OPC UA SDK
Client + Server Development

Server Development Einführung

Getestet? Du willst es?

Lizenzmodell Preise Angebot Jetzt bestellen

Der Server

Start

Das passiert beim Aufruf von 'Start':

1. Es wird geprüft, ob eine **Adresse festgelegt** wurde (Addresseigenschaft).
2. Der Server ändert seinen Status (**OpcServer.State** Eigenschaft) auf den Wert ***Starting***.
3. Der Server **prüft seine Konfiguration** auf Gültigkeit und Schlüssigkeit.
4. Anschließend versucht der Server **für jede Endpunktbeschreibung einen Host** zu **erstellen**.
5. Darauf folgt der **Start aller Manager** (NodeManager, SessionManager, ...)
6. Schließlich wird **jeder** der erstellten **Hosts gestartet**, welche für die Endpunkt-spezifische Kommunikation mit den Clients zuständig sind.
7. Der Server ändert seinen Status (**OpcServer.State** Eigenschaft) auf den Wert ***Started***.

Stop

Das passiert beim Aufruf von 'Stop':

1. Der Server ändert seinen Status (**OpcServer.State** Eigenschaft) auf den Wert ***Stopping***.
2. **Beendet alle Manager** (NodeManager, SessionManager, ...)
3. Der Server **gibt** alle **erworbenen Ressourcen** wieder **frei**.
4. **Beendet** die **Hosts der Endpunktbeschreibungen**.
5. Der Server ändert seinen Status (**OpcServer.State** Eigenschaft) auf den Wert ***Stopped***.

Parameter

Damit der Server den Clients Zugang zu seinen OPC UA Services geben kann, müssen die richtigen Parameter festgelegt werden. **Generell** wird die **Adresse des Servers** (**OpcServer.Address** Eigenschaft) **benötigt**, unter der er zu erreichen ist. Die Uri-Instanz (= Uniform Resource Identifier) liefert allen Clients die primär nötigen Informationen über den Server. So enthält zum Beispiel die Serveradresse „opc.tcp://192.168.0.80:4840“ die Information des Schemas „opc.tcp“ (möglich sind „http“, „https“, „opc.tcp“, „net.tcp“ und „net.pipe“), welches festlegt, über welches Protokoll die Daten wie ausgetauscht werden sollen. Generell ist bei OPC UA Servern im lokalen Netzwerk „opc.tcp“ zu empfehlen. Server außerhalb des lokalen Netzwerks sollten „http“, besser noch „https“ verwenden. Weiter definiert die Adresse, dass der Server auf dem Rechner mit der IP Adresse „192.168.0.80“ ausgeführt wird und auf Anfragen über den Port mit der Nummer „4840“ lauscht (was der Standardport für OPC UA ist, benutzerdefinierte Portnummern sind ebenfalls möglich). Anstelle der statischen IP Adresse kann auch der DNS Name des Rechners verwendet werden, so könnte anstelle von „127.0.0.1“ auch „localhost“ verwendet werden.

Soll der Server **keinen Endpunkt** (engl. Endpoint), dessen Strategie (engl. Policy) den Sicherheitsmodus **„None“** (möglich sind zudem „Sign“ und „SignAndEncrypt“) hat, zum Datenaustausch bereitstellen, **muss mindestens eine Endpoint-Policy** manuell **konfiguriert werden** (**OpcServer.Security.EndpointPolicies** Eigenschaft). Wird hingegen ein **Endpunkt mit der Strategie „None“** vom Server bereitgestellt, kann ein Client diesen automatisch für eine einfache und schnelle Verbindung auswählen. Wird bei der Definition der Endpunkte den einzelnen Endpoint-Policies ein Policy-Level (eine Zahl) **gemäß OPC Foundation zugewiesen** können Clients diese auch entsprechend

handhaben. Dabei sieht die OPC Foundation vor, dass je höher der Level der Policy eines Endpunktes ist, desto „besser“ ist der entsprechende Endpunkt (zu beachten ist, dass das lediglich eine Richtlinie ist, die niemand weder prüft, noch durchsetzt).

