basysKom

QT OPC UA

Industrial integration out of the box

Qt World Summit Berlin 2019

AGENDA

- Intro
- OPC UA on three slides 😁
- Qt OPC UA – overview
- Tutorial
- A quick glance on the QML API
- Wrap up

INTRO

About basysKom

basysKom GmbH

- is a Qt Service Partner since 2004
- is located in Darmstadt and Nürnberg
- is employing ~30 people
- is part of the UX Gruppe
- provides services (consulting, training, coaching and development) around Qt
- focuses on technical/industrial applications of Qt (HMI and application development)
- initially developed and upstreamed Qt OPC UA

About myself

- Development Lead @ basysKom
- Maintainer of Qt OPC UA

OPC UA

on three slides...

What is OPC UA?

M2M Communication protocol / framework

Core application areas

- Industrial automation (manufacturing)
- Process control
- Applied in more and more areas
  (see: opcfoundation.org/markets-collaboration/)

Standardized by the OPC Foundation

Freely available

Numerous implementations

What is OPC UA?

Client/Server

- Servers provide access to a fixed set of standardised services
- Clients use these to access/manipulate „objects“ on the server
- The same process/device can act as client or server at the same time (on different connections!)

Examples for servers

- Sensors
- Embedded-Devices/PLCs
- OPC UA aggregators/protocol bridges
- IT (ERP, MES, ...)

Examples for clients

- HMI & visualisation
- Applications (desktop/mobile)
- Embedded-Devices/PLCs
- IT (ERP, MES, ...)

Data modeling

Address space

- Provides access to the „data objects“ on the server
- Contains a graph made up of „Nodes“ and „References“

Nodes

- 7 different types of nodes (Variable, Object, Method, ...)
- Contain attributes (browsername, displayname, value, node id, ...)

References

- Connect nodes
- Have a semantic (IsSubType, Organizes, HasComponent, ...)

The server contains data and meta data

OPC UA provides the means to create complex data models

- Especially interesting in combination with Companion Specifications (Woodworking, CNC machines, ...)

QT OPC UA - OVERVIEW

Mission statement Qt OPC UA

Make it easy to use OPC UA services from Qt

Focus on HMIs

- Client API only
- Be Qt-ish
- Asynchronous
- Easy to use
- Usability over (ultimate) performance

Qt OPC UA is an API, not a stack

Wraps an existing stack

- open62541 (MPLv2)
- Unified Automation C++ SDK (commercial)

Licensing & Installation

Triple licensed

FOSS

- GPLv2
- LGPLv3

Commercial

- Qt Commercial License

Availability

- „Qt for Automation“
- code.qt.io/cgit/qt/qtopcua.git/

Installation

- doc-snapshots.qt.io/qtopcua/
- blog.basyskom.com/building-qt-opc-ua-with-open62541/

Basics

QOpcUaClient

- Discovery of servers and endpoints
- Connection management
- Factory for QOpcUaNodes
- Batch read/write operations

QOpcUaNode

- Representation of a „Node“ on the server
- Access to attributes of the node
- Monitoring of value changes on the server
- Remote method calls
- Browsing

basysKom

05.11.2019

FRANK MEERKÖTTER

QT OPC UA – DISCOVERY & CONNECTION MANAGEMENT

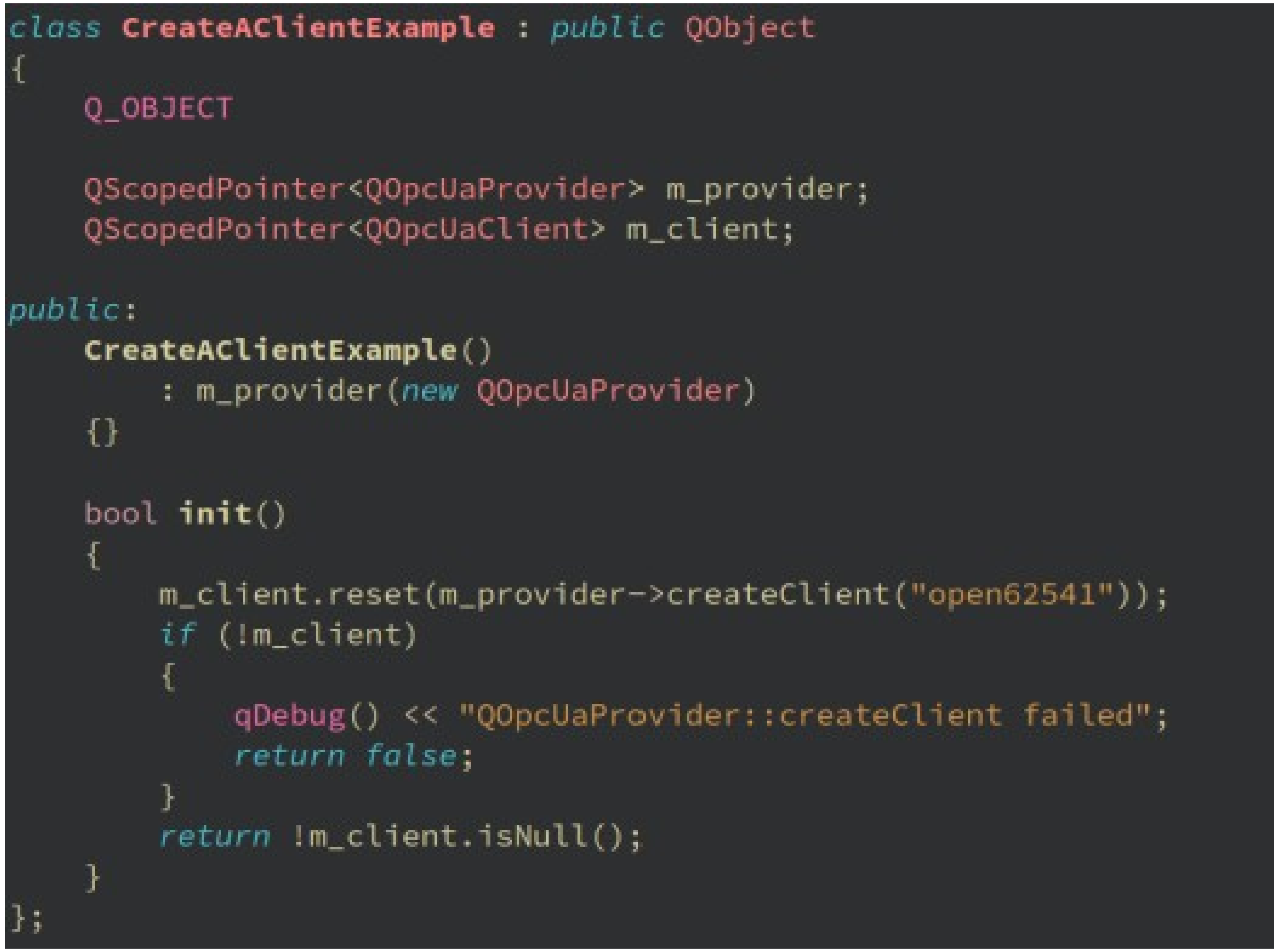
How to create a client?

Where do I get one?

- QOpcUaProvider
- A factory to instantiate backends

Example

- 000_creating_a_client



```
class CreateAClientExample : public QObject
{
    Q_OBJECT

    QScopedPointer<QOpcUaProvider> m_provider;
    QScopedPointer<QOpcUaClient> m_client;

public:
    CreateAClientExample()
        : m_provider(new QOpcUaProvider)
    {}

    bool init()
    {
        m_client.reset(m_provider->createClient("open62541"));
        if (!m_client)
        {
            qDebug() << "QOpcUaProvider::createClient failed";
            return false;
        }
        return !m_client.isNull();
    }
};
```

How to connect to a server?

