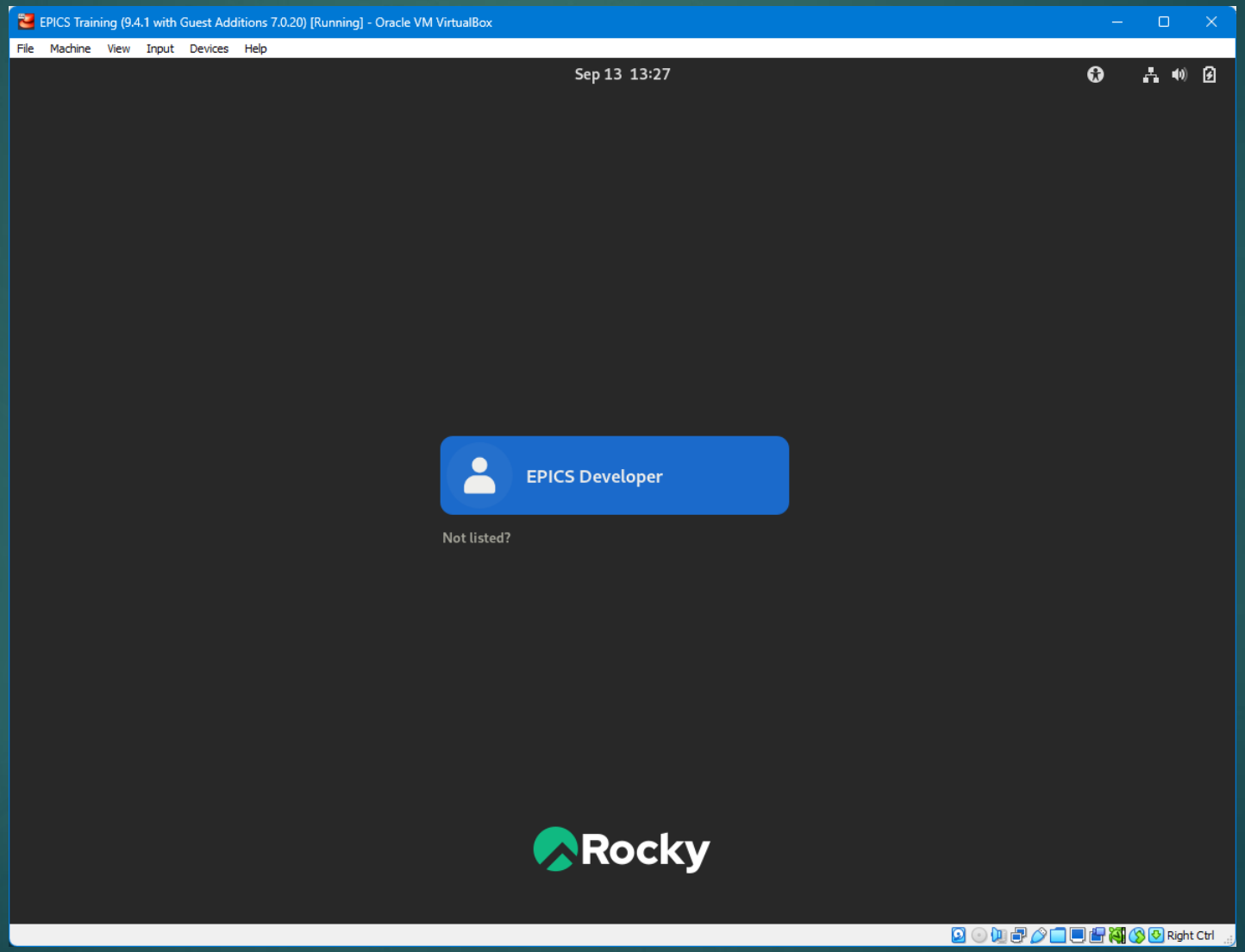
OPC UA Security

DEMO / HANDS-ON INSTRUCTIONS

Ralph Lange

Log Onto the Training-VM



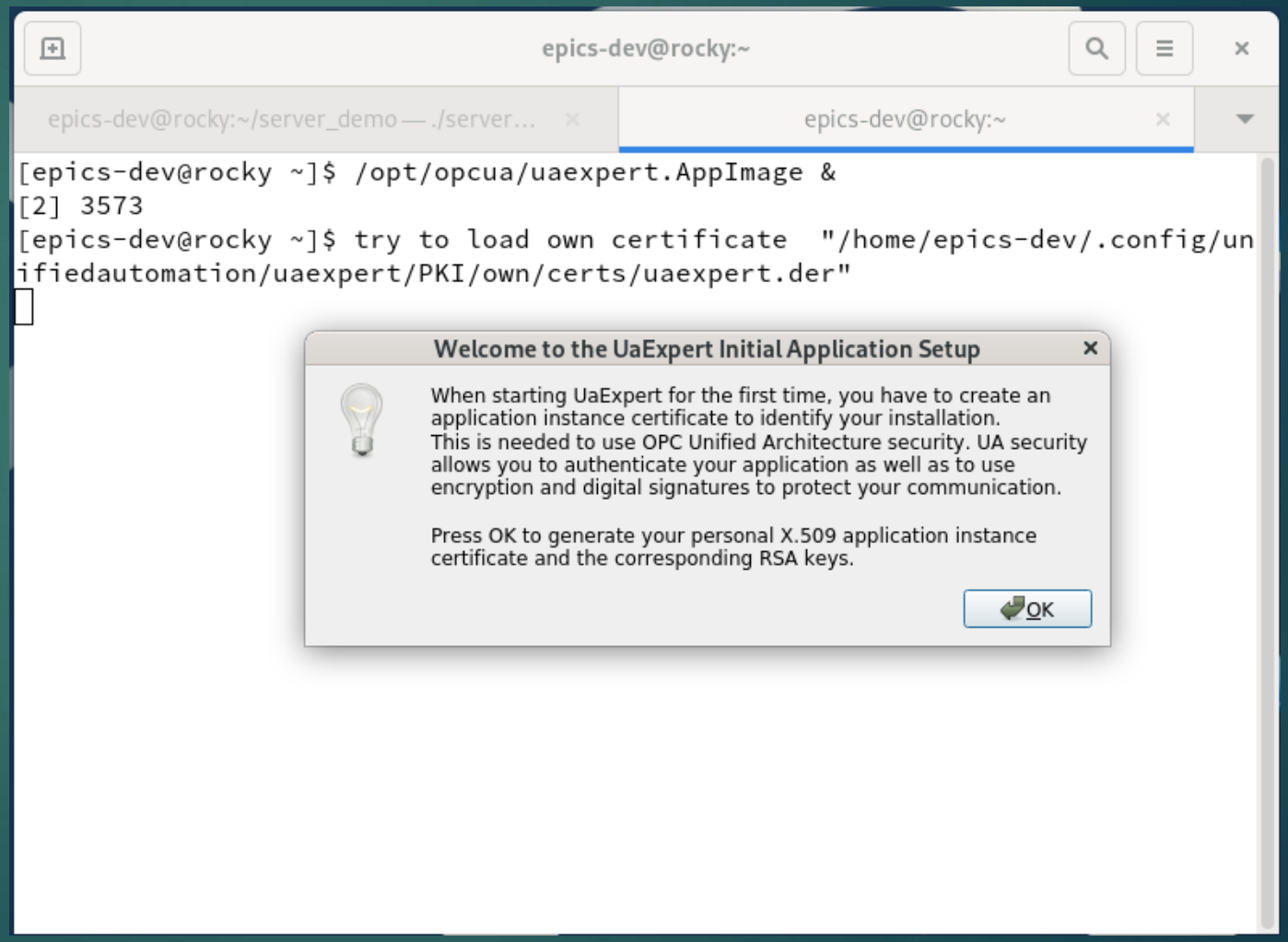
Start the Demo Server

