
Update: 2022.12

DAC_OPCUAX

支持安全策略和应用证书的 OPCUA 驱动 FOR IOSERVER



1OPCUA 概述	1
1.1 安全架构(Security architecture).....	1
1.2 安全策略(SecurityPolicy)	2
1.3 安全模式(SecurityMode)	2
1.4 理解证书	3
1.5 运行环境	4
2 驱动配置	5
2.1OPCUA 驱动配置	5
2.2 账号和安全策略的关系	7
2.3 标签表格式	7
3 从旧版升级迁移	9
3.1 升级评估测试	9
3.2 复制文件	9
3.3 升级到带证书模式	9
3.4 不带证书模式升级	9
附录 1 制作自签名证书	12

1 OPCUA 概述

OPCUA 是 OPC 基金会于 2008 年发布的统一架构跨平台工业信息交换协议，用于替换原 OPCDA 标准，适用于现场设备，控制系统，制造执行系统和企业资源规划系统等应用领域的制造软件。具体参见 OPC 官方网站(<https://opcfoundation.org/>)的介绍。

实时库的数据网关 **ioserver** 的 OPCUA 驱动有两个：

- **dac_opcua** 旧版(**dac_opcua.dll/dac_opcua.so**)，通信层不支持应用证书和消息加密签名。
- **dac_opcuax** 新版(**dac_opcuax.dll/dac_opcuax.so**)，自 rdb2022.9 版开始提供，支持通信层加密通道，支持消息的加密和签名。用于替代旧版的 **opcua** 驱动，适合某些安全性要求高的场景。

*本文描述的是新版 OPCUA 驱动 **dac_opcuax** 驱动的用法以及安全证书的应用方法。可完全替代旧版的 OPCUA 驱动，也支持不带证书的场景，强烈建议升级。*

1.1 安全架构(Security architecture)

