

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**OPC unified architecture –
Part 12: Discovery and global services**

**Architecture unifiée OPC –
Partie 12: Services globaux et de découverte**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2020 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

Electropedia - www.electropedia.org

The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

67 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

A propos de l'IEC

La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC

Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Recherche de publications IEC -

webstore.iec.ch/advsearchform

La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,...). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et une fois par mois par email.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: sales@iec.ch.

Electropedia - www.electropedia.org

Le premier dictionnaire d'électrotechnologie en ligne au monde, avec plus de 22 000 articles terminologiques en anglais et en français, ainsi que les termes équivalents dans 16 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary

67 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

INTERNATIONAL STANDARD

NORME INTERNATIONALE

**OPC unified architecture –
Part 12: Discovery and global services**

**Architecture unifiée OPC –
Partie 12: Services globaux et de découverte**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 25.040.40

ISBN 978-2-8322-8455-1

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

CONTENTS

FOREWORD.....	8
1 Scope.....	10
2 Normative references	10
3 Terms, definitions, abbreviated terms and conventions.....	11
3.1 Terms and definitions.....	11
3.2 Abbreviated terms and symbols	13
3.3 Conventions for namespaces	13
4 The discovery process.....	14
4.1 Overview.....	14
4.2 Registration and announcement of Applications	15
4.2.1 Overview	15
4.2.2 Hosts with a LocalDiscoveryServer	15
4.2.3 Hosts without a LocalDiscoveryServer	16
4.3 The discovery process for Clients to find Servers.....	16
4.3.1 Overview	16
4.3.2 Security	17
4.3.3 Simple Discovery with a DiscoveryUrl	17
4.3.4 Local Discovery	17
4.3.5 MulticastSubnet Discovery.....	18
4.3.6 Global Discovery	19
4.3.7 Combined Discovery Process for Clients	19
5 Local Discovery Server.....	20
5.1 Overview.....	20
5.2 Security considerations for Multicast DNS.....	21
6 Global Discovery Server	21
6.1 Overview.....	21
6.2 Network architectures	22
6.2.1 Overview	22
6.2.2 Single MulticastSubnet	22
6.2.3 Multiple MulticastSubnet.....	23
6.2.4 No MulticastSubnet.....	23
6.2.5 Domain Names and MulticastSubnets.....	24
6.3 Information Model	25
6.3.1 Overview	25
6.3.2 Directory.....	25
6.3.3 DirectoryType	25
6.3.4 FindApplications	26
6.3.5 ApplicationRecordDataType.....	27
6.3.6 RegisterApplication.....	28
6.3.7 UpdateApplication	29
6.3.8 UnregisterApplication	30
6.3.9 GetApplication	30
6.3.10 QueryApplications	31
6.3.11 QueryServers (deprecated).....	33
6.3.12 ApplicationRegistrationChangedAuditEventType.....	34
7 Certificate management overview	35

7.1	Overview.....	35
7.2	Pull Management.....	36
7.3	Push management.....	36
7.4	Provisioning.....	37
7.5	Common Information Model.....	38
7.5.1	Overview.....	38
7.5.2	TrustListType.....	38
7.5.3	OpenWithMasks.....	39
7.5.4	CloseAndUpdate.....	40
7.5.5	AddCertificate.....	41
7.5.6	RemoveCertificate.....	42
7.5.7	TrustListDataType.....	42
7.5.8	TrustListMasks.....	43
7.5.9	TrustListOutOfDateAlarmType.....	43
7.5.10	CertificateGroupType.....	43
7.5.11	CertificateType.....	44
7.5.12	ApplicationCertificateType.....	45
7.5.13	HttpsCertificateType.....	45
7.5.14	UserCredentialCertificateType.....	45
7.5.15	RsaMinApplicationCertificateType.....	46
7.5.16	RsaSha256ApplicationCertificateType.....	46
7.5.17	CertificateGroupFolderType.....	46
7.5.18	TrustListUpdatedAuditEventType.....	47
7.6	Information Model for Pull Certificate Management.....	48
7.6.1	Overview.....	48
7.6.2	CertificateDirectoryType.....	48
7.6.3	StartSigningRequest.....	49
7.6.4	StartNewKeyPairRequest.....	51
7.6.5	FinishRequest.....	53
7.6.6	GetCertificateGroups.....	54
7.6.7	GetTrustList.....	55
7.6.8	GetCertificateStatus.....	56
7.6.9	CertificateRequestedAuditEventType.....	57
7.6.10	CertificateDeliveredAuditEventType.....	58
7.7	Information Model for Push Certificate Management.....	58
7.7.1	Overview.....	58
7.7.2	ServerConfiguration.....	59
7.7.3	ServerConfigurationType.....	59
7.7.4	UpdateCertificate.....	61
7.7.5	ApplyChanges.....	62
7.7.6	CreateSigningRequest.....	63
7.7.7	GetRejectedList.....	64
7.7.8	CertificateUpdatedAuditEventType.....	64
8	KeyCredential management.....	65
8.1	Overview.....	65
8.2	Pull management.....	66
8.3	Push management.....	66
8.4	Information Model for pull management.....	67
8.4.1	Overview.....	67

8.4.2	KeyCredentialManagement	68
8.4.3	KeyCredentialServiceType	68
8.4.4	StartRequest	69
8.4.5	FinishRequest	70
8.4.6	Revoke	71
8.4.7	KeyCredentialAuditEventType	72
8.4.8	KeyCredentialRequestedAuditEventType	73
8.4.9	KeyCredentialDeliveredAuditEventType	73
8.4.10	KeyCredentialRevokedAuditEventType	73
8.5	Information Model for push management	74
8.5.1	General	74
8.5.2	KeyCredentialConfiguration	74
8.5.3	KeyCredentialConfigurationType	75
8.5.4	UpdateCredential	75
8.5.5	DeleteCredential	76
8.5.6	KeyCredentialUpdatedAuditEventType	77
8.5.7	KeyCredentialDeletedAuditEventType	77
9	Authorization Services	78
9.1	Overview	78
9.2	Implicit	78
9.3	Explicit	79
9.4	Chained	80
9.5	Information Model for Requesting Access Tokens	81
9.5.1	Overview	81
9.5.2	AuthorizationServices	82
9.5.3	AuthorizationServiceType	82
9.5.4	RequestAccessToken	83
9.5.5	GetServiceDescription	84
9.5.6	AccessTokenIssuedAuditEventType	85
9.6	Information Model for configuring Servers	85
9.6.1	Overview	85
9.6.2	AuthorizationServices	86
9.6.3	AuthorizationServiceConfigurationType	86
Annex A (informative)	Deployment and configuration	87
A.1	Firewalls and discovery	87
A.2	Resolving references to remote Servers	89
Annex B (normative)	Constants	91
Annex C (normative)	OPC UA Mapping to mDNS	92
C.1	DNS Server (SRV) record syntax	92
C.2	DNS Text (TXT) record syntax	92
C.3	DiscoveryUrl mapping	93
Annex D (normative)	Server Capability Identifiers	94
Annex E (normative)	DirectoryServices	95
E.1	Global Discovery via other directory services	95
E.2	UDDI	95
E.3	LDAP	96
Annex F (normative)	Local Discovery Server	98
F.1	Certificate store directory layout	

F.2	Installation directories on Windows	99
Annex G (normative)	Application installation process	100
G.1	Provisioning with Pull Management	100
G.2	Provisioning with Push Management	100
G.3	Setting permissions	101
Annex H (informative)	Comparison with RFC 7030	102
H.1	Overview	102
H.2	Obtaining CA Certificates	102
H.3	Initial enrolment	102
H.4	Client Certificate reissuance	103
H.5	Server key generation	103
H.6	Certificate Signing Request (CSR) attributes request	103
Figure 1	– The Registration process with an LDS	16
Figure 2	– The simple Discovery process	17
Figure 3	– The Local Discovery process	18
Figure 4	– The MulticastSubnet Discovery process	18
Figure 5	– The Global Discovery process	19
Figure 6	– The Discovery Process for Clients	20
Figure 7	– The relationship between GDS and other components	21
Figure 8	– The Single MulticastSubnet architecture	22
Figure 9	– The Multiple MulticastSubnet architecture	23
Figure 10	– The No MulticastSubnet architecture	24
Figure 11	– The Address Space for the GDS	25
Figure 12	– The Pull Certificate management model	36
Figure 13	– The Push Certificate management model	37
Figure 14	– The Certificate Management AddressSpace for the GlobalDiscoveryServer	48
Figure 15	– The AddressSpace for the Server that supports Push Management	59
Figure 16	– The Pull Model for KeyCredential management	66
Figure 17	– The Push Model for KeyCredential management	67
Figure 18	– The Address Space used for Pull KeyCredential management	68
Figure 19	– The AddressSpace used for Push KeyCredential management	74
Figure 20	– Roles and Authorization Services	78
Figure 21	– Implicit authorization	79
Figure 22	– Explicit authorization	80
Figure 23	– Chained authorization	81
Figure 24	– The Model for Requesting Access Tokens from Authorization Services	82
Figure 25	– The Model for configuring Servers to use Authorization Services	85
Figure A.1	– Discovering Servers outside a firewall	87
Figure A.2	– Discovering Servers behind a firewall	88
Figure A.3	– Using a Discovery Server with a firewall	89
Figure A.4	– Following References to Remote Servers	90
Figure E.1	– The UDDI or LDAP Discovery process	95
Figure E.2	– UDDI registry structure	96

Figure E.3 – Sample LDAP hierarchy	97
Table 1 – GDS NamespaceMetadataType Object definition	14
Table 2 – Directory Object definition	25
Table 3 – DirectoryType definition.....	26
Table 4 – FindApplications Method AddressSpace definition.....	27
Table 5 – ApplicationRecordDataType definition	28
Table 6 – RegisterApplication Method AddressSpace definition	29
Table 7 – UpdateApplication Method AddressSpace definition	30
Table 8 – UnregisterApplication Method AddressSpace definition	30
Table 9 – GetApplication Method AddressSpace definition.....	31
Table 10 – QueryApplications Method AddressSpace definition	33
Table 11 – QueryServers Method AddressSpace definition	34
Table 12 – ApplicationRegistrationChangedAuditEventType definition	35
Table 13 – TrustListType definition	39
Table 14 – OpenWithMasks Method AddressSpace definition	40
Table 15 – CloseAndUpdate Method AddressSpace definition	41
Table 16 – AddCertificate Method AddressSpace definition	41
Table 17 – RemoveCertificate Method AddressSpace definition.....	42
Table 18 – TrustListDataType definition	42
Table 19 – TrustListMasks values	43
Table 20 – TrustListOutOfDateAlarmType definition.....	43
Table 21 – CertificateGroupType definition	44
Table 22 – CertificateType definition	45
Table 23 – ApplicationCertificateType definition.....	45
Table 24 – HttpsCertificateType definition.....	45
Table 25 – UserCredentialCertificateType definition.....	46
Table 26 – RsaMinApplicationCertificateType definition	46
Table 27 – RsaSha256ApplicationCertificateType definition.....	46
Table 28 – CertificateGroupFolderType definition	47
Table 29 – TrustListUpdatedAuditEventType definition	47
Table 30 – CertificateDirectoryType ObjectType definition	49
Table 31 – StartSigningRequest Method AddressSpace definition.....	51
Table 32 – StartNewKeyPairRequest Method AddressSpace definition	53
Table 33 – FinishRequest Method AddressSpace definition	54
Table 34 – GetCertificateGroups Method AddressSpace definition.....	55
Table 35 – GetTrustList Method AddressSpace definition	56
Table 36 – GetCertificateStatus Method AddressSpace definition	57
Table 37 – CertificateRequestedAuditEventType definition	58
Table 38 – CertificateDeliveredAuditEventType definition	58
Table 39 – ServerConfiguration Object definition	59
Table 40 – ServerConfigurationType definition.....	60
Table 41 – UpdateCertificate Method AddressSpace Definition	62

Table 42 – ApplyChanges Method AddressSpace Definition	63
Table 43 – CreateSigningRequest Method AddressSpace definition.....	64
Table 44 – GetRejectedList Method AddressSpace definition.....	64
Table 45 – CertificateUpdatedAuditEventType definition	65
Table 46 – KeyCredentialManagement Object definition	68
Table 47 – KeyCredentialServiceType definition	69
Table 48 – StartRequest Method AddressSpace definition	70
Table 49 – FinishRequest Method AddressSpace definition	71
Table 50 – Revoke Method AddressSpace definition.....	72
Table 51 – KeyCredentialAuditEventType definition	72
Table 52 – KeyCredentialRequestedAuditEventType definition	73
Table 53 – KeyCredentialDeliveredAuditEventType definition	73
Table 54 – KeyCredentialRevokedAuditEventType definition	74
Table 55 – KeyCredentialConfiguration Object definition.....	74
Table 56 – KeyCredentialConfigurationType definition	75
Table 57 – UpdateCredential Method AddressSpace definition	76
Table 58 – DeleteCredential Method AddressSpace definition	77
Table 59 – KeyCredentialUpdatedAuditEventType definition	77
Table 60 – KeyCredentialUpdatedAuditEventType definition	77
Table 61 – AuthorizationServices Object definition.....	82
Table 62 – AuthorizationServiceType definition.....	82
Table 63 – RequestAccessToken Method AddressSpace definition	84
Table 64 – GetServiceDescription Method AddressSpace definition.....	85
Table 65 – AccessTokenIssuedAuditEventType definition	85
Table 66 – AuthorizationServices Object definition.....	86
Table 67 – AuthorizationServiceConfigurationType definition	86
Table C.1 – Allowed mDNS service names	92
Table C.2 – DNS TXT record string format.....	93
Table C.3 – DiscoveryUrl to DNS SRV and TXT Record Mapping	93
Table D.1 – Examples of <i>ServerCapabilityIdentifiers</i>	94
Table E.1 – UDDI tModels.....	96
Table E.2 – LDAP object class schema.....	97
Table F.1 – Application Certificate store directory layout.....	98
Table H.1 – Verifying that a Server is allowed to provide Certificates	102
Table H.2 – Verifying that a Client is allowed to request Certificates	102

INTERNATIONAL ELECTROTECHNICAL COMMISSION

OPC UNIFIED ARCHITECTURE –

Part 12: Discovery and global services

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International standard IEC 62541-12 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

The text of this standard is based on the following documents:

FDIS	Report on voting
65E/711/FDIS	65E/723/RVD

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

Throughout this document and the other parts of the IEC 62541 series, certain document conventions are used:

Italics are used to denote a defined term or definition that appears in the "Terms and definition" clause in one of the parts of the IEC 62541 series.

