

# Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces —

## Part 2: DeviceNet

The European Standard EN 50325-2:2000 has the status of a  
British Standard

ICS 43.180

## National foreword

This British Standard is the official English language version of EN 50325-2:2000.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Monitoring and control aspects of AMT, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

### Cross-references

The British Standards which implement international or European publications referred to in this document may be found in the BSI Standards Catalogue under the section entitled “International Standards Correspondence Index”, or by using the “Find” facility of the BSI Standards Electronic Catalogue.

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

**Compliance with a British Standard does not of itself confer immunity from legal obligations.**

This British Standard, having been prepared under the direction of the DISC Board, was published under the authority of the Standards Committee and comes into effect on 15 July 2001

### Summary of pages

This document comprises a front cover, an inside front cover, the EN title page, pages 2 to 123 and a back cover.

The BSI copyright date displayed in this document indicates when the document was last issued.

### Amendments issued since publication

Amd. No.	Date	Comments

BSI 07-2001

EUROPEAN STANDARD

**EN 50325-2**

NORME EUROPÉENNE

EUROPÄISCHE NORM

October 2000

---

ICS 43.180

English version

**Industrial communications subsystem based on ISO 11898 (CAN)  
for controller-device interfaces —  
Part 2: DeviceNet**

This European Standard was approved by CENELEC on 2000-01-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists only in English. A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official version.

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

**CENELEC**

European Committee for Electrotechnical Standardization  
Comité Européen de Normalisation Electrotechnique  
Europäisches Komitee für Elektrotechnische Normung

**Central Secretariat: rue de Stassart 35, B - 1050 Brussels**

## Foreword

This European Standard was prepared by the Technical Committee CENELEC TC 65CX, Fieldbus.

The text of the draft was submitted to the Unique Acceptance Procedure and was approved by CENELEC as EN 50325-1 on 2000-01-01.

The following dates were fixed:

- latest date by which the EN has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2001-05-01
- latest date by which the national standards conflicting with the EN have to be withdrawn (dow) 2003-01-01

This European Standard is divided into three parts:

- Part 1 General requirements
- Part 2 DeviceNet specification
- Part 3 Smart Distributed System (SDS) specification

NOTE This European Standard exists only in English.

The specifications for DeviceNet and SDS are based on ISO 11898, a broadcast-oriented communications protocol. However, ISO 11898 specifies only part of a complete communication system, and additional specifications are needed for other layers to ensure precise data exchange functionality and support of inter-operating devices. The DeviceNet and SDS specifications build on ISO 11898 to describe a complete industrial communication system.

### General information on licensing

CENELEC calls attention to the fact that patent rights are linked to Part 3 of this 50325<sup>1)</sup>. The patent holder, Honeywell Inc., has assured to CENELEC that it is willing to grant a licence under these patents on reasonable and non discriminatory terms and conditions to anyone wishing to obtain such a license, applying the rules of CEN/CENELEC Memorandum 8.

Honeywell's undertakings (policy letter on licensing, the license offer and the form of license) in this respect are on file with CENELEC and available for inspection by all interested parties at the CENELEC Central Secretariat.

The license details may be obtained from

The Director  
(Industrial Marketing and Applied Technology Sensing and Controls Europe)  
Honeywell Control Systems Ltd.  
Newhouse Industrial Estate,  
Motherwell,  
Lanarkshire  
Scotland ML1 5SB  
GB

---

<sup>1)</sup> in preparation.

## Contents

<b>Introduction</b> .....	<b>5</b>
<b>1 Scope</b> .....	<b>6</b>
<b>2 Normative references</b> .....	<b>6</b>
<b>3 Definitions and abbreviations</b> .....	<b>7</b>
<b>4 Classification</b> .....	<b>11</b>
4.1 General .....	11
4.2 DeviceNet communication model .....	12
4.3 DeviceNet and CAN.....	12
<b>5 Characteristics</b> .....	<b>13</b>
5.1 DeviceNet connections .....	13
5.2 DeviceNet messaging protocol .....	15
5.3 DeviceNet communication object classes .....	30
5.4 Network access state machine .....	54
5.5 Predefined master/slave connection set.....	57
5.6 Physical layer .....	66
<b>6 Product information</b> .....	<b>70</b>
6.1 Instructions for installation, operation and maintenance.....	70
6.2 Marking .....	70
<b>7 Normal service, transport and mounting conditions</b> .....	<b>70</b>
7.1 Normal service conditions.....	70
7.2 Conditions during transport and storage.....	71
7.3 Mounting .....	71
<b>8 Constructional and performance requirements</b> .....	<b>72</b>
8.1 Indicators and configuration switches .....	72
8.2 DeviceNet cable.....	74
8.3 Terminating resistors .....	75
8.4 Connectors.....	75
8.5 Device taps and power taps.....	76
8.6 Network powered devices.....	77
8.7 Miswiring protection .....	78
8.8 Power supplies.....	78
8.9 Electromagnetic compatibility .....	78
<b>9 Tests</b> .....	<b>80</b>
9.1 General .....	80
9.2 Electrical and EMC testing.....	80
9.3 Logical testing .....	87
<b>Annex A (normative) Common services</b> .....	<b>90</b>
A.1 DeviceNet service codes and names .....	90
A.2 Common service definitions .....	90
<b>Annex B (normative) DeviceNet error codes</b> .....	<b>96</b>
<b>Annex C (normative) Connection path attribute definition</b> .....	<b>97</b>
C.1 General.....	97
C.2 Segment type format .....	97
C.3 Segment definition rules.....	99
<b>Annex D (normative) Data type specification and encoding</b> .....	<b>100</b>
D.1 Data type specification .....	100
D.2 Data type encoding.....	101
<b>Annex E (normative) Communication Objects library</b> .....	<b>103</b>
E.1 General.....	103
E.2 Object Class Codes.....	103
E.3 Identity object (class code: 01 <sub>hex</sub> ) .....	103
E.4 Message router object (class code: 02 <sub>hex</sub> ).....	109

E.5 DeviceNet object (class code: 03 <sub>hex</sub> ).....	112
E.6 DeviceNet connection object (class code: 05 <sub>hex</sub> ).....	112
E.7 Acknowledge handler object (class code: 2B <sub>hex</sub> ).....	112
<b>Annex F (normative) Value ranges .....</b>	<b>122</b>

## Introduction

DeviceNet is intended for use in, but is 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.

## 1 Scope

This Part 2 of EN 50325 contains 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.

## 2 Normative references

This European Standard incorporates by dated or undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this European Standard only when incorporated in it by amendment or revision. For undated references, the latest edition of the publication referred to applies.

EN 50081-2	1993	Electromagnetic compatibility (EMC) - Generic emission standard Part 2: Industrial environment
EN 50082-2	1995	Electromagnetic compatibility - Generic immunity standard Part 2: Industrial environment
EN 55011	1998	Industrial, scientific and medical (ISM) radio-frequency equipment – Radio disturbance characteristics - Limits and methods of measurement (CISPR 11:1997, modified)
EN 60529	1991	Degrees of protection provided by enclosures (IP code) (IEC 60529:1989)
EN 60947-5-2	1998	Low-voltage switchgear and controlgear -- Part 5: Control circuit devices and switching elements --Section 2: Proximity switches (IEC 60947-5-2:1997, modified)
EN 61000-4-2	1995	Electromagnetic compatibility (EMC) -- Part 4: Testing and measuring techniques -- Section 2: Electrostatic discharge immunity test (IEC 61000-4-2:1995)
EN 61000-4-3	1996	Electromagnetic compatibility (EMC) -- Part 4: Testing and measuring techniques --Section 3: Radiated, radio-frequency, electromagnetic field immunity test (IEC 61000-4-3:1995, modified)
EN 61000-4-4	1995	Electromagnetic compatibility (EMC) -- Part 4: Testing and measuring techniques --Section 4: Electrical fast transient / burst immunity test (IEC 61000-4-4:1995)
EN 61000-4-5	1995	Electromagnetic compatibility (EMC) -- Part 4: Testing and measuring techniques --Section 5: Surge immunity test (IEC 61000-4-5:1995)
EN 61000-4-6	1996	Electromagnetic compatibility (EMC) --Part 4: Testing and measuring techniques -- Section 6: Immunity to conducted disturbances, induced by radio-frequency fields (IEC 61000-4-6:1996)
EN 61131-3	1993	Programmable controllers -- Part 3: Programming languages (IEC 61131-3:1993)
ISO/IEC 7498-1	1994	Information technology - Open Systems Interconnection - Basic Reference Model : The Basic Model
ISO 11898	1993	Road vehicles - Interchange of digital information - Controller area network (CAN) for high-speed communication



### 3 Definitions and abbreviations

For the purposes of this Part 2 of the European Standard, 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 a DeviceNet network 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

#### 3.6

##### **broadcast**

communication from one node to all other nodes

#### 3.7

##### **CAN<sub>H</sub>**

positive half of the differential physical CAN signal

#### 3.8

##### **CAN<sub>L</sub>**

negative half of the differential physical CAN signal

#### 3.9

##### **change of state**

process of data exchange which occurs only when a device's or controller's data changes state according to specific change criteria

#### 3.10

##### **client**

- 1) object which uses the services of another (server) object to perform a task. See server (0)
- 2) 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**

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 network are assigned the same MAC ID

### 3.22

#### **explicit messaging**

each explicit message 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**

UCMM capable device that has gained ownership of the predefined master/slave connection set within a server such 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 it has allocated

### 3.26

#### **group 2 server**

unconnected message manager (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 its components are not referenced to the V- of the network physical layer. See non-isolated device (0)

### 3.32

#### **master**

node which gathers and distributes I/O using the Predefined Connection Set

### 3.33

#### **medium access control (MAC) ID**

network 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 network.

### 3.36

#### **non-isolated device**

device in which all components are referenced to the V- of the network physical layer. See isolated device (0)

**3.37**  
**object**

- 1) abstract representation of a device's capabilities. Objects may be composed of any or all of the following components: a) Data (information which changes with time); b) Configuration (parameters for behaviour); c) Procedures (actions that may be performed using data and configuration)
- 2) 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 (0)

**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 facilitate 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 (0)

**3.44**  
**service**

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

**3.45**  
**slave**

a DeviceNet slave returns data to its master using the Predefined Connection Set. The communication method used is set by the master

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

MAC ID of a node that is transmitting a message

**3.47**  
**strobing**

process of data exchange which occurs when a device, e.g. a controller, sends a single request for data from one or more devices

NOTE Each device receiving the message then responds with its requested data in a predetermined sequence.

**3.48**  
**trigger**

service used by an application to initiate the production of data

**3.49**

**UCMM capable device**

device that supports the UCMM

**3.50**

**UCMM incapable device**

device that does not support the UCMM

**3.51**

**unconnected explicit message**

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

**3.52**

**Unconnected Message Manager (UCMM)**

function within a node that receives and processes unconnected explicit messages

**3.53**

**unsigned short integer (USINT)**

8-bit integer

**3.54**

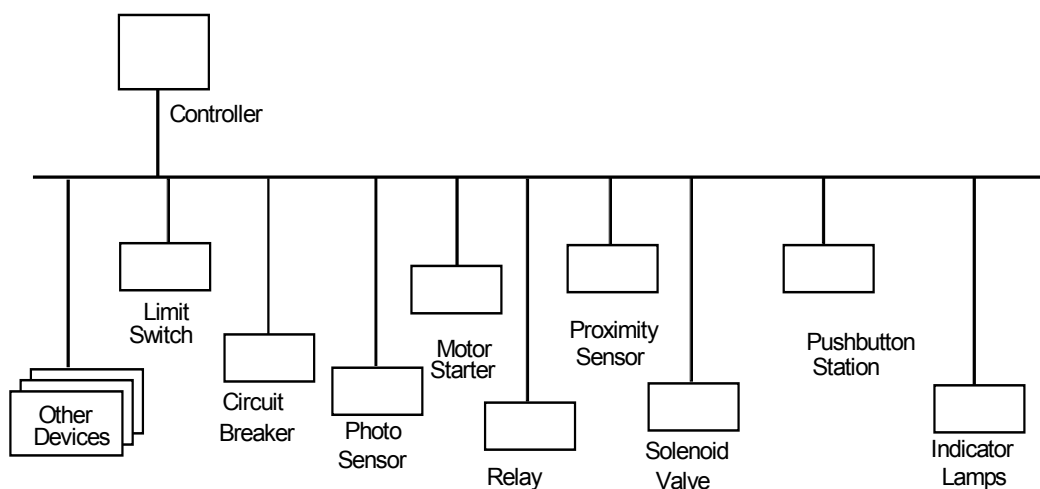
**unsigned integer (UINT)**

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 DC 24 V. 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 8 bytes of data may be transmitted without fragmentation. A maximum of 64 nodes may be connected using a linear network 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.



**Figure 1 - Typical DeviceNet network**

This Part 2 of EN 50325 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 the Connection ID (CID).

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

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 moves 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 2 of EN 50325 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 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 network 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 network relationship;
- **DeviceNet object:** provides the configuration and status of a physical DeviceNet network connection;
- **Message router:** forwards an 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.

NOTE See CAN Specification - Version 2.0, Part A : Sept. 1991, Robert Bosch GmbH.

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

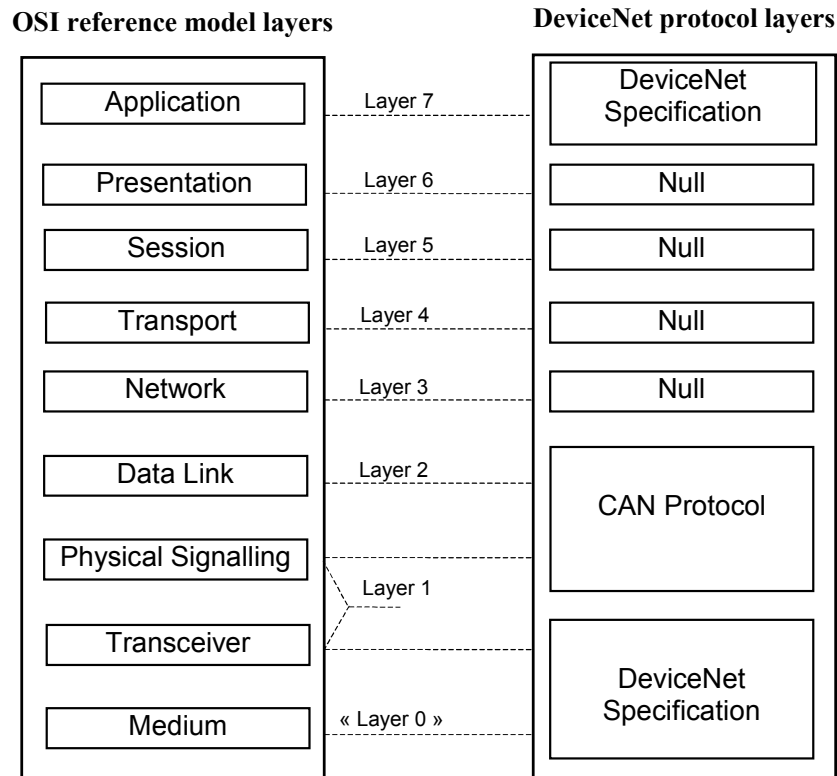


Figure 2 – DeviceNet protocol architecture compared with the OSI reference model

## 5 Characteristics

### 5.1 DeviceNet connections

#### 5.1.1 General

DeviceNet is a connection based network. 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 connections.

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

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	1	X	X	X	X	7f0 - 7ff	Invalid CAN Identifiers	
10	9	8	7	6	5	4	3	2	1	0				

**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 source or 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 the 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).

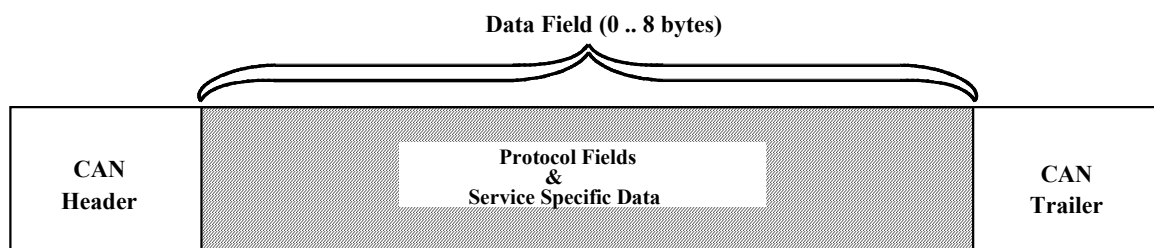


Figure 4 - Explicit message CAN data field use

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

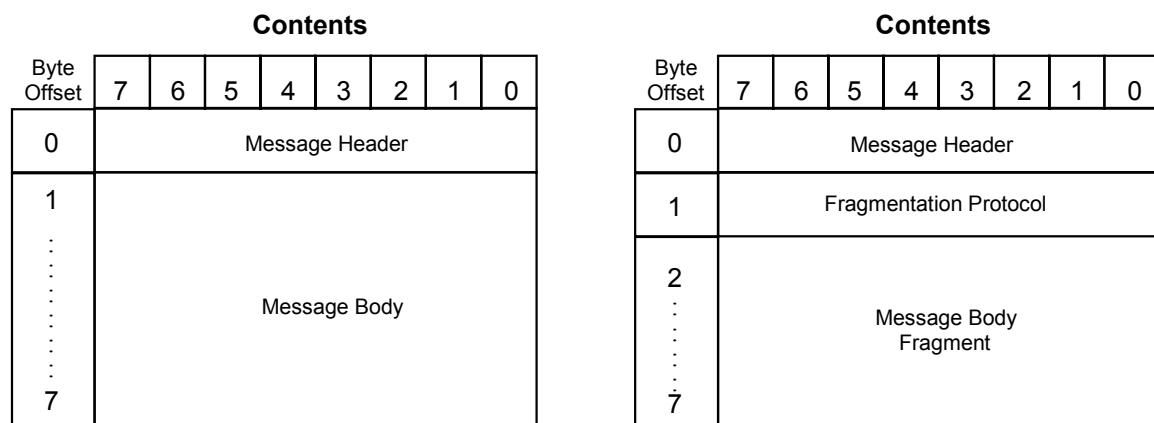


Figure 5 - Explicit message data field format

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:

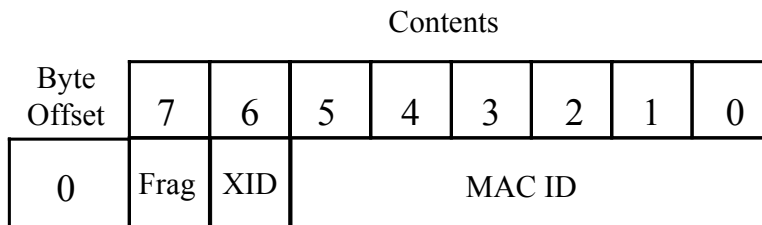


Figure 6 - Explicit message header format

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.

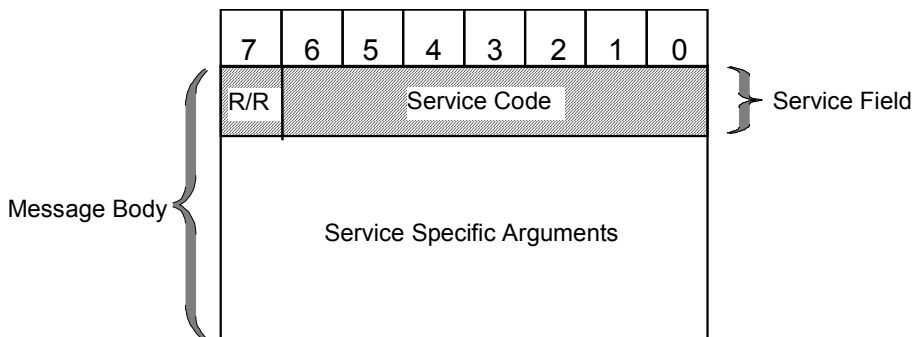


Figure 7 - Service field format

Service field contents:

- **Service code:** the value specified within the least significant 7 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 document (see Annex A).

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 8 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 explicit messaging connection and close connection services provided by the UCMM. See 9.3.3 for test specifications regarding establishment of explicit messaging connections.

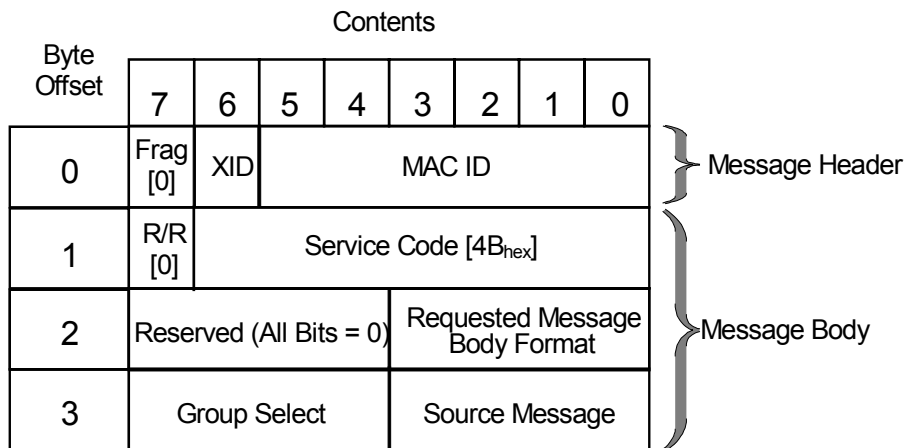
The UCMM processes two services which manage the allocation and deallocation of explicit messaging connections:

- **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.	
1) The class and instance ID fields are specified as 8 bit integers (USINT). 2) The class ID is specified as an 8 bit integer (USINT) and the instance ID is specified as a 16 bit integer (UINT). 3) The class and instance ID fields are specified 16 bit field integers (UINT). 4) 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 following 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

1) 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>

1) The client places this value within the message ID component of the message Group 1 or 3 Identifier.  
2) 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.

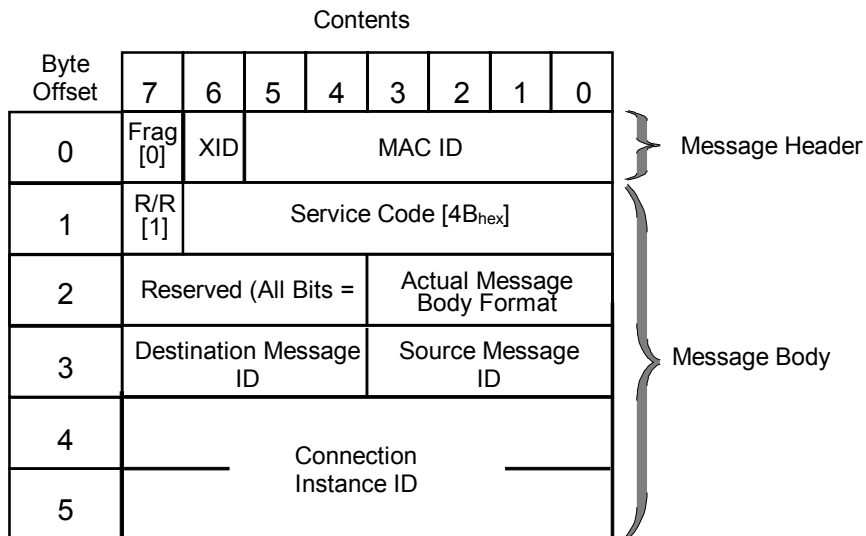
The UCMM within the server validates the open explicit messaging connection request arguments. If they are valid, the UCMM invokes the create service of the connection class to obtain a connection object instance (see 5.3.2.6). The resulting connection object is automatically configured to be an explicit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

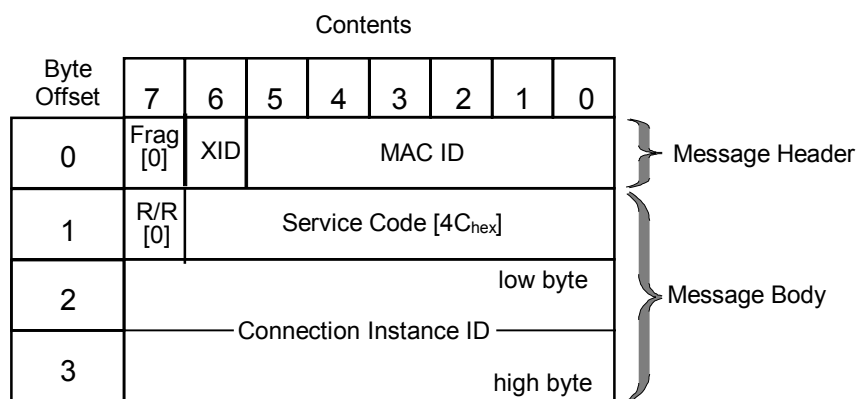
#### 5.2.1.5.4 Close connection request

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 it 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 as 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.