为了更好的理解配置和证书，这里先简单介绍一下 OPCUA 的安全架构。下图为 OPCUA 规范第二章截图。

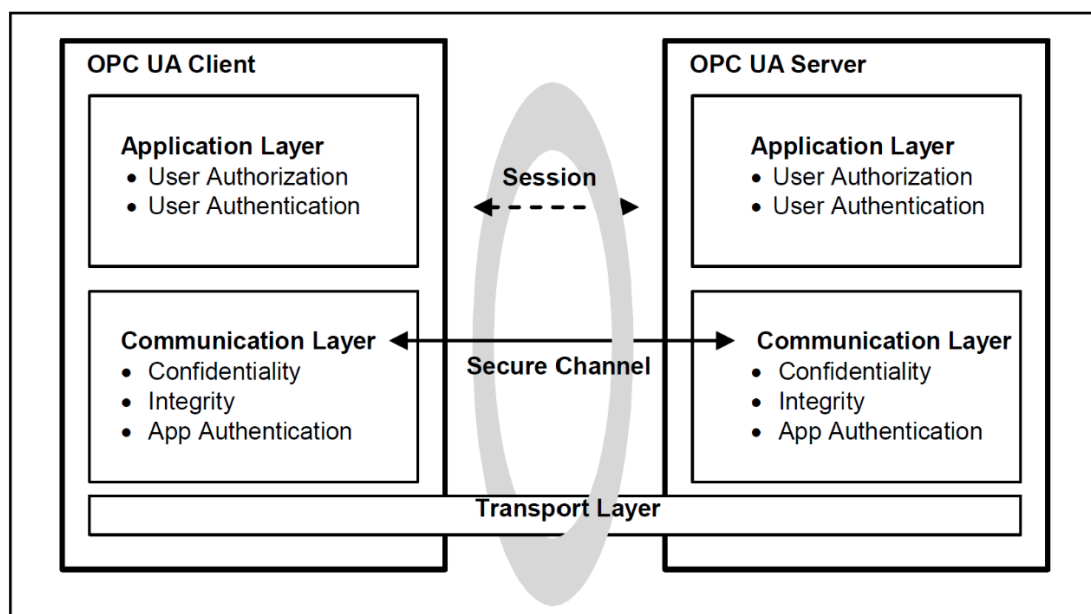


Figure 2 – OPC UA security architecture

图中看出整个通信分为三层：

- 传输层(Transport Layer)：这里一般是 TCP

-
- 安全通道(Secure Channel): 安全通道, 非对称加密握手协商一次性对称加密密钥用于后面的消息加密传输。后面配置的安全策略, 证书就是用于这一层的。
 - Session: 会话层, 应用信息编码和交互, 包含匿名或者用户/账户模式登录验证。这个和 Secure Channel 没有关系, 简单说就是, 只要服务端支持, 使用带证书的加密通道也可以用匿名方式登录建立会话。

1.2 安全策略(SecurityPolicy)

安全策略(SecurityPolicy)定义了使用的加密算法, `dac_opcuax` 驱动支持的 OPCUA 安全策略包括:

- None : <http://opcfoundation.org/UA/SecurityPolicy#None>
- Basic256: <http://opcfoundation.org/UA/SecurityPolicy#Basic256>
- Basic128Rsa15:
<http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15>
- Basic256Sha256:
<http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256>

其中 Basic256 和 Basic128Rsa15 已经被标准废弃, 不建议使用, 已经不安全了。None 是不使用加密和签名的普通通道。

1.3 安全模式(SecurityMode)

安全模式(SecurityMode)是指对是否对消息做签名或加密。

- None : 不签名, 也不加密, 适合上面的安全策略 None 模式。
- Sign : 签名, 目的是防止消息被篡改。发送端对消息+双方握手协商出的私密盐参数计算出 SHA256 散列值附在消息尾部, 对端也对消息做同样的运算后比对 SHA256 散列值是否相同以判断是否被更改。由于双方握手协商的盐参数第三方无法得知, 因此修改消息后无法计算出正确的 SHA256 散列值, 很轻易的被接收端识别出消息被篡改。
- Sign&Encrypt: 签名和加密一起使用, 加密目的是防止被偷窥。发送端使用握手时协商的算法(比如 AES_CBC)和协商时产生的一次性对称加密密钥(为何不用非对称的 RSA 加密消息, 因为这个算法太慢, 不适合加密数据, 只适合握手协商时传递一次性密钥和 SHA256 散列盐参数使用)。

双方握手过程和 HTTPS 的 TLS 基本相同, TLS 一般不需要客户端证书, 只是客户端对服务端的证书的识别采用根证书机构签名识别, OPCUA 需要互相交换证书, 握手时发送给对端的信息使用对端的公钥加密, 接收端使用私钥解密, 防止握手信息被偷窥, 支持双方信任模式, 可以使用自签名证书。

1.4 理解证书

OPCUA 的证书为 X.509 格式的证书，可以用 openssl 工具来制作自签名证书，后面有专门的章节来介绍如何制作证书。

证书主要包含应用标识(applicationUri)，非对称加密公钥(一般是 RSA)，域名 IP 等信息，证书在握手时需要发送给对端，如果双方验证后都信任对方，则开始下一步握手协商，否则建立安全通道失败。

与证书配对的还有一个私钥文件，存放的非对称加密(一般是 RSA，2048bit 密钥长度在当前已经够用了)的私钥，要保管好不可泄漏，用来解密对端用本端公钥加密的信息。

RSA 非对称加密的特点是信息使用公钥加密后只能用对应的私钥解密，握手时双方将含有公钥的证书发送给对方后，双方验证并信任对端证书后，交互的握手信息使用对端公钥加密的，保证不会被第三方偷窥到。但是一旦私钥被泄露后，就不安全了，应该把泄露私钥的证书加入 OPCUA Server 的废弃证书列表拒接连接。

OPCUA 支持自签名证书，无需第三方根证书机构签名背书，是否信任对端的证书完全由通信的双方决定。通常大多数的 OPCUA 应用在审核对端证书时有两个证书中的参数必须和环境相同。

