

Australian/New Zealand Standard™

**Low-voltage switchgear and  
controlgear—Controller-device  
interfaces (CDIs)**

**Part 3: DeviceNet**



Standards Australia



## AS/NZS 62026.3:2001

---

This Joint Australian/New Zealand Standard was prepared by Joint Technical Committee EL-006, Industrial Switchgear and Controlgear. It was approved on behalf of the Council of Standards Australia on 21 March 2001 and on behalf of the Council of Standards New Zealand on 4 May 2001. It was published on 5 June 2001.

---

The following interests are represented on Committee EL-006:

Australasian Railway Association  
Australian Chamber of Commerce and Industry  
Australian Electrical and Electronic Manufacturers Association  
Bureau of Steel Manufacturers of Australia  
Electrical Contractors Association of New Zealand  
Electricity Supply Association of Australia  
Independent Electrical Switchboard Manufacturers Association  
Institution of Engineers Australia  
Ministry of Economic Development New Zealand  
National Electrical and Communications Association  
Testing Interests (Australia)  
WorkCover N. S. W.

---

### Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about joint Australian/New Zealand Standards can be found by visiting the Standards Australia web site at [www.standards.com.au](http://www.standards.com.au) or Standards New Zealand web site at [www.standards.co.nz](http://www.standards.co.nz) and looking up the relevant Standard in the on-line catalogue.

Alternatively, both organizations publish an annual printed Catalogue with full details of all current Standards. For more frequent listings or notification of revisions, amendments and withdrawals, Standards Australia and Standards New Zealand offer a number of update options. For information about these services, users should contact their respective national Standards organization.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Please address your comments to the Chief Executive of either Standards Australia International or Standards New Zealand at the address shown on the back cover.

---

*This Standard was issued in draft form for comment as DR 00318.*

# Australian/New Zealand Standard™

## Low-voltage switchgear and controlgear—Controller-device interfaces (CDIs)

### Part 3: DeviceNet

First published as AS/NZS 62026.3:2001.

#### **COPYRIGHT**

© Standards Australia/Standards New Zealand

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Jointly published by Standards Australia International Ltd, GPO Box 5420, Sydney, NSW 2001 and Standards New Zealand, Private Bag 2439, Wellington 6020

ISBN 0 7337 3865 6



**Documents**  
Document Sharing Hub

## PREFACE

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee EL-006, Industrial Switchgear and Controlgear.

The objective of this Standard is to specify requirements for interfaces between controllers and switching elements, normal service conditions for devices, constructional and performance requirements and tests to verify conformance to requirements.

This Standard is Part 3 of a series which, when complete, will consist of the following:

AS/NZS

- 62026 Low-voltage switchgear and controlgear—Controller-device interfaces (CDIs)
- 62026.1 Part 1: General rules
- 62026.2 Part 2: Actuator sensor interface (AS-i)
- 62026.3 Part 3: DeviceNet (This Standard)
- 62026.5 Part 5: Smart distributed system (SDS)
- 62026.6 Part 6: Seriplex (Serial multiplexed control Bus)

This Standard is identical with and has been reproduced from IEC 62026-3:2000, *Low-voltage switchgear and controlgear—Controller-device interfaces (CDIs)—Part 3: DeviceNet*.

This Standard covers a DeviceNet intended for use in, but not limited to, industrial automation applications. These applications may include devices such as limit switches, proximity sensors, electro-pneumatic valves, relays, motor-starters, operator interface panels, analogue inputs, analogue outputs and controllers.

The provisions of the general rules in AS/NZS 62026.1 are applicable to this Joint Australian/New Zealand Standard, where specifically called for. General rules, clauses and subclauses thus applicable, as well as tables, figures and annexes, are identified by reference to Part 1 of the IEC Standard from which this Standard is reproduced, for example subclause 7.2.4.1 of IEC 62026-1.

A reference to an International Standard identified in the Normative References Clause by ~~striketrough (example)~~ is replaced by a reference to the Australian or Australian/New Zealand Standard(s) listed immediately thereafter and identified by shading (example). Where the struck-through referenced document and the referenced Australian or Australian/New Zealand Standard are identical, this is indicated in parenthesis after the title of the latter.

As this Standard is reproduced from an International Standard, the following applies:

- (a) Its number does not appear on each page of text and its identity is shown only on the cover and title page.
- (b) In the source text 'this standard' should read 'this Australian/New Zealand Standard'.
- (c) A full point should be substituted for a comma when referring to a decimal marker.

The term 'normative' has been used in this Standard to define the application of the annex to which it applies. A 'normative' annex is an integral part of a Standard.

## CONTENTS

|                                                                | <i>Page</i> |
|----------------------------------------------------------------|-------------|
| Clause                                                         |             |
| 1 Scope .....                                                  | 1           |
| 2 Normative references .....                                   | 1           |
| 3 Definitions and abbreviations .....                          | 3           |
| 4 Classification .....                                         | 8           |
| 4.1 General .....                                              | 8           |
| 4.2 DeviceNet communication model .....                        | 9           |
| 4.3 DeviceNet and CAN .....                                    | 9           |
| 5 Characteristics .....                                        | 10          |
| 5.1 DeviceNet connections .....                                | 10          |
| 5.2 DeviceNet messaging protocol .....                         | 12          |
| 5.3 DeviceNet communication object classes .....               | 28          |
| 5.4 Link access state machine .....                            | 56          |
| 5.5 Predefined master/slave connection set .....               | 59          |
| 5.6 Physical layer .....                                       | 69          |
| 6 Product information .....                                    | 74          |
| 7 Normal service, mounting and transport conditions .....      | 74          |
| 7.1 Normal service conditions .....                            | 74          |
| 7.2 Conditions during transport and storage .....              | 75          |
| 7.3 Mounting .....                                             | 76          |
| 8 Constructional and performance requirements .....            | 76          |
| 8.1 Indicators and configuration switches .....                | 76          |
| 8.2 DeviceNet cable .....                                      | 79          |
| 8.3 Terminating resistors .....                                | 80          |
| 8.4 Connectors .....                                           | 80          |
| 8.5 Device taps and power taps .....                           | 82          |
| 8.6 Link powered devices .....                                 | 83          |
| 8.7 Miswiring protection .....                                 | 84          |
| 8.8 Power supplies .....                                       | 84          |
| 8.9 Electromagnetic compatibility (EMC) .....                  | 85          |
| 9 Tests .....                                                  | 87          |
| 9.1 General .....                                              | 87          |
| 9.2 Electrical and EMC testing .....                           | 87          |
| 9.3 Logical testing .....                                      | 95          |
| Annex A (normative) Common services .....                      | 98          |
| Annex B (normative) DeviceNet error codes .....                | 105         |
| Annex C (normative) Connection path attribute definition ..... | 106         |
| Annex D (normative) Data type specification and encoding ..... | 109         |
| Annex E (normative) Communication objects library .....        | 113         |
| Annex F (normative) Value ranges .....                         | 133         |

|                                                                                              |    |
|----------------------------------------------------------------------------------------------|----|
| Figure 1 – Typical DeviceNet controller-device interfaces .....                              | 8  |
| Figure 2 – DeviceNet protocol architecture compared with the OSI reference model .....       | 10 |
| Figure 3 – DeviceNet's use of the CAN identifier field .....                                 | 11 |
| Figure 4 – Explicit message CAN data field use .....                                         | 12 |
| Figure 5 – Explicit message data field format .....                                          | 12 |
| Figure 6 – Explicit message header format .....                                              | 13 |
| Figure 7 – Service field format .....                                                        | 14 |
| Figure 8 – Open explicit messaging connection request format .....                           | 15 |
| Figure 9 – Open explicit messaging connection response format .....                          | 17 |
| Figure 10 – Close connection request format .....                                            | 19 |
| Figure 11 – Close connection response format .....                                           | 19 |
| Figure 12 – Non-fragmented explicit request message format .....                             | 20 |
| Figure 13 – Non-fragmented success response message format .....                             | 21 |
| Figure 14 – Error response message .....                                                     | 21 |
| Figure 15 – Data field of an I/O message .....                                               | 22 |
| Figure 16 – Format of DeviceNet fragmentation protocol .....                                 | 23 |
| Figure 17 – I/O message fragment format .....                                                | 23 |
| Figure 18 – Explicit message fragment format .....                                           | 24 |
| Figure 19 – Acknowledgement message format .....                                             | 25 |
| Figure 20 – Duplicate MAC ID check CAN identifier field .....                                | 27 |
| Figure 21 – Duplicate MAC ID check message data field format .....                           | 27 |
| Figure 22 – Transportclass_trigger .....                                                     | 31 |
| Figure 23 – Initial_comm_characteristics attribute format .....                              | 32 |
| Figure 24 – I/O connection object state transition diagram .....                             | 39 |
| Figure 25 – Explicit messaging connection object state transition diagram .....              | 41 |
| Figure 26 – Allocation choice byte contents .....                                            | 49 |
| Figure 27 – Allocate_master/slave_connection_set request message .....                       | 50 |
| Figure 28 – Success response to allocate_master/slave_connection_set request .....           | 51 |
| Figure 29 – Parent explicit messaging connection logic .....                                 | 53 |
| Figure 30 – Release choice byte contents .....                                               | 54 |
| Figure 31 – Release_master/slave_connection_set request message .....                        | 54 |
| Figure 32 – Success response to release_master/slave_connection_set request .....            | 55 |
| Figure 33 – Link access state transition diagram .....                                       | 57 |
| Figure 34 – Predefined master/slave I/O connection state transition diagram .....            | 62 |
| Figure 35 – Predefined master/slave explicit messaging state transition diagram .....        | 64 |
| Figure 36 – Physical layer block diagram .....                                               | 70 |
| Figure 37 – Device containing a non-isolated physical layer .....                            | 72 |
| Figure 38 – Device containing an isolated physical layer .....                               | 72 |
| Figure 39 – Physical configuration of thick and thin cable .....                             | 79 |
| Figure 40 – Power supply rise time test circuit .....                                        | 87 |
| Figure 41 – Current consumption test circuit .....                                           | 88 |
| Figure 42 – Power ON test circuit .....                                                      | 89 |
| Figure 43 – Test circuit for reversal of $U_+$ and $U_-$ , and also $U_-$ interruption ..... | 89 |
| Figure 44 – Differential impedance test circuit .....                                        | 90 |
| Figure 45 – Transmit level test set-up .....                                                 | 91 |

|                                                                                         | <i>Page</i> |
|-----------------------------------------------------------------------------------------|-------------|
| Figure 46 – Transmit levels.....                                                        | 91          |
| Figure 47 – Timing test set-up .....                                                    | 92          |
| Figure 48 – Timing.....                                                                 | 92          |
| Figure 49 – CDI test configuration.....                                                 | 93          |
| Figure 50 – Test circuit for logical tests.....                                         | 95          |
| Figure C.1 – Segment type/format .....                                                  | 106         |
| Figure C.2 – Logical segment .....                                                      | 107         |
| Figure C.3 – Symbolic segment .....                                                     | 107         |
| Figure C.4 – Data segment .....                                                         | 108         |
| Figure E.1 – State transition diagram for identity object .....                         | 118         |
| Figure E.2 – Typical timing relationships for acknowledged data production.....         | 128         |
| Figure E.3 – Typical timing diagram for a change of state system .....                  | 128         |
| Figure E.4 – Message flow – One connection object, one consumer .....                   | 128         |
| Figure E.5 – Message flow – Multi-consumer configuration .....                          | 129         |
| <br>                                                                                    |             |
| Table 1 – Message body format values .....                                              | 16          |
| Table 2 – Group select values.....                                                      | 16          |
| Table 3 – Source message ID in open explicit messaging connection request.....          | 17          |
| Table 4 – Destination message ID in an open explicit messaging connection response..... | 18          |
| Table 5 – UCMM error conditions/codes .....                                             | 19          |
| Table 6 – Fragment type bit values .....                                                | 23          |
| Table 7 – Ack status bit values .....                                                   | 26          |
| Table 8 – Connection class attribute .....                                              | 28          |
| Table 9 – Connection class services .....                                               | 29          |
| Table 10 – Connection object instance attributes .....                                  | 30          |
| Table 11 – Values assigned to the state attribute.....                                  | 31          |
| Table 12 – Values assigned to the instance_type attribute.....                          | 31          |
| Table 13 – Values defined for the produced_connection_id attribute .....                | 32          |
| Table 14 – Values defined for the consumed_connection_id attribute .....                | 32          |
| Table 15 – Values for the initial production characteristics nibble.....                | 33          |
| Table 16 – Values for the initial consumption characteristics nibble .....              | 33          |
| Table 17 – Values for the watchdog_timeout_action.....                                  | 35          |
| Table 18 – Connection object instance services.....                                     | 38          |
| Table 19 – Service data for connection object apply_attributes response.....            | 38          |
| Table 20 – I/O connection state-event matrix .....                                      | 39          |
| Table 21 – Explicit messaging connection state-event matrix .....                       | 42          |
| Table 22 – I/O connection object attribute access .....                                 | 43          |
| Table 23 – Explicit messaging connection object attribute access.....                   | 44          |
| Table 24 – Class attributes for the DeviceNet object .....                              | 45          |
| Table 25 – DeviceNet object class common services .....                                 | 45          |
| Table 26 – Instance attributes for the DeviceNet object .....                           | 45          |
| Table 27 – Bit rate attribute values .....                                              | 46          |
| Table 28 – BOI attribute values.....                                                    | 47          |
| Table 29 – DeviceNet object instance common services .....                              | 48          |
| Table 30 – DeviceNet object class-specific services .....                               | 48          |