Italics are also used to denote the name of a service input or output parameter or the name of a structure or element of a structure that are usually defined in tables.

The *italicized terms and names* are, with a few exceptions, written in camel-case (the practice of writing compound words or phrases in which the elements are joined without spaces, with each element's initial letter capitalized within the compound). For example, the defined term is AddressSpace instead of Address Space. This makes it easier to understand that there is a single definition for AddressSpace, not separate definitions for Address and Space.

A list of all parts of the IEC 62541 series, published under the general title *OPC Unified Architecture*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "<http://webstore.iec.ch>" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

OPC UNIFIED ARCHITECTURE –

Part 12: Discovery and global services

1 Scope

This part of IEC 62541 specifies how OPC Unified Architecture (OPC UA) *Clients* and *Servers* interact with *DiscoveryServers* when used in different scenarios. It specifies the requirements for the *LocalDiscoveryServer*, *LocalDiscoveryServer-ME* and *GlobalDiscoveryServer*. It also defines information models for *Certificate* management, *KeyCredential* management and *Authorization Services*.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC TR 62541-1, *OPC Unified Architecture – Part 1: Overview and concepts*

IEC TR 62541-2, *OPC Unified Architecture – Part 2: Security Model*

IEC 62541-3, *OPC Unified Architecture – Part 3: Address Space Model*

IEC 62541-4, *OPC Unified Architecture – Part 4: Services*

IEC 62541-5, *OPC Unified Architecture – Part 5: Information Model*

IEC 62541-6, *OPC Unified Architecture – Part 6: Mappings*

IEC 62541-7, *OPC Unified Architecture – Part 7: Profiles*

IEC 62541-9, *OPC Unified Architecture – Part 9: Alarms and conditions*

IEC 62541-14, *OPC Unified Architecture – Part 14: PubSub*

X.500: ISO/IEC 9594-1:2017, *Information technology – Open Systems Interconnection – The Directory – Part 1: Overview of concepts, models and services*

IETF RFC 1035, *DNS-Name: Domain Names – Implementation and Specification*
<http://www.ietf.org/rfc/rfc1035.txt>

IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification*
<http://www.ietf.org/rfc/rfc2986.txt>

IETF RFC 3927, *Auto-IP: Dynamic Configuration of IPv4 Link-Local Addresses*
<http://www.ietf.org/rfc/rfc3927.txt>

IETF RFC 5958, *Asymmetric Key Packages*
<http://www.ietf.org/rfc/rfc5958.txt>

IETF RFC 6762, *mDNS: Multicast DNS*

<http://www.ietf.org/rfc/rfc6762.txt>

IETF RFC 6763, *DNS-SD: DNS Based Service Discovery*

<http://www.ietf.org/rfc/rfc6763.txt>

IETF RFC 7030: *Enrollment over Secure Transport*

<http://www.ietf.org/rfc/rfc7030.txt>

PKCS #12: *Personal Information Exchange Syntax*

<http://www.emc.com/emc-plus/rsa-labs/pkcs/files/h11301-wp-pkcs-12v1-1-personal-information-exchange-syntax.pdf>

DI: *OPC Unified Architecture for Devices (DI)*

<https://opcfoundation.org/developer-tools/specifications-unified-architecture/opc-unified-architecture-for-devices-di/>

ADI: *OPC Unified Architecture for Analyzer Devices (ADI)*

<https://opcfoundation.org/developer-tools/specifications-unified-architecture/opc-unified-architecture-for-analyzer-devices-adi/>

PLCopen: *OPC Unified Architecture / PLCopen Information Model*

<https://opcfoundation.org/developer-tools/specifications-unified-architecture/opc-unified-architecture-plcopen-information-model/>

FDI: *OPC Unified Architecture for FDI*

<https://opcfoundation.org/developer-tools/specifications-unified-architecture/opc-unified-architecture-for-fdi/>

ISA-95: *ISA-95 Common Object Model*

<https://opcfoundation.org/developer-tools/specifications-unified-architecture/isa-95-common-object-model/>

3 Terms, definitions, abbreviated terms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC TR 62541-1, IEC TR 62541-2, IEC 62541-3, IEC 62541-4, IEC 62541-6 and IEC 62541-9 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

3.1.1

Certificate Group

context used to describe the *Trust List* and *Certificate(s)* associated with an *Application*

3.1.2

Certificate Request

PKCS #10 encoded structure used to request a new *Certificate* from a *Certificate Authority*

3.1.3

KeyCredential

unique identifier and a secret used to access a *Server*, an *Authorization Service* or a *Broker*

Note 1 to entry: A user name and password is an example of a credential.

3.1.4

KeyCredentialService

software application that provides *KeyCredentials* needed to access a *Server*, an *Authorization Service* or a *Broker*

3.1.5

DirectoryService

software application, or a set of applications, that stores and organizes information about resources such as computers or services

3.1.6

DiscoveryServer

Application that maintains a list of OPC UA *Servers* that are available on the network and provides mechanisms for *Clients* to obtain this list

3.1.7

DiscoveryUrl

URL for a network *Endpoint* that provides the information required to connect to a *Client* or *Server*

3.1.8

GlobalDiscoveryServer

GDS

DiscoveryServer that maintains a list of OPC UA *Applications* available in an administrative domain

Note 1 to entry: A GDS can also provide certificate management services.

3.1.9

IPAddress

unique number assigned to a network interface that allows Internet Protocol (IP) requests to be routed to that interface

Note 1 to entry: An *IPAddress* for a host can change over time.

3.1.10

LocalDiscoveryServer

LDS

DiscoveryServer that maintains a list of all *Servers* that have registered with it

Note 1 to entry: *Servers* normally register with the LDS on the same host.

3.1.11

LocalDiscoveryServer-ME

LDS-ME

LocalDiscoveryServer that includes the *MulticastExtension*

3.1.12

MulticastExtension

extension to a *LocalDiscoveryServer* that adds support for the *mDNS* protocol

3.1.13

MulticastSubnet

network that allows multicast packets to be sent to all nodes connected to the network

Note 1 to entry: A *MulticastSubnet* is not necessarily the same as a TCP/IP subnet.

3.1.14

Network Service

secured resource on a network that provides functionality used by *Clients* and/or *Servers*

Note 1 to entry: An Authorization Service (AS) is an example of a Network Service.

3.1.15

ServerCapabilityIdentifier

short identifier which uniquely identifies a set of discoverable capabilities supported by a *Server*

Note 1 to entry: The list of the currently defined *ServerCapabilityIdentifiers* is in Annex D.

3.2 Abbreviated terms and symbols

API	application programming interface
CA	certificate authority
CRL	certificate revocation list
CSR	certificate signing request
DER	distinguished encoding rules
DNS	Domain Name System
EST	Enrolment over Secure Transport
GDS	Global Discovery Server
IANA	Internet Assigned Numbers Authority
LDAP	Lightweight Directory Access Protocol
LDS	Local Discovery Server
LDS-ME	Local Discovery Server with Multicast Extension
mDNS	Multicast Domain Name System
NAT	network address translation
PEM	Privacy-Enhanced Mail
PFX	Personal Information Exchange
PKCS	Public-Key Cryptography Standards
SHA1	Secure Hash Algorithm
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UA	Unified Architecture
UDDI	Universal Description, Discovery And Integration

3.3 Conventions for namespaces

This document uses multiple namespaces to define *Nodes*. The following abbreviated terms are used in the definitions for these *Nodes*:

CORE	http://opcfoundation.org/UA/
GDS	http://opcfoundation.org/UA/GDS/

The default namespace for each *Node* is defined at the top of the table. All of the *BrowseNames* in the table use the default namespace unless the *BrowseName* is preceded by one of the above abbreviations.

The *NamespaceMetadataType Object* for the GDS namespace is defined in Table 1.

Table 1 – GDS NamespaceMetadataType Object definition

Attribute	Value		
BrowseName	http://opcfoundation.org/UA/GDS/		
Namespace	GDS (see 3.3)		
TypeDefinition	NamespaceMetadataType defined in IEC 62541-5.		
References	NodeClass	BrowseName	Value
HasProperty	Variable	NamespaceUri	http://opcfoundation.org/UA/GDS/
HasProperty	Variable	NamespaceVersion	1.04
HasProperty	Variable	NamespacePublicationDate	2016-12-31

4 The discovery process

4.1 Overview

The discovery process allows applications to find other applications on the network and then discover how to connect to them. Note that this discussion builds on the discovery related concepts defined in IEC 62541-4. Discoverable applications are generally *Servers*, however, some *Clients* will support reverse connections as described in IEC 62541-6 and want *Servers* to be able to discover them.

Clients and *Servers* can be on the same host, on different hosts in the same subnet, or even on completely different locations in an administrative domain. The following clauses describe the different configurations and how discovery can be accomplished.

The mechanisms for *Clients* to discover *Servers* are specified in 4.3.

The mechanisms for *Servers* to make themselves discoverable are specified in 4.2.

The *Discovery Services* are specified in IEC 62541-4. They are implemented by individual *Servers* and by dedicated *DiscoveryServers*. The following dedicated *DiscoveryServers* provide a way for applications to discover registered OPC UA applications in different situations:

- a *LocalDiscoveryServer* (LDS) maintains discovery information for all applications that have registered with it, usually all applications available on the host that it runs on;
- a *LocalDiscoveryServer* with the *MulticastExtension* (LDS-ME) maintains discovery information for all applications that have been announced on the local *MulticastSubnet*;
- a *GlobalDiscoveryServer* (GDS) maintains discovery information for applications available in an administrative domain.

LDS and LDS-ME are specified in Clause 5. The GDS is specified in Clause 6.

Annex A provides guidelines on the use of firewalls in discovery. Annex B defines the Global Discovery NodeSet. Annex F defines a directory layout for applications to store their *Certificates* on a file system. Annex G provides instructions for the application installation procedure. Annex H describes how Certificate management described in this document compares to RFC7030.

4.2 Registration and announcement of Applications

4.2.1 Overview

This subclause describes how an application registers itself so it can be discovered. Most *Applications* will want other applications to discover them. *Applications* that do not wish to be discovered openly should not register with a *DiscoveryServer*. In this case, such *Applications* should only publish a *DiscoveryUrl* via some out-of-band mechanism to be discovered by specific *Applications*.

4.2.2 Hosts with a LocalDiscoveryServer

Applications register themselves with the LDS on the same host if they wish to be discovered. The registration ensures that the applications is visible for local discovery (see 4.3.4) and *MulticastSubnet* discovery if the LDS is an LDS-ME (see 4.3.5).

The OPC UA Standard (IEC 62541-4) defines a *RegisterServer2 Service*, which provides additional registration information. All *Applications* and the *LocalDiscoveryServer* shall support the *RegisterServer2 Service* and, for backwards compatibility, the older *RegisterServer Service*. If an *Application* encounters an older LDS that returns a *Bad_ServiceUnsupported* error when calling *RegisterServer2 Service*, it shall try again with *RegisterServer Service*.

The *RegisterServer2 Service* allows the *Application* to specify zero or more *ServerCapability Identifiers*. *ServerCapabilityIdentifiers* are short, string identifiers of well-known OPC UA features. *Applications* can use these identifiers as a filter during discovery.

