



DPC: 24 / 30484404 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 29 January 2024

Origin: International

Latest date for receipt of comments: 20 March 2024

Project No. 2023/02968

Responsible committee: GEL/65/5 Devices and integration in enterprise systems

Interested committees:

Title: BS EN IEC 62541-12 OPC Unified Architecture

Part 12: Discovery and global services

Please notify the secretary if you are aware of any keywords that might assist in classifying or identifying the standard or if the content of this standard

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

**WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD.
THIS DRAFT IS NOT CURRENT BEYOND 20 March 2024**

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <http://standardsdevelopment.bsigroup.com>

Responsible Committee Secretary: Ms Charmaine Mukherjee (BSI)

Direct tel:

E-mail: charmaine.mukherjee@bsigroup.com

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/Figure/ Table/Note	Type of comment	Commend (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	

PROJECT NUMBER:

IEC 62541-12 ED2

DATE OF CIRCULATION:

2024-01-26

CLOSING DATE FOR VOTING:

2024-04-19

SUPERSEDES DOCUMENTS:

65E/973/RR

IEC SC 65E : DEVICES AND INTEGRATION IN ENTERPRISE SYSTEMS

SECRETARIAT:

United States of America

SECRETARY:

Mr Donald (Bob) Lattimer

OF INTEREST TO THE FOLLOWING COMMITTEES:

PROPOSED HORIZONTAL STANDARD:

☐

Other TC/SCs are requested to indicate their interest, if any, in this CDV to the secretary.

FUNCTIONS CONCERNED:

☐ EMC☐ ENVIRONMENT☐ QUALITY ASSURANCE☐ SAFETY☒ SUBMITTED FOR CENELEC PARALLEL VOTING☐ NOT SUBMITTED FOR CENELEC PARALLEL VOTING**Attention IEC-CENELEC parallel voting**

The attention of IEC National Committees, members of CENELEC, is drawn to the fact that this Committee Draft for Vote (CDV) is submitted for parallel voting.

The CENELEC members are invited to vote through the CENELEC online voting system.

This document is still under study and subject to change. It should not be used for reference purposes.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Recipients of this document are invited to submit, with their comments, notification of any relevant "In Some Countries" clauses to be included should this proposal proceed. Recipients are reminded that the CDV stage is the final stage for submitting ISC clauses. (SEE [AC/22/2007](#) OR [NEW GUIDANCE DOC](#)).

TITLE:

OPC Unified Architecture - Part 12: Discovery and global services

PROPOSED STABILITY DATE: 2026

NOTE FROM TC/SC OFFICERS:

CONTENTS

Page

FIGURES v

TABLES vi

FOREWORD..... ix

1 Scope..... 1

2 Normative references 1

3 Terms, definitions, and conventions 2

 3.1 Terms and definitions 2

 3.2 Abbreviations and symbols 4

4 The Discovery Process 5

 4.1 Overview 5

 4.2 Registration and Announcement of Applications..... 5

 4.2.1 Overview 5

 4.2.2 Hosts with a LocalDiscoveryServer..... 5

 4.2.3 Hosts without a LocalDiscoveryServer 6

 4.3 The Discovery Process for Clients to Find Servers 6

 4.3.1 Overview 6

 4.3.2 Simple Discovery with a DiscoveryUrl 7

 4.3.3 Local Discovery 7

 4.3.4 MulticastSubnet Discovery 8

 4.3.5 Global Discovery 8

 4.3.6 Combined Discovery Process for Clients 9

 4.4 The Discovery Process for Reverse Connections 10

 4.4.1 Overview 10

 4.4.2 Out-of-band Discovery 10

 4.4.3 Global Discovery for Reverse Connections 10

5 Local Discovery Server 11

 5.1 Overview 11

 5.2 Security Considerations for Multicast DNS 11

 5.3 Network Architectures 11

 5.3.1 Overview 11

 5.3.2 Single MulticastSubnet..... 11

 5.3.3 Multiple MulticastSubnet 12

 5.3.4 No MulticastSubnet..... 13

 5.3.5 Domain Names and MulticastSubnets..... 13

6 Global Discovery Server 14

 6.1 Overview 14

 6.2 Roles and Privileges 14

 6.3 Client connections to global services 14

 6.4 Local Discovery..... 15

 6.5 Application Registration Workflow..... 16

 6.6 Information Model 18

 6.6.1 Overview 18

 6.6.2 Directory..... 19

 6.6.3 DirectoryType 19

 6.6.4 FindApplications 19

48	6.6.5	ApplicationRecordDataType	21
49	6.6.6	RegisterApplication	21
50	6.6.7	UpdateApplication	22
51	6.6.8	UnregisterApplication	23
52	6.6.9	GetApplication	23
53	6.6.10	QueryApplications	24
54	6.6.11	QueryServers (deprecated)	25
55	6.6.12	ApplicationRegistrationChangedAuditEventType	27
56	7	Certificate Management	27
57	7.1	Overview	27
58	7.2	Roles and Privileges	28
59	7.3	Pull Management	29
60	7.4	Push Management	30
61	7.5	Application Setup	30
62	7.6	Pull Management Workflow	31
63	7.7	Push Management Workflow	34
64	7.8	Common Information Model	36
65	7.8.1	Overview	36
66	7.8.2	TrustLists	36
67	7.8.3	CertificateGroups	44
68	7.8.4	CertificateTypes	46
69	7.9	Information Model for Pull Certificate Management	50
70	7.9.1	Overview	50
71	7.9.2	CertificateDirectoryType	50
72	7.9.3	StartSigningRequest	51
73	7.9.4	StartNewKeyPairRequest	53
74	7.9.5	FinishRequest	54
75	7.9.6	RevokeCertificate	55
76	7.9.7	GetCertificateGroups	56
77	7.9.8	GetCertificates	56
78	7.9.9	GetTrustList	57
79	7.9.10	GetCertificateStatus	58
80	7.9.11	CheckRevocationStatus	58
81	7.9.12	CertificateRequestedAuditEventType	59
82	7.9.13	CertificateDeliveredAuditEventType	60
83	7.10	Information Model for Push Certificate Management	60
84	7.10.1	Overview	60
85	7.10.2	ServerConfiguration	62
86	7.10.3	ServerConfigurationType	63
87	7.10.4	UpdateCertificate	64
88	7.10.5	GetCertificates	65
89	7.10.6	ApplyChanges	66
90	7.10.7	CreateSigningRequest	66
91	7.10.8	CancelChanges	67
92	7.10.9	GetRejectedList	68
93	7.10.10	ResetToServerDefaults	68
94	7.10.11	TransactionDiagnosticsType	69
95	7.10.12	TransactionErrorType	70
96	7.10.13	CertificateUpdateRequestedAuditEventType	70

97	7.10.14	CertificateUpdatedAuditEventType	70
98	8	KeyCredential Management	71
99	8.1	Overview	71
100	8.2	Roles and Privileges	71
101	8.3	Pull Management	72
102	8.4	Push Management	73
103	8.5	Information Model for Pull Management	73
104	8.5.1	Overview	73
105	8.5.2	KeyCredentialManagementFolderType	74
106	8.5.3	KeyCredentialManagement	74
107	8.5.4	KeyCredentialServiceType	74
108	8.5.5	StartRequest	75
109	8.5.6	FinishRequest	76
110	8.5.7	Revoke	77
111	8.5.8	KeyCredentialAuditEventType	77
112	8.5.9	KeyCredentialRequestedAuditEventType	78
113	8.5.10	KeyCredentialDeliveredAuditEventType	78
114	8.5.11	KeyCredentialRevokedAuditEventType	78
115	8.6	Information Model for Push Management	79
116	8.6.1	KeyCredentialConfigurationFolderType	79
117	8.6.2	CreateCredential	80
118	8.6.3	KeyCredentialConfiguration	80
119	8.6.4	KeyCredentialConfigurationType	80
120	8.6.5	GetEncryptingKey	81
121	8.6.6	UpdateCredential	82
122	8.6.7	DeleteCredential	82
123	8.6.8	KeyCredentialUpdatedAuditEventType	83
124	8.6.9	KeyCredentialDeletedAuditEventType	83
125	9	AuthorizationServices	84
126	9.1	Overview	84
127	9.2	Roles and Privileges	84
128	9.3	Implicit	85
129	9.4	Explicit	86
130	9.5	Chained	86
131	9.6	Information Model for Requesting Access Tokens	87
132	9.6.1	Overview	87
133	9.6.2	AuthorizationServicesFolderType	88
134	9.6.3	AuthorizationServices	88
135	9.6.4	AuthorizationServiceType	88
136	9.6.5	RequestAccessToken	89
137	9.6.6	GetServiceDescription	90
138	9.6.7	AccessTokenIssuedAuditEventType	90
139	9.7	Information Model for Configuring Servers	91
140	9.7.1	Overview	91
141	9.7.2	AuthorizationServiceConfigurationFolderType	91
142	9.7.3	AuthorizationServices	92
143	9.7.4	AuthorizationServiceConfigurationType	92
144	10	Namespaces	92
145	10.1	Namespace Metadata	92