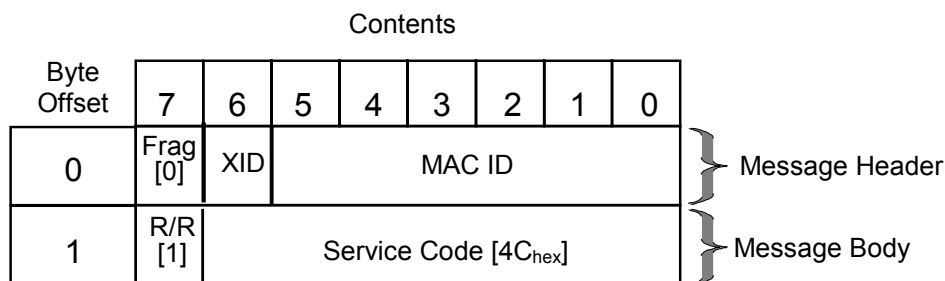
**Figure 10 - Close connection request format**

#### 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

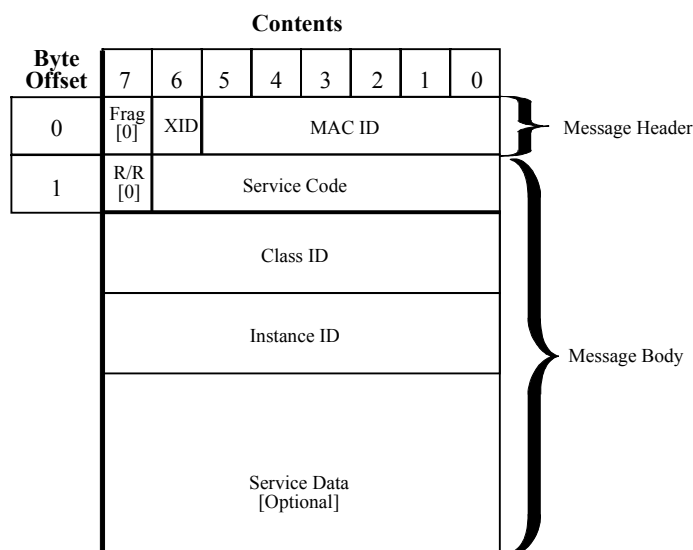
Error condition descriptions:

- **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:



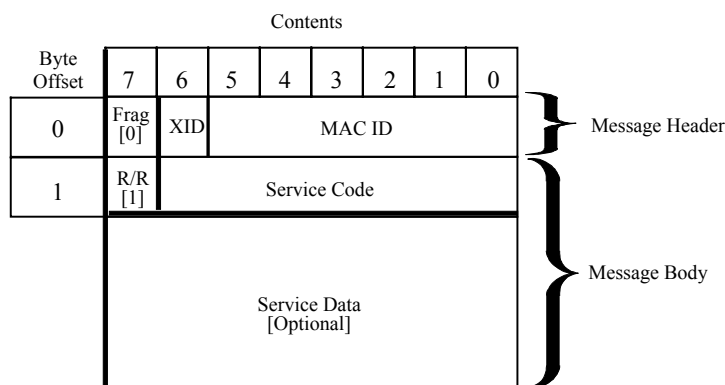
**Figure 12 – Non-fragmented explicit request message format**



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.
- **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.

		Contents							
Byte Offset		7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID						
1	R/R [1]	Service Code [14 <sub>hex</sub> ]							
2	General Error Code								
3	Additional Code								

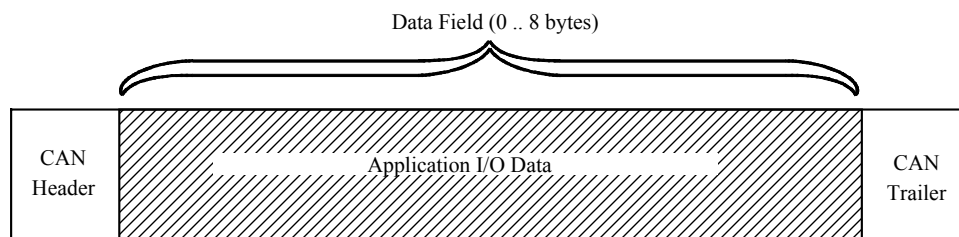
**Figure 14 - Error response message**

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:

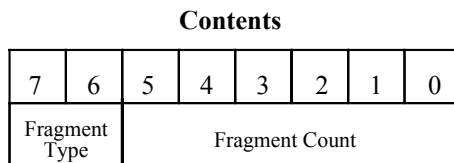
- Explicit messaging connection object instances examine the length of each message to be transmitted. If the message is greater than eight bytes in length, then the fragmentation protocol is used.
- I/O connection object instances examine the `produced_connection_size` attribute of the connection object (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):

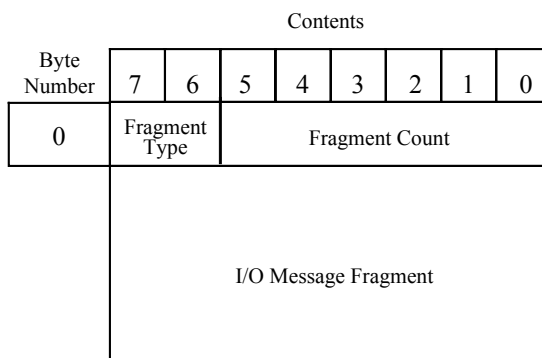
**Table 6 - Fragment type bit values**

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 acknowledge <sup>4)</sup>

1) 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.  
 2) This fragment is neither the first nor the last fragment in the series.  
 3) Marks this as the last fragment.  
 4) The value the receiver of a fragmented message uses to acknowledge the reception of a fragment.

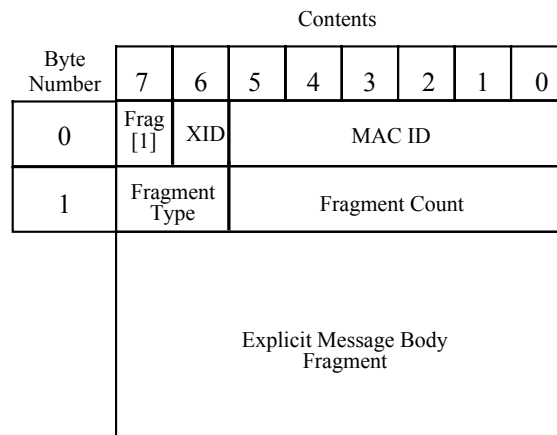
- **Fragment count:** Marks each separate fragment such that the receiver may determine whether or not a fragment has been missed. If the fragment type is first fragment, then this field takes on a special meaning (as described in Table 6). The fragment count is increased by one for each successive fragment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

- If the fragment count is  $3F_{hex}$ , 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 first fragment or the fragment count is not equal to 0 or  $3F_{hex}$ ; 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 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 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

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.

Byte Number	Contents							
	7	6	5	4	3	2	1	0
0	Frag [1]	XID	MAC ID					
1	Fragment Type (3)		Fragment Count					
2	Ack Status							

**Figure 19 - Acknowledgement message format**

Acknowledgement message contents:

- **Fragment type:** Indicates that this is a fragment acknowledge 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 acknowledgement timer is started. The amount of time to wait is application specific.
- If the wait for 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 waiting 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 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										MESSAGE ID MEANING			
10	9	8	7	6	5	4	3	2	1	0			
1	0	MAC ID					Group 2 Message ID			Group 2 Messages			
1	0	Destination MAC ID					1	1	1	Duplicate MAC ID Check Message			

**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 Offset	Contents							
	7	6	5	4	3	2	1	0
0	R/R	Physical Port Number						
1	Vendor ID							low byte
2								high byte
3								low byte
4	Serial Number							
5								
6								high byte

**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 0.

### 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 15) and the description of the inactivity/watchdog timer (5.3.2.5.3) for more details.
5-255	Reserved	

1) 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.

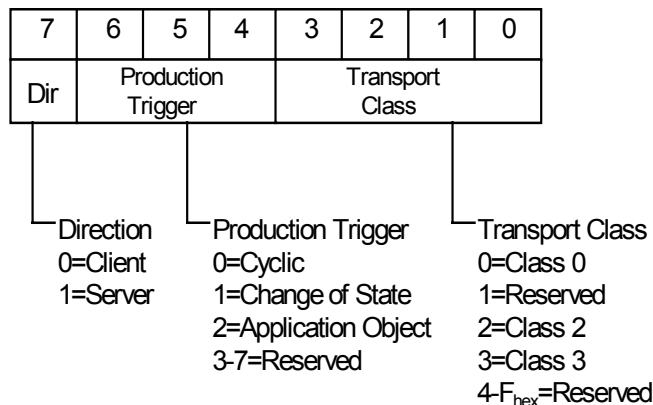
When a transition is made to the established state, all timers associated with the connection object shall be activated (see 5.3.2.5).

- 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) :



**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
FFF <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
FFF <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			

**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>

1) 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).

2) 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, it shall be read and subsequently loaded into the producing connection object's produced\_connection\_id attribute.

3) 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).

4) 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).

5) 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.

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>

1) 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).

2) 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

3) 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).

4) The producing node generates the CID value. The CID value shall be loaded into this connection object's consumed\_connection\_id attribute.

5) 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.

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 an transmit length limit place the value FFFF<sub>hex</sub> 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 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 an receive length limit place the value  $FFFF_{hex}$  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 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 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.
- 16) **consumed\_connection\_path:** The `consumed_connection_path` attribute is made up of a byte stream which defines the application object(s) that are to receive the data consumed 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 consume.
- 17) **production\_inhibit\_time:** This attribute is used to configure the minimum time between successive data production. This attribute determines the value loaded into the timer in milliseconds. A value of 0 (the default value for this attribute) indicates no inhibit time.

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 `production_inhibit_time` 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 `production_inhibit_time` and is reported in the service data field of a `set_attribute_single` response message.

The `production_inhibit_time` value is loaded into the production inhibit timer each time new data production occurs.

When a connection object is in the established state, any modifications to the `production_inhibit_time` attribute have no effect on a currently running production inhibit timer. The new `production_inhibit_time` value is loaded into the production inhibit timer when the next data production occurs.

When the `apply_attributes` service is received, the `production_inhibit_time` shall be verified against the `expected_packet_rate` attribute. If the `expected_packet_rate` value is greater than zero, but less than the `production_inhibit_time` value, than an error shall be returned. The additional error code returned to the `apply_attributes` service is `11hex` indicating the `production_inhibit_time` attribute.

### 5.3.2.5 Connection timing

#### 5.3.2.5.1 General

Three types of timers are involved in a connection:

- Transmission trigger timer;
- Inactivity/watchdog timer;
- Production inhibit timer.

#### 5.3.2.5.2 Transmission trigger timer

Expiry of the transmission trigger time is an indication that the associated connection object may need to transmit a message. If a production has not occurred since the timer was activated, then the connection object shall produce to avoid an inactivity/watchdog time-out at the server end-point(s).

#### 5.3.2.5.3 Inactivity/watchdog timer

This timer is used by consuming connection objects including:

- Client end-point connection objects whose transportClass\_trigger attribute indicates either transport class 2 or 3;
- All server end-point connection objects.

This timer is activated when the connection transitions to the established state. The tasks listed below shall be performed by a connection object upon detecting that a valid message has been consumed:

- The current inactivity/watchdog timer shall be stopped;
- A new inactivity/watchdog timer shall be activated.

These actions shall take place before the received message is processed.

Expiry of the inactivity/watchdog time is an indication that the connection object has timed out while waiting to consume. The connection object shall then perform the following steps:

- issue an indication of this event to the application;
- perform the action indicated by the watchdog\_timeout\_action attribute.

The initial value loaded into the inactivity/watchdog timer is either 10 000 ms or the expected\_packet\_rate multiplied by 4, depending on which value is numerically greater. All subsequent activations of the inactivity/watchdog timer use the expected\_packet\_rate multiplied by 4 as the number of milliseconds to load into the inactivity/watchdog timer. If the expected\_packet\_rate attribute contains the value 0, then the inactivity/watchdog timer is neither activated nor used by the connection object.

#### 5.3.2.5.4 Production inhibit timer

This timer shall be managed by the connection object within the client end-point of a connection when the production\_inhibit\_time attribute value is non-zero. This timer is started when data is produced by the application object. The connection object shall not produce new data if this timer is running, although a retry may be sent. Expiry of the production inhibit time allows the connection object to send new data.

The production inhibit timer is initialised with the value in the production\_inhibit\_time attribute. If the production\_inhibit\_time attribute contains the value 0, then the production inhibit timer is neither activated nor used by the client end-point.

#### 5.3.2.6 Connection object instance services

The connection object instance supports the following DeviceNet common services (as described in Table 18):

**Table 18 - Connection object instance services**

Need in implementation	Service name	Service description
Required	Get_attribute_single	Reads a connection object attribute.
Optional	Set_attribute_single	Modifies a connection object attribute. The connection object returns information in the service data field of a set_attribute_single response when the expected_packet_rate attribute is modified as indicated in 5.3.2.4
Optional	Reset	Resets the inactivity/watchdog timer associated with a connection object. When a connection in the timed out state receives a reset request it also transitions back to the established state.
Optional	Delete	Deletes a connection object and releases all associated resources.
Optional	Apply_attributes	Delivers the connection object to the application which performs the set of tasks necessary to create the specified connection. A connection instance returns the produced_connection_id and consumed_connection_id attributes within the service data field of an apply_attributes response message as indicated in Table 19.
Service codes are defined in Annex A.		

The information to be specified by a connection object instance within the service data field of a successful apply\_attributes response is shown in Table 19.

**Table 19 - Service data for connection object apply\_attributes response**

Name	Data type	Description of parameter
Produced CID	UINT	Contains the value within the connection instance's produced_connection_id attribute
Consumed CID	UINT	Contains the value within the connection instance's consumed_connection_id attribute

The connection object also supports the following internal services:

- **Send\_message:** Used internally to trigger the transmission of a message.
- **Receive\_data:** Used internally to deliver a received frame to the connection object. The connection object determines whether or not it needs to reassemble a series of data fragments into a complete message before processing the message.

### 5.3.2.7 Connection instance behaviour

Figure 24 provides a general overview of the behaviour associated with an I/O connection object (instance\_type attribute = I/O).



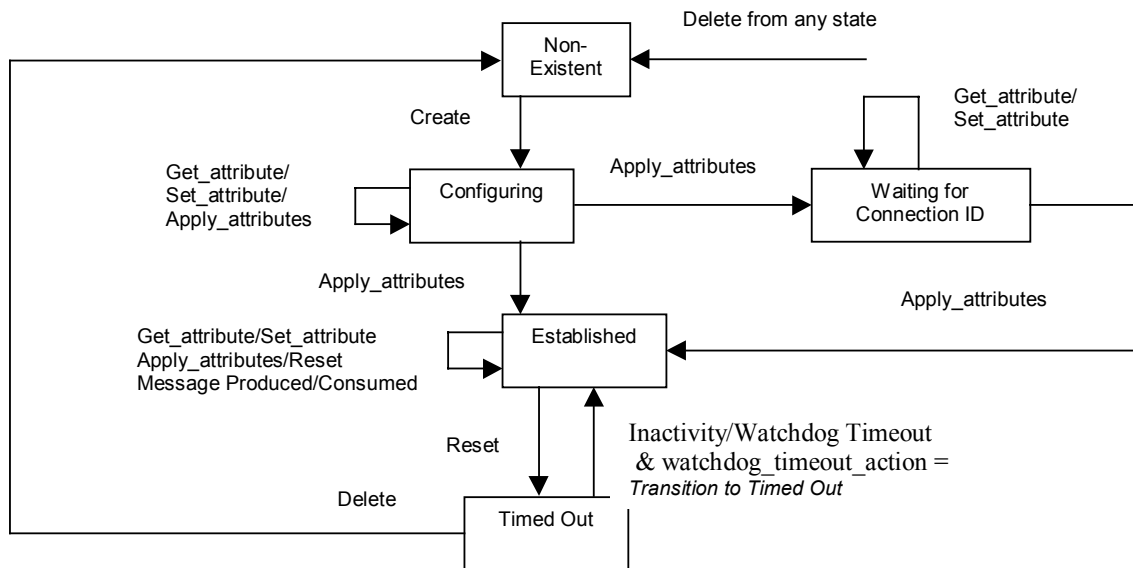


Figure 24 - I/O connection object state transition diagram

Table 20 provides a detailed state event matrix for an I/O connection object.

Table 20 – I/O connection state event matrix

I/O connection object state					
Event	Non-existent	Configuring	Waiting for CID	Established	Timed out
Connection class receives a create request	Class creates a connection object. Set instance_type to I/O. Set other attributes to default values. Transition to configuring.	Not applicable	Not applicable	Not applicable	Not applicable
Connection class receives a delete request	Error: object does not exist (general error code 16 <sub>hex</sub> ).	Release all associated resources. Transition to non-existent.	Release all associated resources. Transition to non-existent.	Release all associated resources. Transition to non-existent.	Release all associated resources. Transition to non-existent.
Set_attribute_single	Error: object does not exist (general error code 16 <sub>hex</sub> ).	Validate/service the request according to 5.3.2.8. Return response.	If request to modify produced or consumed_connection_id then validate the value, service the request and return response. If this is a request to access any other attribute, then return an Error Response whose General Error Code is set to 0C <sub>hex</sub> .	Validate/service the request according to 5.3.2.8. Return response.	Validate/service the request according to 5.3.2.8. Return response.
Get_attribute_single	Error: object does not exist (general error code 16 <sub>hex</sub> ).	Validate/service the request according to 5.3.2.8. Return response.	Validate/service the request according to 5.3.2.8. Return response.	Validate/service the request according to 5.3.2.8. Return response.	Validate/service the request according to 5.3.2.8. Return response.

I/O connection object state					
Event	Non-existent	Configuring	Waiting for CID	Established	Timed out
Reset	Error: Object does not exist (General Error Code 16 <sub>hex</sub> )	Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0C <sub>hex</sub> )	Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0C <sub>hex</sub> )	If the Inactivity/Watchdog Timer is running, cancel it and re-start using the value in the expected_packet_rate attribute.	Using the value in the expected_packet_rate attribute, start the Inactivity/Watchdog timer (if the expected_packet_rate attribute has not been set to zero) and transition back to the Established state .
Apply_Attributes	Error: Object does not exist (General Error Code 16 <sub>hex</sub> )	Deliver Connection Object to the application which validates the attribute information. If either of the connection_id attributes (produced or consumed) needs to be configured and cannot be generated by this node, then perform the other steps necessary to configure the connection, return a Success Response and transition to the Waiting For CID state. If all attributes are configured, then perform the steps necessary to satisfy this connection, start all required timers, and transition to the Established state. If an error is detected, then an Error Response is returned and the Connection remains in the Configuring state <sup>1)</sup> .	If either of the connection_id attributes (produced or consumed) still needs to be configured and cannot be generated by this node, then return a Success Response and remain in the Waiting For CID state. If the produced and/or consumed connection id attributes are configured, transition to the Established state and return a Success Response <sup>1)</sup> .	All modifications take place immediately once the connection has transitioned to the Established state. Return Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0C <sub>hex</sub> )	Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0C <sub>hex</sub> )
Receive_Data	Not applicable	Discard the message	Discard the message	If a complete, valid message <sup>2)</sup> has been received, reset the Inactivity/Watchdog Timer <sup>3)</sup> and deliver the I/O message to the application	Discard the message

I/O connection object state					
Event	Non-existent	Configuring	Waiting for CID	Established	Timed out
Send_Message	Not applicable	Return internal error - do not send the message	Return internal error - do not send the message	Examine the connection_size attribute of the connection and transmit the message, fragmented if necessary.	Return internal error - do not send the message
Inactivity/Watchdog Timer expires	Not applicable	Not applicable	Not applicable	Examine the watchdog_timeout_action attribute of the connection and perform the indicated action.	Not applicable

1) If the configuration indicates that a Message ID needs to be allocated and an available Message ID does not exist in the specified Message Group, then an Error Response whose General Error Code indicates *Resource Unavailable* (02<sub>hex</sub>) is returned. If a Connection Object attribute value passed the range check when it was initially configured but the attribute value conflicts with another piece of information in the node when the Apply request is processed, then an Error Response is returned whose General Error Code is set to *Invalid Attribute Value* (09<sub>hex</sub>) and whose Additional Code is set to the Attribute ID of the *offending* Connection Object Attribute ID.

2) The Connection Object verifies that the length of the received I/O Message is less than or equal to the consumed\_connection\_size attribute prior to processing the message. If the length of the received message is less than or equal to the consumed\_connection\_size attribute, then the I/O Connection Object resets the Inactivity/Watchdog Timer, exhibits the externally visible behaviour indicated by its attribute settings, and delivers the message to the application. If the length of the received message is greater than the consumed\_connection\_size attribute, then the I/O Connection Object immediately discards the message and discontinues any subsequent processing. This is the only message content validation performed by an I/O Connection Object. Any subsequent validation shall be performed by the application.

3) If a fragmented message is being received, then the Inactivity/Watchdog Timer is not reset until the entire message has been received.

Figure 25 provides a general overview of the behaviour associated with an explicit messaging connection.

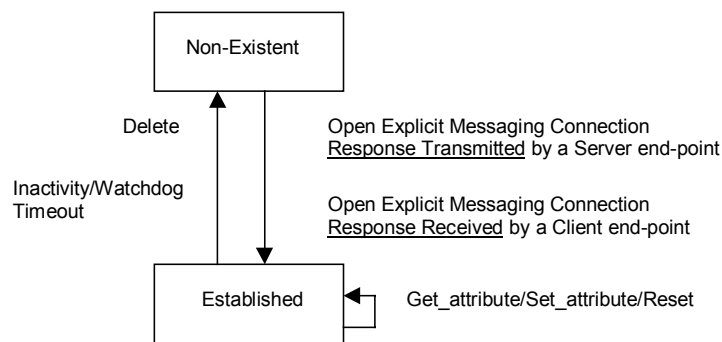


Figure 25 - Explicit messaging connection object state transition diagram

Licensed Copy: :FULLNAME, : DATE, Uncontrolled Copy, (c) BSI

Table 21 provides a detailed state event matrix for an explicit messaging connection object.

**Table 21 - Explicit messaging connection state event matrix**

Explicit messaging connection object state		
Event	Non-existent	Established
UCMM receives an open explicit messaging connection request/response and invokes the create service of the connection class	If possible, class creates connection object with the instance_type attribute set to explicit messaging <sup>1)</sup> . Other attributes are automatically configured using the information in the open explicit messaging connection request/response. Transition to established. If request received, transmit open explicit messaging connection response.	Not applicable
UCMM receives a close request	Error: object does not exist (general error code 16 <sub>hex</sub> )	Release all associated resources. Transition to non-existent.
Connection class receives a delete request	Error: object does not exist (general error code 16 <sub>hex</sub> )	Release all associated resources. Transition to non-existent.
Set_attribute_single	Error: object does not exist (general error code 16 <sub>hex</sub> )	Validate/service the request according to 5.3.2.8. Return response.
Get_attribute_single	Error: object does not exist (general error code 16 <sub>hex</sub> )	Validate/service the request according to 5.3.2.8. Return response.
Reset	Error: object does not exist (general error code 16 <sub>hex</sub> )	Cancel the current inactivity/watchdog timer. Using the value in the expected_packet_rate attribute, restart the inactivity/watchdog timer.
Apply_attributes	Error: Object does not exist (general error code 16 <sub>hex</sub> )	Error: The object cannot perform the requested service in its current mode/state. (General error code value = 0C <sub>hex</sub> )
Receive_data	Not applicable	If a valid message or message fragment has been received, then reset the inactivity/watchdog timer <sup>2)</sup> . Either process/store the fragment or handle the explicit message.
Send_message	Not applicable	Examine the length of the message to transmit and transmit the message, fragmented if necessary.
Inactivity/watchdog timer expires	Not applicable	Release all associated resources. Transition to Non-existent.
<p>1) If the configuration indicates that a message ID needs to be allocated and an available message ID does not exist in the specified message group, then an error response whose general error code indicates resource unavailable (02<sub>hex</sub>) is returned.</p> <p>2) The MAC ID field within the message header of all explicit messages and/or message fragments is examined. If the destination MAC ID is specified in the CID (CAN identifier field), 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 node's MAC ID shall be specified in the message header. If either of these checks fail, then the inactivity/watchdog timer is not reset and the message/message fragment is discarded.</p>		

### 5.3.2.8 Connection object attribute access rules

During the configuration of a connection instance using the set\_attribute service, a node shall check the value of each separate attribute which is to be modified. If an error is detected, then an error response shall be returned.