|                                                                                            |     |
|--------------------------------------------------------------------------------------------|-----|
| Table 31 – Allocate_master/slave_connection_set request service data field parameters..... | 49  |
| Table 32 – Allocate_master/slave_connection_set response parameters .....                  | 50  |
| Table 33 – Release_master/slave_connection_set request service data field parameters ..... | 54  |
| Table 34 – DeviceNet object-specific additional error codes.....                           | 56  |
| Table 35 – Link access state-event matrix .....                                            | 58  |
| Table 36 – Predefined master/slave connection set identifier fields.....                   | 60  |
| Table 37 – Connection instance IDs for predefined master/slave connections .....           | 61  |
| Table 38 – Predefined master/slave I/O connection state-event matrix.....                  | 62  |
| Table 39 – Predefined master/slave I/O connection state-event matrix.....                  | 64  |
| Table 40 – Predefined master/slave I/O connection object attribute access.....             | 65  |
| Table 41 – General physical layer characteristics .....                                    | 69  |
| Table 42 – Transmitter characteristics .....                                               | 70  |
| Table 43 – Receiver characteristics .....                                                  | 71  |
| Table 44 – Total drop line cable length .....                                              | 73  |
| Table 45 – Load limits.....                                                                | 74  |
| Table 46 – Module status indicator .....                                                   | 76  |
| Table 47 – CDI status indicator .....                                                      | 76  |
| Table 48 – Combined module/CDI status indicator .....                                      | 77  |
| Table 49 – I/O status indicator .....                                                      | 78  |
| Table 50 – DeviceNet bit rate switch encoding .....                                        | 78  |
| Table 51 – Indicator and switch marking .....                                              | 78  |
| Table 52 – Cable: data pair specification .....                                            | 79  |
| Table 53 – Cable: d.c. power pair specification .....                                      | 80  |
| Table 54 – Cable: general specifications.....                                              | 80  |
| Table 55 – Open connector general specifications .....                                     | 80  |
| Table 56 – Sealed connector general specifications.....                                    | 81  |
| Table 57 – Contact specifications .....                                                    | 81  |
| Table 58 – Electrical specifications .....                                                 | 81  |
| Table 59 – Connector contact arrangement.....                                              | 82  |
| Table 60 – Internal pass-through conductor specifications .....                            | 82  |
| Table 61 – Internal drop conductor specifications .....                                    | 82  |
| Table 62 – Internal pass-through conductor specifications .....                            | 83  |
| Table 63 – Internal power drop conductor specifications .....                              | 83  |
| Table 64 – Voltage regulator specifications.....                                           | 84  |
| Table 65 – DeviceNet power supply specifications .....                                     | 84  |
| Table 66 – Immunity performance criteria .....                                             | 85  |
| Table A.1 – DeviceNet service codes and names.....                                         | 98  |
| Table A.2 – Service data and success response data field parameters .....                  | 99  |
| Table A.3 – Service data for get_attributes_all success response .....                     | 99  |
| Table A.4 – Service data for set_attributes_all request .....                              | 100 |
| Table A.5 – Service data for create success response .....                                 | 101 |
| Table A.6 – Service data for get_attribute_single request .....                            | 102 |
| Table A.7 – Service data for get_attribute_single success response .....                   | 102 |
| Table A.8 – Service data for set_attribute_single request.....                             | 103 |
| Table A.9 – Service data for error response request .....                                  | 103 |

|                                                                                    | <i>Page</i> |
|------------------------------------------------------------------------------------|-------------|
| Table B.1 – DeviceNet error codes .....                                            | 105         |
| Table D.1 – Elementary data types .....                                            | 110         |
| Table E.1 – Object class codes and names.....                                      | 113         |
| Table E.2 – Identity object class attributes.....                                  | 114         |
| Table E.3 – Identity object instance attributes.....                               | 114         |
| Table E.4 – Bit definitions for status instance attribute of identity object ..... | 115         |
| Table E.5 – Identity object common services .....                                  | 116         |
| Table E.6 – Identity object reset request.....                                     | 116         |
| Table E.7 – Identity reset service parameter values .....                          | 116         |
| Table E.8 – Identity object get_attributes_all response, class level .....         | 117         |
| Table E.9 – Identity object get_attributes_all response, instance level .....      | 117         |
| Table E.10 – State-event matrix for identity object.....                           | 119         |
| Table E.11 – Message router class attributes.....                                  | 120         |
| Table E.12 – Message router instance attributes.....                               | 120         |
| Table E.13 – Message router common services .....                                  | 120         |
| Table E.14 – Get_attributes_all response, class level .....                        | 121         |
| Table E.15 – Get_attributes_all response, instance level .....                     | 121         |
| Table E.16 – Acknowledgement handler object class attributes.....                  | 122         |
| Table E.17 – Acknowledgement handler instance attributes.....                      | 123         |
| Table E.18 – Acknowledgement handler object common services .....                  | 124         |
| Table E.19 – Acknowledgement handler object-specific services .....                | 124         |
| Table E.20 – Acknowledgement handler add_ackdata_path request .....                | 124         |
| Table E.21 – Acknowledgement handler remove_ackdata_path request.....              | 125         |
| Table E.22 – State-event matrix for producing I/O application object.....          | 130         |
| Table E.23 – State-event matrix for acknowledgement handler object.....            | 131         |
| Table F.1 – Class ID ranges .....                                                  | 133         |
| Table F.2 – Service code ranges.....                                               | 133         |
| Table F.3 – Attribute ID ranges .....                                              | 134         |
| Table F.4 – MAC ID range .....                                                     | 134         |
| Table F.5 – Device type ranges .....                                               | 134         |

NOTES

## STANDARDS AUSTRALIA/STANDARDS NEW ZEALAND

**Australian/New Zealand Standard****Low-voltage switchgear and controlgear—Controller-device interfaces (CDIs)****Part 3: DeviceNet**

Any IEC table, figure or passage of text that is struck-through is not part of this Standard. Any Australian/New Zealand table, figure or passage of text that is added (and identified by shading) is part of this Standard.

**1 Scope**

This International Standard specifies an interface system between a single controller or multiple controllers, and control circuit devices or switching elements. The interface system uses two twisted shielded conductor pairs within one cable – one of these pairs provides a differential communication medium and the other pair provides power to the devices. This part establishes requirements for the interchangeability of components with such interfaces.

This standard specifies the following particular requirements for DeviceNet:

- requirements for interfaces between controllers and switching elements;
- normal service conditions for devices;
- constructional and performance requirements;
- tests to verify conformance to requirements.

These particular requirements apply in addition to the general requirements of IEC 62026-1.

**2 Normative references**

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 62026. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 62026 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

References to International Standards that are struck through in this Clause are replaced by references to equivalent Australian or Australian/New Zealand Standards that are listed immediately thereafter and identified by shading. Any Australian or Australian/New Zealand Standard that is identical to the International Standard it replaces is appropriately identified.

CISPR 11:1997, *Industrial, scientific and medical (ISM) radio-frequency equipment – Electromagnetic disturbance characteristics – Limits and methods of measurement*

~~IEC 60529:1989, *Degrees of protection provided by enclosures (IP-code)*~~

AS 1939, *Degrees of protection provided by enclosures for electrical equipment (IP code)* (identical to IEC 60529:1989)

~~IEC 60947-1:1999, *Low-voltage switchgear and controlgear—Part 1: General rules*~~

AS/NZS 3947.1, *Low-voltage switchgear and controlgear, Part 1: General rules*

~~IEC 60947-5-2:1997, *Low-voltage switchgear and controlgear—Part 5-2: Control circuit devices and switching elements—Proximity switches*~~

AS/NZS 3947.5.2, *Low-voltage switchgear and controlgear, Part 5.2: Control circuit devices and switching elements—Proximity switches* (identical to IEC 60947-5-2:1997 including Amd.1:1999)

IEC 61000-4-2:1995, *Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 2: Electrostatic discharge immunity test*. Basic EMC publication

~~IEC 61000-4-3:1995, *Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 3: Radiated, radio-frequency, electromagnetic field immunity test*~~

AS/NZS 61000.4.3, *Electromagnetic compatibility (EMC), Part 4.3: Testing and measurement techniques—Radiated radio-frequency electromagnetic field immunity test* (identical to IEC 61000-4-3:1995 including Amd.1:1998)

IEC 61000-4-4:1995, *Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 4: Electrical fast transient/burst immunity test*. Basic EMC publication

~~IEC 61000-4-5:1995, *Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 5: Surge immunity tests*~~

AS/NZS 61000.4.5, *Electromagnetic compatibility (EMC), Part 4.5: Testing and measurement techniques—Surge immunity tests* (identical to IEC 61000-4-5:1995 including Corrig.1:1995)

~~IEC 61000-4-6:1996, *Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 6: Immunity to conducted disturbances, induced by radio-frequency fields*~~

AS/NZS 61000.4.6, *Electromagnetic compatibility (EMC), Part 4.6: Testing and measurement techniques—Immunity to conducted disturbances, induced by radio-frequency fields* (identical to IEC 61000-4-6:1996)

~~IEC 61131-3:1993, *Programmable controllers—Part 3: Programming languages*~~

AS 4168.3, *Programmable controllers, Part 3: Programming languages* (identical to IEC 61131-3:1993)

~~IEC 62026-1:2000, *Low-voltage switchgear and controlgear—Controller-device interfaces (CDIs)—Part 1: General rules*~~

AS/NZS 62026.1, *Low-voltage switchgear and controlgear—Controller-device interfaces (CDIs), Part 1: General rules* (identical to IEC 62026-1:2000)

~~ISO/IEC 7498-1:1994, *Information technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*~~

AS/NZS 2777.1, *Information processing systems—Open systems interconnection—Basic reference model*, Part 1: *The basic model* (identical to ISO/IEC 7498-1:1994)

ISO 11898:1993, *Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication*

ANSI/B93.55M-1981 (R1988), *Hydraulic Fluid Power – Solenoid Piloted Industrial Valves – Interface Dimensions for Electrical Connectors*

### 3 Definitions and abbreviations

For the purpose of this International Standard, the definitions given in IEC 62026-1 and the following definitions and abbreviations apply.