146	10.2	Handling of OPC UA Namespaces	93
147		Annex A (informative) Deployment and Configuration	94
148	A.1	Firewalls and Discovery.....	94
149	A.2	Resolving References to Remote Servers	96
150		Annex B (normative) NodeSet and Constants	97
151	B.1	NodeSet	97
152	B.2	Numeric Node Ids	97
153		Annex C (normative) OPC UA Mapping to mDNS.....	98
154	C.1	DNS Server (SRV) Record Syntax	98
155	C.2	DNS Text (TXT) Record Syntax	98
156	C.3	DiscoveryUrl Mapping	99
157		Annex D (normative) Server Capability Identifiers	100
158		Annex E (normative) DirectoryServices	101
159	E.1	Global Discovery via Other Directory Services	101
160	E.2	UDDI	101
161	E.3	LDAP	102
162		Annex F (normative) Local Discovery Server	104
163	F.1	Certificate Store Directory Layout	104
164	F.2	Installation Directories on Windows	104
165		Annex G (normative) Application Setup	106
166	G.1	Application Setup with Pull Management.....	106
167	G.2	Application Setup with the Push Management.....	106
168	G.3	Setting Permissions	107
169		Annex H (informative) Comparison with RFC 7030.....	108
170	H.1	Overview	108
171	H.2	Obtaining CA Certificates	108
172	H.3	Initial Enrolment.....	108
173	H.4	Client Certificate Reissuance.....	108
174	H.5	Server Key Generation	109
175	H.6	Certificate Signing Request (CSR) Attributes Request.....	109
176			
177			

FIGURES

178		
179	Figure 1 – The Registration Process with an LDS	6
180	Figure 2 – The Simple Discovery Process	7
181	Figure 3 – The Local Discovery Process	8
182	Figure 4 – The MulticastSubnet Discovery Process	8
183	Figure 5 – The Global Discovery Process	9
184	Figure 6 – The Discovery Process for Clients	9
185	Figure 7 – The Global Discovery Process for Reverse Connections	10
186	Figure 8 – The Single MulticastSubnet Architecture	12
187	Figure 9 – The Multiple MulticastSubnet Architecture	12
188	Figure 10 – The No MulticastSubnet Architecture	13
189	Figure 11 – The Relationship Between GDS and other components	15
190	Figure 12 – Application Registration Workflow	17
191	Figure 13 – The Address Space for the GDS	19
192	Figure 14 – The Pull Management Model for Certificates	29
193	Figure 15 – The Push Certificate Management Model	30
194	Figure 16 – Certificate Pull Management Workflow	32
195	Figure 17 – The Pull Management Private Key Options	33
196	Figure 18 – The Certificate Push Management Workflow	35
197	Figure 19 – The Push Management Private Key Options	36
198	Figure 20 – The Certificate Management AddressSpace for the GlobalDiscoveryServer	50
199	Figure 21 – The AddressSpace for the Server that supports Push Management	61
200	Figure 22 – The Transaction Lifecycle when using PushManagement	62
201	Figure 23 – The Pull Model for KeyCredential Management	72
202	Figure 24 – The Push Model for KeyCredential Management	73
203	Figure 25 – The Address Space used for Pull KeyCredential Management	74
204	Figure 26 – The Address Space used for Push KeyCredential Management	79
205	Figure 27 – Roles and AuthorizationServices	84
206	Figure 28 – Implicit Authorization	85
207	Figure 29 – Explicit Authorization	86
208	Figure 30 – Chained Authorization	87
209	Figure 31 – The Model for Requesting Access Tokens from AuthorizationServices	88
210	Figure 32 – The Model for Configuring Servers to use AuthorizationServices	91
211	Figure 33 – Discovering Servers Outside a Firewall	94
212	Figure 34 – Discovering Servers Behind a Firewall	94
213	Figure 35 – Using a Discovery Server with a Firewall	95
214	Figure 36 – Following References to Remote Servers	96
215	Figure 37 – The UDDI or LDAP Discovery Process	101
216	Figure 38 – UDDI Registry Structure	102
217	Figure 39 – Sample LDAP Hierarchy	103
218		

TABLES

220		
221	Table 1 – Well-known Roles for a GDS.....	14
222	Table 2 – Privileges for a GDS.....	14
223	Table 3 – Application Registration Workflow Steps.....	18
224	Table 4 – Directory Object Definition.....	19
225	Table 5 – DirectoryType Definition.....	19
226	Table 6 – FindApplications Method AddressSpace Definition.....	20
227	Table 7 – ApplicationRecordDataType Structure.....	21
228	Table 8 – ApplicationRecordDataType Definition.....	21
229	Table 9 – RegisterApplication Method AddressSpace Definition.....	22
230	Table 10 – UpdateApplication Method AddressSpace Definition.....	23
231	Table 11 – UnregisterApplication Method AddressSpace Definition.....	23
232	Table 12 – GetApplication Method AddressSpace Definition.....	24
233	Table 13 – ApplicationRecordDataType to ApplicationDescription Mapping.....	24
234	Table 14 – QueryApplications Method AddressSpace Definition.....	25
235	Table 15 – ApplicationRecordDataType to ServerOnNetwork Mapping.....	26
236	Table 16 – QueryServers Method AddressSpace Definition.....	27
237	Table 17 – ApplicationRegistrationChangedAuditEventType Definition.....	27
238	Table 18 – Well-known Roles for a CertificateManager.....	28
239	Table 19 – Well-known Roles for Server managed by a CertificateManager.....	28
240	Table 20 – Privileges for a CertificateManager.....	29
241	Table 21 – Certificate Pull Management Workflow Steps.....	33
242	Table 22 – TrustListType Definition.....	37
243	Table 23 – OpenWithMasks Method AddressSpace Definition.....	38
244	Table 24 – CloseAndUpdate Method AddressSpace Definition.....	39
245	Table 25 – AddCertificate Method AddressSpace Definition.....	40
246	Table 26 – RemoveCertificate Method AddressSpace Definition.....	41
247	Table 27 – TrustListDataType Structure.....	41
248	Table 28 – TrustListDataType Definition.....	41
249	Table 29 – TrustListMasks Enumeration.....	41
250	Table 30 – TrustListMasks Definition.....	42
251	Table 31 – TrustListValidationOptions Values.....	42
252	Table 32 – TrustListValidationOptions Definition.....	42
253	Table 33 – TrustListOutOfDateAlarmType definition.....	43
254	Table 34 – TrustListUpdateRequestedAuditEventType Definition.....	43
255	Table 35 – TrustListUpdatedAuditEventType Definition.....	44
256	Table 36 – CertificateGroupType Definition.....	44
257	Table 37 – GetRejectedList Method AddressSpace Definition.....	45
258	Table 38 – CertificateGroupFolderType Definition.....	46
259	Table 39 – CertificateType Definition.....	46
260	Table 40 – ApplicationCertificateType Definition.....	46
261	Table 41 – HttpsCertificateType Definition.....	47
262	Table 42 – RsaMinApplicationCertificateType Definition.....	47