The set of known *ServerCapabilityIdentifiers* is specified in Annex D and is limited to features that are considered to be important enough to report before an application makes a connection. For example, support for the GDS information model or the Alarms information model are *Server* capabilities that have a *ServerCapabilityIdentifier* defined.

Before an application registers with the LDS, it should call the *GetEndpoints Service* and choose the most secure endpoint supported by the LDS and then call *RegisterServer2* or *RegisterServer*.

Registration with LDS or LDS-ME is illustrated in Figure 1.

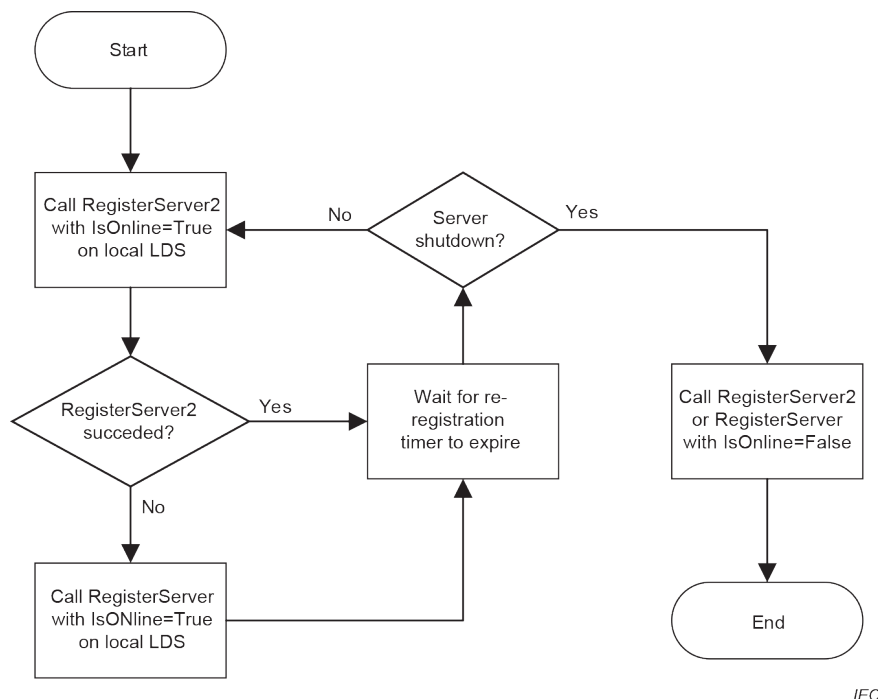


Figure 1 – The Registration process with an LDS

See IEC 62541-4 for more information on the re-registration timer and the *IsOnline* flag.

4.2.3 Hosts without a LocalDiscoveryServer

Dedicated systems (usually embedded systems) with exactly one *Server* installed may not have a separate LDS. Such *Servers* shall become their own LDS or LDS-ME by implementing *FindServers* and *GetEndpoints* *Services* at the well-known address for an LDS. They should also announce themselves on the *MulticastSubnet* with a basic *MulticastExtension*. This requires a small subset of an mDNS Responder (see *mDNS* and Annex C) that announces the *Server* and responds to mDNS probes. The *Server* may not provide the caching and address resolution implemented by a full mDNS Responder.

4.3 The discovery process for Clients to find Servers

4.3.1 Overview

The discovery process allows *Clients* to find *Servers* on the network and then discover how to connect to them. Once a *Client* has this information, it can save it and use it to connect directly to the *Server* again without going through the discovery process. *Clients* that cannot connect with the saved connection information should assume the *Server* configuration has changed and therefore repeat the discovery process.

A *Client* has several choices for finding *Servers*:

- out-of-band discovery (i.e. entry into a GUI) of a *DiscoveryUrl* for a *Server*;
- calling *FindServers* on the LDS installed on the *Client* host;
- calling *FindServers* on a remote LDS, where the *HostName* for the remote host is manually entered;
- calling *FindServersOnNetwork* (see IEC 62541-4) on the LDS-ME installed on *Client* host;
- supporting the LDS-ME functionality locally in the *Client*;
- searching for *Servers* known to a *GlobalDiscoveryServer*.

The *DiscoveryUrl* provides all of the information a *Client* needs to connect to a *DiscoveryEndpoint* (see 4.3.3).

4.3.2 Security

Clients should be aware of rogue *DiscoveryServers* that might direct them to rogue *Servers*. *Clients* can use the SSL/TLS server certificate (if available) to verify that the *DiscoveryServer* is a server that they trust and/or ensure that they trust any *Server* provided by the *DiscoveryServer*. See IEC TR 62541-2 for a detailed discussion of these issues.

4.3.3 Simple Discovery with a DiscoveryUrl

Every *Server* has one or more *DiscoveryUrls* that allow access to its *Endpoints*. Once a *Client* obtains (e.g. via manual entry into a form) the *DiscoveryUrl* for the *Server*, it reads the *EndpointDescriptions* using the *GetEndpoints Service* defined in IEC 62541-4.

The discovery process for this scenario is illustrated in Figure 2.

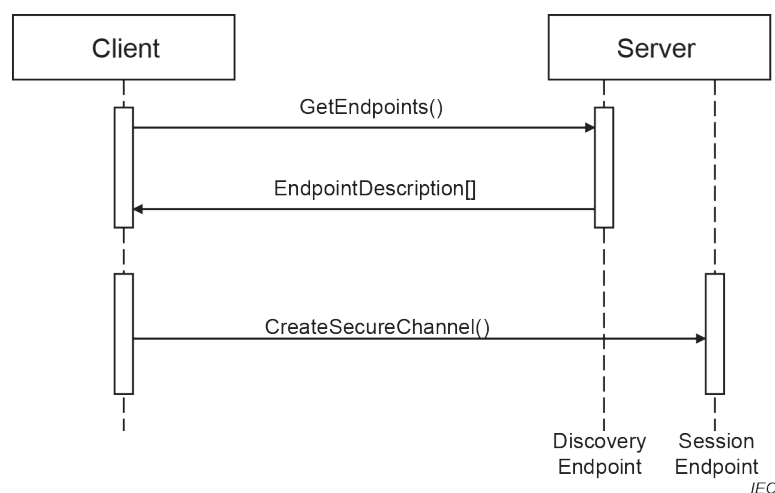


Figure 2 – The simple Discovery process

4.3.4 Local Discovery

In many cases, *Clients* do not know which *Servers* exist but possibly know which hosts can have *Servers* on them. In this situation, the *Client* will look for the *LocalDiscoveryServer* on a host by constructing a *DiscoveryUrl* using the well-known addresses defined in IEC 62541-6.

If a *Client* finds a *LocalDiscoveryServer* then it will call the *FindServers Service* on the LDS to obtain a list of *Servers* and their *DiscoveryUrls*. The *Client* would then call the *GetEndpoints* service for one of the *Servers* returned. The discovery process for this scenario is illustrated in Figure 3.

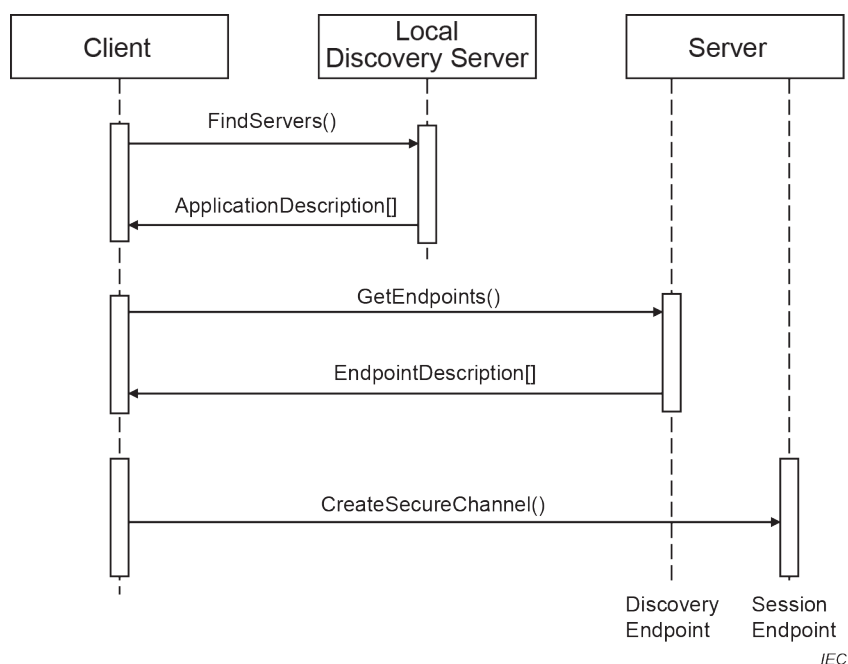


Figure 3 – The Local Discovery process

4.3.5 MulticastSubnet Discovery

In some situations *Clients* will not know which hosts have *Servers*. In these situations the *Client* will look for a *LocalDiscoveryServer* with the *MulticastExtension* on its local host and requests a list of *DiscoveryUrls* for *Servers* and *DiscoveryServers* available on the *MulticastSubnet*.

The discovery process for this scenario is illustrated in Figure 4.

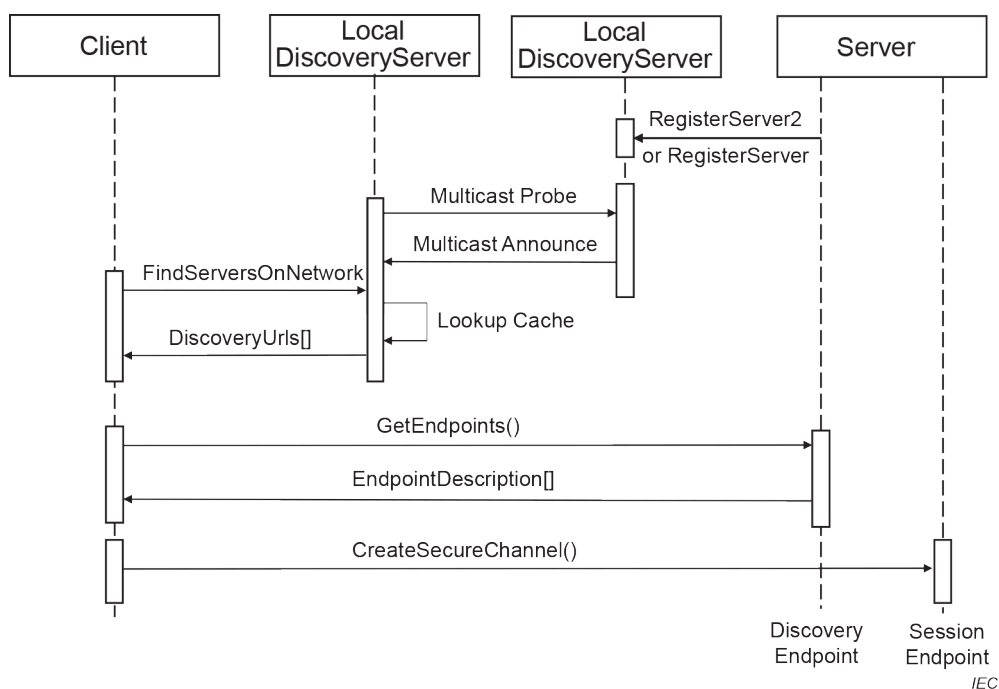


Figure 4 – The MulticastSubnet Discovery process

In this scenario, the *Server* uses the *RegisterServer2 Service* to tell a *LocalDiscoveryServer* to announce the *Server* on the *MulticastSubnet*. The *Client* will receive the *DiscoveryUrl* and *ServerCapabilityIdentifiers* for the *Server* when it calls *FindServersOnNetwork* and then connects directly to the *Server*. When a *Client* calls *FindServers* it only receives the *Servers* running on the same host as the LDS.

Clients running on embedded systems may not have a LDS-ME available on the system. These *Clients* can support an mDNS Responder that understands how OPC UA concepts are mapped to mDNS messages and maintains the same table of *Servers* as maintained by the LDS-ME. This mapping is described in Annex C.

4.3.6 Global Discovery

A GDS is an OPC UA *Server* that allows *Clients* to search for *Servers* in the administrative domain. It may also provide Certificate Services (see Clause 7). It provides *Methods* that allow applications to search for other applications (See Clause 6). To access the GDS, the *Client* will create a *Session* with the *GDS* and use the *Call* service to invoke the *QueryApplications Method* (see 6.3.11). The *QueryServers Method* is similar to the *FindServers* service except that it provides more advanced search and filter criteria. The discovery process is illustrated in Figure 5.