#### 3.1

##### **acknowledged fragmentation**

fragmentation performed on an explicit message, in which the transmission of a fragment from the transmitting object is followed by the transmission of an acknowledgement by the receiving object. The reception of each fragment is acknowledged by the receiving object

#### 3.2

##### **ack status**

field within an acknowledgement/response message format that indicates whether or not an error has been encountered by the receiver of a fragmented message. This applies specifically to the DeviceNet fragmentation protocol

#### 3.3

##### **application objects**

set of object classes and their object instances that are available within the node. These objects manage and provide the exchange of data and messages across DeviceNet controller-device interfaces (CDIs) and within the DeviceNet compliant node

#### 3.4

##### **attribute**

description of an externally accessible characteristic or feature of an object. Attributes typically provide status information or govern the operation of an object

#### 3.5

##### **bit-strobe**

communication using strobing (see definition 3.13 of IEC 62026-1)

#### 3.6

##### **broadcast**

communication from one node to all other nodes

#### 3.7

##### **CAN (Controller Area Network)**

ISO specification that defines a generic physical layer and data link medium access procedure based on non-destructive bit-wise arbitration. See ISO 11898

#### 3.8

##### **CAN\_H**

positive half of the differential physical CAN signal

**3.9****CAN\_L**

negative half of the differential physical CAN signal

**3.10****client**

- a) object which uses the services of another (server) object to perform a task. See server (3.43)
- b) initiator of a message to which a server reacts

**3.11****common service**

service used by DeviceNet objects (see annex A)

**3.12****communication objects**

objects that manage and provide run-time exchange of messages across DeviceNet

**3.13****connection**

logical binding between two or more application objects. These application objects may be located at the same node or at different nodes

**3.14****connection ID (CID)**

connection identifier assigned to all transmissions that are associated with a particular connection between multiple nodes

**3.15****connection object**

object that manages the communication-specific aspects associated with connections between nodes

**3.16****consumer**

end-point of a connection that is responsible for receiving data

**3.17****destination MAC ID**

MAC ID of a node that is to receive a message

**3.18****device tap**

physical point of attachment from a DeviceNet device to a trunk cable or a drop cable

**3.19****device type**

identification of a collection of device-dependent information describing a viable combination of options selected for all layers in the communication stack

**3.20****dominant**

one of two complementary logic levels on the physical signal. The dominant level is a logical "0"

**3.21****duplicate MAC ID detection**

DeviceNet-defined protocol that ensures no two nodes on the same link are assigned the same MAC ID

**3.22****explicit messaging**

message that commands the performance of a particular task and the return of the results of the task performance to the requester

**3.23****fragmentation**

DeviceNet protocol provided by the connection object that defines a method by which data greater than eight (8) bytes in length may be transmitted

**3.24****group 2 client**

Unconnected Message Manager (UCMM) capable device that has gained ownership of the predefined master/slave connection set within a server so that it may act as the client on those connections

**3.25****group 2 only client**

device that is acting as a group 2 client to a group 2 only server. The group 2 only client provides the UCMM functionality for group 2 only servers that are allocated to it

**3.26****group 2 server**

UCMM capable device that has been configured to act as the server for the predefined master/slave identifier connections

**3.27****group 2 only server**

slave device that is UCMM incapable and uses the predefined master/slave connection set to establish communications. A group 2 only device can transmit and receive only those identifiers defined by the predefined master/slave connection set

**3.28****I/O connection**

connection between a producer and one or more consumers for the purpose of exchanging application-specific, time-critical I/O data

**3.29****I/O data**

information which is transferred between I/O points and the controllers which use and set the values

**3.30****I/O messaging**

exchange of data in a previously defined format

**3.31****isolated device**

device in which some of the components are not referenced to the *U*- of the physical layer. See non-isolated device (3.36)

**3.32****master**

node which gathers and distributes I/Os using the predefined master/slave connection set

**3.33****Medium Access Control (MAC) ID**

link address of a DeviceNet node

**3.34****multicast connection**

logical connection from one object to multiple other objects. A multicast connection allows data to be transferred in a single transaction from a producer to multiple consumers sharing the same connection

**3.35****node**

DeviceNet entity which is identified at the data link level by a unique MAC ID

NOTE Multiple DeviceNet nodes may be implemented in one device but they appear as logically distinct nodes on the DeviceNet link.

**3.36****non-isolated device**

device in which all components are referenced to the *U*– of the physical layer. See isolated device (3.31)

**3.37****object**

- a) abstract representation of a device's capabilities. Objects may be composed of any or all of the following components: 1) data (information which changes with time); 2) configuration (parameters for behaviour); 3) procedures (actions that may be performed using data and configuration)
- b) collection of related data (in the form of variables) and procedures for operating on that data

**3.38****point-to-point connection**

connection that exists between two objects only. Explicit messaging connections are always point-to-point. I/O connections may be either point-to-point or multicast. See multicast connection (3.34)

**3.39****predefined master/slave connection set**

utilisation of an explicit messaging connection to create and configure connection objects within each connection end-point. Uses the general rules as a basis for the definition of a set of connections which facilitates communications typically seen in a master/slave relationship

**3.40****producer**

end-point of a connection that is responsible for sending data

**3.41****recessive**

one of two complementary logic levels on the physical signal. The recessive level is a logical "1"

**3.42****serial number**

unique 32-bit integer assigned by each manufacturer to every DeviceNet device. The number is stored within the device as an attribute of the identity object and is unique with respect to the manufacturer

**3.43****server**

object which provides services to another (client) object. See client (3.10)

**3.44****service**

operation or function that an object performs upon request from another object

**3.45****slave**

node that returns data to its master using the predefined master/slave connection set and a communication method set by the master

**3.46****source MAC ID**

MAC ID of a node that is transmitting a message

**3.47****trigger**

service used by an application to initiate the production of data

**3.48****UCMM capable device**

device that supports the UCMM (see 3.51)

**3.49****UCMM incapable device**

device that does not support the UCMM (see 3.51)

**3.50****unconnected explicit message**

explicit message between nodes that have not yet established a connection between each other

**3.51****Unconnected Message Manager (UCMM)**

function within a node that receives and processes unconnected explicit messages

**3.52****Unsigned Short Integer (USINT)**

8-bit integer

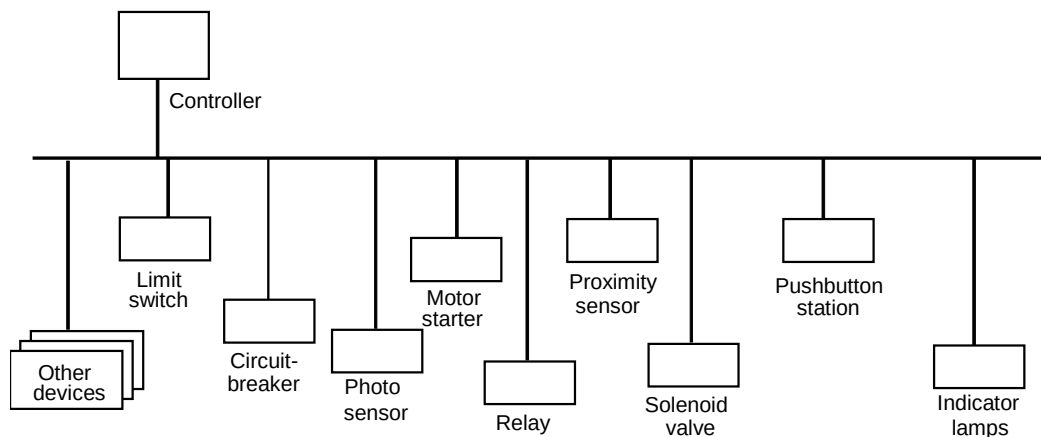
**3.53****Unsigned Integer (UINT)**

16-bit integer

## 4 Classification

### 4.1 General

DeviceNet interfaces controlling devices to control circuit devices or switching elements. DeviceNet uses two twisted shielded conductor pairs within one cable; one of these pairs provides a differential communication medium and the other pair provides power to the devices. The maximum current supported is 8 A at 24 V d.c. Data is transmitted at bit rates of 125 kbit/s, 250 kbit/s or 500 kbit/s with maximum cable lengths of 500 m, 250 m, and 100 m respectively. A maximum of eight bytes of data may be transmitted without fragmentation. A maximum of 64 nodes may be connected using a linear topology with a trunk line and drop lines (see figure 1). DeviceNet supports the transmission of I/O data, diagnostics, messaging and programming/configuration. Data exchange may be event-driven (change of state), cyclic, polled or multicast.



IEC 1204/2000

**Figure 1 – Typical DeviceNet controller-device interfaces**

This part of IEC 62026 defines a connection-based scheme to facilitate all application communications. A DeviceNet connection provides a communication path between multiple end-points. The end-points of a connection are applications that need to share data. Transmissions associated with a particular connection are assigned an identification value when a connection is established. This identification value is called connection ID (CID).

Connection objects model the communication characteristics of a particular application-to-application relationship.

The DeviceNet's connection-based scheme defines a dynamic means by which the following two types of connections may be established.

- **I/O connections:** provide dedicated, special purpose communication paths between a producing application and one or more consuming applications. Application-specific I/O data move through these paths.

I/O messages are exchanged across I/O connections. An I/O message consists of a CID and associated I/O data. The connection end-points shall have knowledge of the intended use or meaning of the I/O message.

This part of IEC 62026 does not define any particular use for I/O messaging. There are a wide variety of functions that may be accomplished using I/O messaging. The meaning and/or intended use of all I/O messages shall be made known to the system either by the particular type of product transmitting an I/O message, or based upon the configuration performed using explicit messaging.

- **Explicit messaging connections:** provide generic, multi-purpose communication paths between two devices. Explicit messages provide the typical request/response-oriented communications.

Explicit messages are exchanged across explicit messaging connections. Explicit messages are used to command the performance of a particular task and to report the results of performing the task.

DeviceNet defines an explicit messaging protocol that states the meaning of the message. An explicit message consists of a CID and associated messaging protocol information.

The rules that govern the dynamic establishment of these connections are used as a foundation upon which a predefined set of connections is defined.

## 4.2 DeviceNet communication model

The abstract object-oriented communication model of a DeviceNet node includes the following:

- **unconnected message manager (UCMM):** processes DeviceNet unconnected explicit messages;
- **identity object:** identifies and provides general information about the device;
- **connection class:** allocates and manages internal resources associated with both I/O and explicit messaging connections;
- **connection object:** manages the communication specific aspects associated with a particular application-to-application relationship;
- **DeviceNet object:** provides the configuration and status of a physical DeviceNet CDI;
- **message router:** forwards explicit request messages to the appropriate object;
- **application objects:** implement the intended purpose of the product.

## 4.3 DeviceNet and CAN

DeviceNet is based on ISO 11898 and uses the Controller Area Network (CAN) technology.

The relationships between DeviceNet, CAN and the OSI reference model (ISO/IEC 7498-1) are shown in figure 2.

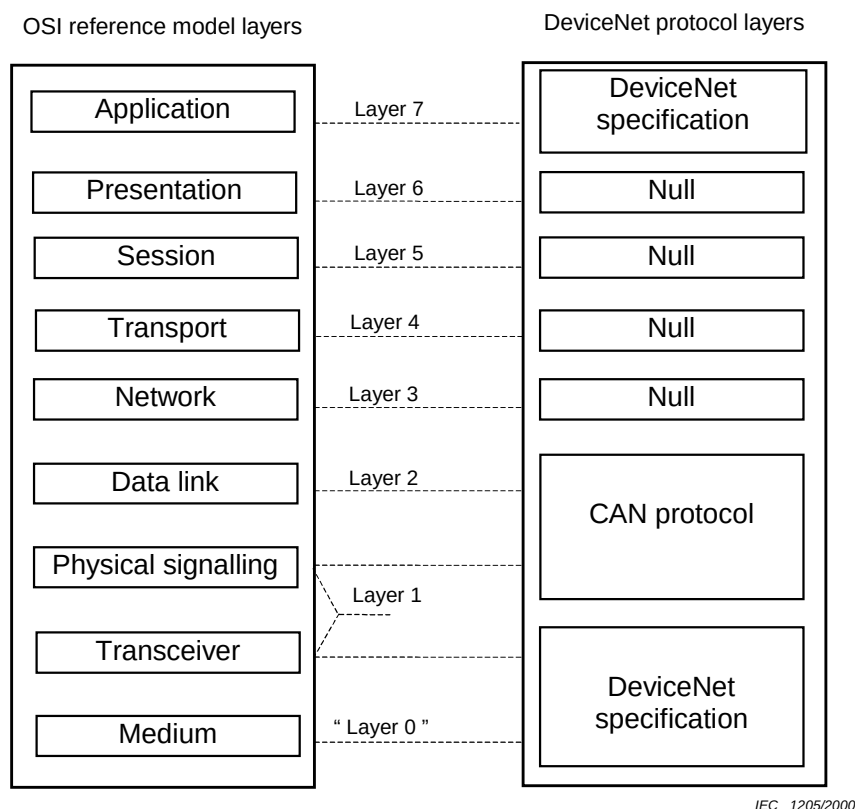


Figure 2 – DeviceNet protocol architecture compared with the OSI reference model

## 5 Characteristics

### 5.1 DeviceNet connections

#### 5.1.1 General

DeviceNet is a connection-based controller-device interface. DeviceNet connections are used to provide logical paths between multiple applications. When a connection is established, the transmissions associated with that connection are assigned a connection ID (CID). If the connection involves a bi-directional exchange, then two CID values shall be assigned to the connection.

DeviceNet uses the CAN identifier field and defines the steps involved in the dynamic establishment of I/O and explicit messaging connections.

#### 5.1.2 DeviceNet's use of the CAN identifier field

NOTE DeviceNet uses the 11 bit CAN identifier field ("Standard frame format") as described in ISO 11898.

The 11 CAN identifier bits available within DeviceNet are subdivided into four separate message groups: group 1, group 2, group 3 and group 4.