QOpcUaClient::connectToEndpoint() which takes an QOpcUaEndpointDescription

An endpoint is an address a client can connect to

A server can provide several endpoints

Endpoints are defined by a combination of server URL, protocol, accepted UserIdentityToken, MessageSecurityMode and SecurityPolicy

- e.g. opc.tcp://plc4711.basyskom.com:3412 with username/password, SignAndEncrypt (Mode) and http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256 (Policy)

How to get an QOpcUaEndpointDescription?

OPC UA defines several discovery models

- See OPC UA part 12

Simple discovery

- We know the discovery URL of a server

Local discovery

- We know the URL of an LDS (local discovery server)

Not supported with Qt OPC UA

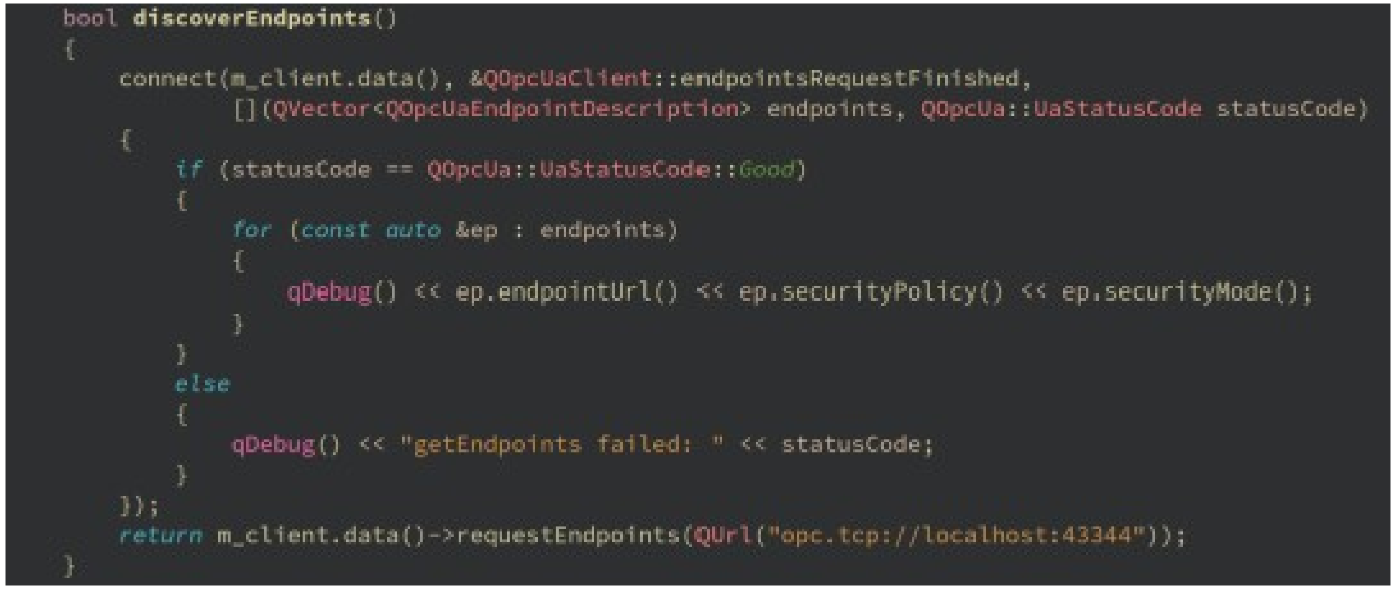
- Multicast discovery

Not yet(!) supported with Qt OPC UA

- GDS (global discovery server)
- Ongoing work by The Qt Company

Simple discovery

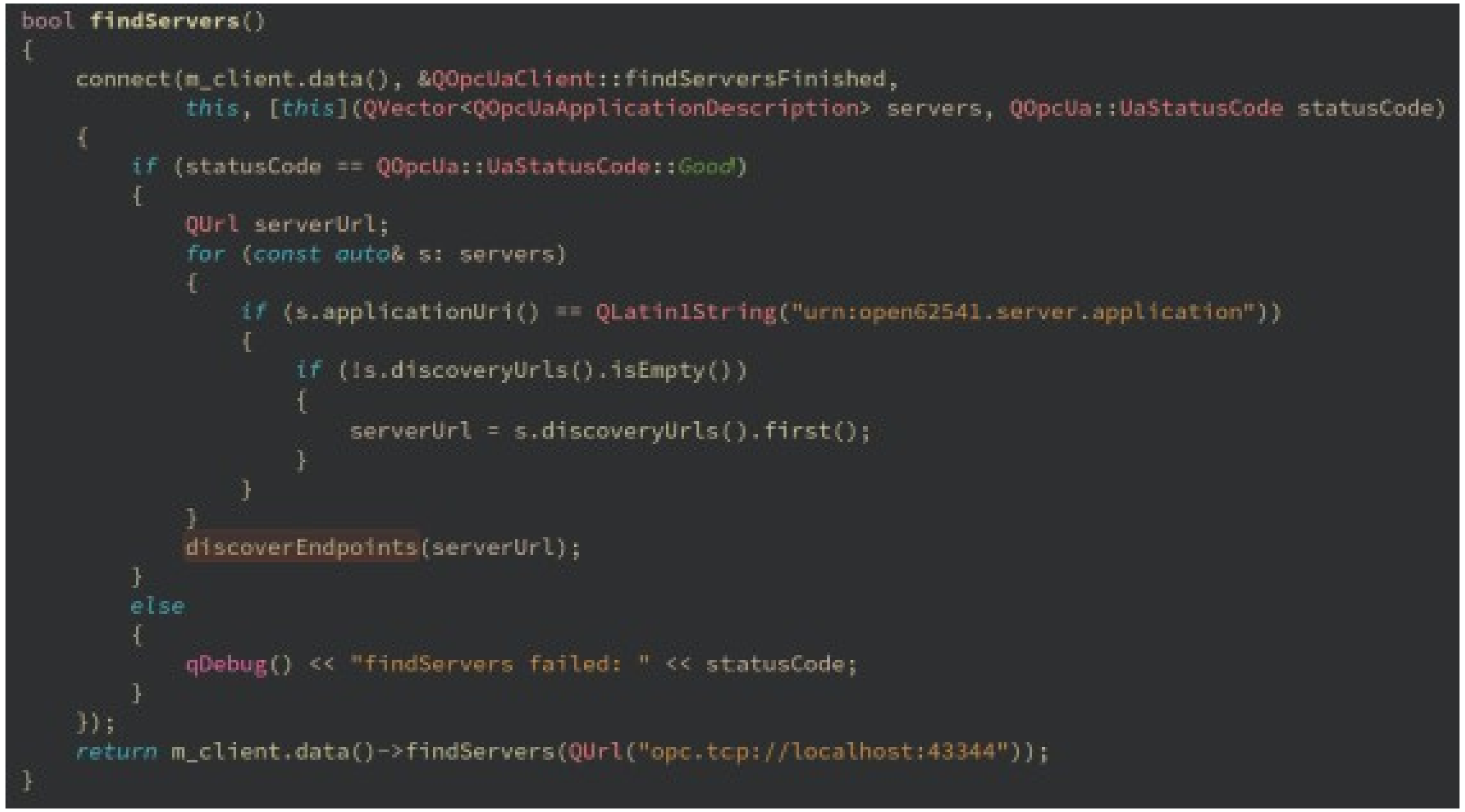
- We know the discovery URL of server
- Call QOpcUaClient::requestEndpoints()
- Wait for the endPointsRequestFinished() signal



```
bool discoverEndpoints()
{
    connect(m_client.data(), &QOpcUaClient::endpointsRequestFinished,
            [](QVector<QOpcUaEndpointDescription> endpoints, QOpcUa::UaStatusCode statusCode)
    {
        if (statusCode == QOpcUa::UaStatusCode::Good)
        {
            for (const auto &ep : endpoints)
            {
                qDebug() << ep.endpointUrl() << ep.securityPolicy() << ep.securityMode();
            }
        }
        else
        {
            qDebug() << "getEndpoints failed: " << statusCode;
        }
    });
    return m_client.data()->requestEndpoints(QUrl("opc.tcp://localhost:43344"));
}
```