263	Table 43 – RsaSha256ApplicationCertificateType Definition	47
264	Table 44 – EccApplicationCertificateType Definition	48
265	Table 45 – EccNistP256ApplicationCertificateType Definition.....	48
266	Table 46 – EccNistP384ApplicationCertificateType Definition.....	48
267	Table 47 – EccBrainpoolP256r1ApplicationCertificateType Definition	48
268	Table 48 – EccBrainpoolP384r1ApplicationCertificateType Definition	49
269	Table 49 – EccCurve25519ApplicationCertificateType Definition	49
270	Table 50 – EccCurve448ApplicationCertificateType Definition.....	49
271	Table 51 – CertificateDirectoryType ObjectType Definition.....	50
272	Table 52 – StartSigningRequest Method AddressSpace Definition	53
273	Table 53 – StartNewKeyPairRequest Method AddressSpace Definition	54
274	Table 54 – FinishRequest Method AddressSpace Definition	55
275	Table 55 – Revoke Method AddressSpace Definition	56
276	Table 56 – GetCertificateGroups Method AddressSpace Definition	56
277	Table 57 – GetCertificates Method AddressSpace Definition	57
278	Table 58 – GetTrustList Method AddressSpace Definition	58
279	Table 59 – GetCertificateStatus Method AddressSpace Definition	58
280	Table 60 – CheckRevocationStatus Method AddressSpace Definition	59
281	Table 61 – CertificateRequestedAuditEventType Definition	60
282	Table 62 – CertificateDeliveredAuditEventType Definition.....	60
283	Table 63 – ServerConfiguration Object Definition.....	62
284	Table 64 – ServerConfigurationType Definition	63
285	Table 65 – UpdateCertificate Method AddressSpace Definition	65
286	Table 66 – GetCertificates Method AddressSpace Definition	66
287	Table 67 – ApplyChanges Method AddressSpace Definition.....	66
288	Table 68 – CreateSigningRequest Method AddressSpace Definition	67
289	Table 69 – CancelChanges Method AddressSpace Definition	68
290	Table 70 – GetRejectedList Method AddressSpace Definition	68
291	Table 71 – ResetToServerDefaults Method AddressSpace Definition	69
292	Table 72 – TransactionDiagnosticsType Definition.....	69
293	Table 73 – TransactionErrorType Structure	70
294	Table 74 – TransactionErrorType Definition.....	70
295	Table 75 – CertificateUpdateRequestedAuditEventType Definition	70
296	Table 76 – CertificateUpdatedAuditEventType Definition	71
297	Table 77 – Well-known Roles for a KeyCredentialService	71
298	Table 78 – Well-known Roles for Server managed by a KeyCredentialService.....	72
299	Table 79 – Privileges for a KeyCredentialService.....	72
300	Table 80 – KeyCredentialManagementFolderType Definition.....	74
301	Table 81 – KeyCredentialManagement Object Definition.....	74
302	Table 82 – KeyCredentialServiceType Definition	74
303	Table 83 – StartRequest Method AddressSpace Definition.....	76
304	Table 84 – FinishRequest Method AddressSpace Definition.....	77
305	Table 85 – Revoke Method AddressSpace Definition	77

306	Table 86 – KeyCredentialAuditEventType Definition	78
307	Table 87 – KeyCredentialRequestedAuditEventType Definition	78
308	Table 88 – KeyCredentialDeliveredAuditEventType Definition	78
309	Table 89 – KeyCredentialRevokedAuditEventType Definition	79
310	Table 90 – KeyCredentialConfigurationFolderType Definition	79
311	Table 91 – CreateCredential Method AddressSpace Definition	80
312	Table 92 – KeyCredentialConfiguration Object Definition	80
313	Table 93 – KeyCredentialConfigurationType Definition	81
314	Table 94 – GetEncryptingKey Method AddressSpace Definition	82
315	Table 95 – UpdateCredential Method AddressSpace Definition	82
316	Table 96 – DeleteCredential Method AddressSpace Definition	83
317	Table 97 – KeyCredentialUpdatedAuditEventType Definition	83
318	Table 98 – KeyCredentialDeletedAuditEventType Definition	83
319	Table 99 – Well-known Roles for an AuthorizationService	84
320	Table 100 – Privileges for an AuthorizationService	84
321	Table 101 – AuthorizationServicesFolderType Definition	88
322	Table 102 – AuthorizationServices Object Definition	88
323	Table 103 – AuthorizationServiceType Definition	88
324	Table 104 – RequestAccessToken Method AddressSpace Definition	90
325	Table 105 – GetServiceDescription Method AddressSpace Definition	90
326	Table 106 – AccessTokenIssuedAuditEventType Definition	91
327	Table 107 – AuthorizationServicesFolderType Definition	91
328	Table 108 – AuthorizationServices Object Definition	92
329	Table 109 – AuthorizationServiceConfigurationType Definition	92
330	Table 110 – NamespaceMetadata Object for this Document	93
331	Table 111 – Namespaces used in this document	93
332	Table 112 – Allowed mDNS Service Names	98
333	Table 113 – DNS TXT Record String Format	98
334	Table 114 – DiscoveryUrl to DNS SRV and TXT Record Mapping	99
335	Table 115 – Examples of CapabilityIdentifiers	100
336	Table 116 – UDDI tModels	102
337	Table 117 – LDAP Object Class Schema	103
338	Table 118 – Application Certificate Store Directory Layout	104
339	Table 119 – Verifying that a Server is allowed to Provide Certificates	108
340	Table 120 – Verifying that a Client is allowed to request Certificates	108
341		
342		
343		

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.
- This fourth edition cancels and replaces the third edition published in 2020. This edition constitutes a technical revision.
- This edition includes the following significant technical changes with respect to the previous edition:
- a) Added a "Quantity Model" which can be referenced from EngineeringUnit Properties. The model defines quantities and assigned units. In addition it provides alternative units and the conversion to them.
 - b) Added additional rules for ValuePrecision Property:
 - Can also be used for other subtypes like Duration and Decimal.
 - Added rules when ValuePrecision has negative values.

394 The text of this International Standard is based on the following documents:

CDV	Report on voting
65E/XX/CDV	65E/XX/RVC

395
396 Full information on the voting for the approval of this International Standard can be found in the
397 report on voting indicated in the above table.

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

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

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

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

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

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

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

- 415 • reconfirmed,
- 416 • withdrawn,
- 417 • replaced by a revised edition, or
- 418 • amended.

419

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

420

OPC UNIFIED ARCHITECTURE

Part 12: Discovery and Global Services

1 Scope

This part 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 *AuthorizationServices*.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments and errata) applies.

IEC 62541-1, OPC Unified Architecture - Part 1: Overview and Concepts

IEC 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

IEC 62541-17, OPC Unified Architecture - Part 17: Alias Names

IEC 62541-20, OPC Unified Architecture - Part 20: File Transfer

IEC 62541-21, OPC Unified Architecture - Part 21: Device Onboarding

Auto-IP, Dynamic Configuration of IPv4 Link-Local Addresses

<http://www.ietf.org/rfc/rfc3927.txt>

DNS-Name, Domain Names – Implementation and Specification

<http://www.ietf.org/rfc/rfc1035.txt>

DHCP, Dynamic Host Configuration Protocol

<http://www.ietf.org/rfc/rfc2131.txt>

mDNS, Multicast DNS

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

DNS-SD, DNS Based Service Discovery

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

RFC 5958, Asymmetric Key Packages

<http://www.ietf.org/rfc/rfc5958.txt>

PKCS #10, Certification Request Syntax Specification

<http://www.ietf.org/rfc/rfc2986.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>

RFC 7030, Enrollment over Secure Transport

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

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/>

X.500, ISO/IEC 9594-1:2017 – The Directory Part 2: Overview of concepts

<https://www.iso.org/standard/72550.html>

IEEE 802.1AR, IEEE Std 802.1AR-2018 – Secure Device Identity

https://standards.ieee.org/standard/802_1AR-2018.html

3 Terms, definitions, and conventions

3.1 Terms and definitions

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

3.1.1

CertificateManager

a software application that manages the *Certificates* used by *Applications* in an administrative domain.

3.1.2

CertificateGroup

a context used to manage the *TrustList* and *Certificate(s)* associated with *Applications* or *Users*.

3.1.3

CertificateRequest

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

Note 1 to entry: Devices have hardware based mechanisms, such as a TPM, to protect Private Keys.

3.1.4

ClientUrl

a physical address available on a network that allows *Servers* to initiate a reverse connection.

3.1.5**DirectoryService**

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

3.1.6**DiscoveryServer**

an *Application* that maintains a list of *OPC UA Applications* that are available on the network and provides mechanisms other *OPC UA Applications* to obtain this list.

3.1.7**DiscoveryUrl**

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

3.1.8**GlobalDiscoveryServer (GDS)**

a *Server* that provides numerous services related to discovery and security management.

Note 1 to entry: a GDS may also be a *CertificateManager*.

Note 2 to entry: a GDS may also be a *KeyCredentialService*.

Note 3 to entry: a GDS may also be a *AuthorizationService*.

3.1.9**GlobalService**

a *Server* that provides centrally managed capabilities needed for a system.

Note 4 to entry: a *GlobalDiscoveryServer*, a *CertificateManager*, a *KeyCredentialService* and an *AuthorizationService* are all examples of *GlobalServices*.

3.1.10**IPAddress**

a 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 may change over time.

3.1.11**KeyCredential**

a unique identifier and a secret used to access an *AuthorizationService* or a *Broker*.

Note 1 to entry: a user name and password is an example of a *KeyCredential*.

3.1.12**KeyCredentialService**

a software application that provides *KeyCredentials* needed to access an *AuthorizationService* or a *Broker*.

3.1.13**LocalDiscoveryServer (LDS)**

a *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.

3.1.14**LocalDiscoveryServer-ME (LDS-ME)**

a *LocalDiscoveryServer* that includes the *MulticastExtension*.