Soll der Server eine Zugriffssteuerung verwenden, zum Beispiel über eine ACL (= Access Control List), also eine Zugriffssteuerungsliste, **müssen die Benutzerdaten zur Feststellung der möglichen/gültigen Identitäten der Benutzer** des Servers festgelegt werden (dies funktioniert auch im laufenden Betrieb). Hierbei besteht die Möglichkeit, die Identitäten der Benutzer über ein Benutzername-Passwort-Paar (**OpcServerIdentity** Klasse) oder über ein Zertifikat (**OpcCertificateIdentity** Klasse) festzulegen. Die entsprechenden Identitäten müssen dann **dem Server mitgeteilt werden** (**OpcServer.Security.UserNameAcl/CertificateAcl** Eigenschaft). Diese Zugriffskontrollisten müssen, damit sie vom Server berücksichtigt werden, aktiviert werden (**OpcServer.Security.UserNameAcl/CertificateAcl.IsEnabled** Eigenschaft).

Endpunkte

Die Endpunkte (engl. Endpoints) ergeben sich aus dem Kreuzprodukt der verwendeten Basis-Adressen des Servers und der vom Server unterstützten Sicherheitsstrategien. Dabei ergeben sich die Basis-Adressen aus jedem unterstützten Schema-Port-Paar, wobei mehrere Schemen (möglich sind „http“, „https“, „opc.tcp“, „net.tcp“ und „net.pipe“) zum Datenaustausch auf unterschiedlichen Ports festgelegt werden können. Die dann dabei verlinkten Strategien (engl. Policies) legen das Vorgehen beim Datenaustausch fest. Bestehend aus dem Policy-Level, dem Sicherheitsmodus (engl. Security-Mode) und dem Sicherheitsalgorithmus (engl. Security-Algorithm) legt jede Policy die Art des sicheren Datenaustauschs fest.

Werden zum Beispiel zwei Sicherheitsstrategien (engl. Security-Policies) verfolgt, dann könnten diese wie folgt definiert sein:

- Security-Policy A: Level=0, Security-Mode=None, Security-Algorithm=None
- Security-Policy B: Level=1, Security-Mode=Sign, Security-Algorithm=Basic256

Werden weiter zum Beispiel drei Basis-Adressen (engl. Base-Addresses) wie folgt für verschiedene Schemen festgelegt:

- Base-Address A: "https://mydomain.com/"
- Base-Address B: "opc.tcp://192.168.0.123:4840/"
- Base-Address C: "opc.tcp://192.168.0.123:12345/"

So ergeben sich daraus durch das Kreuzprodukt die folgenden Endpunktbeschreibungen:

- Endpoint 1: Address="https://mydomain.com/", Level=0, Security-Mode=None, Security-Algorithm=None
- Endpoint 2: Address="https://mydomain.com/", Level=1, Security-Mode=Sign, Security-Algorithm=Basic256
- Endpoint 3: Address="opc.tcp://192.168.0.123:4840/", Level=0, Security-Mode=None, Security-Algorithm=None
- Endpoint 4: Address="opc.tcp://192.168.0.123:4840/", Level=1, Security-Mode=Sign, Security-Algorithm=Basic256
- Endpoint 5: Address="opc.tcp://192.168.0.123:12345/", Level=0, Security-Mode=None, Security-Algorithm=None
- Endpoint 6: Address="opc.tcp://192.168.0.123:12345/", Level=1, Security-Mode=Sign, Security-

Algorithm=Basic256

Dabei wird der Addressteil des Endpunktes immer vom Server festgelegt (via Konstruktor oder via **Address** Eigenschaft). Während der Server standardmäßig einen Endpunkt mit dem Security-Mode „None“ definiert, muss manuell die Policy der Endpunkte konfiguiert werden (**OpcServer.Security.EndpointPolicies** Eigenschaft), wenn kein solcher oder zusätzliche verwendet werden sollen.