Local discovery

- We know the URL of a discovery server
- Call QOpcUaClient::findServers()
- Wait for the findServersRequestFinished() signal
- Use the result to call QOpcUaClient::requestEndpoints



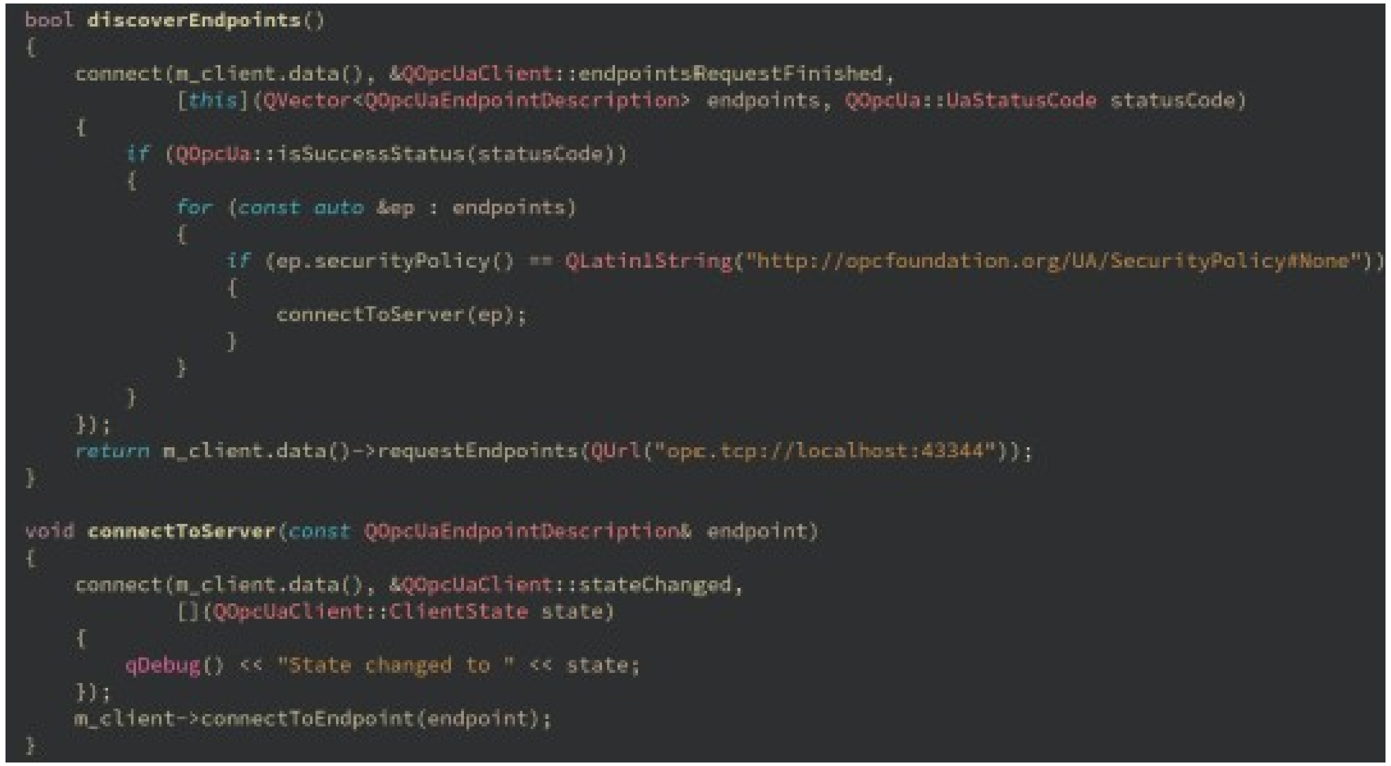
```
bool findServers()
{
    connect(m_client.data(), &QOpcUaClient::findServersFinished,
            this, [this](QVector<QOpcUaApplicationDescription> servers, QOpcUa::UaStatusCode statusCode)
    {
        if (statusCode == QOpcUa::UaStatusCode::Good)
        {
            QUrl serverUrl;
            for (const auto& s: servers)
            {
                if (s.applicationUri() == QLatin1String("urn:open62541.server.application"))
                {
                    if (!s.discoveryUrls().isEmpty())
                    {
                        serverUrl = s.discoveryUrls().first();
                    }
                }
            }
            discoverEndpoints(serverUrl);
        }
        else
        {
            qDebug() << "findServers failed: " << statusCode;
        }
    });
    return m_client.data()->findServers(QUrl("opc.tcp://localhost:43344"));
}
```

Finally! How to connect to a server?

Call QOpcUaClient::connectToEndpoint()

Wait for either signal

- StateChanged(QOpcUaClient::ClientState)
- Connected()



```cpp
bool discoverEndpoints()
{
    connect(m_client.data(), &QOpcUaClient::endpointsRequestFinished,
            [this](QVector<QOpcUaEndpointDescription> endpoints, QOpcUa::UaStatusCode statusCode)
    {
        if (QOpcUa::isSuccessStatus(statusCode))
        {
            for (const auto &ep : endpoints)
            {
                if (ep.securityPolicy() == QLatin1String("http://opcfoundation.org/UA/SecurityPolicy#None"))
                {
                    connectToServer(ep);
                }
            }
        }
    });
    return m_client.data()->requestEndpoints(QUrl("opc.tcp://localhost:43344"));
}

void connectToServer(const QOpcUaEndpointDescription& endpoint)
{
    connect(m_client.data(), &QOpcUaClient::stateChanged,
            [](QOpcUaClient::ClientState state)
    {
        qDebug() << "State changed to " << state;
    });
    m_client->connectToEndpoint(endpoint);
}
```

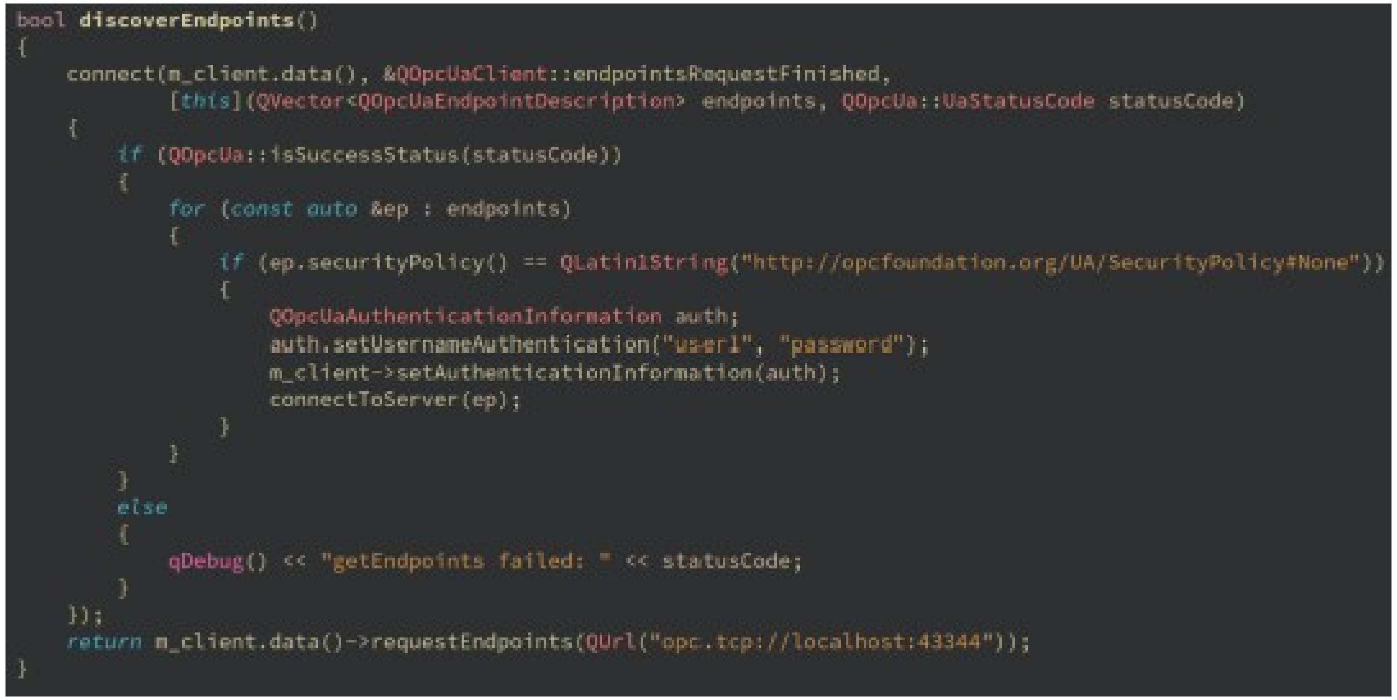
How to connect to a server that requires authentification?

