tool
- ▶ Available on Linux, Windows, MacOS
- ▶ Supports many key and certificate formats (including ssh)
- ▶ Supports CA-based workflows (including sub-CA hierarchies)
Nice visualization of CA signature chains
- ▶ Allows to save and load templates
Great help in creating correct certificates efficiently
- ▶ Keeps keys, certs and templates in a database

Create the xca Database

13

- ▶ Start

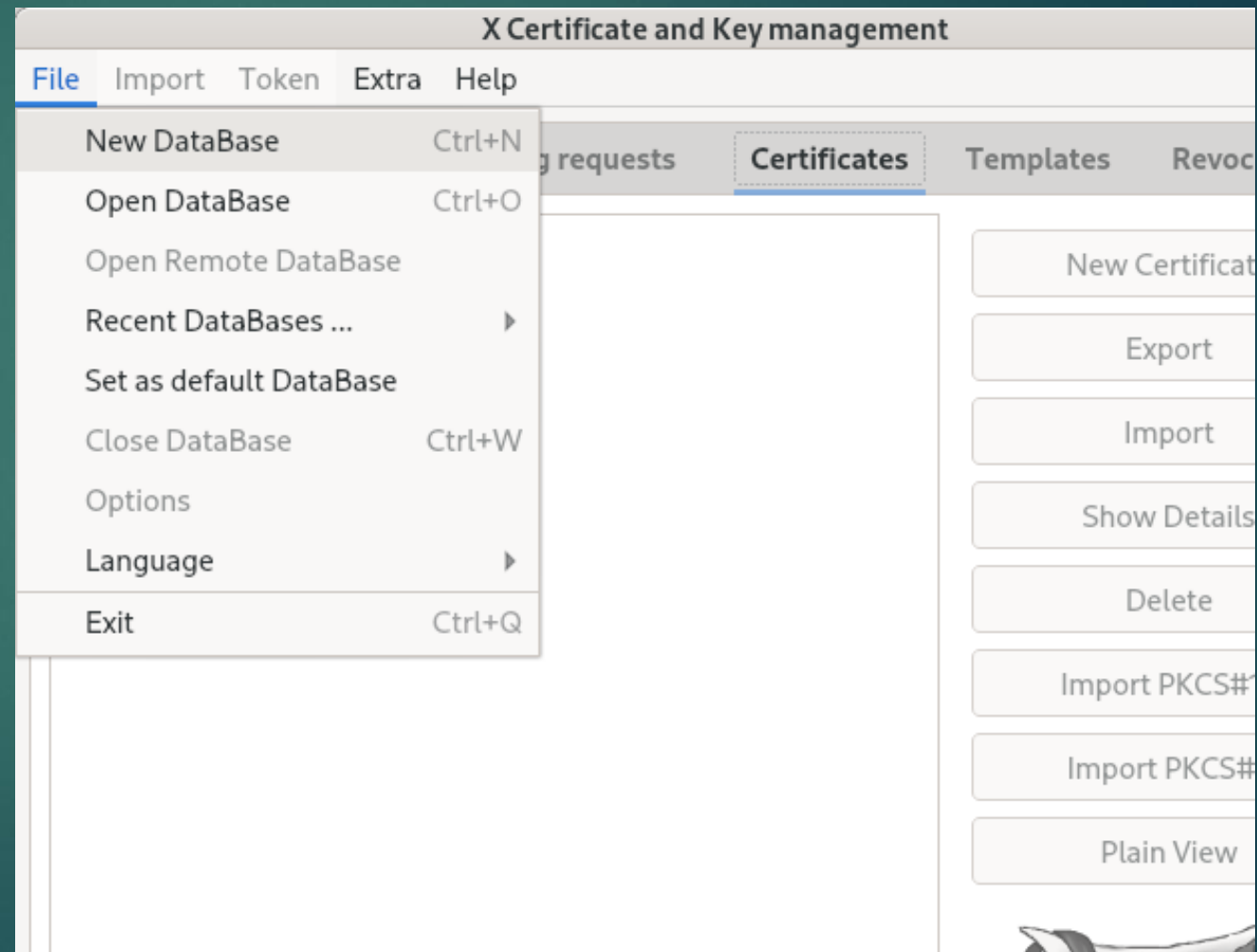
```
$ xca &
```

- ▶ Then

File -> New DataBase

- ▶ Give it a name

