

	<p>Systeme der Gebäudeautomation Teil 5: Datenkommunikationsprotokoll (ISO/DIS 16484-5:2002) Englische Fassung prEN ISO 16484-5:2002</p>	<p>DIN EN ISO 16484-5</p>
<p>ICS 35.240.99; 97.120</p>	<div data-bbox="603 302 1066 398"><p>Entwurf</p></div> <p data-bbox="212 492 1098 553">Building automation and control systems — Part 5: Data communication protocol (ISO/DIS 16484-5:2002); English version prEN ISO 16484-5:2002</p> <p data-bbox="212 584 1098 676">Systèmes d'automatisation et de commande du bâtiment — Partie 5: Protocole de communication de données (ISO/DIS 16484-5:2002); Version anglaise prEN ISO 16484-5:2002</p> <p data-bbox="212 813 616 846">Anwendungswarnvermerk</p> <p data-bbox="212 878 1236 907">Dieser Norm-Entwurf wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.</p> <p data-bbox="212 938 1433 996">Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfes besonders zu vereinbaren.</p> <p data-bbox="212 1028 600 1057">Stellungnahmen werden erbeten</p> <ul data-bbox="212 1075 1485 1211" style="list-style-type: none">— vorzugsweise als Datei per E-Mail an nhrs@din.de in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter http://www.din.de/stellungnahme abgerufen werden;— oder in Papierform an den Normenausschuss Heiz- und Raumluftechnik (NHRs) im DIN Deutsches Institut für Normung e. V., 10772 Berlin (Hausanschrift: Burggrafenstraße 6, 10787 Berlin). <p data-bbox="212 1308 507 1341">Nationales Vorwort</p> <p data-bbox="212 1373 1485 1496">Diese Europäische Norm EN ISO 16484-5 übernimmt unverändert den englischen Originaltext des ANSI/ASHRAE Standard 135 „BACnet – A data Communication Protocol for Building and Control Networks“, der vom Standing Standard Program Committee 135 der American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. erarbeitet worden ist.</p> <div data-bbox="1129 1767 1469 1859"><p>Fortsetzung Seite 2 und 2 Seiten prEN und 567 Seiten ISO-Original</p></div> <p data-bbox="261 1968 1433 2027">Normenausschuss Heiz- und Raumluftechnik (NHRs) im DIN Deutsches Institut für Normung e. V. DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE</p>	<p data-bbox="1117 275 1433 304">Einsprüche bis 2003-10-31</p> <p data-bbox="1117 336 1445 394">Vorgesehen als Ersatz für DIN V ENV 1805-1:1998-03</p>

Diese Europäische Norm EN ISO 16484-5 mit dem Titel „Data Communication Protocol (Datenübertragungsprotokoll)“ wurde vom Technischen Komitee CEN/TC 247 „Mess-, Steuer- und Regeleinrichtungen für haustechnische Anlagen“ unverändert vom englischen Originaltext übernommen. Der Normenausschuss Heiz- und Raumluftechnik (NHRS) war für das DIN Deutsches Institut e. V. an der Erstellung dieser Norm beteiligt. Da davon auszugehen ist, dass die Fachleute der Kommunikationstechnik sich bereits mit der englischen Fassung befassen haben, wurde von einer Übersetzung ins Deutsche abgesehen. Nach dem DIN Präsidiumsbeschluss 13/1983 ist insbesondere auf dem Gebiet der Informatik die Veröffentlichung in Englisch zulässig, insbesondere wenn dadurch Fehler bei der Übertragung vermieden werden können.

Diese Europäische Norm EN ISO 16484-5 stellt ein Kommunikationsverfahren zur Verfügung, mit dem Geräte der Gebäudeautomation untereinander Informationen austauschen können, unabhängig von der gebäudetechnischen Anlage, zu der sie gehören. Das in dieser Norm spezifizierte Protokoll kann universell für den Datenaustausch von Geräten und Systemen in der Gebäudeautomation sowohl zum allgemeinen Einsatz, als auch für anwenderspezifische Lösungen eingesetzt werden.

Änderungen

Gegenüber DIN V ENV 1805-1:1998-03 wurden folgende Änderungen vorgenommen:

Gegenüber DIN V ENV 1805-1:1998 wurde die ISO Fassung der BACnet Spezifikation ISO/DIS 16484-5:2001-05-17 übernommen. Diese entspricht der ANSI/ASHRAE 135-2001, the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE).

English version

Building automation and control systems - Part 5: Data communication protocol (ISO/DIS 16484-5:2002)

Systèmes d'automatisation et de commande du bâtiment -
Partie 5: Protocole de communication de données
(ISO/DIS 16484-5:2002)

This draft European Standard is submitted to CEN members for parallel enquiry. It has been drawn up by the Technical Committee CEN/TC 247.

If this draft becomes a European Standard, CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

This draft European Standard was established by CEN in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the Management Centre has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Malta, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

Warning : This document is not a European Standard. It is distributed for review and comments. It is subject to change without notice and shall not be referred to as a European Standard.



EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: rue de Stassart, 36 B-1050 Brussels

Foreword

This document (prEN ISO 16484-5) has been prepared by Technical Committee ISO/TC 205 "Building environment design" in collaboration with Technical Committee CEN/TC 247 "Building Automation, Controls and Building Management", the secretariat of which is held by SNV.

This document is currently submitted to the parallel Enquiry.

This document will supersede ENV 1805-1:1998.

NOTE FROM CMC The foreword is susceptible to be amended on reception of the German language version. The confirmed or amended foreword, and when appropriate, the normative annex ZA for the references to international publications with their relevant European publications will be circulated with the German version.

Endorsement notice

The text of the International Standard ISO/DIS 16484-5:2002 has been approved by CEN as a draft European Standard without any modifications.



Building automation and control systems —

Part 5: Data communication protocol

Systèmes d'automatisation et de commande du bâtiment —

Partie 5: Protocole de communication de données

ICS 35.240.99; 91.040.01

ISO/CEN PARALLEL ENQUIRY

The CEN Secretary-General has advised the ISO Secretary-General that this ISO/DIS covers a subject of interest to European standardization. **In accordance with the ISO-lead mode of collaboration as defined in the Vienna Agreement, consultation on this ISO/DIS has the same effect for CEN members as would a CEN enquiry on a draft European Standard.** Should this draft be accepted, a final draft, established on the basis of comments received, will be submitted to a parallel two-month FDIS vote in ISO and formal vote in CEN.

To expedite distribution, this document is circulated as received from the committee secretariat. ISO Central Secretariat work of editing and text composition will be undertaken at publication stage.

Pour accélérer la distribution, le présent document est distribué tel qu'il est parvenu du secrétariat du comité. Le travail de rédaction et de composition de texte sera effectué au Secrétariat central de l'ISO au stade de publication.

In accordance with the provisions of Council Resolution 15/1993 this document is circulated in the English language only.

Conformément aux dispositions de la Résolution du Conseil 15/1993, ce document est distribué en version anglaise seulement.

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

Copyright notice

This ISO document is a draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to ISO at the address below or ISO's member body in the country of the requester:

Copyright Manager
ISO Central Secretariat
1 rue de Varembé
1211 Geneva 20 Switzerland
tel. + 41 22 749 0111
fax + 41 22 734 0179
e-mail central@iso.ch

Reproduction may be subject to royalty payments or a licensing agreement.
Violators may be prosecuted.

Building automation and control systems – Part 5: Data communication protocol

CONTENTS

FOREWORD	vii
1 PURPOSE	1
2 SCOPE	1
3 DEFINITIONS	1
3.1 Terms Adopted from International Standards	1
3.2 Terms Defined for this Standard	2
3.3 Abbreviations and Acronyms Used in this Standard	5
4 BACnet PROTOCOL ARCHITECTURE	8
4.1 The BACnet Collapsed Architecture	9
4.2 BACnet Network Topology	11
4.3 Security	13
5 THE APPLICATION LAYER	14
5.1 The Application Layer Model	14
5.2 Segmentation of BACnet Messages	18
5.3 Transmission of BACnet APDUs	19
5.4 Application Protocol State Machines	23
5.5 Application Protocol Time Sequence Diagrams	36
5.6 Application Layer Service Conventions	45
6 THE NETWORK LAYER	47
6.1 Network Layer Service Specification	47
6.2 Network Layer PDU Structure	48
6.3 Messages for Multiple Recipients	53
6.4 Network Layer Protocol Messages	54
6.5 Network Layer Procedures	56
6.6 BACnet Routers	58
6.7 Point-To-Point Half-Routers	63
7 DATA LINK/PHYSICAL LAYERS: ISO 8802-3 ("Ethernet") LAN	68
7.1 The Use of ISO 8802-2 Logical Link Control (LLC)	68
7.2 Parameters Required by the LLC Primitives	68
7.3 Parameters Required by the MAC Primitives	68
7.4 Physical Media	68
8 DATA LINK/PHYSICAL LAYERS: ARCNET LAN	70
8.1 The Use of ISO 8802-2 Logical Link Control (LLC)	70
8.2 Parameters Required by the LLC Primitives	70
8.3 Mapping the LLC Services to the ARCNET MAC Layer	70
8.4 Parameters Required by the MAC Primitives	70
8.5 Physical Media	70

9	DATA LINK/PHYSICAL LAYERS: MASTER-SLAVE/TOKEN PASSING (MS/TP) LAN	72
9.1	Service Specification	72
9.2	Physical Layer	74
9.3	MS/TP Frame Format	76
9.4	Overview of the MS/TP Network	77
9.5	MS/TP Medium Access Control	78
9.6	Cyclic Redundancy Check (CRC)	94
9.7	Interfacing MS/TP LANs with Other BACnet LANs	95
9.8	Responding BACnet User Processing of Messages from MS/TP	95
9.9	Repeaters	95
10	DATA LINK/PHYSICAL LAYERS: POINT-TO-POINT (PTP)	97
10.1	Overview	97
10.2	Service Specification	97
10.3	Point-to-Point Frame Format	102
10.4	PTP Medium Access Control Protocol	104
11	DATA LINK/PHYSICAL LAYERS: LonTalk LAN	125
11.1	The Use of ISO 8802-2 Logical Link Control (LLC)	125
11.2	Parameters Required by the LLC Primitives	125
11.3	Mapping the LLC Services to the LonTalk Application Layer	125
11.4	Parameters Required by the Application Layer Primitives	125
11.5	Physical Media	126
12	MODELING CONTROL DEVICES AS A COLLECTION OF OBJECTS	127
12.1	Analog Input Object Type	130
12.2	Analog Output Object Type	135
12.3	Analog Value Object Type	140
12.4	Averaging Object Type	145
12.5	Binary Input Object Type	148
12.6	Binary Output Object Type	153
12.7	Binary Value Object Type	159
12.8	Calendar Object Type	164
12.9	Command Object type	166
12.10	Device Object Type	169
12.11	Event Enrollment Object Type	175
12.12	File Object Type	180
12.13	Group Object Type	182
12.14	Life Safety Point Object Type	184
12.15	Life Safety Zone Object Type	190
12.16	Loop Object Type	196
12.17	Multi-state Input Object Type	203
12.18	Multi-state Output Object Type	207
12.19	Multi-state Value Object Type	211
12.20	Notification Class Object type	216
12.21	Program Object Type	219
12.22	Schedule Object Type	224
12.23	Trend Log Object Type	227
13	ALARM AND EVENT SERVICES	233
13.1	Change of Value Reporting	234
13.2	Intrinsic Reporting	236
13.3	Algorithmic Change Reporting	239
13.4	Alarm and Event Occurrence and Notification	245
13.5	AcknowledgeAlarm Service	248
13.6	ConfirmedCOVNotification Service	250
13.7	UnconfirmedCOVNotification Service	252
13.8	ConfirmedEventNotification Service	253
13.9	UnconfirmedEventNotification Service	256
13.10	GetAlarmSummary Service	258
13.11	GetEnrollmentSummary Service	260

13.12	GetEventInformation Service	263
13.13	LifeSafetyOperation Service.....	265
13.14	SubscribeCOV Service	267
13.15	SubscribeCOVProperty Service	269
14	FILE ACCESS SERVICES	272
14.1	AtomicReadFile Service	273
14.2	AtomicWriteFile Service	276
15	OBJECT ACCESS SERVICES	278
15.1	AddListElement Service	278
15.2	RemoveListElement Service.....	280
15.3	CreateObject Service	282
15.4	DeleteObject Service	284
15.5	ReadProperty Service	285
15.6	ReadPropertyConditional Service.....	287
15.7	ReadPropertyMultiple Service.....	292
15.8	ReadRange Service.....	295
15.9	WriteProperty Service	299
15.10	WritePropertyMultiple Service.....	301
16	REMOTE DEVICE MANAGEMENT SERVICES	304
16.1	DeviceCommunicationControl Service	304
16.2	ConfirmedPrivateTransfer Service	306
16.3	UnconfirmedPrivateTransfer Service	308
16.4	ReinitializeDevice Service.....	309
16.5	ConfirmedTextMessage Service.....	311
16.6	UnconfirmedTextMessage Service.....	313
16.7	TimeSynchronization Service.....	314
16.8	UTCTimeSynchronization Service.....	315
16.9	Who-Has and I-Have Services.....	316
16.10	Who-Is and I-Am Services	318
17	VIRTUAL TERMINAL SERVICES.....	320
17.1	Virtual Terminal Model	320
17.2	VT-Open Service	324
17.3	VT-Close Service.....	326
17.4	VT-Data Service	327
17.5	Default-terminal Characteristics	329
18	ERROR, REJECT, and ABORT CODES	333
18.1	Error Class - DEVICE	333
18.2	Error Class - OBJECT	333
18.3	Error Class - PROPERTY	333
18.4	Error Class - RESOURCES	334
18.5	Error Class - SECURITY	334
18.6	Error Class - SERVICES	335
18.7	Error Class - VT	336
18.8	Reject Reason	336
18.9	Abort Reason	336
19	BACnet PROCEDURES	338
19.1	Backup and Restore	338
19.2	Command Prioritization.....	341
20	ENCODING BACnet PROTOCOL DATA UNITS.....	345
20.1	Encoding the Fixed Part of BACnet APDUs.....	345
20.2	Encoding the Variable Part of BACnet APDUs	355
21	FORMAL DESCRIPTION OF APPLICATION PROTOCOL DATA UNITS	369
22	CONFORMANCE AND INTEROPERABILITY	409
22.1	Conformance to BACnet	409
22.2	BACnet Interoperability	410
23	EXTENDING BACnet TO ACCOMMODATE VENDOR PROPRIETARY INFORMATION	412
23.1	Extending Enumeration Values	412