If the produced\_connection\_id and/or consumed\_connection\_id attributes contain a non-default value upon reception of the apply request, then the relevant portion of the initial\_comm\_characteristics attribute is ignored and the ID value is validated and used.

Table 22 and Table 23 indicate when an attribute can be read or written via a get or set operation based on the connection's state and instance\_type.

If a request to get/set a supported attribute is received but the current state and/or type of connection dictates that the requested access is invalid, then the return error status indicates that the object cannot perform the requested service in its current mode/state (general error code value = 0C<sub>hex</sub>). Error codes are specified in Annex B.

Table 22 - I/O connection object attribute access

I/O connection state					
Attribute	Non-existent	Configuring	Waiting for CID	Established	Timed out
State	Not available	Get only	Get only	Get only	Get only
Instance_type	Not available	Get only	Get only	Get only	Get only
TransportClass_trigger	Not available	Get/set	Get only	Get only	Get only
Produced_connection_id	Not available	Get/set	Get/set	Get/set	Get/set
Consumed_connection_id	Not available	Get/set	Get/set	Get/set	Get/set
Initial_comm_characteristics	Not available	Get/set	Get only	Get only	Get only
Produced_connection_size	Not available	Get/set	Get only	Get only	Get only
Consumed_connection_size	Not available	Get/set	Get only	Get only	Get only
Expected_packet_rate <sup>1)</sup>	Not available	Get/set	Get only	Get/set	Get/set
Watchdog_timeout_action	Not available	Get/set	Get only	Get/set	Get/set
Produced_connection_path_length	Not available	Get only	Get only	Get only	Get only
Produced_connection_path	Not available	Get/set	Get only	Get only	Get only
Consumed_connection_path_length	Not available	Get only	Get only	Get only	Get only
Consumed_connection_path	Not available	Get/set	Get only	Get only	Get only
Production_inhibit_time	Not available	Get/set	Get only	Get/set	Get/set

1) When a connection object is in the established state, any modifications to the expected\_packet\_rate attribute have immediate effect on the inactivity/watchdog timer. The following steps are performed by a connection object in the established state when a request is received to modify the expected\_packet\_rate attribute:

- the current inactivity/watchdog timer is cancelled
- a new inactivity/watchdog timer is activated based on the new value in the expected\_packet\_rate attribute.

Table 23 - Explicit messaging connection object attribute access

Explicit messaging connection state		
Attribute	Non-existent	Established
state	Not available	Get only
instance_type	Not available	Get only
transportClass_trigger	Not available	Get only
produced_connection_id	Not available	Get only
consumed_connection_id	Not available	Get only
initial_comm_characteristics	Not available	Get only
produced_connection_size	Not available	Get only
consumed_connection_size	Not available	Get only
expected_packet_rate <sup>1)</sup>	Not available	Get/set <sup>1)</sup>
watchdog_timeout_action	Not available	Get only
produced_connection_path_length	Not available	Get only
produced_connection_path	Not available	Get only
consumed_connection_path_length	Not available	Get only
consumed_connection_path	Not available	Get only
production_inhibit_time	Not available	Get only

1) When a connection object is in the established state, any modifications to the expected\_packet\_rate attribute have immediate effect on the inactivity/watchdog timer. The following steps are performed by a connection object in the established state when a request is received to modify the expected\_packet\_rate attribute:

- the current inactivity/watchdog timer is cancelled
- a new inactivity/watchdog timer is activated based on the new value in the expected\_packet\_rate attribute.

### 5.3.2.9 Dynamic management of message IDs

Dynamically establishing both I/O and explicit messaging connections requires all end-points to implement internal message ID allocation procedures. These procedures shall also mark a previously allocated message ID as available when that message ID is no longer in use.

To reduce the likelihood of CID errors, the following rules shall be followed:

- a) The CID allocation process shall ensure that no two nodes on DeviceNet have configurations that allow transmission of an identical bit pattern within the CAN identifier field.
- b) The CID deallocation process shall ensure that all end-points of a connection have timed out prior to reusing a message ID.
  - If the message ID is associated with a connection that activates an inactivity/watchdog timer, then a new inactivity/watchdog timer is activated. Upon expiration of this timer, the message ID is marked as available.
  - If the message ID is associated with a connection that does not activate an inactivity/watchdog timer, then the message ID can be immediately marked as available.

**5.3.3 DeviceNet object class definition (Class ID code: 3)**

**5.3.3.1 General**

The DeviceNet object is used to provide the configuration and status of a physical attachment to DeviceNet. A product shall support one DeviceNet object per physical network attachment.

**5.3.3.2 DeviceNet object class attributes**

The class attributes for the DeviceNet object are defined in Table 24.

**Table 24 - Class attributes for the DeviceNet object**

Attribute ID	Need in implementation	Access rule	Attribute name	Data type	Attribute description	Value
1	Required	Get	revision	UINT	Revision of the object.	65 5352

**5.3.3.3 DeviceNet object class common services**

The DeviceNet object class shall support the DeviceNet common services shown in Table 25.

**Table 25 - DeviceNet object class common services**

Need in Implementation	Service name	Service description
Required	Get_attribute_single	Reads a DeviceNet object class attribute
Service codes are defined in Annex A.		

### 5.3.3.4 DeviceNet object instance attributes

Table 26 defines the instance attributes for the DeviceNet object.

**Table 26 - Instance attributes for the DeviceNet object**

Attribute ID	Need in Implementation	Access rule	Name	Data type	Brief description of attribute	Values
1	Optional	Get/set	MAC ID	USINT	Node address	Range 0-63
2	Optional	Get/set	Bit rate	USINT	Bit Rate	Range 0-2
3	Optional	Get/set	BOI	BOOL	Bus-Off Interrupt	
4	Optional	Get/set	Bus-off counter	USINT	Number of times CAN went to the bus-off state	Range 0-255
5	Optional <sup>1)</sup>	Get	Allocation information	STRUCT		
			Allocation choice byte	BYTE	See 5.3.3.5.2.1.2	
			Master's MAC ID	USINT	MAC ID of master (from allocate)	Range 0-63 and 255. Modified via allocate only.
6	Optional	Get	MAC ID switch changed	BOOL	Node address switch(es) have changed since last power-up/reset.	0 = No change 1 = Change since last reset or power-up.
7	Optional	Get	Bit rate switch changed	BOOL	Bit rate switch(es) have changed since last power-up/reset.	0 = No change 1 = Change since last reset or power-up.
8	Optional	Get	MAC ID switch value	USINT	Actual value of node address switch(es)	Range 0-63
9	Optional	Get	Bit rate switch value	USINT	Actual value of bit rate switch(es)	Range 0-2
1) The allocation information attribute shall be supported if the predefined master/slave connection set is supported.						

The following attributes are described below:

#### 1) MAC ID

This attribute contains the MAC ID of this device. A device that uses switches which are user accessible when the product is installed to set the MAC ID shall return an error response whose general error code is set to 0E<sub>hex</sub> (Attribute not settable) in response to a set\_attribute\_single request specifying the MAC ID attribute. The MAC ID attribute is considered non-volatile in that once configured the attribute shall be remembered after a power cycle or device reset. The default value of the MAC ID shall be 63.

Modification of the MAC ID requires that a device deletes all connection objects and re-executes the network access state machine defined in 5.4.

#### 2) Bit rate

The bit rate attribute indicates the selected bit rate. Values are shown in Table 27.

**Table 27 - Bit rate attribute values**

Value	Meaning
0	125 kBit/s
1	250 kBit/s
2	500 kBit/s
3-255	Reserved

A device that uses switches which are user accessible when the product is installed to set the bit rate shall return an error response whose general error code is set to 0E<sub>hex</sub> (attribute not settable) in response to a set\_attribute\_single request specifying the bit rate attribute. The bit rate attribute is considered non-volatile in that once configured the attribute shall be remembered after a power cycle or device reset. The default bit rate shall be 125 kBit/s.

Modification of the bit rate shall not take effect until the device is either physically reset or reset by sending the reset service to the identity object.

### 3) **Bus-Off Interrupt** (as described in Table 28)

The bus-off interrupt attribute consists of one bit that defines how a CAN device processes the bus-off interrupt.

**Table 28 - BOI attribute values**

Value	Meaning
FALSE	Hold the CAN chip in its bus-off (reset) state upon detection of a bus-off indication.
TRUE	If possible, fully reset the CAN chip and continue communicating upon detection of a bus-off indication

When the bus-off interrupt attribute is FALSE (set to 0) and a CAN chip bus-off event is detected:

- the transceiver shall be held in its reset/bus-off state;
- the device shall enter the communication fault state (see 5.4)

When the bus-off interrupt attribute is TRUE (set to 1) and a CAN chip bus-off event is detected, it may be possible to return the MAC and transceiver implementation to its normal operating mode and continue communicating based on the network access state machine detailed in 5.4.

If the bus-off interrupt attribute is not supported, a device shall implement the behaviour indicated by attribute value FALSE in Table 28.

### 4) **Bus-Off counter**

The bus-off counter is initialised to 0 at power-up or device initialisation.

The bus-off counter stops counting when it reaches maximum count.

The DeviceNet object resets the bus-off counter to 0 whenever it receives a `set_attribute_single` request specifying the bus-off counter attribute. The `set_attribute_single` data is not used and can be any value.

### 5) **Allocation information**

The allocation information attribute shall be supported if the predefined master/slave connection set is supported. It indicates whether or not the predefined master/slave connection set defined in 5.5 has been allocated. If it has been allocated, then this attribute indicates the device that has performed the allocation and the connection(s) that are currently allocated.

This attribute is modified when a success response associated with an `allocate_master/slave_connection_set` service (defined in 5.5) is generated. This attribute is not modifiable by the `set_attribute_single` service. An error response whose general error code field is set to `0Ehex` (attribute not settable) is returned if a `set_attribute_single` request specifies this attribute.

The allocation information attribute consists of the following:

#### — **Allocation choice byte**

The allocation choice byte indicates which of the predefined master/slave connections are active (in the configuring, or established state). Its format is specified in 5.3.3.5.2.1.2.

The allocation choice byte is initialised to 00 at device power-up or reset.

#### — **Master's MAC ID**

The master's MAC ID contains the MAC ID of the device that has allocated the predefined master/slave connection set via the `allocate_master/slave_connection_set` service. This contains the allocator's MAC ID field copied from the `allocate_master/slave_connection_set` request.



The range of values is 0-63 and 255 decimal. A value in the range 0-63 indicates that the predefined master/slave connection set is currently allocated and denotes the MAC ID of the device that performed the allocation. The value 255 means the predefined master/slave connection set has not been allocated.

The master's MAC ID attribute is initialised to 255 at device power-up/reset.

### 5.3.3.5 DeviceNet object instance services

The subclauses that follow describe the common services and object class specific services supported by the DeviceNet object instance.

#### 5.3.3.5.1 Common services

The DeviceNet object instance shall support the common services shown in Table 29.

**Table 29 - DeviceNet object instance common services**

Need in Implementation	Service name	Service description
Required	Get_attribute_single	Reads a DeviceNet object attribute value.
Optional	Set_attribute_single	Modifies a DeviceNet object attribute value.
Service codes are defined in Annex A.		

#### 5.3.3.5.2 Object class specific services

The DeviceNet object instance shall support the object class specific services shown in Table 30.

**Table 30 - DeviceNet object class specific services**

Service code	Need in implementation	Service name	Service description
4B <sub>hex</sub>	Optional	Allocate_master/slave_connection_set	Requests the use of the predefined master/slave connection set.
4C <sub>hex</sub>	Optional	Release_master/slave_connection_set	Indicates that the specified connections within the predefined master/slave connection set shall be deleted

These services are used to allocate and deallocate the predefined master/slave connection set described in 5.5.

A device that behaves as the client across the predefined master/slave connection set is referred to as a master. A device that behaves as the server across the predefined master/slave connection set is referred to as a slave.

Before a device can function as a master it shall first allocate the predefined master/slave connection set within the slave. Only one master shall have the predefined master/slave connection set allocated at any given time. When a master no longer requires its slave, it shall release all connections, causing the slave to deallocate the predefined master/slave connection set.

#### 5.3.3.5.2.1 Allocate\_master/slave\_connection\_set (Service code: 4B<sub>hex</sub>)

##### 5.3.3.5.2.1.1 General

This service allocates the predefined master/slave connection set. General error codes are defined in Annex B. DeviceNet object specific additional error code values are defined in 5.3.3.6. This service shall be transmitted using either the unconnected explicit request message of the master/slave connection set (see 5.5.2), or an explicit messaging connection.

The allocate\_master/slave\_connection\_set service performs the following:

- Connection object create;
- Connection object configure.

See 9.3.4 and 9.3.5 for test specifications regarding the allocation of master/slave connection set for both explicit and I/O messaging connections.

**5.3.3.5.2.1.2 Request service data field parameters**

The information in Table 31 is specified within the service data field of an allocate\_master/slave\_connection\_set request.

**Table 31 - Allocate\_master/slave\_connection\_set request service data field parameters**

Name	Data type	Description of parameter
Allocation choice	BYTE	Indicates which connections from the predefined master/slave connection set are to be allocated/configured for use by the master.
Allocator's MAC ID	USINT	Contains the MAC ID associated with the node requesting the allocation.

The allocation choice parameter is specified within a single byte (see Figure 26). Each bit denotes an explicit message and/or I/O connection(s) from the predefined master/slave connection set that are to be allocated, or in the case of acknowledge suppression, a command. If the bit is set to 1, then a request is being made to allocate the corresponding connection. If a bit is set to 0, then the requester does not want to allocate the corresponding connection.

7	6	5	4	3	2	1	0
Reserved	Acknowledge Suppression	Cyclic	Change of State	Reserved	Bit Strobed	Polled	Explicit Message

**Figure 26 - Allocation choice byte contents**

Bits 3 and 7 shall be set to 0 and the slave shall verify this requirement when the allocate\_master/slave\_connection\_set request is received.

Figure 27 shows the format of this explicit message.

		Contents								
Byte Offset		7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID							
1	R/R [0]	Service [4B <sub>hex</sub> ]								
		Class ID [3]								
		Instance ID [1]								
		Allocation Choice								
		0	0	Allocator's MAC ID						

**Figure 27 - Allocate\_master/slave\_connection\_set request message**

Allocate\_master/slave\_connection\_set request message contents:

- **Frag (0)/transaction ID/MAC ID:** As defined in 5.2.1.2.
- **R/R bit (0):** Indicates this is a request message.
- **Service code (4B<sub>hex</sub>):** Identifies this message as an allocate\_master/slave\_connection\_set service.
- **Class ID:** Defines the object class to which this request is directed. When this message is transmitted across the unconnected explicit request message port of the master/slave connection set (see 5.5.2), the class ID value is specified within an 8 bit integer field. The class ID shall be set to 3.

- **Instance ID:** Defines the particular instance within the object class to which this request is directed. When this message is transmitted across the unconnected explicit request message port of the master/slave connection set (see 5.5.2), the instance ID value is specified within an 8 bit integer field. The instance ID shall be set to 1.
- **Allocation Choice:** Specified by the byte following the instance ID field.
- **Allocator's MAC ID:** Specified within the byte following the allocation choice field.

**5.3.3.5.2.1.3 Success response service data field parameters**

The information contained within the service data field of a successful allocate\_master/slave\_connection\_set response is described in Table 32.

**Table 32 - Allocate\_master/slave\_connection\_set response parameters**

Name	Data type	Description of parameter
Message body format	Defined below	This is semantically equivalent to the actual message body format parameter returned with an open explicit message connection response message (as described in 5.2.1.6). This argument is significant when the allocate_master/slave_connection_set request was received across the unconnected explicit request message port of the master/slave connection set (see 5.5.2). It indicates the message body format associated with subsequent messages transmitted across the explicit messaging connection within the predefined master/slave connection set. If the allocate_master/slave_connection_set request was received across an explicit messaging connection within a UCMM capable device, then this parameter is set to the actual message body format associated with that explicit messaging connection.

Figure 28 shows the format of a success response to the allocate\_master/slave\_connection\_set request.

Contents

Byte Offset	7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID					
1	R/R [1]	Service Code [4B <sub>hex</sub> ]						
2	Reserved (All Bits = 0)				Message Body Format			

**Figure 28 - Success response to allocate\_master/slave\_connection\_set request**

Allocate\_master/slave\_connection\_set request success response contents:

- **Frag (0)/transaction ID/MAC ID:** As defined in 5.2.1.2.
- **R/R bit (1):** Indicates this is a response message.
- **Service code (4B<sub>hex</sub>):** Identifies this message as an allocate\_master/slave\_connection\_set service.
- **Reserved bits:** These bits are not used by the receiver of the response and shall be set to 0 by the transmitter of the response message.
- **Message body format:** as described in Table 32.

#### 5.3.3.5.2.2 Allocate\_master/slave\_connection\_set server behaviour

1. If an error is encountered, then none of the requested connections shall be allocated. If this request cannot be fully serviced, then none of the requested allocations shall take place.
2. If the receiving device does not support the predefined master/slave connection set, then an error response is returned. The general error code within the error response shall be set to 08 to indicate service not supported.
3. The receiving device (slave) validates the allocator's MAC ID parameter within the request as follows:
  - If the predefined master/slave connection set is allocated and this request is not from the current master, then the slave returns an error. The general error code within the error response shall be set to 0C<sub>hex</sub>, with the additional error code set to an object specific value of 01.
  - If the predefined master/slave connection set is not allocated, then the slave validates the allocation choice parameter as specified in (d).
  - If the predefined master/slave connection set is allocated and this request is from the current master, then the slave shall validate the allocation choice parameter as specified in (d).
4. The slave shall validate the allocation choice parameter within the request. If the slave does not support one of the connections specified in the allocation choice argument, then an error response shall be returned. The general error code within the error response shall be set to 02, with the additional error code set to an object specific value of 02.

If any of the connection(s) being requested are supported by this slave and have already been allocated to the master denoted by the allocator's MAC ID argument, the slave shall return an error response with the general error code set to 0B<sub>hex</sub>, with the additional error code set to an object specific value of 02. If the requested I/O connection is in the timed-out state the slave shall reallocate the I/O connection, setting it to the configuring state.

If the allocation choice byte has no bits set the slave shall return an error response with the general error code set to 09, with the additional error code set to an object specific value of 02.

If a resource that is required for use with the requested connections is not available, then an error response shall be returned with the general error code set to 02, and the additional error code set to an object specific value of 04.

The change of state and cyclic allocation choices are mutually exclusive. If an allocation request would result in both the cyclic and change of state bits being set, an error response shall be returned. The general error code within the error response shall be set to 09 (invalid attribute value) with an additional error code of 02.

If a master has allocated the change of state/cyclic connection set, and a subsequent allocation request is received with the polled allocation bit set, an error shall be returned. The general error code within the error response shall be set to 02 to indicate resource unavailable. If the allocation choice byte has the acknowledge suppression bit set and neither the change of state bit nor the cyclic bit are set, an error response shall be returned. The general error code within the error response shall be set to 09 with an additional error code of 02.

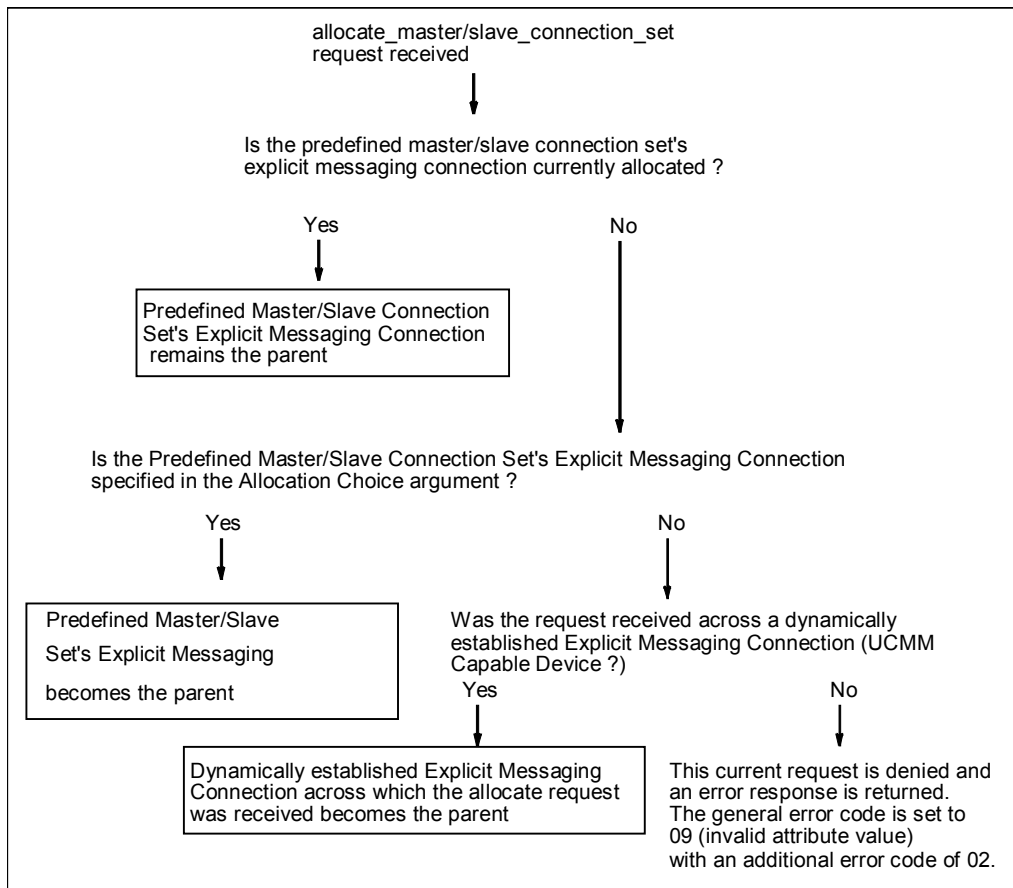
5. The slave notes the fact that the predefined master/slave connection set has been allocated to the MAC ID within the allocator's MAC ID field by updating the DeviceNet object's allocation information attribute. If necessary, the `produced_connection_id` and/or `consumed_connection_id` attributes of the connection object(s) may now be initialised.

The allocation choice byte of the allocation information attribute indicates which connection objects from the predefined master/slave connection set are active (in the configuring, or established state). This byte is updated whenever a master/slave connection object changes state.

6. Any allocated I/O connection(s) transition to the configuring state. With respect to the predefined master/slave connection objects, an implied `apply_attributes` service accompanies a `set_attribute_single` of the `expected_packet_rate` attribute while the connection is in the configuring state. A `set_attribute_single` of the `expected_packet_rate` attribute triggers the execution of the steps performed with an `apply_attributes` service and causes the predefined master/slave I/O connection object to transition to the established state. If an error is encountered when performing the `apply_attributes` portion, then the `expected_packet_rate` attribute shall be reset to its previous value, and an error shall be returned whose general error code is set to 09 and whose additional error code is set to the offending connection object's attribute ID.
7. If allocated, the explicit messaging connection transitions to the established state. The inactivity/watchdog timer is started using the initial value specified in 5.3.2.5.3.

UCMM capable devices which support the predefined master/slave connection set shall support the allocation and use of the predefined master/slave connection set explicit messaging connection.

8. Either the master/slave connection set's explicit messaging connection or a dynamically established explicit messaging connection acts as the allocated I/O connection(s) parent until the I/O connection(s) are in the established state. The term parent is used to indicate that if the explicit messaging connection is deleted and none of the allocated I/O connections exist in the established state, then the predefined master/slave connection set is automatically released by the slave. The logic describing which explicit messaging connection is to act as the parent is defined in Figure 29.



**Figure 29 - Parent explicit messaging connection logic**

The slave automatically releases the predefined master/slave connection set and all connections return to the non-existent state if all of the following conditions are true:

- if none of the connection objects associated with the predefined master/slave connection set exist in the established state;
- if the parent explicit messaging connection object is not in the established state.

### 5.3.3.5.2.3 Release\_master/slave\_connection\_set (Service code: 4C<sub>hex</sub>)

#### 5.3.3.5.2.3.1 General

This service is used to deallocate the predefined master/slave connection set within a slave. General error codes are defined in Annex B. DeviceNet object specific additional error code values are defined in 5.3.3.6. This service can be transmitted across the unconnected explicit request message port of the master/slave connection set (see 5.5.2) as well as an explicit messaging connection.

#### 5.3.3.5.2.3.2 Request service data field parameters

The information in Table 33 is specified within the service data field of a release\_master/slave\_connection\_set request.