- ▶ You can leave it without a password

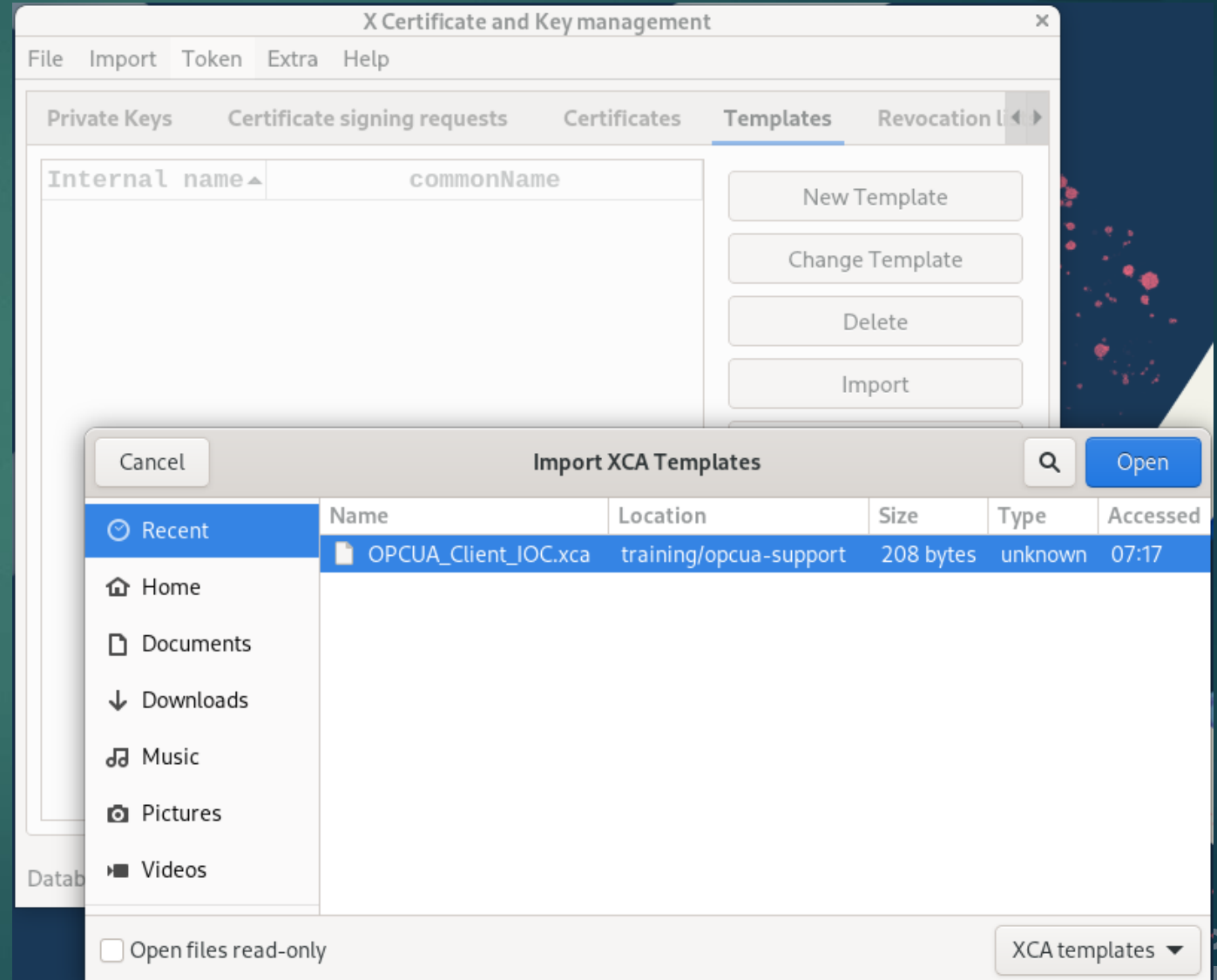


Import the IOC Cert Template

14

- ▶ Tab **Templates**
- ▶ Button **Import**
- ▶ Select the template **OPCUA_Client_IOC**

This loads the template from the file into the xca database



Prepare the Secure Connection

15

Set up the Client Certificate

- ▶ Create the Key Pair and self-signed Client Certificate for the IOC
- ▶ `st.cmd`: switch the IOC to use security
- ▶ `st.cmd`: load the self-signed Client Certificate and its private key