23.2	Using the PrivateTransfer Services to Invoke Non-Standardized Services	412
23.3	Adding Proprietary Properties to a Standardized Object	413
23.4	Adding Proprietary Object Types to BACnet	413
23.5	Restrictions on Extending BACnet	414
24	NETWORK SECURITY	415
24.1	Security Architecture	415
24.2	Authentication Mechanisms	416
24.3	Data Confidentiality Mechanism	418
24.4	RequestKey Service	419
24.5	Authenticate Service	420
25	REFERENCES	423
ANNEX A - PROTOCOL IMPLEMENTATION CONFORMANCE STATEMENT (NORMATIVE)		425
ANNEX B - GUIDE TO SPECIFYING BACnet DEVICES (INFORMATIVE)		427
ANNEX C - FORMAL DESCRIPTION OF OBJECT TYPE STRUCTURES (INFORMATIVE)		428
ANNEX D - EXAMPLES OF STANDARD OBJECT TYPES (INFORMATIVE)		439
D.1	Example of an Analog Input Object	439
D.2	Example of an Analog Output Object	439
D.3	Example of an Analog Value Object	440
D.4	Example of an Averaging Object	440
D.5	Example of a Binary Input Object	440
D.6	Example of a Binary Output Object	442
D.7	Example of a Binary Value Object	443
D.8	Example of a Calendar Object	443
D.9	Example of a Command Object	444
D.10	Example of a Device Object	445
D.11	Example of an Event Enrollment Object	446
D.12	Example of a File Object	448
D.13	Example of a Group Object	448
D.14	Example of a Life Safety Point Object	449
D.15	Example of a Life Safety Zone Object	449
D.16	Example of a Loop Object	450
D.17	Example of a Multi-state Input Object	451
D.18	Example of a Multi-state Output Object	452
D.19	Example of a Multi-state Value Object	453
D.20	Example of a Notification Class Object	453
D.21	Example of a Program Object	454
D.22	Example of a Schedule Object	455
D.23	Example of a Trend Log Object	456
ANNEX E - EXAMPLES OF BACnet APPLICATION SERVICES (INFORMATIVE)		457
E.1	Alarm and Event Services	457
E.2	File Access Services	461
E.3	Object Access Services	462
E.4	Remote Device Management Services	470
E.5	Virtual Terminal Services	473
E.6	Security Services	474
ANNEX F - EXAMPLES OF APDU ENCODING (INFORMATIVE)		476
F.1	Example Encodings for Alarm and Event Services	476
F.2	Example Encodings for File Access Services	485
F.3	Example Encodings for Object Access Services	487
F.4	Example Encodings for Remote Device Management Services	501
F.5	Example Encodings for Virtual Terminal Services	506
F.6	Example Encodings for Security Services	508
ANNEX G - CALCULATION OF CRC (INFORMATIVE)		510
G.1	Calculation of the Header CRC	510
G.2	Calculation of the Data CRC	516
ANNEX H - COMBINING BACnet NETWORKS WITH NON-BACnet NETWORKS (NORMATIVE)		521
H.1	Mapping Non-BACnet Networks onto BACnet Routers	521

H.2	Multiple 'Virtual' BACnet Devices in a Single Physical Device	521
H.3	Using BACnet with the DARPA Internet Protocols	521
H.4	Using BACnet with the IPX Protocol	522
ANNEX I - COMMANDABLE PROPERTIES WITH MINIMUM ON AND OFF TIMES (INFORMATIVE)		524
ANNEX J - BACnet/IP (NORMATIVE)		526
J.1	General	526
J.2	BACnet Virtual Link Layer	526
J.3	BACnet/IP Directed Messages	530
J.4	BACnet/IP Broadcast Messages	530
J.5	Addition of Foreign B/IP Devices to an Existing B/IP Network	532
J.6	Routing Between B/IP and non-BP/IP BACnet Networks	533
J.7	Routing Between Two B/IP BACnet Networks	534
J.8	Use of IP Multicast within BACnet/IP	536
J.9	Sources for Internet Information	537
ANNEX K - BACnet INTEROPERABILITY BUILDING BLOCKS (BIBBs) (NORMATIVE)		538
K.1	Data Sharing BIBBs	538
K.1.1	BIBB - Data Sharing - ReadProperty - A (DS-RP-A)	538
K.1.2	BIBB - Data Sharing - ReadProperty - B (DS-RP-B)	538
K.1.3	BIBB - Data Sharing - ReadPropertyMultiple - A (DS-RPM-A)	538
K.1.4	BIBB - Data Sharing - ReadPropertyMultiple - B (DS-RPM-B)	538
K.1.5	BIBB - Data Sharing - ReadPropertyConditional - A (DS-RPC-A)	538
K.1.6	BIBB - Data Sharing - ReadPropertyConditional - B (DS-RPC-B)	539
K.1.7	BIBB - Data Sharing - WriteProperty - A (DS-WP-A)	539
K.1.8	BIBB - Data Sharing - WriteProperty - B (DS-WP-B)	539
K.1.9	BIBB - Data Sharing - WritePropertyMultiple - A (DS-WPM-A)	539
K.1.10	BIBB - Data Sharing - WritePropertyMultiple - B (DS-WPM-B)	539
K.1.11	BIBB - Data Sharing - COV - A (DS-COV-A)	539
K.1.12	BIBB - Data Sharing - COV - B (DS-COV-B)	540
K.1.13	BIBB - Data Sharing - COVP - A (DS-COVP-A)	540
K.1.14	BIBB - Data Sharing - COVP - B (DS-COVP-B)	540
K.1.15	BIBB - Data Sharing - COV - Unsolicited - A (DS-COVU-A)	540
K.1.16	BIBB - Data Sharing - COV - Unsolicited - B (DS-COVU-B)	540
K.2	Alarm and Event Management BIBBs	540
K.2.1	BIBB - Alarm and Event - Notification - A (AE-N-A)	541
K.2.2	BIBB - Alarm and Event - Notification Internal - B (AE-N-I-B)	541
K.2.3	BIBB - Alarm and Event - Notification External - B (AE-N-E-B)	541
K.2.4	BIBB - Alarm and Event - ACK - A (AE-ACK-A)	541
K.2.5	BIBB - Alarm and Event - ACK - B (AE-ACK-B)	541
K.2.6	BIBB - Alarm and Event - Alarm Summary - A (AE-ASUM-A)	541
K.2.7	BIBB - Alarm and Event - Alarm Summary - B (AE-ASUM-B)	542
K.2.8	BIBB - Alarm and Event - Enrollment Summary - A (AE-ESUM-A)	542
K.2.9	BIBB - Alarm and Event - Enrollment Summary - B (AE-ESUM-B)	542
K.2.10	BIBB - Alarm and Event - Information - A (AE-INFO-A)	542
K.2.11	BIBB - Alarm and Event - Information - B (AE-INFO-B)	542
K.2.12	BIBB - Alarm and Event - LifeSafety - A (AE-LS-A)	542
K.2.13	BIBB - Alarm and Event - LifeSafety - B (AE-LS-B)	542
K.3	Scheduling BIBBs	543
K.3.1	BIBB - Scheduling - A (SCHED-A)	543
K.3.2	BIBB - Scheduling - Internal - B (SCHED-I-B)	543
K.3.3	BIBB - Scheduling - External - B (SCHED-E-B)	543
K.4	Trending BIBBs	543
K.4.1	BIBB - Trending - Viewing and Modifying Trends - A (T-VMT-A)	543
K.4.2	BIBB - Trending - Viewing and Modifying Trends Internal - B (T-VMT-I-B)	543
K.4.3	BIBB - Trending - Viewing and Modifying Trends External - B (T-VMT-E-B)	543
K.4.4	BIBB - Trending - Automated Trend Retrieval - A (T-ATR-A)	544
K.4.5	BIBB - Trending - Automated Trend Retrieval - B (T-ATR-B)	544
K.5	Device and Network Management BIBBs	544

K.5.1	BIBB - Device Management-Dynamic Device Binding-A (DM-DDB-A).....	544
K.5.2	BIBB - Device Management-Dynamic Device Binding-B (DM-DDB-B).....	544
K.5.3	BIBB - Device Management-Dynamic Object Binding-A (DM-DOB-A).....	545
K.5.4	BIBB - Device Management-Dynamic Object Binding-B (DM-DOB-B).....	545
K.5.5	BIBB - Device Management-DeviceCommunicationControl-A (DM-DCC-A).....	545
K.5.6	BIBB - Device Management-DeviceCommunicationControl-B (DM-DCC-B).....	545
K.5.7	BIBB - Device Management-Private Transfer-A (DM-PT-A).....	545
K.5.8	BIBB - Device Management-Private Transfer-B (DM-PT-B).....	545
K.5.9	BIBB - Device Management-Text Message-A (DM-TM-A).....	546
K.5.10	BIBB - Device Management-Text Message-B (DM-TM-B).....	546
K.5.11	BIBB - Device Management-TimeSynchronization-A (DM-TS-A).....	546
K.5.12	BIBB - Device Management-TimeSynchronization-B (DM-TS-B).....	546
K.5.13	BIBB - Device Management-UTCTimeSynchronization-A (DM-UTC-A).....	546
K.5.14	BIBB - Device Management-UTCTimeSynchronization-B (DM-UTC-B).....	547
K.5.15	BIBB - Device Management-ReinitializeDevice-A (DM-RD-A).....	547
K.5.16	BIBB - Device Management-ReinitializeDevice-B (DM-RD-B).....	547
K.5.17	BIBB - Device Management-Backup and Restore-A (DM-BR-A).....	547
K.5.18	BIBB - Device Management-Backup and Restore-B (DM-BR-B).....	547
K.5.19	BIBB - Device Management-Restart-A (DM-R-A).....	548
K.5.20	BIBB - Device Management-Restart-B (DM-R-B).....	548
K.5.21	BIBB - Device Management-List Manipulation-A (DM-LM-A).....	548
K.5.22	BIBB - Device Management-List Manipulation-B (DM-LM-B).....	548
K.5.23	BIBB - Device Management-Object Creation and Deletion-A (DM-OCD-A).....	548
K.5.24	BIBB - Device Management-Object Creation and Deletion-B (DM-OCD-B).....	549
K.5.25	BIBB - Device Management-Virtual Terminal-A (DM-VT-A).....	549
K.5.26	BIBB - Device Management-Virtual Terminal-B (DM-VT-B).....	549
K.5.27	BIBB - Network Management-Connection Establishment-A (NM-CE-A).....	549
K.5.28	BIBB - Network Management-Connection Establishment-B (NM-CE-B).....	549
K.5.29	BIBB - Network Management-Router Configuration-A (NM-RC-A).....	550
K.5.30	BIBB - Network Management-Router Configuration-B (NM-RC-B).....	550
ANNEX L - DESCRIPTIONS AND PROFILES OF STANDARDIZED BACnet DEVICES (NORMATIVE).....		551
L.1	BACnet Operator Workstation (B-OWS).....	551
L.2	BACnet Building Controller (B-BC).....	551
L.3	BACnet Advanced Application Controller (B-AAC).....	552
L.4	BACnet Application Specific Controller (B-ASC).....	552
L.5	BACnet Smart Actuator (B-SA).....	553
L.6	BACnet Smart Sensor (B-SS).....	553
L.7	Profiles of the Standard BACnet Devices.....	554
HISTORY OF REVISIONS.....		555

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard ISO 16484-5-1 was prepared by Technical Committee ISO/TC 205, Building Environment Design, Working Group 3 WG 3, Building Control Systems Design.

ISO 16484 consists of the following parts, under the general title Building automation and control systems — Specification:

- Part 1: Overview and vocabulary
- Part 2: Hardware
- Part 3: Functions
- Part 4: Applications
- Part 5: Data communication protocol
- Part 6: Data communication conformance testing
- Part 7: Project implementation

Annexes A, H, J, K, and L form a normative part of this part of ISO 16484-5. Annexes B, C, D, E, F, G, and I are for information only.

— *Entwurf* —

1 PURPOSE

The purpose of this standard is to define data communication services and protocols for computer equipment used for monitoring and control of HVAC&R and other building systems and to define, in addition, an abstract, object-oriented representation of information communicated between such equipment, thereby facilitating the application and use of digital control technology in buildings.

2 SCOPE

2.1 This protocol provides a comprehensive set of messages for conveying encoded binary, analog, and alphanumeric data between devices including, but not limited to:

- (a) hardware binary input and output values,
- (b) hardware analog input and output values,
- (c) software binary and analog values,
- (d) text string values,
- (e) schedule information,
- (f) alarm and event information,
- (g) files, and
- (h) control logic.

2.2 This protocol models each building automation and control computer as a collection of data structures called "objects," the properties of which represent various aspects of the hardware, software, and operation of the device. These objects provide a means of identifying and accessing information without requiring knowledge of the details of the device's internal design or configuration.

3 DEFINITIONS

3.1 Terms Adopted from International Standards

The following terms used in this standard are defined by international standards or draft standards for open system interconnection (OSI). The definitions are repeated here and a reference to the appropriate standard is provided. Clause 25 contains the titles of all national and international standards referenced in this clause and elsewhere in this standard. Words or phrases in *italics* refer to terms defined elsewhere in this clause.

3.1.1 abstract syntax: the specification of application layer data or *application-protocol-control-information* by using notation rules which are independent of the encoding technique used to represent them (ISO 8822).

3.1.2 application: a set of a USER's information processing requirements (ISO 8649).

3.1.3 application-entity: the aspects of an *application-process* pertinent to OSI (ISO 7498).

3.1.4 application-process: an element within a *real open system* which performs the information processing for a particular *application* (ISO 7498).

3.1.5 application-protocol-control-information: information exchanged between *application-entities*, using presentation services, to coordinate their joint operation (ISO 9545).

3.1.6 application-protocol-data-unit: a unit of data specified in an application protocol and consisting of *application-protocol-control-information* and possibly application-user-data (ISO 9545).

3.1.7 application-service-element: that part of an *application-entity* which provides an OSI environment capability, using underlying services when appropriate (ISO 7498).

3.1.8 concrete syntax: those aspects of the rules used in the formal specification of data which embody a specific representation of that data (ISO 7498).

3.1.9 peer-entities: *entities* within the same layer (ISO 7498).

3. DEFINITIONS

3.1.10 real open system: a *real system* which complies with the requirements of OSI standards in its communication with other *real systems* (ISO 7498).

3.1.11 real system: a set of one or more computers, the associated software, peripherals, terminals, human operators, physical processes, information transfer means, etc., that forms an autonomous whole capable of performing information processing and/or information transfer (ISO 7498).

3.1.12 (N)-service-access-point: the point at which (N)-services are provided by an (N)-*entity* to an (N+1)-*entity* (ISO 7498).

3.1.13 (N)-service-data-unit: an amount of (N)-interface-data whose identity is preserved from one end of an (N)-connection to the other (ISO 7498).

3.1.14 service-user: an *entity* in a single open system that makes use of a service through *service-access-points* (ISO TR 8509).

3.1.15 service-provider: an abstract of the totality of those entities which provide a service to peer *service-users* (ISO TR 8509).

3.1.16 transfer-syntax: that *concrete syntax* used in the transfer of data between open systems (ISO 7498).

3.1.17 service-primitive; primitive: an abstract, implementation-independent representation of an interaction between the *service-user* and the *service-provider* (ISO TR 8509).

3.1.18 request (primitive): a representation of an interaction in which a *service-user* invokes some procedure (ISO TR 8509).

3.1.19 indication (primitive): a representation of an interaction in which a *service-provider* either

- a) indicates that it has, on its own initiative, invoked some procedure; or
- b) indicates that a procedure has been invoked by the *service-user* at the peer *service-access-point* (ISO TR 8509).

3.1.20 response (primitive): a representation of an interaction in which a *service-user* indicates that it has completed some procedure previously invoked by an interaction represented by an *indication* primitive (ISO TR 8509).

3.1.21 confirm (primitive): a representation of an interaction in which a *service-provider* indicates, at a particular *service-access-point*, completion of some procedure previously invoked, at that *service-access-point*, by an interaction represented by a *request* primitive (ISO TR 8509).

3.1.22 user element: the representation of that part of an *application-process* which uses those *application-service-elements* needed to accomplish the communications objectives of that *application-process* (ISO 7498).

3.2 Terms Defined for this Standard

3.2.1 access control: a method for regulating or restricting access to *network resources*.

3.2.2 alarm: 1. An annunciation, either audible or visual or both, that alerts an operator to an off-normal condition that may require corrective action. 2. An abnormal condition detected by a device or controller that implements a rule or logic specifically designed to look for that condition.

3.2.3 algorithmic change reporting: the detection and reporting of an alarm or event, based on an algorithm specified in an Event Enrollment object. See *intrinsic reporting* in 3.2.27.

3.2.4 BACnet device: any device, real or virtual, that supports digital communication using the BACnet protocol.

3.2.5 BACnet-user: that portion of an *application-process* that is represented by the BACnet *user element*.

3.2.6 bridge: a device that connects two or more *segments* at the physical and data link layers. This device may also perform message filtering based upon MAC layer addresses.