Call QOpcUaClient::

- SetAuthentificationInformation() with a
- QOpcUaAuthentificationInformation object

Default is

- QOpcUaTokenPolicy::Anonymous

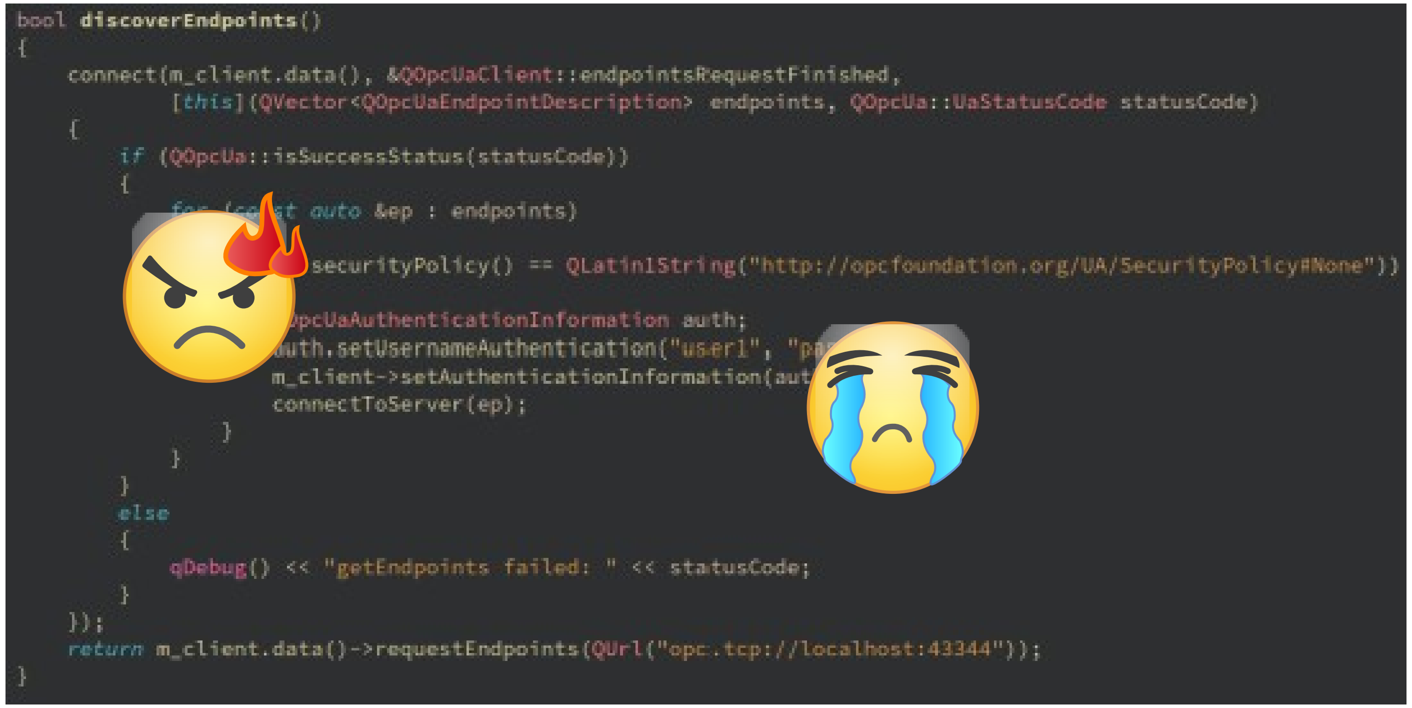


```
bool discoverEndpoints()
{
    connect(m_client.data(), &QOpcUaClient::endpointsRequestFinished,
            [this](QVector<QOpcUaEndpointDescription> endpoints, QOpcUa::UaStatusCode statusCode)
    {
        if (QOpcUa::isSuccessStatus(statusCode))
        {
            for (const auto &ep : endpoints)
            {
                if (ep.securityPolicy() == QLatin1String("http://opcfoundation.org/UA/SecurityPolicy#None"))
                {
                    QOpcUaAuthenticationInformation auth;
                    auth.setUsernameAuthentication("user1", "password");
                    m_client->setAuthenticationInformation(auth);
                    connectToServer(ep);
                }
            }
        }
        else
        {
            qDebug() << "getEndpoints failed: " << statusCode;
        }
    });
    return m_client.data()->requestEndpoints(QUrl("opc.tcp://localhost:43344"));
}
```

Wait?!! Wait, wait!

Isn‘t that terrible insecure?

- Yes, it is!
- Still - there are controllers out there that work like that (or are set up to not authenticate at all)



```
bool discoverEndpoints()
{
    connect(m_client.data(), &QOpcUaClient::endpointsRequestFinished,
            [this](QVector<QOpcUaEndpointDescription> endpoints, QOpcUa::UaStatusCode statusCode)
    {
        if (QOpcUa::isSuccessStatus(statusCode))
        {
            for (const auto &ep : endpoints)
                  securityPolicy() == QLatin1String("http://opcfoundation.org/UA/SecurityPolicy#None"))

                 pcUaAuthenticationInformation auth;
                 uth.setUsernameAuthentication("user1", "pa
                m_client->setAuthenticationInformation(aut
                connectToServer(ep);
            }
        }
    }
    else
    {
        qDebug() << "getEndpoints failed: " << statusCode;
    }
    });
    return m_client.data()->requestEndpoints(QUrl("opc.tcp://localhost:43344"));
}
```

How to connect to a secure endpoint?