**Table 33 - Release\_master/slave\_connection\_set request service data field parameters**

Name	Data type	Description of parameter
Release choice	BYTE	Indicates which predefined master/slave connections are to be released.

The release choice parameter is specified within a single byte (see Figure 30). Each bit denotes an explicit message and/or I/O connection(s) to be released. If the bit is set to 1, then a request is being made to release the corresponding connection. If a bit is set to 0, then the requester does not want to release the corresponding connection.

7	6	5	4	3	2	1	0
Reserved	Ignored	Cyclic	Change of state	Reserved	Bit strobed	Polled	Explicit message

**Figure 30 - Release choice byte contents**

Bits 3 and 7 shall be set to 0 and the slave shall verify this requirement when the release\_master/slave\_connection\_set request is received.

A value of 00 is invalid.

Figure 31 shows the format of this explicit message.

		Contents							
Byte Offset		7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID						
1	R/R [0]	Service [4C <sub>hex</sub> ]							
		Class ID [3]							
		Instance ID [1]							
		Release Choice							

**Figure 31 - Release\_master/slave\_connection set request message**

Release\_master/slave\_connection set request message contents:

- **Frag (0)/transaction ID/MAC ID:** See 5.2.1.2.
- **R/R bit (0):** Indicates this is a request message.
- **Service code (4C<sub>hex</sub>):** Identifies this message as a release\_master/slave\_connection\_set service.
- **Class ID:** Defines the object class to which this request is directed. When this message is transmitted across the unconnected explicit request message port of the master/slave connection set (see 5.5.2), the class ID value is specified within an 8 bit integer field. The class ID shall be set to 3.
- **Instance ID:** Defines the particular instance within the object class to which this request is directed. When this message is transmitted across the unconnected explicit request message port of the master/slave connection set (see 5.5.2), the instance ID value is specified within an 8 bit integer field. The instance ID shall be set to 1.
- **Release choice:** Specified within the byte following the instance ID field.

### 5.3.3.5.2.3.3 Success response message

Figure 32 shows the format of a success response to the release\_master/slave\_connection\_set request.

		Contents							
Byte Offset		7	6	5	4	3	2	1	0
0	Frag [0]	XID	MAC ID						
1	R/R [1]	Service [4C <sub>hex</sub> ]							

**Figure 32 - Success response to release\_master/slave\_connection\_set request**

Release\_master/slave\_connection\_set request success\_response contents:

- **Frag (0)/transaction ID/MAC ID:** See 5.2.1.2.

- **R/R bit (1):** Indicates this is a response message.
- **Service code (4C<sub>hex</sub>):** Identifies this as a release\_master/slave\_connection\_set service.

#### 5.3.3.5.2.4 Release\_master/slave\_connection\_set server behaviour

If an error is encountered, then none of the specified connections are released. If this request cannot be fully serviced, then none of the requested releases take place.

If the receiving device does not support the predefined master/slave connection set, then an error response shall be returned. The general error code within the error response shall be set to 08.

The receiving device (slave) shall validate the release choice parameter within the request. If the slave does not support one of the connections specified in the release choice argument, then an error response shall be returned. The general error code within the error response shall be set to 02, with the additional error code set to an object specific value of 02.

If the release choice byte has no bits set (all bits are 0) the slave shall return an error response with the general error code set to 09 (invalid attribute value), with the additional error code set to an object specific value of 02.

If one of the specified connections is in the non-existent state, then an error response shall be returned. The general error code within the error response shall be set to 0B<sub>hex</sub>.

The slave shall verify that it is in a state that allows it to discontinue use of the specified connection(s). If this is not the case, then an error response shall be returned. The general error code within the error response shall be set to 0C<sub>hex</sub>.

If the request is valid, then the slave shall release all resources associated with the specified connection(s). If this results in all predefined master/slave connections being released, the slave shall note the fact that the predefined master/slave connection set is no longer allocated by updating the allocation information attribute.

The slave shall not check to see that the release request came from its master.

If this request has resulted in none of the predefined master/slave connections existing in the established state, then the slave shall release the predefined master/slave connection set and all connections shall return to the non-existent state.

#### 5.3.3.6 Error codes specific to the DeviceNet object

Table 34 lists additional error codes specific to the DeviceNet object.

**Table 34 - DeviceNet object specific additional error codes**

Value	Meaning
01	Predefined master/slave connection set allocation conflict. This is returned when an allocate_master/slave_connection_set request is received and the slave has already allocated the predefined master/slave connection set to another master.
02	Invalid allocation/release choice parameter. This is returned when an allocate/release_master/slave_connection_set request is received and: <ol style="list-style-type: none"> <li>1) The slave does not support the choice specified in the choice parameter.</li> <li>2) The slave was asked to allocate/release connection(s) already allocated/released.</li> <li>3) The allocation choice/release byte contained all zeros, an invalid combination of bits, or did not contain the explicit message allocation choice when required.</li> </ol>
03	A server that does not support UCMM received a message that was not an allocate or release message on the unconnected explicit request message port of the master/slave connection set (see 5.5.2).
04	Resource required for use with the predefined master/slave connection set is unavailable

### 5.4 Network access state machine

#### 5.4.1 General

This subclause defines the network access state machine that every DeviceNet node shall implement. The network access state machine is described by the following:



- Tasks that shall be performed prior to communicating on the network;
- Network events that affect a node's ability to communicate on the network.

### 5.4.2 State transition diagram and event matrix

Figure 33 provides a general overview of the network access state machine.

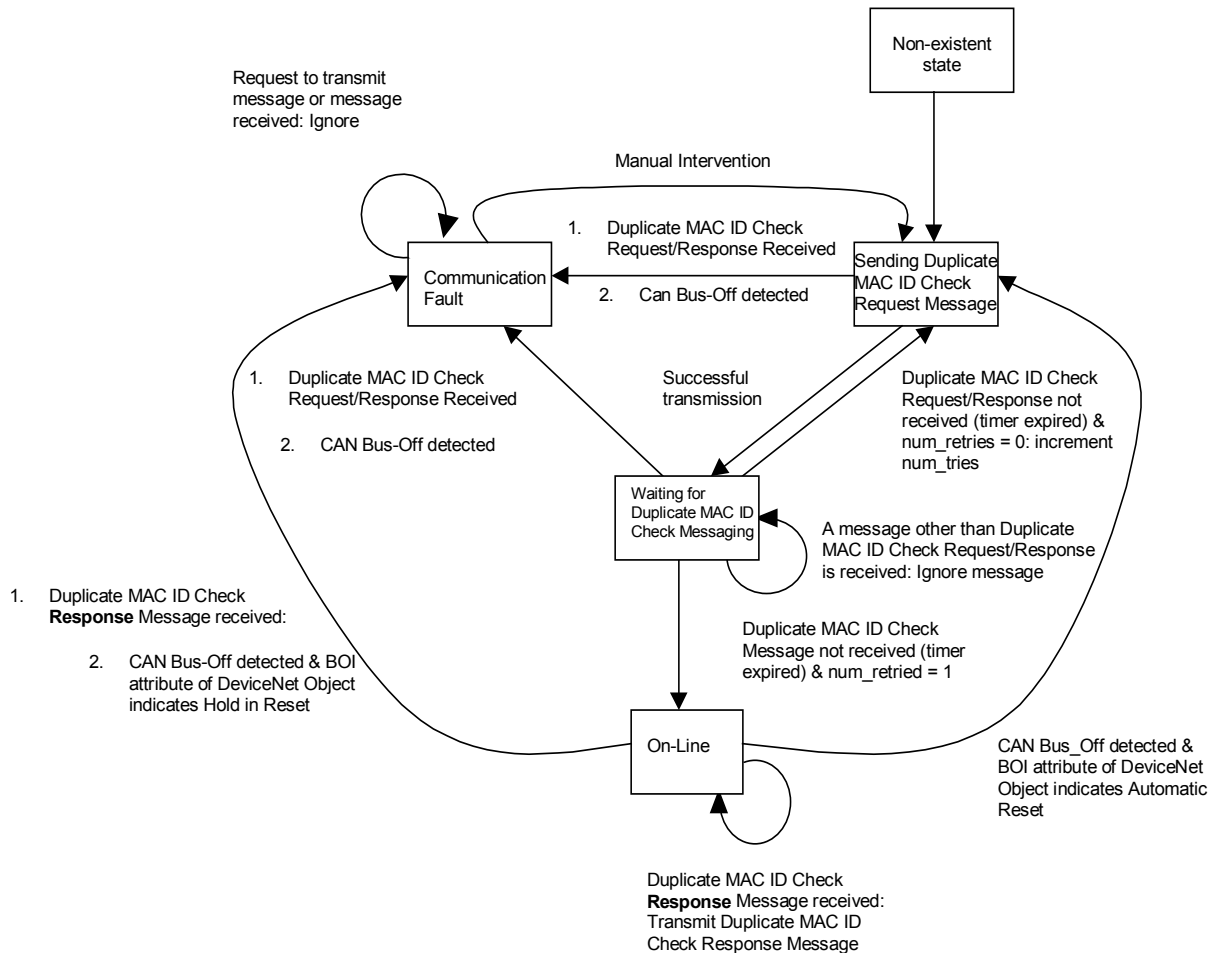


Figure 33 - Network access state transition diagram

Table 35 provides a detailed state event matrix for the network access state machine.

**Table 35 - Network access state event matrix**

Event	State			
	Sending duplicate MAC ID check request	Waiting for duplicate MAC ID check message	On-line	Communication fault
Successful transmission of the duplicate MAC ID check request message	Activate 1s timer <sup>1)</sup> . Transition to waiting for duplicate MAC ID check message.	Not applicable	Not applicable	Not applicable
Bus-off detected	Transceiver held in reset. Transition to the communication fault state.	Transceiver held in reset. Transition to the communication fault state	Access the DeviceNet object's BOI attribute. If the BOI attribute indicates that the transceiver shall be held in reset, then transition to communication fault. If the BOI attribute indicates that the MAC and transceiver shall be automatically reset, then: 1. reset, 2. request the transmission of the duplicate MAC ID check request message, and 3. transition to the sending duplicate MAC ID check request state.	Not applicable
Duplicate MAC ID check request message received	Duplicate MAC ID Detected. Transition to the communication fault state	Duplicate MAC ID Detected. Transition to the communication fault state	Transmit the duplicate MAC ID check response message	Discard message
Duplicate MAC ID check response message received	Duplicate MAC ID detected. Transition to the communication fault state.	Duplicate MAC ID detected. Transition to the communication fault state.	Duplicate MAC ID detected. Transition to the communication fault state.	Discard message
1 s duplicate MAC ID check message timer expires <sup>1)</sup>	Not applicable	If this is the first time-out, then request the transmission of the duplicate MAC ID check request message again and transition to sending duplicate MAC ID check request. If this is the second consecutive time-out, then transition to on-line.	Not applicable	Not applicable
Internal message transmission request	Return internal error	Return internal error	Transmit message	Return internal error
A message other than duplicate MAC ID check request/response is received	Discard message	Discard message	Process the received message as appropriate	Discard message
1) Valid timer range is 0,9 s to 1,5 s.				

The CAN fault confinement state machine considers the possibility that during system start-up /wake-up), only one node may be present on the network. If this node transmits a message, it will experience an acknowledgement error and will automatically repeat the message. In this situation the node will transition to error passive but not bus-off. For this reason, the only event that signifies an unsuccessful transmission of a duplicate MAC ID check message (request or response) is the bus-off event. If an error passive or error warning indication is received during the transmission of a duplicate MAC ID check message, then it shall be ignored, as it has no effect on the duplicate MAC ID detection state machine.

### 5.4.3 Duplicate MAC ID detection

The main step involved in the network access state machine is the execution of a duplicate MAC ID detection algorithm. Each node on DeviceNet is assigned a unique MAC ID, and to protect against errors, all DeviceNet nodes shall participate in the duplicate MAC ID detection algorithm.

NOTE The protocol associated with the duplicate MAC ID detection algorithm is described in 5.2.4.

As stated in 5.2.4, a special message within message Group 2 is defined for performing duplicate MAC ID detection.

A DeviceNet node shall receive and process any duplicate MAC ID check message that contains its MAC ID in the message Group 2 identifier field.

After transmitting a duplicate MAC ID check request message, a module shall wait 1 s before timing out and taking the appropriate action defined by the network access state machine.

The duplicate MAC ID check request message shall be transmitted twice without receiving a subsequent duplicate MAC ID check request or response message before transitioning to on-line.

See 9.2.3 and 9.3.2 for test specifications regarding the Power ON behaviour and handling of the Duplicate MAC ID mechanisms.

## 5.5 Predefined master/slave connection set

### 5.5.1 General

The preceding subclauses present the general rules for establishing connections between devices. The general rules call for the utilisation of an explicit messaging connection to create and configure connection objects within each connection end-point. This subclause uses the general rules as a basis for the definition of a set of connections, which facilitate communications typically seen in a master/slave relationship. These connections are referred to collectively as the predefined master/slave connection set.

- **Group 2 server:** An unconnected message manager (UCMM) capable device that has been configured to act as the server for the predefined master/slave identifier connections.
- **Group 2 client:** A UCMM capable device that has gained ownership of the predefined master/slave connection set within a server such that it may act as the client on those connections.
- **UCMM capable device:** A device that supports the UCMM.
- **UCMM incapable device:** A device that does not support the UCMM.
- **Group 2 only server:** A 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.
- **Group 2 only client:** A 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 it has allocated.

### 5.5.2 Predefined master/slave connection set messages

The CAN identifier fields associated with the predefined master/slave connection set are shown in Table 36 together with the identifiers that shall be used with all connection based messaging used in the predefined master/slave connection set.

**Table 36 - Predefined master/slave connection set identifier fields**

Identifier bits											Usage	Range	
10	9	8	7	6	5	4	3	2	1	0			
0	Group 1 message ID				Source MAC ID						Group 1 messages	000-3FF	
0	1	1	0	1	Source MAC ID						Slave's I/O change of state or cyclic message		
0	1	1	1	0	Source MAC ID						Slave's I/O bit-strobe response message		
0	1	1	1	1	Source MAC ID						Slave's I/O poll response or change of state/cyclic acknowledge message		
1	0	MAC ID				Group 2 message ID			Group 2 messages				400-5FF
1	0	Source MAC ID				0	0	0	Master's I/O bit-strobe command message				
1	0	Source MAC ID				0	0	1	Reserved				
1	0	Destination MAC ID				0	1	0	Master's change of state or cyclic acknowledge message				
1	0	Source MAC ID				0	1	1	Slave's explicit response messages				
1	0	Destination MAC ID				1	0	0	Master's explicit request messages				
1	0	Destination MAC ID				1	0	1	Master's I/O poll command/change of state/cyclic message				
1	0	Destination MAC ID				1	1	0	Group 2 only unconnected explicit request messages				
1	0	Destination MAC ID				1	1	1	Duplicate MAC ID check messages				

The following types of messages are included in Table 36:

- **I/O bit-strobe command/response messages:** The bit-strobe command is an I/O message that is transmitted by the master. Multiple slaves may receive and react to the same bit-strobe command. The bit-strobe response is an I/O message that a slave transmits back to the master when the bit-strobe command is received.
- **I/O poll command/response messages:** The poll command is an I/O message that is transmitted by the master. A poll command is directed to a specific slave. The poll response is an I/O message that a slave transmits back to the master when the poll command is received.
- **I/O change of state/cyclic messages:** The change of state/cyclic message is transmitted by either the master or the slave. A change of state/cyclic message is directed to a specific node. An acknowledge message shall be returned in response to this message unless the configuration includes suppression of acknowledge messages.
- **Explicit response/request messages:** See 5.2.1.6.
- **Group 2 only unconnected explicit request messages:** The Group 2 only unconnected explicit request port is used to allocate/release the predefined master/slave connection set.
- **Duplicate MAC ID check message:** See 5.2.4.

### 5.5.3 Slave connection object characteristics

#### 5.5.3.1 General

This subclause presents the externally visible characteristics of the connection objects associated with the predefined master/slave connection set within slave devices. The predefined master/slave connection objects described for slave devices are:

- **The bit-strobe connection:** responsible for receiving the master's bit-strobe command and returning the associated bit-strobe response.
- **The poll connection:** responsible for receiving the master's poll command and returning the associated poll response.
- **The explicit messaging connection:** responsible for the reception of explicit requests and returning associated responses.
- **The change of state/cyclic connection:** responsible for the sending the change of state/cyclic message and receiving the acknowledge response if not suppressed.

This subclause gives additional information for connection objects within the predefined master/slave connection set. Except where noted, all information specified in 5.3 applies to the connection objects described in this subclause.

### 5.5.3.2 Connection instance IDs

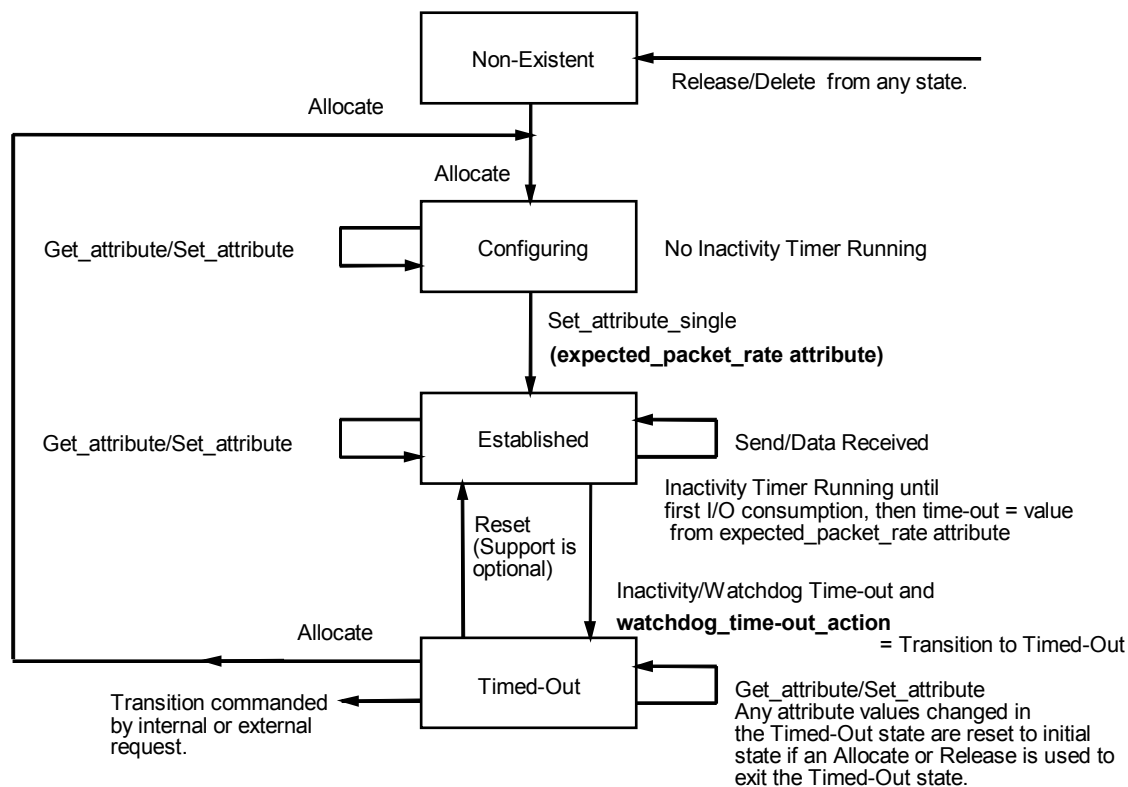
Every existing connection object has an assigned connection instance ID which identifies the connection object within the connection class. The connection instance IDs that shall be used by a slave device to identify predefined master/slave connection objects are shown in Table 37.

**Table 37 - Connection instance IDs for predefined master/slave connections**

Connection instance ID	Description
1	Designates the explicit messaging connection into the server
2	Designates the poll I/O connection
3	Designates the bit-strobe I/O connection
4	Designates the slave's change of state or cyclic I/O connection

### 5.5.3.3 Predefined master/slave connection instance behaviour

Figure 34 illustrates the predefined master/slave connection set I/O connection object state transition diagram.



The allocate and release services reset the connection instance. All connection object attributes are reset to their default values.

**Figure 34 - Predefined master/slave I/O connection state transition diagram**

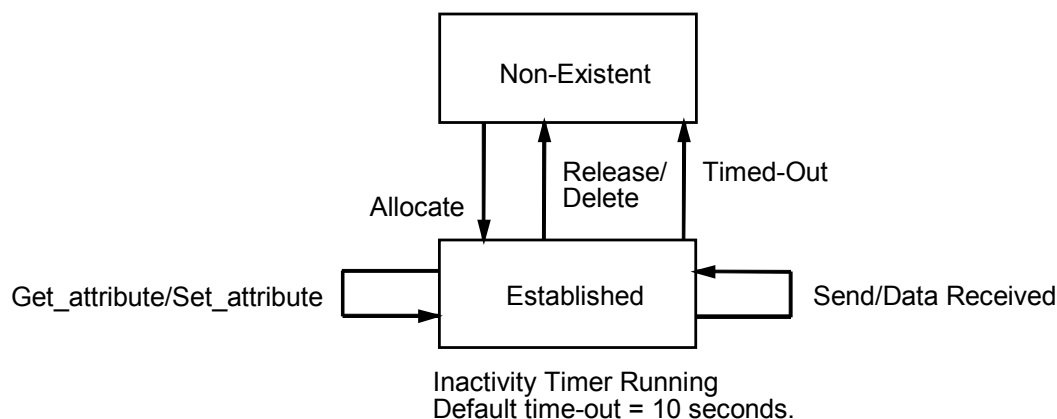
For attribute modification, predefined master/slave I/O connections shall support at least the modification of the expected\_packet\_rate attribute.

The state event matrix presented below provides a formal definition of the behaviour of I/O connections within the predefined master/slave connection set. This state event matrix inherits from and/or overrides actions presented in the I/O connection object state event matrix in 5.3.

**Table 38 - Predefined master/slave I/O connection state event matrix**

I/O connection object state				
Event	Non-existent	Configuring	Established	Timed out
DeviceNet object receives an allocate master/slave connection set request that passes all error checks specified in 5.3.3.5.2.2. This request specifies one of the predefined master/slave I/O connections.	Create a connection object for each requested I/O connection and set attributes to default values specified in 5.3.3.5.2.1. Transition to configuring.	This is an error. See 5.3.3.5.2.2.	This is an error. See 5.3.3.5.2.2.	Set attributes to default values specified in 5.3.3.5.2.1. Transition to configuring.
Connection class receives a delete request or the UCMM receives a close request and the request specifies a predefined master/slave I/O connection object.	As described in Table 20 .	Release all associated resources. Transition to non-existent. <sup>1)</sup>	Release all associated resources. Transition to non-existent. <sup>1)</sup>	Release all associated resources. Transition to non-existent. <sup>1)</sup>
DeviceNet object receives a release master/slave connection set request that passes all error checks specified in 5.3.3.5.2.2. This request specifies one of the predefined master/slave I/O connections.	This is an error. See 5.3.3.5.2.2.	Release all associated resources. Transition to non-existent. <sup>1)</sup>	Release all associated resources. Transition to non-existent. <sup>1)</sup>	Release all associated resources. Transition to non-existent. <sup>1)</sup>
Set_attribute_single	As described in Table 20 .	Validate/service the request according to the access rules presented in 5.5.3.4. If this is a valid request to set the expected_packet_rate attribute, then perform the steps specified in Table 20 under the Apply_Attributes event in the configuring state and transition to established. return appropriate response.	Validate/service the request according to the access rules presented in 5.5.3.4. Return appropriate response.	Validate/service the request according to the access rules presented in 5.5.3.4. Return appropriate response.
Get_attribute_single	As described in Table 20 .	Validate/service the request according to the access rules presented in 5.5.3.4. Return appropriate response.	Validate/service the request according to the access rules presented in 5.5.3.4. Return appropriate response.	Validate/service the request according to the access rules presented in 5.5.3.4. Return appropriate response
Reset	As described in Table 20 .	As described in Table 20 .	As described in Table 20 .	As described in Table 20 .
Apply_attributes	As described in Table 20 .	As described in Table 20 .	As described in Table 20 .	As described in Table 20 .
Receive_data	As described in Table 20 .	As described in Table 20 .	As described in .	As described in Table 20 .
Send_message	As described in Table 20 .	As described in Table 20 .	As described in Table 20 .	As described in Table 20 .
Inactivity/watchdog timer expires	As described in Table 20 .	As described in Table 20 .	As described in Table 20 . <sup>1)</sup>	As described in Table 20 .
<p>1) Whenever a connection object within the predefined master/slave connection set transitions out of the established state, the entire predefined master/slave connection set may need to be automatically released. When the predefined master/slave connection set is released, all associated connection objects return to the non-existent state. See 5.3.3.5.2.3 for more details concerning the automatic release of the predefined master/slave connection set.</p> <p>2) Since an implied apply_attributes accompanies a set_attribute_single of the expected_packet_rate attribute of a predefined master/slave I/O connection object in the configuring state, the only time an apply_attributes explicit messaging request will succeed is when the expected_packet_rate has yet to be successfully modified via the set_attribute_single request and, thus, still contains the default value of 0.</p>				

The state transition diagram (see Figure 35) shows the behaviour of the predefined master/slave explicit messaging connection.