For connection-based messages, the CID is placed within the CAN identifier field. Figure 3 also describes the components of a DeviceNet CID.

| IDENTIFIER BITS |                    |                    |   |   |                             |   |                    |   |   |   | HEX RANGE | IDENTITY USAGE          |
|-----------------|--------------------|--------------------|---|---|-----------------------------|---|--------------------|---|---|---|-----------|-------------------------|
| 10              | 9                  | 8                  | 7 | 6 | 5                           | 4 | 3                  | 2 | 1 | 0 |           |                         |
| 0               | Group 1 message ID |                    |   |   | Source MAC ID               |   |                    |   |   |   | 000 - 3ff | Message group 1         |
| 1               | 0                  | MAC ID             |   |   |                             |   | Group 2 message ID |   |   |   | 400 - 5ff | Message group 2         |
| 1               | 1                  | Group 3 message ID |   |   | Source MAC ID               |   |                    |   |   |   | 600 - 7bf | Message group 3         |
| 1               | 1                  | 1                  | 1 | 1 | Group 4 message ID (0 - 2f) |   |                    |   |   |   | 7c0 - 7ef | Message group 4         |
| 1               | 1                  | 1                  | 1 | 1 | 1                           | 1 | X                  | X | X | X | 7f0 - 7ff | Invalid CAN identifiers |
| 10              | 9                  | 8                  | 7 | 6 | 5                           | 4 | 3                  | 2 | 1 | 0 |           |                         |

IEC 1206/2000

Figure 3 – DeviceNet's use of the CAN identifier field

As shown in figure 3, the CAN identifier field on DeviceNet contains the following:

- **message ID:** identifies a message within a message group inside a particular node. When a connection is established, the nodes utilise a message ID in combination with a MAC ID to generate a CID. The resulting CID is specified in the CAN identifier field associated with related transmissions;
- **source MAC ID:** groups 1 and 3 require the specification of a source MAC ID within the CAN identifier field;
- **MAC ID:** message group 2 allows the specification of either the source or the destination within the MAC ID portion of the CAN identifier field.

Both explicit messaging and I/O connections may be established in message groups 1, 2 and 3.

Group 2 message ID values 6 and 7 are used as follows:

- group 2 message ID 6 is used for the configuration of the communications utilised in a master/slave application (see 5.5);
- group 2 message ID 7 is used in the detection of nodes that have been assigned identical MAC IDs (see 5.4).

Group 3 message ID values 5, 6 and 7 are used as follows:

- group 3 message ID 5 is used when sending responses associated with unconnected explicit messaging requests;
- group 3 message ID 6 is used when sending unconnected explicit messaging requests;
- group 3 message ID 7 is invalid and shall not be used.

Group 4 is reserved for future use.

### 5.1.3 Connection establishment

#### 5.1.3.1 Explicit messaging connections and UCMM

Message group 3 defines the identifier codings to support unconnected explicit messaging. Unconnected explicit messages establish and manage explicit messaging connections. Unconnected request messages are specified by transmitting a group 3 message whose message ID component is set to 6. The only valid services that can be transmitted as unconnected explicit request messages are:

- open explicit messaging connection request;
- close connection request.

Responses to unconnected explicit requests are transmitted as unconnected response messages. Unconnected response messages are specified by transmitting a group 3 message whose message ID component is set to 5. The only valid services that can be transmitted as unconnected explicit response messages are:

- open explicit messaging connection response;
- close connection response;
- error response.

### 5.1.3.2 I/O connections

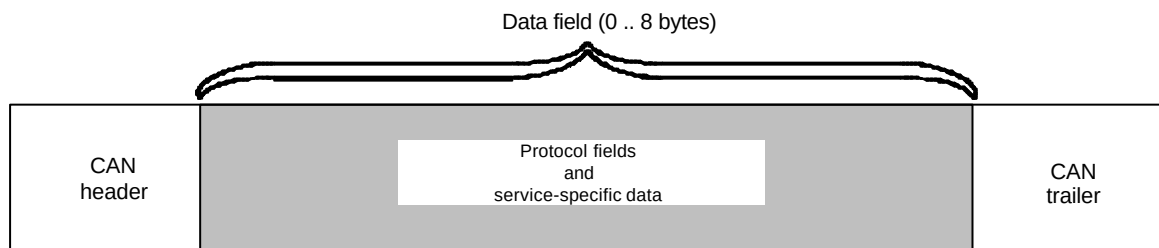
I/O connections are dynamically established by interfacing with the connection class across a previously established explicit messaging connection. A connection instance shall be created and configured at each node.

## 5.2 DeviceNet messaging protocol

### 5.2.1 Explicit messaging

#### 5.2.1.1 General

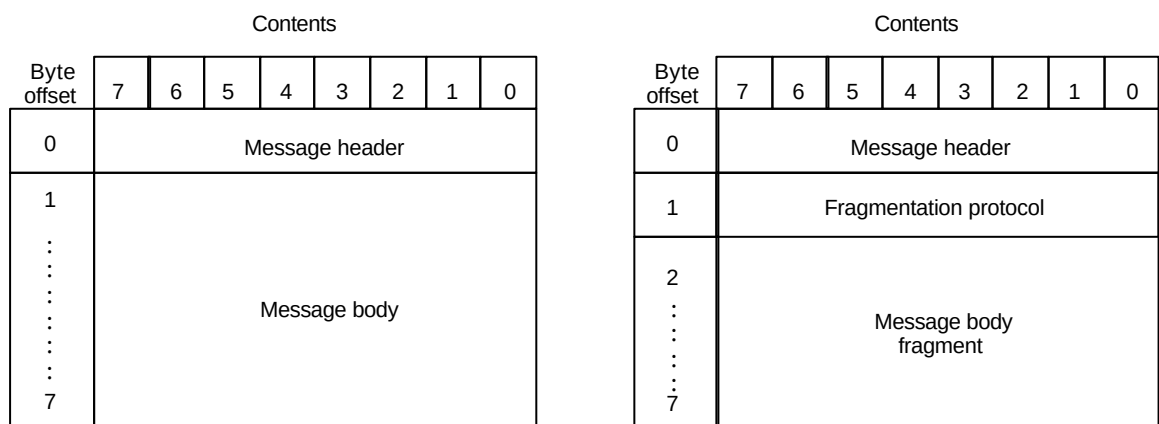
This subclause describes the explicit messaging protocol and presents details associated with the dynamic establishment of explicit messaging connections (see figure 4).



IEC 1207/2000

**Figure 4 – Explicit message CAN data field use**

Figure 5 provides the format of the CAN data field associated with explicit messages.



IEC 1208/2000

**Figure 5 – Explicit message data field format**

The data field of a transmission that contains the complete explicit message includes:

- a message header;
- the entire message body.

If the explicit message is greater than eight bytes in length, it shall be transmitted in a fragmented manner. The fragmentation/re-assembly function is provided by the connection object. A piece of a fragmented explicit message includes:

- a message header;
- the fragmentation protocol (see 5.2.3.2);
- a message body fragment.

#### 5.2.1.2 Message header

The message header is located within byte offset zero in the CAN data field of an explicit message and shall be formatted as shown in figure 6.

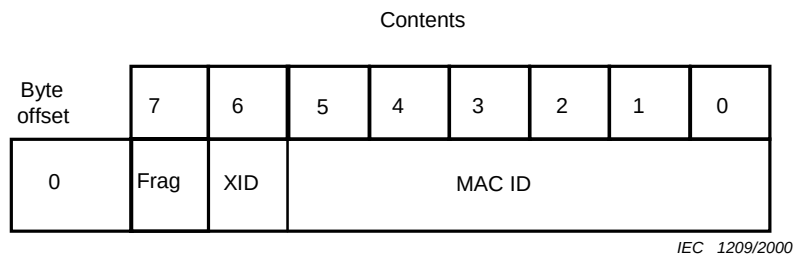


**Figure 6 – Explicit message header format**

Message header contents:

- **frag** (fragment bit): indicates whether or not this transmission is a piece of a fragmented explicit message:
  - 0 = non-fragmented
  - 1 = fragmented
- **XID (transaction ID)**: this field is utilised by an application to match a response with the associated request. This field is simply echoed by the server in a response message. The server module does not utilise this field to perform any type of duplicate message detection logic;
- **MAC ID**: contains either the source or destination MAC ID.

When an explicit message is received, the MAC ID field within the message header is examined. If the destination MAC ID is specified in the CID, then the source MAC ID of the other end-point shall be specified in the message header. If the source MAC ID is specified in the CID, then the receiving module's MAC ID shall be specified in the message header. If neither of these conditions are true, then the message shall be discarded.

#### 5.2.1.3 Message body

The message body contains a service field and service-specific arguments.

The first argument specified within the message body is the service field, which identifies the particular request or response being delivered. Figure 7 illustrates the format of the service field.

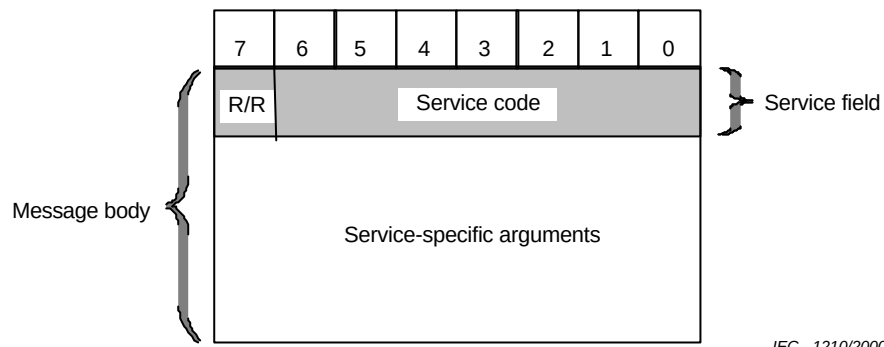


**Figure 7 – Service field format**

Service field contents:

- **service code:** the value specified within the least significant seven bits of the service field byte that indicates the type of service being transmitted;
- **R/R:** the most significant bit in the service field. Its value determines whether a message is a request or a response:

0 = request

1 = response

DeviceNet defines a set of common services. DeviceNet common services are the open set whose parameters and required behaviours are defined in this standard (see annex A).

The information following the service field is specific to the particular type of service being transmitted.

#### 5.2.1.4 Fragmentation protocol

If the transmission is a piece of a fragmented explicit message, then the data field contains the message header, the fragmentation protocol and a message body fragment. The fragmentation protocol facilitates the fragmentation and reassembly of explicit messages which have a message body greater than eight bytes (see 5.2.3).

#### 5.2.1.5 UCMM services

##### 5.2.1.5.1 General

The Unconnected Message Manager (UCMM) provides for the dynamic establishment of explicit messaging connections. This subclause presents a detailed description of the service specific arguments associated with the open and close explicit messaging connection services provided by the UCMM. See 9.3.3 for the test specifications regarding the establishment of explicit messaging connections.

The UCMM processes two services which manage the allocation and deallocation of explicit messaging connections:

- **open explicit messaging connection:** service code = 4B<sub>hex</sub>. Used to establish an explicit messaging connection;
- **close connection:** service code = 4C<sub>hex</sub>. Used to delete a connection object and deallocate all associated resources.

These services are accessed using the unconnected explicit request and response CAN identifier fields shown in figure 3. When processing an unconnected explicit request, the UCMM may need to return an error indication to the requestor and, as such, the error response explicit message may be transmitted with the unconnected explicit response CAN identifier.

#### 5.2.1.5.2 Open explicit messaging connection request

This service requests the establishment of a logical connection between two nodes across which explicit messages will be transmitted. This service is transmitted as an unconnected request message (message group 3, message ID 6) (see figure 8).

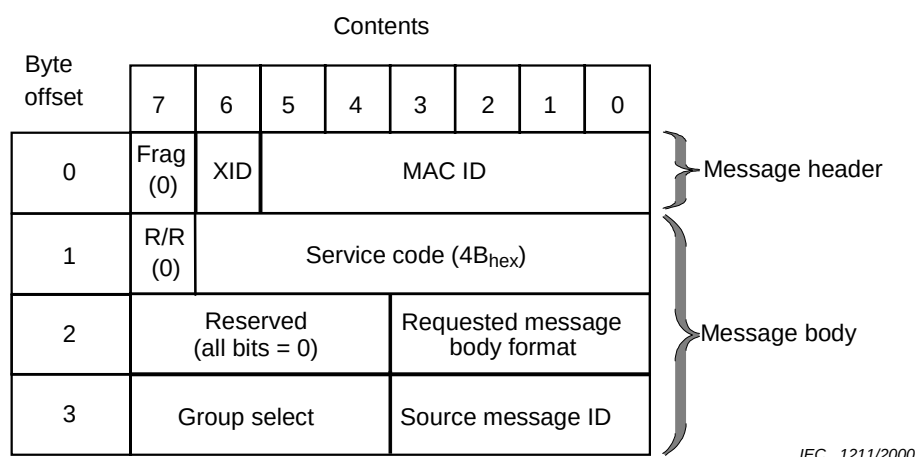


**Figure 8 – Open explicit messaging connection request format**

Open explicit messaging connection request contents:

- **frag (0)/transaction ID/MAC ID:** see 5.2.1.2. The destination MAC ID is always specified in the message header associated with an open explicit messaging connection request/response;
- **R/R bit (0):** indicates this is a request message;
- **service code (4B<sub>hex</sub>):** identifies this message as an open explicit messaging connection service;
- **reserved bits:** these shall be set to zero by the transmitter;
- **requested message body format:** the field used by the client to request a particular message body format for subsequent explicit messages transmitted over this connection.