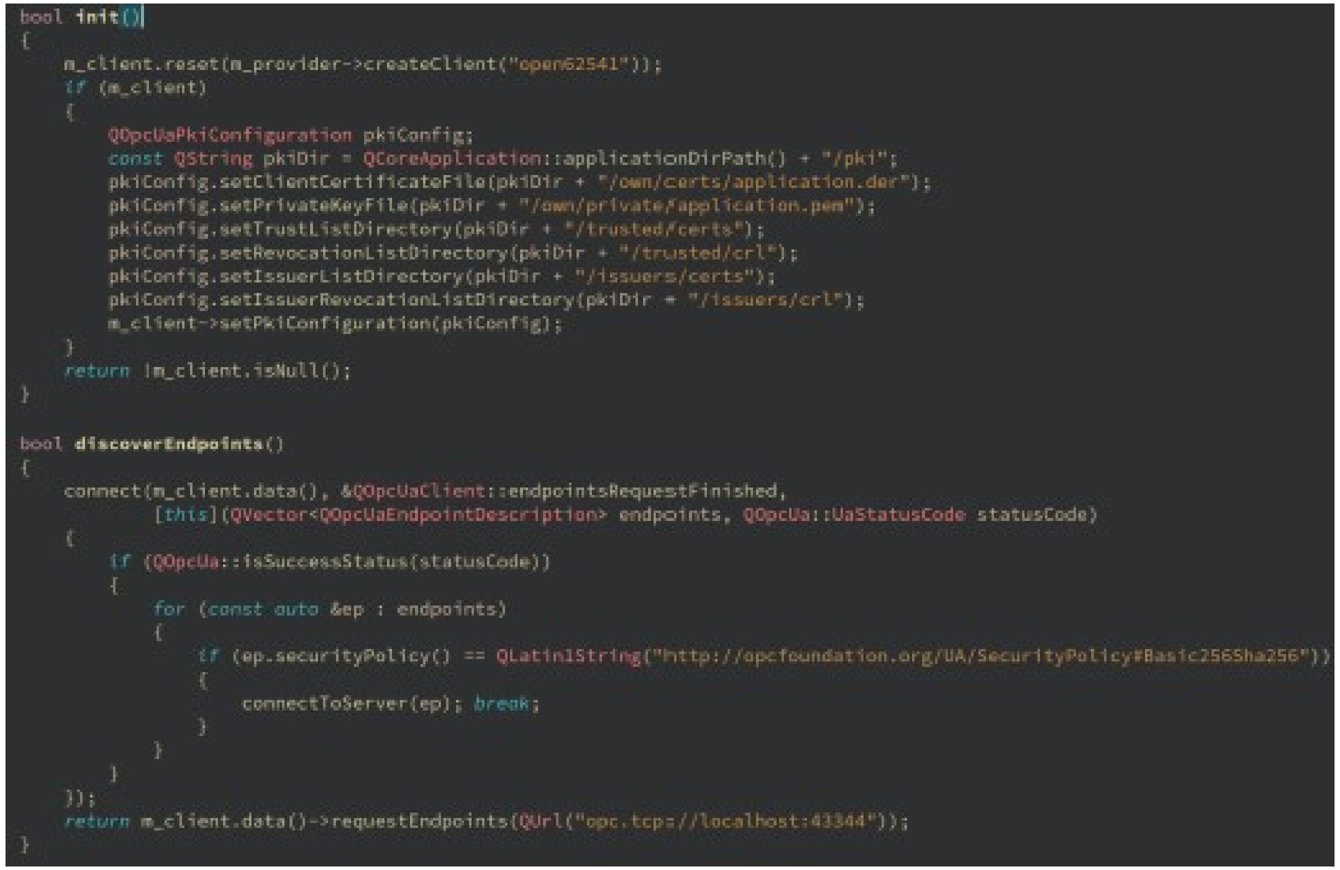
OPC UA defines transport security

Requires installation of a CA-infrastructure (or distribution of self-signed client/server certificates)

Servers should offer endpoints with mode Sign&Encrypt and an up-to-date security policy

- See QOpcUaEndpointDescription::securityLevel()

Call QOpcUaClient::setPkiConfiguration()



```
bool init()
{
    m_client.reset(m_provider->createClient("open62541"));
    if (m_client)
    {
        QOpcUaPkiConfiguration pkiConfig;
        const QString pkiDir = QCoreApplication::applicationDirPath() + "/pki";
        pkiConfig.setClientCertificateFile(pkiDir + "/own/certs/application.der");
        pkiConfig.setPrivateKeyFile(pkiDir + "/own/private/application.pem");
        pkiConfig.setTrustListDirectory(pkiDir + "/trusted/certs");
        pkiConfig.setRevocationListDirectory(pkiDir + "/trusted/crl");
        pkiConfig.setIssuerListDirectory(pkiDir + "/issuers/certs");
        pkiConfig.setIssuerRevocationListDirectory(pkiDir + "/issuers/crl");
        m_client->setPkiConfiguration(pkiConfig);
    }
    return !m_client.isNull();
}

bool discoverEndpoints()
{
    connect(m_client.data(), &QOpcUaClient::endpointsRequestFinished,
            [this](QVector<QOpcUaEndpointDescription> endpoints, QOpcUa::UaStatusCode statusCode)
    {
        if (QOpcUa::isSuccessStatus(statusCode))
        {
            for (const auto &ep : endpoints)
            {
                if (ep.securityPolicy() == QLatin1String("http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256"))
                {
                    connectToServer(ep); break;
                }
            }
        }
    });
    return m_client.data()->requestEndpoints(QUrl("opc.tcp://localhost:43344"));
}
```

QT OPC UA – ACCESSING DATA

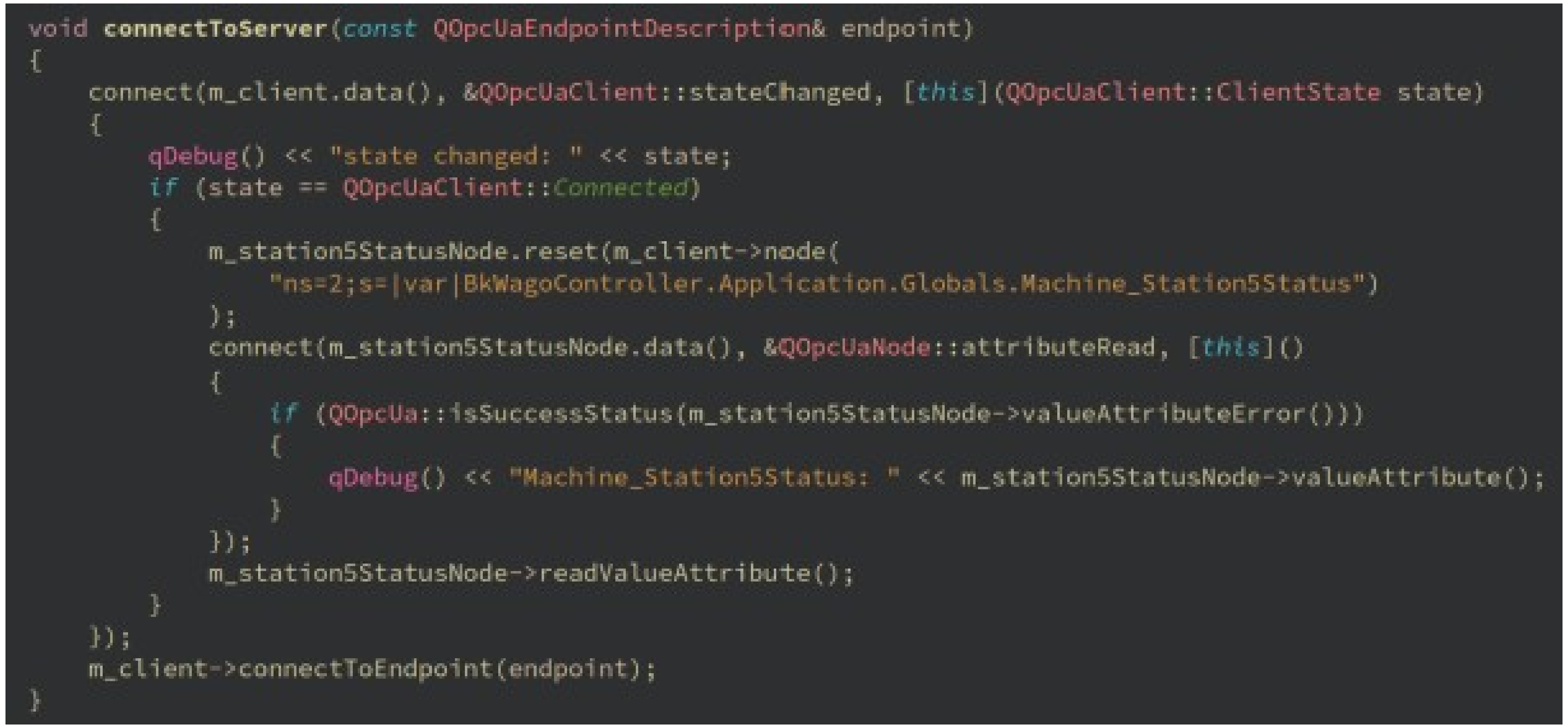
How to read a value from the server?

Call QOpcUaClient::node() with a node id to retrieve a QOpcUaNode

QOpcUaNode represents a node on the server

Node object needs to be updated before accessing its attributes

- Call readAttributes()
- Wait for completion signal
- Check the attribute status code!



