|  |  |
| --- | --- |
|  | OPC UA Specification |
| OPC 10000-12 |
| OPC Unified Architecture  Part 12: Discovery and Global Services  Draft 1.05  2022-03-09 |

|  |  |  |  |
| --- | --- | --- | --- |
| Specification Type | Industry Standard Specification | Comments: |  |
|  |  |  |  |
| Document Number | OPC 10000-12 |  |  |
| Title: | OPC Unified Architecture | Date: |  |
|  |  |  |  |
| Version: |  | Software | MS-Word |
|  |  | Source: |  |
|  |  |  |  |
| Author: |  | Status: |  |
|  |  |  |  |

# Additions to GDS model

## Clarifications

Application authentication at GDS

* How does Authorization work if the application started signing request with Anonymous the first time and GDS admin confirmed the request on GDS side?
* If the GDS allows anonymous signing and acceptance by GDS admin, the GDS shall store the application cert that started the Request. Needs to be extended in FinishRequest.

Method GetCertificateStatus

If we use GetCertificateStatus in the standard workflow, the method would return Bad\_UserAccessDenied if there is no signed certificate assigned yet and user has no admin rights. Clarify that if GDS allows StartSigningRequest with Anonymous, it shall also allow GetCertificateStatus with Anonymous.

We should also clarify that GetCertificateStatus returns also TRUE if no signed certificate was assigned before.

## Missing features

Push configuration on GDS

If an application is registered with the GDS, there is no standard way to activate the push model for the application on the GDS.

Pull configuration on Server

If a standard tool registeres a server, there is no way to configure PULL on the server. There is also no capability information if server supports/prefers PULL or PUSH.

This is something that should be added if endpoint configuration is added. The PULL configuraiton makes only sense if also the endpoint(s) can be configured from a standard tool.

Expose a mode / capability for anonymous registration

We should add GDS capability including the the capability to support anonymous registration and signing request.

Add Capablity information to the CertificateGroup

It would be good to have information about the expected timing for processing of signing requests (1) internal CA – processing in milliseconds (2) external CA – processing takes longer but is still done immediately (3) signing requires manual admin confirmation

Application registration for none-OPC applications

We need new Register/Update/Unregister Methods for non-OPC UA Applications. Using a KeyValuePair as record allows us to define the list of keys for different application types.

# Global Services

## Introduction

TBD – general introduction and reference to global services in the spec

## Permission Sets

TBD - central place to defined the permission sets for the global services

Introduce also a anonymous permission that can be referenced.

AnonymousPermission

## Client connections to global services

A *GlobalDiscoveryServer* is an OPC UA Server implementing different global services for discovery, *Certificate* management, user or PubSub key management, user authorization, software and device management.

The number of OPC UA *Applications* using the different services as OPC UA *Client* may be huge and the OPC UA *Server* ist most likely not able to handle connections from all OPC UA *Clients* at the same time.

Therefore an OPC UA *Client* connected to a GDS should minimize the time it is connected to the GDS to the currently required actions. The OPC UA Client shall disconnect as soon as they completed the sequence of actions needed to interact with the services. The OPC UA *Clients* shall not keep connections open between the execution of sequences.

A GDS OPC UA *Server* is allowed to close *Sessions* with OPC UA Clients not authenticated as one of the GDS administrative roles if it runs out of connection resources. If the GDS want to close *Sessions*, it should first close *Sessions* with AnonymousPermission. Otherwise it could close the *Session* that was inactive for the longest time not using GDS *Method* calls.

It is also recommended to use a short maximum session timeout on the GDS OPC UA *Server*.

Actions done cyclicly by OPC UA *Applications* like cyclically Pull certificate management shall start at random offset in the cycle period.

## DirectoryType

### SetAuthorizedApplication

*SetAuthorizedApplication* is used to assign an *Application Instance Certificate* to an *Application* managed by the *GlobalDiscoveryServer* and assignes the GDS *CertificateGroups* used for the *Application*.

The *Certificate* is used by the registered *Application* for *Application* authentication to the GDS until the *Application* has a GDS signed *Certificate*.

This *Method* shall only be invoked by authorized users.

**Signature**

**SetAuthorizedApplication**(

[in] NodeId applicationId

[in] NodeId[] certificateGroupIds

[in] ByteString certificate

);

|  |  |
| --- | --- |
| **Argument** | **Description** |
| applicationId | The identifier assigned by the GDS to the *Application*. |
|  |  |
|  |  |

**Method Result Codes (defined in Call Service)**

|  |  |
| --- | --- |
| **Result Code** | **Description** |
| Bad\_NotFound | The *ApplicationId* is not known to the GDS. |
| Bad\_UserAccessDenied | The current user does not have the rights required. |

Table 1 specifies the *AddressSpace* representation for the *SetAuthorizedApplication* *Method*.