3.1.15**MulticastExtension**

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

3.1.16**MulticastSubnet**

a 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..

3.1.17**Privilege**

a named set of rights which cannot be expressed as *Permissions* granted on *Nodes*.

Note 1 to entry: For example, a *Privilege* can be defined when the right to call a *Method* depends on the parameters passed to the *Method*.

Note 5 to entry: a *Privilege* is a document convention that does not appear in the *Server AddressSpace*.

3.1.18**PullManagement**

a workflow where a *Client* manages its configuration by using a *GlobalService*.

3.1.19**PushManagement**

a workflow where a *GlobalService* manages a *Server's* configuration.

3.1.20**ServerCapabilityIdentifier**

a short identifier which uniquely identifies a set of discoverable capabilities supported by an *OPC UA Application*.

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

3.2 Abbreviations and symbols

API	Application Programming Interface
CA	Certificate Authority
CRL	Certificate Revocation List
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EST	Enrolment over Secure Transport
GDS	Global Discovery Server
IANA	The Internet Assigned Numbers Authority
LDAP	Lightweight Directory Access Protocol
LDS	Local Discovery Server
LDS-ME	Local Discovery Server with the 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 Socket Layer
TLS	Transport Layer Security
TPM	Trusted Platform Module
UA	Unified Architecture
UDDI	Universal Description, Discovery and Integration

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.

4.2 Registration and Announcement of Applications

4.2.1 Overview

The clause 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 4.3.3) and *MulticastSubnet* discovery if the LDS is a LDS-ME (see 4.3.4).

The OPC UA Standard (IEC 62541-4) defines a *RegisterServer2 Service* which provides additional registration information. All *Applications* and *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*. *CapabilityIdentifiers* are short, string identifiers of well-known OPC UA features. *Applications* can use these identifiers as a filter during discovery.

The set of known *CapabilityIdentifiers* is specified in Annex D and is limited to features which 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.

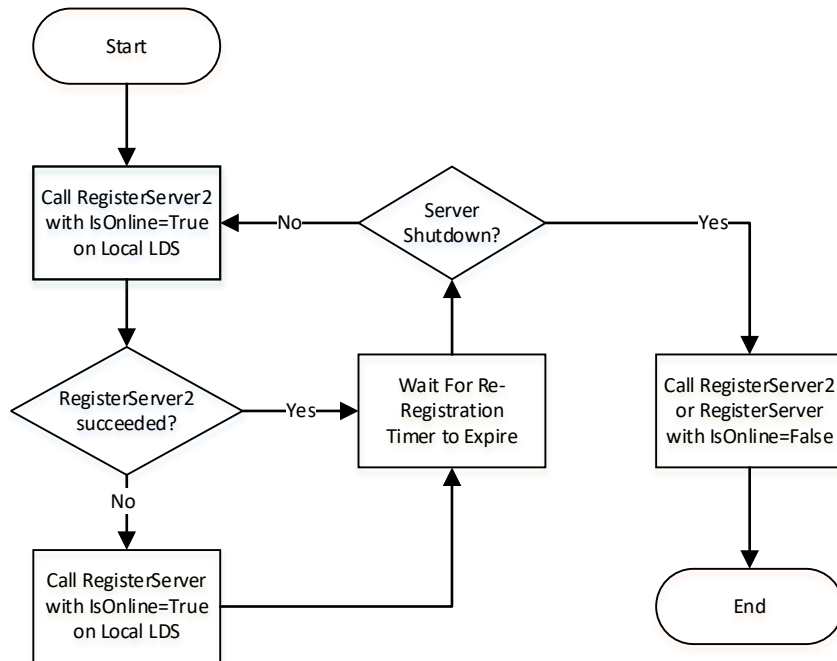


Figure 1 – The Registration Process with an LDS

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 4.3.2).

Clients should be aware of rogue *DiscoveryServers* that might direct them to rogue *Servers*. That said, this problem is mitigated when a *Client* connects to a *Server* and verifies that it trusts the *Server*. In addition, the *CreateSession Service* returns parameters that allow *Client* to verify that the previously acquired results from a LDS have not been altered. See IEC 62541-2 and IEC 62541-4 for a detailed discussion of these issues.

A similar potential for a rogue GDS exists if the *Client* has not been configured to trust the GDS *Certificate* or if the *Client* does not use security when connecting to the GDS. Note that a *Client* that uses security but automatically trusts a GDS *Certificate* is not protected from a rogue GDS even though the connection itself is secure. This problem is also mitigated by verifying trust whenever a *Client* connects to a *Server* discovered via the GDS.

4.3.2 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.

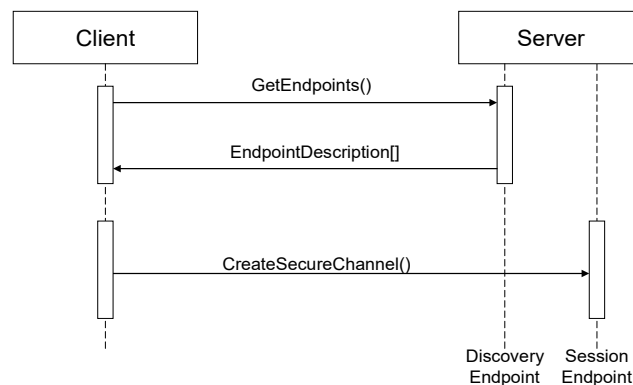


Figure 2 – The Simple Discovery Process

4.3.3 Local Discovery

In many cases *Clients* do not know which *Servers* exist but possibly know which hosts might 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.

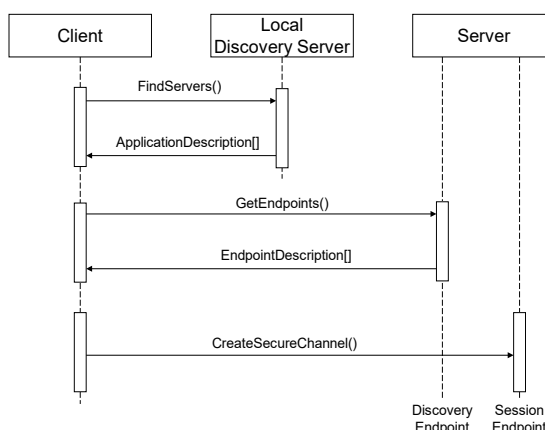


Figure 3 – The Local Discovery Process

4.3.4 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.

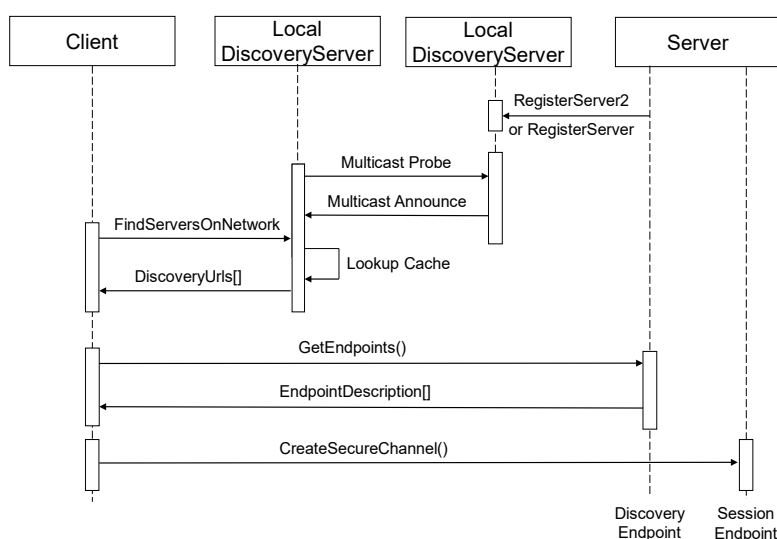


Figure 4 – The MulticastSubnet Discovery Process

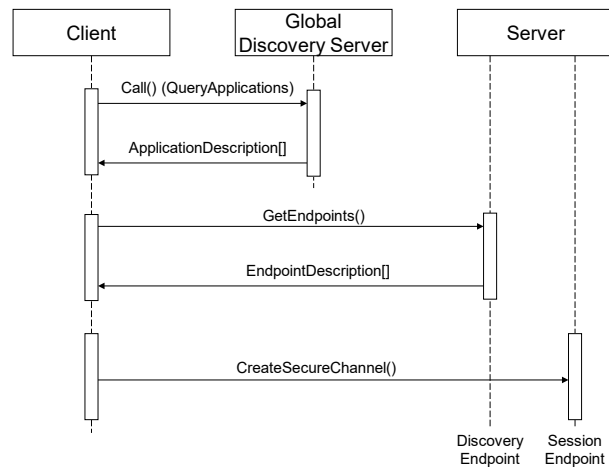
In this scenario the *Server* uses the *RegisterServer2* Service to tell a *LocalDiscoveryServer* to announce the *Server* on the *MulticastSubnet*. The *Client* will receive the *DiscoveryUrl* and *CapabilityIdentifiers* for the *Server* when it calls *FindServersOnNetwork* and then connects directly 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 which 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.