Establish trust

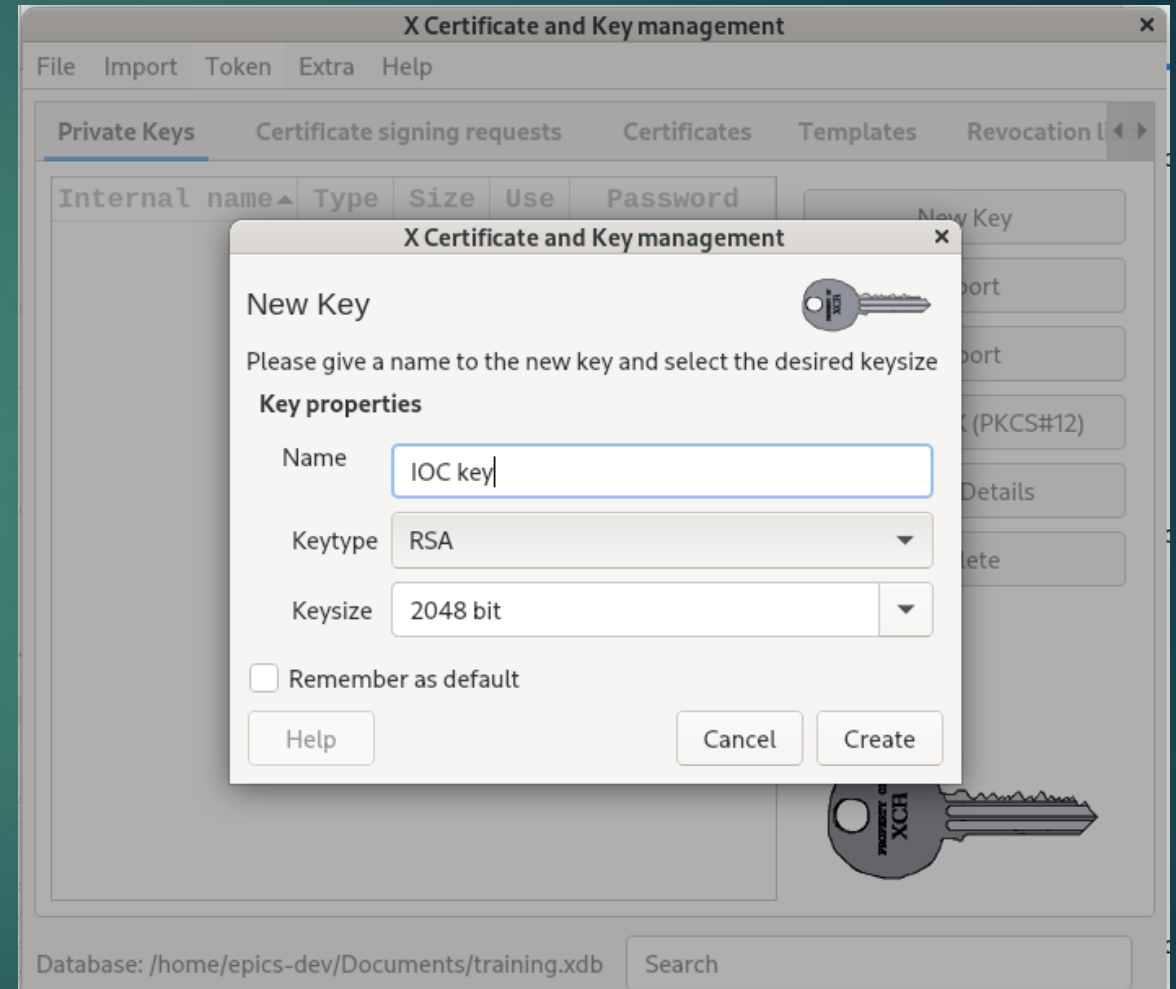
- ▶ Copy the Client Certificate into the server's **trusted/certs** list
- ▶ Copy the Server Certificate into the IOC's **trusted/certs** list

Connect!