**Figure 35 - Predefined master/slave explicit messaging state transition diagram**

Table 39 provides a detailed state event matrix for the predefined master/slave explicit messaging connection object. This state event matrix inherits from and/or overrides actions presented in the I/O connection object state event matrix in 5.3.

**Table 39 - Predefined master/slave I/O connection state event matrix**

Explicit messaging connection object state		
Event	Non-existent	Established
DeviceNet object receives an allocate master/slave connection set request that passes all error checks specified in 5.3.3.5.2.2. This request specifies the predefined master/slave explicit messaging connection.	Create the predefined master/slave explicit messaging connection object for each requested I/O connection and set attributes to default values (see 5.3.3.5.2.1). Transition to established.	This is an error. See 5.3.3.5.2.2.
UCMM receives a close request or the connection class receives a delete request and the request specifies the predefined master/slave explicit messaging connection object.	As described in Table 21.	As described in Table 21.
DeviceNet object receives a release master/slave connection set request that passes all error checks specified in 5.3.3.5.2.2. This request specifies the predefined master/slave explicit messaging connection.	This is an error. See 5.3.3.5.2.2.	Release all associated resources. Transition to non-existent. <sup>1)</sup>
Set_Attribute_Single	As described in Table 21.	As described in Table 21.
Get_Attribute_Single	As described in Table 21.	As described in Table 21.
Reset	As described in Table 21.	As described in Table 21.
Apply_Attributes	As described in Table 21.	As described in Table 21.
Receive_Data	As described in Table 21.	As described in Table 21.
Send_message	As described in Table 21.	As described in Table 21.
Inactivity/Watchdog Timer expires	As described in Table 21.	As described in Table 21.
<sup>1)</sup> Whenever a connection object within the predefined master/slave connection set transitions out of the established state, the entire predefined master/slave connection set may need to be automatically released. When the predefined master/slave connection set is released, all associated connection objects return to the non-existent state. See 5.3.3.5.2.3 for more details concerning the automatic release of the predefined master/slave connection set.		

### 5.5.3.4 Connection instance attribute access rules

Refer to 5.3.2.8 for a general description of the access rules associated with connection objects. Table 40 shows access rules specific to the predefined master/slave I/O connection objects and inherits from and/or overrides rules presented in 5.3.2.8.

**Table 40 - Predefined master/slave I/O connection object attribute access**

Attribute	I/O connection state			
	Non-existent	Configuring	Established	Timed out
State	As described in Table 22.	As described in Table 22.	As described in Table 22.	As described in Table 22.
Instance_type	As described in Table 22.	As described in Table 22.	As described in Table 22.	As described in Table 22.
TransportClass_trigger	As described in Table 22.	Get/set <sup>1)</sup>	As described in Table 22.	As described in Table 22.
Produced_connection_id	As described in Table 22.	Get only	Get only	Get only
Consumed_connection_id	As described in Table 22.	Get only	Get only	Get only
initial_comm_characteristics	As described in Table 22.	Get only	Get only	Get only
Produced_connection_size	As described in Table 22.	Get/set <sup>2)</sup>	As described in Table 22.	As described in Table 22.
Consumed_connection_size	As described in Table 22.	Get/set	As described in Table 22.	As described in Table 22.
Expected_packet_rate	As described in Table 22.	As described in Table 22.	As described in Table 22.	As described in Table 22.
Watchdog_timeout_action	As described in Table 22.	As described in Table 22.	As described in Table 22.	As described in Table 22.
Produced_connection_path_length	As described in Table 22.	As described in Table 22.	As described in Table 22.	As described in Table 22.
Produced_connection_path	As described in Table 22.	As described in Table 22.	As described in Table 22.	As described in Table 22.
Consumed_connection_path_length	As described in Table 22.	As described in Table 22.	As described in Table 22.	As described in Table 22.
Consumed_connection_path	As described in Table 22.	As described in Table 22.	As described in Table 22.	As described in Table 22.

1) The transportClass\_trigger attribute of a Predefined Master/Slave I/O Connection Object within a Slave device shall only be modifiable to one the following values: 82<sub>hex</sub> - server/transport class 2, 83<sub>hex</sub> - server/transport class 3

2) The produced\_connection\_size attribute of the bit-strobe I/O connection shall not be settable to a value greater than 8.

### 5.5.4 Master connection object characteristics

This volume does not present characteristics with respect to connection objects within a master. The master shall exhibit the external behaviour necessary to interface with its slaves.

### 5.5.5 Bit-strobe command/response messages

#### 5.5.5.1 General

Bit-strobe command and response messages move small packets of I/O data between a master and its bit-strobed slaves.

#### 5.5.5.2 Bit-strobe command message

The bit-strobe command sends one bit of output data to each slave.

The bit-strobe command message contains a bit string of 64 bits (8 bytes) of output data, one output bit for each possible MAC ID on the network.

A slave device may be designed to do one or all of the following:

- Ignore the bit-strobe command;
- Consume the bit-strobe command and its output data;
- Consume the bit-strobe command as a trigger and ignore the output data.



Slaves shall default to ignoring the bit-strobe command until the bit-strobe connection is allocated.

A bit-strobe command message transmitted with no data in the CAN data field is interpreted as a receive\_idle event by an application object. A bit-strobe command message that contains data is interpreted as a run event by an application object. The behaviour of an application object upon detection of either the receive\_idle or run event is application object specific. See the application object descriptions for more information concerning these events.

### **5.5.5.3 Bit-strobe response message**

The bit-strobe response returns up to eight bytes of input data to the master from each slave.

A bit-strobe response message that contains no data and is configured to contain data indicates a master device no valid bit-strobe data event. The behaviour of the master upon detection of this event is implementation specific.

### **5.5.5.4 Bit-strobe message characteristics**

The master implements the communication resources associated with the bit-strobe command as a client transport Class 0 connection to transmit the command and a set of server transport Class 0 connection objects to receive the responses.

The slave implements a single server transport Class 2 or 3 connection to receive the bit-strobe command and send the associated response.

### **5.5.6 Poll command/response messages**

#### **5.5.6.1 General**

The poll command and response messages transfer I/O data between a master and its polled slaves.

#### **5.5.6.2 Poll command message**

The poll command sends up to 65,535 bytes of output data (non-fragmented or fragmented) to the destination slave device.

A slave device is permitted to do one or all of the following:

- Ignore the poll command if no poll connection is allocated;
- Consume the poll command and its output data;
- Consume the poll command as a trigger and ignore the output data.

Slaves shall default to ignoring the poll command until the poll connection is allocated.

A poll command message transmitted with no data in the CAN data field is interpreted as a receive\_idle event by an application object. A poll command message that contains data is interpreted as a run event by an application object. The behaviour of an application object upon detection of the receive\_idle or run event is application object specific.

#### **5.5.6.3 Poll response message**

The poll response returns up to 65,535 bytes (non-fragmented or fragmented) of input data to the master from the slave.

A poll response message that contains no data and is configured to contain data indicates a master device no valid poll data event. The master behaviour upon detection of this event is implementation specific.

### **5.5.7 Change of state/cyclic connections**

#### **5.5.7.1 General**

The predefined master/slave connection set supports change of state or cyclic data transfer between the master and the slave. This data transfer may be either acknowledged or unacknowledged.

The change of state/cyclic connection sets use connection instance 2 for master to slave data transfer and slave to master acknowledgement. Connection instance 4 is used for slave to master data transfer and master to slave acknowledgement. If a device does not support the poll, and has no support for output data, connection instance 2 need not be created.

To keep the system behaviour consistent, the slave's change of state/cyclic connection (instance 4) shall have the consumed path set the acknowledge handler object, and the instance of the acknowledge handler object shall be 1.

Because the polled and change of state/cyclic connection sets share connection instance 2, the slave shall follow certain procedures based on the allocation request to ensure the correct behaviour. As described above, when only the change of state/cyclic allocation bit is on, connection instances 2 and 4 are created. Connection instance 4 produces from the default input path and consumes acknowledgements to instance 1 of the acknowledge handler object. Connection instance 2 consumes to the default output path and produces a zero length acknowledgement. When the allocation request contains only the polled allocation bit, connection instance 2 consumes to the default output path and produces from the default input path.

If both the change of state/cyclic *and* the polled allocation bits are set, connection instance 2 shall behave as though only the polled allocation bit was set, and connection instance 4 shall continue to behave as described above. This is also the required behaviour if the master allocates the poll connection set in one message, and then the change of state/cyclic connection set in a following message.

To achieve unacknowledged data production, the acknowledge suppression bit is set with either the change of state or cyclic bit in the allocation choice byte. Connection instance 2 is configured as a transport class 0 connection for master to slave data production. Connection instance 4 is configured as a transport class 0 connection for slave to master data production. When the poll bit is set in the same allocation choice, which causes connection instance 2 to be configured as the poll, while connection instance 4 continues to be configured as an unacknowledged change of state or cyclic connection.

#### **5.5.7.2 Change of state/cyclic messages trigger acknowledgement**

The change of state/cyclic messages move up to 65,535 bytes of data between a master and slave using change of state or cyclic production triggers. Data production may be either acknowledged or unacknowledged.

The following subclauses describe:

- Change of state/cyclic message (acknowledged);
- Change of state/cyclic message (unacknowledged);
- Change of state/cyclic message characteristics;
- Change of state/cyclic example application.

##### **5.5.7.2.1 Master's change of state/cyclic message**

The master's change of state/cyclic sends up to 65,535 bytes of data (non-fragmented or fragmented) to the destination slave device. Data production is triggered by either a change of state or transmission trigger time-out.

A change of state/cyclic message transmitted with no data in the CAN data field is interpreted as a receive\_idle event by an application object. A change of state/cyclic message that contains data is interpreted as a run event by an application object. The behaviour of an application object upon detection of the run event is application object specific.

##### **5.5.7.2.2 Slave's change of state/cyclic acknowledge message**

The slave's change of state/acknowledge message returns up to 65,535 bytes (non-fragmented or fragmented) of data to the master from the slave. By default, the acknowledge message is zero length.

##### **5.5.7.2.3 Slave's change of state/cyclic message**

The slave's change of state/cyclic message sends up to 65,535 bytes of data (non-fragmented or fragmented) to the master from the slave. Data production is triggered by either change of state or transmitted trigger time-out.

A change of state/cyclic message that contains no data and is configured to contain data indicates a master device no valid change of state/cyclic data event. The master behaviour upon detection of this event is implementation specific.

#### **5.5.7.2.4 Master's change of state/cyclic acknowledge message**

The master's change of state/cyclic acknowledge message returns up to 65,535 bytes (non-fragmented or fragmented) of data to the slave from the master. By default, the acknowledge message is zero length.

#### **5.5.8 Group 2 only devices**

To establish communications with a Group 2 only device, a client shall allocate the predefined master/slave connection set. The request to allocate a Group 2 only device is transmitted as a Group 2 only unconnected explicit request. Instead of using the UCMM to establish explicit messaging connections, Group 2 only devices receive and process Group 2 only unconnected explicit request messages.

The only services that are valid when transmitted as Group 2 only unconnected explicit request messages are:

- Allocate\_master/slave\_connection\_set message;
- Release\_master/slave\_connection\_set message.

These services are described in 5.3.2.6.

Responses to Group 2 only unconnected explicit requests are returned by transmitting a Group 2 message whose MAC ID component is set to the responding device's MAC ID (source MAC ID) and whose message ID is set to 3.

Group 2 only devices shall use the slave's explicit response message identifier field only for transmitting Group 2 only unconnected response messages and predefined master/slave explicit messaging connection response messages.

If a Group 2 only server receives a Group 2 only unconnected request message that is not an allocate\_master/slave\_connection\_set or release\_master/slave\_connection\_set request, then an error response shall be returned whose general error code is set to 02, with the additional error code set to a DeviceNet object specific value of 03 (see 5.3.3.6).

## 5.6 Physical layer

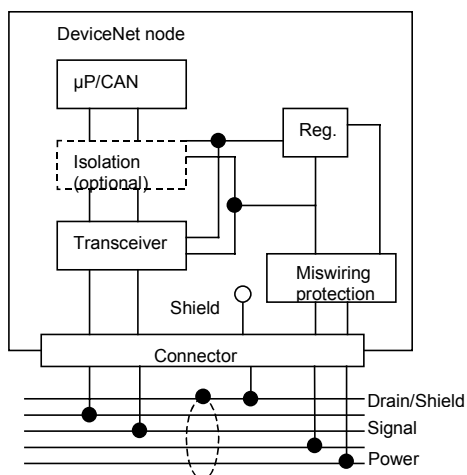
### 5.6.1 General

Table 41 shows the general physical layer specifications which shall be met in DeviceNet systems (see 9.2.6 for test specifications).

**Table 41 - General physical layer characteristics**

Characteristic	Specification
Bit rates	125 kBit/s, 250 kBit/s, 500 kBit/s
Maximum total length of trunk line and maximum cable length between any two devices	500 m at 125 kBit/s 250 m at 250 kBit/s 100 m at 500 kBit/s
Number of nodes supported by the transceiver	64
Signalling	In accordance with ISO 11898
Modulation	Baseband
Encoding	NRZ with bit stuffing
Coupling to transmission medium	DC coupled differential Tx/Rx
Isolation (between transceiver circuitry and CAN chip)	500 V dc. Applies to isolated devices only.
Differential input impedance typical (recessive state)	Shunt C = 5 pF Shunt R = 25 k $\Omega$ (power on)
Differential input impedance min. (recessive state)	Shunt C = 24 pF Shunt R = 20 k $\Omega$ (power on)
Voltage on the power conductors	11 V dc to 25 V dc
Maximum signal voltage range	-25 V dc to +18 V dc (CAN_H, CAN_L) <sup>1)</sup>
1) Voltages at CAN_H and CAN_L shall be referenced to the transceiver ground. The potential at the transceiver ground may be higher than the V- terminal by an amount equal to the voltage drop across a Schottky diode or equivalent. This voltage shall be 0,6 V maximum with respect to V-.	

The physical layer, comprising the transceiver, connector, mis-wiring protection circuitry, voltage regulator, transmission medium, and optional isolation shall be as shown in Figure 36. Testing of the robustness of the physical layer implementation within a device is specified in 9.2.9.



**Figure 36 - Physical layer block diagram**

### 5.6.2 Transceiver

A transceiver shall provide transmission and reception of CAN signals to and from the network. The transceiver shall receive differential signals from the network and forwards them to the CAN controller, and shall transmit differential signals from the CAN controller to the network.

To be compatible with the power system design, the transceiver CAN\_H and CAN\_L terminals shall support a minimum of  $\pm 5$  V common mode operation, i.e. it shall tolerate ground differences of  $\pm 5$  V.

The transceiver shall be continuously powered by the network.

The required characteristics for the transmitter and the receiver are given in Table 42 and Table 43 (see 9.2.7 for test specifications).

**Table 42 - Transmitter characteristics**

Transmitter characteristic	Specification
Differential output level (nominal)	2,0 V peak to peak
Differential output level (minimum) (measured at connector, 50 $\Omega$ load)	1,5 V peak to peak
Minimum dominant bus voltage at CAN_H	2,75 V <sup>1)</sup>
Maximum dominant bus voltage at CAN_H	4,5 V <sup>1)</sup>
Minimum dominant bus voltage at CAN_L	0,5 V <sup>1)</sup>
Maximum dominant bus voltage at CAN_L	2,0 V <sup>1)</sup>
Minimum recessive bus voltage at CAN_H and CAN_L	2,0 V <sup>1)</sup>
Maximum recessive bus voltage at CAN_H and CAN_L	3,0 V <sup>1)</sup>
Max. transmitter delay (incl. Isolation)	120 ns
Output short circuit protection	internally limited
1) Voltages at CAN_H and CAN_L shall be referenced to the transceiver ground. The potential at the transceiver ground may be higher than the V- terminal by an amount equal to the voltage drop across a Schottky diode or equivalent. This voltage shall be 0,6 V maximum with respect to V-.	

**Table 43 - Receiver characteristics**

Receiver characteristic	Specification
Differential input voltage dominant	0,95 V min
Differential Input voltage recessive	0,45 V max.
Hysteresis	150 mV typical
Max. receiver delay (include. Isolation)	130 ns
Operating voltage range (CAN_H and CAN_L)	-5 to +10 V <sup>1)</sup>
1) Voltages at CAN_H and CAN_L shall be referenced to the transceiver ground. The potential at the transceiver ground may be higher than the V- terminal by an amount equal to the voltage drop across a Schottky diode or equivalent. This voltage shall be 0,6 V maximum with respect to V-.	

A device shall have an acknowledge delay of less than or equal to 302 ns. The acknowledge delay comprises the propagation delays of the transmitter and receiver and CAN controller. Any combination of these delays is permitted provided that the total time is less than or equal to 302 ns and the maximum transmitter and receiver delays given in Table 42 and Table 43 respectively are not exceeded.

NOTE The maximum transmitter delay time in Table 42 is 120 ns and the maximum receiver delay time in Table 44 is 130 ns. A maximum CAN controller delay time of 53 ns ensures that the acknowledge delay does not exceed 302 ns.

See 9.2.8 for test specifications.

### 5.6.3 Grounding

To prevent ground loops, the network shall be grounded in only one location, preferably at the power supply. The physical layer circuitry in all devices shall be referenced to V-. A device shall not cause current to flow between V- and ground.

### 5.6.4 Isolation

#### 5.6.4.1 General

Devices shall be either non-isolated devices or isolated devices.

NOTE Isolated devices and non-isolated devices may co-exist and communicate on the network.

#### 5.6.4.2 Non-isolated devices

Within a device that contains a non-isolated physical layer, components ground referenced to V- may connect to other external devices. These external devices shall be ground isolated, and this requirement shall be stated in the manufacturer's documentation.

The shield connection on the DeviceNet connector should be connected through a parallel RC network ( $R = 1M\Omega$ ,  $C = 0,01 \mu F$  (500 V)) to the device enclosure.

NOTE Best electromagnetic conformance can be achieved with the conductor kept very short along this path and the enclosure being a closed structure of conductive material. If the device does not have such an enclosure, the shield pin of the connector may be left unconnected.

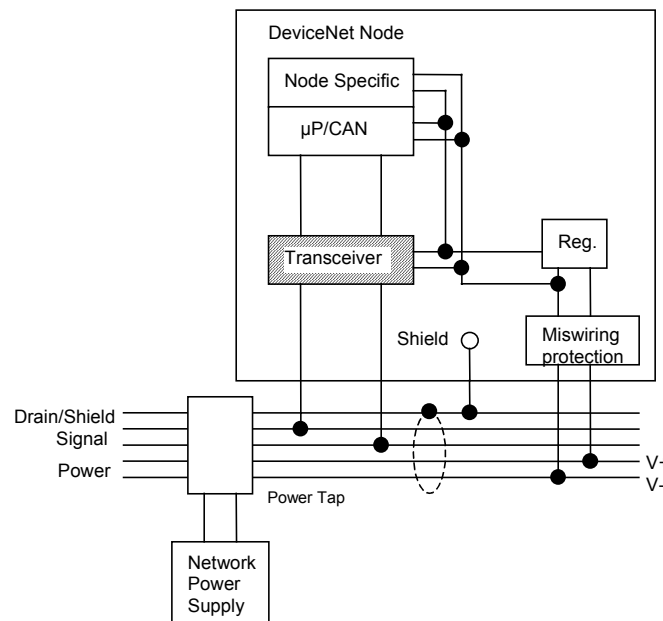


Figure 37 - Device containing a non-isolated physical layer

#### 5.6.4.3 Isolated device

These devices shall obtain power from the network for the transceiver and isolation circuitry (see Figure 38). Network power may be used for other circuitry provided that this network powered circuitry is ground referenced to V- or is otherwise ground isolated.

For an isolated device the CAN controller may still be active during a de-energized power line condition. Isolated devices shall be capable of sensing when their transceivers have become de-energized.

Within an isolated device, components ground referenced to V- may connect to other external devices through serial ports, parallel ports, or I/O connections. These external devices shall be ground isolated and this requirement shall be stated in the manufacturer's documentation.

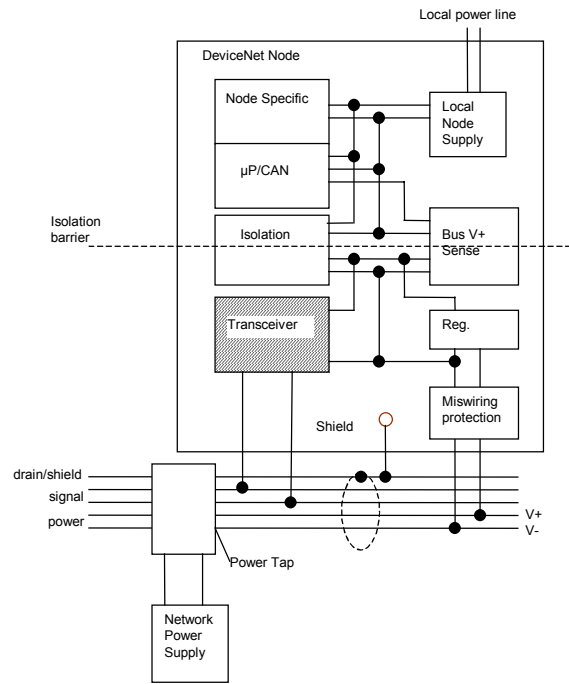


Figure 38 - Device containing an isolated physical layer

The shield connection of the network should be connected via a parallel RC circuit to the device enclosure.

### 5.6.5 Transmission medium

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.

DeviceNet provides two different types of cable - thick cable and thin cable. Thick cable provides higher current capability and allows communication over longer distances. Both thick and thin cable may be used for trunk lines and/or drop lines in any combination.

### 5.6.6 Topology

#### 5.6.6.1 General

The DeviceNet medium shall have a linear network topology. If used, drop lines may allow one or more nodes to be attached. Branching structures are only allowed on the drop line.

Terminating resistors shall be connected at each end of the trunk line. The cable distance between any two points in the cable system shall not exceed the maximum cable distance allowed for the bit rate.

#### 5.6.6.2 Trunk lines

Either thick or thin cable may be used to construct trunk lines. The maximum cable distance with a mix of both thick and thin cable shall be calculated using the formulas below:

$$\begin{aligned} l_{\text{thick}} + 5 \times l_{\text{thin}} &= 500 \text{ m} && \text{at 125 kBit/s} \\ l_{\text{thick}} + 2,5 \times l_{\text{thin}} &= 250 \text{ m} && \text{at 250 kBit/s} \\ l_{\text{thick}} + l_{\text{thin}} &= 100 \text{ m} && \text{at 500 kBit/s} \end{aligned}$$

where  $l_{\text{thick}}$  is the length of thick cable and  $l_{\text{thin}}$  is the length of thin cable.

#### 5.6.6.3 Drop lines

The drop line length is the cable distance measured from the tap on the trunk line to the transceiver of each device and shall not exceed 6 m. The total length of drop line cable allowable on the network depends upon the bit rate and shall not exceed the values specified in Table 44.

**Table 44 - Total drop line cable length**

Bit rate	Cable length
125 kBit/s	156 m
250 kBit/s	78 m
500 kBit/s	39 m

## 5.6.7 Network power

### 5.6.7.1 Power configuration

The network power shall be supplied by a rated 24 V dc source and can support up to 8 A on any section of thick cable trunk line or up to 3 A on any section of thin cable trunk line. Multiple power supplies can be used.

### 5.6.7.2 Load limits

The maximum network current shall be determined using the information shown in Table 45 (see 9.2.2 for test specifications).

**Table 45 - Load limits**

Characteristic	Specification
Maximum voltage drop on V- and V+	5 V each
Maximum thick cable trunk line current	8 A
Maximum thin cable trunk line current	3 A
Maximum drop line current range	0,75 A to 3,0 A <sup>1)</sup>
Voltage range at each node	11 V to 25 V
Operating current of each device <sup>2)</sup>	specified by manufacturer
1) Maximum drop line current depends upon the length of the drop (see below). 2) The operating current represents the average current drawn from the network. Peak operating current shall be specified if it exceeds the average current by more than 10%.	