The server node responding to this open explicit messaging request defines the message body format to be used over this connection. See table 1 for message body format values. Servers shall do one of the following:

- refuse the request and return the appropriate format value in the open explicit messaging connection response;
- accept this request by echoing the same format value in the open explicit messaging connection response;

**Table 1 – Message body format values**

| Value                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Meaning                                                                               |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                          | DeviceNet (8/8). Class = 8-bit integer, instance ID = 8-bit integer <sup>1)</sup>     |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                          | DeviceNet (8/16). Class = 8-bit integer, instance ID = 16-bit integer <sup>2)</sup>   |
| 2                                                                                                                                                                                                                                                                                                                                                                                                                                                          | DeviceNet (16/16). Class = 16-bit integer, instance ID = 16-bit integer <sup>3)</sup> |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                          | DeviceNet (16/8). Class = 16-bit integer, instance ID = 8-bit integer <sup>4)</sup>   |
| 4 – F <sub>hex</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                       | Reserved                                                                              |
| NOTE Messages transmitted across this connection are formatted as described in 5.2.1.6.                                                                                                                                                                                                                                                                                                                                                                    |                                                                                       |
| <sup>1)</sup> The class and instance ID fields are specified as 8-bit integers (USINT).<br><sup>2)</sup> The class ID is specified as an 8-bit integer (USINT) and the instance ID is specified as a 16-bit integer (UINT).<br><sup>3)</sup> The class and instance ID fields are specified as 16-bit integers (UINT).<br><sup>4)</sup> The class ID is specified as a 16-bit integer (UINT) and the instance ID is specified as an 8-bit integer (USINT). |                                                                                       |

- **group select:** the field that indicates the message group across which messages associated with this connection are to be exchanged. The group select values are shown in table 2;

**Table 2 – Group select values**

| Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Meaning                       |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Message group 1               |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Message group 2 <sup>1)</sup> |
| 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Reserved                      |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Message group 3               |
| 4 – F <sub>hex</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Reserved                      |
| <sup>1)</sup> The message group 2 identifier allows for specification of either the source or destination MAC ID. For explicit messaging connections established across message group 2, the client places the MAC ID of the server in the connection ID when transmitting messages across this connection. The server places its own MAC ID in the connection ID when transmitting messages across this connection. This process requires the server to allocate two separate message IDs from its group 2 pool. |                               |

The client selects the message group across which the transmissions associated with this explicit messaging connection will take place. If the server cannot satisfy the request, then it shall reject the request and return an error response;

- **source message ID:** the use of this field depends on the value within the group select field (see table 3):

**Table 3 – Source message ID in open explicit messaging connection request**

| If group select equals:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Then the source message ID:                                                                                                                                                                                                                                               |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 or 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Specifies the message ID the client has allocated from its group 1 or 3 message ID pool. The client shall use this message ID in combination with its own MAC ID to generate the connection ID specified when it transmits a message across this connection <sup>1)</sup> |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Is ignored/set to the value zero (0) <sup>2)</sup>                                                                                                                                                                                                                        |
| <sup>1)</sup> The client places this value within the message ID component of the message group 1 or 3 Identifier.<br><sup>2)</sup> Explicit messaging connections established across message group 2 require the server to allocate two group 2 message IDs and return them in the open explicit messaging connection response message. The client shall use one of these message IDs to generate the connection ID it specifies when transmitting a message across this connection. The other shall be used by the server to generate the connection ID it specifies when transmitting a message across this connection. |                                                                                                                                                                                                                                                                           |

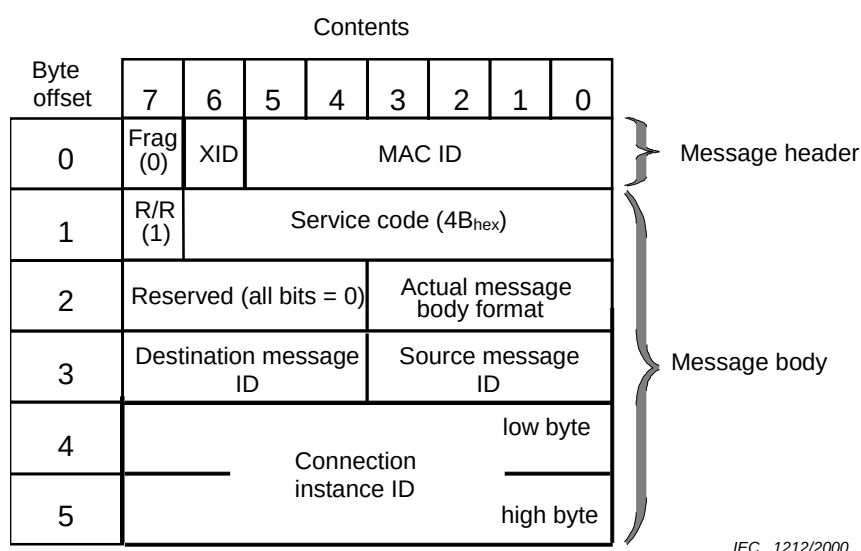
The UCMM within the server validates the open explicit messaging connection request arguments. If they are valid, the UCMM invokes the create service of the connection class to obtain a connection object instance (see 5.3.2.6). The resulting connection object is automatically configured to be an explicit messaging connection object.

If the server supports multiple message body formats and the client has requested one of those formats, then the server echoes the requested message body format in the open explicit response message. If the server does not support multiple message body formats, then the server specifies its default format in the open explicit messaging connection response.

If no errors are detected, then an open explicit messaging connection success response is returned. If an error is detected, then an error response message is returned.

#### 5.2.1.5.3 Open explicit messaging connection success response

This service is used to respond to a successful open explicit messaging connection request message (see figure 9).

**Figure 9 – Open explicit messaging connection response format**

Open explicit messaging connection response contents:

- **frag (0)/transaction ID/MAC ID:** see 5.2.1.2. The destination MAC ID is always specified in the message header associated with an open explicit messaging connection request/response;
- **R/R bit (1):** indicates that this is a response message;
- **service code (4B<sub>hex</sub>):** identifies this message as an open explicit messaging connection service;
- **reserved bits:** these shall be set to zero by the transmitter;
- **actual message body format:** the field used by the server to define the format of the message body associated with subsequent explicit messages transmitted over this connection (as described in table 1);
- **destination message ID:** the use of this field depends on the message group across which the connection takes place (as described in table 4);

**Table 4 – Destination message ID in an open explicit messaging connection response**

| If group select within the open request was set to: | Then the destination message ID in the open response:                                                                                             |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 or 3                                              | Is ignored and shall be set to the value zero (0)                                                                                                 |
| 1                                                   | Is used by the client in combination with the server's MAC ID to generate the connection ID it specifies when transmitting across this connection |

- **source message ID:** the message ID value that the server has allocated. The server allocates a message ID from its group 1, 2, or 3 message ID pool that is used in conjunction with its own MAC ID (source MAC ID) to generate the connection ID that is specified when it transmits a message across this connection;
- **connection instance ID:** the server creates an explicit messaging connection object when it successfully services an open request. This field holds the instance ID value (16-bit integer field) assigned to that explicit messaging connection object.

#### 5.2.1.5.4 Close connection request

This service is used to terminate a connection (either I/O or messaging) within one of the end points. The reception of the close connection request message by the UCMM results in the invocation of the connection class's delete service (see 5.3.2.6). A close connection request message is transmitted as an unconnected request message.

The responder verifies that the specified connection instance exists. If the connection instance exists and can be deleted, then it is deleted. All resources associated with the connection instance are freed. If the request is successfully serviced, then a close connection response is returned. If the request is not successful, an error response is returned.

Close connection request contents (see figure 10):

- **frag (0)/transaction ID/MAC ID:** see 5.2.1.2;
- **R/R bit (0):** indicates that this is a request message;
- **service code (4C<sub>hex</sub>):** identifies that this is a close connection service;
- **connection instance ID:** the field that specifies the connection instance to be deleted. The format for the connection instance ID within this message is always specified as a 16-bit integer.

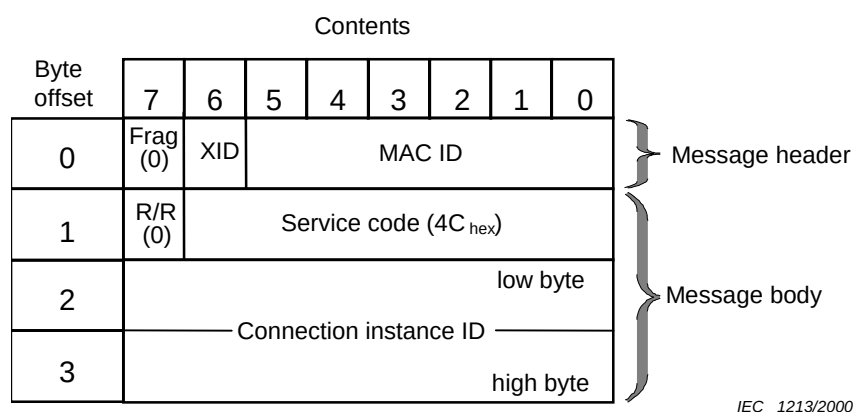


Figure 10 – Close connection request format

### 5.2.1.5.5 Close connection response

This service is used to respond to a successful close connection request message.

Close connection response format contents (see figure 11):

- **frag (0)/transaction ID/MAC ID:** see 5.2.1.2;
- **R/R bit (1):** indicates that this is a response message;
- **service code (4C<sub>hex</sub>):** identifies this message as a close connection service.

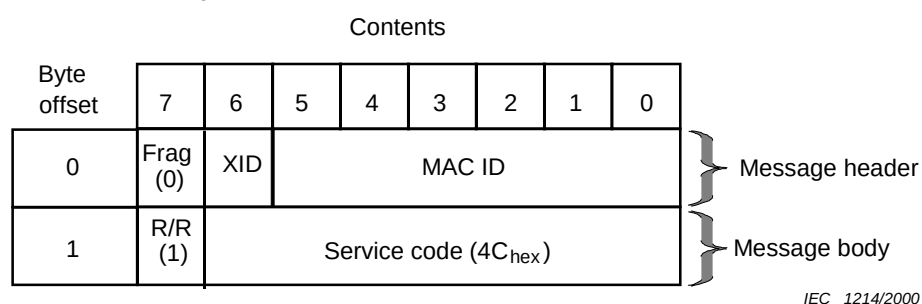


Figure 11 – Close connection response format

### 5.2.1.5.6 Error response

Table 5 shows a standard set of UCMM related error conditions and the error code (both general error code and additional error code) information to be used in an associated error response message. The format of an error response message is described in 5.2.1.7.

Table 5 – UCMM error conditions/codes

| Error condition                    | General error name    | General error code (hex) | Additional error code (hex) |
|------------------------------------|-----------------------|--------------------------|-----------------------------|
| Service code not open or close     | Service not supported | 08                       | FF                          |
| Group select resource error        | Resource unavailable  | 02                       | 01                          |
| Group select out of range          | Invalid parameter     | 20                       | 01                          |
| No server connections available    | Resource unavailable  | 02                       | 02                          |
| No server message IDs available    | Resource unavailable  | 02                       | 03                          |
| Client source message ID invalid   | Invalid parameter     | 20                       | 02                          |
| Duplicate client source message ID | Resource unavailable  | 02                       | 04                          |
| Connection instance ID invalid     | Object does not exist | 16                       | FF                          |

Error condition description:

- **service not open or close:** a service received across the UCMM port is not of type open or close and is therefore not supported by UCMM;
- **group select resource error:** the group select argument indicates utilising a message group which is not supported by the device;
- **group select out of range:** the group select field contains an invalid value;
- **no server connections available:** the maximum number of connections supported by this server has already been reached;
- **no server message IDs available:** the server has allocated all message IDs within the message group requested by the client;
- **client source message ID invalid:** the source message ID received with an open explicit message connection request is invalid for the specified message group;
- **duplicate client source message ID:** the source message ID and source MAC ID received within an open explicit messaging connection request are already in use for a group 1 or group 3 explicit messaging connection;
- **connection instance ID invalid:** the connection instance ID received with the close connection request does not exist.

#### 5.2.1.6 Connection-based explicit messaging

A connection-based explicit message is a message transmitted over an explicit messaging connection. A connection-based explicit message shall conform to the formats described in this subclause.

Figure 12 shows the format for the message associated with a non-fragmented explicit request.