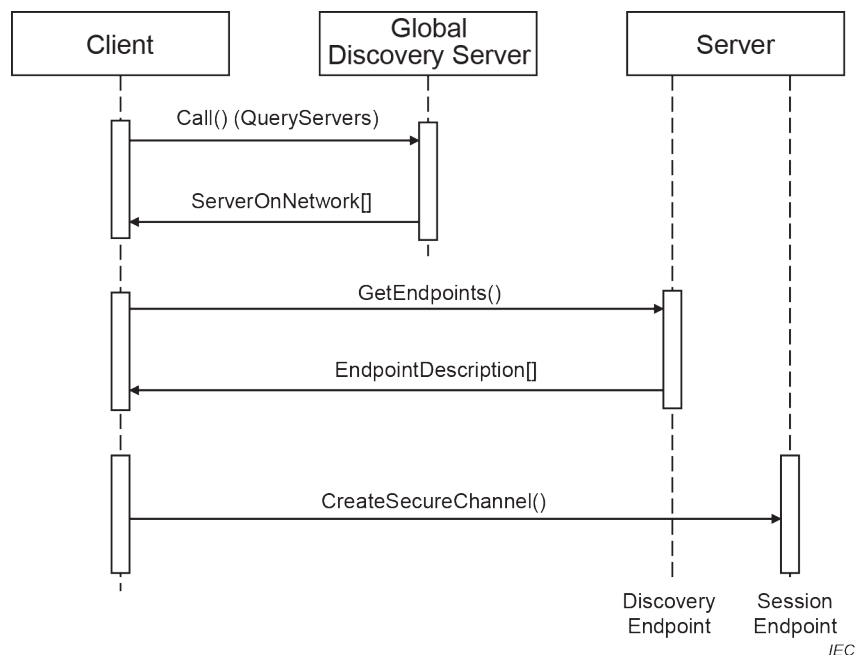


Figure 5 – The Global Discovery process

The GDS may be coupled with any of the previous network architectures. For each *MulticastSubnet*, one or more LDSs may be registered with a GDS.

The *Client* can also be configured with the URL of the GDS using an out of band mechanism.

The complete discovery process is shown in Figure 6.

4.3.7 Combined Discovery Process for Clients

The use cases in the preceding clauses imply a number of choices that have to be made by *Clients* when a *Client* needs to connect to a *Server*. These choices are combined together in Figure 6.



Figure 6 – The Discovery Process for Clients

FindServersOnNetwork can be called on the local LDS, however, it can also be called on a remote LDS which is part of a different *MulticastSubnet*.

An out-of-band mechanism is a way to find a URL or a *HostName* that is not described by this document. For example, a user could manually enter a URL or use system specific APIs to browse the network neighbourhood.

A *Client* that goes through the discovery process can save the URL that was discovered. If the *Client* restarts later, it can use that URL and bypass the discovery process. If reconnection fails, the *Client* will have to go through the process again.

5 Local Discovery Server

5.1 Overview

Each host that could have multiple discoverable applications installed should have a standalone *LocalDiscoveryServer* installed. The *LocalDiscoveryServer* shall expose one or more *Endpoints* which support the *FindServers* and *GetEndpoints* services defined in IEC 62541-4 for all applications on the host. In addition, the *LocalDiscoveryServer* shall provide at least one *Endpoint* which implements the *RegisterServer* service for these applications.

In systems (usually embedded systems) with exactly one *Server* installed this *Server* may also be the LDS (see 4.2.3).

An LDS-ME will announce all applications that it knows about on the local *MulticastSubnet*. In order to support this, a *LocalDiscoveryServer* supports the *RegisterServer2 Service* defined in IEC 62541-4. For backward compatibility, a *LocalDiscoveryServer* also supports the *RegisterServer Service* which is defined in IEC 62541-4.

Each host with OPC UA Applications (Clients and Servers) installed should have a *LocalDiscoveryServer* with a *MulticastExtension*.

The *MulticastExtension* incorporates the functionality of the mDNS Responder described in the Multicast DNS (mDNS) specification (see *mDNS*). In addition, the *LocalDiscoveryServer* that supports the *MulticastExtension* supports the *FindServersOnNetwork* Service described in IEC 62541-4.

5.2 Security considerations for Multicast DNS

The Multicast DNS (mDNS) specification is used for various commercial and consumer applications. This provides a benefit in that implementations exist; however, system administrators could choose to disable Multicast DNS operations. For this reason, *Applications* shall not rely on Multicast DNS capabilities.

Multicast DNS operations are insecure because of their nature; therefore, they should be disabled in environments where an attacker could cause problems by impersonating another host. This risk is minimized if OPC UA security is enabled and all *Applications* use *Certificate TrustLists* to control access.

6 Global Discovery Server

6.1 Overview

The *LocalDiscoveryServer* is useful for networks where the host names can be discovered. However, this is typically not the case in large systems with multiple servers on multiple subnets. For this reason, there is a need for an enterprise wide *DiscoveryServer* called a *GlobalDiscoveryServer*. The *GlobalDiscoveryServer* (GDS) is an OPC UA *Server* that allows *Clients* to search for *Servers* in the administrative domain. It provides methods that allow applications to register themselves and to search for other applications.

The essential element of a *GlobalDiscoveryServer* (GDS) is that it can provide the *Certificate* management services defined in Clause 7. These services can simplify *Certificate* management even in medium to small systems; therefore, a GDS can be deployed in smaller systems. Different implementations are expected. Some of them will likely provide a front-end to an existing *DirectoryService* such as LDAP (see Annex E). By standardizing on an OPC UA based interface, OPC UA *Clients* do not need to have knowledge of different *DirectoryServices*.

If an administrator registers a *LocalDiscoveryServer* with the GDS, then the GDS shall periodically update its database by calling *FindServersOnNetwork* or *FindServers* on the LDS. Figure 7 shows the relationship between a GDS and the LDS-ME or LDS.

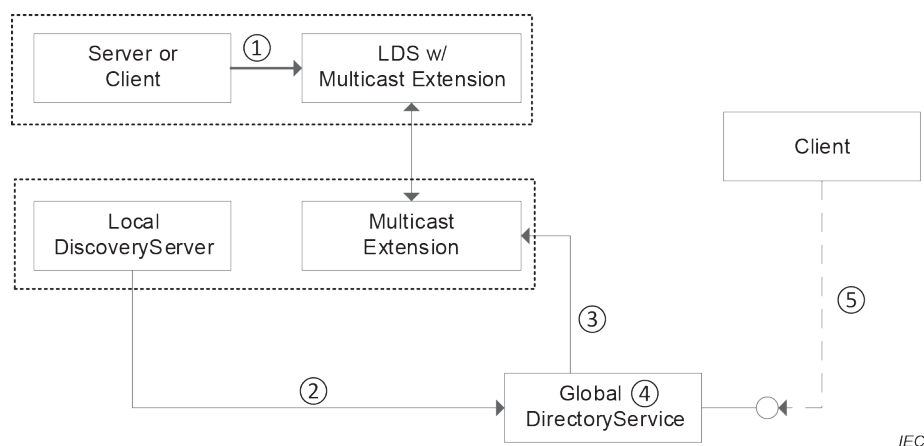


Figure 7 – The relationship between GDS and other components

The steps shown in Figure 7 are:

- 1) The Server calls *RegisterServer2* on the LDS running on the same machine.
- 2) The administrator registers LDS-ME installations with the GDS.
- 3) The GDS calls *FindServersOnNetwork* on the LDS-ME to find all applications on the same *MulticastSubnet*.
- 4) The GDS creates a record for each application returned by the LDS-ME. These records shall be approved before they are made available to *Clients* of the GDS. This approval can be obtained from an *Administrator* or the GDS can connect to the *Server* and verifies that it has a trusted *Certificate*.
- 5) The *Client* calls *QueryServers Method* on the GDS to discover applications.

The *Information Model* used for registration and discovery is shown in 6.2. Any *Client* shall be able to call the *QueryServers Method* to find applications known to GDS. The complete definitions for each of the types used are described in 7.5.

6.2 Network architectures

6.2.1 Overview

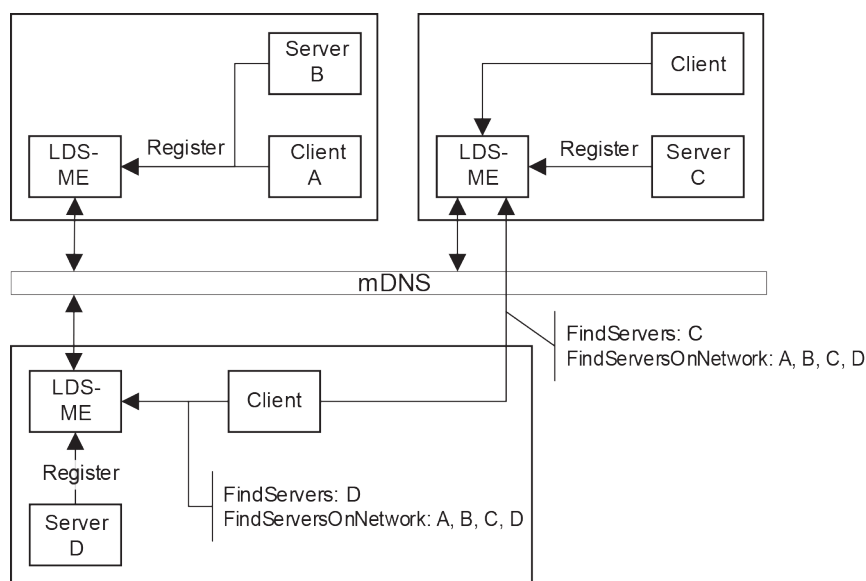
The discovery mechanisms defined in this document are expected to be used in many different network architectures. The following three architectures are illustrated:

- Single *MulticastSubnet*;
- Multiple *MulticastSubnets*;
- No *MulticastSubnet* (or multiple *MulticastSubnets* with exactly one host each).

A *MulticastSubnet* is a network segment where all hosts on the segment can receive multicast packets from the other hosts on the segment. A physical LAN segment is typically a *MulticastSubnet* unless the administrator has specifically disabled multicast communication. In some cases, multiple physical LAN segments can be connected as a Single *MulticastSubnet*.

6.2.2 Single MulticastSubnet

The Single *MulticastSubnet* Architecture is shown in Figure 8.



IEC

Figure 8 – The Single MulticastSubnet architecture

In this architecture, every host has an LDS-ME and uses mDNS to maintain a cache of the applications on the *MulticastSubnet*. A *Client* can call *FindServersOnNetwork* on any LDS-ME and receive the same set of applications. When a *Client* calls *FindServers*, it only receives the applications running on the same host as the LDS.

6.2.3 Multiple MulticastSubnet

The Multiple *MulticastSubnet* Architecture is shown in Figure 9.

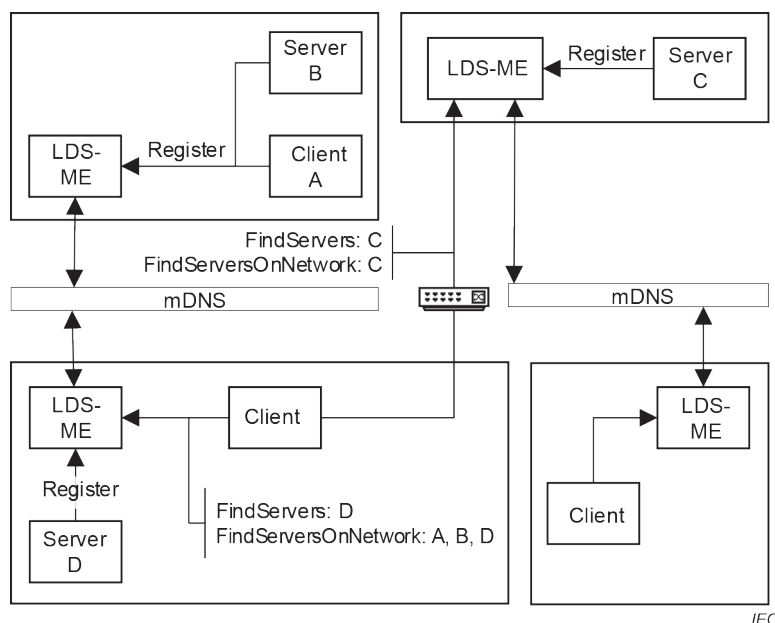


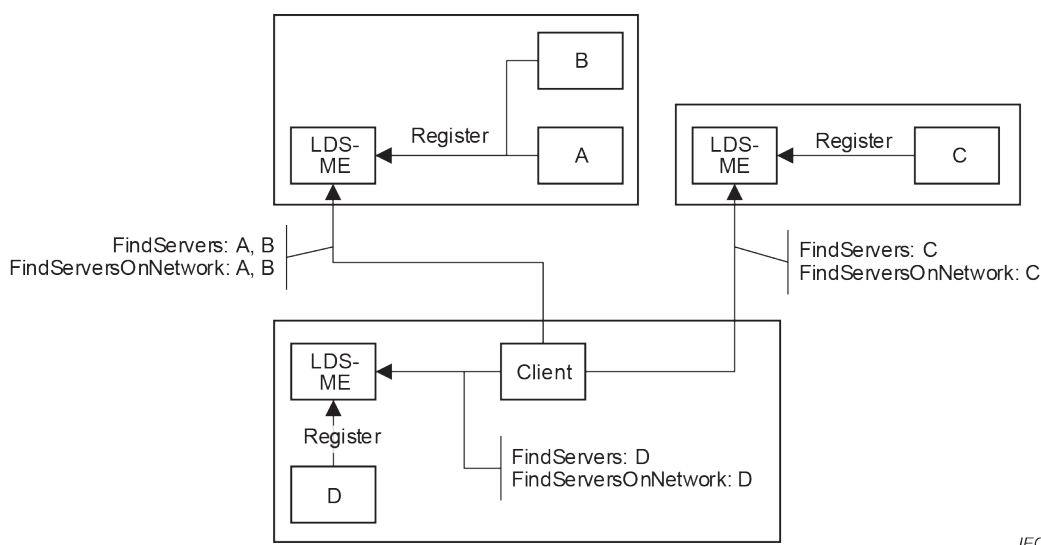
Figure 9 – The Multiple MulticastSubnet architecture

This architecture is the same as the previous architecture, except in this architecture the mDNS messages do not pass through routers connecting the *MulticastSubnets*. This means that a *Client* calling *FindServersOnNetwork* will only receive a list of applications running on the *MulticastSubnets* that the LDS-ME is connected to.