Table 1 – SetAuthorizedApplication Method AddressSpace Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | SetAuthorizedApplication | | | | |
| **References** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **ModellingRule** |
| HasProperty | Variable | InputArguments | Argument[] | PropertyType | Mandatory |

A new Method SetAuthorizedApplication allows to link an OPC UA application certificate not known to GDS but used by the registered application for application authentication to the GDS until the application has a GDS signed certificate. This is maybe also a necessary option for management of self-signed certificate. The Method links also the Application to the CertificateGroup(s). It requires GDS administration rights.

# Application Registration

The OPC UA *Application* or the *Application* configuration tool connects to the GDS for initial installation with GDS including application registration. This requires a user that is allowed to execute the initial installation.



Figure 1 – Application Registration Workflow

Table 2 – Application Registration Steps

|  |  |
| --- | --- |
| **Step** | **Description** |
| Application installation | The registration of an OPC UA *Application* with a GDS is normally executed as part of the initial installation of configuration of the OPC UA *Application*.  It can be executed by a configuration tool that is part of the OPC UA *Application* or by a generic GDS configuration tool. |
| Connect | For the connection management with the GDS the services *OpenSecureChannel*, *CreateSession* and *ActivateSession* are used to create a connection with *MessageSecurityMode* *SignAndEncrypt* and a user that has the permission to register OPC UA *Applications* with the GDS. If the user has no rights, the GDS can provide a mechanism to accept registrations on the GDS side. See **TBD** for more details. |
| FindApplications | The first step after connect is the check if there is already a registration available.  The *DirectoryType* *Method* *FindApplications* is used to pass the *ApplicationUri* of the OPC UA *Application* to the GDS. The Method returns an array of application records where the size of the array defines the next steps.   * If the array is empty, the next step is *RegisterAppilcation*. * If the array size is one, and a registration is expected, the next step is *Browse* CertificateGroups. * If the array size is one and a registration is not expected, the registration must be verified with a GDS administrator. * If the array size is more than one, this indicates a configuration error and the status must be verified with a GDS administrator. |
| RegisterApplication | The *DirectoryType* *Method* *RegisterApplication* is used to pass in an application record with the application information.  If the *Method* succeeds an *ApplicationId* is returned. This *ApplicationId* must be persisted for further interaction with the GDS regarding this OPC UA *Application*.  If the Method fails, the right to add an OPC UA *Application* or other failure reasons must be checked with a GDS administrator. |
| Browse CertificateGroups | The *Browse* *Service* is used to get the list of GDS managed *CertificateGroups* by browsing the *CertificateGroups* *Folder* of the *Directory* *Object*.  If the Browse returns only the well known *DefaultApplicationGroup*, a null *NodeId* can be used in all *Method* calls related to *CertificateGroups*.  If more *CertificateGroups* are returned, the user must select the relevant *CertificateGroups* needed for the OPC UA *Application*.  The selected *CertificateGroupIds* must be persisted together with the *ApplicationId*. |
| Registration end options | The following options are possible to complete the registration   1. Continue with PULL certificate management using the existing connection to the GDS. This options is typically used by OPC UA *Client* *Applications* executing the registration in an interactive mode for their own identity. See certificate management workflow for next steps. 2. Continue with PULL certificate management inside a headless application. 3. Continue with PUSH certificate management. |
| SetAuthorizedApplication | For option (2) the current application certificate can be set to allow application authentication for the initial certificate management sequence. |
| Configure Push | For option (3) the application must be configured for PUSH in the GDS. **Details TBD** |
| Disconnect | For options (2) and (3) the configuration tool disconnects from the GDS. |

# Certificate Management

## Introduction

## Pull Certificate Management

The OPC UA *Application* continues the installation or connects to the GDS for cyclic check of certificate status and trust list update. The cycle time is defined by the *UpdateFrequency* on the related *TrustList* *Object* in the GDS.

In this sequence the OPC UA *Application* that want to get certificates from the GDS is the *Client* that executes the workflow and the GDS is the *Server* processing the request in the sequence.

The application is authenticated with the application certificate signed by the GDS (or the certificate assigned during registration). **UserTokenType is always Anonymous**.

The application needs the ApplicationId assigned during installation as input.

The workflow for Pull certificate management is shown in Figure 2 and described in Table 3. The two options for the key pair creation are described in Figure 3.



Figure 2 – Pull Certificate Management Workflow



Figure 3 – The Pull Certificate Management Private Key Options

The steps of the Pull certificate management workflow are described in detail in Table 3.