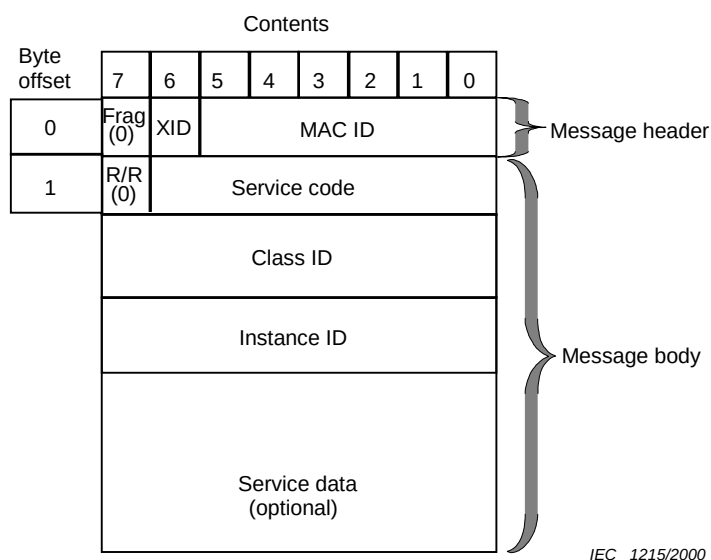


Figure 12 – Non-fragmented explicit request message format

Non-fragmented explicit request message contents:

- **frag (0)/transaction ID/MAC ID:** see 5.2.1.2;
- **R/R bit (0):** indicates that this is a request message;
- **service code:** defines the service being requested;

COPYRIGHT

- **class ID:** defines the object class towards which this request is directed. The class ID is specified within either an 8- or 16-bit integer field based on the actual message body format value returned in the open explicit messaging connection response;
- **instance ID:** defines the particular instance within the object class towards which this request is directed. The instance ID is specified within either an 8- or 16-bit integer field based on the actual message body format value returned in the open explicit messaging connection response. DeviceNet uses the value zero to denote that the request is directed towards the class itself rather than a specific instance within the class;
- **service data:** carries request-specific data. Formats for the DeviceNet common services are described in annex A. Class-specific and object-specific service definitions include the format of this field.

Figure 13 shows the format for the message associated with a non-fragmented success response:

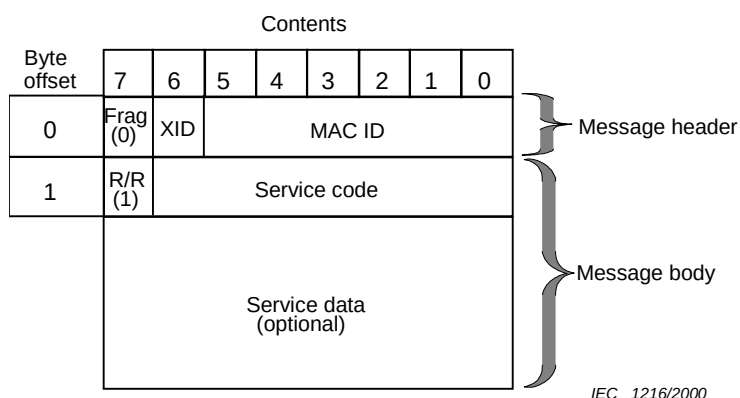


**Figure 13 – Non-fragmented success response message format**

Non-fragmented success response message contents:

- **frag (0)/transaction ID/MAC ID:** see 5.2.1.2;
- **R/R bit (1):** indicates that this is a response message;
- **service code:** contains the service code sent in the request message;
- **service data:** carries request-specific data.

#### 5.2.1.7 Error response message

The error response message is returned when an error is encountered while attempting to service a previously received explicit request message. The error response may be sent as either a connection-based (request was received across an explicit messaging connection) or unconnected (request was an unconnected explicit request message) response message. Figure 14 shows the format of an error response message.

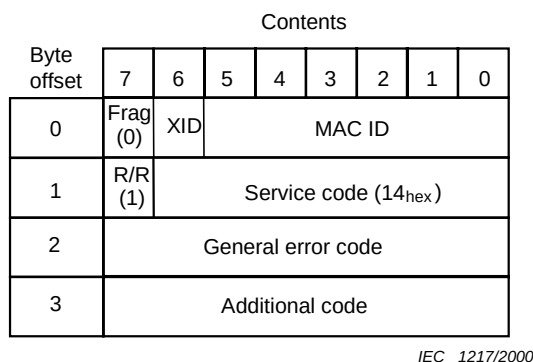


**Figure 14 – Error response message**

Error response message contents:

- **frag (0)/transaction ID/MAC ID:** see 5.2.1.2;
- **R/R bit (1):** indicates that this is a response message;
- **service code (14<sub>hex</sub>):** identifies this message as an error response;
- **general error code:** identifies the encountered error. See annex B for a list of general error codes;
- **additional code:** contains an object-specific or service-specific value that further describes the error condition. If the responding object has no additional information to specify, then the value FF<sub>hex</sub> is placed within this field.

### 5.2.2 Input/output messaging

DeviceNet defines a fragmentation protocol for transfer of an I/O message greater than eight bytes in length. This is the only protocol information carried within the data field of an I/O message (see figure 15).

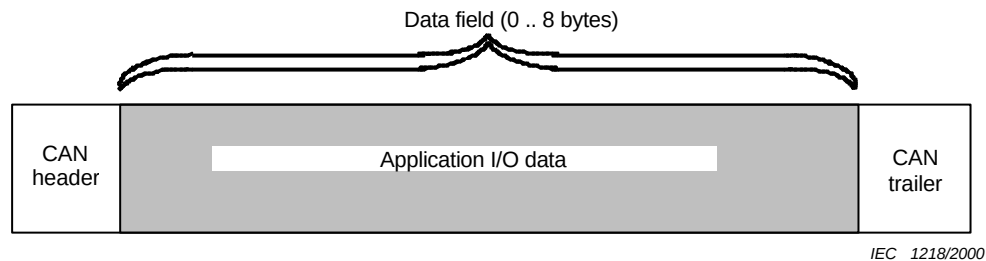


Figure 15 – Data field of an I/O message

### 5.2.3 Fragmentation/reassembly

#### 5.2.3.1 General

This subclause defines the means by which a message whose length is greater than eight bytes is fragmented and reassembled. The fragmentation/reassembly function is provided by the DeviceNet connection object.

The logic that triggers a fragmented transmission is:

- explicit messaging connection object instances examine the length of each message to be transmitted. If the message is greater than eight bytes in length, then the fragmentation protocol is used;
- I/O connection object instances examine the produced\_connection\_size attribute of the connection object (see 5.3.2). If the produced\_connection\_size attribute is greater than eight, then the fragmentation protocol is used.

Two types of fragmentation are defined:

- **acknowledged:** performed when fragmenting an explicit message;
- **unacknowledged:** performed when fragmenting an I/O message.

#### 5.2.3.2 Fragmentation protocol

The fragmentation protocol is located within a single byte in the CAN data field and is formatted as shown in figure 16.

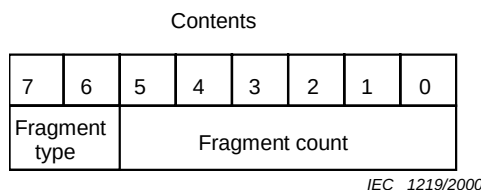


**Figure 16 – Format of DeviceNet fragmentation protocol**

Fragmentation protocol contents:

- **fragment type:** indicates whether this is the first fragment, one of the middle fragments or the last fragment (see table 6);

**Table 6 – Fragment type bit values**

| Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Meaning                                                                                |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | First fragment. The fragment count field shall contain the value 0 or 3F <sup>1)</sup> |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Middle fragment <sup>2)</sup>                                                          |
| 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Last fragment <sup>3)</sup>                                                            |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Fragment acknowledgement <sup>4)</sup>                                                 |
| <sup>1)</sup> If the fragment count contains the value zero (0), then this is the first in a series of fragments. If the fragment count field contains the value 3F, then this is also the last transmission in the series.<br><sup>2)</sup> This fragment is neither the first nor the last fragment in the series.<br><sup>3)</sup> Marks this as the last fragment.<br><sup>4)</sup> The value the receiver of a fragmented message uses to acknowledge the reception of a fragment. |                                                                                        |

- **fragment count:** marks each separate fragment such that the receiver may determine whether or not a fragment has been missed. If the fragment type is the first fragment, then this field takes on a special meaning (as described in table 6). The fragment count is increased by one for each successive fragment in a series and resets to zero at overflow ( $\text{fragmentcount} = (\text{fragmentcount} + 1) \bmod 64$ ).

For I/O message fragmentation, the fragmentation protocol information is placed within byte offset 0 (see figure 17).

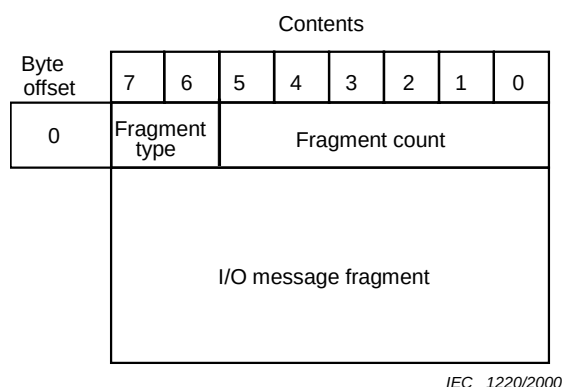


**Figure 17 – I/O message fragment format**

For explicit message fragmentation, the fragmentation protocol information is placed within byte offset 1 (see figure 18).

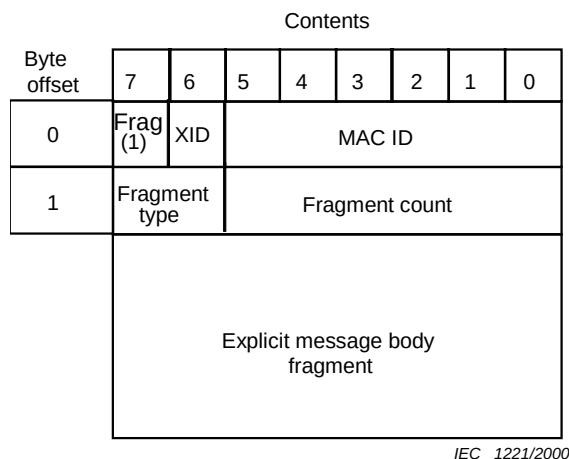


**Figure 18 – Explicit message fragment format**

The frag bit within the message header in figure 18 is set to 1 to indicate that this is a piece of the explicit message, not the entire message. The value 1 here also indicates that the next byte contains the fragmentation protocol.

The receiver of a fragmented series of transmissions parses the fragmented message as defined in this subclause. This procedure applies to both I/O and explicit message fragmentation.

If the first transmission is expected by the connection and the fragment type is equal to the first fragment:

- if the fragment count is  $3F_{\text{hex}}$ , then this is the only transmission in the series, and the connection processes the message and awaits the beginning of a new series;
- if the fragment count is 0, then this is the first in a series of transmissions and the connection stores the fragment and saves the fragment count.

If the first transmission is expected by the connection and either the fragment type is not equal to the first fragment or the fragment count is not equal to 0 or  $3F_{\text{hex}}$ , the connection discards the transmission and awaits the beginning of a new message.

If the first transmission is not expected by the connection, then the connection verifies:

- that the fragment type is not the first fragment, and
- that the fragment count is numerically one (1) greater than the previous value received.

If one of these checks fails, then an error has been detected. If both checks pass, then the fragment is appended to the previously received fragment(s), and the fragment type is parsed to determine whether or not more fragments are to be expected.

If more fragments are forthcoming, then the connection saves the received fragment count and waits for the next fragment. If this is the last transmission in the series and an error has not been detected, then the connection processes the message and resets to looking for the beginning of a new series.

If an explicit message is being fragmented, then the receiver shall generate and transmit an acknowledgement after the reception of each fragment.

If an error is detected, then an error recovery specific to whether this is a fragmented I/O or explicit message takes place.

If the detection of a missed fragment was triggered by the reception of the first fragment in the next series, then any processing associated with the current series is immediately discontinued, the fragments stored in memory are discarded and processing immediately begins on the new series.

### 5.2.3.3 Unacknowledged fragmentation

The fragmentation of an I/O message is performed in an unacknowledged fashion. Unacknowledged fragmentation consists of the back-to-back transmission of the fragments from the transmitting module. The receiving module(s) returns no acknowledgements.

When an I/O connection's `send_message` service is invoked, it examines its `connection_size` attribute to determine whether or not a fragmented series of messages is to be transmitted. If the `connection_size` attribute is greater than eight (8), then the fragmentation protocol is placed within the I/O.

If the `connection_size` attribute is less than or equal to eight (8) bytes, then the raw data is transmitted without the presence of the fragmentation protocol.