- 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

Start the UaExpert Browser

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

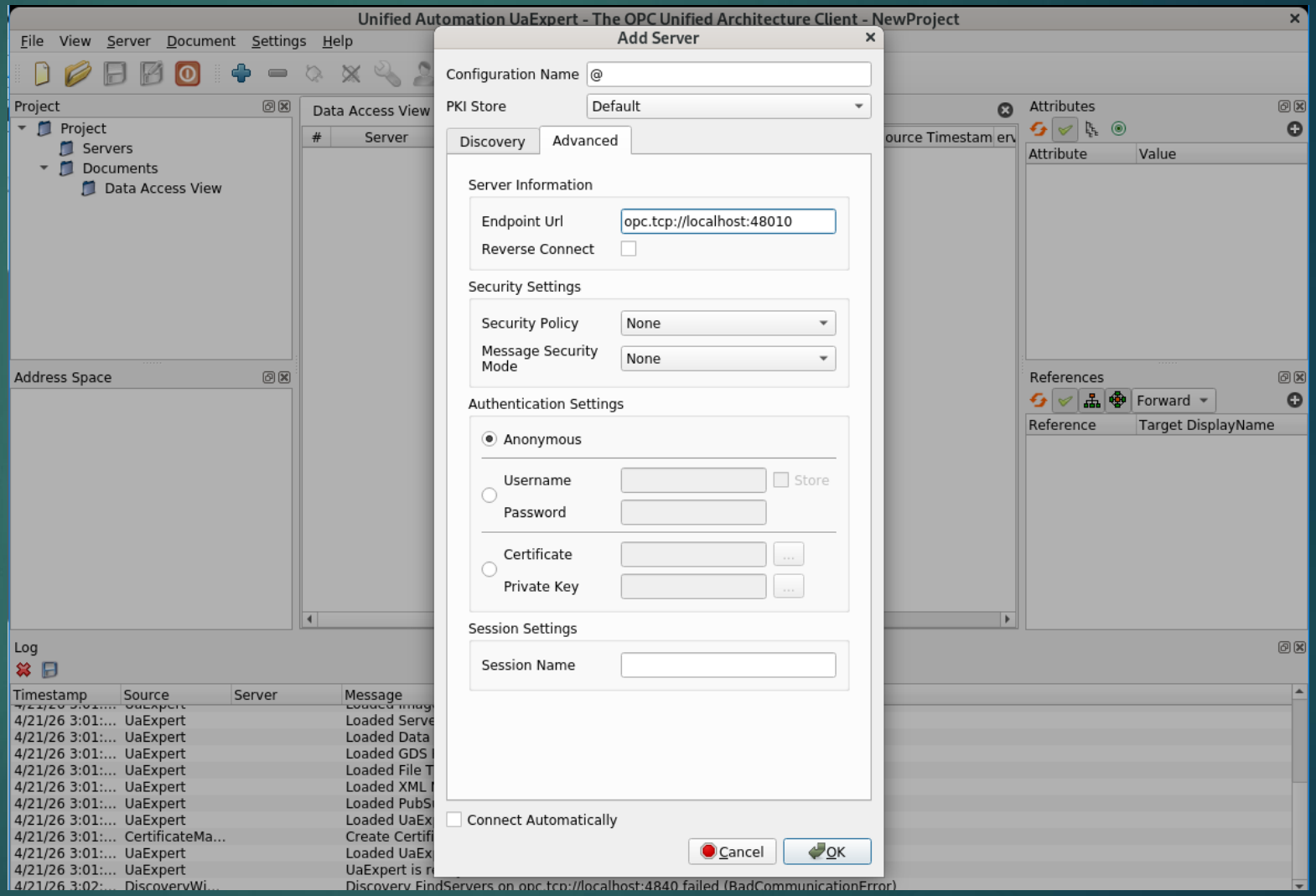
  Just enter any data.