- 3.2.7 broadcast:** a message sent as a single unit, which may apply to more than one device.
- 3.2.8 change of state:** an event that occurs when a measured or calculated Boolean or discrete enumerated value changes.
- 3.2.9 change of value:** an event that occurs when a measured or calculated analog value changes by a predefined amount.
- 3.2.10 client:** a system or device that makes use of another device for some particular purpose via a service *request* instance. A client requests service from a *server*.
- 3.2.11 context:** a set of data or information that completely describes a particular communication environment at a particular point in time.
- 3.2.12 controller:** a device for regulation or management of a system or component.
- 3.2.13 data confidentiality:** the property that information is not made available or disclosed to unauthorized individuals, entities, or processes.
- 3.2.14 data integrity:** the property that data has not been altered or destroyed in an unauthorized manner.
- 3.2.15 data origin authentication:** the corroboration that the source of data received is as claimed.
- 3.2.16 directly connected network:** a network that is accessible from a router without messages being relayed through an intervening router. A PTP connection is to a directly connected network if the PTP connection is currently active and no intervening router is used.
- 3.2.17 download:** a particular type of file transfer that refers to the transfer of an executable program or database to a remote device where it may be executed.
- 3.2.18 entity:** something that has a separate and distinct existence. An identifiable item that is described by a set or collection of properties.
- 3.2.19 error detection:** a procedure used to identify the presence of errors in a communication.
- 3.2.20 error recovery:** a procedure invoked in response to a detected error that permits the information exchange to continue.
- 3.2.21 gateway:** a device that connects two or more dissimilar *networks*, permitting information exchange between them.
- 3.2.22 global:** pertaining to all devices or nodes on a communication *internetwork*.
- 3.2.23 global broadcast:** a message addressed to all devices or *nodes* on all *networks* in a BACnet *internetwork*.
- 3.2.24 half router:** a device or *node* that can participate as one partner in a PTP connection. The two half-router partners that form an active PTP connection together make up a single *router*.
- 3.2.25 initialization:** the process of establishing a known state, usually from a power up condition. Initialization may require re-establishment of a node's logical or physical address.
- 3.2.26 internetwork:** a set of two or more *networks* interconnected by *routers*. In a BACnet *internetwork*, there exists exactly one message path between any two nodes.
- 3.2.27 intrinsic reporting:** the detection and reporting of an alarm or event, based on an algorithm defined as part of the *object type* specification. No external reference to an Event Enrollment is involved. See *algorithmic change reporting* in 3.2.3.
- 3.2.28 key:** a sequence of symbols that controls the operations of encipherment and decipherment.
- 3.2.29 local:** pertaining to devices on the same *network* as the referenced device.

3. DEFINITIONS

- 3.2.30 local broadcast:** a message addressed to all devices or *nodes* on the same *network* as the originator.
- 3.2.31 medium:** the physical transmission *entity*. Typical media are twisted-pair wire, fiber optic cable, and coaxial cable.
- 3.2.32 medium access control:** a process used to maintain order and provide access to the communication *medium*.
- 3.2.33 network:** a set of one or more *segments* interconnected by *bridges* that have the same network address.
- 3.2.34 network resource:** any physical or logical *entity* that may be accessed via a communication *medium*.
- 3.2.35 node:** an addressable device connected to the communication *medium*.
- 3.2.36 object profile:** an object profile is a means of defining objects beyond those defined in Clause 12. A profile defines the set of properties, behavior, and/or requirements for a proprietary object, or for proprietary extensions to a standard object.
- 3.2.37 object type:** a generic classification of data that is defined by a set of *properties*.
- 3.2.38 operator authentication:** the corroboration that the operator logging on to a device is as claimed.
- 3.2.39 peer entity authentication:** the corroboration that a peer entity in an association is the one claimed.
- 3.2.40 physical segment:** a single contiguous *medium* to which BACnet nodes are attached.
- 3.2.41 printable character:** a character that represents a printable symbol as opposed to a device control character. These include, but are not limited to, upper- and lowercase letters, punctuation marks, and mathematical symbols. The exact set depends upon the character set being used. In ANSI X3.4 the printable characters are represented by single octets in the range X'20' - X'7E'.
- 3.2.42 property:** a particular characteristic of an *object type*.
- 3.2.43 proprietary:** within the context of BACnet, any extension of or addition to *object types*, *properties*, PrivateTransfer services, or enumerations specified in this standard.
- 3.2.44 receiving BACnet-user:** the *BACnet-user* that receives an *indication* or *confirm* service primitive.
- 3.2.45 remote:** pertaining to devices or *nodes* on a different *network* than the referenced device.
- 3.2.46 remote broadcast:** a message addressed to all devices or *nodes* on a different *network* than the originator.
- 3.2.47 repeater:** a device that connects two or more *physical segments* at the physical layer.
- 3.2.48 requesting BACnet-user:** the *BACnet-user* that assumes the role of a *client* in a confirmed service.
- 3.2.49 responding BACnet-user:** the *BACnet-user* that assumes the role of a *server* in a confirmed service.
- 3.2.50 router:** a device that connects two or more *networks* at the network layer.
- 3.2.51 security:** any of a variety of procedures used to ensure that information exchange is guarded to prevent disclosure to unauthorized individuals. Security measures are intended to prevent disclosure of sensitive information even to those who have valid access to the communication *network*. Security is distinct from access control, although some security can be provided by limiting physical access to the communication medium itself.
- 3.2.52 segment:** a segment consists of one or more *physical segments* interconnected by repeaters.
- 3.2.53 sending BACnet-user:** the *BACnet-user* that issues a *request* or *response* service primitive.

3.2.54 server: a system or device that responds to a service *request* instance for some particular purpose. The server provides service to a *client*.

3.2.55 synchronization: a facility that allows processes to define and identify specific places in a transmission or exchange that can be used to reset a communication session to a predefined state.

3.2.56 timestamp: the date and time recorded for and accompanying the record of an event or operation.

3.2.57 unit_time: the length of time required to transmit one octet with a start bit and a single stop bit. Ten bit-times.

3.2.58 upload: the process of transferring an executable program image or a database from a remote device in such a manner as to allow subsequent download.

3.3 Abbreviations and Acronyms Used in this Standard

A	application layer (prefix)
AE	application entity
ANSI	American National Standards Institute
APCI	application protocol control information
APDU	application layer protocol data unit
API	application program interface
ARCNET	attached resource computer network
ASE	application service element
ASN.1	Abstract Syntax Notation One (ISO 8824)
B' '	denotes that binary notation is used between the single quotes
BAC	building automation and control
BBMD	BACnet/IP broadcast management device
BDT	broadcast distribution table
B/IP	BACnet/IP
B/IP-M	BACnet/IP multicast
BVLC	BACnet virtual link control
BVLCI	BACnet virtual link control information
BVLL	BACnet virtual link layer
C	conditional
C(=)	conditional (The parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.)
CNF	confirm primitive
COV	change of value
CRC	cyclic redundancy check
D' '	denotes that decimal notation is used between the single quotes
DA	local destination MAC layer address
DADR	ultimate destination MAC layer address
DER	data expecting reply
DES	Data Encryption Standard (FIPS 46-1)
DID	ARCNET destination MAC address
DLEN	1-octet length of ultimate destination MAC layer address
DNET	2-octet ultimate destination network number
DSAP	LLC destination service access point (X'82' for BACnet)
EXEC	capable of executing a service request
FDT	foreign device table
ICI	interface control information
IL	ARCNET information length field
IND	indication primitive
INF	"Infinity", a unique binary pattern representing positive infinity (see ANSI/IEEE 754-1985)
-INF	"Negative infinity", a unique binary pattern representing negative infinity (see ANSI/IEEE 754-1985)
IEEE	Institute of Electrical and Electronics Engineers

3. DEFINITIONS

INIT	capable of initiating a service request
IP	Internet Protocol - RFC 791
ISO	International Organization for Standardization
L	data link (prefix)
LAN	local area network
LLC	logical link control (ISO 8802-2)
LPCI	link protocol control information
LPDU	link protocol data unit
LSAP	link service access point (X'82' for BACnet)
LSDU	link service data unit
M	mandatory
M(=)	mandatory (The parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.)
MA	medium access (prefix)
MAC	medium access control
MPCI	MAC protocol control information
MPDU	MAC layer protocol data unit
MSDU	MAC service data unit
MS/TP	master-slave/token-passing
N	network layer (prefix)
NaN	"Not a Number", a unique binary pattern representing an invalid number (see ANSI/IEEE 754-1985)
NP	network priority
NPCI	network protocol control information
NPDU	network layer protocol data unit
NRZ	non-return to zero
NSAP	network service access point
NSDU	network service data unit
O	indicates that support of a property is optional
OSI	open systems interconnection
P	physical layer (prefix)
PAC	ARCNET data packet header octet
PCI	protocol control information
PDU	protocol data unit
PICS	protocol implementation conformance statement
PK	Private Key
PPCI	physical layer protocol control information
PPDU	physical protocol data unit
PPP	Point-To-Point protocol - RFC 1661
PSDU	physical service data unit
PTP	point-to-point
R	indicates that a property shall be supported and readable using BACnet services
REQ	request primitive
RFC	request for comment
RSP	response primitive
S	selection
S(=)	selection (The parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.)
SA	local network source MAC layer address
SAP	service access point
SC	ARCNET system code (X'CD' for BACnet)
SDU	service data unit
SID	ARCNET source MAC address
SK	Session Key
SLEN	1-octet length of original source MAC layer address
SLIP	Serial Line Internet Protocol -RFC 1055
SNET	2-octet original source network number

SPC	standard project committee
SSAP	LLC source service access point (X'82' for BACnet)
TSM	transaction state machine
U	user option
U(=)	user option (The parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.)
UART	universal asynchronous receiver/transmitter
UDP	User Datagram Protocol - RFC 768
VT	virtual terminal
W	indicates that a property shall be supported, readable, and writable using BACnet services
X'	denotes that hexadecimal notation is used between the single quotes
XID	eXchange IDentification (ISO 8802-2)

4. BACnet PROTOCOL ARCHITECTURE

4 BACnet PROTOCOL ARCHITECTURE

The Open System Interconnection (OSI) - Basic Reference Model (ISO 7498) is an international standard that defines a model for developing multi-vendor computer communication protocol standards. The OSI model addresses the general problem of computer-to-computer communication and breaks this very complex problem into seven smaller, more manageable sub-problems, each of which concerns itself with a specific communication function. Each of these sub-problems forms a "layer" in the protocol architecture.

The seven layers are arranged in a hierarchical fashion as shown in Figure 4-1. A given layer provides services to the layers above and relies on services provided to it by the layers below. Each layer can be thought of as a black box with carefully defined interfaces on the top and bottom. An application process connects to the OSI application layer and communicates with a second, remote application process. This communication appears to take place between the two processes as if they were connected directly through their application layer interfaces. Minimal knowledge or understanding of the other layers is required. In a similar manner, each layer of the protocol relies on lower layers to provide communication services and establishes a virtual peer-to-peer communication with its companion layer on the other system. The only real connection takes place at the physical layer.

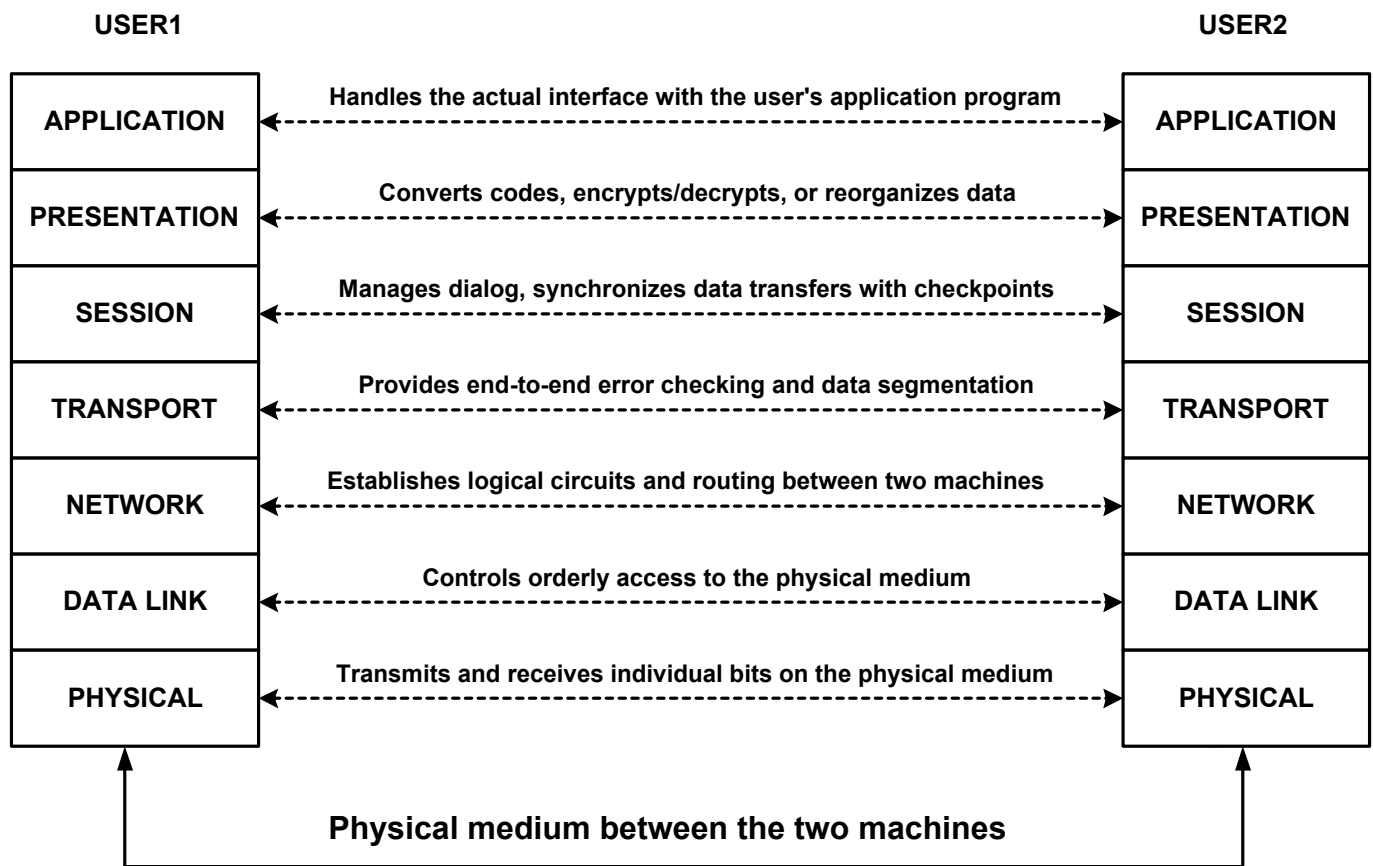


Figure 4-1. The ISO Open Systems Interconnection Basic Reference Model.

The OSI model addresses computer-to-computer communication from a very general perspective. It was designed to deal with the problems associated with computers in large, complex networks communicating with other computers in networks anywhere in the world. In this environment, computers can be separated by long distances and the messages might pass through several intermediate points, each of which may have to make routing decisions or perform some type of translation. Complex synchronization and error recovery schemes may also be needed.

The cost of implementing such a protocol today is prohibitively high for most building automation applications and is not generally required. Nevertheless, the OSI model is a good one to use for a building automation protocol if consideration is

given to including only the OSI functionality that is actually needed, thereby collapsing the seven-layer architecture. In a collapsed architecture, only selected layers of the OSI model are included. The other layers are effectively null, thus reducing message length and communication processing overhead. Such a collapsed architecture permits the building automation industry to take advantage of lower cost, mass-produced processor and local area network technologies such as have been developed for the process control and office automation industries. The use of readily available, widespread technologies, such as Ethernet¹, ARCNET², and LonTalk³, will lower the cost, increase performance, and open new doors to system integration.