Aufklärung über Zertifikate

Zertifikate in OPC UA

Zertifikate dienen dazu, die **Authentizität** (einfach: Echtheit) und **Integrität** (einfach: Vertraulichkeit) von Client- und Serveranwendungen **sicherzustellen**. Sie dienen somit einer Client- sowie einer Serveranwendung als eine Art Personalausweis. Da dieser „Personalausweis“ in Form einer Datei vorliegt, muss diese irgendwo gespeichert werden. Wo die Zertifikate gespeichert werden, kann individuell entschieden werden. Unter Windows kann jedes Zertifikat direkt an das **System** übergeben werden und Windows kümmert sich um den **Speicherort**. Alternativ können auch **benutzerdefinierte Speicherorte** (= Verzeichnisse) festgelegt werden.

Es gibt verschiedene Typen von Speicherorten für Zertifikate:

- **Speicherort für Anwendungszertifikate**
  Der auch als **Application Certificate Store** bezeichnete Speicherort enthält ausschließlich die Zertifikate der Anwendungen, die diesen Store als Application Certificate Store verwenden. Hier speichert eine Client- beziehungsweise Serveranwendung Ihr eigenes Zertifikat.
- **Speicherort für Zertifikate vertrauenswürdiger Zertifikatausteller**
  Der auch als **Trusted Issuer Certificate Store** bezeichnete Speicherort enthält ausschließlich Zertifikate von Zertifizierungsstellen, welche weitere Zertifikate ausstellen. Hier speichert eine Client- beziehungsweise Serveranwendung alle Zertifikate der Aussteller (engl. issuer), deren Zertifikate standardmäßig als vertrauenswürdig (engl. trusted) eingestuft werden sollen.
- **Speicherort für vertrauenswürdige Zertifikate**
  Der auch als **Trusted Peer Store** bezeichnete Speicherort enthält ausschließlich Zertifikate, die als vertrauenswürdig eingestuft werden. Hier speichert ein Client die **Zertifikate vertrauenswürdiger Server** beziehungsweise ein Server die **Zertifikate vertrauenswürdiger Clients**.
- **Speicherort für verweigerte Zertifikate**
  Der auch als **Rejected Certificate Store** bezeichnete Speicherort enthält ausschließlich Zertifikate, die als nicht vertrauenswürdig eingestuft werden. Hier speichert ein Client die **Zertifikate nicht vertrauenswürdiger Server** beziehungsweise ein Server die **Zertifikate nicht vertrauenswürdiger Clients**.

Unabhängig, ob der Speicherort irgendwo im System liegt oder im Dateisystem über ein Verzeichnis, es gilt generell, dass Zertifikaten, die im **Trusted Store** liegen, **vertraut** wird und Zertifikaten, die im **Rejected Store** liegen, **nicht vertraut** wird. Zertifikate die in keinem von beiden enthalten sind, werden automatisch in den Trusted Store gespeichert, wenn das Zertifikat des im Zertifikat hinterlegten Zertifikatsaustellers im Trusted Issuer Store existiert; andernfalls wird es automatisch in den Rejected Store gespeichert. Ist selbst ein vertrauenswürdiges Zertifikat abgelaufen oder können dessen hinterlegten Informationen nicht erfolgreich durch die Zertifizierungsstelle geprüft werden, dann wird das Zertifikat als nicht vertrauenswürdig eingestuft und in den Rejected Store gespeichert. Dabei wird es auch aus dem

Trusted Peer Store wieder entfernt. Ebenso kann ein Zertifikat ungültig werden, wenn es in einer Zertifikatsperrliste (engl. Certificate Revocation List - CRL) gelistet ist, welche im jeweiligen Store separat geführt werden kann.

Ein Zertifikat, das der Client vom Server beziehungsweise das der Server vom Client erhält, wird **vorerst immer** als *unbekannt* eingestuft und somit auch als **nicht vertrauenswürdig** (engl. untrusted) behandelt. Damit ein Zertifikat als vertrauenswürdig behandelt wird, muss es als solches deklariert werden. Dies geschieht, indem das Zertifikat des Clients im Trusted Store des Servers beziehungsweise das Zertifikat des Servers im Trusted Store des Clients gespeichert wird.