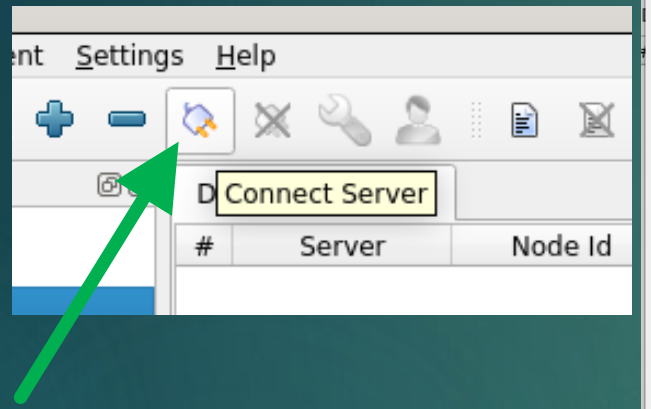
UaExpert: Add the Demo Server

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

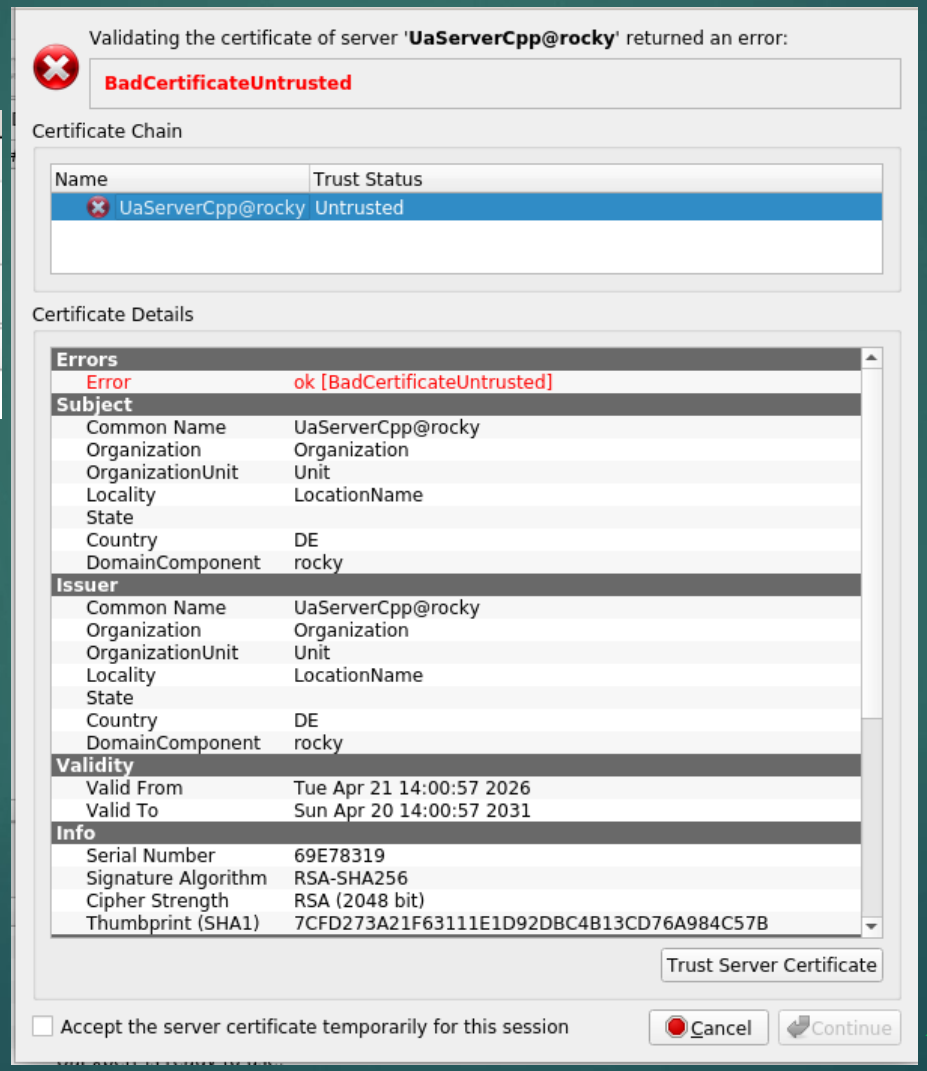
opc.tcp://localhost:48010



UaExpert: Connect

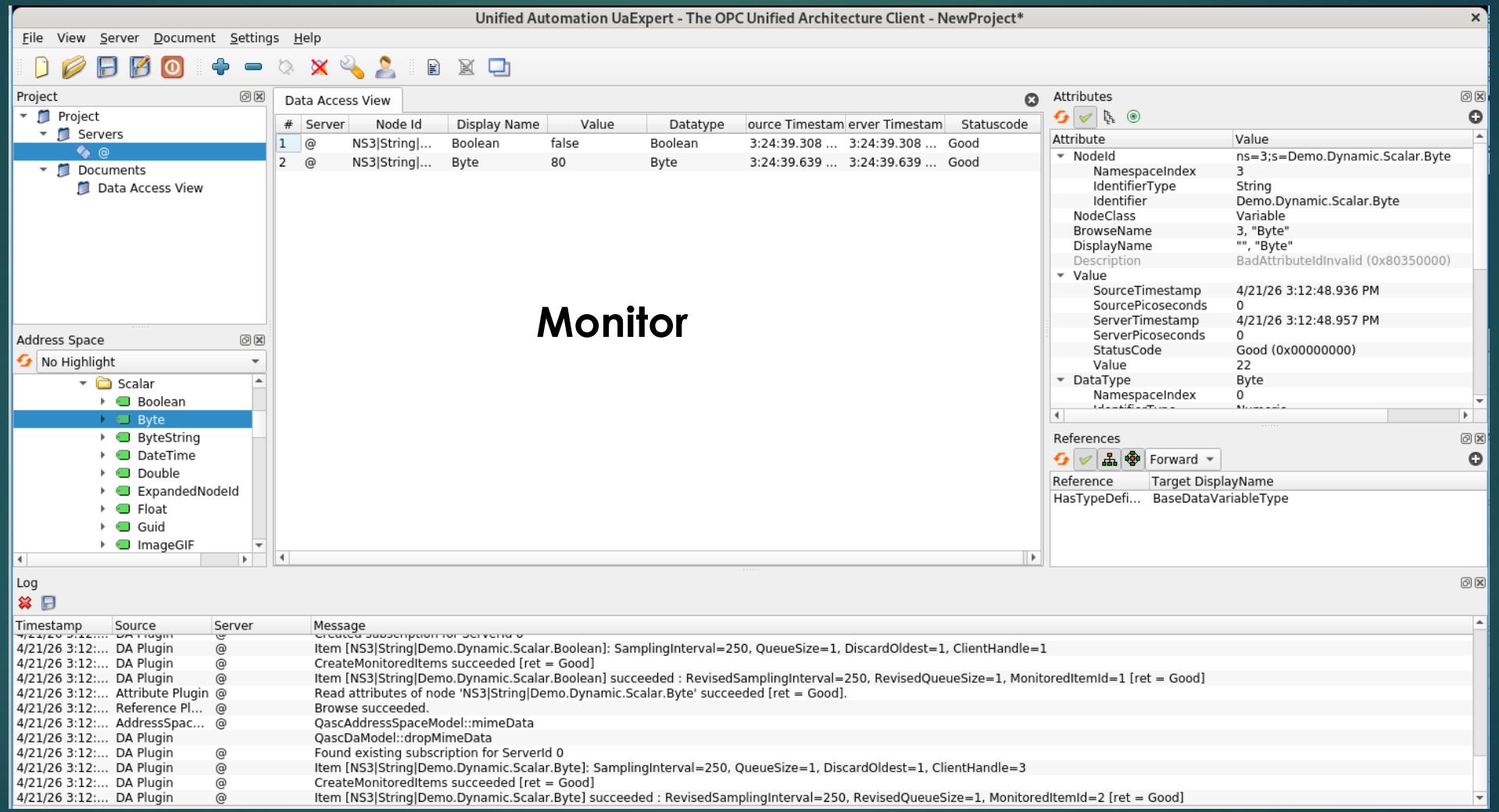


- Connect by clicking on the *Connect Server* icon



- Trust the unknown self-signed server certificate
- and *Continue*

UaExpert: Familiarize Yourself



Browse

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 O.linux-x86_64 ../..
mkdir -p O.Common
make -C O.linux-x86_64 -f ../Makefile TOP=../.. \
    T_A=linux-x86_64 install
make[2]: Entering directory '/home/epics-dev/training/opcua-support/configure/O.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/O.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


```
[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 pdbbase
# 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_RsaOaep Aes256_Sha256_RsaPss Basic128Rsa15 Basic256 Basic256Sha
256 None
```