Create a Key Pair for the IOC

16

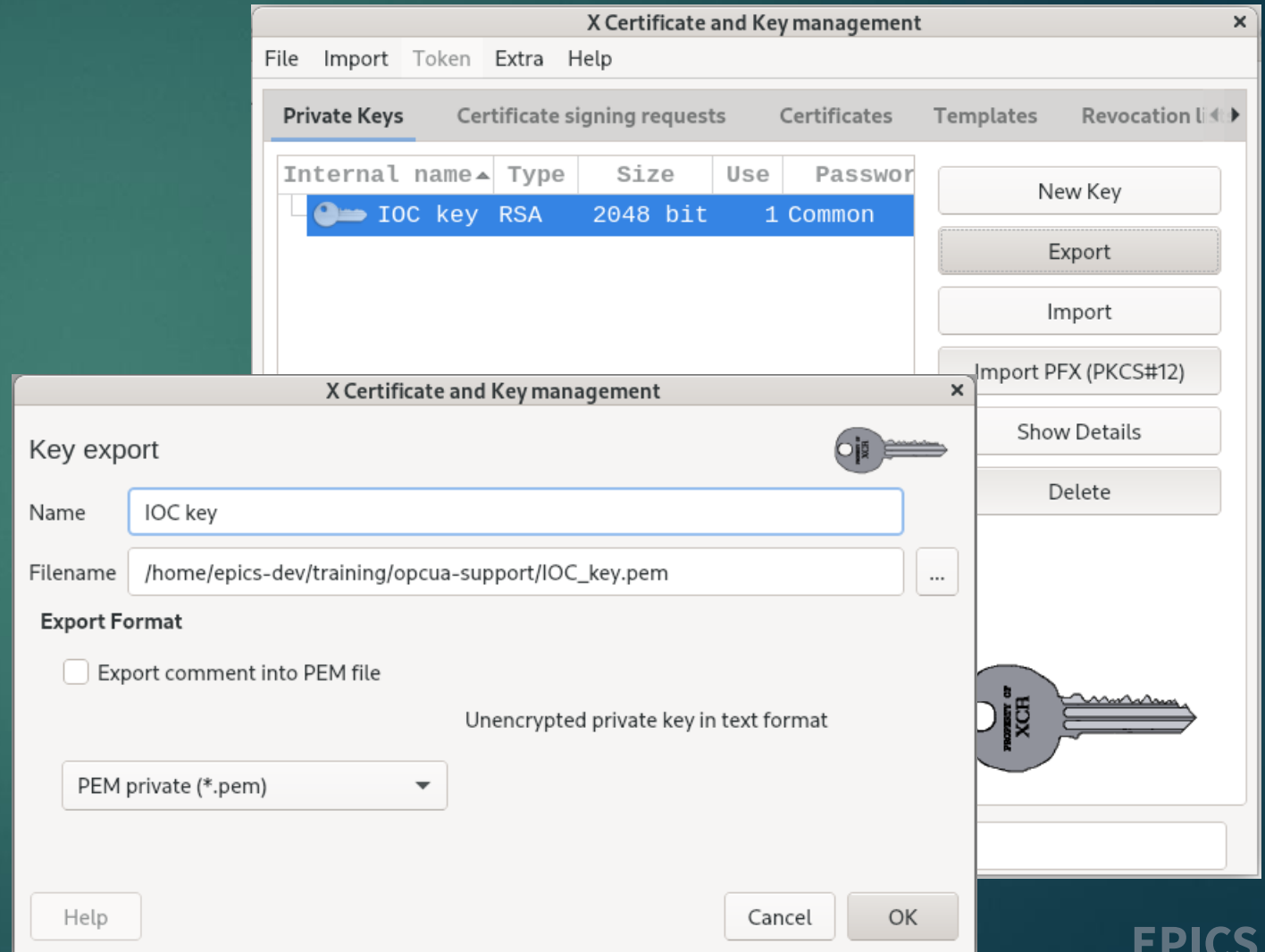
- ▶ Tab **Private Keys**
- ▶ Button **New Key**
- ▶ Call it **IOC key**



Export the IOC Private Key

17

- ▶ Tab **Private Keys**
- ▶ Select the key
- ▶ Button **Export**
- ▶ Select **PEM** format
- ▶ Click **OK**



Create the IOC's Client Certificate

18

- ▶ Tab **Certificates**
- ▶ Button **New Certificate**
- ▶ Sub-Tab **Source**
- ▶ Make it self-signed
- ▶ Select and fully apply the template **OPCUA_Client_IOC**

The screenshot shows the 'X Certificate and Key management' application window. The 'Certificates' tab is selected, and the 'New Certificate' button is visible. A dialog box titled 'Create x509 Certificate' is open, showing the 'Source' sub-tab. The 'Signing request' section has 'Copy extensions from the request' checked. The 'Signing' section has 'Create a self signed certificate' selected. The 'Signature algorithm' is set to 'SHA256'. The 'Template for the new certificate' is set to 'OPCUA_Client_IOC'. Buttons for 'Apply extensions', 'Apply subject', and 'Apply all' are at the bottom.