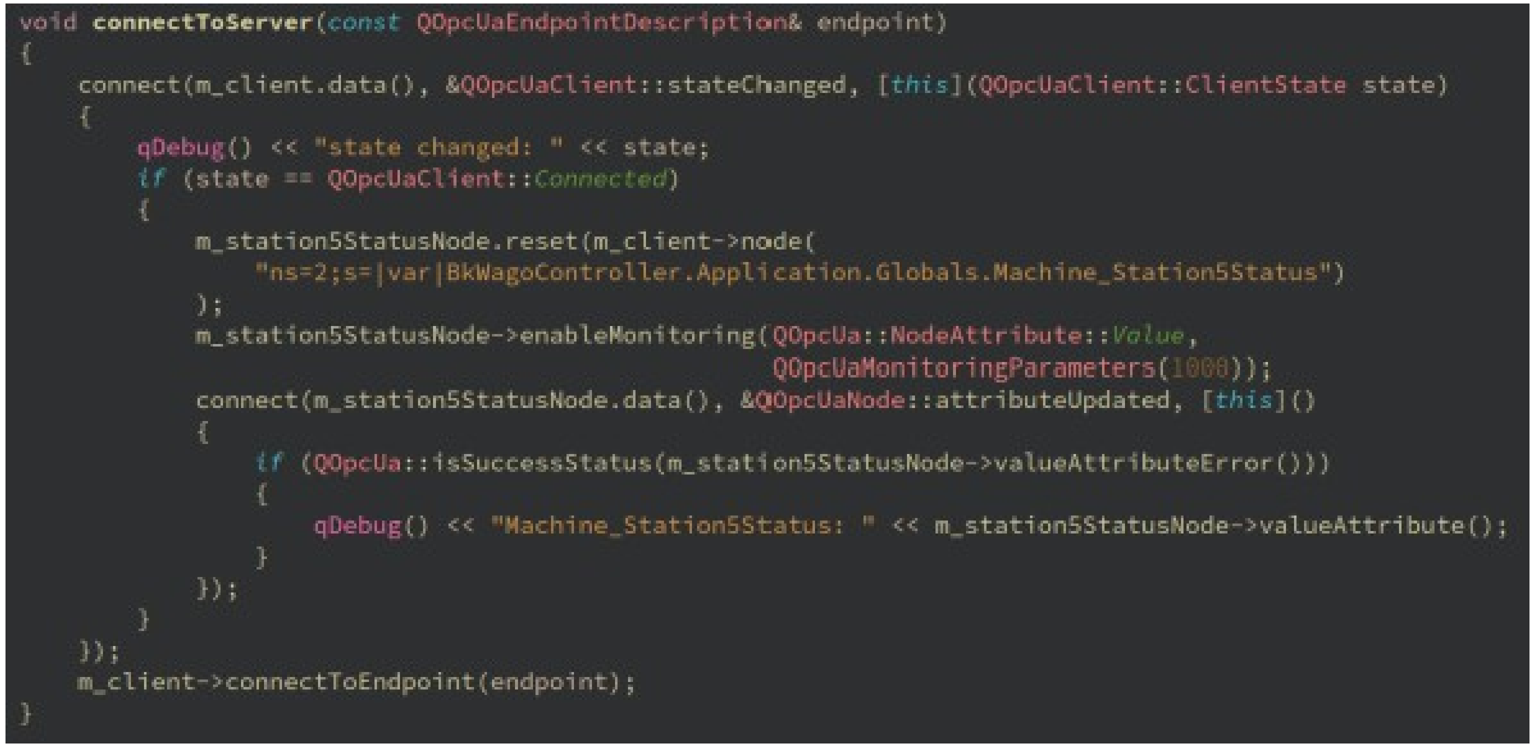
```
void connectToServer(const QOpcUaEndpointDescription& endpoint)
{
    connect(m_client.data(), &QOpcUaClient::stateChanged, [this](QOpcUaClient::ClientState state)
    {
        qDebug() << "state changed: " << state;
        if (state == QOpcUaClient::Connected)
        {
            m_station5StatusNode.reset(m_client->node(
                "ns=2;s=|var|BkWagoController.Application.Globals.Machine_Station5Status")
            );
            connect(m_station5StatusNode.data(), &QOpcUaNode::attributeRead, [this]()
            {
                if (QOpcUa::isSuccessStatus(m_station5StatusNode->valueAttributeError()))
                {
                    qDebug() << "Machine_Station5Status: " << m_station5StatusNode->valueAttribute();
                }
            });
            m_station5StatusNode->readValueAttribute();
        }
    });
    m_client->connectToEndpoint(endpoint);
}
```

How to monitor a value for data changes?

OPC UA offers MonitoredItems and Subscriptions

Qt OPC UA abstracts them via QOpcUaNode::enableMonitoring()

- See QOpcUaMonitoringParameter



```
void connectToServer(const QOpcUaEndpointDescription& endpoint)
{
    connect(m_client.data(), &QOpcUaClient::stateChanged, [this](QOpcUaClient::ClientState state)
    {
        qDebug() << "state changed: " << state;
        if (state == QOpcUaClient::Connected)
        {
            m_station5StatusNode.reset(m_client->node(
                "ns=2;s=|var|BkWagoController.Application.Globals.Machine_Station5Status")
            );
            m_station5StatusNode->enableMonitoring(QOpcUa::NodeAttribute::Value,
                                                   QOpcUaMonitoringParameters(1000));
            connect(m_station5StatusNode.data(), &QOpcUaNode::attributeUpdated, [this]()
            {
                if (QOpcUa::isSuccessStatus(m_station5StatusNode->valueAttributeError()))
                {
                    qDebug() << "Machine_Station5Status: " << m_station5StatusNode->valueAttribute();
                }
            });
        }
    });
    m_client->connectToEndpoint(endpoint);
}
```

Issues?

Due to hard-code/assume namespace indices

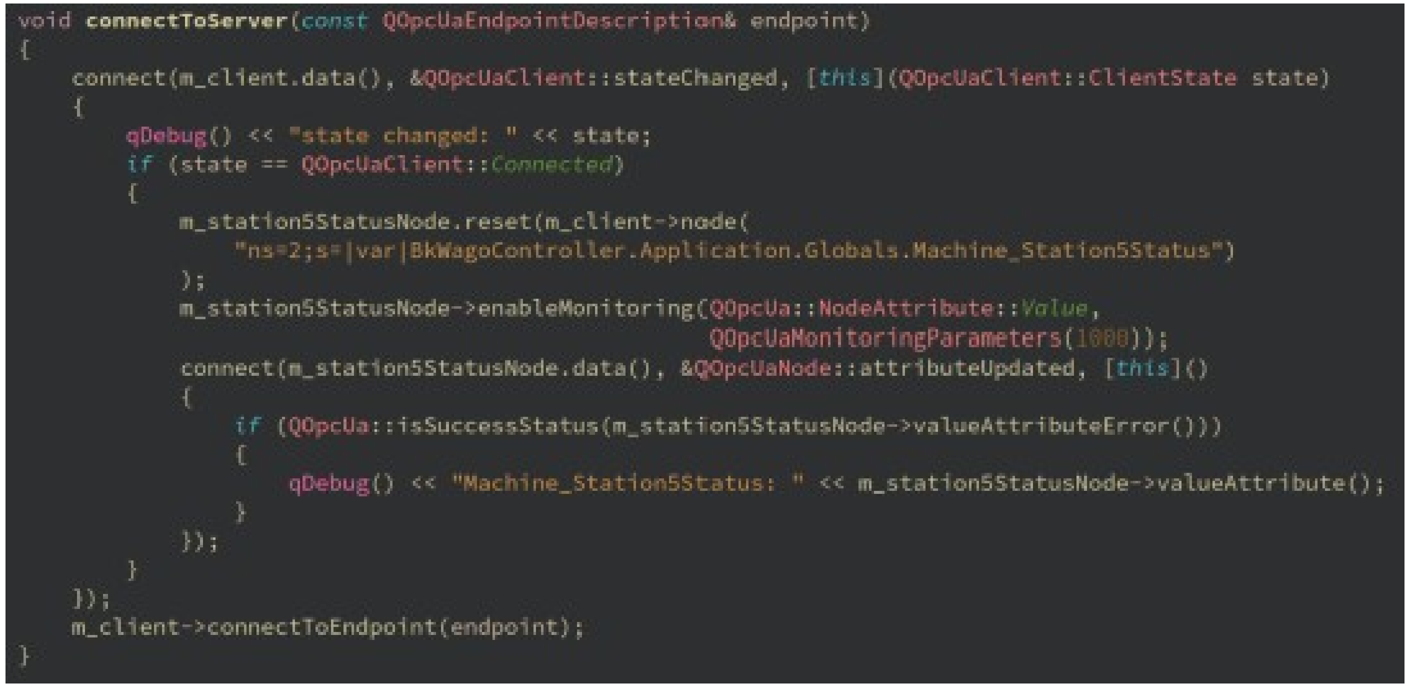
- See QOpcUaClient::nameSpaceArray()