在 windows 10 系统中，鼠标双击.der 证书可以显示证书信息。如下图是本系统自带证书(kipway_uaclient_cert.der)的信息：

字段	值
公钥参数	05 00
使用者密钥标识符	5589a383bac28c937190115d1a38cb6afed09c
授权密钥标识符	KeyID=5589a383bac28c937190115d1a38cb6
基本约束	Subject Type=End Entity, Path Length Constr
密钥用法	Digital Signature, Non-Repudiation, Key Enci
增强型密钥用法	服务器身份验证 (1.3.6.1.5.5.7.3.1), 客户端身份
使用者可选名称	DNS Name=jy-dell-i3, DNS Name=kipway.ne
指纹	5728200e660420a0c20f0a655133b476ef61e6

DNS Name=kipway.net
IP Address=127.0.0.1
IP Address=192.168.1.125
IP Address=192.168.1.36
IP Address=192.168.1.19
IP Address=192.168.1.59
IP Address=192.168.1.233
IP Address=192.168.1.230
URL=urn:kipway.client.application

DNS 或 IP: 应用程序(**ioserver**)所在机器的 DNS 或者 IP 要在“使用者可选名称”DNS

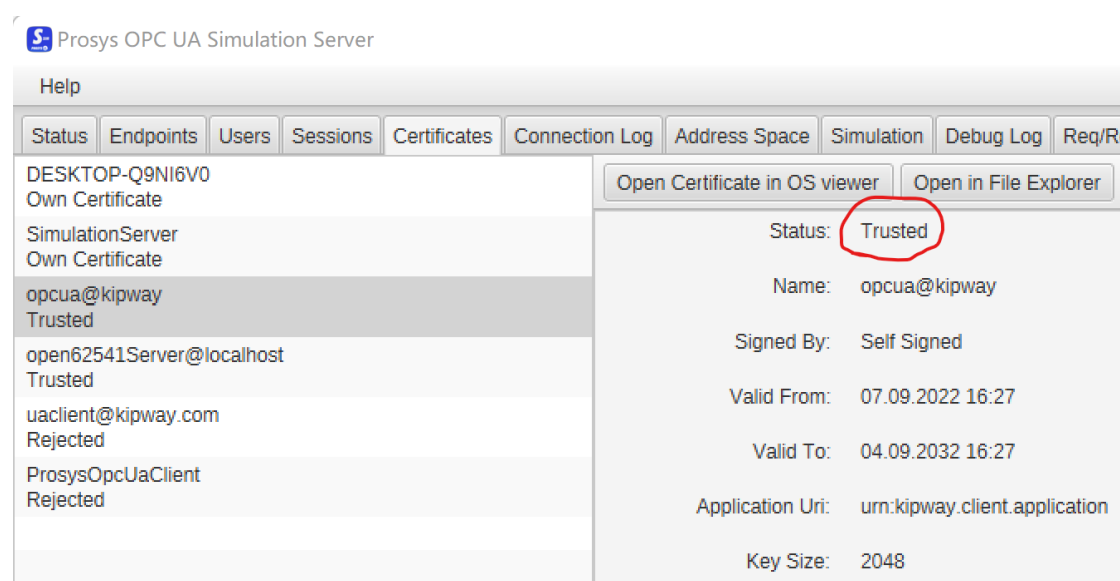
或者 IP 列表中。

应用程序标识(applicationUri)要和“使用者可选名称”中 URL 内容一致。

如 dac_opxuax 驱动例子驱动配置 opcua1.ini 中的 applicationUri 参数要按照如下配置。

applicationUri = urn:kipway.client.application

除此之外，好需要把客户端证书加入到 OPCUA Server 的信任列表中。下面使用 ProsysOPC UA Simulation Server 作为测试服务器，截图如下，看到状态为 Trusted 表示成功加入了信任列表。



1.5 运行环境

Windows:

最低要求: Windows server2008, windows7 即以上系统。推荐 windows server 2012/2016 版。

Linux(X86_64):