A *Client* that wants to connect to a remote *MulticastSubnet* shall use out of band discovery (i.e. manual entry) of a *HostName* or *DiscoveryUrl*. Once a *Client* finds an LDS-ME on a remote *MulticastSubnet*, it can use *FindServersOnNetwork* to discover all applications on that *MulticastSubnet*.

6.2.4 No MulticastSubnet

The No *MulticastSubnet* Architecture is shown in Figure 10.



IEC

Figure 10 – The No MulticastSubnet architecture

In this architecture, the mDNS is not used at all because the Administrator has disabled multicast at a network level or by turning off multicast capabilities of each LDS-ME.

A *Client* that wants to discover applications needs to use an out of band mechanism to find the *HostName* and call *FindServers* on the LDS of that host. *FindServersOnNetwork* can also work, but it will never return more than what *FindServers* returns.

6.2.5 Domain Names and MulticastSubnets

The mDNS specification requires that a fully qualified domain name be announced on the network. If a *Server* is not configured with a fully qualified domain name, then mDNS requires that the "local" top level domain be appended to the domain names. The "local" top level domain indicates that the domain can only be considered to be unique within the subnet where the domain name was used. This means *Clients* need to be aware that URLs received from any LDS-ME other than the one on the *Client's* machine could contain "local" domains that are not reachable or will connect to a different machine with the same domain name that happens to be on the same subnet as the *Client*. It is recommended that *Clients* ignore all URLs with the "local" top level domain unless they are returned from the LDS-ME running on the same machine.

System administrators can eliminate this problem by configuring a normal DNS with the fully qualified domain names for all machines that need to be accessed by *Clients* outside the *MulticastSubnet*.

Servers configured with fully qualified domain names should specify the fully qualified domain name in its *ApplicationInstance Certificate*. *Servers* shall not specify domains with the "local" top level domain in their *Certificate*. *Clients* using a URL returned from an LDS-ME shall ignore the "local" top level domain when checking the domain against the *Server Certificate*.

Note that domain name validation is a necessary but not sufficient check against rogue *Servers* or man-in-the-middle attacks when *Server Certificates* do not contain fully qualified domain names. The *Certificate* trust relationship established by administrators is the primary mechanism used to protect against these risks.

6.3 Information Model

6.3.1 Overview

The *GlobalDiscoveryServer Information Model* used for *discovery* is shown in Figure 11. Most of the interactions between the *GlobalDiscoveryServer* and *Application* administrator or the *Client* will be via *Methods* defined on the *Directory* folder.

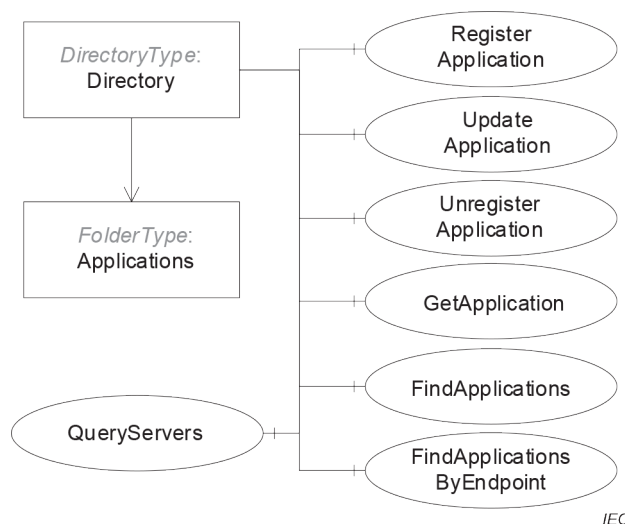


Figure 11 – The Address Space for the GDS

6.3.2 Directory

This *Object* is the root of the *GlobalDiscoveryServer AddressSpace*, and it is the target of an *Organizes* reference from the *Objects* folder defined in IEC 62541-5. It organizes the information that can be accessed into subfolders. The implementation of a GDS can customize and organize the folders in any manner it desires. For example, folders can exist for information models, or for optional services or for various locations in an administrative domain. It is defined in Table 2.

Table 2 – Directory Object definition

Attribute	Value				
BrowseName	Directory				
Namespace	GDS (see 3.3)				
TypeDefinition	DirectoryType defined in 6.3.3.				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modelling Rule

6.3.3 DirectoryType

DirectoryType is the *ObjectType* for the root of the *GlobalDiscoveryServer AddressSpace*. It organizes the information that can be accessed into subfolders. It also provides methods that allow applications to register or find applications. It is defined in Table 3.

Table 3 – DirectoryType definition

Attribute	Value				
BrowseName	DirectoryType				
Namespace	GDS (see 3.3)				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>FolderType</i> defined in IEC 62541-5.					
Organizes	Object	Applications	-	FolderType	Mandatory
HasComponent	Method	FindApplications	Defined in 6.3.4		Mandatory
HasComponent	Method	RegisterApplication	Defined in 6.3.6		Mandatory
HasComponent	Method	UpdateApplication	Defined in 6.3.7		Mandatory
HasComponent	Method	UnregisterApplication	Defined in 6.3.8		Mandatory
HasComponent	Method	GetApplication	Defined in 6.3.9		Mandatory
HasComponent	Method	QueryApplications	Defined in 6.3.10		Mandatory
HasComponent	Method	QueryServers	Defined in 6.3.11		Mandatory

The *Applications* folder may contain *Objects* representing the *Applications* known to the GDS. These *Objects* may be organized into subfolders as determined by the GDS. Some characteristics for organizing applications are the networks, the physical location, or the supported profiles. The *QueryServers Method* can be used to search for OPC UA *Applications* based on various criteria.

A GDS is not required to expose its *Applications* as browsable *Objects* in its *AddressSpace*, however, each *Application* shall have a unique *NodeId* which can be passed to *Methods* used to administer the GDS.

The *FindApplications Method* returns the *Applications* associated with an *ApplicationUri*. It can be called by any *Client* application.

The *RegisterApplication Method* is used to add a new *Application* to the GDS. It requires administrative privileges.

The *UpdateApplication Method* is used to update an existing *Application* in the GDS. It requires administrative privileges.

The *UnregisterApplication Method* is used to remove an *Application* from the GDS. It requires administrative privileges.

The *QueryApplications Method* is used to find *Client* or *Server* applications that meet the criteria provided. This *Method* replaces the *QueryServers Method*.

The *QueryServers Method* is used to find *Servers* that meet the criteria specified. It can be called by any *Client* application. This *Method* has been replaced by the *QueryApplications Method*.

6.3.4 FindApplications

FindApplications is used to find the *ApplicationId* for an OPC UA *Application* known to the GDS. In normal situations, the list of records returned will not have more than one entry; however, system configuration errors can create situations where the GDS has multiple entries for a single *ApplicationUri*. If this happens, a human will likely have to look at records to determine which record is the true match for the *ApplicationUri*.

If the returned array is null or zero length then the GDS does not have an entry for the *ApplicationUri*.

Signature

```
FindApplications(
    [in] String applicationUri
    [out] ApplicationRecordDataType[] applications
);
```

Argument	Description
applicationUri	The <i>ApplicationUri</i> that identifies the <i>Application</i> of interest.
applications	A list of application records that match the <i>ApplicationUri</i> . The <i>ApplicationRecordDataType</i> is defined in 6.3.5.

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_UserAccessDenied	The current user does not have the rights required.

Table 4 specifies the *AddressSpace* representation for the *FindApplications Method*.

Table 4 – FindApplications Method AddressSpace definition

Attribute	Value				
BrowseName	FindApplications				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

6.3.5 ApplicationRecordDataType

This type defines a *DataType* which represents a record in the GDS (see Table 5).

Table 5 – ApplicationRecordDataType definition

Name	Type	Value
applicationId	NodeId	The unique identifier assigned by the GDS to the record. This <i>NodeId</i> may be passed to other <i>Methods</i> .
applicationUri	String	The URI for the <i>Application</i> associated with the record.
applicationType	ApplicationType	The type of application. This type is defined in IEC 62541-4.
applicationNames	LocalizedText[]	One or more localized names for the application. The first element is the default <i>ApplicationName</i> for the application when a non-localized name is needed.
productUri	String	A globally unique URI for the product associated with the application. This URI is assigned by the vendor of the application.
discoveryUrls	String[]	The list of discovery URLs for an application. The first element is the default if a <i>Client</i> needs to choose one URL. The first HTTPS URL specifies the domain used as the Common Name of HTTPS <i>Certificates</i> . If the <i>ApplicationType</i> is <i>Client</i> then all of the URLs shall have the 'inv+' prefix which indicates they support reverse connect.
serverCapability Identifiers	String[]	The list of server capability identifiers for the application. The allowed values are defined in Annex D.

6.3.6 RegisterApplication

RegisterApplication is used to register a new *Application* Instance with a *GlobalDiscoveryServer*.

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

Servers that support transparent redundancy shall register as a single application and pass the *DiscoveryUrls* for all available instances and/or network paths.

RegisterApplication will create duplicate records if the *ApplicationUri* already exists since misconfiguration of applications can result in different applications having the same *ApplicationUri*. Before calling this *Method* the *Client* shall call *FindApplications* to check if a record for the application it is using already exists. If records are found that appear to belong to different applications (e.g. the *DiscoveryUrls* are different) then the *Client* shall report a warning before continuing.

If registration was successful and auditing is supported, the GDS shall generate the *ApplicationRegistrationChangedAuditEventType* (see 6.3.12).

Signature

```
RegisterApplication (
    [in] ApplicationRecordDataType application
    [out] NodeId applicationId
);
```

Argument	Description
application	The application that is to be registered with the <i>GlobalDiscoveryServer</i> .
applicationId	A unique identifier for the registered <i>Application</i> . This identifier is persistent and is used in other <i>Methods</i> used to administer applications.

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_InvalidArgument	The application or one of the fields of the application record is not valid. The text associated with the error shall indicate the exact problem.
Bad_UserAccessDenied	The current user does not have the rights required.

Table 6 specifies the *AddressSpace* representation for the *RegisterApplication Method*.

Table 6 – RegisterApplication Method AddressSpace definition

Attribute	Value				
BrowseName	RegisterApplication				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

6.3.7 UpdateApplication

UpdateApplication is used to update an existing *Application* in a *GlobalDiscoveryServer*.

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

If the update was successful and auditing is supported, the GDS shall generate the *ApplicationRegistrationChangedAuditEventType* (see 6.3.12).

Signature

```
UpdateApplication (
    [in] ApplicationRecordDataType application
);
```

Argument	Description
application	The application that is to be updated in the GDS database.

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_NotFound	The applicationId is not known to the GDS.
Bad_InvalidArgument	The application or one of the fields of the application record is not valid. The text associated with the error shall indicate the exact problem.
Bad_UserAccessDenied	The current user does not have the rights required.

Table 7 specifies the *AddressSpace* representation for the *UpdateApplication Method*.

Table 7 – UpdateApplication Method AddressSpace definition

Attribute	Value				
BrowseName	UpdateApplication				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

6.3.8 UnregisterApplication

UnregisterApplication is used to remove an *Application* from a *GlobalDiscoveryServer*.

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

A *Server Application* that is unregistered may be automatically added again if the GDS is configured to populate itself by calling *FindServersOnNetwork* and the *Server Application* is still registering with its local LDS.