4.1 The BACnet Collapsed Architecture

BACnet is based on a four-layer collapsed architecture that corresponds to the physical, data link, network, and application layers of the OSI model as shown in Figure 4-2. The application layer and a simple network layer are defined in the BACnet standard. BACnet provides five options that correspond to the OSI data link and physical layers. Option 1 is the logical link control (LLC) protocol defined by ISO 8802-2 Type 1, combined with the ISO 8802-3 medium access control (MAC) and physical layer protocol. ISO 8802-2 Type 1 provides unacknowledged connectionless service only. ISO 8802-3 is the international standard version of the well-known "Ethernet" protocol. Option 2 is the ISO 8802-2 Type 1 protocol combined with ARCNET (ATA/ANSI 878.1). Option 3 is a Master-Slave/Token-Passing (MS/TP) protocol designed specifically for building automation and control devices as part of the BACnet standard. The MS/TP protocol provides an interface to the network layer that looks like the ISO 8802-2 Type 1 protocol and controls access to an EIA-485 physical layer. Option 4, the Point-To-Point protocol, provides mechanisms for hardwired or dial-up serial, asynchronous communication. Option 5 is the LonTalk protocol. Collectively these options provide a master/slave MAC, deterministic token-passing MAC, high-speed contention MAC, dial-up access, star and bus topologies, and a choice of twisted-pair, coax, or fiber optic media. The details of these options are described in Clauses 7 through 11.

A four-layer collapsed architecture was chosen after careful consideration of the particular features and requirements of BAC networks, including a constraint that protocol overhead needed to be as small as possible. The reasoning behind the selection of the physical, data link, network, and application layers for inclusion in the BACnet architecture is outlined in this subclause.

What layers are required for the proper operation of a BAC network? BAC networks are local area networks. This is true even though in some applications it is necessary to exchange information with devices in a building that is very far away. This long-distance communication is done through telephone networks. The routing, relaying, and guaranteed delivery issues are handled by the telephone system and can be considered external to the BAC network. BAC devices are static. They don't move from place to place and the functions that they are asked to perform do not change in the sense that a manufacturing device may make one kind of part today and some very different part tomorrow. These are among the features of BAC networks that can be used to evaluate the appropriateness of the layers in the OSI model.

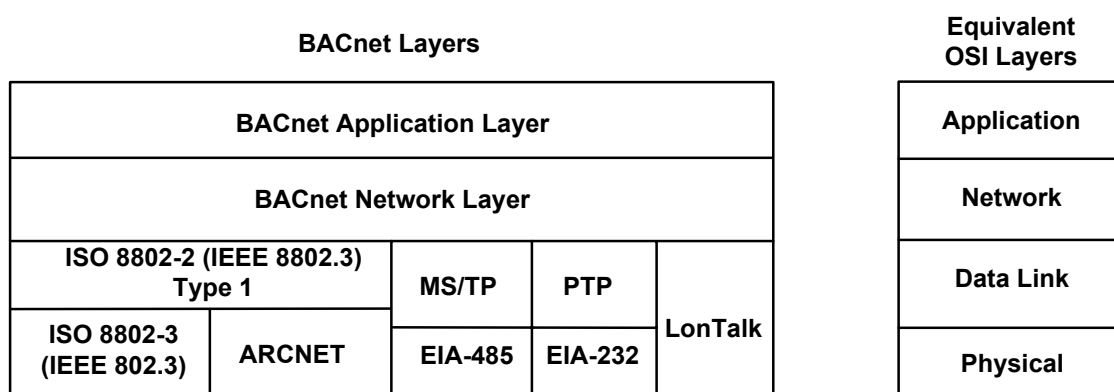


Figure 4-2. BACnet collapsed architecture.

¹ Ethernet is a registered trademark of Digital Equipment Corporation, Intel, and Xerox and is the basis for international standard ISO 8802-3.

² ARCNET is a registered trademark of Datapoint Corporation.

³ LonTalk is a registered trademark of Echelon Corporation.

4. BACnet PROTOCOL ARCHITECTURE

The physical layer provides a means of connecting the devices and transmitting the electronic signals that convey the data. Clearly the physical layer is needed in a BAC protocol.

The data link layer organizes the data into frames or packets, regulates access to the medium, provides addressing, and handles some error recovery and flow control. These are all functions that are required in a BAC protocol. The conclusion is that the data link layer is needed.

Functions provided by the network layer include translation of global addresses to local addresses, routing messages through one or more networks, accommodating differences in network types and in the maximum message size permitted by those networks, sequencing, flow control, error control, and multiplexing. BACnet is designed so that there is only one logical path between devices, thus eliminating the need for optimal path routing algorithms. A network is made up of one or more physical segments connected by repeaters or bridges but with a single local address space. In the case of a single network, most network layer functions are either unnecessary or duplicate data link layer functions. For some BACnet systems, however, the network layer is a necessity. This is the case when two or more networks in a BACnet internet use different MAC layer options. When this occurs, there is a need to recognize the difference between local and global addresses and to route messages to the appropriate networks. BACnet provides this limited network layer capability by defining a network layer header that contains the necessary addressing and control information.

The transport layer is responsible for guaranteeing end-to-end delivery of messages, segmentation, sequence control, flow control, and error recovery. Most of the functions of the transport layer are similar to functions in the data link layer, though different in scope. The scope of transport layer services is end-to-end whereas the scope of data link services is point-to-point across a single network. Since BACnet supports configurations with multiple networks, the protocol must provide the end-to-end services of the transport layer. Guaranteed end-to-end delivery and error recovery are provided in the BACnet application layer via message retry and timeout capabilities. Message segmentation and end-to-end flow control is required for buffer and processor resource management. This is because potentially large amounts of information may be returned for even simple BACnet requests. These functions are provided in the BACnet application layer. Last, sequence control is required in order to properly reassemble segmented messages. This is provided in the BACnet application layer within the segmentation procedure. Since BACnet is based on a connectionless communication model, the scope of the required services is limited enough to justify implementing these at a higher layer, thus saving the communication overhead of a separate transport layer.

The session layer is used to establish and manage long dialogues between communicating partners. Session layer functions include establishing synchronization checkpoints and resetting to previous checkpoints in the event of error conditions to avoid restarting an exchange from the beginning. Most communications in a BAC network are very brief. For example, reading or writing one or a few values, notifying a device about an alarm or event, or changing a setpoint. Occasionally longer exchanges take place, such as uploading or downloading a device. The few times when the services of this layer would be helpful do not justify the additional overhead that would be imposed on the vast majority of transactions, which are very brief and do not need them.

The presentation layer provides a way for communicating partners to negotiate the transfer syntax that will be used to conduct the communication. This transfer syntax is a translation from the abstract user view of data at the application layer to sequences of octets treated as data at the lower layers. If only one transfer syntax is permitted, then the presentation layer function reduces to an encoding scheme for representing the application data. BACnet defines such a fixed encoding scheme and includes it in the application layer, making an explicit presentation layer unnecessary.

The application layer of the protocol provides the communication services required by the applications to perform their functions, in this case monitoring and control of HVAC&R and other building systems. Clearly an application layer is needed in the protocol.

In summary:

- (a) The resource and overhead costs for implementing a full OSI seven-layer architecture make it impractical for current building automation devices.
- (b) Following the OSI model offers advantages in terms of adopting existing computer networking technology. This can result in cost savings and make integration with other computer network systems easier.

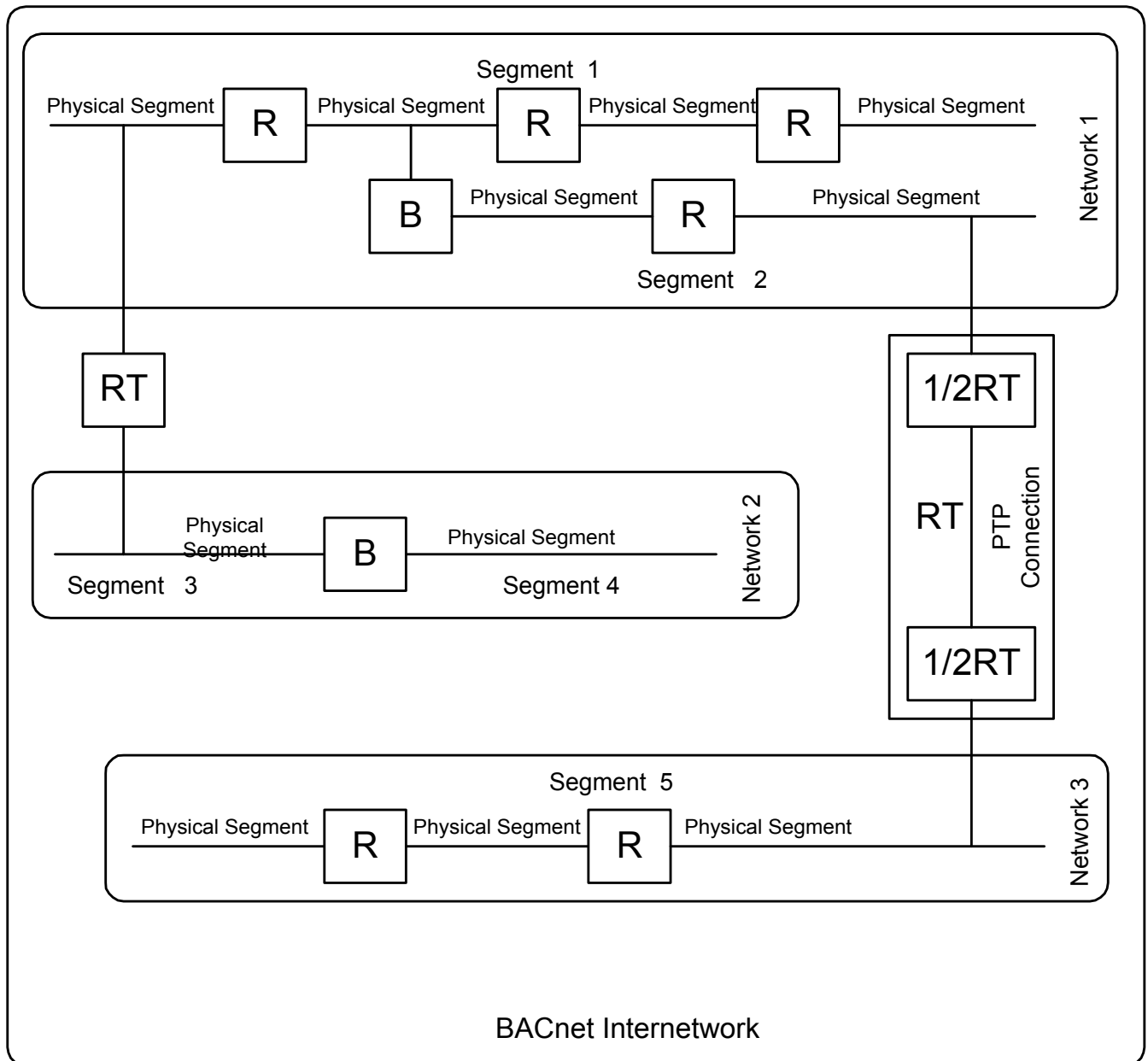
- (c) The expectations and environment of building automation systems permit simplification of the OSI model by eliminating the functionality of some of the layers.
- (d) A collapsed architecture made up of the physical, data link, network, and application layers is the optimum solution for today's building automation systems.

4.2 BACnet Network Topology

In the interest of application flexibility, BACnet does not prescribe a rigid network topology. Rather, BACnet devices are physically connected to one of four types of local area networks (LANs) or via dedicated or dial-up serial, asynchronous lines. These networks may then be further interconnected by BACnet routers as described in Clause 6.

In terms of LAN topology, each BACnet device is attached to an electrical medium or *physical segment*. A BACnet *segment* consists of one or more physical segments connected at the physical layer by *repeaters*. A BACnet *network* consists of one or more segments interconnected by *bridges*, devices that connect the segments at the physical and data link layers and may perform message filtering based upon MAC addresses; a network forms a single MAC address domain. Multiple networks, possibly employing different LAN technologies, may be interconnected by BACnet *routers* to form a BACnet *internetwork*. In a BACnet internetwork, there exists exactly one message path between any two nodes. These concepts are shown graphically in Figure 4-3.

4. BACnet PROTOCOL ARCHITECTURE



B = Bridge
R = Repeater
RT = Router
1/2RT = Half Router

Figure 4-3. A BACnet internetwork illustrating the concepts of Physical Segments, Repeaters, Segments, Bridges, Networks, Half Routers, and Routers.

4.3 Security

The principal security threats to BACnet systems are people who, intentionally or by accident, modify a device's configuration or control parameters. Problems due to an errant computer are outside the realm of security considerations. One important place for security measures is the operator-machine interface. Since the operator-machine interface is not part of the communication protocol, vendors are free to include password protection, audit trails, or other controls to this interface as needed. In addition, write access to any properties that are not explicitly required to be "writable" by this standard may be restricted to modifications made only in virtual terminal mode or be prohibited entirely. This permits vendors to protect key properties with a security mechanism that is as sophisticated as they consider appropriate. BACnet also defines services that can be used to provide peer entity, data origin, and operator authentication. See Clause 24.

5. THE APPLICATION LAYER

5 THE APPLICATION LAYER

5.1 The Application Layer Model

This clause presents a model of the BACnet application layer. The purpose of the model is to describe and illustrate the interaction between the application layer and application programs, the relationship between the application layer and lower layers in the protocol stack, and the peer-to-peer interactions with a remote application layer. This model is not an implementation specification.

An Application Process is that functionality within a system that performs the information processing required for a particular application. All parts of the Application Process outside the Application Layer, (i.e., those that do not concern the communication function) are outside the scope of BACnet. The part of the Application Process that is within the Application Layer is called the Application Entity. In other words, an Application Entity is that part of the Application Process related to the BACnet communication function. An application program interacts with the Application Entity through the Application Program Interface (API). This interface is not defined in BACnet, but it would probably be a function, procedure, or subroutine call in an actual implementation. These concepts are illustrated in Figure 5-1. The shaded region indicates the portion of the Application Process that is within the BACnet Application Layer.

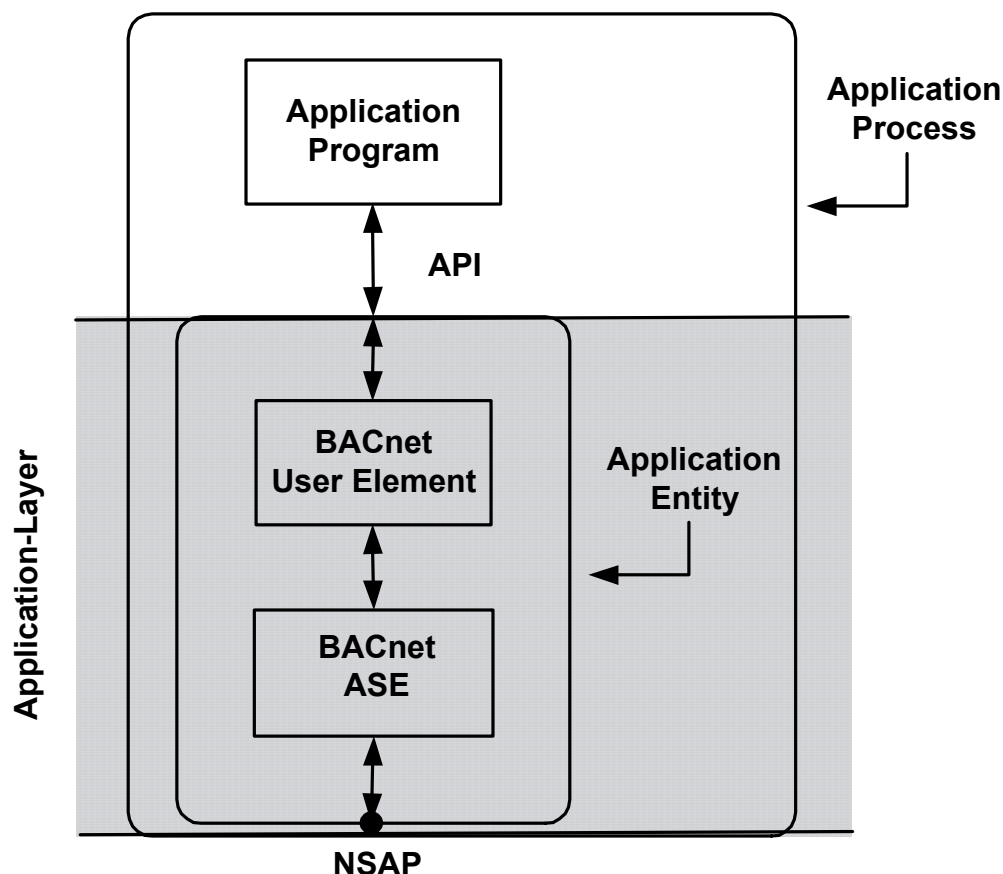


Figure 5-1. Model of a BACnet Application Process.