4.3.5 Global Discovery

A GDS is an OPC UA *Server* which allows *Clients* to search for *Servers* within the administrative domain of the GDS. It provides *Methods* that allow applications to search for other applications (See 6). To access the GDS, the *Client* uses the *Call* service to invoke the *QueryApplications* Method (see 6.6.11) to retrieve a list of *Servers* that meet the filter criteria provided.

729 *QueryApplications Method* is similar to the *FindServers* service except that it provides more
 730 advanced search and filter criteria. The discovery process is illustrated in Figure 5.



731
732 **Figure 5 – The Global Discovery Process**

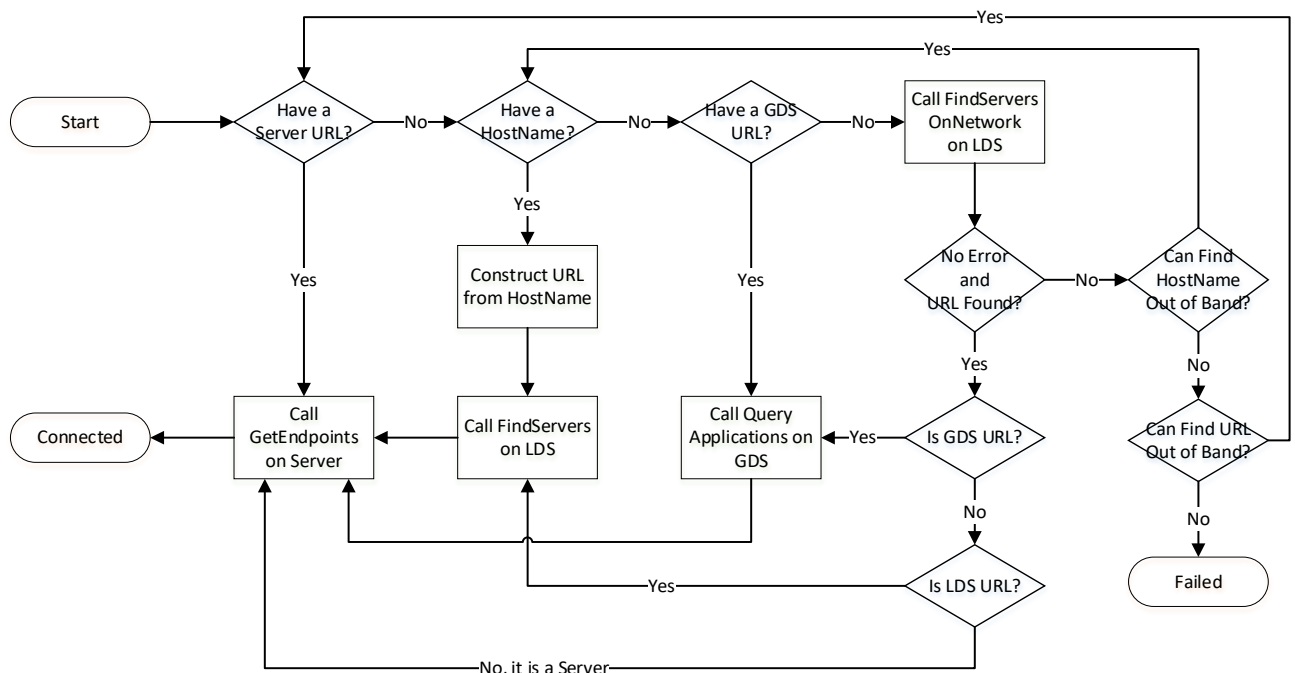
733 The GDS may be coupled with any of the previous network architectures. For each
 734 *MulticastSubnet*, one or more LDSs may be registered with a GDS.

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

736 The complete discovery process is shown in Figure 6.

737 4.3.6 Combined Discovery Process for Clients

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



741
742 **Figure 6 – The Discovery Process for Clients**

743 *FindServersOnNetwork* can be called on the local LDS, however, It can also be called on a
 744 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 standard. 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.

4.4 The Discovery Process for Reverse Connections

4.4.1 Overview

The discovery process for reverse connect does not serve the same purpose as the discovery process for normal connections because reverse connections require the *Server* to be configured to automatically attempt to connect to the *Client* and the *Client* to be configured so it knows what to do with the *Server* when it receives the connection. The limited mechanisms discussed here may help *SecurityAdmins* with the configuration of *Servers*.

A *SecurityAdmin* tasked with configuring *Servers* needs to determine the *ClientUrls* for *Clients* that support reverse connect.

The following choices are available:

- Out-of-band discovery (i.e. entry into a GUI) of a *ClientUrl* for a *Client*;
- Searching for *Clients* known to a *GlobalDiscoveryServer*.

The mechanisms based on an LDS are not available since *Clients* do not register with the LDS.

4.4.2 Out-of-band Discovery

Every *Client* that supports reverse connect has one or more *ClientUrls* that allow *Servers* to connect. Once the *SecurityAdmin* acquires the *ClientUrl* via an out-of-band mechanism, it can configure the *Server* to use it.

4.4.3 Global Discovery for Reverse Connections

A GDS is a *Server* which allows other *SecurityAdmins* to search for *Clients* that support reverse connect within the administrative domain of the GDS. The *SecurityAdmin* uses the *Call* service to invoke the *QueryApplications Method* (see 6.6.11) with “RCP” as a *serverCapabilityFilter* to get a list of *Clients* that support reverse connect from the GDS.

The discovery process is illustrated in Figure 5.

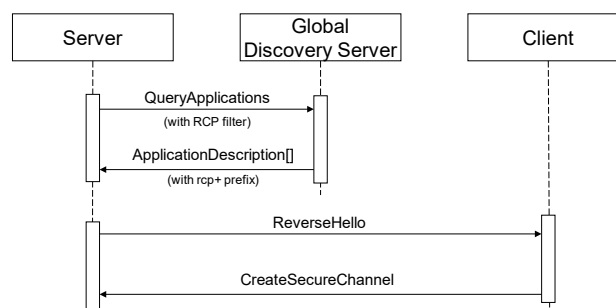


Figure 7 – The Global Discovery Process for Reverse Connections

The *ClientUrls* are returned in the *DiscoveryUrls* parameter of the *ApplicationDescription* record and have the ‘rcp+’ prefix. *DiscoveryUrls* without the prefix are used for forward connections. Once the *SecurityAdmin* has a *ClientUrl* it can configure the *Server* to use it.

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. In addition, the *LocalDiscoveryServer* shall provide at least one *Endpoint* which implements the *RegisterServer* service for these applications.

The *FindServers Service* returns the information for the *LocalDiscoveryServer* and all *Servers* that are known to the LDS. The *GetEndpoints Service* returns the *EndpointDescriptions* for the *LocalDiscoveryServer* that allow *Servers* to call the *RegisterServer* or *RegisterServer2 Services*. The *LocalDiscoveryServer* does not support *Sessions* so information needed for establishing *Sessions*, such as supported *UserTokenPolicies*, is not provided.

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.

5.3 Network Architectures

5.3.1 Overview

The discovery mechanisms defined in this standard 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*

5.3.2 Single MulticastSubnet

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

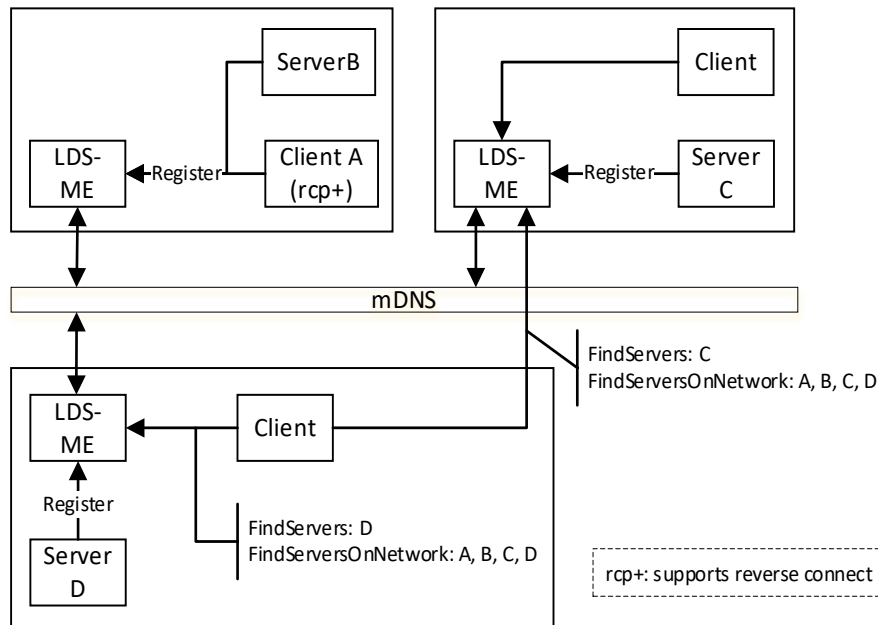


Figure 8 – The Single MulticastSubnet Architecture

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.