If un-registration was successful and auditing is supported, the GDS shall generate the *ApplicationRegistrationChangedAuditEventType* (see 6.3.12).

Signature

```
UnregisterApplication (
    [in] NodeId applicationId
);
```

Argument	Description
applicationId	The identifier assigned by the GDS to the <i>Application</i> .

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_NotFound	The <i>ApplicationId</i> is not known to the GDS.
Bad_UserAccessDenied	The current user does not have the rights required.

Table 8 specifies the *AddressSpace* representation for the *UnregisterApplication Method*.

Table 8 – UnregisterApplication Method AddressSpace definition

Attribute	Value				
BrowseName	UnregisterApplication				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

6.3.9 GetApplication

GetApplication is used to find an OPC UA *Application* known to the GDS.

Signature

```
GetApplication (
    [in] NodeId applicationId
    [out] ApplicationRecordDataType application
);
```

Argument	Description
applicationId	The <i>ApplicationId</i> that identifies the <i>Application</i> of interest.
application	The application record that matches the <i>ApplicationId</i> . The <i>ApplicationRecordDataType</i> is defined in 6.3.5.

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_NotFound	The no record found for the specified <i>ApplicationId</i> .
Bad_UserAccessDenied	The current user does not have the rights required.

Table 9 specifies the *AddressSpace* representation for the *GetApplication Method*.

Table 9 – GetApplication Method AddressSpace definition

Attribute	Value				
BrowseName	GetApplication				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

6.3.10 QueryApplications

QueryApplications is used to find *Client* or *Server* applications that meet the specified filters. The only *Clients* returns are those that support the reverse connection capability described in IEC 62541-6.

QueryApplications returns *ApplicationDescriptions* instead of the *ServerOnNetwork Structures* returned by *QueryServers*. This is more useful to some *Clients* because it matches the return type of *FindServers*.

Any *Client* is able to call this *Method*, however, the set of results returned may be restricted based on the *Client's* user credentials.

The applications returned shall pass all of the filters provided (i.e. the filters are combined in an AND operation). The *capabilities* parameter is an array and an application will pass this filter if it supports all of the specified capabilities.

Each time the GDS creates or updates an application record it shall assign a monotonically increasing identifier to the record. This allows *Clients* to request records in batches by specifying the identifier for the last record received in the last call to *QueryApplications*. To support this, the GDS shall return records in order starting from the lowest record identifier. The GDS shall also return the last time the counter was reset. If a *Client* detects that this time is more recent than the last time the *Client* called the *Method* it shall call the *Method* again with a *startingRecordId* of 0.

Signature

```
QueryApplications (
    [in] UInt32 startingRecordId
    [in] UInt32 maxRecordsToReturn
    [in] String applicationName
    [in] String applicationUri
    [in] UInt32 applicationType
    [in] String productUri
    [in] String[] capabilities
    [out] DateTime lastCounterResetTime
    [out] UInt32 nextRecordId
    [out] ApplicationDescription[] applications
);
```

Argument	Description
INPUTS	
startingRecordId	Only records with an identifier greater than this number will be returned. Specify 0 to start with the first record in the database.
maxRecordsToReturn	The maximum number of records to return in the response. 0 indicates that there is no limit.
applicationName	The <i>ApplicationName</i> of the applications to return. Supports the syntax used by the LIKE <i>FilterOperator</i> described in IEC 62541-4. Not used if an empty string is specified. The filter is only applied to the default <i>ApplicationName</i> .
applicationUri	The <i>ApplicationUri</i> of the applications to return. Supports the syntax used by the LIKE <i>FilterOperator</i> described in IEC 62541-4. Not used if an empty string is specified.
applicationType	A mask indicating what types of applications are returned. The mask values are: 0x1 – Servers; 0x2 – Clients; If the mask is 0 then all applications are returned.
productUri	The <i>ProductUri</i> of the applications to return. Supports the syntax used by the LIKE <i>FilterOperator</i> described in IEC 62541-4. Not used if an empty string is specified.
capabilities	The capabilities supported by the applications returned. The applications returned shall support all of the capabilities specified. If no capabilities are provided this filter is not used.
OUTPUTS	
lastCounterResetTime	The last time the counters were reset.
nextRecordId	The identifier of the next record. It is passed as the <i>startingRecordId</i> in subsequent calls to <i>QueryApplications</i> to fetch the next batch of records. It is 0 if there are no more records to return.
applications	A list of <i>Applications</i> that meet the criteria. The <i>ApplicationDescription</i> structure is defined in IEC 62541-4.

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_UserAccessDenied	The current user does not have the rights required.

Table 10 specifies the *AddressSpace* representation for the *QueryApplications* Method.

Table 10 – QueryApplications Method AddressSpace definition

Attribute	Value				
BrowseName	QueryApplications				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

6.3.11 QueryServers (deprecated)

QueryServers is used to find *Server* applications that meet the specified filters.

Any *Client* is able to call this *Method*, however, the set of results returned may be restricted based on the *Client's* user credentials.

The applications returned shall pass all of the filters provided (i.e. the filters are combined in an AND operation). The *serverCapabilities* parameter is an array and an application will pass this filter if it supports all of the specified capabilities.

Each time the GDS creates or updates an application record it shall assign a monotonically increasing identifier to the record. This allows *Clients* to request records in batches by specifying the identifier for the last record received in the last call to *QueryServers*. To support this, the GDS shall return records in order starting from the lowest record identifier. The GDS shall also return the last time the counter was reset. If a *Client* detects that this time is more recent than the last time the *Client* called the *Method* it shall call the *Method* again with a *startingRecordId* of 0.

Signature

```

QueryServers (
    [in] UInt32 startingRecordId
    [in] UInt32 maxRecordsToReturn
    [in] String applicationName
    [in] String applicationUri
    [in] String productUri
    [in] String[] serverCapabilities
    [out] DateTime lastCounterResetTime
    [out] ServerOnNetwork[] servers
);

```

Argument	Description
INPUTS	
startingRecordId	Only records with an identifier greater than this number will be returned. Specify 0 to start with the first record in the database.
maxRecordsToReturn	The maximum number of records to return in the response. 0 indicates that there is no limit.
applicationName	The <i>ApplicationName</i> of the <i>Applications</i> to return. Supports the syntax used by the LIKE <i>FilterOperator</i> described in IEC 62541-4. Not used if an empty string is specified. The filter is only applied to the default <i>ApplicationName</i> .
applicationUri	The <i>ApplicationUri</i> of the <i>Servers</i> to return. Supports the syntax used by the LIKE <i>FilterOperator</i> described in IEC 62541-4. Not used if an empty string is specified.
productUri	The <i>ProductUri</i> of the <i>Servers</i> to return. Supports the syntax used by the LIKE <i>FilterOperator</i> described in IEC 62541-4. Not used if an empty string is specified.
serverCapabilities	The applications returned shall support all of the server capabilities specified. If no server capabilities are provided this filter is not used.
OUTPUTS	
lastCounterResetTime	The last time the counters were reset.
servers	A list of <i>Servers</i> which meet the criteria. The <i>ServerOnNetwork</i> structure is defined in IEC 62541-4.

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_UserAccessDenied	The current user does not have the rights required.

Table 11 specifies the *AddressSpace* representation for the *QueryServers Method*.

Table 11 – QueryServers Method AddressSpace definition

Attribute	Value				
BrowseName	QueryServers				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

6.3.12 ApplicationRegistrationChangedAuditEventType

This event is raised when the *RegisterApplication*, *UpdateApplication* or *UnregisterApplication Methods* are called.

Its representation in the *AddressSpace* is formally defined in Table 12.

Table 12 – ApplicationRegistrationChangedAuditEventType definition

Attribute	Value				
BrowseName	ApplicationRegistrationChangedAuditEventType				
Namespace	GDS (see 3.3)				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditUpdateMethodEventType</i> defined in IEC 62541-5.					

This *EventType* inherits all *Properties* of the *AuditUpdateMethodEventType*. Their semantics are defined in IEC 62541-5.

7 Certificate management overview

7.1 Overview

Certificate management functions comprise the management and distribution of certificates and *Trust Lists* for OPC UA Applications. An application that provides the certificate management functions is called *CertificateManager*. GDS and *CertificateManager* will typically be combined in one application. The basic concepts regarding *Certificate* management are described in IEC TR 62541-2.

There are two primary models for *Certificate* management: pull and push management. In pull management, the application acts as a *Client* and uses the *Methods* on the *CertificateManager* to request and update *Certificates* and *Trust Lists*. The application is responsible for ensuring the *Certificates* and *Trust Lists* are kept up to date. In push management, the application acts as a *Server* and exposes *Methods* which the *CertificateManager* can call to update the *Certificates* and *Trust Lists* as required.

The GDS is intended to work in conjunction with different Certificate Management services such as Active Directory. The GDS provides a standard OPC UA based information model that all OPC UA applications can support without needing to know the specifics of a particular Certificate Management system.

The *CertificateManager* should support the following features:

- provisioning (first time setup for a device/application);
- renewal (renewing expired or compromised certificates);
- *Trust List* update (updating the *Trust Lists* including the *Revocation Lists*);
- revocation (removing a device/application from the system).

Although it is generally assumed that *Client* applications will use the Pull model and *Server* applications will use the Push model, this is not required.

During provisioning, the *CertificateManager* shall be able to operate in a mode where any *Client* is allowed to connect securely with any valid *Certificate* and user credentials are used to determine the rights a *Client* has; this eliminates the need to configure *Trust Lists* before connecting to the *CertificateManager* for provisioning.

Application vendors may decide to build the interaction with the *CertificateManager* as a separate component, e.g. as part of an administration application with access to the OPC UA configuration of this *Application*. This is transparent for the *CertificateManager* and will not be considered further in the rest of this clause.

This document does not define how to administer a *CertificateManager* but a *CertificateManager* shall provide an integrated system that includes both push and pull management.

7.2 Pull Management

Pull Management is performed by using the *CertificateManager* information model – in particular the Methods - defined in 7.6. The interactions between *Application* and *CertificateManager* during Pull Management are illustrated in Figure 12.

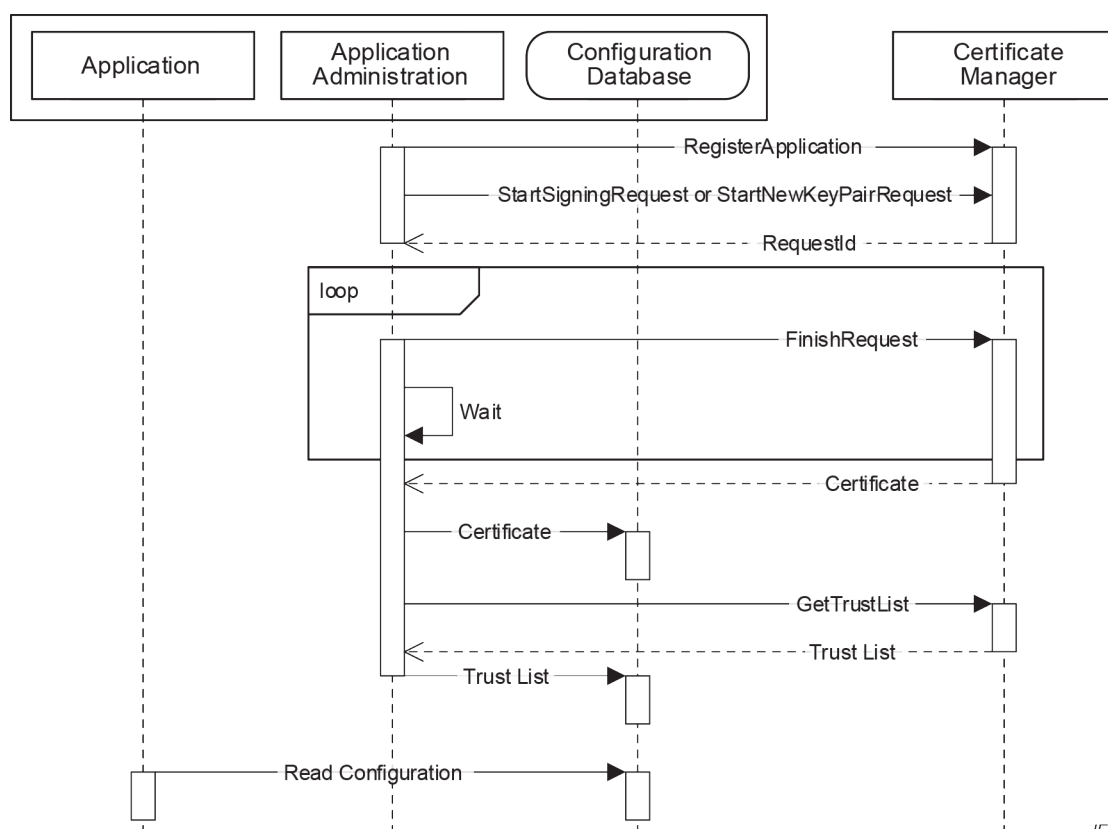


Figure 12 – The Pull Certificate management model

The Application Administration component may be part of the Application or a standalone utility that understands how the Application persists its configuration information in its Configuration Database.