The maximum drop line current is also constrained by the following equation:

$$i = 4,5/l$$

where:

i is the maximum allowable drop line current (A)

l is the drop length (m)

## 6 Product information

### 6.1 Instructions for installation, operation and maintenance

In accordance with 6.1 of EN 50325-1.

### 6.2 Marking

In accordance with 6.2 of EN 50325-1.

## 7 Normal service, transport and mounting conditions

### 7.1 Normal service conditions

#### 7.1.1 General

Components of a DeviceNet network shall be capable of operating under the following conditions:

NOTE If the conditions for operation differ from those given in this Part, the user shall state the deviation from the standard conditions and consult the manufacturer on the suitability for use under such conditions.



## **7.1.2 Ambient air temperature**

### **7.1.2.1 Thick cable**

The cable shall operate normally within an ambient temperature range of -20 °C to +60 °C when carrying a current of 8 A. This current rating shall be derated linearly to zero at 80 °C.

### **7.1.2.2 Thin cable**

The cable shall operate normally within an ambient temperature range of -20 °C to +70 °C when carrying a current of 1,5 A. This current rating shall be derated linearly to zero at 80 °C.

### **7.1.2.3 Device taps**

Device taps shall operate normally within an ambient temperature range of -40 °C to +70 °C when carrying full current. This current rating shall be derated linearly to zero at 80 °C. Maximum continuous current on power conductors is 3 A for micro connectors and 8 A for other connectors unless otherwise specified.

### **7.1.2.4 Power taps**

Power taps shall operate normally within an ambient temperature range of -40 °C to +70 °C when carrying full current. This current rating shall be derated linearly to zero at 80 °C.

### **7.1.2.5 Other network components**

All other components of a DeviceNet network shall operate between the ambient temperatures of -25 °C to +70 °C if not otherwise defined, e.g. in conjunction with a specific actuator or sensor type. The operating characteristics shall be maintained over the permissible range of ambient temperature.

## **7.1.3 Altitude**

DeviceNet components shall be capable of operating at altitude in accordance with 7.1.3 of EN 50325-1.

## **7.1.4 Humidity**

DeviceNet components shall be capable of operating in humid conditions in accordance with 7.1.4 of EN 50325-1.

## **7.1.5 Pollution degree**

DeviceNet components shall be capable of operating in polluted conditions in accordance with 7.1.5 of EN 50325-1.

## **7.1.6 Sealed connectors**

DeviceNet components shall be provided with sealed connectors in accordance with 7.1.6 of EN 50325-1.

## **7.2 Conditions during transport and storage**

Conditions during transport and storage shall be in accordance with 7.2 of EN 50325-1.

## **7.3 Mounting**

DeviceNet components shall be mounted in accordance with 7.3 of EN 50325-1.

## 8 Constructional and performance requirements

### 8.1 Indicators and configuration switches

#### 8.1.1 Status indicators

##### 8.1.1.1 General

Devices may have the following status indicators:

- One module status indicator and one network status indicator, or one combined module/network status indicator;
- I/O status indicator.

Any other indicators shall be clearly distinguishable from those defined above.

##### 8.1.1.2 Module status indicator

This shall be a bicolour (green/red) indicator and shall indicate whether or not the DeviceNet circuitry is powered and operating correctly. The indicator states shall be as specified in Table 46.

**Table 46 - Module status indicator**

Indicator status	Meaning
Off	Not powered
Green	Operating correctly
Flashing green	Missing, incomplete or incorrect configuration
Flashing red	Recoverable fault.
Red	Unrecoverable fault
Flashing red-green	Undergoing self test.
NOTE: Indicator flash rates are given in 8.1.1.8	

##### 8.1.1.3 Network status indicator

This shall be a bicolour (green/red) indicator and shall indicate the status of the communication link as specified in Table 47.

**Table 47 - Network status indicator**

Indicator status	Meaning
Off	The duplicate MAC ID detection test has not been completed or the DeviceNet circuitry is not powered.
Flashing green	Device is on-line but has no connections established
Green	Device is on-line and has connections established
Flashing red	One or more I/O connections have timed out.
Red	The device has detected an error and cannot communicate on the network.
NOTE: Indicator flash rates are given in 8.1.1.8	

##### 8.1.1.4 Module and network status indicators at power-up

The following indicator test sequence shall be performed at power-up:

- network status indicator OFF;
- module status indicator ON green for 0,25 seconds  $\pm$  0,1 seconds;
- module status indicator ON red for 0,25 seconds  $\pm$  0,1 seconds;
- module status indicator ON green;
- network status indicator ON green for 0,25 seconds  $\pm$  0,1 seconds;
- network status indicator ON red for 0,25 seconds  $\pm$  0,1 seconds;
- network status indicator OFF;
- both indicators assume states according to Table 46 and Table 47.

### 8.1.1.5 Combined module/network status indicator

This shall be a bicolour (green/red) indicator and shall provide limited device and communication status information.

The combined module/network status indicator states shall be as specified in Table 48.

**Table 48 - Combined module/network status indicator**

Indicator status	Meaning
Off	Device is not on-line: - The device may not have completed the duplicate MAC ID test. - The device may not be powered.
Green	The device is operating in a normal condition and the device is on-line with connections in the established state.
Flashing green	The device is operating in a normal condition and either the device is on-line with no connections in the established state or configuration is missing, incomplete or incorrect.
Flashing red	Recoverable fault and/or one or more I/O connections are in the timed out state.
Red	The device has an unrecoverable fault that has rendered it incapable of communicating on the network
NOTE: Indicator flash rates are given in 8.1.1.8	

### 8.1.1.6 Combined module/network status indicator at power-up

The following indicator test sequence shall be performed at power-up:

- combined module/network status indicator ON green for 0,25 seconds  $\pm$  0,1 seconds;
- combined module/network indicator ON red for 0,25 seconds  $\pm$  0,1 seconds;
- combined module/network indicator OFF;
- indicator assumes state according to Table 48.

### 8.1.1.7 I/O status indicator

This shall be a bicolour (green/red) indicator and shall provide information concerning the states of inputs and/or outputs.

The I/O status indicator states shall be as specified in Table 49.

**Table 49 - I/O status indicator**

Indicator status	Meaning
Off	All outputs are inactive. All inputs are inactive
Flashing green	One or more outputs are idle and no outputs are faulted.
Green	One or more outputs are active and under control, no outputs are faulted. One or more inputs are active and producing data, no inputs are faulted.
Flashing red	One or more outputs are faulted. One or more inputs are faulted.
Red	One or more outputs are forced OFF (may be an unrecoverable fault). One or more inputs has an unrecoverable fault.
NOTE: Indicator flash rates are given in 8.1.1.8	

### 8.1.1.8 Indicator flash rate

The flash rate of an indicator shall be 1 flash per second  $\pm$  0,5 seconds. An indicator shall be ON for 0,5 second  $\pm$  0,25 seconds and OFF for 0,5 second  $\pm$  0,25 seconds.

### 8.1.2 DeviceNet bit rate switches

If switches are used to set the DeviceNet bit rate they shall be encoded as shown in Table 50.

**Table 50 - DeviceNet bit rate switch encoding**

Bit rate	Switch setting
125 kBit/s	0
250 kBit/s	1
500 kBit/s	2

### 8.1.3 Indicator and switch marking

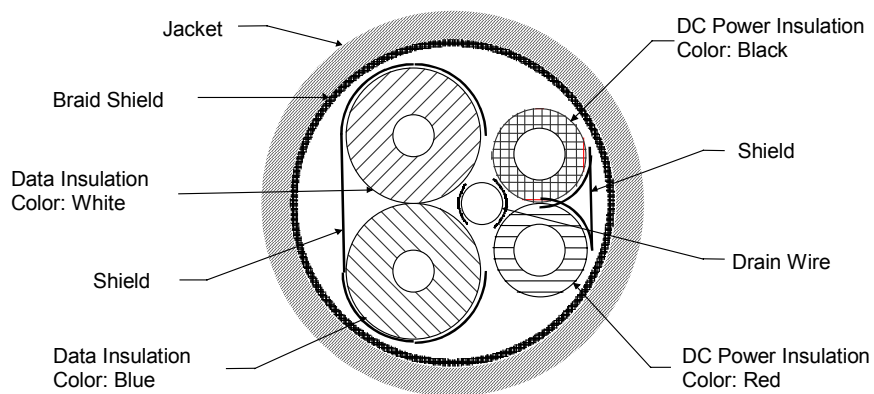
Indicators and switches shall be marked with either the full name or the abbreviated name as specified in Table 51.

**Table 51 - Indicator and switch marking**

Description	Full name	Abbreviated name
Module status indicator	Module status	MS
Network status indicator	Network status	NS
Combined module/network status indicator	Module/network status or Mod/net status	MNS
I/O status indicator	I/O status or I/O	IO
MAC ID switches	Node address	NA
Bit rate switches	Data rate	DR

### 8.2 DeviceNet cable

The physical configuration for both thick and thin cables shall be as shown in Figure 39.



**Figure 39 - Physical configuration of thick and thin cable**

Cable shall be in accordance with specifications shown in Table 52, Table 53 and Table 54:

**Table 52 - Cable: data pair specification**

Characteristic	Thick cable	Thin cable
Conductor pair size	0,823 mm <sup>2</sup> copper (individually tinned)	0,205 mm <sup>2</sup> copper (individually tinned)
Colours	blue, white	blue, white
Nominal pair twist	10 / m	16 / m
Shield over pair	98 % coverage	98 % coverage
Impedance	120 Ω ±10 % at 1 MHz	120 Ω ±10 % at 1 MHz
Maximum propagation delay	4,46 ns/m	4,46 ns/m
Capacitance between conductors	39 pF/m at 1 MHz (nominal)	39 pF/m at 1 MHz (nominal)
Nominal capacitance between one conductor and other conductor connected to shield	79 pF/m at 1 MHz	79 pF/m at 1 MHz
Maximum capacitive imbalance	2,0 pF/m at 1 MHz	2,0 pF/m at 1 MHz
Resistance at 20 ° C	Nominal 18 Ω / 1 000 m Maximum 23 Ω / 1 000 m	Nominal 69 Ω / 1 000 m Maximum 92 Ω / 1 000 m
Maximum attenuation	0,43 dB / 100 m at 125 kHz 0,82 dB / 100 m at 500 kHz 1,2 dB / 100 m at 1 MHz	0,95 dB / 100 m at 125 kHz 1,6 dB / 100 m at 500 kHz 2,3 dB / 100 m at 1 MHz

**Table 53 - Cable: DC power pair specification**

Physical characteristics	Thick cable specification	Thin cable specification
Conductor pair size	1,65 mm <sup>2</sup> copper (individually tinned)	0,326 mm <sup>2</sup> (individually tinned)
Colours	red, black	red, black
Nominal pair twist	10 / m	16 / m
Shield over pair	98 % coverage	98 % coverage
Resistance at 20 ° C	Nominal 11 Ω / 1 000 m Maximum 12 Ω / 1 000 m	Nominal 45,9 Ω / 1 000 m Maximum 57,4 Ω / 1 000 m

**Table 54 - Cable: general specifications**

Characteristics	Thick cable specification	Thin cable specification
Two shielded pairs	Common axis with drain wire in centre	Common axis with drain wire in centre
Overall braid shield	65 % coverage, 0,126 mm <sup>2</sup> Cu braid, (individually tinned)	65 % coverage, 0,126 mm <sup>2</sup> Cu braid, (individually tinned)
Drain wire	0,823 mm <sup>2</sup> copper (individually tinned)	0,326 mm <sup>2</sup> copper (individually tinned)
Resistance at 20 ° C (shields and drain wire)	5,74 Ω / 1 000 m	10,5 Ω / 1 000 m

### 8.3 Terminating resistors

A 121 Ω, 1%, 0,25 W metal film resistor shall be connected at each end of the trunk line.

Terminating resistors shall not be included in devices.

### 8.4 Connectors

#### 8.4.1 General specifications

All connectors shall have five terminals: a signal pair, a power pair, and a shield. Connectors shall be either open or sealed type.

Open connectors shall comply with Table 55.

**Table 55 - Open connector general specifications**

Connector style	Specification
Pluggable	5 pin open style male (pin) contacts with mechanical locking to prevent reverse insertion.
Hard wired	Includes direct soldering, crimping, screw terminal blocks, barrier strips, or other similar attachment methods. Hard wired connection shall enable a device to be removed without severing the trunk. The attachment of devices using these methods shall not be attempted while network power is applied.

Sealed connectors shall be protected to IP 67 (see EN 60529) and shall comply with Table 56:

**Table 56 - Sealed connector general specifications**

Connector style	Specification
Male mini connector	5 pin "mini" style male (pin) contacts with 7/8-16 UN-2A male (outside) connection thread. Shall meet ANSI/B93.55M-1981 design and mating requirements.
Female mini connector	5 pin "mini" style female (socket) contacts with 7/8-16 UN-2B female (inside) connection thread. Shall meet ANSI/B93.55M-1981 design and mating requirements.
Male micro connector	See Annex D of EN 60947-5-2
Female micro connector	See Annex D of EN 60947-5-2

#### 8.4.2 Contact and electrical specifications

Connectors shall comply with Table 57 and Table 58.

**Table 57 - Contact specifications**

Physical characteristics	Specification
Wiping contact plating requirements	0,76 µm gold minimum over 1,3 µm nickel minimum or 0,13µm gold minimum over 0,51 µm palladium-nickel minimum over 1,3 µm nickel. All gold shall be 24 carat.
Wiping contact life	1 000 insertion / extractions
NOTE: This requirement applies to contacts that may be subjected to multiple engagements, and not to hard wired semi-permanent connections such as screw terminals.	

**Table 58 - Electrical specifications**

Electrical characteristics	Specification
Operating voltage	25 V dc minimum
Contact current rating	Connector contacts for use with thick cable shall be rated at 8 A minimum. Connector contacts for use with thin cable shall be rated at 3 A minimum. Power supply contact current ratings shall be equal to or greater than the power supply current rating.
Contact resistance	Initial: less than 1 mΩ. Maximum: 5 mΩ throughout life.

#### 8.4.3 Contact arrangement of device mounted connectors

All device mounted connectors shall have male contacts and shall be arranged as shown in Table 59:

**Table 59 - Connector contact arrangement**

Function	Wire colour	Pluggable and hard wired connector contact numbers	Sealed mini and micro style connector contact numbers
V+	Red	5	2
V-	Black	1	3
CAN_H	White	4	4
CAN_L	Blue	2	5
Shield	Bare	3	1

### 8.5 Device taps and power taps

#### 8.5.1 Device taps

Device taps provide points of attachment onto the trunk line. Devices may be connected to the network either directly to the device tap or with a drop line. Device taps shall also allow removal of a device without disrupting network operation.

Device taps shall have a minimum voltage rating of 25 V dc and shall comply with Table 60 and Table 61.

**Table 60 - Internal pass-through conductor specifications**

Conductor description	Specification
Drain wire conductor	100 mm maximum length of conductor. Maximum resistance 2,3 mΩ.
Power conductors V+ and V-	100 mm maximum length of conductor. Maximum resistance 1,2 mΩ. Rated current 8 A.
Signal conductors CAN_H and CAN_L	100 mm maximum length of conductor. Maximum resistance 2,3 mΩ.

**Table 61 - Internal drop conductor specifications**

Conductor description	Specification
Drain wire conductor	180 mm maximum length and 10 mΩ maximum resistance, both measured between the trunk line tap connector and any port connection.
Power conductors V+ and V-	180 mm maximum length measured between the trunk line tap connector and any port connection. Maximum resistance 10 mΩ for conductors rated up to 3 A, 4 mΩ otherwise, both measured between the trunk line tap connector and any port connection.
Signal conductors CAN_H and CAN_L	180 mm maximum length and 10 mΩ maximum resistance, both measured between the trunk line tap connector and any port connection.

### 8.5.2 Power taps

A power tap connects a power supply to the trunk line. When connected to the network, a power tap shall provide continuous connection for the signal, drain and V- wires. A power tap may also provide:

- overcurrent protection in each direction from the tap;
- power supply protection when multiple power supplies are used;
- a connection to the shield/drain wire for grounding the network.

DeviceNet power taps shall have a minimum voltage rating of 25 V dc and shall comply with Table 62 and Table 63.

**Table 62 - Internal pass-through conductor specifications**

Conductor description	Specification
Drain wire conductor	180 mm maximum length of conductor. Maximum resistance 4,1 mΩ.
V- power conductor	180 mm maximum length of conductor. Maximum resistance 2,1 mΩ. Rated current 8 A.
Signal conductors CAN_H and CAN_L	180 mm maximum length of conductor. Maximum resistance 4,1 mΩ.

**Table 63 - Internal power drop conductor specifications**

Conductor description	Specification
Drain wire conductor to ground terminal	180 mm maximum length and 2,1 mΩ maximum resistance, both measured between any tap connector and the ground terminal.
Power conductors V+ and V-	300 mm maximum length and 3,6 mΩ maximum resistance, both measured between any tap connector and the power supply port connection. Wires shall be rated to carry the operating current.

Power tap connectors (except the power supply connector) shall meet the requirements given in 8.4.

### 8.6 Network powered devices

Devices may draw power from the network. Such devices shall use a voltage regulator with an input voltage between 11 V dc and 25 V dc and incorporate transient protection and filtering.

The voltage regulator shall comply with Table 64.

**Table 64 - Voltage regulator specifications**

Specification	Parameter
Input voltage range	11 V dc to 25 V dc
Isolation (if required)	500 V
Power ON delay	Linear regulators: none. Switch mode regulators: < 100 mA            2 to 10 ms 0,1 A - 0,5 A        5 to 15 ms 0,5 A - 1 A           10 to 20 ms 1 A - 2 A             15 to 30 ms > 2A                  20 to 40 ms
Output short circuit protection	Current limit
Reverse polarity protection	Rectifier in ground path <sup>1)</sup>
1) If the rectifier is used to power the transceiver, it shall have a forward voltage of no greater than 0,6 V.	

### 8.7 Miswiring protection

Devices shall be capable of sustaining misconnection without suffering damage under the following conditions:

- V+ and V- conductors interchanged;
- connection of the drain wire to any other conductor;
- connection of CAN\_H and/or CAN\_L to a voltage not exceeding the range -25 V dc to +18 V dc.

See 9.2.4 and 9.2.5 for test specifications.

### 8.8 Power supplies

Power supplies which are connected to the network shall comply with Table 65 (see 9.2.1 for test specifications).

**Table 65 - DeviceNet power supply specifications**

Specification	Parameter
Nominal voltage at 20 °C	24 V dc $\pm$ 1 % or adjustable to 0,2 %
Line regulation	0,3 % max. over the range specified by the manufacturer
Load regulation	0,3 % max.
Temperature coefficient	0,03 % per K max.
Output ripple	250 mV peak to peak
Load capacitance capability	7000 $\mu$ F max.
Over current protection	100 % to 125 % of rated current
Output voltage rise time (with full load)	250 ms max. to 95 % of final value
Power ON overshoot	0,2 % max.
Stability	0 to 100 % load (all conditions)
Isolation	Output isolated from input voltage and chassis ground

### 8.9 Electromagnetic compatibility

#### 8.9.1 General

All immunity and emission tests are type tests and shall be carried out under representative conditions, both operational and environmental, using the recommended wiring practices and including all equipment necessary for communication and data transfer on the DeviceNet cable (see 9.2.10 for test specifications).

This requirement shall be met by the use of two devices with ongoing I/O communication and one power supply.

#### 8.9.2 Immunity

The test results are specified using the following performance criteria:

- **Criterion A** : No degradation of performance beyond specified limits is allowed when the apparatus is used as intended. See Table 66.



- **Criterion B** : No change of actual operating state or stored data is allowed. During the test, degradation of performance is allowed. No degradation of performance or loss of function beyond specified limits is allowed after the test when the apparatus is used as intended. See Table 66.

**Table 66 - Immunity performance criteria**

Function Type	Criterion A (Radiated, Conducted)	Criterion B (Fast Transient, Electrostatic Discharge, Surge)
Masters (Processors and Adapters)	Program Loss Memory faults I/O reset Data table corruption	Program Loss Memory faults I/O reset Data table corruption
Any Module	Unexpected operation Lockup Operator intervention Damage	Unexpected operation Lockup Operator intervention Damage
External Communications	Node off-line	Node off-line
Internal Communications: - Radiated, Conducted - Fast transient - ESD - Surge	> 1 error bit set / 10 transfers	> 1 error bit set / 10 transfers Lockup Lockup

### 8.9.2.1 Conducted radio-frequency disturbance immunity

In accordance with EN 61000-4-6:

- 10 V rms over the frequency range 150 kHz to 80 MHz, amplitude modulated.

Performance criterion A applies (see 8.9.2).

### 8.9.2.2 Radiated radio-frequency electromagnetic field immunity

In accordance with EN 61000-4-3:

- 10 V/m over the frequency range 80 MHz to 1 000 MHz amplitude modulated, and 900 MHz  $\pm$  5 MHz pulse modulated.

Performance criterion A applies (see 8.9.2).

### 8.9.2.3 Electrostatic discharge (ESD) immunity

In accordance with EN 61000-4-2, 10 discharges of each polarity shall be applied as follows:

- A test voltage of  $\pm 4$  kV shall be applied by the contact discharge method to metallic device enclosures;
- A test voltage of  $\pm 8$  kV shall be applied by the air gap discharge method to non-metallic device enclosures.

Performance criterion B applies (see 8.9.2).

### 8.9.2.4 Fast transient/burst immunity

In accordance with EN 61000-4-4. Test voltages as specified in EN 50082-2 shall be applied at 5 kHz.

Performance criterion B applies (see 8.9.2).

### 8.9.2.5 Surge immunity

In accordance with EN 61000-4-5. Five surges shall be applied at the test voltages specified in EN 50082-2.

Performance criterion B applies (see 8.9.2).

### 8.9.2.6 Voltage variations

Conformity of the power supply to the specified limits given in Table 65 eliminates the need for any further testing.

## 8.9.3 Emissions

### 8.9.3.1 Radiated emissions

Devices shall meet the requirements of EN 50081-2 when tested in accordance with EN 55011.

### 8.9.3.2 Conducted emissions

Devices shall meet the requirements of EN 50081-2 when tested in accordance with EN 55011.

## 9 Tests

### 9.1 General

This clause specifies type tests for electrical and logical requirements. Tests are divided into two parts:

- Electrical and EMC testing;
- Logical testing (behaviour of a device on DeviceNet).

NOTE If DeviceNet specifications are implemented as an integral part of a product, some logical tests may depend on the behaviour of the product itself. Conformance to this Part does not necessarily mean conformance to a specific standard of the whole product in which DeviceNet is integrated. A test of specific applications of DeviceNet functions is outside the scope of this Part.

The tests applicable to a specific EUT (Equipment under test) shall be performed in the sequence indicated.

### 9.2 Electrical and EMC testing

#### 9.2.1 Test of the DeviceNet power supply

##### 9.2.1.1 Test purpose

The following tests verify the power supply requirements that are specific to DeviceNet.

##### 9.2.1.2 Power supply output voltage rise time

###### 9.2.1.2.1 Test circuit

The test circuit shall be as shown in Figure 40. Select the load resistor (R) to ensure that the power supply operates at rated current.

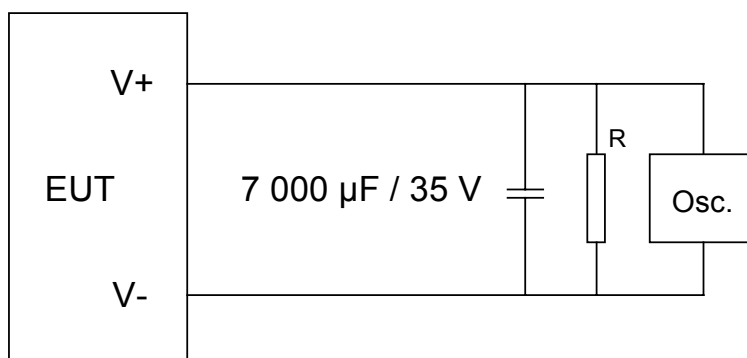


Figure 40 - Power supply rise time test circuit

### 9.2.1.2.2 Test procedure

The oscilloscope connected across the load resistor shall be set to 24 V, 500 ms full scale. The EUT shall be switched on and the rise time from the start of the voltage rise to 95 % of the power supply final voltage shall be recorded.

### 9.2.1.2.3 Criteria for compliance

The rise time shall be less than or equal to 250 ms (see Table 65).