5.3.3 Multiple MulticastSubnet

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

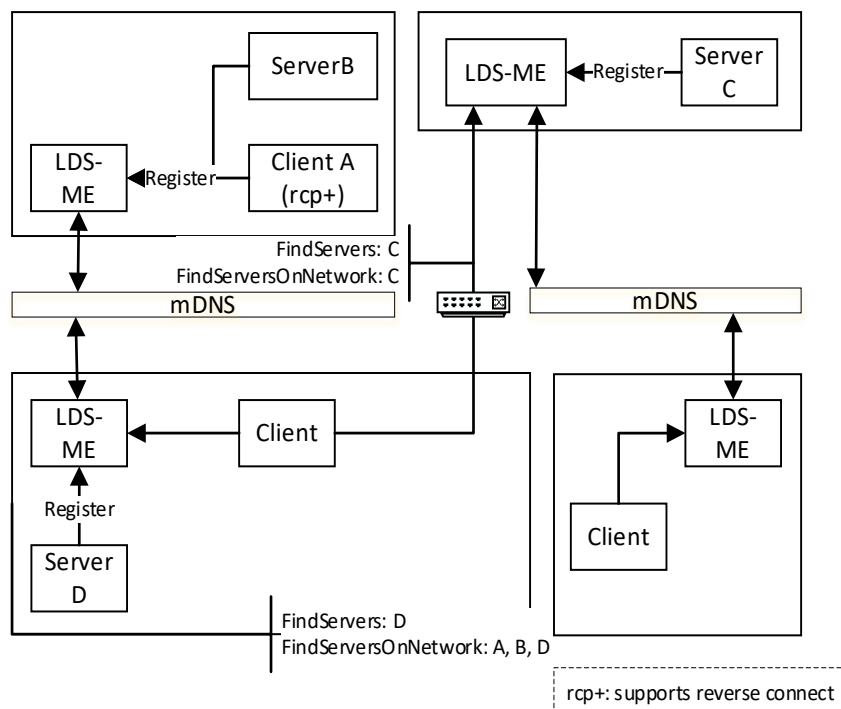


Figure 9 – The Multiple MulticastSubnet Architecture

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*.

5.3.4 No MulticastSubnet

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

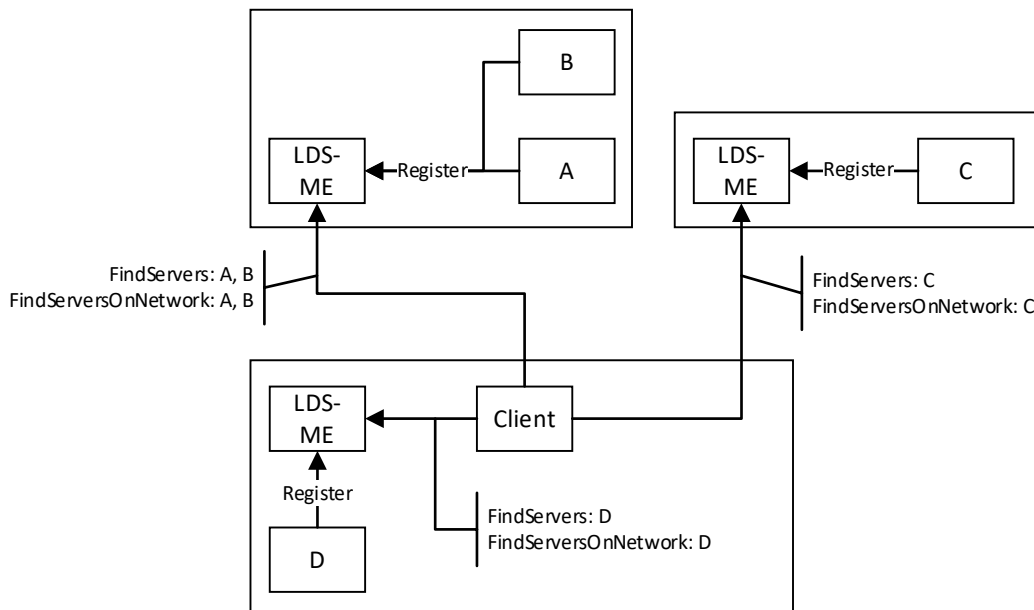


Figure 10 – The No MulticastSubnet Architecture

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* may also work but it will never return more than what *FindServers* returns. *Clients* could also use a GDS if one is available.

5.3.5 Domain Names and MulticastSubnets

The mDNS specification requires that 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* computer could contain 'local' domains which are not reachable or will connect to a different computer 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 computer.

System administrators can eliminate this problem by configuring a normal DNS with the fully qualified domain names for all computers which 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 append the 'local' top level domain to any domains declared in their *Certificate*; an unqualified domain name is used if a more appropriate qualifier does not exist. *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 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* which allows *Clients* to search for *Servers* within the administrative domain. When compared to the LDS, the GDS provides an authoritative source for *Servers* which have been verified by administrators and accessed via a secure communication channel.

The GDS provides *Methods* that allow administrators to register applications and allow applications to search for other applications.

Some GDS implementations may provide a front-end to an existing *DirectoryService* such as LDAP (See Annex E). By standardizing on an OPC UA based interface, *Clients* do not need to have knowledge of different *DirectoryServices*.

6.2 Roles and Privileges

GlobalDiscoveryServers restrict access to many of the features they provide. These restrictions are described either by referring to well-known *Roles* which a *Session* must have access to or by referring to *Privileges* which are assigned to *Sessions* using mechanisms other than the well-known *Roles*. The well-known *Roles* used in for a GDS are listed in Table 1.

Table 1 – Well-known Roles for a GDS

Name	Description
DiscoveryAdmin	This <i>Role</i> grants rights to register, update and unregister any <i>OPC UA Application</i> .
SecurityAdmin	This <i>Role</i> grants the right to change the security configuration of a GDS.

The *Privileges* used in for a GDS are listed in Table 2.

Table 2 – Privileges for a GDS

Name	Description
ApplicationSelfAdmin	This <i>Privilege</i> grants an <i>OPC UA Application</i> the right to update its own registration. The <i>Certificate</i> used to create the <i>SecureChannel</i> is used to determine the identity of the <i>OPC UA Application</i> .
ApplicationAdmin	This <i>Privilege</i> grants rights to update one or more registrations. The <i>Certificate</i> used to create the <i>SecureChannel</i> is used to determine the identity of the <i>OPC UA Application</i> and what the set of registrations it is authorized to update.

6.3 Client connections to global services

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

The number of OPC UA *Applications* using the different services as OPC UA *Client* may be huge and the OPC UA *Server* is most likely not able to handle connections from all OPC UA *Clients* at the same time. Therefore an OPC UA *Client* connected to a GDS should minimize the time it is connected to the GDS to the currently required actions. The OPC UA *Client* shall

disconnect as soon as it completes the sequence of actions needed to interact with the services. The OPC UA *Clients* shall not keep connections open between the execution of sequences.

A GDS OPC UA *Server* is allowed to close *Sessions* with OPC UA *Clients* not authenticated as one of the GDS administrative *Roles* if it runs out of connection resources. If the GDS needs to close *Sessions*, it should first close *Sessions* without GDS management *Privileges*. Otherwise it may close the *Session* that was inactive for the longest time not using GDS global services e.g. *Method* calls.

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

Actions performed cyclically by OPC UA *Applications* during *PullManagement* shall start the second cycle with a random delay that is between one and at least ten percent of the cycle period.

6.4 Local Discovery

If an administrator (e.g. a *Client* with access to the *DirectoryAdmin Role*) registers a *LocalDiscoveryServer* with the GDS, then the GDS periodically adds *Servers* to a list for review by calling *FindServersOnNetwork* or *FindServers* on the LDS. Figure 11 shows the relationship between a GDS and the LDS-ME or LDS.

The GDS shall not make *Servers* discovered in this manner available via *QueryApplications* for *FindApplications* before an administrator has approved the *Server* on a case by case basis or via automated rules. Note that auto-population can result in conflicts where multiple *Servers* have the same *ApplicationUri* due to a configuration error. A GDS should keep track of these conflicts so an administrator can review and resolve them.