Prepare the **xca** Cert Manager

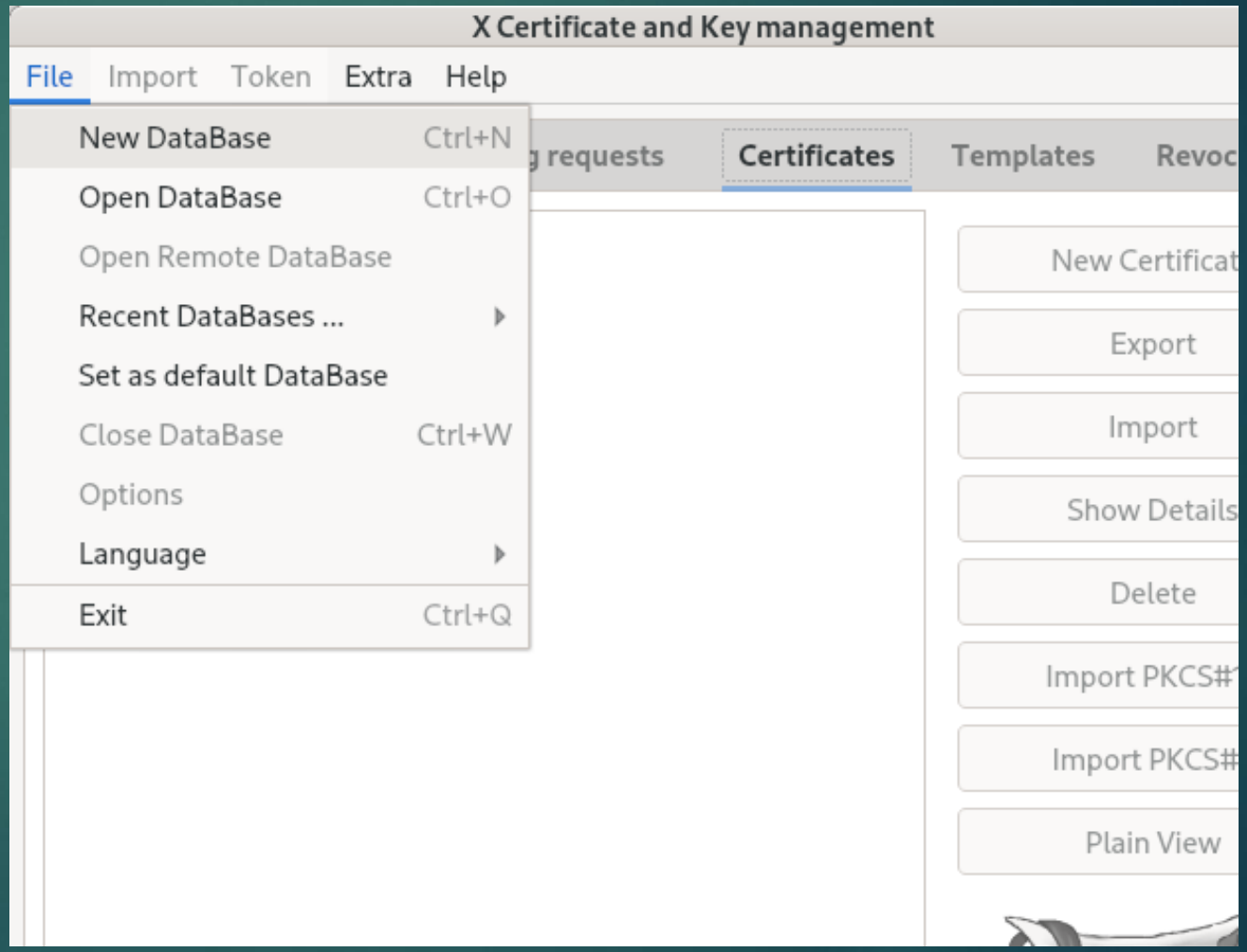
- Powerful free Certificate and Key Management tool
- Available on Linux, Windows, MacOS
- Supports many key and certificate formats (including ssh)
- Supports CA-based workflows (including sub-CA hierarchies)
  Nice visualization of CA signature chains
- Allows to save and load templates
  Great help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

Create the xca Database

- Start

```
$ xca &
```

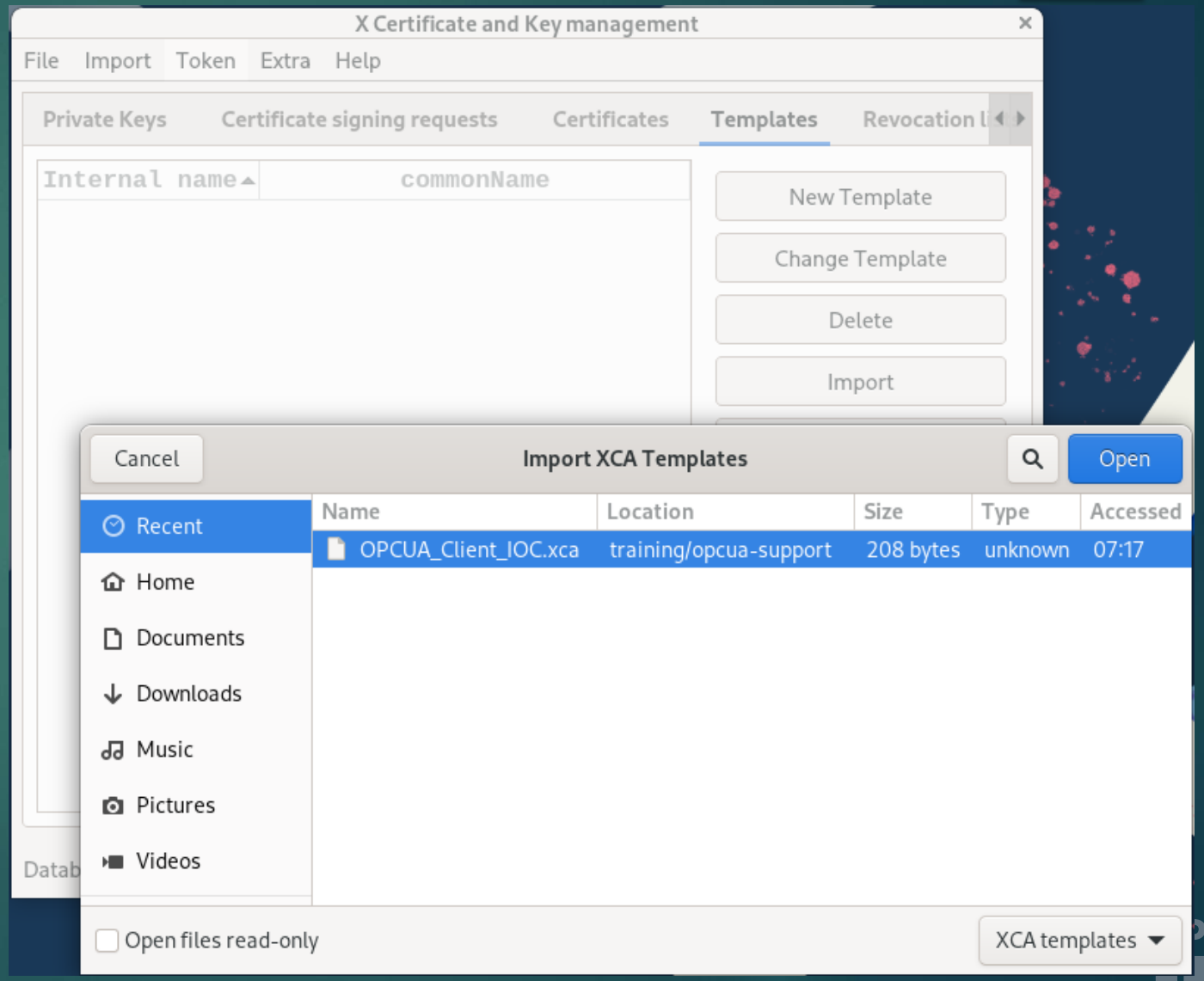
- Then
  File -> New DataBase
- Give it a name
- You can leave it without a password