Table 3 – Pull Certificate Management Workflow Steps

|  |  |
| --- | --- |
| **Step** | **Description** |
| Certificate management begin options | The following options are possible to start the Pull certificate management   1. Continue application installation as part of configuration tool for the OPC UA *Application* using the *Session* available from application registration described in **TBD** add reference 2. Cyclic update check of the OPC UA Application by creating a new connection. TBD describe cycle time is defined by UpdateFrequence on related TrustList |
| Connect | Create a connection for option (2). For the connection management with the GDS the *Services* *OpenSecureChannel*, *CreateSession* and *ActivateSession* are used to create a connection with *MessageSecurityMode* *SignAndEncrypt* and an *Anonymous* user.  The application authentication is used by the GDS to allow OPC UA *Applications* to access the necessary resources to update themselves. |
| Required information | The OPC UA *Application* need to know the following information to execute the pull certificate management workflow   * ApplicationId  *NodeId* of the OPC UA *Application* in the GDS. * CertificateGroupIds Array of *NodeIds* for each *CertificateGroup* in the GDS that is relevant for the OPC UA *Application*. This includes a mapping to OPC UA *Application* the related internal *CertificateGroup* and the *CertificateTypes* needed. The *NodeId* for the default *CertificateGroup* is NULL. * Pending signing requests Array of *RequestId NodeIds* for pending signing requests that need to be completed. This includes the relation to the *CertificateGroup* and *CertificateType*. |
| SigningRequestPending | If one or more signing requests are pending for a *CertificateGroup*, the *FinishRequest Method* is called directly with *ApplicationId* and the *RequestId* for the pending signing request. The repeat count is set to 0 in this case. |
| GetCertificateStatus | The *Method GetCertificateStatus* is called with the *ApplicationId* and the *CertificateGroupId* to check if a certificate update is needed. This is repeated for each *CertificateType* needed for the *CertificateGroup*. |
| Create CSR | If *GetCertificateStatus* returns updateRequired set to True for one or more combinations of *CertificateGroupId* and *CertificateType*, the process for a *Certificate* update is started for the affected combinations.  As first step the OPC UA *Application* create a certificate signing request (CSR). It is strongly recommended, that the OPC UA *Application* creates a new private key for the signing request. |
| StartSigningRequest | The *Method StartSigningRequest* is called for each *CertificateGroupId* and *CertificateType* together with the CSR to request a signed *Certificate* from the GDS. Each *Method* call needs it’s own CSR.  As alternative for OPC UA Applications who are not able to create a good private key, the *Method StartNewKeyPairRequest* can be used. In this case the private key is created by the GDS.  Both Methods return a *RequestId* that can be passed to the *FinishRequest Method*. The repeat count for *FinishRequest* is set to a small number like 2. |
| FinishRequest | The *Method FinishRequest* is called to check the results of a previous *StartSigningRequest* or *StartNewKeyPairRequest*.  The following results are possible:   * If *FinishRequest* returns a *Good* result, the *Method* returns the signed *Certificate* and optionally the private key for the *StartNewKeyPairRequest* case. * If *FinishRequest* returns *BadNothingToDo* which indicates that the request is not completed yet. If the repeat count is not 0, the repeat count is decremented and *FinishRequest* is repeated after a short delay. If the repeat count is 0, the next *CertificateGroup* or *CertificateType* is processed * If *FinishRequest* returns any other *Bad* result, a new request must be sent in the next cycle |
| GetTrustList | If all *Certificates* for a *CertificateGroup* are up-to-date, the trust list is checked for updates by calling the *Method* *GetTrustList*. The Method returns the *NodeId* of the *TrustList Object* for the *CertificateGroup*.  TBD describe check for update. check LastUpdateTime of TrustList before executing the update. Skip read if update is not needed |
| TrusListType:Read | The NodeId of the *TrustList Object* returned by *GetTrustList* is used to open the TrustList for reading and to read the current content of the *TrustList*. |
| Persist | If a *TrustList* update or *Certificate* updates are available, they are persisted for futher use the OPC UA *Application*. The must be persisted at the same time to have a consitend setup. |
| FinishRequest repeat count reached | Persist RequestId and disconnect.  Repeat count should be small e.g. 1 or 2.  There must be a delay between repeats. |
|  |  |

## PUSH Certificate Management

In this sequence the GDS is the Client that executes the workflow and the OPC UA *Application* is the *Server* that processing the request in the sequence. The workflow is started if an update is required.



Figure 4 – The Push Certificate Management Workflow



Figure 5 – The Push Certificate Management Private Key Options

# GDS with user certificate groups

## Introduction

Enhanced GDS configuration with one or more CertificateGroups for user certificates.

## Pull User Certificate

## Pull Server

## Push Server

\_\_\_