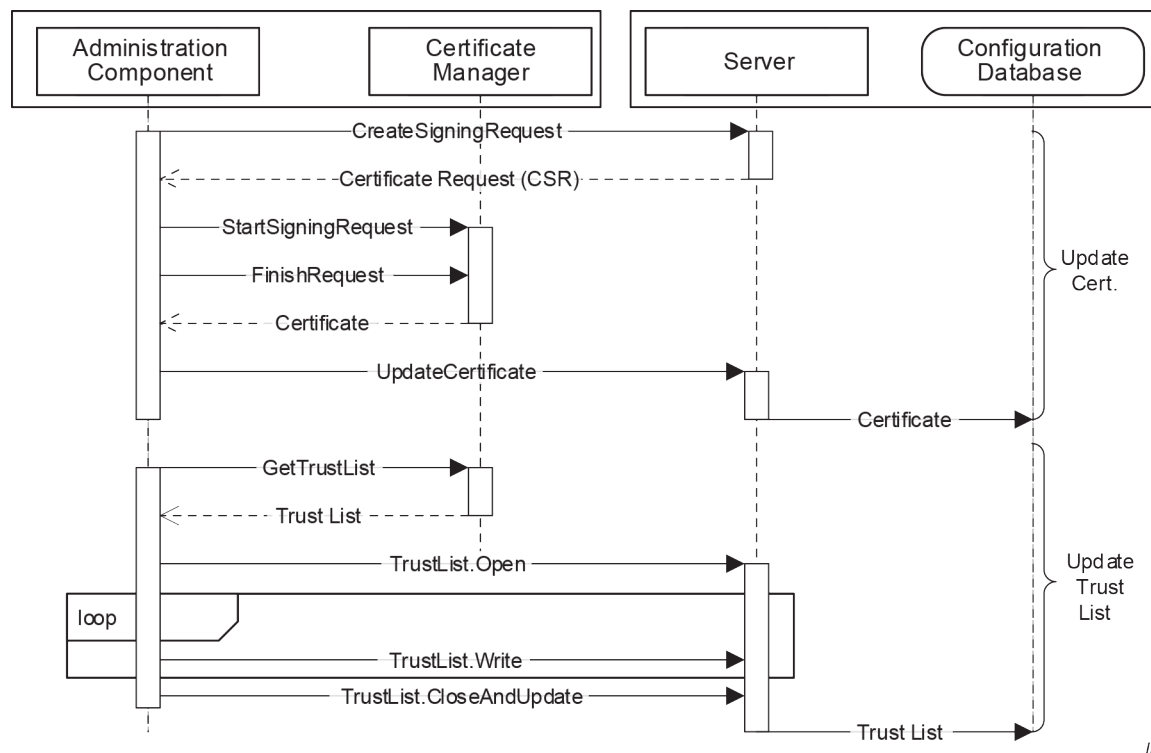
A similar process is used to renew certificates or to periodically update *Trust List*.

Security in Pull management requires an encrypted channel and the use of *Administrator* credentials for the *CertificateManager* that ensure only authorized users can register new *Applications* and request an initial new *Certificate*. Once an *Application* has a *Certificate* it can use this *Certificate* to renew the *Certificate* or to update *Trust Lists* and *Revocation* lists. It is important that a *CertificateManager* does not provide certificate renewals except to the applications that already own the prior certificate.

7.3 Push management

Push management is targeted at *Server* applications and relies on *Methods* defined in 7.7 to get a *Certificate Request* which can be passed onto the *CertificateManager*. After the *CertificateManager* signs the *Certificate*, the new *Certificate* is pushed to the *Server* with the *UpdateCertificate Method*.

The interactions between a *Server Application* and *CertificateManager* during Push Management are illustrated in Figure 13.



IEC

Figure 13 – The Push Certificate management model

The Administration Component may be part of the *CertificateManager* or a standalone utility that uses OPC UA to communicate with the *CertificateManager* (see 7.2 for a more complete description of the interactions required for this use case). The Configuration Database is used by the *Server* to persist its configuration information. The *RegisterApplication Method* (or internal equivalent) is assumed to have been called before the sequence in the diagram starts.

A similar process is used to renew certificates or to periodically update *Trust List*.

Security when using the Push Management Model requires an encrypted channel and the use of Administrator credentials for the *Server* that ensure only authorized users can update *Certificates* or *Trust Lists*. In addition, separate *Administrator* credentials are required for the *CertificateManager* that ensure only authorized users can register new *Servers* and request new *Certificates*.

7.4 Provisioning

Provisioning is the initial installation of an OPC UA *Server* or *Client* into a system in which a GDS is available and managing all certificates. For applications using a *Client* interface, provisioning can be accomplished using a Pull Model. Applications using the *Server* interface can be provisioned using the Push Model.

OPC UA *Servers* will typically auto-generate a self-signed *Certificate* when they first start. They may also have a pre-configured *Trust List* with *Applications* that are allowed to provision the *Server*. For example, a device vendor may use a CA that is used to issue *Certificates* to *Applications* used by their field technicians.

For embedded devices, the *Server* should allow any *Client* that provides the proper *Administrator* credentials to create the secure connection needed for provisioning using push management. Once the device has been given its initial *Trust List*, the *Server* should then restrict access to those *Clients* with *Certificates* in the *Trust List*. A vendor-specific process for provisioning is required if a device does not allow any *Client* to connect securely for provisioning.

See Clause G.1 for more specific examples of how to provision an application.

7.5 Common Information Model

7.5.1 Overview

The common information model defines types that are used in both the Push and the Pull Model.

7.5.2 TrustListType

This type defines a *FileType* that can be used to access a *Trust List*. The *TrustListType* is formally defined in Table 13.

The *CertificateManager* uses this type to implement the Pull Model.

Servers use this type when implementing the Push Model.

An instance of a *TrustListType* shall restrict access to appropriate users or applications. This may be a *CertificateManager* administrative user that can change the contents of a *Trust List*, it may be an Administrative user that is reading a *Trust List* to deploy to an Application host, or it may be an Application that can only access the Trust List assigned to it.

The *Trust List* file is a UA Binary encoded stream containing an instance of *TrustListDataType* (see 7.5.7).

The *Open Method* shall not support modes other than Read (0x01) and the Write + EraseExisting (0x06).

When a *Client* opens the file for writing the *Server* will not actually update the *Trust List* until the *CloseAndUpdate Method* is called. Simply calling *Close* will discard the updates. The bit masks in *TrustListDataType* structure allow the *Client* to only update part of the *Trust List*.

When the *CloseAndUpdate Method* is called the *Server* will validate all new *Certificates* and *CRLs*. If this validation fails the *Trust List* is not updated and the *Server* returns the appropriate *Certificate* error code (see IEC 62541-4).

Table 13 – TrustListType definition

Attribute	Value				
BrowseName	TrustListType				
Namespace	CORE (see 3.3)				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>FileType</i> defined in IEC 62541-5.					
HasProperty	Variable	LastUpdateTime	UtcTime	PropertyType	Mandatory
HasProperty	Variable	UpdateFrequency	Duration	PropertyType	Optional
HasComponent	Method	OpenWithMasks	Defined in 7.5.3		Optional
HasComponent	Method	CloseAndUpdate	Defined in 7.5.4		Optional
HasComponent	Method	AddCertificate	Defined in 7.5.5		Optional
HasComponent	Method	RemoveCertificate	Defined in 7.5.6		Optional

The *LastUpdateTime* indicates when the *Trust List* was last updated via *Trust List Object Methods*. This can be used to determine if a device has an up to date *Trust List* or to detect unexpected modifications. Out of band changes are not necessarily reported by this value.

The *UpdateFrequency Property* specifies how often the *Trust List* needs to be checked for changes. When the *CertificateManager* specifies this value, all *Clients* that read a copy of the *Trust List* should connect to the *CertificateManager* and check for updates to the *Trust List* within 2 times the *UpdateFrequency*. If the *Trust List Object* is contained within a *ServerConfiguration Object* then this value specifies how frequently the *Server* expects the *Trust List* to be updated.

If auditing is supported, the *CertificateManager* shall generate the *TrustListUpdatedAuditEventType* (see 7.5.18) if the *CloseAndUpdate*, *AddCertificate* or *RemoveCertificate Methods* are called.

7.5.3 OpenWithMasks

The *OpenWithMasks Method* allows a *Client* to read only the portion of the *Trust List*.

This *Method* can only be used to read the *Trust List*.

Signature

```
OpenWithMasks (
    [in]  UInt32 masks
    [out] UInt32 fileHandle
);
```

Argument	Description
masks	The parts of the <i>Trust List</i> that are include in the file to read. The masks are defined in 7.5.8.
fileHandle	The handle of the newly opened file.

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_UserAccessDenied	The current user does not have the rights required.

Table 14 specifies the *AddressSpace* representation for the *OpenWithMasks Method*.

Table 14 – OpenWithMasks Method AddressSpace definition

Attribute	Value				
BrowseName	OpenWithMasks				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

7.5.4 CloseAndUpdate

The *CloseAndUpdate Method* closes the file and applies the changes to the *Trust List*. It can only be called if the file was opened for writing. If the *Close Method* is called any cached data is discarded and the *Trust List* is not changed.

The *Server* shall verify that every *Certificate* in the new *Trust List* is valid in accordance with the mandatory rules defined in IEC 62541-4. If an invalid *Certificate* is found, the *Server* shall return an error and shall not update the *Trust List*. If only part of the *Trust List* is being updated, the *Server* creates a temporary *Trust List* that includes the existing *Trust List* plus any updates and validates the temporary *Trust List*.

If the file cannot be processed this *Method* still closes the file and discards the data before returning an error. This *Method* is required if the *Server* supports updates to the *Trust List*.

The structure uploaded includes a mask (see 7.5.8) that specifies which fields are updated. If a bit is not set, then the associated field is not changed.

Signature

```
CloseAndUpdate (
    [in] UInt32 fileHandle
    [out] Boolean applyChangesRequired
);
```

Argument	Description
fileHandle	The handle of the previously opened file.
applyChangesRequired	A flag indicating whether the <i>ApplyChanges Method</i> (see 7.7.5) shall be called before the new <i>Trust List</i> will be used by the <i>Server</i> .

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_UserAccessDenied	The current user does not have the rights required.
Bad_CertificateInvalid	The <i>Server</i> could not validate all <i>Certificates</i> in the <i>Trust List</i> . The <i>DiagnosticInfo</i> shall specify which <i>Certificate(s)</i> are invalid and the specific error.

Table 15 specifies the *AddressSpace* representation for the *CloseAndUpdate Method*.

Table 15 – CloseAndUpdate Method AddressSpace definition

Attribute	Value				
BrowseName	CloseAndUpdate				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

7.5.5 AddCertificate

The *AddCertificate Method* allows a *Client* to add a single *Certificate* to the *Trust List*. The *Server* shall verify that the *Certificate* is valid according to the rules defined in IEC 62541-4. If an invalid *Certificate* is found the *Server* shall return an error and shall not update the *Trust List*.

This *Method* will return a validation error if the *Certificate* is issued by a CA and the *Certificate* for the issuer is not in the *Trust List*.

This *Method* cannot provide CRLs, so issuer Certificates cannot be added with this *Method*. Instead, CA Certificates and their CRLs shall be managed with the *Write Method* on the containing *TrustList Object*.

This *Method* cannot be called if the containing *TrustList Object* is open.

```

AddCertificate (
    [in] ByteString certificate
    [in] Boolean isTrustedCertificate
);

```

Argument	Description
Certificate	The DER encoded Certificate to add.
isTrustedCertificate	If TRUE the Certificate is added to the Trusted Certificates List. If FALSE the Certificate is added to the Issuer Certificates List.

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_UserAccessDenied	The current user does not have the rights required.
Bad_CertificateInvalid	The certificate to add is invalid.
Bad_InvalidState	The object is opened.

Table 16 specifies the *AddressSpace* representation for the *AddCertificate Method*.

Table 16 – AddCertificate Method AddressSpace definition

Attribute	Value				
BrowseName	AddCertificate				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

7.5.6 RemoveCertificate

The *RemoveCertificate Method* allows a *Client* to remove a single *Certificate* from the *Trust List*. It returns *Bad_InvalidArgument* if the thumbprint does not match a *Certificate* in the *Trust List*.

If the *Certificate* is a *CA Certificate* with associated CRLs then all CRLs are removed as well.

This method cannot be called if the file object is open.

```
RemoveCertificate (
    [in] String thumbprint
    [in] Boolean isTrustedCertificate
);
```

Argument	Description
Thumbprint	The SHA1 hash of the <i>Certificate</i> to remove.
isTrustedCertificate	If TRUE the <i>Certificate</i> is removed from the Trusted Certificates List. If FALSE the <i>Certificate</i> is removed from the Issuer Certificates List.