If the application requests to transmit a piece of data whose length is greater than the `connection_size` attribute, then an internal error is indicated and the transmission does not occur. If the receiving I/O connection object detects a missed fragment by determining that the received fragment count is not equal to the previously received fragment count plus one (1), then the following error recovery steps are taken:

- all subsequent fragments in this series are dropped and the application is not informed of an I/O message reception;
- the connection object begins looking for the beginning of a new fragmented series of transmissions and discards the remaining fragments in this series.

### 5.2.3.4 Acknowledged fragmentation

Acknowledged fragmentation is used for fragmented explicit messages. Acknowledged fragmentation consists of the transmission of a fragment from the transmitting node followed by the transmission of an acknowledgement by the receiving node. The receiving node acknowledges the reception of each fragment.

Figure 19 shows the format for the acknowledgement message transmitted by the receiver after each explicit message fragment is received.

| Contents    |                   |     |                |   |   |   |   |   |
|-------------|-------------------|-----|----------------|---|---|---|---|---|
| Byte offset | 7                 | 6   | 5              | 4 | 3 | 2 | 1 | 0 |
| 0           | Frag (1)          | XID | MAC ID         |   |   |   |   |   |
| 1           | Fragment type (3) |     | Fragment count |   |   |   |   |   |
| 2           | Ack status        |     |                |   |   |   |   |   |

IEC 1222/2000

**Figure 19 – Acknowledgement message format**

Acknowledgement message contents:

- **fragment type:** indicates that this is a fragment acknowledgement by placing the value 3 within this field;
- **fragment count:** echoes the last fragment count value received;
- **ack status:** indicates whether or not an error has been detected by the receiver of the fragmented message. The ack status bit values are defined in table 7.

**Table 7 – Ack status bit values**

| Value   | Meaning                                                                                                     |
|---------|-------------------------------------------------------------------------------------------------------------|
| 0       | Success. No errors have been detected and the fragmented transmission shall continue                        |
| 1       | Too much data. The maximum amount of data the receiver can receive across this connection has been exceeded |
| 2 – 255 | Reserved                                                                                                    |

The transmitting node functions as specified below:

- the connection object formulates the message header by setting the frag bit to one (1), the XID (transaction ID) field to the value specified by the application, and initialises the MAC ID field in the same manner it would a non-fragmented explicit message. The message headers associated with each separate fragment are identical;
- the connection object then places the appropriate fragmentation protocol information into the message. The connection object stores the fragment count that was inserted into the message;
- the connection object then takes the next piece of the message body and places it into the message;
- the message is transmitted and a wait for the acknowledgement timer is started. The amount of time to wait is application-specific;
- if the wait for the acknowledgement timer expires, then the connection object automatically retries the last transmission. The connection object retries once. If the timer expires once again (two consecutive wait for acknowledgement time-outs), then the application is informed that an error has been detected and the requested transmission cannot take place;
- if an acknowledgement is received, then the connection object determines whether or not the fragment count in the acknowledgement is equal to the last fragment count it transmitted. If they are equal, then the fragment was successfully delivered and acknowledged, and normal processing continues. If the values are not equal, then the connection object continues to wait for an acknowledgement with a matching fragment count.

The initial state associated with the receiving module entails waiting for either the first fragment in a fragmented transmission or for the reception of a complete explicit message. The receiving side functions as specified below:

- if the message header indicates that this is a fragmented portion of an explicit message, then the connection object examines the fragmentation protocol to determine its validity. If the connection has yet to receive the first transmission in the series (in the initial state) and the fragment type field is not equal to the first fragment, then the fragment is dropped and no acknowledgement is returned;
- if the fragment type indicates that this is the first fragment, then the fragment count shall equal zero (0). If this is the case, then the message fragment is stored and an acknowledgement is returned. If the fragment type indicates first fragment and the fragment count is not zero (0), then the fragment is dropped and no acknowledgement is returned;

- if the fragment count is one (1) greater than the previously received count and the fragment type does not indicate that this is the first fragment, then the next fragment has been received. The fragment is appended to the previously received fragment(s) and an acknowledgement is returned. The fragment count associated with this fragment is stored;
- if the fragment count is neither one (1) greater nor equal to the previously received count, then the fragment is discarded and no acknowledgement is returned. The receiver resets to the initial state;
- when the final fragment is received and the acknowledgement is transmitted, the connection object continues processing the message as if it were non-fragmented.

#### 5.2.4 Duplicate MAC ID detection protocol

Each DeviceNet node shall be assigned a MAC ID and is required to participate in the duplicate MAC ID detection algorithm, as defined in 5.4.

A special message ID value is defined within group 2 to identify the duplicate MAC ID check message (see figure 20).

| IDENTIFIER BITS |   |                    |   |   |   |   |                       |   |   |   | MESSAGE ID<br>MEANING          |
|-----------------|---|--------------------|---|---|---|---|-----------------------|---|---|---|--------------------------------|
| 10              | 9 | 8                  | 7 | 6 | 5 | 4 | 3                     | 2 | 1 | 0 |                                |
| 1               | 0 | MAC ID             |   |   |   |   | Group 2<br>message ID |   |   |   | Group 2 messages               |
| 1               | 0 | Destination MAC ID |   |   |   |   | 1                     | 1 | 1 |   | Duplicate MAC ID check message |

IEC 1223/2000

**Figure 20 – Duplicate MAC ID check CAN identifier field**

The data field associated with the duplicate MAC ID check message has the format shown in figure 21.

| Byte<br>offset | Contents        |                      |   |   |   |   |           |   |
|----------------|-----------------|----------------------|---|---|---|---|-----------|---|
|                | 7               | 6                    | 5 | 4 | 3 | 2 | 1         | 0 |
| 0              | R/R             | Physical port number |   |   |   |   |           |   |
| 1              | Manufacturer ID |                      |   |   |   |   | low byte  |   |
| 2              |                 |                      |   |   |   |   | high byte |   |
| 3              | Serial number   |                      |   |   |   |   | low byte  |   |
| 4              |                 |                      |   |   |   |   |           |   |
| 5              |                 |                      |   |   |   |   |           |   |
| 6              |                 |                      |   |   |   |   | high byte |   |

IEC 1224/2000

**Figure 21 – Duplicate MAC ID check message data field format**

Duplicate MAC ID check message data field contents:

- **R/R bit:** request/response bit. The value in this field indicates whether this is a duplicate MAC ID check request or a response message:  
     0 = request  
     1 = response
- **physical port number:** an identification value internally assigned to each physical DeviceNet port of a device. Devices that have a single port shall place the value zero (0) within this field;
- **manufacturer ID:** a 16-bit integer field (UINT) containing the identification code assigned to the manufacturer of the device that is transmitting the message;
- **serial number:** see 3.42.

### 5.3 DeviceNet communication object classes

#### 5.3.1 General

The DeviceNet communication objects manage and provide the exchange of messages. The services, attributes and behaviours associated with the main communication objects are detailed in this subclause. See annex D for a description of DeviceNet data type specification and encoding, and annex E for the definition of auxiliary communication objects.

#### 5.3.2 Connection object class definition (class ID code: 5)

##### 5.3.2.1 General

The connection class allocates and manages the internal resources associated with both I/O and explicit messaging connections. The specific instance generated by the connection class is referred to as a connection instance or a connection object.

##### 5.3.2.2 Connection object class attribute

The connection class attribute is defined in table 8.

**Table 8 – Connection class attribute**

| Attribute ID | Need in implementation | Access rule | Attribute name | Data type | Attribute description                                                              |
|--------------|------------------------|-------------|----------------|-----------|------------------------------------------------------------------------------------|
| 1            | Optional               | Get         | Revision       | UINT      | Revision of the connection class definition upon which the implementation is based |

The attribute shall be set to 1.

All other attributes are reserved.

##### 5.3.2.3 Connection object class services

The connection class supports the following DeviceNet common services (as described in table 9).

**Table 9 – Connection class services**

| Need in implementation                                                              | Service name              | Service description                                                                                                                                      |
|-------------------------------------------------------------------------------------|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Optional                                                                            | Create                    | Creates a connection object                                                                                                                              |
| Optional                                                                            | Delete                    | Deletes all connection objects and releases all associated resources. When the delete service is sent to the connection class, all instances are deleted |
| Optional                                                                            | Reset                     | Resets all resettable connection objects. The conditions under which a connection is resettable are listed in the state-event matrix in table 20         |
| Optional                                                                            | Find_next_object_instance | Searches for instance IDs associated with existing connection objects                                                                                    |
| 1)                                                                                  | Get_attribute_single      | Reads a connection class attribute value                                                                                                                 |
| 1) Get_attribute_single is required if the connection class attribute is supported. |                           |                                                                                                                                                          |

#### 5.3.2.4 Connection object instance attributes

Table 10 shows the connection instance attributes and their associated data types.

Table 10 – Connection object instance attributes

| Attribute ID (decimal) | Need in implementation | Attribute name                  | Data type      | Brief description of attribute                                                                               |
|------------------------|------------------------|---------------------------------|----------------|--------------------------------------------------------------------------------------------------------------|
| 1                      | Required               | State                           | USINT          | State of the object                                                                                          |
| 2                      | Required               | Instance_type                   | USINT          | Indicates either I/O or messaging connection                                                                 |
| 3                      | Required               | Transportclass_trigger          | BYTE           | Defines behaviour of the connection                                                                          |
| 4                      | Required               | Produced_connection_id          | UINT           | Placed in CAN identifier field when the connection transmits                                                 |
| 5                      | Required               | Consumed_connection_id          | UINT           | CAN identifier field value that denotes message to be received                                               |
| 6                      | Required               | Initial_comm_characteristics    | BYTE           | Defines the message group(s) across which productions and consumptions associated with this connection occur |
| 7                      | Required               | Produced_connection_size        | UINT           | Maximum number of bytes transmitted across this connection                                                   |
| 8                      | Required               | Consumed_connection_size        | UINT           | Maximum number of bytes received across this connection                                                      |
| 9                      | Required               | Expected_packet_rate            | UINT           | Defines timing associated with this connection                                                               |
| 10 – 11                |                        |                                 |                | Reserved                                                                                                     |
| 12                     | Required               | watchdog_timeout_action         | USINT          | Defines how to handle inactivity/watchdog time-outs                                                          |
| 13                     | Required               | produced_connection_path_length | UINT           | Number of bytes in the produced_connection_path attribute                                                    |
| 14                     | Required               | produced_connection_path        | Array of USINT | Specifies the application object(s) whose data is to be produced by this connection object                   |
| 15                     | Required               | consumed_connection_path_length | UINT           | Number of bytes in the consumed_connection_path attribute                                                    |
| 16                     | Required               | consumed_connection_path        | Array of USINT | Specifies the application object(s) that are to receive the data consumed by this connection object          |
| 17                     | Required               | production_inhibit_time         | UINT           | Defines minimum time between new data production                                                             |

The list below provides detailed descriptions of the DeviceNet connection object instance attributes. The defined default attribute values shall be used when no other internal and/or system-defined rules exist.

- 1) **state:** Defines the current state of the connection instance. Table 11 defines the possible states and assigns a value used to indicate that state.

**Table 11 – Values assigned to the state attribute**

| Value | State name                | Description                                                                                                                                                                                                                                                           |
|-------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0     | Non-existent              | No connection exists                                                                                                                                                                                                                                                  |
| 1     | Configuring               | The connection has been created and is waiting: (a) to be configured and (b) to be instructed to apply the configuration                                                                                                                                              |
| 2     | Waiting for connection ID | The connection instance is waiting for its consumed_connection_id and/or produced_connection_id attribute to be set <sup>1)</sup>                                                                                                                                     |
| 3     | Established               | The connection has been configured and the configuration has been applied                                                                                                                                                                                             |
| 4     | Timed-out                 | If a connection object experiences an inactivity/watchdog time-out, then a transition shall be made to this state. See the watchdog_timeout_action attribute description (table 17) and the description of the inactivity/watchdog timer (5.3.2.5.3) for more details |
| 5-255 | Reserved                  |                                                                                                                                                                                                                                                                       |

<sup>1)</sup> When the connection instance attributes are applied, it may not be possible for the node to generate the produced\_connection\_id and/or consumed\_connection\_id values (see initial\_comm\_characteristics attribute in table 15 for rules). If this is the case, then all the tasks required to apply the attributes are performed except for the initialisation of these attributes within the connection and associated link producer/consumer objects, and the connection instance assumes the Waiting for connection ID state

All resources associated with a connection instance that has been dynamically created across an explicit messaging connection shall be released if the explicit messaging connection breaks down before the dynamically created connection instance has transitioned to the established state.