Abhandlung eines Serverzertifikats beim Client:

1. Der Client ermittelt das Zertifikat des Servers, auf dessen Endpunkt er sich verbinden soll.
2. Der Client prüft das Zertifikat des Servers.
    1. Ist das Zertifikat gültig?
        1. Ist das Gültigkeitsdatum überschritten?
        2. Ist das Zertifikat des Ausstellers gültig?
    2. Existiert das Zertifikat im Trusted Peer Store?
        1. Ist es in eine Zertifikatsperrliste (CRL) eingetragen?
    3. Existiert das Zertifikat im Rejected Store?
3. Ist das Zertifikat vertrauenswürdig, dann stellt der Client eine Verbindung zum Server her.

Abhandlung eines Clientzertifikats beim Server:

1. Der Server erhält beim Verbindungsaufbau durch den Client das Zertifikat des Clients.
2. Der Server prüft das Zertifikat des Clients.
    1. Ist das Zertifikat gültig?
        1. Ist das Gültigkeitsdatum überschritten?
        2. Ist das Zertifikat des Ausstellers gültig?
    2. Existiert das Zertifikat im Trusted Peer Store?
        1. Ist es in eine Zertifikatsperrliste (CRL) eingetragen?
    3. Existiert das Zertifikat im Rejected Store?
3. Ist das Zertifikat vertrauenswürdig, dann lässt der Server die Verbindung des Clients zu und bedient ihn.

Im Falle, dass die Prüfung des Zertifikats der jeweiligen Gegenstelle fehlschlägt, kann über benutzerdefinierte Mechanismen die Prüfung erweitert werden und selbst auf Benutzerebene noch entschieden werden, ob das Zertifikat akzeptiert wird oder nicht.

Arten von Zertifikaten

Allgemein: Selbstsignierte Zertifikate vs. signierte Zertifikate

Ein Zertifikat ist vergleichbar mit einer Urkunde. Eine Urkunde kann von jedem ausgestellt und auch von jedem unterzeichnet werden. Hierbei besteht aber ein wesentlicher Unterschied darin, ob der Unterzeichner einer Urkunde auch wirklich für dessen Korrektheit bürgt (wie ein Notar), oder ob der Unterzeichner der Inhaber der Urkunde selbst ist. Insbesondere Urkunden der letzteren Art sind nicht besonders vertrauenerweckend, da keine anerkannte (gesetzliche) Instanz wie zum Beispiel ein Notar sich für den Inhaber der Urkunde verbürgt.

Da Zertifikate mit Urkunden vergleichbar sind und ebenfalls eine (digitale) Unterschrift (= Signierung) aufweisen müssen, verhält es sich hier genauso. Die Signatur eines Zertifikats muss dem Empfänger einer

Zertifikatskopie Auskunft darüber geben, wer sich für dieses Zertifikat verbürgt. Dabei gilt immer, dass der Aussteller (engl. issuer) eines Zertifikats zugleich dieses auch signiert. Wenn der **Aussteller eines Zertifikats gleich der Zielperson** (engl. subject) des Zertifikats ist, dann spricht man von einem **selbstsignierten Zertifikat** (Subject ist gleich Issuer). Wenn der **Aussteller eines Zertifikats nicht gleich der Zielperson** des Zertifikats ist, dann spricht man von einem **(einfachen / normalen / signierten) Zertifikat** (Subject ist nicht gleich Issuer).

Da Zertifikate insbesondere im Kontext der OPC UA zur Authentifizierung einer Identität (einer bestimmten Client- oder Serveranwendung) eingesetzt werden, sollten signierte Zertifikate als Anwendungszertifikate für die eigene Anwendung verwendet werden. Ist hingegen der Aussteller zugleich auch Inhaber des Zertifikats, sollte dessen selbstsigniertem Zertifikat nur dann vertraut werden, wenn man den Inhaber als vertrauenswürdig einstuft. Solche Zertifikate wurden, wie eben beschrieben, durch den Aussteller des Zertifikats signiert. Das hat wiederum zur Folge, dass das Zertifikat des Ausstellers (engl. issuer certificate) im **Trusted Issuer Store** der Anwendung liegen muss. Ist das Zertifikat des Ausstellers dort nicht auffindbar, gilt die Zertifikatskette (engl. certificate chain) als unvollständig, woraus folgt, dass das Zertifikat der Gegenstelle nicht akzeptiert wird. Ist hingegen das Zertifikat vom Aussteller des Anwendungszertifikats wiederum kein selbstsigniertes Zertifikat, dann muss das Zertifikat von dessen Aussteller im **Trusted Issuer Store** verfügbar sein.