Node ids might not be stable

- Prefer to program against a type definition
- See QOpcUaNode::resolveBrowsePath

Inconvenient/slow to poll a large amount of nodes

- See QOpcUaClient::readNodeAttributes



```
void connectToServer(const QOpcUaEndpointDescription& endpoint)
{
    connect(m_client.data(), &QOpcUaClient::stateChanged, [this](QOpcUaClient::ClientState state)
    {
        qDebug() << "state changed: " << state;
        if (state == QOpcUaClient::Connected)
        {
            m_station5StatusNode.reset(m_client->node(
                "ns=2;s=|var|BkWagoController.Application.Globals.Machine_Station5Status")
            );
            m_station5StatusNode->enableMonitoring(QOpcUa::NodeAttribute::Value,
                                                   QOpcUaMonitoringParameters(1000));
            connect(m_station5StatusNode.data(), &QOpcUaNode::attributeUpdated, [this]()
            {
                if (QOpcUa::isSuccessStatus(m_station5StatusNode->valueAttributeError()))
                {
                    qDebug() << "Machine_Station5Status: " << m_station5StatusNode->valueAttribute();
                }
            });
        }
    });
    m_client->connectToEndpoint(endpoint);
}
```

QT OPC UA - QML API

Wasn‘t Qt OPC UA about HMI?!