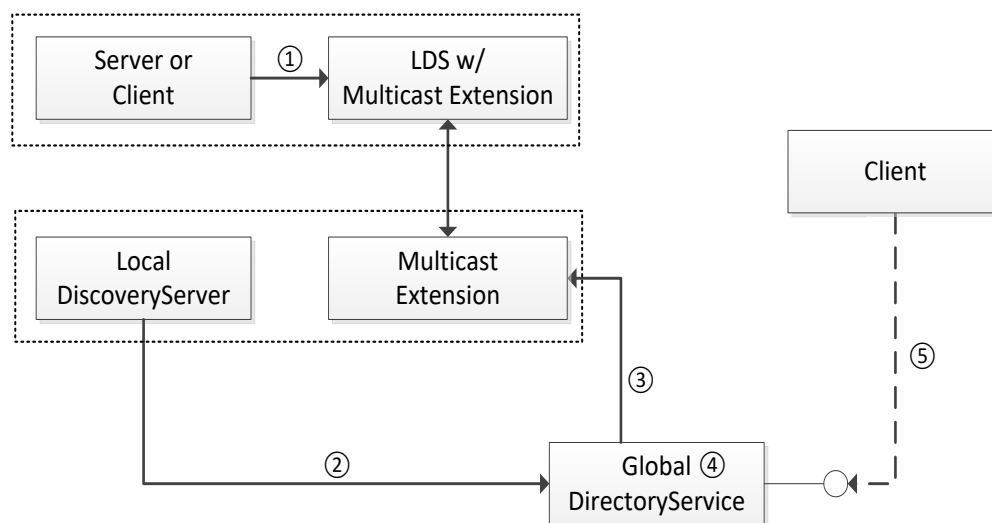


Figure 11 – The Relationship Between GDS and other components

The steps shown in Figure 11 are:

1	The Server calls <i>RegisterServer2</i> on the LDS running on the same computer.
2	The administrator registers LDS-ME installations with the GDS.
3	The GDS calls <i>FindServersOnNetwork</i> on the LDS-ME to find all applications on the same <i>MulticastSubnet</i> .

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 <i>Clients</i> of the GDS. This approval can be obtained from an <i>DomainAdmin</i> .
5	The <i>Client</i> calls <i>QueryApplications Method</i> on the GDS to discover applications.

932 The *Information Model* used for registration and discovery is shown in 6.6.1. Any *Client* shall
933 be able to call the *QueryApplications Method* to find applications known to GDS. The complete
934 definitions for each of the types used are described in 7.8.

935 Once a *Server* is registered with the GDS the record does not dissappear if the *Server* goes
936 offline. If a *Server* is permentently taken offline the administrator needs to manually remove the
937 registration. The interactions described above apply to *Servers* automatically discovered via an
938 LDS. *Servers* can also be discovered by another application and registered automatically with
939 the GDS provided the other application has the necessary administrative rights on the GDS.

940 **6.5 Application Registration Workflow**

941 The OPC UA *Application* or the *Application* configuration tool connects to the GDS for initial
942 installation with GDS including *Application* registration. This requires a user that has the
943 *DiscoveryAdmin Role* or the *ApplicationAdmin Privilege*.

944 The workflow for the *Application* registration is shown in Figure 12.

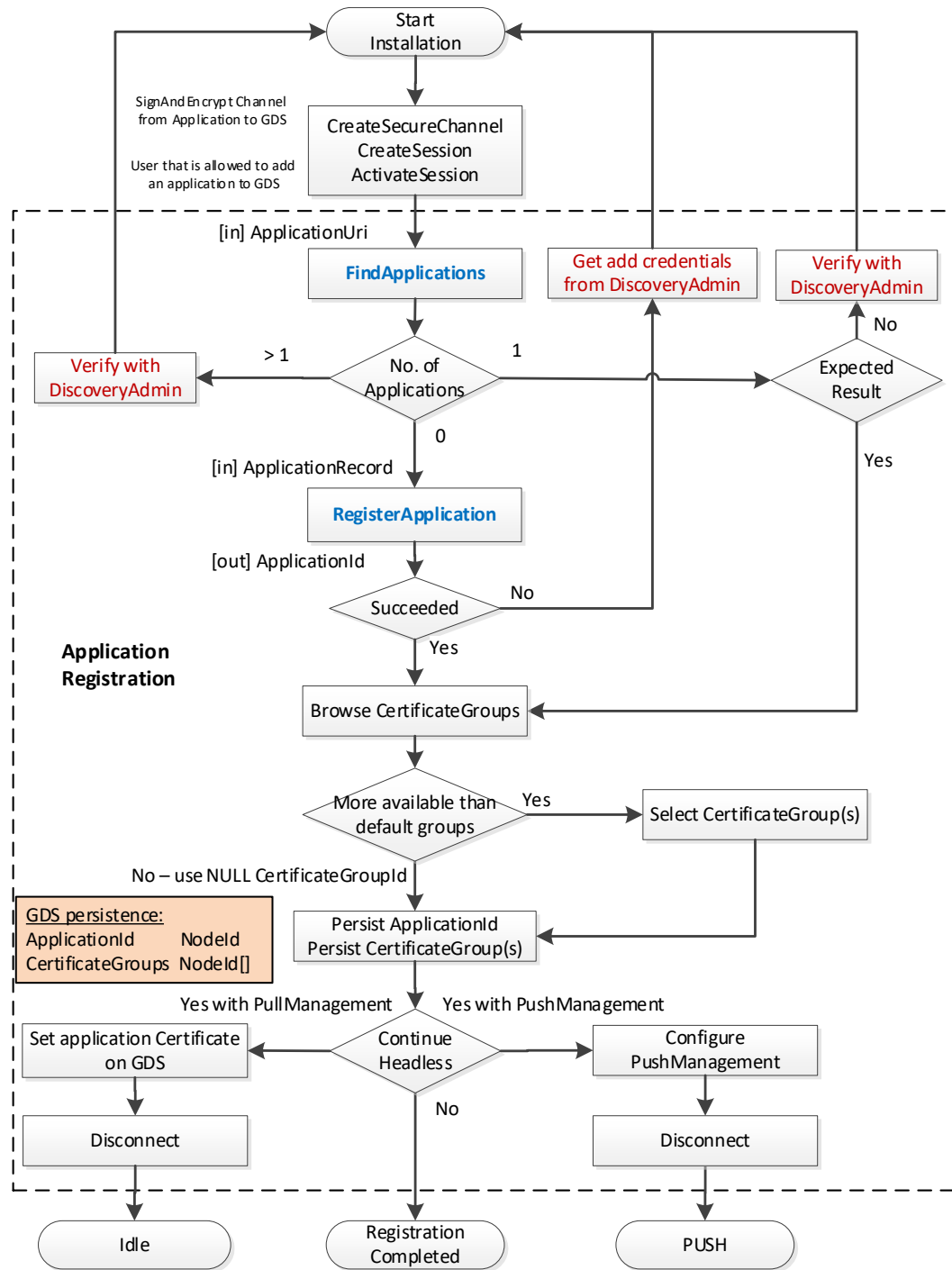


Figure 12 – Application Registration Workflow

The description of the *Application* registration workflow steps is provided in Table 3.