Benutzeridentifizierung

Benutzeridentifizierung mit Zertifikate

Neben dem Einsatz eines Zertifikats als *Personalausweis* für Client- beziehungsweise Serveranwendungen, kann ein Zertifikat auch zur Identifizierung eines Benutzers verwendet werden. Eine Clientanwendung wird stets durch einen bestimmten Benutzer bedient, durch die er mit dem Server operiert. Je nach Serverkonfiguration kann ein Server vom Client zusätzlich Informationen über die Identität des Benutzers des Clients anfordern. Hier besteht die Möglichkeit, dass der Benutzer seine Identität in Form eines Zertifikats ausweist. In wieweit ein Server das Zertifikat auf seine Gültigkeit, Echtheit und Vertraulichkeit prüft hängt vom jeweiligen Server ab. Der vom Framework bereitgestellte Server prüft dabei ausschließlich, ob die Thumbprintinformationen der Benutzeridentität in seiner Zugriffskontrollliste (engl. Access Control List - ACL) für Zertifikat-basierte Benutzeridentitäten auffindbar ist.

Aspekte der Sicherheit

Produktiver Einsatz

Das primäre Ziel des Frameworks ist es, den Einstieg in OPC UA so einfach wie möglich zu gestalten. Dieser Grundgedanke führt leider auch dazu, dass ohne weiterführende Konfiguration des Servers keine völlig sichere Verbindung / Kommunikation zwischen Client und Server stattfindet. Wurde jedoch der finale Spike implementiert und getestet, sollte über die *Aspekte der Sicherheit* nachgedacht werden.

Auch wenn der Server *nur* innerhalb eines lokalen Netzwerks betrieben wird, sollte über den Einsatz von Zugriffskontrolllisten nachgedacht werden (**OpcServer.Security.UserNameAcl/CertificateAcl** Eigenschaft). Hierbei können Benutzeridentitäten (engl. User Identities) über ein Zertifikat (engl. Certificate) oder ein Benutzername-Passwort-Paar definiert werden. Eine Certificate Identity erhöht zum Beispiel die Sicherheit bei der signierten Datenübertragung.

Insbesondere in Fällen, in denen der Server öffentlich erreichbar ist, sollte über andere

Sicherheitsstrategien (engl. Security-Policies) mit entsprechend höherem Sicherheitsmodus (engl. Security-Mode) und passendem Sicherheitsalgorithmus (engl. Security-Algorithm) verhandelt werden. Der standardmäßig verwendete Security-Policy-Mode „None“ bietet diesbezüglich ein sprichwörtlich „offenes Scheuentor“ zu Ihrem Server (**OpcServer.Security.EndpointPolicies** Eigenschaft). Nicht zuletzt sollte auch über den Zugriff über eine anonyme Benutzeridentität nachgedacht werden (**OpcServer.Security.AnonymousAcl.IsEnabled** Eigenschaft). Gemäß der OPC Foundation sollte jede verwendete Endpoint Policy über ihren Level ihre „Güte“ hervorheben, wobei die Regel gilt, dass je höher der Level ist, desto „besser“ ist der Endpunkt, der diese Policy verwendet.

Zum vereinfachten Handling von Zertifikaten akzeptiert der Server standardmäßig jedes Zertifikat (**OpcServer.Security.AutoAcceptUntrustedCertificates** Eigenschaft), auch die, die er unter produktiven Bedingungen verweigern sollte. Denn nur Zertifikate, die dem Server bekannt sind (diese befinden sich im TrustedPeer Zertifikatsspeicher), gelten als wirklich vertrauenswürdig. Davon abgesehen sollte stets die Gültigkeit der Zertifikate geprüft werden, dazu zählt unter anderem das „Verfallsdatum“ des Zertifikats. Andere Eigenschaften des Zertifikats oder lockerere Regeln für die Gültigkeit und Vertrauenswürdigkeit eines Clientzertifikats können zudem manuell durchgeführt werden (**OpcServer.CertificateValidationFailed** Ereignis).

Inhaltsverzeichnis