内核 3.10 及以上, libstdc++.so.6.0.19, glibc 2.17 及以上 64 位环境; 典型系统 centos7.6; 推荐 centos8 和 ubuntu server1804 及以上版本。

Linux(aarch64)

内核 4.4.131 及以上, libstdc++.so.6.0.21, glibc 2.23, 典型系统银河麒麟 V10 系统, 相当于 ubuntu16.04 的 aarch64 系统平台。

无需任何第三方库依赖。

2 驱动配置

所有的配置放在 `ioserver` 的 `config` 子目录中。包含两个配置文件，OPCUA 的配置和标签表，如软件包 `ioserver/config` 中的 `opcuax1.ini` 和 `opcuax1tags.csv` 文件。

2.1 OPCUA 驱动配置

下面以 `linux` 系统 `opcuax1.ini` 为例：

`#opc ua 客户端配置`

`[opcua]`

`#server 的连接点 URL`

`endpointUrl = opc.tcp://127.0.0.1:53530/OPCUA/SimulationServer`

`#通讯层安全策略, 填写, None, Basic256Sha256, Basic256, Basic128Rsa15, 其中 Basic256, Basic128Rsa15 已经弃用。`

`securityPolicy = Basic256Sha256`

`#客户端自己的应用标识, 只有使用加密安全策略时采用, 必须设置和证书里使用者可选名称中 URL 参数相同`

`applicationUri = urn:kipway.client.application`

`#客户端自己的证书文件名, 只有使用加密安全策略时采用, 必须加入 opcua server 的信任证书列表, 或者从 opcua server 那里获取应用端证书`

`certFilename = /etc/opcuacert/kipway_uaclient_cert.der`

`#客户端私钥文件名, 妥善保管, 防止泄露, 只有使用加密安全策略时采用。`

`keyFilename = /etc/opcuacert/kipway_uaclient_key.der`

`#消息加密方式, 填写 0-3: 0: INVALID 无效; 1: NONE(无); 2: SIGN(签名); 3: SIGNANDENCRYPT(签名和加密)`

`#签名防止被篡改, 加密防止被偷窥, 配置 2, 3 必须要有证书。`

msgSecurityMode = 3

#会话层用户验证-用户名；不配置表示匿名

username =

#会话层用户验证-用户密码

password =

[tag]

#标签表 csv 文件,自动识别文件内容编码

tagFilename = opcua1tags.csv

注:

- 1) 证书和私钥文件配置文件名为绝对路径文件名。
- 2) 标签表文件文件名放在 config 目录下,不带路径。可以是 utf8 编码,也可以是 gbk 编码,但是不要混合编码,否则汉字会出现乱码。
- 3) applicationUri 必须和证书里“使用者可选名称”的 URL 相同。
- 4) 运行 ioserver 的机器的 IP 必须在证书“使用者可选名称”的 IP 列表中。如果 IP 条件无法满足要求。还有两个方式:从 OPCUA server 哪里获取客户端应用证书或者自己制作自签名证书。注意此时要将证书里“使用者可选名称”中 URL 字段内的应用端标识 Uri 配置到 applicationUri 字段。
- 5) 将证书加入到 OPCUA server 的信任列表中。
- 6) dac_opcuax 驱动是固定内置为信任对端证书的,因此无需配置 OPCUA server 的证书,也不需配置对端的证书信任列表,工程人员施工配置时,可以人工审核 OPCServer URL 指向的服务端证书是否可信任。
- 7) msgSecurityMode 填写的是数字,没有证书时填写 1,有证书时填写 0, 2, 3 均可。
- 8) 如果连接失败,查看日志,检查错误原因,一般是 OPCUAServer 不支持指定的 securityPolicy, msgSecurityMode, 或者证书里的 applicationUri 和配置里不相同,或者客端 IP 不在客户端证书“使用者可选名称”的 IP 列表中。修改参数重新测试,或者从 OPCUA server 管理员处获取准确的安全策略和消息加密模式。