948

Table 3 – Application Registration Workflow Steps

Step	Description
Application installation	The registration of an application with a GDS is normally executed as part of the initial installation and configuration of the application. It can be executed by a configuration tool that is part of the application or by a generic GDS configuration tool.
Connect	For the connection management with the GDS the services <i>OpenSecureChannel</i> , <i>CreateSession</i> and <i>ActivateSession</i> are used to create a connection with <i>MessageSecurityMode SignAndEncrypt</i> and a user that has the permission to register applications with the GDS. If the user does not have sufficient rights, the GDS can provide a mechanism to accept registrations on the GDS side before they are visible to <i>Clients</i> through <i>QueryApplications</i> .
FindApplications	The first step after connect is to check if there is already a registration available for the <i>ApplicationUri</i> . The <i>DirectoryType Method FindApplications</i> is used to pass the <i>ApplicationUri</i> of the application to the GDS. The Method returns an array of application records where the size of the array defines the next steps. • If the array is empty, the next step is <i>RegisterApplication</i>. • If the array size is one, and the record matches the expected application record, the next step is <i>Browse CertificateGroups</i>. • If the array size is one and the record does not match the expected application record, the registration must be verified with a <i>DiscoveryAdmin</i>. • If the array size is more than one, this indicates a fatal error and the status must be verified with a <i>DiscoveryAdmin</i>.
RegisterApplication	The <i>DirectoryType Method RegisterApplication</i> is used to pass in an application record with the application information. If the <i>Method</i> succeeds an <i>ApplicationId</i> is returned. This <i>ApplicationId</i> should be persisted for further interaction with the GDS regarding this application. If the <i>Method</i> fails, a <i>DiscoveryAdmin</i> is needed to identify and correct the issue. Typical errors include insufficient rights or conflicts with other application records.
Browse CertificateGroups	The <i>Browse Service</i> is used to get the list of GDS managed <i>CertificateGroups</i> by browsing the <i>CertificateGroups Folder</i> of the <i>Directory Object</i> . If more than one <i>CertificateGroup</i> is returned, the user selects the relevant <i>CertificateGroups</i> needed for the application. The selected <i>CertificateGroupIds</i> should be persisted together with the <i>ApplicationId</i> .
Registration end options	The following options are possible to complete the registration with the <i>CertificateManager</i> : 1. Continue with <i>PullManagement</i> using the existing connection to the GDS. This option is typically used by <i>Clients</i> executing the registration in an interactive mode for their own identity. See 7.6 for the <i>PullManagement</i> workflow. 2. Continue with <i>PullManagement</i> inside a headless application. 3. Continue with <i>PushManagement</i>.
Set application <i>Certificate</i> on GDS	For option (2) the current application <i>Certificate</i> must be configured for the application on the GDS to allow <i>Application</i> authentication for the initial <i>PullManagement</i> sequence. This configuration in the GDS is currently not in the scope of this specification.
Configure <i>PushManagement</i>	For option (3) the application must be configured for <i>PushManagement</i> in the <i>CertificateManager</i> . The configuration of the <i>PushManagement</i> in the <i>CertificateManager</i> is currently not in the scope of this specification.
Disconnect	For options (2) and (3) the configuration tool disconnects from the GDS.

949

950 6.6 Information Model

951 6.6.1 Overview

952 The *GlobalDiscoveryServer Information Model* used for *discovery* is shown in Figure 13. Most
 953 of the interactions between the *GlobalDiscoveryServer* and *Application* administrator or the
 954 *Client* will be via *Methods* defined on the *Directory* folder.

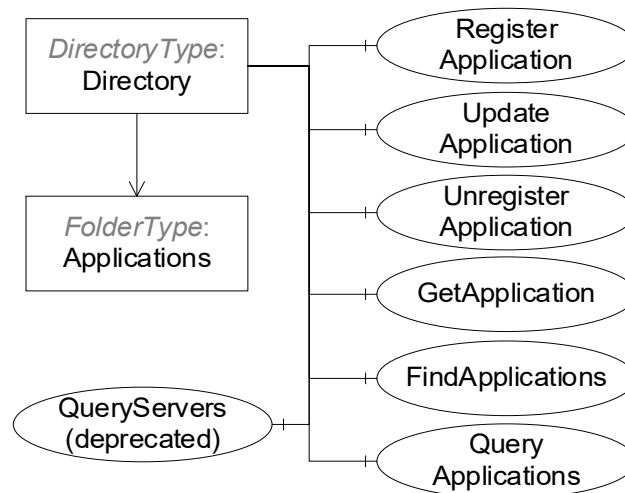


Figure 13 – The Address Space for the GDS

6.6.2 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 4.

Table 4 – Directory Object Definition

Attribute	Value				
BrowseName	2:Directory				
TypeDefinition	2:DirectoryType defined in 6.6.3.				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Conformance Units					
GDS Application Directory					

6.6.3 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 5.

Table 5 – DirectoryType Definition

Attribute	Value				
BrowseName	2:DirectoryType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the 0:FolderType defined in IEC 62541-5.					
0:HasComponent	Object	2:Applications	-	0:FolderType	Mandatory
0:HasComponent	Method	2:FindApplications	Defined in 6.6.4.		Mandatory
0:HasComponent	Method	2:RegisterApplication	Defined in 6.6.6.		Mandatory
0:HasComponent	Method	2:UpdateApplication	Defined in 6.6.7.		Mandatory
0:HasComponent	Method	2:UnregisterApplication	Defined in 6.6.8.		Mandatory
0:HasComponent	Method	2:GetApplication	Defined in 6.6.9.		Mandatory
0:HasComponent	Method	2:QueryApplications	Defined in 6.6.10.		Mandatory
0:HasComponent	Method	2:QueryServers	Defined in 6.6.11.		Mandatory
Conformance Units					
GDS Application Directory					

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*.

6.6.4 FindApplications

FindApplications is used to find the *ApplicationId* for an approved OPC UA *Application* (see 6.6.6 or 6.4). This list of records returned shall have zero or one element.

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 6.6.5.

Method Result Codes (defined in Call Service)

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

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

Table 6 – FindApplications Method AddressSpace Definition

Attribute	Value				
BrowseName	2:FindApplications				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	Mandatory
0:HasProperty	Variable	0:OutputArguments	0:Argument[]	0:PropertyType	Mandatory

6.6.5 ApplicationRecordDataType

This type defines a *DataType* which represents a record in the GDS.

If the *ApplicationType* is *Client* the *serverCapabilities* shall include RCP (reverse connect) and all *DiscoveryUrls* shall begin with the rcp+ prefix which indicates that reverse connections are supported.

If the *ApplicationType* is *ClientAndServer* the *serverCapabilities* may include RCP and all *DiscoveryUrls* that support reverse connect have the rcp+ prefix. If the same URL supports normal connections and reverse connection then there shall be two elements in the *DiscoveryUrls* array with and without the rcp+ prefix.

Table 7 – ApplicationRecordDataType Structure

Name	Type	Description
ApplicationRecordDataType	Structure	Subtype of the <i>Structure DataType</i> defined in IEC 62541-5
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 HTTPS URL specifies the domain used as the Common Name of HTTPS <i>Certificates</i> .
serverCapabilities	String[]	The list of server capability identifiers for the application. The allowed values are defined in Annex D.

Its representation in the *AddressSpace* is defined in Table 8.

Table 8 – ApplicationRecordDataType Definition

Attribute		Value				
BrowseName		2:ApplicationRecordDataType				
IsAbstract		False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Other	
Subtype of the 0:Structure <i>DataType</i> defined in IEC 62541-5.						
Conformance Units						
GDS Application Directory						

6.6.6 RegisterApplication

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

This *Method* shall be called from an authenticated *SecureChannel* and from a *Client* that has access to the *DiscoveryAdmin Role* or the *ApplicationAdmin Privilege* (see 6.2).

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

Servers that support non-transparent redundancy shall register as different applications. In addition, IEC 62541-4 requires the use of the NTRS *ServerCapability* defined in Annex D.

RegisterApplication shall not create duplicate records if the *ApplicationUri* already exists.

If *RegisterApplication* succeeds the *OPC UA Application* is approved and is returned by *QueryApplications* and *FindApplications*.

If registration was successful and auditing is supported, the GDS shall generate the *ApplicationRegistrationChanged AuditEventType* (see 6.6.12).

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 9 specifies the *AddressSpace* representation for the *RegisterApplication Method*.

Table 9 – RegisterApplication Method AddressSpace Definition

Attribute	Value				
BrowseName	2:RegisterApplication				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	Mandatory
0:HasProperty	Variable	0:OutputArguments	0:Argument[]	0:PropertyType	Mandatory

6.6.7 UpdateApplication

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

This *Method* shall be called from an authenticated *SecureChannel* and from a *Client* that has access to the *DiscoveryAdmin Role*, the *ApplicationSelfAdmin Privilege*, or the *ApplicationAdmin Privilege* (see 6.2).

If the update was successful and auditing is supported, the GDS shall generate the *ApplicationRegistrationChanged AuditEventType* (see 6.6.12).

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 10 specifies the *AddressSpace* representation for the *UpdateApplication Method*.

Table 10 – UpdateApplication Method AddressSpace Definition

Attribute	Value				
BrowseName	2:UpdateApplication				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	Mandatory

6.6.8 UnregisterApplication

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

This *Method* shall be called from an authenticated *SecureChannel* and from a *Client* that has access to the *DiscoveryAdmin Role*, the *ApplicationSelfAdmin Privilege*, or the *ApplicationAdmin Privilege* (see 6.2).

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 an *Application* has *Certificates* issued by the *CertificateManager*, these *Certificates* shall be revoked when this *Method* is called.

If un-registration was successful and auditing is supported, the GDS shall generate the *ApplicationRegistrationChanged AuditEventType* (see 6.6.12).

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 needed to unregister the application.

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

Table 11 – UnregisterApplication Method AddressSpace Definition

Attribute	Value				
BrowseName	2:UnregisterApplication				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	Mandatory

6.6.9 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 6.6.5