The Application Entity is itself made up of two parts: the BACnet User Element and the BACnet Application Service Element (ASE). The BACnet ASE represents the set of functions or application services specified in Clauses 13 through 17 and Clause 24. The BACnet User Element carries out several functions in addition to supporting the local API. It represents the implementation of the "service procedure" portion of each application service. It is responsible for maintaining information about the context of a transaction, including generating invoke IDs and remembering which invoke ID goes with which application service request (response) to (from) which device. It is also responsible for maintaining the time-out

counters that are required for the retrying of a transmission. The BACnet User Element also presides over the mapping of a device's activities into BACnet objects.

Information exchanged between two peer application processes is represented in BACnet as an exchange of abstract service primitives, following the ISO conventions contained in the OSI technical report on service conventions, ISO TR 8509. These primitives are used to convey service-specific parameters that are defined in Clauses 13 through 17 and Clause 24. Four service primitives are defined: request, indication, response, and confirm. The information contained in the primitives is conveyed using a variety of protocol data units (PDUs) defined in this standard. In order to make clear which BACnet PDU is being used, the notation will be as follows:

CONF_SERV.request	CONF_SERV.indication	CONF_SERV.response	CONF_SERV.confirm
UNCONF_SERV.request	UNCONF_SERV.indication		
SEGMENT_ACK.request	SEGMENT_ACK.indication		
REJECT.request	REJECT.indication		
ABORT.request	ABORT.indication		

The designation CONF_SERV indicates that BACnet confirmed service PDUs are being used. Similarly, the designations UNCONF_SERV, SEGMENT_ACK, ERROR, REJECT, and ABORT indicate that unconfirmed service PDUs, segment acknowledge PDUs, error PDUs, reject PDUs, and abort PDUs, respectively, are being used.

An application program that needs to communicate with a remote application process accesses the local BACnet User Element through the API. Some of the API parameters, such as the identity (address) of the device to which the service request is to be sent and protocol control information, is passed directly down to the network or data link layers. The remainder of the parameters make up an application service primitive that is passed from the BACnet User Element to the BACnet ASE. Conceptually, the application service primitive results in the generation of an APDU that becomes the data portion of a network service primitive, which is passed to the network layer through the Network Service Access Point (NSAP). Similarly this request passes down through the lower layers of the protocol stack in the local device. This process is illustrated in Figure 5-2. The message is then transmitted to the remote device, where it is passed up through the protocol stack in the remote device, eventually appearing as an indication primitive passed from the remote BACnet ASE to the remote BACnet User Element. The response from the remote device, if any, returns to the initiator of the service in a similar fashion (see 5.5).

In addition to the service primitives and the service specific parameters, the application entity exchanges interface control information (ICI) parameters with the application program via the API. The content of the ICI is dependent upon the service primitive type. The ICI parameters received by the application entity provide the information that is passed on to the lower layers (as ICI across layer interfaces) to help them construct their PDUs. The ICI parameters that are provided by the application entity to the application programs contain information recovered by the lower layers from their respective PDUs.

The following ICI parameters are exchanged with the various service primitives across an API:

'destination_address' (DA): the address of the device(s) intended to receive the service primitive. Its format (device name, network address, etc.) is a local matter. This address may also be a multicast, local broadcast or global broadcast type.

'source_address' (SA): the address of the device from which the service primitive was received. Its format (device name, network address, etc.) is a local matter.

'network_priority' (NP): a four-level network priority parameter described in 6.2.2.

'data_expecting_reply' (DER): a Boolean parameter that indicates whether (TRUE) or not (FALSE) a reply service primitive is expected for the service being issued.

BACnet Protocol Stack and Data Flow

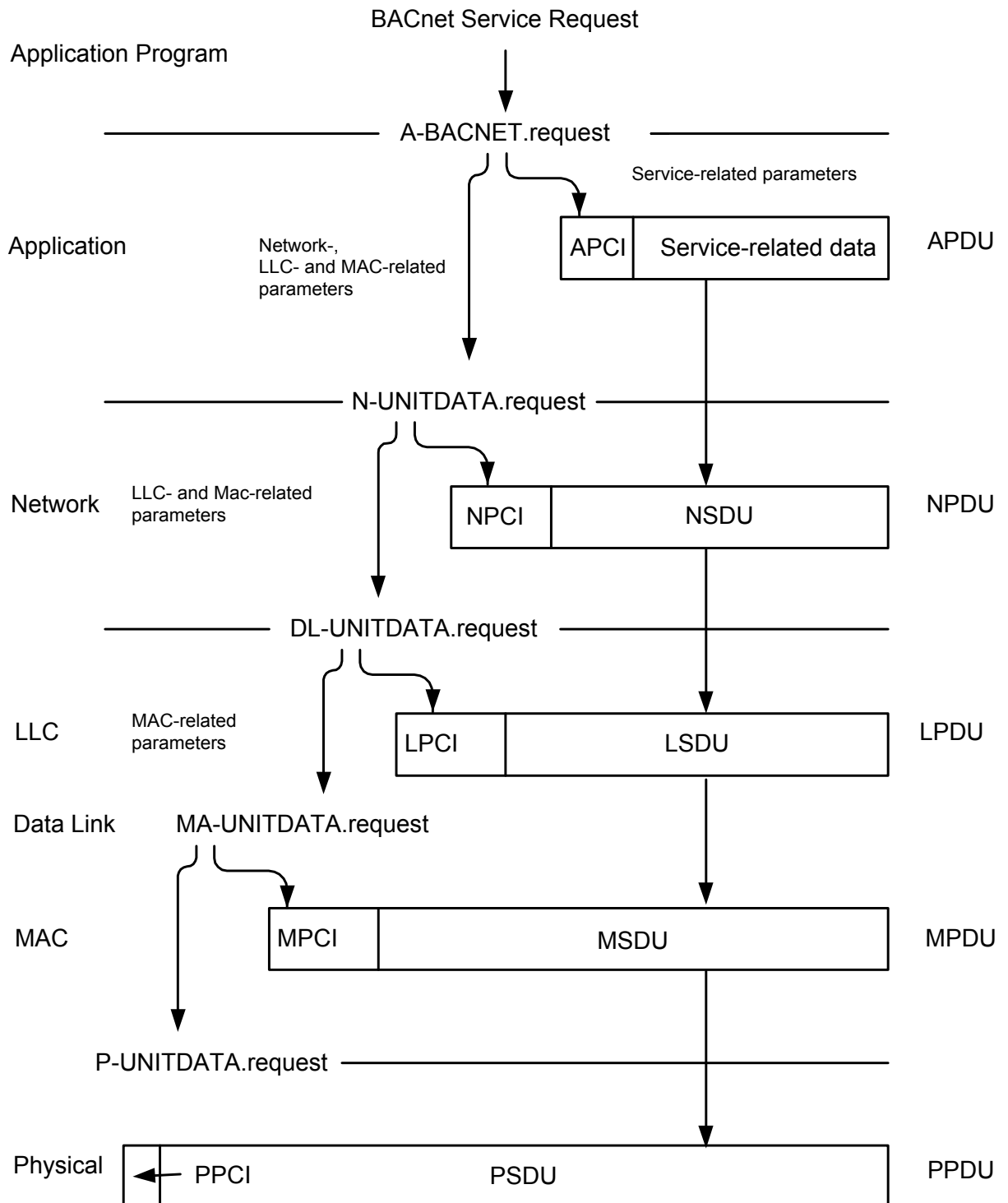


Figure 5-2. BACnet protocol stack and data flow.

Table 5-1 describes the applicability of the ICI parameters to the service primitives.

Table 5-1. Applicability of ICI parameters for abstract service primitives

Service Primitive	DA	SA	NP	DER
CONF_SERV.request	Yes	No	Yes	Yes
CONF_SERV.indication	Yes	Yes	Yes	Yes
CONF_SERV.response	Yes	No	Yes	Yes
CONF_SERV.confirm	Yes	Yes	Yes	No
UNCONF_SERV.request	Yes	No	Yes	No
UNCONF_SERV.indication	Yes	Yes	Yes	No
REJECT.request	Yes	No	Yes	No
REJECT.indication	Yes	Yes	Yes	No
SEGMENT_ACK.request	Yes	No	Yes	No
SEGMENT_ACK.indication	Yes	Yes	Yes	No
ABORT.request	Yes	No	Yes	No
ABORT.indication	Yes	Yes	Yes	No

A "BACnetDevice" is any device, real or virtual, that supports digital communication using the BACnet protocol. Each BACnet Device contains exactly one Device object, as defined in 12.9. A BACnet Device is uniquely located by an NSAP, which consists of a network number and a MAC address.

In most cases, a physical device will implement a single BACnet Device. It is possible, however, that a single physical device may implement a number of "virtual" BACnet Devices. This is described in Annex H.

5.1.1 Confirmed Application Services

BACnet defines confirmed application services based on a client and server communication model. A client requests service from a server via a particular service request instance. The server provides service to a client and responds to the request. This relationship is illustrated in Figure 5-3. The BACnet-user that assumes the role of a client is called the "requesting BACnet-user" and the BACnet-user that assumes the role of the server is called the "responding BACnet-user."

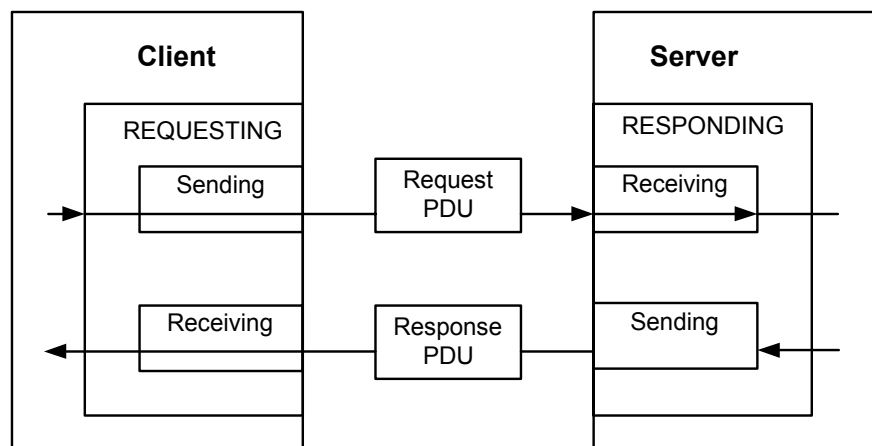


Figure 5-3. Relationship of a client and server.

A requesting BACnet-user issues a CONF_SERV.request primitive, which causes a request PDU to be sent. When a response PDU arrives, the requesting BACnet-user receives a CONF_SERV.confirm primitive. When a request PDU arrives, the

5. THE APPLICATION LAYER

responding BACnet-user receives a CONF_SERV.indication primitive. The responding BACnet-user issues a CONF_SERV.response primitive, which causes a response PDU to be sent. Thus, the requesting BACnet-user and the responding BACnet-user play a role in both sending and receiving PDUs. The term "sending BACnet-user" applies to a BACnet user that initiates the sending of a PDU. The term "receiving BACnet-user" applies to a BACnet-user that receives an indication that a PDU has arrived.

5.1.2 Unconfirmed Application Services

The client and server model, and the terms "requesting BACnet-user" and "responding BACnet-user," do not apply to unconfirmed services. The terms "sending BACnet-user" and "receiving BACnet-user" do apply, however, and they are used to define the service procedure for unconfirmed services.

5.2 Segmentation of BACnet Messages

To provide for messages that are longer than the maximum length supported by a communications network, or by the sending or receiving device, BACnet provides a method to perform application layer segmentation. In BACnet, only Confirmed-Request and Complex-ACK messages may be segmented. Segmentation is an optional feature of BACnet.

5.2.1 Message Segmentation Rules

This subclause prescribes rules for dividing a message into segments.

5.2.1.1 Rules for Segmenting APDU Data Streams

Each BACnet message is encoded into a sequence of tags and values according to the relevant ASN.1 definitions in Clause 21 and the encoding rules of Clause 20. The following rules apply to segmenting this data stream:

- (a) If possible, an entire message shall be sent in a single APDU.
- (b) If an entire message cannot be sent in a single APDU, the message shall be segmented into the minimum number of APDUs possible.
- (c) Messages shall be segmented only at octet boundaries.

5.2.1.2 Maximum APDU Length

The maximum length of a BACnet APDU shall be the smallest of

- (a) the maximum APDU size transmittable by a device, which may be restricted by local buffer limitations and is a local matter;
- (b) the maximum APDU size conveyable by the internetwork to the remote device, which is constrained by the maximum NPDU length permitted by the data links used by the local, remote, and any intervening networks, as specified in Clause 6;
- (c) the maximum APDU size accepted by the remote peer device, which must be at least 50 octets.

If the sending device is the requesting BACnet-user, i.e., the APDU to be sent is a BACnet-Confirmed-Request-PDU or a BACnet-Unconfirmed-Request-PDU, then the maximum APDU size accepted by the remote peer is specified by the Max_APDU_Length_Accepted property of the remote peer's Device object. The value of this property may be read using the read property services described in Clause 15 or the value may be obtained from the 'Max APDU Length Accepted' parameter of an I-Am service request received from the remote device. The remote peer may be solicited to transmit an I-Am service request by sending it a Who-Is service request, as described in 16.9

If the sending device is not the requesting BACnet-user, i.e., the APDU to be sent is a BACnet-Complex-ACK-PDU, then the maximum APDU size accepted by the remote peer is specified in the 'Max APDU Length Accepted' parameter of the BACnet-Confirmed-Request-PDU for which this is a response.

The value determined by the above constraints will be designated the maximum-transmittable-length. Note that maximum-transmittable-length will in general not be a constant unless minimum values are used for each constraint.

5.2.1.3 Maximum Segments Accepted

The maximum number of segments transmitted in a Confirmed-Request or ComplexACK message shall be the smallest of:

- (a) the maximum number of segments transmittable by a device, which may be restricted by local limitations and is a local matter;
- (b) the maximum number of segments accepted by the remote peer device.

If the sending device is the requesting BACnet-user, i.e., the message to be sent is a Confirmed-Request, then the maximum number of segments accepted by the remote peer device is specified in the `Max_Segments_Accepted` property of the remote peer's Device object.

If the sending device is not the requesting BACnet-user, i.e., the message to be sent is a Complex-ACK, then the maximum number of segments accepted by the remote peer device is specified in the 'Max Segments Accepted' parameter of the BACnet-Confirmed-Request-PDU for which this is a response.

5.2.2 Segmentation Protocol Control Information (PCI)

To provide for the possibility of segmented messages, the headers of the BACnet-Confirmed-Request-PDU and BACnet-ComplexACK-PDU contain two Boolean parameters called 'Segmented Message' and 'More Follows.'