X Certificate and Key management

File Import Token Extra Help

Private Keys Certificate signing requests Certificates Templates Revocation List

Internal name commonName CA Serial Expiration Date New Certificate

X Certificate and Key management

Create x509 Certificate

Source Subject Extensions Key usage Netscape Advanced Comment

Signing request

Sign this Certificate signing request

Copy extensions from the request

Modify subject of the request

Signing

Create a self signed certificate

Use this Certificate for signing

Signature algorithm SHA256

Template for the new certificate

OPCUA_Client_IOC

Apply extensions Apply subject Apply all

Create the IOC's Client Certificate

19

- ▶ Sub-Tab **Subject**

Minimal Setup:

- ▶ Set the **countryName** to any two letters
- ▶ Give it a nice **commonName**

The screenshot shows the 'X Certificate and Key management' window with the 'Create x509 Certificate' dialog open. The 'Subject' tab is selected, showing fields for 'Internal Name', 'Distinguished name' (with sub-fields for countryName, organizationalUnitName, stateOrProvinceName, commonName, localityName, emailAddress, and organizationName), and 'Private key'. The 'commonName' field is highlighted with a blue border and contains the text 'iocUaDemoServer@rocky'. The 'Private key' dropdown is set to 'IOC key (RSA:2048 bit)'. There are 'Add' and 'Delete' buttons for the extensions table, and 'Help', 'Cancel', and 'OK' buttons at the bottom.

Type	Content
------	---------

Create the IOC's Client Certificate

20

- ▶ Sub-Tab **Extensions**

Important Setup:

- ▶ In the extensions, replace the placeholders **<IOC>** and **<HOST>**
- ▶ The URI must match what the IOC uses
- ▶ You can use **IP:** instead of **DNS:**
- ▶ Click **OK**