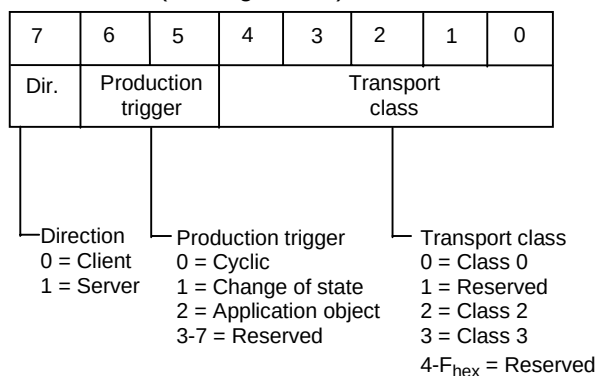
When a transition is made to the established state, all timers associated with the connection object shall be activated (see 5.3.2.5).

- 2) **instance\_type**: Defines the instance type (as described in table 12).

**Table 12 – Values assigned to the instance\_type attribute**

| Value | Meaning            |
|-------|--------------------|
| 0     | Explicit messaging |
| 1     | I/O                |
| 2-255 | Reserved           |

- 3) **transportclass\_trigger**: Identifies whether this is a producing only, consuming only, or both producing and consuming connection. If this end-point is to perform a data production, this attribute also identifies the event that triggers the production. The eight bits are divided as follows (see figure 22).



IEC 1225/2000

**Figure 22 – Transportclass\_trigger**

- 4) **produced\_connection\_id:** contains the CID to be associated with transmissions sent across this connection (if any). This value shall be specified in the CAN identifier field when this connection transmits (see 5.1.2).

**Table 13 – Values defined for the produced\_connection\_id attribute**

| Value                                    | Meaning                                                                                                                                                                         |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 – 7F0 <sub>hex</sub>                   | The value to be placed in the CAN identifier field when this connection transmits                                                                                               |
| 7F1 <sub>hex</sub> – FFFE <sub>hex</sub> | Reserved                                                                                                                                                                        |
| FFFF <sub>hex</sub>                      | Default value assigned to this attribute within an I/O connection. This attribute shall retain this value if this connection instance is not producing any data (consumer only) |

- 5) **consumed\_connection\_id:** Contains the CID which identifies messages to be received across this connection (if any). This is the CAN identifier field value that is used with messages to be received by this connection object (see 5.1.2). The following values are defined (see table 14).

**Table 14 – Values defined for the consumed\_connection\_id attribute**

| Value                                    | Meaning                                                                                                                                                                         |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 – 7F0 <sub>hex</sub>                   | The value that identifies messages to be consumed by this connection object instance. This shall be specified in the CAN identifier field of messages that are to be consumed   |
| 7F1 <sub>hex</sub> – FFFE <sub>hex</sub> | Reserved                                                                                                                                                                        |
| FFFF <sub>hex</sub>                      | Default value assigned to this attribute within an I/O connection. This attribute shall retain this value if this connection instance is not consuming any data (producer only) |

- 6) **initial\_comm\_characteristics:** Defines the message group(s) across which this connection produces and consumes. This byte is divided into two nibbles (see figure 23).

|                                    |   |   |   |                                     |   |   |   |
|------------------------------------|---|---|---|-------------------------------------|---|---|---|
| 7                                  | 6 | 5 | 4 | 3                                   | 2 | 1 | 0 |
| Initial production characteristics |   |   |   | Initial consumption characteristics |   |   |   |

IEC 1226/2000

**Figure 23 – Initial\_comm\_characteristics attribute format**

Table 15 lists the values that are possible within the initial production characteristics nibble (upper nibble) of the initial\_comm\_characteristics attribute.

**Table 15 – Values for the initial production characteristics nibble**

| Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Meaning                                                    |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Produce across message group 1 <sup>1)</sup>               |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Produce across message group 2 (destination) <sup>2)</sup> |
| 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Produce across message group 2 (source) <sup>3)</sup>      |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Produce across message group 3 <sup>4)</sup>               |
| 4 – E <sub>hex</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Reserved                                                   |
| F <sub>hex</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Default value <sup>5)</sup>                                |
| <p><sup>1)</sup> The producing node generates the CID value and loads it into the connection object's produced_connection_id attribute. The producing node allocates a message ID from its group 1 message_ID pool and combines this with its source MAC ID to generate the CID. The numerically lowest available group 1 message ID shall be used in generating the produced_connection_id attribute value. This value shall also be loaded into the corresponding consumed_connection_id attribute associated with the consuming connection object(s).</p> <p><sup>2)</sup> The destination MAC ID shall be placed within the MAC ID component of the group 2 identifier field. In this case, the consuming node generates the CID value to be associated with transmissions across this connection. When the consuming node has generated this value and loaded it into the appropriate connection object's consumed_connection_id attribute, this value shall be read and subsequently loaded into the producing connection object's produced_connection_id attribute.</p> <p><sup>3)</sup> The source MAC ID shall be placed within the MAC ID component of the group 2 identifier. In this case, the producing node generates the CID value and loads it into the connection object's produced_connection_id attribute. The numerically lowest available group 2 message ID shall be used in generating the produced_connection_id attribute value. This value shall also be loaded into the corresponding consumed_connection_id attribute associated with the consuming connection object(s).</p> <p><sup>4)</sup> The producing node generates the CID value and loads it into the connection object's produced_connection_id attribute. The producing node allocates a message ID from its group 3 message ID pool and combines this with its source MAC ID to generate the CID. The numerically lowest available group 3 message ID shall be used in generating the produced_connection_id attribute value. This value shall also be loaded into the corresponding consumed_connection_id attribute associated with the consuming connection object(s).</p> <p><sup>5)</sup> The default value assigned to the initial production characteristics nibble within an I/O connection. If this is a consuming only I/O connection, then the default value remains in this nibble. Explicit messaging connection objects automatically configure this attribute when the connection is established.</p> |                                                            |

Table 16 lists the possible values within the initial consumption characteristics nibble (lower nibble) of the initial\_comm\_characteristics attribute.

**Table 16 – Values for the initial consumption characteristics nibble**

| Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Meaning                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Consume a group 1 message <sup>1)</sup>               |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Consume a group 2 message (destination) <sup>2)</sup> |
| 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Consume a group 2 message (source) <sup>3)</sup>      |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Consume a group 3 message <sup>4)</sup>               |
| 4 – E <sub>hex</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Reserved by DeviceNet                                 |
| F <sub>hex</sub>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Default value <sup>5)</sup>                           |
| <p><sup>1)</sup> The producing node generates the CID value. This value shall be loaded into the consumed_connection_id attribute associated with the consuming connection object(s).</p> <p><sup>2)</sup> The destination MAC ID is specified within the group 2 identifier. The consuming node generates the CID value and loads it into the consumed_connection_id attribute associated with this connection object. The numerically lowest available group 2 message ID shall be used in generating the consumed_connection_id attribute value. This value shall be loaded into the producing connection object's produced_connection_id attribute.</p> <p><sup>3)</sup> The source MAC ID is specified within the group 2 identifier. In this case, the producing node generates the CID value and loads it into the connection object's produced_connection_id attribute. This value shall be loaded into the consumed_connection_id attribute associated with the consuming connection object(s).</p> <p><sup>4)</sup> The producing node generates the CID value. The CID value shall be loaded into this connection object's consumed_connection_id attribute.</p> <p><sup>5)</sup> The default value assigned to the initial consumption characteristics nibble within an I/O connection. If this is a producing only I/O connection, then the default value remains in this nibble. Explicit messaging connections automatically configure this attribute when the connection is established.</p> |                                                       |

The node that generates a CID shall guarantee that it does not allocate the message ID/MAC ID pair in such a way that two separate nodes are capable of transmitting identical bit patterns within the identifier field (see 5.3.2.9).

- 7) **produced\_connection\_size:** The meaning of this attribute is different for explicit messaging connections and for I/O connections.

For explicit messaging connections:

This attribute defines the maximum number of message body bytes that a node is able to transmit across this connection.

Nodes that do not support the transmission of the fragmentation protocol initialise this attribute to the value 7 (message header (1 byte) + message body (7 bytes) = 8 bytes, which is the maximum length of a non-fragmented explicit message). Nodes that cannot or do not predefine a transmit length limit place the value  $\text{FFFF}_{\text{hex}}$  into this attribute.

Nodes that support the fragmentation protocol, but place a limit on the amount of message body bytes that can be transmitted in a single fragmented series initialise this attribute with this limit.

For I/O connections:

If the `transportclass_trigger` attribute identifies that this connection instance is a producer, then this attribute identifies the maximum amount of I/O data that may be produced as a single message across this connection. The amount of I/O data to be transmitted shall be less than or equal to the `produced_connection_size` attribute.

This attribute defaults to 0 within an I/O connection. If this attribute is set to a value greater than 8 in an I/O connection, then the connection shall use the fragmentation protocol.

I/O messages that contain no application I/O data and were configured to contain data indicate a no data event for the receiving application object(s). The behaviour of an application object upon detection of the no data event is application object-specific.

- 8) **consumed\_connection\_size:** The meaning of this attribute is different for explicit messaging connections and for I/O connections.

*For explicit messaging connections:*

This attribute defines the maximum number of message body bytes that this node is able to receive across this connection.

Nodes that do not support the reception of the fragmentation protocol initialise this attribute to the value 7 (message header (1 byte) + message body (7 bytes) = 8 bytes, which is the maximum length of a non-fragmented explicit message). Nodes that cannot or do not predefine a receive length limit place the value  $\text{FFFF}_{\text{hex}}$  into this attribute.

Nodes that support the fragmentation protocol, but place a limit on the amount of message body bytes that can be received in a single fragmented series initialise this attribute with this limit.

*For I/O connections:*

If the `transportclass_trigger` attribute identifies that this connection instance is a consumer, then this attribute identifies the maximum amount of data that may be received as a single message across this connection. The amount of I/O data received shall be less than or equal to the `consumed_connection_size` attribute. If an I/O connection object receives a message whose length is greater than the length specified by this attribute value, then it discards the message and discontinues any subsequent processing.

This attribute defaults to 0 within an I/O connection. If this attribute is set to a value greater than 8 in an I/O connection, then the connection shall use the fragmentation protocol.

- 9) **expected\_packet\_rate:** This attribute is used to generate the values loaded into the transmission trigger timer and the inactivity/watchdog timer. This attribute determines the value(s) loaded into the timers in milliseconds. See 5.3.2.5 for a description of the transmission trigger and inactivity/watchdog timers.

A request to configure this attribute may result in the specification of a time value that a product cannot meet. When a request to modify this attribute is received, the following steps shall be performed:

- if the specified value is not equal to a multiple of the available clock resolution, then the value is rounded up to the next useable value. This useable value is loaded into the `expected_packet_rate` attribute and is reported in the service data field of a `set_attribute_single` response message;
- if the requested value is equal to an increment of the clock resolution, then the requested value is loaded into the `expected_packet_rate` attribute and is reported in the service data field of a `set_attribute_single` response message.

The following steps shall be performed by a connection object in the established state when a request to modify the `expected_packet_rate` attribute is received:

- the current transmission trigger timer and inactivity/watchdog timer are cancelled;
- a new transmission trigger timer and a new inactivity/watchdog timer are activated based on the new value in the `expected_packet_rate` attribute.

This attribute defaults to 2 500 (2 500 ms) within explicit messaging connections and to zero (0 ms) within an I/O connection.

- 10) Reserved.

- 11) Reserved.

- 12) **watchdog\_timeout\_action:** This attribute defines the action the connection object shall perform when the inactivity/watchdog timer expires (see table 17);

**Table 17 – Values for the `watchdog_timeout_action`**

| Value   | Meaning                                                                                                                                                                                                                   |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | Transition to timed-out state. The connection transitions to the timed-out state and remains in this state until it is reset or deleted. This is the default value for this attribute in the case of I/O connections      |
| 1       | Auto delete. The connection class automatically deletes the connection in the event of an inactivity/watchdog time-out. This is the non-modifiable value for this attribute in the case of explicit messaging connections |
| 2       | Auto reset. The connection remains in the established state and restarts the inactivity/watchdog timer                                                                                                                    |
| 3 – 255 | Reserved                                                                                                                                                                                                                  |

- 13) **produced\_connection\_path\_length:** Identifies the number of bytes of information within the `produced_connection_path` attribute. This is set when the `produced_connection_path` attribute is set. This attribute defaults to the value 0.

- 14) **produced\_connection\_path:** The `produced_connection_path` attribute is made up of a byte stream which defines the application object(s) whose data is to be produced by this connection object. The format of this byte stream is specified in annex C. This attribute defaults to non-existent upon creation of the connection. It remains non-existent within explicit messaging connections and within connection objects that do not produce.

- 15) **consumed\_connection\_path\_length:** Identifies the number of bytes of information within the `consumed_connection_path` attribute. This is set when the `consumed_connection_path` attribute is set. This attribute defaults to the value 0.