If the length of a fully encoded message of the type conveyed by one of the above APDUs results in an APDU whose length is less than or equal to the maximum-transmittable-length as determined according to 5.2.1, the 'Segmented Message' and 'More Follows' parameters shall both be set to FALSE.

If, however, the encoded length of a message would result in an APDU length greater than the maximum-transmittable-length as determined according to 5.2.1, the 'Segmented Message' parameter shall be set to TRUE for all segments, and the 'More Follows' parameter shall be set to TRUE for all segments except the last.

Two additional parameters are also present, conditionally, in the header of each APDU carrying a segment of a Confirmed-Request message or a ComplexACK message. The first conditional parameter is the 'Sequence Number.' This one-octet unsigned integer is used by the segment transmitter to indicate the position of the current segment in the series of segments composing the complete message. The second conditional parameter is the 'Proposed Window Size.' This one-octet unsigned integer is used by the segment transmitter to indicate the maximum number of message segments that it is prepared to transmit before it must receive a SegmentACK. The use of these parameters in the transmission of segmented messages is described in 5.3 and 5.4.

The 'Sequence Number' of the initial segment shall be zero. The segment receiver may request the transmission of the next segment or group of segments by sending a SegmentACK-PDU containing the 'Sequence Number' parameter of the last successfully received segment. Such a request shall also serve as an acknowledgment of this segment. If the Window Size is greater than one, such a SegmentACK-PDU shall also serve to acknowledge any previously transmitted but unacknowledged segments.

If either party in a segmented transaction wishes to terminate the transaction, that party may issue an Abort-PDU.

5.3 Transmission of BACnet APDUs

The formal description of the transmission and reception protocol for BACnet APDUs is contained in the Transaction State Machine description given in 5.4. This subclause is intended only as an overview of the protocol.

5.3.1 Confirmed-Request Message Transmission

Upon transmitting a complete unsegmented Confirmed-Request message or upon receiving the SegmentACK acknowledging the final segment of a segmented Confirmed-Request message, a client device shall start a timer that indicates the length of time the message has been outstanding. The timer shall be canceled upon the receipt of an Error, Reject, Abort, SimpleACK, or ComplexACK APDU for the outstanding Confirmed-Request message, and the client application shall be notified. If the timer exceeds the value of the `APDU_Timeout` property in the client's Device object, then the complete Confirmed-Request message shall be retransmitted and the timer shall be reset to zero. All retransmitted Confirmed-Request messages shall follow this same procedure until the message has been retransmitted the number of times indicated in the

5. THE APPLICATION LAYER

Number_Of_APDU_Retries property of the client's Device object. If, after the Confirmed-Request message is retransmitted the appropriate number of times, a response is still not received, the message shall be discarded and the client application shall be notified.

5.3.2 Segmented Confirmed-Request Message Transmission

Before sending the first segment of a segmented Confirmed-Request-PDU, a client device shall choose a Proposed Window Size to indicate the maximum number of message segments it is prepared to transmit before it must receive a SegmentACK. The means of choosing the Proposed Window Size are a local matter, except that the value shall be in the range 1 to 127, inclusive. The Proposed Window Size shall be carried by the parameter of that name in each segment of the Confirmed-Request-PDU. The value of Proposed Window Size shall be the same in each segment of the Confirmed-Request-PDU.

Upon transmitting the first segment of a Confirmed-Request message, a client device shall start a timer that indicates the length of time this message segment has been outstanding. The timer shall be canceled upon the receipt of a Reject, Abort, or SegmentACK APDU for the outstanding Confirmed-Request message segment. If the timer exceeds the value of the APDU_Segment_Timeout property in the client's Device object, then the segment shall be retransmitted and the timer shall be reset to zero. All retransmitted segments shall follow this same procedure until the message segment has been retransmitted the number of times indicated in the Number_Of_APDU_Retries property of the client's Device object. If, after the message segments are retransmitted the appropriate number of times, a response is still not received, the message shall be discarded and the client application shall be notified.

Upon receipt of the first segment of a segmented Confirmed-Request-PDU, the server device shall choose an Actual Window Size to indicate the number of sequential message segments it expects to receive before it transmits a SegmentACK. The means of choosing the Actual Window Size are a local matter, except that the value shall be less than or equal to the 'proposed-window-size' parameter contained in the Confirmed-Request-PDU and shall be in the range 1 to 127, inclusive. The value of Actual Window Size shall be the same in each SegmentACK sent in response to a given Confirmed-Request. Regardless of the value of Actual Window Size, a SegmentACK shall be sent in response to the first segment of a Confirmed-Request.

Upon receipt of a SegmentACK APDU, the client device shall set its Actual Window Size equal to the value associated with the 'actual-window-size' parameter in the SegmentACK APDU. After this point, the client has authorization to send as many segments as the 'actual-window-size' parameter indicates before waiting for a SegmentACK APDU. No more than T_{seg} may be allowed to elapse between the receipt of a SegmentACK APDU and the transmission of a segment. No more than T_{seg} may be allowed to elapse between the transmission of successive segments of a group. After transmitting a set of segments that fills the window or completes the message, a client device shall start a timer that indicates the length of time these message segments have been outstanding. The timer shall be canceled upon receipt of a Reject, Abort, or SegmentACK APDU for some or all of the outstanding Confirmed-Request message segments. If the timer exceeds the value of the APDU_Segment_Timeout property in the client's Device object, then the segments shall be retransmitted and the timer shall be reset to zero. All retransmitted segments shall follow this same procedure until the message segments have been retransmitted the number of times indicated in its Device object's Number_Of_APDU_Retries property. If, after the Confirmed-Request message segments are retransmitted the appropriate number of times, a response is still not received, the message shall be discarded and the client application shall be notified.

It is possible to receive a Reject, Abort, or SegmentACK APDU during the sending of a sequence of Confirmed-Request segments even though the number of outstanding segments is less than indicated by the Actual Window Size. In this case, receipt of a Reject or Abort APDU shall terminate the Confirmed-Request transaction. Receipt of a SegmentACK APDU shall be considered as an acknowledgment for the segments up to and including the number indicated in the 'sequence-number' parameter of the SegmentACK APDU. Any unacknowledged segments shall be retransmitted following the above procedure.

It is recognized that in some cases where a Reject, Abort, or SegmentACK APDU is received, the client device may have sent, or irretrievably queued for sending, one or more (but less than Actual Window Size) additional Confirmed-Request-PDU segments.

5.3.3 Segmented ComplexACK Message Transmission

Before sending the first segment of a segmented ComplexACK-PDU, a server device shall choose a Proposed Window Size to indicate the maximum number of message segments it is prepared to transmit before it must receive a SegmentACK. The

means of choosing the Proposed Window Size are a local matter, except that the value shall be in the range 1 to 127, inclusive. The Proposed Window Size shall be carried by the parameter of that name in each segment of the Confirmed-Request-PDU. The value of Proposed Window Size shall be the same in each segment of the ComplexACK-PDU.

Upon transmitting the first segment of a ComplexACK message, a server device shall start a timer that indicates the length of time this message segment has been outstanding. The timer shall be canceled upon the receipt of an Abort or SegmentACK APDU for the outstanding ComplexACK message segment. If the timer exceeds the value of the APDU_Segment_Timeout property in the server's Device object, then the segment shall be retransmitted and the timer shall be reset to zero. All retransmitted segments shall follow this same procedure until the message segment has been retransmitted the number of times indicated in the Number_Of_APDU_Retries property of the server's Device object. If, after the message segments are retransmitted the appropriate number of times, a response is still not received, the message shall be discarded.

Upon receipt of the first segment of a segmented ComplexACK-APDU, the client device shall choose an Actual Window Size to indicate the number of sequential message segments it expects to receive before it transmits a SegmentACK. The means of choosing the Actual Window Size are a local matter, except that the value shall be less than or equal to the 'proposed-window-size' parameter contained in the ComplexACK-PDU and shall be in the range 1 to 127, inclusive. The value of Actual Window Size shall be the same in each SegmentACK sent in response to a given ComplexACK. Regardless of the value of Actual Window Size, a SegmentACK shall be sent in response to the first segment of a ComplexACK.

Upon receipt of a SegmentACK APDU, the server device shall set its Actual Window Size equal to the value associated with the 'actual-window-size' parameter in the SegmentACK APDU. After this point, the server has authorization to send as many segments as the 'actual-window-size' parameter indicates before waiting for a SegmentACK APDU. No more than T_{seg} may be allowed to elapse between the receipt of a SegmentACK APDU and the transmission of a segment. No more than T_{seg} may be allowed to elapse between the transmission of successive segments of a group. After transmitting a set of segments that fills the window or completes the message, a server device shall start a timer that indicates the length of time these message segments have been outstanding. The timer shall be canceled upon receipt of an Abort or SegmentACK APDU for some or all of the outstanding ComplexACK message segments. If the timer exceeds the value of the APDU_Segment_Timeout property in the server's Device object, then the segments shall be retransmitted and the timer shall be reset to zero. All retransmitted segments shall follow this same procedure until the message segments have been retransmitted the number of times indicated in the Number_Of_APDU_Retries property of the server's Device object. If, after the ComplexACK message segments are retransmitted the appropriate number of times, a response is still not received, the message shall be discarded.

It is possible to receive an Abort or SegmentACK APDU during the sending of a sequence of ComplexACK segments even though the number of outstanding segments is less than indicated by the Actual Window Size. In this case, receipt of an Abort APDU shall terminate the ComplexACK transaction. Receipt of a SegmentACK APDU shall be considered as an acknowledgment for the segments up to and including the number indicated in the 'sequence-number' parameter of the SegmentACK APDU. Any unacknowledged segments shall be retransmitted following the above procedure.

It is recognized that in some cases where an Abort or SegmentACK APDU is received, the server device may have sent, or irretrievably queued for sending, one or more (but less than Actual Window Size) additional ComplexACK segments.

5.3.4 SegmentACK APDU Transmission

A device shall transmit a SegmentACK upon any of the following conditions:

- (a) The device receives the initial segment of a segmented message. In this case, the 'negative-ACK' parameter of the SegmentACK shall have a value of FALSE, indicating that this is a positive acknowledgment, and the 'sequence-number' parameter of the SegmentACK shall have a value of zero, indicating that the first segment has been acknowledged and that the segment transmitter may continue sending, commencing with the next sequential segment.
- (b) The device receives a quantity of unacknowledged, sequentially numbered segments for this transaction equal to the Actual Window Size. In this case, the 'negative-ACK' parameter of the SegmentACK shall have a value of FALSE, indicating that this is a positive acknowledgment, and the 'sequence-number' parameter of the SegmentACK shall have a value equal to the 'sequence-number' parameter of the last received segment, indicating that all segments up to and including 'sequence-number' have been acknowledged and that the segment transmitter may continue sending, commencing with the next sequential segment.

5. THE APPLICATION LAYER

- (c) The device receives a segment out of order (possibly indicating that a segment has been missed). In this case, the segment receiver shall discard the out-of-order segment. In this context, "out of order" means a segment whose 'sequence-number' is not equal to the next expected 'sequence-number.' The 'negative-ACK' parameter of the SegmentACK shall have a value of TRUE, indicating that this is a negative acknowledgment. The 'sequence-number' parameter of the SegmentACK shall have a value equal to the 'sequence-number' parameter of the last received correctly ordered segment, indicating that all segments up to and including 'sequence-number' have been acknowledged and that the segment transmitter should resend, commencing with the next sequential segment after that indicated by the 'sequence-number' parameter contained in the SegmentACK.
- (d) The device receives the final segment of a message. In this case, the 'negative-ACK' parameter of the SegmentACK shall have a value of FALSE, indicating that this is a positive acknowledgment, and the 'sequence-number' parameter of the SegmentACK shall have a value equal to the 'sequence-number' parameter of the final message segment, indicating that all segments up to and including the final segment have been acknowledged.

5.3.5 Duplicate APDUs and Message Segments

5.3.5.1 Terminating Client TSMs

When using the BACnet error recovery procedures there is a possibility of the reception of duplicate messages or message segments during a transaction. At the client, a transaction begins, and a Transaction State Machine is created, when the first or only segment of a Confirmed-Request APDU is sent. The transaction ends when the client discards the Transaction State Machine due to one of the following circumstances:

- (a) after reception from the server of a SimpleACK, unsegmented ComplexACK, Error, Reject, or Abort APDU containing the transaction's invokeID;
- (b) after transmission to the server of a SegmentACK APDU for the final segment of a segmented ComplexACK APDU received from the server;
- (c) after exhausting the timeout and retry logic described in the previous subclauses;
- (d) after transmission to the server of an Abort APDU containing the transaction's invokeID (i.e., the client aborts the transaction).

5.3.5.2 Terminating Server TSMs

At the server, a transaction begins, and a Transaction State Machine is created when the first or only segment of a Confirmed-Request APDU is received. The transaction ends when the server discards the Transaction State Machine due to one of the following circumstances:

- (a) after transmission to the client of a SimpleACK, unsegmented ComplexACK, Error, Reject, or Abort APDU containing the transaction's invokeID;
- (b) after reception from the client of a SegmentACK APDU for the final segment of a segmented ComplexACK APDU transmitted by the server;
- (c) after reception from the client of an Abort APDU containing the transaction's invokeID;
- (d) after exhausting the timeout and retry logic described in the previous subclauses during the transmission of a segmented ComplexACK APDU.

5.3.5.3 Duplicate Message Procedures

The procedure for handling duplicate messages and message segments is as follows:

- (a) The server receives a duplicate Confirmed-Request message. If the server has the capability of detecting a duplicate Confirmed-Request message, the message shall be discarded. If the server cannot distinguish between duplicate and

non-duplicate messages, then the Confirmed-Request message shall be serviced. In this case, the client shall discard the server's response since the Invoke ID of the response will not bind to an active state transaction state machine.

- (b) The server receives a duplicate Confirmed-Request message segment, that is, one that has already been acknowledged with a SegmentACK. In this case, the server shall discard the duplicate segment but shall return an appropriate SegmentACK APDU. A segment can be identified uniquely by the peer address, Invoke ID, and Sequence Number of the segment.
- (c) The client receives a duplicate ComplexACK segment, that is, one that has already been acknowledged with a SegmentACK. In this case, the client shall discard the duplicate segment but shall return an appropriate SegmentACK APDU. A segment can be identified uniquely by the peer address, Invoke ID, and Sequence Number of the segment.
- (d) A Device receives a duplicate SegmentACK APDU. In this case, the device shall discard the duplicate SegmentACK APDU. Other actions, including the possible re-sending of message segments, shall occur as specified in 5.4.

5.3.6 Stale Resource Disposal

The error recovery procedure described here requires resources from both the server and the client. In the event that the error recovery process fails, the resources dedicated to this process need to be freed. In general, the resources that need to be freed are transaction specific and consist of a Transaction State Machine (TSM), timers, and APDU or APDU segment buffers. The exact time period before the resources should be freed is a local matter dependent upon the system design. As a design suggestion, it is recommended that resources should be considered stale and consequently freed:

- (a) at the client, when a complete response to the Confirmed-Request APDU is received;
- (b) at the client, when a Confirmed-Request APDU has been retransmitted the number of times specified in the Number_Of_APDU_Retries property without success;
- (c) at the client, when a Confirmed-Request APDU segment has been retransmitted the number of times specified in the Number_Of_APDU_Retries property without success;
- (d) at the server, when a complete response to a Confirmed-Request APDU has been transmitted and any associated SegmentACK received;
- (e) at the server, when a ComplexACK APDU segment has been retransmitted the number of times specified in the Number_Of_APDU_Retries property without success;
- (f) at any device, when a SegmentACK APDU has been transmitted and additional segments have not been received before the segment timeout expires.

5.4 Application Protocol State Machines

BACnet APDUs may be divided into two classes: those sent by requesting BACnet-users (clients) and those sent by responding BACnet-users (servers). All BACnet devices shall be able to act as responding BACnet-users and therefore shall be prepared to receive APDUs sent by requesting BACnet-users. Many devices will also be able to act as requesting BACnet-users, and such devices shall be prepared to receive APDUs sent by responding BACnet-users.