2.2 账号和安全策略的关系

用户账号以及是否匿名登录属于 OPCUA 应用层的，安全策略属于传输层，理论上是没有关系的。

但是为了保证用户账号密码不在传输过程中被偷窥而泄露，大部分主流的 OPCUA Server 都要求客户端在使用用户/账号登录时必须使用证书和安全策略(**Basic256Sha256**, **Basic256**, **Basic128Rsa15** 之一)。

按照下面的规则配置：

- 如果配置证书和安全策略，可配置匿名和用户/账号模式登录。
- 如果没有配置证书和安全策略，配置匿名(**username** 留空)模式登录。

在升级就工程时一定要注意上面两点。

2.3 标签表格式

标签表格式和旧版 `dac_opcua` 驱动的标签表完全相同, 下表和《ioserver 使用说明.pdf》里面的完全相同。

表 2-1 OPCUA 标签表字段属性定义

rdbtagname	实时库标签名
nsindex	opcua server 中标签的命名空间索引(namespaceIndex)，从 OPCUA server 那里获取，一个 0-65535 的整数。
nodeid	opcua server 标签的 nodeid(node identifier) 。默认为字符串，从 rdb2020.8 开始支持数字，" i ="开头为数字型，" s ="开头为字符串型。例如下面三个都是支持的： i=10032 s=m1.intval2 m1.intval2
datatype	数据类型： Digital 开关量 Int32 4 字节整数 Float32 单精度浮点型，4 字节 Int64 8 字节整数 Float64 双精度浮点型，8 字节 String 字符串型 ----- 下面是 rdb2022.7 开始支持分别配置实时库类型和 opcua 类型； 用于某些 opcua server 下行控制数据类型必须相同的场景

	<p>格式</p> <p>实时库类型: opcua 类型</p> <p>中间是西文冒号分开, 前面是实时库类型, 后面是 opcua 类型。</p> <p>如果实时库类型和 opcua 类型完全相同, 只填写实时库类型, 不要冒号和 opcua 类型(兼容 rdb2022.7 以前版本)。</p> <p>opcua 特有类型定义如下:</p> <p>char 1 字节整数, 自动转换为实时库 int32</p> <p>byte 1 字节无符号整数, 自动转换为实时库 int32</p> <p>short 2 字节整数, 自动转换为实时库 int32</p> <p>word 2 字节无符号整数, 自动转换为实时库 int32</p> <p>dword 4 字节无符号整数, 自动转换为实时库 int32</p> <p>uint64 8 字节无符号整数, 自动转换为实时库 int64</p> <p>例子 1: 实时库 digital, opcua 是 char digital:char</p> <p>例子 2: 实时库是 float32, opcua 是 word float32:word</p> <p>例子 3: 实时库是 int32, opcua 是 short int32:short</p> <p>注意: 如果是只读标签, 不用配置 opcua 类型, IOServer 会自动转换, 按照以前版本配置即可。</p> <p>只有 rw 标签, 需要向下控制输出的, 才需要精确配置 opcua 类型, 以便不支持自动转换的 opcua server 产生 0x80740000(类型不匹配)错误。</p>
access	访问方式, r 表示读, rw 表示读写, 带有 w 的表示可以写, 会被 ioserver 注册到实时库当作可 device 写方式的标签, 即控制标签, 其他客户端如果有写 device 权限, 即控制权限, 就可向这个设备发送下传控制数据。
Unit	工程单位。
Description	描述

具体参见 `/config/opcuaxltags.csv` 中的配置。注意 csv 文件的第一行和第二行不要改动。

3 从旧版升级迁移

升级时标签表不需要改动，只需改变 `opcua` 的配置，如果熟练，只需要几分钟，但您需要先花点时间用第三方工具对 OPCUA Server 做前期评估测试。

3.1 升级评估测试

确定升级的必要性，测试 OPCUA Server 是否支持证书和安全加密策略。使用第三方的工具测试 Server，推荐如下两个工具：

- **Prosys OPC UA Client**
- **Matrikon OPC UA Explorer**

确定 Server 支持证书或者不支持证书但支持匿名，则可以升级。

3.2 复制文件

`dac_opcuax.dll/dac_opcuax.so` 复制到安装的目标系统的 **drivers** 目录，linux 使用脚本安装后在 `/usr/local/bin/ioserver/drivers`