Import the IOC Cert Template

- 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

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

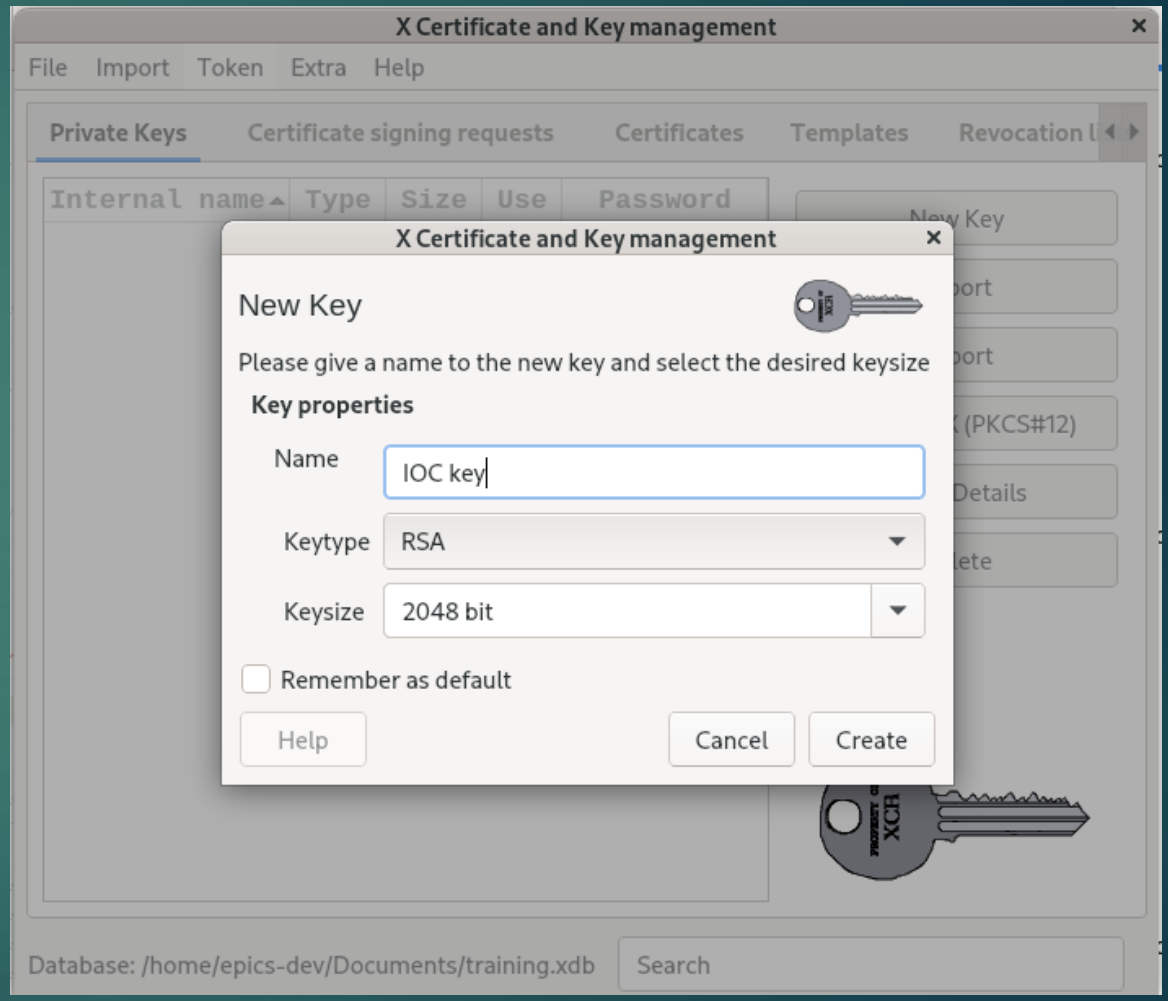
Establish trust

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

Connect!