APDUs sent by requesting BACnet-users (clients):

BACnet-Unconfirmed-Request-PDU
BACnet-Confirmed-Request-PDU
BACnet-SegmentACK-PDU with 'server' = FALSE
BACnet-Abort-PDU with 'server' = FALSE

5. THE APPLICATION LAYER

APDUs sent by responding BACnet-users (servers):

BACnet-SimpleACK-PDU
BACnet-ComplexACK-PDU
BACnet-Error-PDU
BACnet-Reject-PDU
BACnet-SegmentACK-PDU with 'server' = TRUE
BACnet-Abort-PDU with 'server' = TRUE

Both the requesting and the responding BACnet-user shall create and maintain a Transaction State Machine (TSM) for each transaction. The TSM shall be created when the transaction begins and shall be disposed of when the transaction ends. In the state machine descriptions that follow, the creation of a TSM is represented by a transition out of the IDLE state, and the disposal of a TSM is represented by a transition into the IDLE state. A transaction is uniquely identified by the client BACnetAddress, the server BACnetAddress, and the Invoke ID (if any).

When a PDU is received from the network layer, the PDU type, the source and destination BACnetAddresses, and the Invoke ID (if any) of the PDU shall be examined to determine the type (requesting BACnet-user or responding BACnet-user) and the identity of the TSM to which the PDU shall be passed. If no such TSM exists, one shall be created.

When a request is received from the application program, the request type, the source and destination BACnetAddresses, and the Invoke ID (if any) of the request shall be examined to determine the type (requesting BACnet-user or responding BACnet-user) and the identity of the TSM to which the request shall be passed. If no such TSM exists, one shall be created.

In order to simplify the state machine description, only the case of segmentation by the Application Entity is shown. Segmentation by the Application Program is possible as well. In this case, wherever the current TSM receives a segment or group of segments and sends SegmentACK, the modified TSM would instead pass the segments to the Application Program, and SegmentACK would be sent only upon direction from the Application Program via the SEGMENT_ACK.request primitive. Reception by the modified state machine of a SegmentACK-PDU would cause it to pass a SEGMENT_ACK.indication primitive to the Application Program.

5.4.1 Variables And Parameters

The following variables are defined for each instance of Transaction State Machine:

RetryCount	used to count APDU retries
SegmentRetryCount	used to count segment retries
SentAllSegments	used to control APDU retries and the acceptance of server replies
LastSequenceNumber	stores the sequence number of the last segment received in order
InitialSequenceNumber	stores the sequence number of the first segment of a sequence of segments that fill a window
ActualWindowSize	stores the current window size
ProposedWindowSize	stores the window size proposed by the segment sender
SegmentTimer	used to perform timeout on PDU segments
RequestTimer	used to perform timeout on Confirmed Requests

The following parameters are used in the description:

T_{seg}	This parameter is the length of time a node shall wait for a SegmentACK-PDU after sending the final segment of a sequence. Its value is the value of the APDU_Segment_Timeout property of the node's Device object.
------------------------	---

T_{wait_for_seg}	This parameter is the length of time a node shall wait after sending a SegmentACK-PDU for an additional segment of the message. Its value is equal to four times the value of the APDU_Segment_Timeout property of the node's Device object.
T_{out}	This parameter represents the value of the APDU_Timeout property of the node's Device object.
N_{retry}	This parameter represents the value of the Number_Of_APDU_Retries property of the node's Device object.

5.4.2 Function InWindow

The function "InWindow" performs a modulo 256 compare of two unsigned eight-bit sequence numbers. All computations and comparisons are modulo 256 operations on unsigned eight-bit quantities.

function InWindow(seqA, seqB)

- (a) if seqA minus seqB, modulo 256, is less than ActualWindowSize, then return TRUE
- (b) else return FALSE.

Example (not normative): if ActualWindowSize is equal to 4, then

InWindow(0, 0) returns TRUE
InWindow(1, 0) returns TRUE
InWindow(3, 0) returns TRUE
InWindow(4, 0) returns FALSE
InWindow(4, 5) returns FALSE (since the modulo 256 difference 4 - 5 = 255)
InWindow(0, 255) returns TRUE (since the modulo 256 difference 0 - 255 = 1)

5.4.3 Function FillWindow

The function "FillWindow" sends PDU segments either until the window is full or until the last segment of a message has been sent. No more than **T_{seg}** may be allowed to elapse between the receipt of a SegmentACK APDU and the transmission of a segment. No more than **T_{seg}** may be allowed to elapse between the transmission of successive segments of a sequence.

function FillWindow(sequenceNumber)

- (a) Set local variable ix to zero.
- (b) If the next segment to transmit (the segment numbered sequenceNumber plus ix) is the final segment, goto step (g).
- (c) Issue an N-UNITDATA.request with 'data_expecting_reply' = TRUE to transmit the next BACnet APDU segment, with 'segmented-message' = TRUE, 'more-follows' = TRUE, 'proposed-window-size' equal to ProposedWindowSize, and 'sequence-number' = sequenceNumber plus ix, modulo 256.
- (d) Set ix equal to ix plus one.
- (e) If ix is less than ActualWindowSize, goto step (b).
- (f) Goto step (i).
- (g) Issue an N-UNITDATA.request with 'data_expecting_reply' = TRUE to transmit the final BACnet APDU segment with 'segmented-message' = TRUE, 'more-follows' = FALSE, 'proposed-window-size' = ProposedWindowSize, and 'sequence-number' = sequenceNumber plus ix, modulo 256.
- (h) Set SentAllSegments to TRUE, indicating that all segments have been transmitted at least once.
- (i) Return to the caller.

5. THE APPLICATION LAYER

5.4.4 State Machine for Requesting BACnet User (client)

5.4.4.1 IDLE

In the IDLE state, the device waits for the local application program to request a service.

SendUnconfirmed

If UNCONF_SERV.request is received from the local application program,

then issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Unconfirmed-Request-PDU, and enter the IDLE state.

SendConfirmedUnsegmented

If CONF_SERV.request is received from the local application program and the length of the APDU is less than or equal to maximum-transmittable-length as determined according to 5.2.1,

then assign an 'invokeID' to this transaction; set SentAllSegments to TRUE; set RetryCount to zero; start RequestTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = TRUE to transmit a BACnet-Confirmed-Request-PDU with 'segmented-message' = FALSE; and enter the AWAIT_CONFIRMATION state to await a reply.

CannotSend

If CONF_SERV.request is received from the local application program and the length of the APDU is greater than maximum-transmittable-length as determined according to 5.2.1 and the Max_Segments_Accepted property of the destination's Device object is known and the total APDU cannot be transmitted without exceeding the maximum number of segments accepted,

then send an ABORT.indication to the local application program and enter the IDLE state.

SendConfirmedSegmented

If CONF_SERV.request is received from the local application program and the length of the APDU is greater than maximum-transmittable-length as determined according to 5.2.1, and the Max_Segments_Accepted property of the destination's Device object is not known, or Max_Segments_Accepted is known and the total APDU can be transmitted without exceeding the maximum number of segments accepted.

then assign an 'invokeID' to this transaction; set SentAllSegments to FALSE; set RetryCount to zero; set SegmentRetryCount to zero; set InitialSequenceNumber to zero; set ProposedWindowSize to whatever value is desired; set ActualWindowSize to 1; start SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = TRUE to transmit a BACnet-Confirmed-Request-PDU containing the first segment of the message, with 'segmented-message' = TRUE, 'more-follows' = TRUE, 'sequence-number' = zero, and 'proposed-window-size' = ProposedWindowSize; and enter the SEGMENTED_REQUEST state to await an acknowledgment. (The method used to determine ProposedWindowSize is a local matter, except that the value shall be in the range 1 to 127, inclusive.)

UnexpectedSegmentInfoReceived

If an unexpected PDU indicating the existence of an active server TSM (BACnet-ComplexACK-PDU with 'segmented-message' = TRUE or BACnet-SegmentACK-PDU with 'server' = TRUE) is received from the network layer,

then issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = FALSE and enter the IDLE state.

UnexpectedPDU_Received

If an unexpected PDU not indicating the existence of an active server TSM (BACnet-SimpleACK-PDU, BACnet-ComplexACK-PDU with 'segmented-message' = FALSE, BACnet-Error-PDU, BACnet-Reject-PDU, or BACnet-Abort-PDU with 'server' = TRUE) is received from the network layer,

then enter the IDLE state. (There is no reason to issue REJECT.indication, ABORT.indication, etc., as the client has no knowledge of the transaction in question.)

5.4.4.2 SEGMENTED_REQUEST

In the SEGMENTED_REQUEST state, the device waits for a BACnet-SegmentACK-PDU for one or more segments of a BACnet-Confirmed-Request-PDU.

DuplicateACK_Received

If a BACnet-SegmentACK-PDU whose 'server' parameter is TRUE is received from the network layer and InWindow ('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber) returns a value of FALSE,

then restart SegmentTimer and enter the SEGMENTED_REQUEST state to await an acknowledgment.

NewACK_Received

If a BACnet-SegmentACK-PDU whose 'server' parameter is TRUE is received from the network layer and InWindow ('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber) returns a value of TRUE and there is at least one segment remaining to send,

then set InitialSequenceNumber equal to the 'sequence-number' parameter of the BACnet-SegmentACK-PDU plus one, modulo 256; set ActualWindowSize equal to the 'actual-window-size' parameter of the BACnet-SegmentACK-PDU; restart SegmentTimer; set SegmentRetryCount to zero; call FillWindow (InitialSequenceNumber) to transmit one or more BACnet-Confirmed-Request-PDUs containing the next ActualWindowSize segments of the message; and enter the SEGMENTED_REQUEST state to await an acknowledgment.

FinalACK_Received

If a BACnet-SegmentACK-PDU whose 'server' parameter is TRUE is received from the network layer and InWindow ('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber) returns a value of TRUE and there are no more segments to send,

then stop SegmentTimer; start RequestTimer; and enter the AWAIT_CONFIRMATION state to await a reply.

Timeout

If SegmentTimer becomes greater than T_{seg} and SegmentRetryCount is less than N_{retry} ,

then increment SegmentRetryCount; restart SegmentTimer; call FillWindow (InitialSequenceNumber) to retransmit one or more BACnet-Confirmed-Request-PDUs containing the next ActualWindowSize segments of the message; and enter the SEGMENTED_REQUEST state to await an acknowledgment.

FinalTimeout

If SegmentTimer becomes greater than T_{seg} and SegmentRetryCount is greater than or equal to N_{retry} ,

then stop SegmentTimer; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

AbortPDU_Received

If a BACnet-Abort-PDU whose 'server' parameter is TRUE is received from the network layer,

then stop SegmentTimer; send ABORT.indication to the local application program; and enter the IDLE state.

SimpleACK_Received

If a BACnet-SimpleACK-PDU is received from the network layer and SentAllSegments is TRUE,

then stop RequestTimer; send CONF_SERV.confirm(+) to the local application program; and enter the IDLE state.

5. THE APPLICATION LAYER

UnsegmentedComplexACK_Received

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is FALSE and SentAllSegments is TRUE,

then stop RequestTimer; send CONF_SERV.confirm(+) to the local application program; and enter the IDLE state.

SegmentedComplexACK_Received

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is zero and this device supports segmentation and SentAllSegments is TRUE,

then stop RequestTimer; compute ActualWindowSize based on the 'proposed-window-size' parameter of the received BACnet-ComplexACK-PDU and on local conditions; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = FALSE, and 'actual-window-size' = ActualWindowSize; start SegmentTimer; set LastSequenceNumber to zero; set InitialSequenceNumber to zero; and enter the SEGMENTED_CONF state to receive the remaining segments. (The method used to determine ActualWindowSize is a local matter, except that the value shall be less than or equal to the 'proposed-window-size' parameter of the received BACnet-ComplexACK-PDU and shall be in the range 1 to 127, inclusive.)

ErrorPDU_Received

If a BACnet-Error-PDU is received from the network layer and SentAllSegments is TRUE,

then stop RequestTimer; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

RejectPDU_Received

If a BACnet-Reject-PDU is received from the network layer and SentAllSegments is TRUE,

then stop RequestTimer; send REJECT.indication to the local application program; and enter the IDLE state.

UnexpectedPDU_Received

If a BACnet-SimpleACK-PDU, BACnet-ComplexACK-PDU, BACnet-Error-PDU, or BACnet-Reject-PDU is received from the network layer and SentAllSegments is FALSE,
or if a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and this device does not support segmentation,

or if a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is not zero,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = FALSE; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

SendAbort

If ABORT.request is received from the local application program,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = FALSE; and enter the IDLE state.

5.4.4.3 AWAIT_CONFIRMATION

In the AWAIT_CONFIRMATION state, the device waits for a response to a BACnet-Confirmed-Request-PDU.

SimpleACK_Received

If a BACnet-SimpleACK-PDU is received from the network layer,

then stop RequestTimer; send CONF_SERV.confirm(+) to the local application program; and enter the IDLE state.

UnsegmentedComplexACK_Received

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is FALSE,

then stop RequestTimer; send CONF_SERV.confirm(+) to the local application program; and enter the IDLE state.

SegmentedComplexACK_Received

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is zero and this device supports segmentation,

then stop RequestTimer; compute ActualWindowSize based on the 'proposed-window-size' parameter of the received BACnet-ComplexACK-PDU and on local conditions; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = FALSE, and 'actual-window-size' = ActualWindowSize; start SegmentTimer; set LastSequenceNumber to zero; set InitialSequenceNumber to zero; and enter the SEGMENTED_CONF state to receive the remaining segments. (The method used to determine ActualWindowSize is a local matter, except that the value shall be less than or equal to the 'proposed-window-size' parameter of the received BACnet-ComplexACK-PDU and shall be in the range 1 to 127, inclusive.)

ErrorPDU_Received

If a BACnet-Error-PDU is received from the network layer,

then stop RequestTimer; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

RejectPDU_Received

If a BACnet-Reject-PDU is received from the network layer,

then stop RequestTimer; send REJECT.indication to the local application program; and enter the IDLE state.

AbortPDU_Received

If a BACnet-Abort-PDU whose 'server' parameter is TRUE is received from the network layer,

then stop RequestTimer; send ABORT.indication to the local application program; and enter the IDLE state.

SegmentACK_Received

If a BACnet-SegmentACK-PDU whose 'server' parameter is TRUE is received from the network layer,

then discard the PDU as a duplicate, and re-enter the current state.

UnexpectedPDU_Received

If an unexpected PDU (BACnet-ComplexACK-PDU with 'segmented-message' = TRUE and 'sequence-number' not equal to zero or 'segmented-message' = TRUE and this device does not support segmentation) is received from the network layer,

then stop RequestTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = FALSE; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

TimeoutUnsegmented

If RequestTimer becomes greater than T_{out} and RetryCount is less than Number_Of_APDU_Retries and the length of the Confirmed Request APDU is less than or equal to maximum-transmittable-length as determined according to 5.2.1,

5. THE APPLICATION LAYER

then stop RequestTimer; increment RetryCount; issue an N-UNITDATA.request with 'data_expect_reply' = TRUE to transmit a BACnet-Confirmed-Request-PDU with 'segmented-message' = FALSE; start RequestTimer; and enter the AWAIT_CONFIRMATION state to await a reply.

TimeoutSegmented

If RequestTimer becomes greater than T_{out} and RetryCount is less than Number_Of_APDU_Retries and the length of the Confirmed-Request APDU is greater than maximum-transmittable-length as determined according to 5.2.1,

then stop RequestTimer; increment RetryCount; set SegmentRetryCount to zero; set SentAllSegments to FALSE; start SegmentTimer; set InitialSequenceNumber to zero; set ActualWindowSize to 1; issue an N-UNITDATA.request with 'data_expect_reply' = TRUE to transmit a BACnet-Confirmed-Request-PDU containing the first segment of the message, with 'segmented-message' = TRUE, 'more-follows' = TRUE, and 'sequence-number' = zero; and enter the SEGMENTED_REQUEST state to await an acknowledgment.