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_UserAccessDenied	The current user does not have the rights required.
Bad_InvalidArgument	The certificate to remove was not found.
Bad_InvalidState	The object is opened.

Table 17 specifies the *AddressSpace* representation for the *RemoveCertificate Method*.

Table 17 – RemoveCertificate Method AddressSpace definition

Attribute	Value				
BrowseName	RemoveCertificate				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

7.5.7 TrustListDataType

This type defines a *DataType* that stores the *Trust List* of a *Server*. Its values are defined in Table 18.

Table 18 – TrustListDataType definition

Name	Type	Value
TrustListDataType	structure	
specifiedLists	TrustListMasks	A bit mask that indicates which lists contain information. The <i>TrustListMasks</i> enumeration in 7.5.8 defines the allowed values.
trustedCertificates	ByteString[]	The list of <i>Application</i> and <i>CA Certificates</i> which are trusted.
trustedCrls	ByteString[]	The CRLs for the <i>Certificates</i> in the <i>trustedCertificates</i> list.
issuerCertificates	ByteString[]	The list of <i>CA Certificates</i> that are necessary to validate <i>Certificates</i> .
issuerCrls	ByteString[]	The CRLs for the <i>CA Certificates</i> in the <i>issuerCertificates</i> list.

7.5.8 TrustListMasks

This is a *DataType* that defines the values used for the *SpecifiedLists* field in the *TrustListDataType*. Its values are defined in Table 19.

Table 19 – TrustListMasks values

Value	Value
None_0	No fields are provided.
TrustedCertificates_1	The TrustedCertificates are provided.
TrustedCrls_2	The TrustedCrls are provided.
IssuerCertificates_4	The IssuerCertificates are provided.
IssuerCrls_8	The IssuerCrls are provided.
All_15	All fields are provided.

7.5.9 TrustListOutOfDateAlarmType

This *SystemOffNormalAlarmType* is raised by the *Server* when the *UpdateFrequency* elapses and the *Trust List* has not been updated. This alarm automatically returns to normal when the *Trust List* is updated. This type is defined in Table 20.

Table 20 – TrustListOutOfDateAlarmType definition

Attribute	Value				
BrowseName	TrustListOutOfDateAlarmType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>SystemOffNormalAlarmType</i> defined in IEC 62541-9.					
HasProperty	Variable	TrustListId	NodeId	PropertyType	Mandatory
HasProperty	Variable	LastUpdateTime	UtcTime	PropertyType	Mandatory
HasProperty	Variable	UpdateFrequency	Duration	PropertyType	Mandatory

TrustListId Property specifies the *NodeId* of the out of date *Trust List* Object.

LastUpdateTime Property specifies when the *Trust List* was last updated.

UpdateFrequency Property specifies how frequently the *Trust List* needs to be updated.

7.5.10 CertificateGroupType

This type is used for *Objects* that represent *Certificate Groups* in the *AddressSpace*. A *Certificate Group* is a context that contains a *Trust List* and one or more *Certificates* that can be assigned to an *Application*. This type exists to allow an *Application* that has multiple *Trust Lists* and/or *Application Certificates* to express them in its *AddressSpace*. This type is defined in Table 21.

Table 21 – CertificateGroupType definition

Attribute	Value				
BrowseName	CertificateGroupType				
Namespace	CORE (see 3.3)				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>BaseObjectType</i> defined in IEC 62541-5.					
HasComponent	Object	TrustList	-	TrustListType	Mandatory
HasProperty	Variable	CertificateTypes	NodeId[]	PropertyType	Mandatory
HasComponent	Object	CertificateExpired		CertificateExpirationAlarmType	Optional
HasComponent	Object	TrustListOutOfDate		TrustListOutOfDateAlarmType	Optional

The *TrustList Object* is the *Trust List* associated with the *Certificate Group*.

The *CertificateTypes Property* specifies the *NodeIds* of the *CertificateTypes* that may be assigned to *Applications* that belong to the *Certificate Group*. For example, a *Certificate Group* with the *NodeId* of *RsaMinApplicationCertificateType* (see 7.5.15) and the *NodeId* *RsaSha256ApplicationCertificate* (see 7.5.16) specified allows an *Application* to have one *Application Instance Certificates* for each type. Abstract base types may be used in this value and indicate that any subtype is allowed. If this list is empty, then the *Certificate Group* does not allow *Certificates* to be assigned to *Applications* (i.e. the *Certificate Group* exists to allow the associated *Trust List* to be read or updated). All *CertificateTypes* for a given *Certificate Group* shall be subtypes of a single common type which shall be either *ApplicationCertificateType* or *HttpsCertificateType*.

The *CertificateExpired Object* is an *Alarm* which is raised when the *Certificate* associated with the *CertificateGroup* is about to expire. The *CertificateExpirationAlarmType* is defined in IEC 62541-9.

The *TrustListOutOfDate Object* is an *Alarm* which is raised when the *Trust List* has not been updated within the period specified by the *UpdateFrequency* (see 7.5.2). The *TrustListOutOfDateAlarmType* is defined in 7.5.9.

7.5.11 CertificateType

This type is an abstract base type for types that describe the purpose of a *Certificate*. This type is defined in Table 22.

Table 22 – CertificateType definition

Attribute	Value				
BrowseName	CertificateType				
Namespace	CORE (see 3.3)				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>BaseObjectType</i> defined in IEC 62541-5					
HasSubtype	ObjectType	ApplicationCertificateType	Defined in 7.5.12		
HasSubtype	ObjectType	HttpsCertificateType	Defined in 7.5.13		
HasSubtype	ObjectType	UserCredentialCertificateType	Defined in 7.5.14		

7.5.12 ApplicationCertificateType

This type is an abstract base type for types that describe the purpose of an *ApplicationInstanceCertificate*. This type is defined in Table 23.

Table 23 – ApplicationCertificateType definition

Attribute	Value				
BrowseName	ApplicationCertificateType				
Namespace	CORE (see 3.3)				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>CertificateType</i> defined in 7.5.11					
HasSubtype	ObjectType	RsaMinApplicationCertificateType	Defined in 7.5.15		
HasSubtype	ObjectType	RsaSha256ApplicationCertificateType	Defined in 7.5.16		

7.5.13 HtpsCertificateType

This type is used to describe Certificates that are intended for use as HTTPS *Certificates*. This type is defined in Table 24.

Table 24 – HtpsCertificateType definition

Attribute	Value				
BrowseName	HtpsCertificateType				
Namespace	CORE (see 3.3)				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>CertificateType</i> defined in 7.5.11.					

7.5.14 UserCredentialCertificateType

This type is used to describe Certificates that are intended for use as user credentials. This type is defined in Table 25.

Table 25 – UserCredentialCertificateType definition

Attribute	Value				
BrowseName	UserCredentialCertificateType				
Namespace	CORE (see 3.3)				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>CertificateType</i> defined in 7.5.11.					

7.5.15 RsaMinApplicationCertificateType

This type is used to describe *Certificates* intended for use as an *ApplicationInstanceCertificate*. They shall have an RSA key size of 1 024 bits or 2 048 bits. All *Applications* which support the *Basic128Rsa15* and *Basic256* profiles (see IEC 62541-7) shall have a *Certificate* of this type. This type is defined in Table 26.

Table 26 – RsaMinApplicationCertificateType definition

Attribute	Value				
BrowseName	RsaMinApplicationCertificateType				
Namespace	CORE (see 3.3)				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>ApplicationCertificateType</i> defined in 7.5.12.					

7.5.16 RsaSha256ApplicationCertificateType

This type is used to describe *Certificates* intended for use as an *ApplicationInstanceCertificate*. They shall have an RSA key size of 2 048 bits, 3 072 bits or 4 096 bits. All *Applications* that support the *Basic256Sha256* profile (see IEC 62541-7) shall have a *Certificate* of this type. This type is defined in Table 27.

Table 27 – RsaSha256ApplicationCertificateType definition

Attribute	Value				
BrowseName	RsaSha256ApplicationCertificateType				
Namespace	CORE (see 3.3)				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>ApplicationCertificateType</i> defined in 7.5.12					

7.5.17 CertificateGroupFolderType

This type is used for *Folders* that organize *Certificate Groups* in the *AddressSpace*. This type is defined in Table 28.

Table 28 – CertificateGroupFolderType definition

Attribute	Value				
BrowseName	CertificateGroupFolderType				
Namespace	CORE (see 3.3)				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>FolderType</i> defined in IEC 62541-5.					
Organizes	Object	DefaultApplicationGroup		CertificateGroupType	Mandatory
Organizes	Object	DefaultHttpsGroup		CertificateGroupType	Optional
Organizes	Object	DefaultUserTokenGroup		CertificateGroupType	Optional
Organizes	Object	<AdditionalGroup>		CertificateGroupType	Optional Placeholder

The *DefaultApplicationGroup Object* represents the default *Certificate Group* for *Applications*. It is used to access the default *Application Trust List* and to define the *CertificateTypes* allowed for the *ApplicationInstanceCertificate*. This *Object* shall specify the *ApplicationCertificateType NodeId* (see 7.5.12) as a single entry in the *CertificateTypes* list or it shall specify one or more subtypes of *ApplicationCertificateType*.

The *DefaultHttpsGroup Object* represents the default *Certificate Group* for HTTPS communication. It is used to access the default HTTPS *Trust List* and to define the *CertificateTypes* allowed for the *HTTPS Certificate*. This *Object* shall specify the *HttpsCertificateType NodeId* (see 7.5.13) as a single entry in the *CertificateTypes* list or it shall specify one or more subtypes of *HttpsCertificateType*.

This *DefaultUserTokenGroup Object* represents the default *Certificate Group* for validating user credentials. It is used to access the default user credential *Trust List* and to define the *CertificateTypes* allowed for user credentials *Certificate*. This *Object* shall leave *CertificateTypes* list empty.

7.5.18 TrustListUpdatedAuditEventType

This event is raised when a *Trust List* is changed.

This is the result of a *CloseAndUpdate Method* on a *TrustListType Object* being called.

It shall also be raised when the *AddCertificate* or *RemoveCertificate* Method causes an update to the *Trust List*.

Its representation in the *AddressSpace* is formally defined in Table 29.

Table 29 – TrustListUpdatedAuditEventType definition

Attribute	Value				
BrowseName	TrustListUpdatedAuditEventType				
Namespace	CORE (see 3.3)				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditUpdateMethodEventType</i> defined in IEC 62541-5.					

This *EventType* inherits all *Properties* of the *AuditUpdateMethodEventType*. Their semantic is defined in IEC 62541-5.

7.6 Information Model for Pull Certificate Management

7.6.1 Overview

The *GlobalDiscoveryServer AddressSpace* used for *Certificate* management is shown in Figure 14. Most of the interactions between the *GlobalDiscoveryServer* and *Application* administrator or the *Client* will be via *Methods* defined on the *Directory* folder.

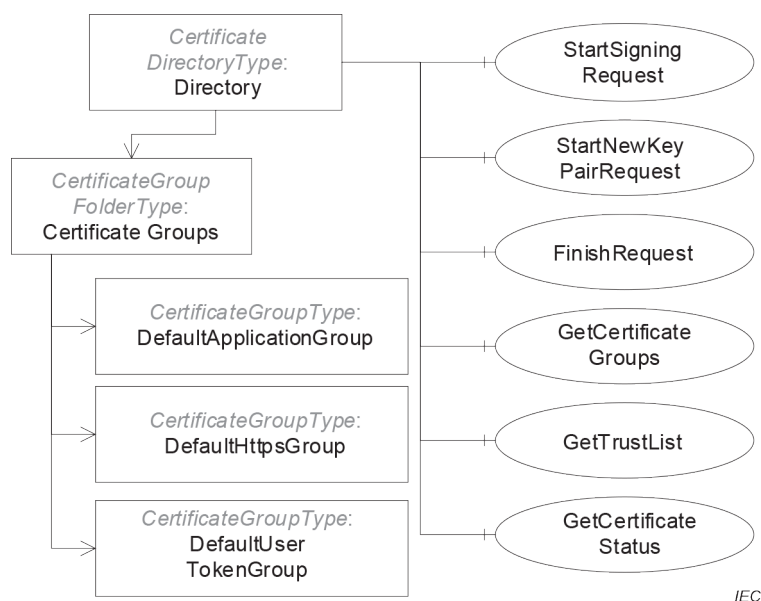


Figure 14 – The Certificate Management AddressSpace for the GlobalDiscoveryServer

7.6.2 CertificateDirectoryType

This *ObjectType* is the *TypeDefinition* for the root of the *CertificateManager AddressSpace*. It provides additional *Methods* for *Certificate* management which are shown in Table 30.