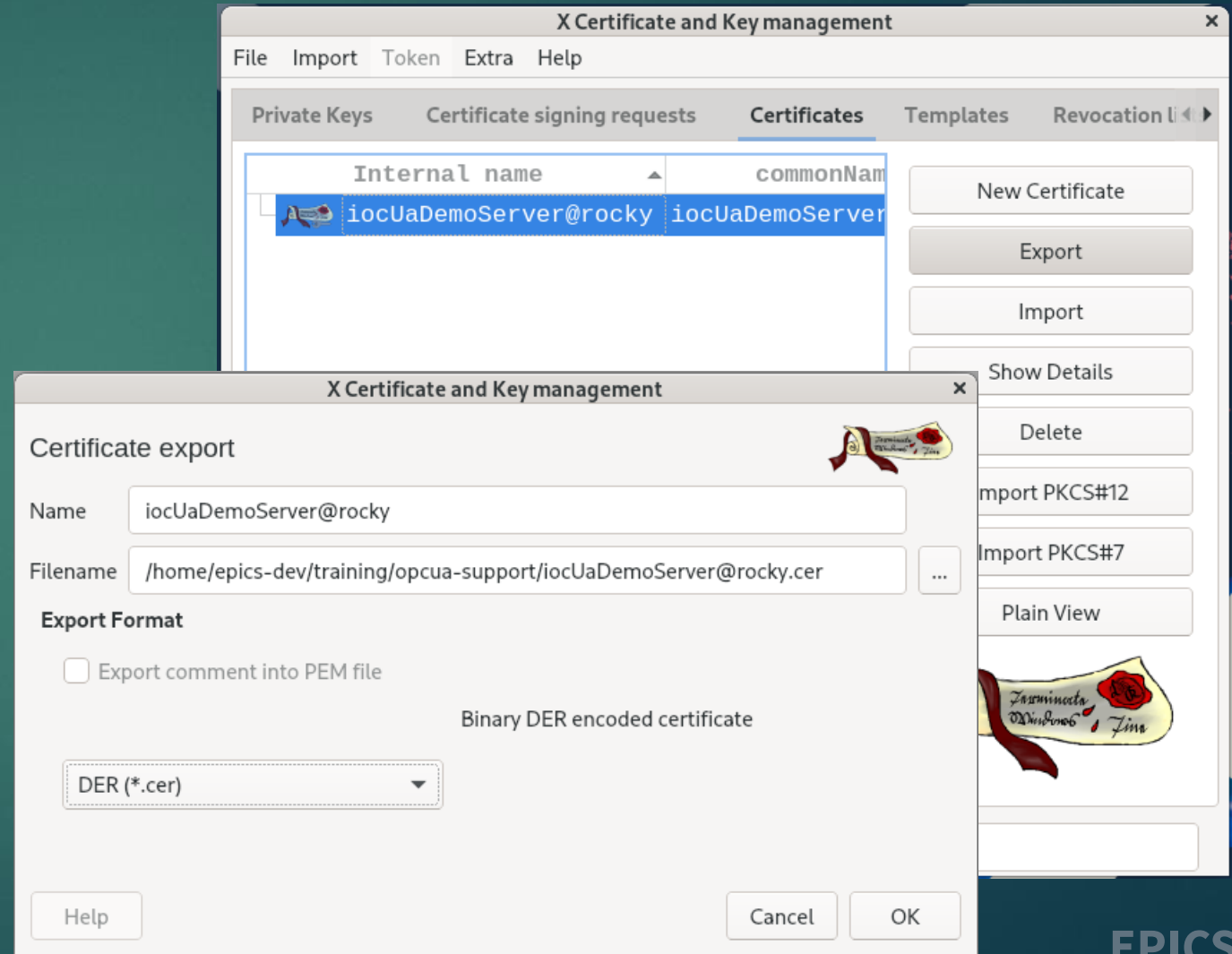
The screenshot shows the 'X Certificate and Key management' dialog box, specifically the 'Create x509 Certificate' window with the 'Extensions' tab selected. The dialog has a title bar with 'X Certificate and Key management' and a close button. Below the title bar, the window title is 'Create x509 Certificate'. The 'Extensions' tab is active, showing various configuration options. The 'X509v3 Basic Constraints' section includes a 'Type' dropdown set to 'End Entity', a 'Path length' field, and a 'Critical' checkbox checked. The 'Key identifier' section has two checkboxes: 'X509v3 Subject Key Identifier' (checked) and 'X509v3 Authority Key Identifier' (unchecked). The 'Validity' section includes 'Not before' and 'Not after' date pickers set to '2026-04-22 09:21 GMT' and '2027-04-22 09:21 GMT' respectively, and a 'Time range' section with a '12' month dropdown and an 'Apply' button. Below these are options for 'Midnight', 'Local time', and 'No well-defined expiration'. The 'X509v3 Name Constraints' section has an empty field and an 'Edit' button. The 'X509v3 Subject Alternative Name' section has a green checkmark and a text field containing 'URI:urn:iocUaDemoServer@rocky:EPICS:IOC, DNS:rocky|', with an 'Edit' button. The 'X509v3 Issuer Alternative Name' section has an empty field and an 'Edit' button. The 'X509v3 CRL Distribution Points' section has an empty field and an 'Edit' button. The 'Authority Information Access' section has an empty field and an 'Edit' button. At the bottom, there is an 'OCSP Must Staple' checkbox (unchecked) and three buttons: 'Help', 'Cancel', and 'OK'.

Export the Certificate

21

- ▶ Tab **Certificates**
- ▶ Select the certificate
- ▶ Button **Export**
- ▶ Select **DER** format
- ▶ Click **OK**

- ▶ Exit xca



Set Up the IOC's PKI File Store

22

- ▶ Create the minimal necessary directories for the IOC PKI file store
- ▶ Move the created self-signed Client Certificate and its private key into it

```
$ mkdir -p iocBoot/iocUaDemoServer/pki/own  
$ mkdir -p iocBoot/iocUaDemoServer/pki/trusted/certs  
$ mv IOC_key.pem iocUaDemoServer@rocky.cer iocBoot/iocUaDemoServer/pki/own/
```

Configure the IOC to use Security

23

Edit **st.cmd**
(gedit is available)

- ▶ Remove the setting that switches off Security

Instead:

- ▶ Set the location of the PKI file store
- ▶ Load the client certificate

```
Open [icon] st.cmd ~/training/opcua-support/iocBoot/iocUaDemoServer Save
5
6 < envPaths
7
8 cd "${TOP}"
9
10 ## Register all support components
11 dbLoadDatabase "db/opcuaIoc.dbd"
12 opcuaIoc_registerRecordDeviceDriver pdbname
13
14 # Pretty minimal setup: one session with a 200ms subscription on top
15 opcuaSession OPC1 opc.tcp://localhost:48010
16 opcuaSubscription SUB1 OPC1 200
17
18 # Configure security
19 epicsEnvSet PKI /home/epics-dev/training/opcua-support/iocBoot/iocUaDemoServer/pki
20 opcuaSetupPKI ${PKI}
21 opcuaClientCertificate ${PKI}/own/iocUaDemoServer@rocky.cer ${PKI}/own/IOC_key.pem
22
23 # Set up a namespace mapping
24 # (the databases use ns=2, but the demo server >=v1.8 uses ns=3)
25
26 opcuaMapNamespace OPC1 2 "http://www.unifiedautomation.com/DemoServer/"
27
28 # Load the databases for the UaServerCpp demo server
29 # (you can set DEBUG=<n> to set default values in all TPRO fields)
30
31 dbLoadRecords "db/UaDemoServer-server.db", "P=OPC:,R=,SESS=OPC1,SUBS=SUB1"
32 dbLoadRecords "db/Demo.Dynamic.Arrays.db", "P=OPC:,R=DDA:,SESS=OPC1,SUBS=SUB1"
33 dbLoadRecords "db/Demo.Dynamic.Scalar.db", "P=OPC:,R=DDS:,SESS=OPC1,SUBS=SUB1"
```

Establish Trust

Remember that trust needs to be established both ways:

- ▶ Make the IOC trust the Demo Server's certificate
*copy the server certificate in the IOC's **trusted/certs** store*

```
$ cp ~/server_demo/config/pkiserver/own/certs/uaservercpp.der iocBoot/iocUaDemoServer/pki/trusted/certs/
```

- ▶ Make the Demo Server trust the IOC's certificate
*copy the IOC certificate in the server's **trusted/certs** store*

```
$ cp iocBoot/iocUaDemoServer/pki/own/iocUaDemoServer@rocky.cer ~/server_demo/config/pkiserver/trusted/certs/
```

- ▶ Start the IOC ... and ...

```
$ ( cd iocBoot/iocUaDemoServer/; ../../bin/linux-x86_64/opcuaIoc st.cmd )
```

Connect securely

▶ ... and ... yay!

```
iocInit
Starting iocInit
#####
## EPICS R7.0.10
## Rev. 2026-04-13T11:16+0000
## Rev. Date build date/time:
#####
OPC UA Client Device Support 0.11.2 (-); using Open62541 Client API v1.3.15
iocRun: All initialization complete
OPC UA: Autoconnecting sessions
[2026-04-22 11:48:32.733 (UTC+0000)] info/server      Reloading the trust-list
[2026-04-22 11:48:32.736 (UTC+0000)] info/server      Reloading the issuer-list
[2026-04-22 11:48:32.736 (UTC+0000)] info/server      Reloading the revocation-list
OPC UA session OPC1: connected as 'Anonymous' with security level 120 (mode=SignAndEncrypt; policy=Aes128_Sha256_Rsa0aep)
epics>
```

Using CA-Based Certificates

26

Very similar, but different:

- ▶ Create your CA: a new key and a self-signed CA certificate
- ▶ Export the CA Certificate
- ▶ Create a new IOC Client Certificate, containing the IOC key, but sign it with the CA key (instead of self-signing)
- ▶ Export the IOC Client Certificate and copy it to the IOC's **own** folder
- ▶ Change the st.cmd file to load the new Client Certificate
- ▶ Create a new empty CRL, signed with the CA key
- ▶ Copy into the server PKI:
the CA certificate (into **trusted/certs**) and the CRL (into **trusted/crl**)

Using CA-Based Certificates

27

Remember that I said “be prepared for a lot of frustration” ??

- ▶ I verified this workflow in trustworthy documentation places (and using AI, obviously)
- ▶ I spent a few hours yesterday creating and exporting certs and CRLs, experimenting with formats, file suffixes and locations
- ▶ I could not get the `f*@&#^$*&^$@!!` thing to connect: “BadSecurityCheckFailed”
- ▶ I am sorry.
I will eventually get it to work and send you the updated slides.

Thanks for Your Attention!

28



Any questions left
unanswered?



OPC UA room on
EPICS Chat