FinalTimeout

If RequestTimer becomes greater than T_{out} and RetryCount is greater than or equal to Number_Of_APDU_Retries, then stop RequestTimer; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

SendAbort

If ABORT.request is received from the local application program,

then stop RequestTimer; issue an N-UNITDATA.request with 'data_expect_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = FALSE; and enter the IDLE state.

5.4.4.4 SEGMENTED_CONF

In the SEGMENTED_CONF state, the device waits for one or more segments in response to a BACnet-SegmentACK-PDU.

NewSegmentReceived_NoSpace

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; and the segment cannot be saved due to local conditions,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expect_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = FALSE; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

NewSegmentReceived

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'more-follows' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; and whose 'sequence-number' parameter is not equal to InitialSequenceNumber plus ActualWindowSize, modulo 256,

then save the BACnet-ComplexACK-PDU segment; increment LastSequenceNumber, modulo 256; restart SegmentTimer; and enter the SEGMENTED_CONF state to receive additional segments.

LastSegmentOfGroupReceived

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; whose 'more-follows' parameter is TRUE; and whose 'sequence-number' parameter is equal to InitialSequenceNumber plus ActualWindowSize, modulo 256,

then save the BACnet-ComplexACK-PDU segment; increment LastSequenceNumber, modulo 256; issue an N-UNITDATA.request with 'data_expect_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, server = FALSE, and 'actual-window-size' = ActualWindowSize; restart SegmentTimer; and enter the SEGMENTED_CONF state to receive additional segments.

LastSegmentOfComplexACK_Received

If a ComplexACK PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; and whose 'more-follows' parameter is FALSE (i.e., the final segment),

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = FALSE, and 'actual-window-size' = ActualWindowSize; send CONF_SERV.confirm(+) containing all of the received segments to the local application program; and enter the IDLE state.

SegmentReceivedOutOfOrder

If a BACnet-ComplexACK-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is not equal to LastSequenceNumber plus 1, modulo 256,

then discard the BACnet-ComplexACK-PDU segment; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = TRUE, 'server' = FALSE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; restart SegmentTimer; and enter the SEGMENTED_CONF state to receive the remaining segments.

AbortPDU_Received

If a BACnet-Abort-PDU whose 'server' parameter is TRUE is received from the network layer,

then stop SegmentTimer; send ABORT.indication to the local application program; and enter the IDLE state.

UnexpectedPDU_Received

If an unexpected PDU (BACnet-SimpleACK-PDU, BACnet-ComplexACK-PDU with 'segmented-message' = FALSE, BACnet-Error-PDU, BACnet-Reject-PDU, or BACnet-SegmentACK-PDU with 'server' = TRUE) is received from the network layer,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = FALSE; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

Timeout

If SegmentTimer becomes greater than T_{seg} times four,

then stop SegmentTimer; send CONF_SERV.confirm(-) to the local application program; and enter the IDLE state.

SendAbort

If ABORT.request is received from the local application program,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = FALSE; and enter the IDLE state.

5.4.5 State Machine for Responding BACnet User (server)

5.4.5.1 IDLE

In the IDLE state, the device waits for a PDU from the network layer.

UnconfirmedReceived

If a BACnet-Unconfirmed-Request-PDU is received from the network layer,

then send an UNCONF_SERV.indication to the local application program, and enter the IDLE state.

5. THE APPLICATION LAYER

ConfirmedUnsegmentedReceived

If a BACnet-Confirmed-Request-PDU whose 'segmented-message' parameter is FALSE is received from the network layer,

then send a CONF_SERV.indication to the local application program, start RequestTimer; and enter the AWAIT_RESPONSE state.

ConfirmedSegmentedReceivedNotSupported

If a BACnet-Confirmed-Request-PDU whose 'segmented-message' parameter is TRUE is received from the network layer, and this device does not support the reception of segmented messages,

then issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE and 'abort-reason' = SEGMENTATION_NOT_SUPPORTED, and enter the IDLE state.

ConfirmedSegmentedReceived

If a BACnet-Confirmed-Request-PDU whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is zero is received from the network layer and the local device supports the reception of segmented messages,

then compute ActualWindowSize based on the 'proposed-window-size' parameter of the received BACnet-Confirmed-Request-PDU and on local conditions; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = TRUE, and 'actual-window-size' = ActualWindowSize; start SegmentTimer; set LastSequenceNumber to zero; set InitialSequenceNumber to zero; and enter the SEGMENTED_REQUEST state to receive the remaining segments. (The method used to determine ActualWindowSize is a local matter, except that the value shall be less than or equal to the 'proposed-window-size' parameter of the received BACnet-Confirmed-Request-PDU and shall be in the range 1 to 127, inclusive.)

AbortPDU_Received

If a BACnet-Abort-PDU whose 'server' parameter is FALSE is received from the network layer,

then enter the IDLE state.

UnexpectedPDU_Received

If an unexpected PDU (BACnet-Confirmed-Request-PDU with 'segmented-message' = TRUE and 'sequence-number' not equal to zero or BACnet-SegmentACK-PDU with 'server' = FALSE) is received from the network layer,

then issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE; and enter the IDLE state.

5.4.5.2 SEGMENTED_REQUEST

In the SEGMENTED_REQUEST state, the device waits for segments of a BACnet-Confirmed-Request-PDU.

NewSegmentReceived

If a BACnet-Confirmed-Request-PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'more-follows' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; and whose 'sequence-number' parameter is not equal to InitialSequenceNumber plus ActualWindowSize, modulo 256,

then save the BACnet-Confirmed-Request-PDU segment; increment LastSequenceNumber, modulo 256; restart SegmentTimer; and enter the SEGMENTED_REQUEST state to receive the remaining segments.

LastSegmentOfGroupReceived

If a BACnet-Confirmed-Request-PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; whose 'more-

follows' parameter is TRUE; and whose 'sequence-number' parameter is equal to InitialSequenceNumber plus ActualWindowSize, modulo 256,

then save the BACnet-Confirmed-Request-PDU segment; increment LastSequenceNumber, modulo 256; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = TRUE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; restart SegmentTimer; set InitialSequenceNumber = LastSequenceNumber; and enter the SEGMENTED_REQUEST state to receive the remaining segments.

LastSegmentOfMessageReceived

If a BACnet-Confirmed-Request-PDU is received from the network layer whose 'segmented-message' parameter is TRUE; whose 'sequence-number' parameter is equal to LastSequenceNumber plus 1, modulo 256; and whose 'more-follows' parameter is FALSE (i.e., the final segment),

then save the BACnet-Confirmed-Request-PDU segment; increment LastSequenceNumber, modulo 256; stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = TRUE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; set InitialSequenceNumber = LastSequenceNumber; send CONF_SERV.indication(+) containing all of the received segments to the local application program; start RequestTimer; and enter the AWAIT_RESPONSE state.

SegmentReceivedOutOfOrder

If a BACnet-Confirmed-Request-PDU is received from the network layer whose 'segmented-message' parameter is TRUE and whose 'sequence-number' parameter is not equal to LastSequenceNumber plus 1, modulo 256,

then discard the PDU; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = TRUE, 'server' = TRUE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; restart SegmentTimer; set InitialSequenceNumber = LastSequenceNumber; and enter the SEGMENTED_REQUEST state to receive the remaining segments.

AbortPDU_Received

If a BACnet-Abort-PDU whose server parameter is FALSE is received from the network layer,

then stop SegmentTimer and enter the IDLE state.

UnexpectedPDU_Received

If an unexpected PDU (BACnet-Confirmed-Request-PDU with 'segmented-message' = FALSE or BACnet-SegmentACK-PDU with 'server' = FALSE) is received from the network layer,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE; and enter the IDLE state.

Timeout

If SegmentTimer becomes greater than T_{seg} times four,

then stop SegmentTimer and enter the IDLE state.

SendAbort

If ABORT.request is received from the local application program,

then stop SegmentTimer, issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE, and enter the IDLE state.

5.4.5.3 AWAIT_RESPONSE

In the AWAIT_RESPONSE state, the device waits for the local application program to respond to a BACnet-Confirmed-Request-PDU. See 9.8 for specific considerations in MS/TP networks.

5. THE APPLICATION LAYER

SendSimpleACK

If a CONF_SERV.response(+) is received from the local application program, which is to be conveyed via a BACnet-SimpleACK-PDU,

then issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-SimpleACK-PDU and enter the IDLE state.

SendUnsegmentedComplexACK

If a CONF_SERV.response(+) is received from the local application program, which is to be conveyed via a BACnet-ComplexACK-PDU, and the length of the APDU is less than or equal to maximum-transmittable-length as determined according to 5.2.1,

then issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-ComplexACK-PDU with 'segmented-message' = FALSE and enter the IDLE state.

CannotSendSegmentedComplexACK

If a CONF_SERV.response(+) is received from the local application program, which is to be conveyed via a BACnet-ComplexACK-PDU, and the length of the APDU is greater than maximum-transmittable-length as determined according to 5.2.1, and either

- (a) this device does not support the transmission of segmented messages or
- (b) the client will not accept a segmented response (the 'segmented-response-accepted' parameter in BACnet-ConfirmedRequest-PDU is FALSE), or
- (c) the client's max-segments-accepted parameter in the BACnet-ConfirmedRequest-PDU is fewer than required to transmit the total APDU,

then, for case (a) and (b), issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE and 'abort-reason' = SEGMENTATION_NOT_SUPPORTED or for case (c) issue BUFFER_OVERFLOW and enter the IDLE state.

SendSegmentedComplexACK

If a CONF_SERV.response(+) is received from the local application program, which is to be conveyed via a BACnet-ComplexACK-PDU, and the length of the APDU is greater than maximum-transmittable-length as determined according to 5.2.1, and the device supports the transmission of segmented messages, and the client will accept a segmented response ('segmented-response-accepted' parameter in BACnet-ConfirmedRequest-PDU is TRUE),

then set SegmentRetryCount to zero; set InitialSequenceNumber to zero; set ProposedWindowSize to whatever value is desired; set ActualWindowSize to 1; start SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = TRUE to transmit a BACnet-ComplexACK-PDU containing the first segment of the message, with 'segmented-message' = TRUE, 'more-follows' = TRUE, 'sequence-number' = zero, and 'proposed-window-size' = ProposedWindowSize; and enter the SEGMENTED_RESPONSE state to await an acknowledgment.

SendErrorPDU

If a CONF_SERV.response(-) is received from the local application program,

then issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Error-PDU and enter the IDLE state.

SendAbort

If ABORT.request is received from the local application program,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expect_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE; and enter the IDLE state.

SendReject

If REJECT.request is received from the local application program,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expect_reply' = FALSE to transmit a BACnet-Reject-PDU; and enter the IDLE state.

AbortPDU_Received

If a BACnet-Abort-PDU whose 'server' parameter is FALSE is received from the network layer,

then send ABORT.indication to the local application program; and enter the IDLE state.

DuplicateRequestReceived

If a BACnet-Confirmed-Request-PDU whose 'segmented-message' parameter is FALSE is received from the network layer,

then discard the PDU as a duplicate request, and re-enter the current state.

DuplicateSegmentReceived

If a BACnet-Confirmed-Request-PDU whose 'segmented-message' parameter is TRUE is received from the network layer,

then discard the PDU as a duplicate segment; issue an N-UNITDATA.request with 'data_expect_reply' = FALSE to transmit a BACnet-SegmentACK-PDU with 'negative-ACK' = FALSE, 'server' = TRUE, 'sequence-number' = LastSequenceNumber, and 'actual-window-size' = ActualWindowSize; and re-enter the current state.

UnexpectedPDU_Received

If an unexpected PDU (BACnet-SegmentACK-PDU whose 'server' parameter is FALSE) is received from the network layer,

then issue an N-UNITDATA.request with 'data_expect_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE; send ABORT.indication to the local application program; and enter the IDLE state.

Timeout

If RequestTimer becomes greater than T_{out} ,

then issue an N-UNITDATA.request with 'data_expect_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE; send ABORT.indication to the local application program; and enter the IDLE state.

5.4.5.4 SEGMENTED_RESPONSE

In the SEGMENTED_RESPONSE state, the device waits for a BACnet-SegmentACK-PDU for a segment or segments of a BACnet-ComplexACK-PDU.

DuplicateACK_Received

If a BACnet-SegmentACK-PDU whose 'server' parameter is FALSE is received from the network layer and InWindow('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber) returns a value of FALSE,

then restart SegmentTimer and enter the SEGMENTED_RESPONSE state to await an acknowledgment or timeout.

NewACK_Received

If a BACnet-SegmentACK-PDU whose 'server' parameter is FALSE is received from the network layer and InWindow('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber) returns a value of TRUE and there is at least one segment remaining to send,

5. THE APPLICATION LAYER

then set InitialSequenceNumber equal to the 'sequence-number' parameter of the BACnet-SegmentACK-PDU plus one, modulo 256; set ActualWindowSize equal to the 'actual-window-size' parameter of the BACnet-SegmentACK-PDU; restart SegmentTimer; set SegmentRetryCount to zero; call FillWindow(InitialSequenceNumber) to issue an N-UNITDATA.request with 'data_expecting_reply' = TRUE to transmit one or more BACnet-ComplexACK-PDUs containing the next ActualWindowSize segments of the message; and enter the SEGMENTED_RESPONSE state to await an acknowledgment.

FinalACK_Received

If a BACnet-SegmentACK-PDU whose 'server' parameter is FALSE is received from the network layer and InWindow('sequence-number' parameter of the BACnet-SegmentACK-PDU, InitialSequenceNumber) returns a value of TRUE and there are no more segments to send,

then stop SegmentTimer, and enter the IDLE state.

Timeout

If SegmentTimer becomes greater than T_{seg} and SegmentRetryCount is less than Number_Of_APDU_Retries,

then increment SegmentRetryCount; restart SegmentTimer; call FillWindow(InitialSequenceNumber) to reissue an N-UNITDATA.request with 'data_expecting_reply' = TRUE to transmit one or more BACnet-ComplexACK-PDUs containing the next ActualWindowSize segments of the message; and enter the SEGMENTED_RESPONSE state to await an acknowledgment.

FinalTimeout

If SegmentTimer becomes greater than T_{seg} and SegmentRetryCount is greater than or equal to Number_Of_APDU_Retries,

then stop the SegmentTimer, and enter the IDLE state.

AbortPDU_Received

If a BACnet-Abort-PDU whose 'server' parameter is FALSE is received from the network layer,

then stop SegmentTimer; send ABORT.indication to the local application program; and enter the IDLE state.

UnexpectedPDU_Received

If an unexpected PDU (BACnet-Confirmed-Request-PDU) is received from the network layer,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE; and enter the IDLE state.

SendAbort

If ABORT.request is received from the local application program,

then stop SegmentTimer; issue an N-UNITDATA.request with 'data_expecting_reply' = FALSE to transmit a BACnet-Abort-PDU with 'server' = TRUE; and enter the IDLE state.

5.5 Application Protocol Time Sequence Diagrams

The flow sequence of service primitives can be represented by time-sequence diagrams. Each diagram is partitioned into three or four fields. The field labeled "Provider" represents the service-provider and the two fields labeled "User" represent the two service-users. The fourth field, if present, represents an application program. For the application layer, the vertical lines between user and provider represent the interface between the BACnet User Element and the BACnet ASE. For lower layers these vertical lines represent the service-access-points between the service-users and the service-provider. Moving from top to bottom in the diagram represents the passage of time. Arrows, placed in the areas representing the service-user, indicate the main flow of information during the execution of an interaction described by a service-primitive (i.e., to or from