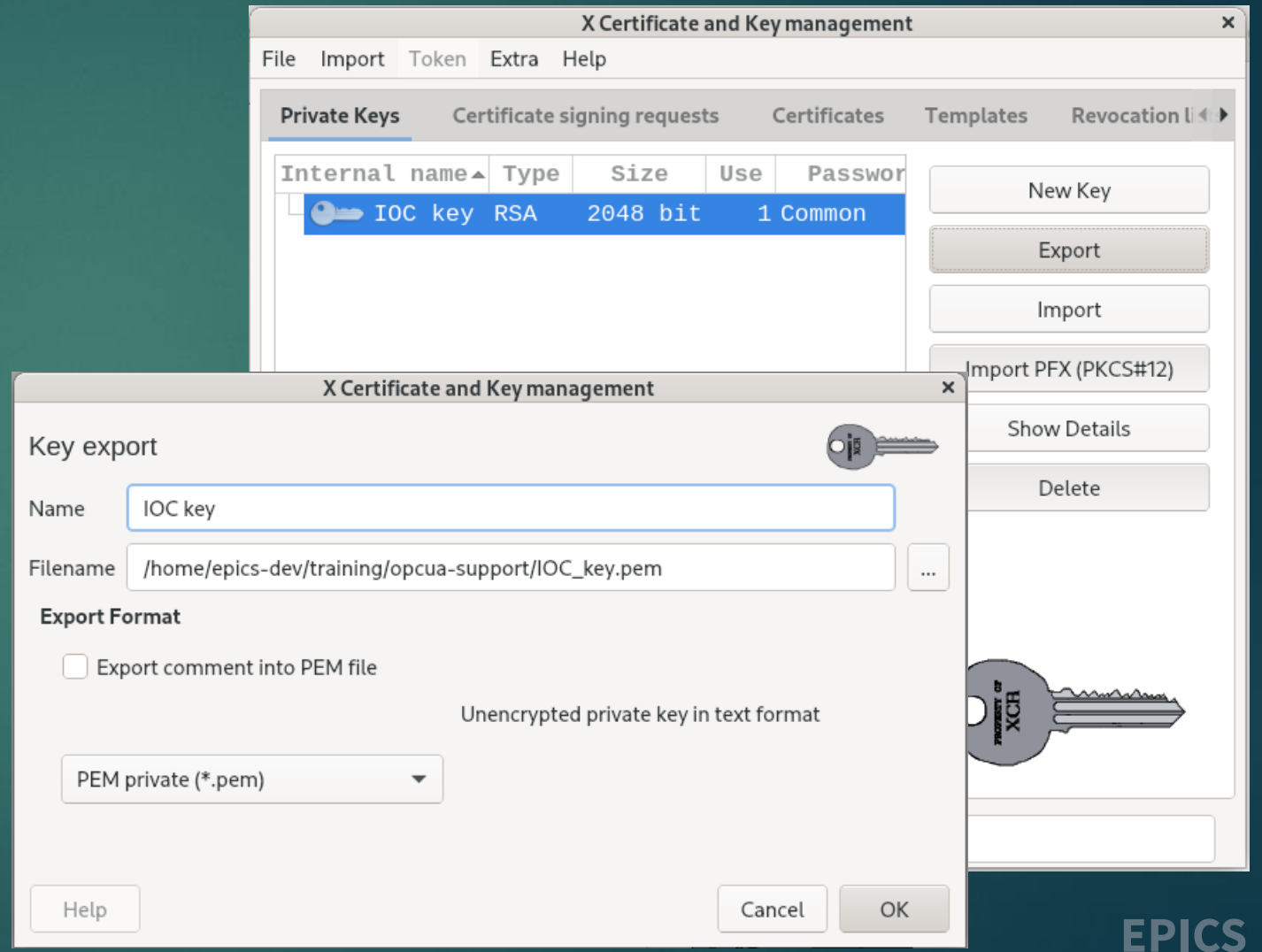
Create a Key Pair for the IOC

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



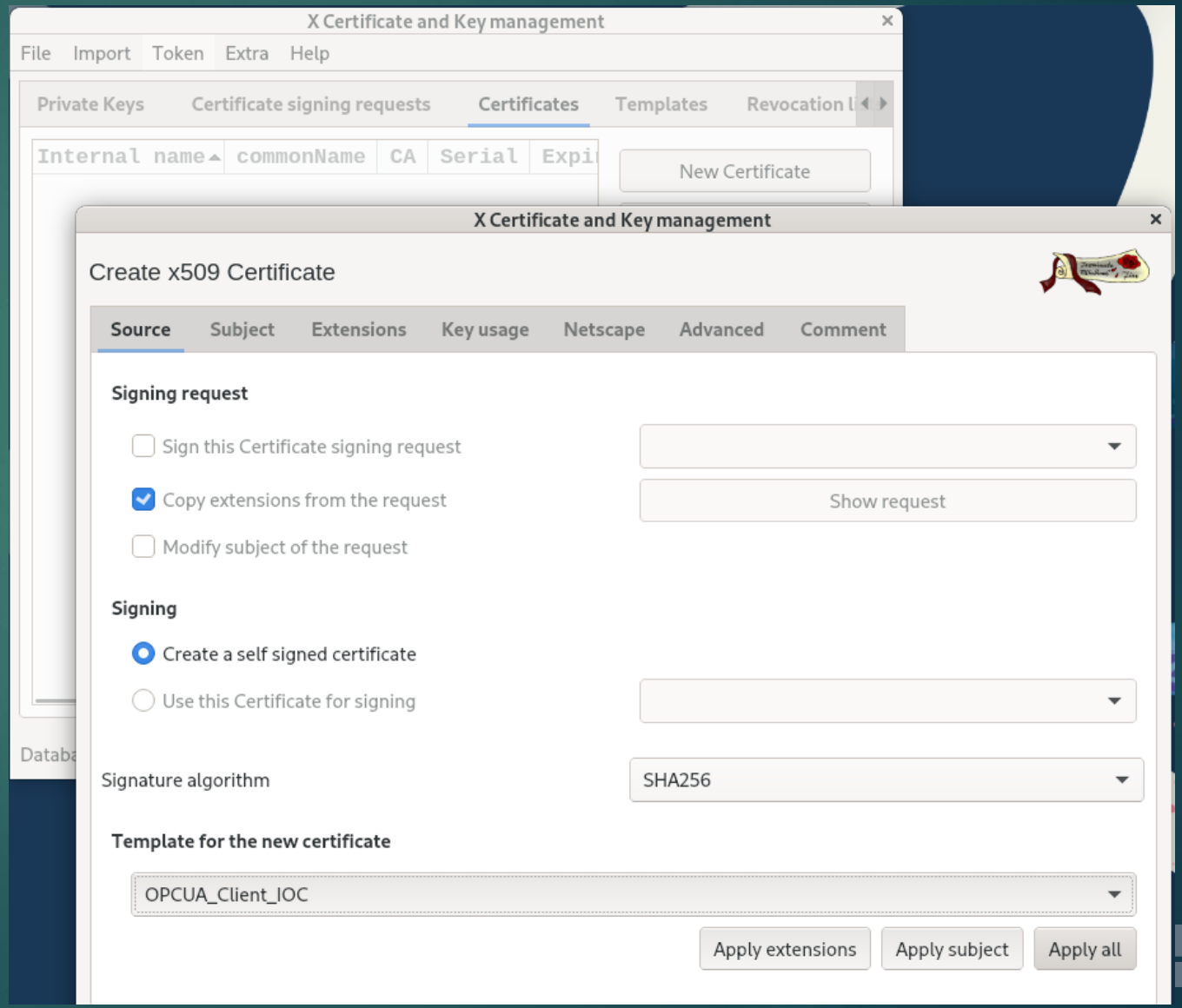
Export the IOC Private Key

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