### 9.2.1.3 Power supply ripple

#### 9.2.1.3.1 Test circuit

The test circuit shall be as shown in Figure 40 with the capacitor disconnected..

#### 9.2.1.3.2 Test procedure

The power supply shall be switched on and the ac peak to peak ripple of the power supply output voltage measured using the oscilloscope.

#### 9.2.1.3.3 Criteria for compliance

In order to meet the power supply requirements, the peak to peak ripple shall be less than or equal to 250 mV. (see Table 65)

### 9.2.2 Device peak current consumption

#### 9.2.2.1 Test circuit

The test circuit shall be as shown in Figure 41.

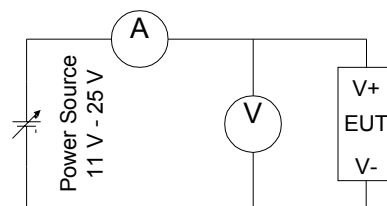


Figure 41 - Current consumption test circuit

#### 9.2.2.2 Test procedure

Determine the operating state that results in maximum current consumption from the network. The EUT shall be operated in this state. If current fluctuations occur, the peak current drawn from the network shall be recorded.

The tests shall be performed at the upper and lower limits of the voltage range specified in Table 45.

#### 9.2.2.3 Criteria for compliance

Determine the applicable current limit by inspection of the device and/or the manufacturer's documentation. The current readings obtained shall be in accordance with Table 45 and shall be less than or equal to 8 A.

### 9.2.3 Power ON behaviour

#### 9.2.3.1 Test purpose

The purpose of this test is to determine that the first transmitted signal is a valid duplicate MAC ID message under the following conditions:

- when added to an existing working network;
- when the power is cycled on the network.

### 9.2.3.2 Test circuit

The test circuit shall be as shown in Figure 42.

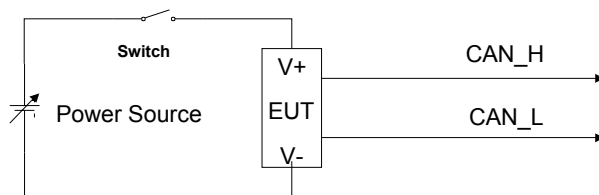


Figure 42 – Power ON test circuit

### 9.2.3.3 Test procedure

The switch shall be closed and the CAN\_H and CAN\_L signal lines shall be monitored.

### 9.2.3.4 Criteria for compliance

The first transmitted signal shall be a valid duplicate MAC ID request.

### 9.2.4 Reverse connection of V+ and V-

#### 9.2.4.1 Test purpose

The purpose of this test is to verify compliance with requirements for the miswiring protection.

#### 9.2.4.2 Test circuit

The test circuit shall be as shown in Figure 43.

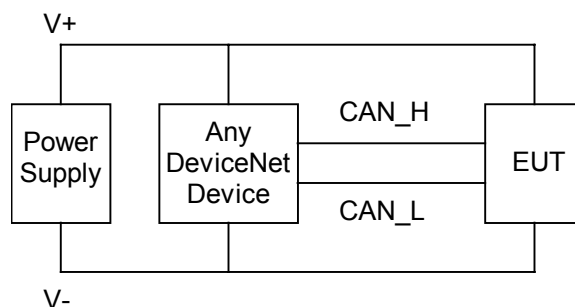


Figure 43 - Test circuit for reversal of V+ and V-, and also V- interruption

### 9.2.4.3 Test procedure

The V+ terminal of the EUT shall be connected to the V- of the power supply and network, and the V- terminal of the EUT shall be connected to the V+ of the power supply and network. CAN\_H and CAN\_L shall be monitored.

### 9.2.4.4 Criteria for compliance

It shall be verified that the CAN\_H and CAN\_L signals remain within the limits specified in Table 42 and Table 43.

### 9.2.5 Disconnection of V-

#### 9.2.5.1 Test purpose

The purpose of this test is to verify compliance with requirements for the miswiring protection.

### 9.2.5.2 Test circuit

The test circuit shall be as shown in Figure 43.

### 9.2.5.3 Test procedure

The V- connection shall be disconnected at the EUT. CAN\_H and CAN\_L shall be monitored.

### 9.2.5.4 Criteria for compliance

It shall be verified that the CAN\_H and CAN\_L signals remain within the limits specified in Table 42 and Table 43.

## 9.2.6 Differential input impedance test

### 9.2.6.1 Test purpose

The purpose of this test is to verify that the input impedance of the EUT meets the requirements of the physical layer.

### 9.2.6.2 Test circuit

The test circuit shall be as shown in Figure 44.

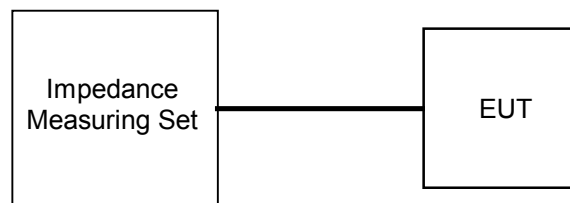


Figure 44 - Differential impedance test circuit

### 9.2.6.3 Test procedure

Power shall be applied to the EUT. The transceiver of the EUT shall be disabled in such a way that the impedance is not affected. The input impedance of the EUT between CAN\_H and CAN\_L shall then be measured at a frequency of  $100 \text{ kHz} \pm 1 \text{ kHz}$ .

### 9.2.6.4 Criteria for compliance

The input impedance of the EUT between CAN\_H and CAN\_L shall not exceed the values specified in Table 41.

## 9.2.7 Transmit levels

### 9.2.7.1 Test purpose

The purpose of this test is to determine whether the values of CAN\_H and CAN\_L with respect to V- under simulated worst case impedance conditions are within the limits specified as part of the physical layer requirements.

### 9.2.7.2 Test circuit

The test circuit shall be as shown in Figure 45.

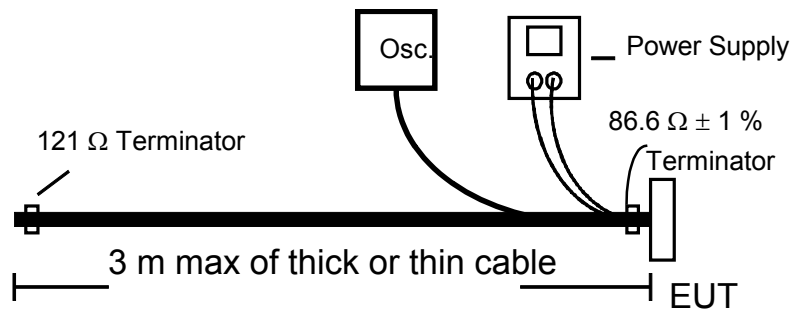


Figure 45 - Transmit level test setup

### 9.2.7.3 Test procedure

For this test, the EUT shall be transmitting. With no other device on the network, the EUT will automatically start sending duplicate MAC ID request messages after completing its power-up sequence. The actual message content from the EUT shall be observed on the oscilloscope. The following values shall be recorded (see Figure 46):

- Recessive CAN\_L voltage with respect to V-
- Recessive CAN\_H voltage with respect to V-
- Dominant CAN\_L voltage with respect to V-
- Dominant CAN\_H voltage with respect to V-
- Differential output level.

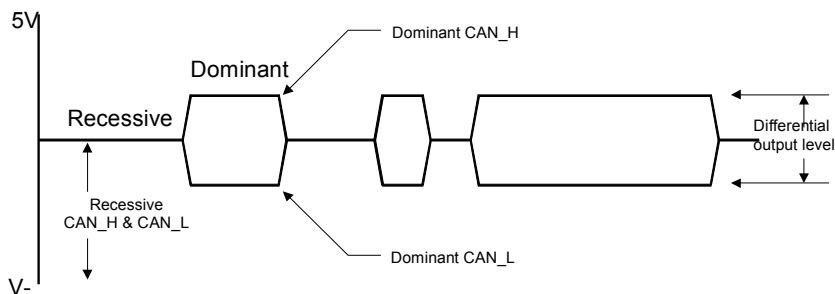


Figure 46 - Transmit Levels

### 9.2.7.4 Criteria for compliance

The values shall be in accordance with those specified in Table 42.

### 9.2.8 Acknowledge Delay

#### 9.2.8.1 Test purpose

The purpose of this test is to check that the acknowledge delay is within the limits specified as part of the physical layer requirements.

#### 9.2.8.2 Test circuit

The test circuit shall be as shown in Figure 47.

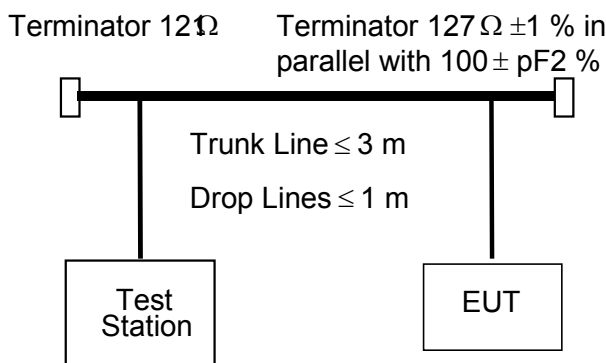


Figure 47 - Timing test setup

### 9.2.8.3 Test procedure

The EUT and the Test Station shall be set to the highest bit rate supported by the EUT. The Test Station shall repetitively send a `get_attribute_single` request to the EUT without opening a messaging connection. This ensures that the EUT sends no messages other than the acknowledge bit. An oscilloscope shall be used to measure the acknowledge response time at the EUT. This is the time from the falling edge of the last bit sent by the Test Station to the rising edge of the acknowledge bit sent by the EUT (see Figure 48).

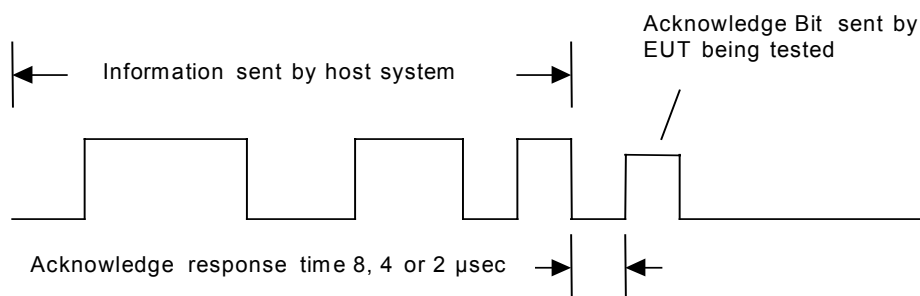


Figure 48 - Timing

### 9.2.8.4 Criteria for compliance

Depending on the bit rate used, the following values shall be subtracted from the response times measured:

125 kBit/s	8 $\mu$ s
250 kBit/s	4 $\mu$ s
500 kBit/s	2 $\mu$ s

The resultant device acknowledge delay shall be less than or equal to the value specified in 5.6.2.

## 9.2.9 Network tests

### 9.2.9.1 Test purpose

The purpose of this test is to verify the robustness of the physical layer implementation within the EUT. It shall determine whether the missing acknowledgement bits and bus-off conditions under worst case conditions are within the limits specified within the manufacturer's documentation.

### 9.2.9.2 Test circuit

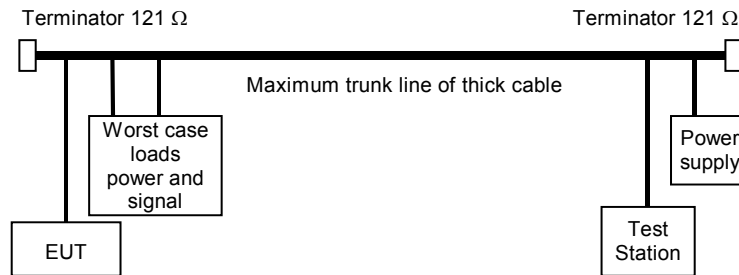
The test circuit shall be as shown in Figure 49.

- the system shall be run at the maximum bit rate supported by the EUT;
- the cable system shall be constructed of the maximum trunk line length of thick cable suitable for this bit rate;
- the maximum cumulative drop line shall be constructed of the maximum thin cable drop line length suitable for this bit rate;

- 62 devices or 62 device equivalents shall be used to load the network near the EUT.

NOTE A device equivalent is the minimum differential input impedance as defined in Table 41.

- a power supply shall be used that meets the specifications described in Table 65 (minimum current capacity 4 A);
- a worst case network power load shall be connected adjacent to the EUT. This load shall cause the V-line to develop a 5 V +0,0/-0,1 V drop from the power supply to the EUT;
- a Test Station shall be used that is capable of both producing valid network traffic to the EUT and monitoring all network traffic.



**Figure 49 - Network test configuration**

#### 9.2.9.3 Test procedure

Communications shall be established between the Test Station and the EUT. Any missing acknowledgement bits or bus-off conditions shall be recorded.

#### 9.2.9.4 Criteria for compliance

The recorded missing acknowledgement bits and bus-off conditions shall be as specified by the manufacturer of the EUT.

#### 9.2.10 Electromagnetic compatibility testing

Unless otherwise noted, the tests shall be carried out at an ambient temperature of  $+23 \pm 5$  °C.

The EUT shall be mounted in free air and shall be connected to the DeviceNet network according to the manufacturer's instructions and supplied with its rated voltage.



## 9.3 Logical testing

### 9.3.1 General

Tests shall only be performed on nodes that support the particular functionality being tested.

Logical tests require the presence of a Test Station on the network, which can:

- send DeviceNet messages;
- receive and evaluate DeviceNet messages;
- record and time stamp traffic on the network.

### 9.3.2 Duplicate MAC ID check test

#### 9.3.2.1 Test purpose

The purpose of this test is to verify that the EUT meets the requirements regarding the handling of Duplicate MAC ID mechanisms.

#### 9.3.2.2 Test circuit

The test circuit shall be as shown in Figure 50.

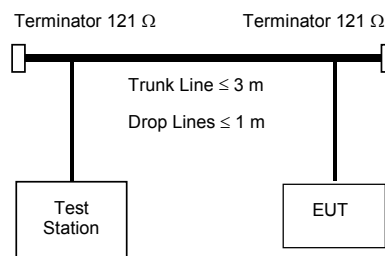


Figure 50 - Test circuit for logical tests

#### 9.3.2.3 Test procedure

##### 9.3.2.3.1 Successful duplicate MAC ID check

The EUT MAC ID shall be set to 0. Any necessary external power shall be applied to the EUT. The EUT shall then be physically connected to the network. The Test Station shall then be connected to the network and shall record the network traffic.

##### 9.3.2.3.2 Unsuccessful duplicate MAC ID check

The EUT MAC ID shall be set to 0. Any necessary external power shall be applied to the EUT. The EUT shall then be physically connected to the network. The Test Station shall respond to the first duplicate MAC ID check request message with a duplicate MAC ID check response message.

#### 9.3.2.4 Criteria for compliance

##### 9.3.2.4.1 Successful duplicate MAC ID check

It shall be verified that:

- the EUT transmits a duplicate MAC ID check request message;
- the EUT transmits a second duplicate MAC ID check request between 0,9 s and 1,5 s later;
- the EUT assumes the on-line state no earlier than 0,9 s after transmitting the second duplicate MAC ID check request message;
- the EUT transmits no other messages before assuming the on-line state.

##### 9.3.2.4.2 Unsuccessful duplicate MAC ID check

The EUT shall assume the communication fault state (see Table 35) after receiving the duplicate MAC ID check response message.

### **9.3.3 UCMM**

#### **9.3.3.1 Test purpose**

The purpose of this test is to determine whether a node that supports UCMM meets the requirements associated with the creation of explicit messaging connections.

#### **9.3.3.2 Test circuit**

The test circuit shall be as shown in Figure 50.

#### **9.3.3.3 Test procedure**

The Test Station shall transmit a UCMM open request message using Message Group 1. After opening this messaging connection an explicit message shall be transmitted across this connection. The Test Station shall transmit a UCMM close request message. The Test Station shall transmit a second explicit message and wait 10 s for a response.

The above procedure shall be repeated for Message Groups 2 and 3.

#### **9.3.3.4 Criteria for compliance**

It shall be verified that:

- the first explicit message is transmitted and received successfully in at least one Message Group;
- the EUT does not transmit a response to the second explicit message.

### **9.3.4 Allocation of predefined master/slave connection set - explicit messaging connection**

#### **9.3.4.1 Test purpose**

The purpose of this test is to determine whether the EUT meets the requirements associated with the allocation of a predefined Master/slave connection set for explicit messaging connections.

#### **9.3.4.2 Test circuit**

The test circuit shall be as shown in Figure 50.

#### **9.3.4.3 Test procedure**

The Test Station shall send a request to allocate the predefined master/slave connection set (allocation choice = 1). After allocating this messaging connection, an explicit message shall be transmitted across this connection. The Test Station shall de-allocate the predefined master/slave connection set (release choice = 1). The Test Station shall transmit a second explicit message and wait 10 s for a response.

#### **9.3.4.4 Criteria for compliance**

It shall be verified that:

- the first explicit message is transmitted and received as defined in the predefined master/slave connection set;
- the EUT does not transmit a response to the second explicit message.

### **9.3.5 Allocation of predefined master/slave connection set - I/O messaging connection**

#### **9.3.5.1 Test purpose**

The purpose of this test is to determine whether the EUT meets the requirements associated with the allocation of a predefined Master/slave connection set for I/O messaging connections.

#### **9.3.5.2 Test circuit**

The test circuit shall be as shown in Figure 50.

### 9.3.5.3 Test procedure

The Test Station shall send a request to allocate the predefined master/slave connection set. After allocating this messaging connection, an I/O message shall be transmitted across this connection. The Test Station shall de-allocate the predefined master/slave connection set. The Test Station shall transmit a second I/O message and wait 10 s for a response.

This procedure shall be performed for all allocation choices supported, with the exception of all explicit message allocation choices.

### 9.3.5.4 Criteria for compliance

It shall be verified that:

- the first I/O message is transmitted and received as defined in the predefined master/slave connection set;
- the EUT does not transmit a response to the second I/O message.

## Annex A (normative)

### Common services

#### A.1 DeviceNet service codes and names

Service codes and names are given in Table A.1 :

**Table A.1- DeviceNet service codes and names**

Service code (hex)	Service name
00	Reserved
01	Get_attributes_all
02	Set_attributes_all
03 - 04	Reserved
05	Reset
06	Start
07	Stop
08	Create
09	Delete
0A - 0C	Reserved
0D	Apply_attributes
0E	Get_attribute_single
0F	Reserved
10	Set_attribute_single
11 - 13	Reserved
14	Error response
15	Restore
16	Save
17	No_operation (NOP)
18 - 31	Reserved

#### A.2 Common service definitions

##### A.2.1 General

Object classes/instances that utilise these services shall provide a detailed description of their specific use and parameter formats, and whether or not any of these services are required. All request service data field parameters and all success response data field parameters shall be as shown in Table A.2 unless otherwise specified.

**Table A.2 - Service data and success response data field parameters**

Name	Data type	Description of parameter
Object specific data	Object/class service-specific	Contains class/instance specific parameters.

##### A.2.2 Get\_attributes\_all (Service code: 01<sub>hex</sub>)

###### A.2.2.1 General

Get\_attributes\_all returns the contents of all attributes of an object or class.

###### A.2.2.2 Service requirements

The following list details requirements associated with the get\_attributes\_all service:

- The structure of the information in the service data field of the response shall conform to the get\_attributes\_all response structure defined by the object or class.
- If the request is successfully serviced, the service data field of the response contains the attribute data. If an error is detected, then an error response is returned

### A.2.2.3 Request service data field parameters

None.

### A.2.2.4 Success response service data field parameters

Service data for get\_attributes\_all success response is given in Table A.3 :

**Table A.3 - Service data for get\_attributes\_all success response**

Name	Data type	Description of parameter
Attribute data	Object/class attribute-specific	A stream of information containing all of the attributes.

### A.2.3 Set\_attributes\_all (Service code: 02<sub>hex</sub>)

#### A.2.3.1 General

Modifies the contents of the attributes of the class or object.

#### A.2.3.2 Service requirements

The following list details requirements associated with the set\_attributes\_all service:

- The structure of the information in the service data field of the request conforms to the definition of the set\_attributes\_all request for the object or class.
- If the ability to modify an attribute changes based on the state of the object, the object definition shall provide a detailed description of how this service is affected.
- Attributes shall be modified only if all attribute specific value verifications are successful. The first attribute failing verification shall be specified in the additional code parameter of the error response message.
- If any other error is detected, then an error response is returned.
- If all verification checks are successful, then the attributes are modified and a set\_attributes\_all success response is returned.

### A.2.3.3 Request service data field parameters

Service data for set\_attributes\_all request is given in Table A.4 :

**Table A.4 - Service data for set\_attributes\_all request**

Name	Data type	Description of parameter
Attribute data	Object/class attribute-specific	A stream of information containing all of the attributes.

### A.2.3.4 Success response service data field parameters

None.

### A.2.4 Reset (Service code: 05<sub>hex</sub>)

#### A.2.4.1 General

Invokes the reset service of the specified class/object.

#### A.2.4.2 Service requirements

The following list details requirements associated with the reset service:

- If the execution of the reset service request would place the object/device in a state that enables it to respond to the requester, then the response shall not be returned until the service has completed. Otherwise, the object/device shall respond prior to executing the reset service.

- If an error is detected, then an error response shall be returned. Otherwise, a successful reset response shall be returned.

### **A.2.5 Start (Service code: 06<sub>hex</sub>)**

#### **A.2.5.1 General**

Invokes the start service of the specified class/object.

#### **A.2.5.2 Service requirements**

The following list details requirements associated with the start service:

- If an error is detected, then an error response shall be returned. Otherwise, a successful start response shall be returned.

### **A.2.6 Stop (Service code: 07<sub>hex</sub>)**

#### **A.2.6.1 General**

Invokes the stop service of the specified class/object.

#### **A.2.6.2 Service requirements**

The following list details requirements associated with the stop service:

- If an error is detected, then an error response shall be returned. Otherwise, a successful stop response shall be returned.

### **A.2.7 Create (Service code: 08<sub>hex</sub>)**

#### **A.2.7.1 General**

Creates a new object within the specified class.

#### **A.2.7.2 Service requirements**

The following list details requirements associated with the create service:

- The object instance is created and initialized in accordance with the object definition.
- If an error is detected, the object instance shall not be created, and an error response shall be returned. Otherwise, a successful create response shall be returned.

#### **A.2.7.3 Success response service data field parameters**

Service data for create success response is given in Table A.5 :

**Table A.5 - Service data for create success response**

Name	Data type	Description of parameter
Instance ID	UINT	The integer value assigned to identify the newly created object.
Object specific data	Object/class service-specific	Contains class/instance specific parameters.

### **A.2.8 Delete (Service code: 09<sub>hex</sub>)**

#### **A.2.8.1 General**

Deletes an object instance of the specified class.

#### **A.2.8.2 Service requirements**

The following list details requirements associated with the delete service:

- All resources are released.

- If an error is detected, then an error response shall be returned. Otherwise, a successful delete response shall be returned.

### A.2.9 Apply\_attributes (Service code: 0D<sub>hex</sub>)

#### A.2.9.1 General

Causes pending attribute values to become active.

#### A.2.9.2 Service requirements

The following list details requirements associated with the apply\_attributes service:

- Data for pending attributes shall be verified before it is used.
- If an error is detected, then an error response shall be returned. Otherwise, a successful apply\_attributes response shall be returned.

### A.2.10 Get\_attribute\_single (Service code: 0E<sub>hex</sub>)

#### A.2.10.1 General

Returns the contents of the specified attribute.

#### A.2.10.2 Service requirements

The following list details requirements associated with the get\_attribute\_single service:

- The service causes the class/object to return the contents of the specified attribute.
- If an error is detected, then an error response shall be returned. Otherwise, a successful get\_attribute\_single response shall be returned along with the requested attribute data.

#### A.2.10.3 Request service data field parameters

Service data for get\_attribute\_single request is given in Table A.6 :

**Table A.6- Service data for get\_attribute\_single request**

Name	Data type	Description of parameter
Attribute ID	USINT	Identifies the attribute to be read/returned

#### A.2.10.4 Success response service data field parameters

Service data for get\_attribute\_single success response is given in Table A.7 :

**Table A.7 - Service data for get\_attribute\_single success response**

Name	Data type	Description of parameter
Attribute data	Attribute-specific	Contains the requested attribute data

### A.2.11 Set\_attribute single (Service code: 10<sub>hex</sub>)

#### A.2.11.1 General

Modifies an attribute value.

#### A.2.11.2 Service requirements

The following list details requirements associated with the set\_attribute\_single service:

- The attribute data is verified prior to the modification taking place.
- If an error is detected, then an error response shall be returned. Otherwise a successful set\_attribute\_single response shall be returned.