Windows 则是工程人员安装的目录。

配置文件和标签表和原来一样还是放在 `ioserver/config` 目录下。

3.3 升级到带证书模式

按照 2.1 描述配置证书和安全策略。

3.4 不带证书模式升级

`dac_opcuax` 驱动也支持不带证书模式，但是按照 2.2 描述，大部分 OPCUA Server 只能在证书安全策略模式先才支持用户/账号登录。

先用第三方工具确定 OPCUA Server 支持匿名或者非安全通道下的用户/账号登录模式之一均可。然后按照下面描述配置非安全通道下的客户端配置。

使用匿名方式登录，除了 `endpointUrl` 和 `tagFilename` 字段外，其他字段全部注释掉，如果 OPCUA Server 支持非安全通道的用户/账号登录，可填写正确的用户名和密码，如下例子：

[opcua]

#server 的连接点 URL，完整填写，如下例

endpointUrl = opc.tcp://127.0.0.1:53530/OPCUA/SimulationServer

#通讯层安全策略

#securityPolicy = None

#客户端自己的应用标识

#applicationUri = urn:kipway.client.application

#无客户端证书

#certFilename =

#无客户端私钥

#keyFilename =

#消息加密方式，填写 0-3； 0：无效； 1：无； 2：签名； 3：签名和加密

#msgSecurityMode = 1

#会话层用户验证-用户名；不配置表示匿名

#username =

#会话层用户验证-用户密码

#password =

[tag]

#填写原标签表 csv 文件,自动识别文件内容编码, 支持 utf8 和 gbk

tagFilename = ua1tags.csv

附录 1 制作自签名证书

第一选择(推荐), 从 OPCUA server 管理员处获取应用证书。

其次, 如果条件满足 (主要是 IP 地址), 可以使用本驱动自带的有效期长达 10 年的客户端证书 `kipway_uaclient_cert.der`。

最后选择才是自己制作证书。

本驱动附带了一个 python3 脚本, 从 open62541 项目修改而来, 去掉了 python3 组件 `netifaces` 的依赖, 方便使用。

选择 ubuntu20.04 或者 22.04 桌面版系统, 开发工作的最佳 Linux 平台, 默认安装常用工具比如 `perl`, `openssl`, `python3` 等, 方便直接使用。

将 `uacert` 目录复制到目标 ubuntu 系统。

使用 `vim` 修改 python3 脚本 `create_self-signed.py` 文件

修改其中的 DNS2 和 IP 列表, 如下图。

```
#set IP address
os.environ['IPADDRESS1'] = "127.0.0.1"
os.environ['IPADDRESS2'] = "192.168.1.125"
os.environ['IPADDRESS3'] = "192.168.1.36"
os.environ['IPADDRESS4'] = "192.168.1.19"
os.environ['IPADDRESS5'] = "192.168.1.59"
os.environ['IPADDRESS6'] = "192.168.1.233"
os.environ['IPADDRESS7'] = "192.168.1.230"

os.environ['HOSTNAME'] = socket.gethostname()

#填写其他的DNS2
os.environ['HOSTNAME2'] = "kipway.net"
openssl_conf = os.path.join(certsdire, "localhost.cnf")

os.chdir(os.path.abspath(args.outdir))
```

然后执行

```
./create_self-signed.py -c myuaclient
```

其中 `myuaclient` 是证书名不需要加扩展名, 改成自己喜欢的名字。

证书测试, 推荐使用 **Prosys OPC UA Simulation Server** 作为 OPCUA Server 端工具。

```
./create_self-signed.py -u urn:kipway.server.application -c myuaclient
```