Create the IOC's Client Certificate

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

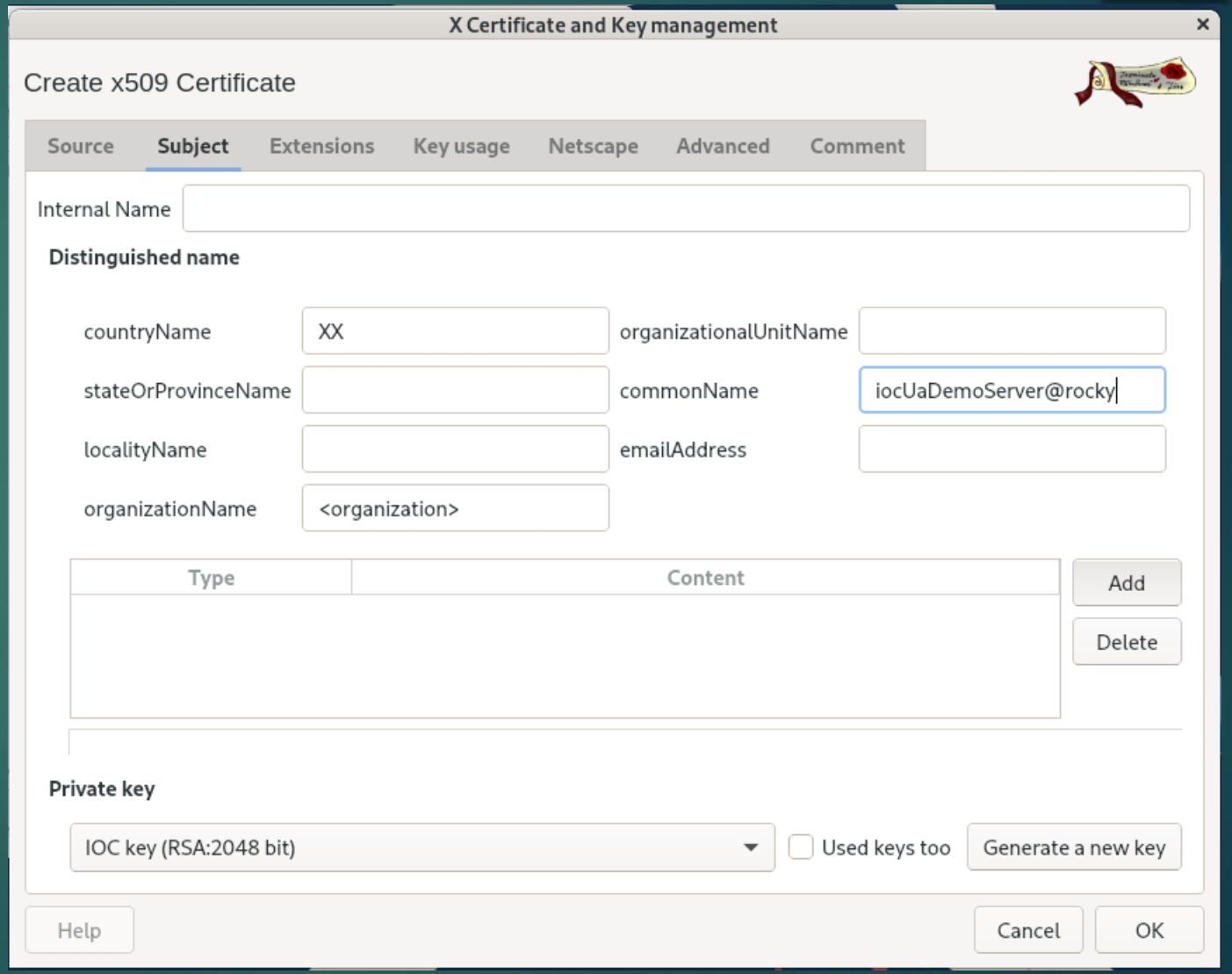


Create the IOC's Client Certificate

- Sub-Tab **Subject**

Minimal Setup:

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

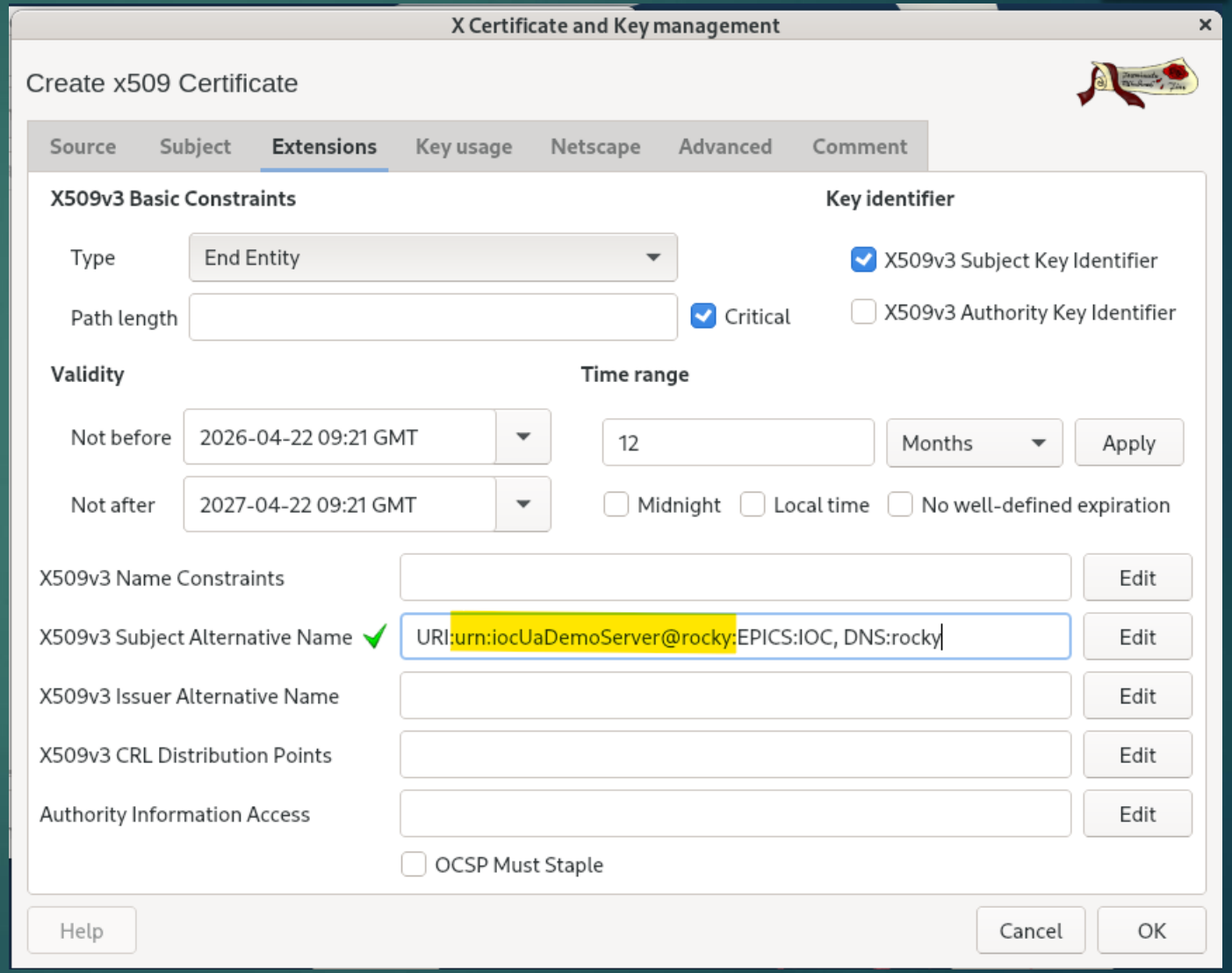


Create the IOC's Client Certificate

- Sub-Tab **Extensions**

Important Setup:

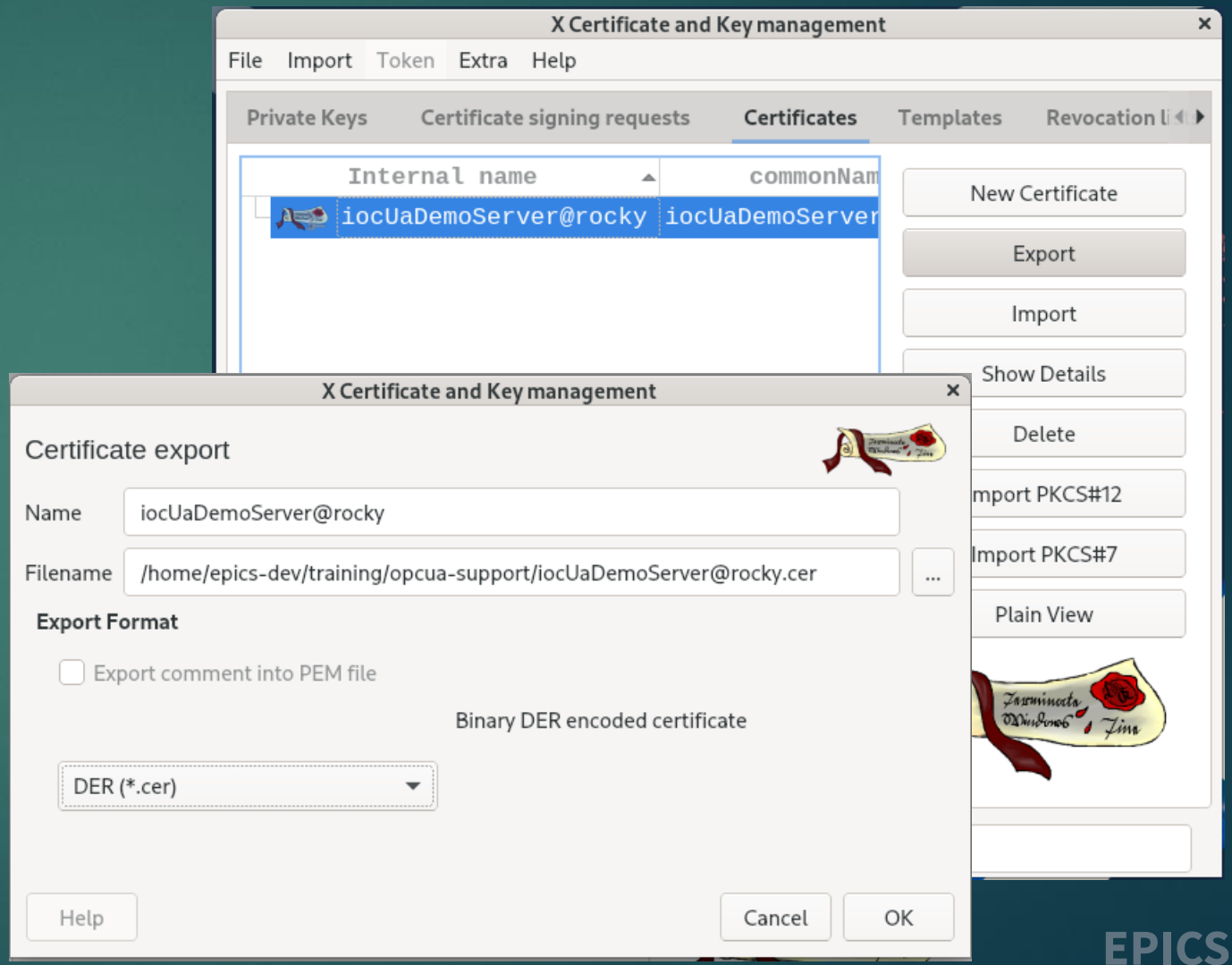
- In the extensions, replace the placeholders **<IOC>** and **<HOST>**
- The URI must match what the IOC uses
- You can use **IP:** instead of **DNS:**
- Click **OK**



Export the Certificate

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

- Exit xca



Set Up the IOC's PKI File Store

- 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

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 | st.cmd | ~/training/opcua-support/iocBoot/iocUaDemoServer | Save

```
< envPaths

cd "${TOP}"

## Register all support components
dbLoadDatabase "dbd/opcuaIoc.dbd"
opcuaIoc_registerRecordDeviceDriver pdbbase

# Pretty minimal setup: one session with a 200ms subscription on top
opcuaSession OPC1 opc.tcp://localhost:48010
opcuaSubscription SUB1 OPC1 200

# Configure security
epicsEnvSet PKI /home/epics-dev/training/opcua-support/iocBoot/iocUaDemoServer/pki
opcuaSetupPKI ${PKI}
opcuaClientCertificate ${PKI}/own/iocUaDemoServer@rocky.cer ${PKI}/own/IOC_key.pem

# Set up a namespace mapping
# (the databases use ns=2, but the demo server >=v1.8 uses ns=3)

opcuaMapNamespace OPC1 2 "http://www.unifiedautomation.com/DemoServer/"

# Load the databases for the UaServerCpp demo server
# (you can set DEBUG=<n>) to set default values in all TPRO fields)

dbLoadRecords "db/UaDemoServer-server.db", "P=OPC:,R=,SESS=OPC1,SUBS=SUB1"
dbLoadRecords "db/Demo.Dynamic.Arrays.db", "P=OPC:,R=DDA:,SESS=OPC1,SUBS=SUB1"
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 )
```

EPICS

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_RsaOaep)
epics>
```

Using CA-Based Certificates

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

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!

Any questions left unanswered?

OPC UA room on EPICS Chat