Standard Qt approach to connect an HMI to a data backend

- QObjects with properties exposed to QML
- Models

Completely valid approach for Qt OPC UA applications

Qt OPC UA also offers an API where no C++ needs to be written for OPC UA access

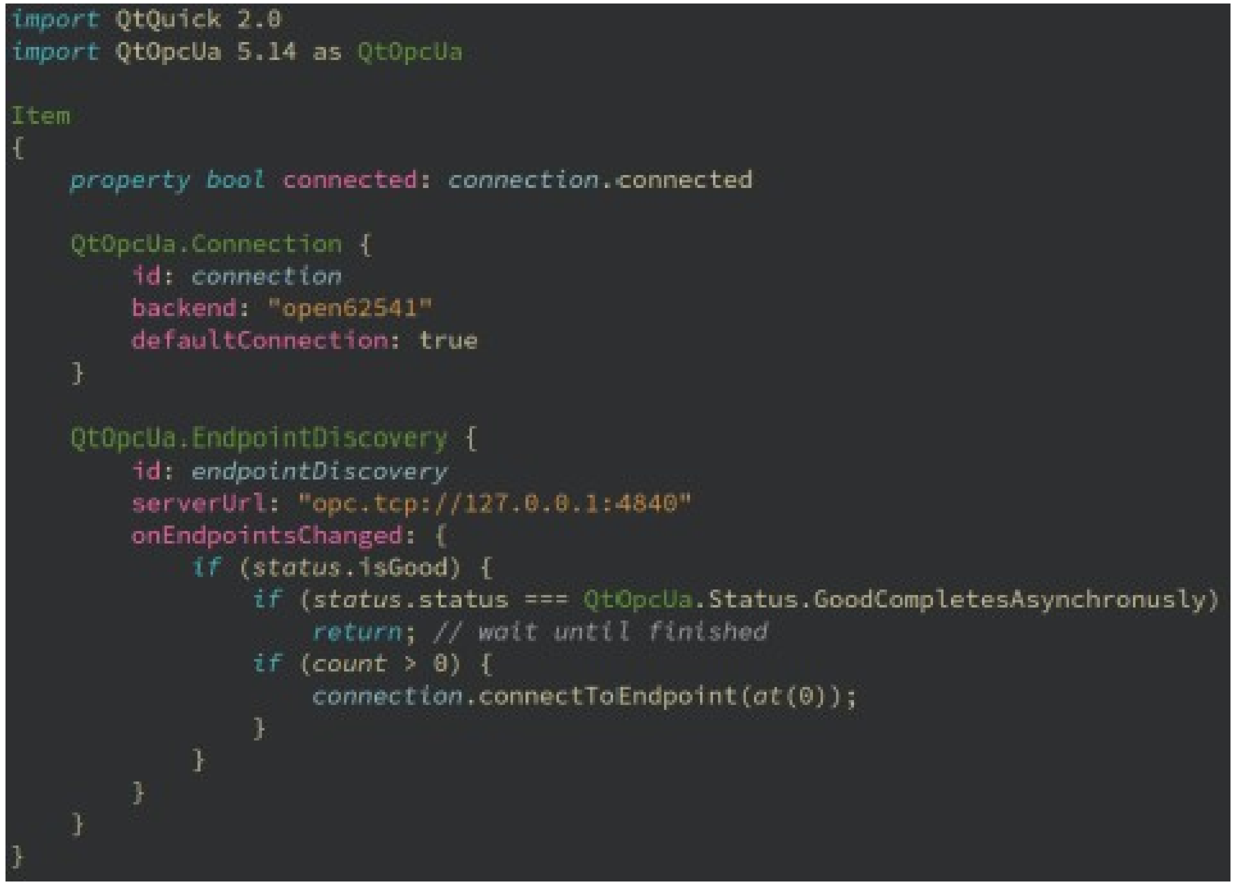
Simple discovery using the QML API

Connection is roughly equivalent to the QOpcUaClient

- Backend property „hides“ the QOpcUaProvider

Supports simple discovery via EndpointDiscovery

Supports local discovery via ServerDiscovery

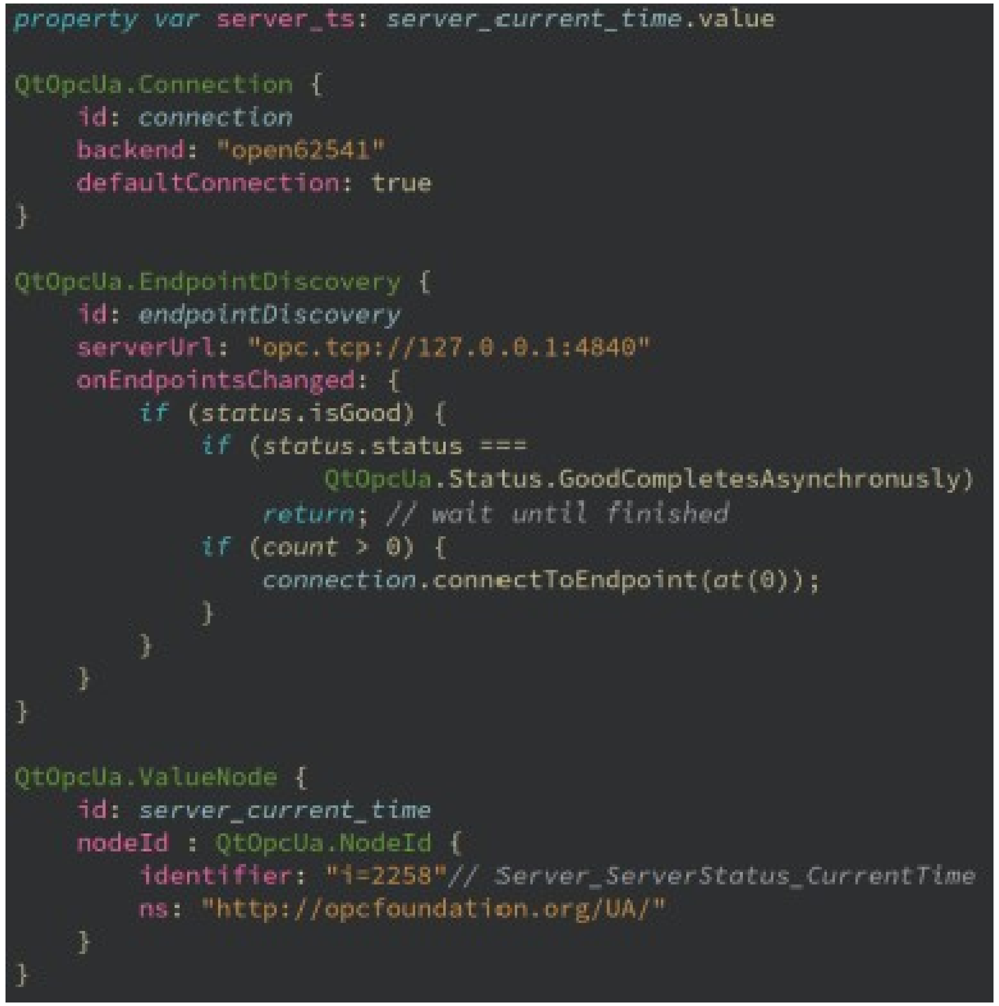


```
import QtQuick 2.0
import QtOpcUa 5.14 as QtOpcUa

Item
{
    property bool connected: connection.connected

    QtOpcUa.Connection {
        id: connection
        backend: "open62541"
        defaultConnection: true
    }

    QtOpcUa.EndpointDiscovery {
        id: endpointDiscovery
        serverUrl: "opc.tcp://127.0.0.1:4840"
        onEndpointsChanged: {
            if (status.isGood) {
                if (status.status === QtOpcUa.Status.GoodCompletesAsynchronously)
                    return; // wait until finished
                if (count > 0) {
                    connection.connectToEndpoint(at(0));
                }
            }
        }
    }
}
```

Accessing values using QML

ValueNode

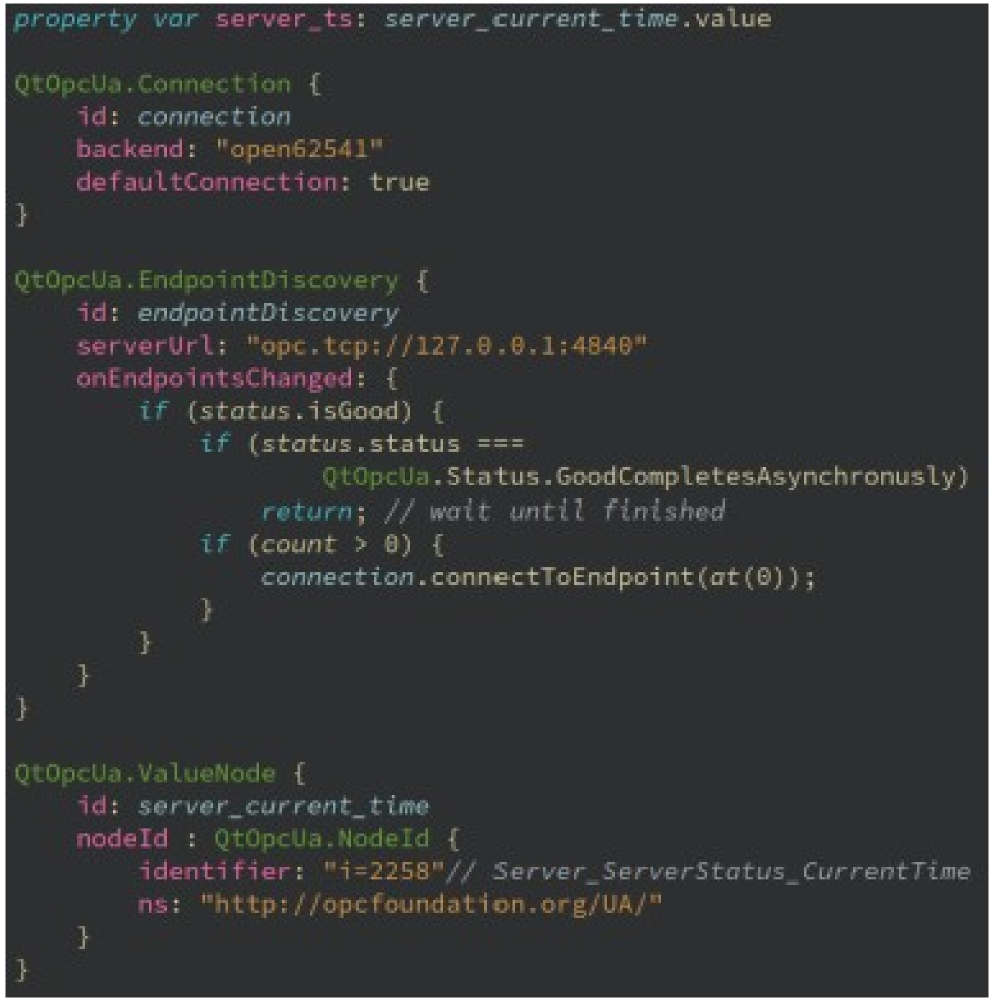
- Convenient access to an automatically updated value on the server
- Can be used directly in property bindings



```
property var server_ts: server_current_time.value

QtOpcUa.Connection {
    id: connection
    backend: "open62541"
    defaultConnection: true
}

QtOpcUa.EndpointDiscovery {
    id: endpointDiscovery
    serverUrl: "opc.tcp://127.0.0.1:4840"
    onEndpointsChanged: {
        if (status.isGood) {
            if (status.status ===
                    QtOpcUa.Status.GoodCompletesAsynchronusly)
                return; // wait until finished
            if (count > 0) {
                connection.connectToEndpoint(at(0));
            }
        }
    }
}

QtOpcUa.ValueNode {
    id: server_current_time
    nodeId : QtOpcUa.NodeId {
        identifier: "i=2258"// Server_ServerStatus_CurrentTime
        ns: "http://opcfoundation.org/UA/"
    }
}
```

QML Demo

Demo time!



```
property var server_ts: server_current_time.value

QtOpcUa.Connection {
    id: connection
    backend: "open62541"
    defaultConnection: true
}

QtOpcUa.EndpointDiscovery {
    id: endpointDiscovery
    serverUrl: "opc.tcp://127.0.0.1:4840"
    onEndpointsChanged: {
        if (status.isGood) {
            if (status.status ===
                    QtOpcUa.Status.GoodCompletesAsynchronusly)
                return; // wait until finished
            if (count > 0) {
                connection.connectToEndpoint(at(0));
            }
        }
    }
}

QtOpcUa.ValueNode {
    id: server_current_time
    nodeId : QtOpcUa.NodeId {
        identifier: "i=2258"// Server_ServerStatus_CurrentTime
        ns: "http://opcfoundation.org/UA/"
    }
}
```

CONCLUSION

There is more!

Qt OPC UA allows you to

- browse the server address space
- call remote methods
- monitor for OPC UA Events
- configure DataChangeFilters
- access slices of a server side array
- write attributes
- add/remove references and nodes
- access ExtensionObjects
- ...

Come talk to us about your OPC UA use case

- basysKom is offering Qt OPC UA and open62541 trainings
- Meet us at the basysKom booth

THANK YOU!

QUESTIONS?