### A.2.11.3 Request service data field parameters

Service data for set\_attribute\_single request is given in Table A.8 :

**Table A.8 - Service data for set\_attribute\_single request**

Name	Data type	Description of parameter
Attribute ID	USINT	Identifies the attribute to be read/returned
Attribute data	Attribute specific	Contains the value to which the specified attribute is to be modified.

### A.2.12 Error response (Service code: 14<sub>hex</sub>)

#### A.2.12.1 General

This service is used to respond negatively to a previously received explicit request message. This code shall only be present in an explicit response message.

#### A.2.12.2 Required behaviour

Not applicable.

#### A.2.12.3 Request service data field parameters

Not applicable.

#### A.2.12.4 Service data field parameters

Service data for error response request is given in Table A.9 :

**Table A.9 - Service data for error response request**

Name	Data type	Description of parameter
General error code	USINT	Identifies the encountered error. See Annex B for a list of general error codes.
Additional code	USINT	Contains an object/service specific value that further describes the error condition. If the responding object has no additional information to specify, then the value 255 shall be placed within this field.

### A.2.13 Restore (Service Code: 15<sub>hex</sub>)

#### A.2.13.1 General

Restores the contents of a class/object's attributes from a storage location accessible by the save service. Attribute data is copied from a storage area to the currently active memory area used by the class/object.

#### A.2.13.2 Service requirements

The following list details requirements associated with the restore service:

- Attribute data shall be verified before the copy from the "storage area" to the "actively used" area is performed.
- If the ability to modify an attribute changes based on the state of the object, the object definition shall provide a detailed description of how this service is affected.
- Attributes shall be modified only if all attribute specific value verifications are successful. If verification fails, an error response shall be returned. The first attribute failing verification shall be specified in the additional code parameter of the error response message.
- If any other error is detected, then an error response shall be returned.
- If all verification checks are successful, then the attributes are modified and a restore success response shall be returned.



#### **A.2.14 Save (Service Code: 16<sub>hex</sub>)**

##### **A.2.14.1 General**

Copies the contents of a class/object's attributes to a location accessible by the restore service.

##### **A.2.14.2 Service requirements**

The following list details requirements associated with the save service:

- The service shall report success only after the copy has been completed and verified.
- If an error is detected, then an error response shall be returned. Otherwise, a successful save response shall be returned.

#### **A.2.15 No\_operation (NOP) (Service Code: 17<sub>hex</sub>)**

##### **A.2.15.1 General**

This service causes the receiving object to return a no\_operation response. The receiving object shall not perform any other internal action.

##### **A.2.15.2 Required behaviour**

If the object to which the request is delivered supports the service, then a success response shall be returned. Otherwise, an error response shall be returned.

##### **A.2.15.3 Request service data field parameters**

None.

##### **A.2.15.4 Success response service data field parameters**

None.

## Annex B (normative)

### DeviceNet error codes

Table B.1 lists the error codes that may be present in the general error code field of an error response message.

**Table B.1 - DeviceNet error codes**

Error code (hex)	Error name	Description of error
00 - 01		Reserved
02	Resource unavailable	Resources needed for the object to perform the requested service are unavailable
03 - 07		Reserved
08	Service not supported	The requested service was not implemented or was not defined for this object class/instance
09	Invalid attribute value	Invalid attribute data detected
0A		Reserved
0B	Already in requested mode/state	The object is already in the mode/state being requested by the service
0C	Object state conflict	The object is unable to perform the requested service in its current mode/state
0D		Reserved
0E	Attribute not settable	The attribute specified may not be modified
0F	Privilege violation	A permission/privilege check failed
10	Device state conflict	The current mode/state of the node does not allow the execution of the requested service
11	Reply data too large	The data to be transmitted in the response buffer is larger than the allocated response buffer
12		Reserved
13	Not enough data	The service did not supply enough data to perform the specified operation.
14	Attribute not supported	The attribute specified in the request is not supported
15	Too much data	The service supplied more data than was expected
16	Object does not exist	The object specified does not exist in the node.
17		Reserved
18	No stored attribute data	The attribute data of this object was not saved prior to the requested service.
19	Store operation failure	The attempt to save the attribute data of this object was not successful
1A - 1E		Reserved
1F	Manufacturer specific error	A manufacturer specific error has been encountered. The additional code field of the error response defines the particular error encountered. This general error code shall only be used when none of the error codes which are either listed in this table or defined within an object class definition accurately reflect the error.
20	Invalid parameter	A parameter associated with the request was invalid. This code shall be used when a parameter does not meet the requirements of this specification and/or the requirements defined in an application object specification.
21 - CF		Reserved
D0 - FF		Reserved for object class specific error codes

## Annex C (normative)

### Connection path attribute definition

#### C.1 General

The path attribute definition defines the specific application object(s) to/from which data is moved. These attributes may optionally contain configuration information for the referenced application object(s).

A connection path attribute is divided into the following segments:

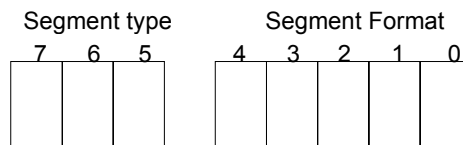
- Logical segment;
- Symbolic segment;
- Data segment.

A segment contains information defining the segment type, the format of the segment type data, and the actual segment data.

#### C.2 Segment type format

##### C.2.1 Introduction

The first byte of each segment of a path attribute is the segment type/format byte. This byte determines how the segment is to be interpreted. (See Figure C.1).

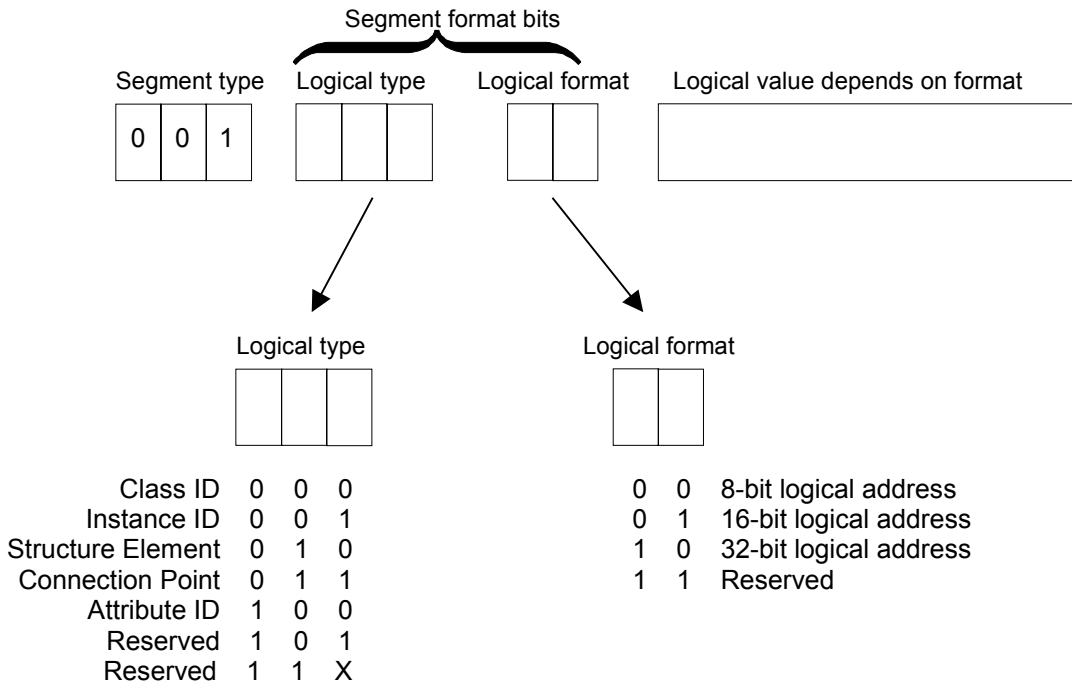


**Figure C.1 - Segment type/format**

The meaning of the segment format bits is determined by the specified segment type. The segment type bits and associated segment formats are described in the following sections.

##### C.2.2 Logical segment

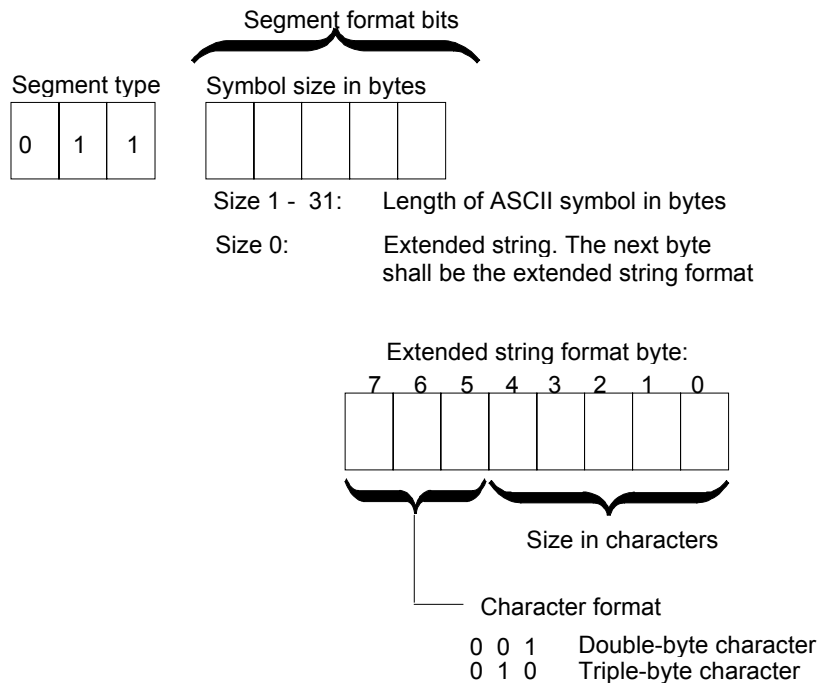
The logical segment selects a particular application object class, application object instance, and/or application object attribute within a node. (See Figure C.2).



**Figure C.2 - Logical segment**

**C.2.3 Symbolic segment**

The symbolic segment contains a string symbol, which shall be interpreted by the device. Symbols may contain a maximum of 31 characters. (See Figure C.3).

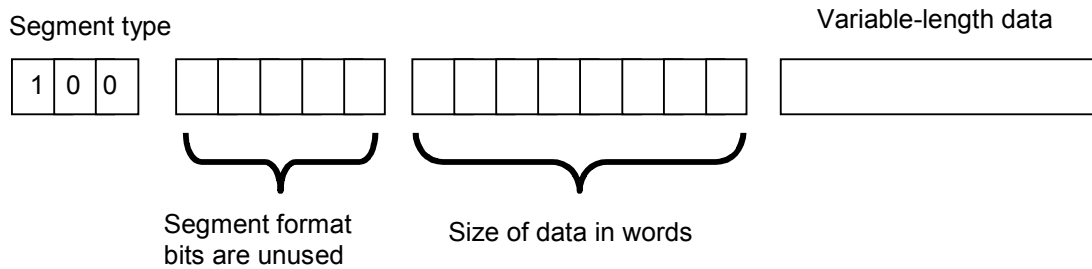


**Figure C.3 - Symbolic segment**

Licensed Copy: :FULLNAME, : DATE, Uncontrolled Copy, (c) BSI

### C.2.4 Data segment

The data segment contains parameters for the target application. The size of the data segment shall be specified in number of 16 bit words and depends on the application. The data segment may be any number of 16 bit words up to 255. A path may contain more than one data segment if required. (see Figure C.4).



**Figure C.4 – Data segment**

The data may be configuration information for the object, additional parameters necessary for a connection or any other set of application object specific information.

### C.3 Segment definition rules

The definition of any rules related to the specification of the logical address and any associated data segment information shall be the responsibility of the application object class i.e. an application object class shall define the means by which it is referenced by a path.

## Annex D (normative)

### Data type specification and encoding

#### D.1 Data type specification

##### D.1.1 General

The specification of a data type comprises:

- the range of values that variables of the type may assume;
- the operations performed on these variables.

Both elementary and derived data types as defined in EN 61131-3 are specified. In addition, because the function blocks defined in EN 61131-3 have both an associated data structure and a set of defined standard operations, these elements are also specified as data types .

##### D.1.2 Data type values

Data is made up of elementary (primitive) data types. These elementary data types are used to construct derived (constructed) data types.

The elementary data types and the values (range) of variables of each type are given in Table D.1 together with the values of certain parameters which are identified as “implementation-dependent” in EN 61131-3.

**Table D.1- Elementary data types**

Keyword	Description	Range	
		Minimum	Maximum
BOOL	Boolean	1)	
SINT	Short integer	-128	127
INT	Integer	-32,768	32,767
DINT	Double integer	$-2^{31}$	$2^{31}-1$
LINT	Long integer	$-2^{63}$	$2^{63}-1$
USINT	Unsigned short integer	0	255
UINT	Unsigned integer	0	65,535
UDINT	Unsigned double integer	0	$2^{32}-1$
ULINT	Unsigned long integer	0	$2^{64}-1$
REAL	Floating point	2)	
LREAL	Long floating point	3)	
ITIME	Duration (short)	12)	
TIME	Duration	4)	
FTIME	Duration (high resolution)	5), 6)	
LTIME	Duration (long)	6), 7)	
DATE	Date only	8)	
TIME_OF_DAY or TOD	Time of day	9)	
DATE_AND_TIME or DT	Date and time of day	10)	
STRING	character string (1 byte per character)	6)	
STRING2	character string (2 bytes per character)	6)	
STRINGN	character string (N bytes per character)	6)	
SHORT_STRING	character string (1 byte per character, 1 byte length indicator)	6)	
BYTE	bit string - 8 bits	11)	
WORD	bit string - 16 bits	11)	
DWORD	bit string - 32 bits	11)	
LWORD	bit string - 64 bits	11)	

Keyword	Description	Range	
		Minimum	Maximum
STRUCT	Structure of data types	Not applicable	
1) The possible values of variables of type BOOL shall be 0 and 1, corresponding to FALSE and TRUE respectively.			
2) The range and encoding of values for variables of type REAL are defined in EN 61131-3 for the basic single floating point format.			
3) The range and encoding of values for variables of type LREAL are defined in EN 61131-3 for the basic double floating point format.			
4) The range and encoding of values for variables of type TIME is the same as for variables of type DINT, representing the time duration in milliseconds i.e. a range of -24d20h31m23,648s to +24d20h31m23,647s.			
5) The range and encoding of values for variables of type FTIME is the same as for variables of type DINT, representing the time duration in microseconds i.e. a range of -35m47,483648s to +35m47,483547s.			
6) This is an extension to EN 61131-3.			
7) The range and encoding of values for variables of type LTIME is the same as for variables of type LINT, representing the time duration in microseconds i.e. a range of -106751991d4h0m54,775808s to +106751991d4h0m54,775807s.			
8) The encoding of values for variables of type DATE is the same as for variable of type UINT representing the range from 1972-01-01, the start of the Co-ordinated Universal Time (UTC) era to 2151-06-06 (a total range of 65,536 days).			
9) The encoding of values for variables of type TIME_OF_DAY is the same as for values of type UDINT representing the range from TOD = 00:00:00,000 to TOD = 23:59:59,999 with a resolution of 1 ms.			
10) The range of values for variables of type DATE_AND_TIME is from 1972-01-01-00:00:00,000 to 2151-06-06-23:59:59,999.			
11) Values of bit string data types is in the range $b_{N-1}$ to $b_0$ where N is the number of bits.			
12) The range and encoding of values for variables of type ITIME is the same as for variables of type INT, representing the time duration in milliseconds i.e. a range of -32s768 ms to +32s767 ms.			

### D.1.3 Language extensions

The following language extensions to EN 61131-3 may be used in EN 50325-2:

- ITIME data type
- FTIME data type
- LTIME data type
- STRING2 data type
- STRINGN data type
- SHORT\_STRING data type;
- Structured bit string types;
- Operations on STRING2 variables;
- Numbered bit string access;
- Structured bit string access.

### D.2 Data type encoding

#### D.2.1 General

Values shall be encoded with the least significant byte first. The least significant bit of each byte shall also be encoded first. The encoding for each data type is described below.

#### D.2.2 Boolean

A Boolean value shall be encoded in a single byte, with the value 0 for FALSE and 1 for TRUE.

#### D.2.3 SINT, USINT, and BYTE

These 8 bit values shall be encoded in a single byte.

#### D.2.4 INT, UINT, and WORD

These 16 bit values shall be encoded in two bytes, least significant byte first.

#### D.2.5 REAL, DINT, UDINT, and DWORD

These 32 bit values shall be encoded in four bytes, least significant byte first. The assignment of a REAL value uses the EN 61131-3 notation.

#### **D.2.6 LREAL, LINT, ULINT, and LWORD**

These 64 bit values shall be encoded in eight bytes, least significant byte first. The assignment of an LREAL value uses the EN 61131-3 notation.

#### **D.2.7 STRING**

This value shall be encoded as a 16 bit character count value, least significant byte first, followed by one byte characters.

#### **D.2.8 STRING2**

This value shall be encoded as a 16 bit character count value, least significant byte first, followed by two byte characters. The two byte characters are encoded as least significant byte first.

#### **D.2.9 STRINGN**

This value shall be encoded as a 16 bit character size value (n), least significant byte first, followed by 16 bit character count value, least significant byte first, followed by n byte characters. The n byte characters are encoded as least significant byte first.

#### **D.2.10 SHORT\_STRING**

This value shall be encoded as a 8 bit character count value followed by one byte characters.

#### **D.2.11 DATE\_AND\_TIME**

This value shall be encoded as TIME\_OF\_DAY followed by DATE.

#### **D.2.12 STRUCT**

This value is a group of data types.



## Annex E (normative)

### Communication Objects library

#### E.1 General

DeviceNet communication in a node is modelled as a collection of objects. An object provides an abstract representation of a data structure within a node.

An object class is a set of objects that all represent the same type of object. An object instance is the actual representation of a particular object within a class. Each instance of a class has the same set of attributes, but has its own particular set of attribute values.

An object instance and/or an object class have attributes, provide services and implement behaviour.

Attributes are characteristics of an object and/or an object class. Attributes provide status information or govern the operation of an object. Services are invoked to trigger the object class or instance to perform a task. The behaviour of an object indicates how it responds to particular events.

#### E.2 Object Class Codes

Table E.1 lists object class codes and their names:

**Table E.1 - Object class codes and names**

Code	Name
01 <sub>hex</sub>	Identity
02 <sub>hex</sub>	Message Router
03 <sub>hex</sub>	DeviceNet
05 <sub>hex</sub>	DeviceNet Connection
2B <sub>hex</sub>	Acknowledge Handler
NOTE: All other public class codes are reserved.	

#### E.3 Identity object (class code: 01<sub>hex</sub>)

##### E.3.1 General

The identity object identifies and provides general information about the node. The identity object shall be present in all DeviceNet nodes.

If autonomous components of a node exist, multiple instances of the identity object shall be used.

##### E.3.2 Class attributes

The identity object instance attributes are described in Table E.2 :

**Table E.2 - Identity object class attributes**

Attribute ID	Need in implementation	Access rule	Name	DeviceNet Data type	Description of attribute	Value
1	Optional	Get	Revision	UINT	Revision of this object.	The value assigned to this attribute shall be one (1).
2	Optional	Get	Max. instance	UINT	Maximum instance number of an object currently created in this class level of the node.	The largest instance number of a created object at this class hierarchy level.
NOTE: All other public attribute IDs are reserved.						

### E.3.3 Instance attributes

The identity object instance attributes are described in Table E.3 :

**Table E.3 - Identity object instance attributes**

Attribute ID	Need in implementation	Access Rule	Name	DeviceNet data type	Description of attribute
1	Required	Get	Manufacturer	UINT	Identification of each manufacturer by number
2	Required	Get	Device type	UINT	Indication of general type of product
3	Required	Get	Product code	UINT	Identification of a particular product of an individual manufacturer
4	Required	Get	Revision	STRUCT	Revision of the item the identity object represents
			Major revision	USINT	
			Minor revision	USINT	
5	Required	Get	Status	WORD	Summary status of device
6	Required	Get	Serial_number	UDINT	Serial number of device
7	Required	Get	Product name	SHORT_STRING	Human readable identification
8	Optional	Get	State	USINT	Present state of the device

NOTE: All other public attribute IDs are reserved.

The following terms are used in the name of the identity object instance attribute:

**Manufacturer:** A unique value allocated to each manufacturer.

**Device type:** A device type is used to identify similar devices which:

- exhibit the same behaviour;
- produce and/or consume the same set of data;
- contain the same set of configurable attributes.

Additionally, the device type will also provide a scope for the product code numbers.

See Annex F for definition of allowed device types values.

**Product code:** The product code identifies a product within a particular device type. Each manufacturer shall assign a product code to each of its products. The value zero is invalid and shall not be used.

**Revision:** The revision attribute, which consists of major and minor revisions, identifies the revision of the item that the identity object is representing. The major revision attribute is limited to 7 bits - the eighth bit shall be set to zero. The value zero is not valid for either field.

**Status:** This attribute represents the current status of the entire node. Its value changes as the state of the node changes. The status attribute is a word, with the bit definitions as shown in Table E.4 :

**Table E.4 - Bit definitions for status instance attribute of identity object**

Bit (s):	Called:	Definition
0	Owned	TRUE indicates that the node (or an object within the node) has an owner. For master/slave communications the setting of this bit means that the predefined master/slave connection set has been allocated to a master. For non-master/slave communications the meaning of this bit is not defined.
1		Shall be set to 0
2	Configured	TRUE indicates the application of the node has been configured to do something different to the initial default. This does not include configuration of the communications.
3		Shall be set to 0
4,5,6,7		Manufacturer specific
8	Minor recoverable fault	TRUE indicates the node detected a recoverable problem with itself. The problem does not cause the node to go into one of the faulted states. See E.3.6.
9	Minor unrecoverable fault	TRUE indicates the node detected an unrecoverable problem with itself,. The problem does not cause the node to go into one of the faulted states. See E.3.6
10	Major recoverable fault	TRUE indicates the node detected a problem with itself, which caused the node to go into the "major recoverable fault" state. See E.3.6
11	Major unrecoverable fault	TRUE indicates the node detected a problem with itself, which caused the node to go into the "major non-recoverable fault" state. See E.3.6
12, 13 ,14,15		Shall be set to 0

The events that constitute a fault (recoverable or unrecoverable) shall be determined by the manufacturer.

**Serial Number:** See 0.

**Product name:** 32 character string (maximum) defined by the manufacturer.

**State:** The state of the node shall be represented as follows:

- 0 = Non-existent
- 1 = Device Self Testing
- 2 = Standby
- 3 = Operational
- 4 = Major Recoverable Fault
- 5 = Major Unrecoverable Fault
- 6 - 254 = Reserved
- 255 = Default value if attribute is not supported

### E.3.4 Common services

#### E.3.4.1 General

The identity object shall provide the common services listed in Table E.5 :

**Table E.5 - Identity object common services**

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0E <sub>hex</sub>	Optional <sup>1)</sup>	Required	Get_attribute_single	Returns the contents of the specified attribute.
05 <sub>hex</sub>	Optional	Required	Reset	Invokes the reset service for the device.
01 <sub>hex</sub>	Optional	Optional	Get_attributes_all	Returns a predefined listing of this objects attributes (See E.3.4.3 ).

1) The get\_attribute\_single service is required if any attributes are implemented.

### E.3.4.2 Reset service

When the identity object receives a reset request, it shall:

- determine if it is able to provide the type of reset requested;
- respond to the request;
- attempt to perform the type of reset requested.

The reset service has the object-specific parameter shown in Table E.6 :

**Table E.6 - Identity reset request**

Name	Type	Description of request parameters
Type	USINT	Type of reset

The parameter type for the reset service has the bit specifications shown in Table E.7 :

**Table E.7 - Identity reset service parameter values**

Value:	Type of reset:
0	Emulate as closely as possible cycling power on the item the identity object represents. This value is the default if this parameter is omitted.
1	Return as closely as possible to the initial configuration, then emulate cycling power as closely as possible.

### E.3.4.3 Get\_attributes\_all response

At the class level, the order of the attributes returned in the “object/service specific reply data” portion of the get\_attributes\_all response shall be as shown in Table E.8 :

**Table E.8 - Identity get\_attributes\_all response, class level**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Revision (low byte); default = 1							
1	Revision (high byte); default = 0							
2	Max. instance (low byte); default = 1							
3	Max. instance (high byte); default = 0							
NOTE: Default values shall be used for all unsupported attributes.								

At the instance level, the order of the attributes returned in the “object/service specific reply data” portion of the get\_attributes\_all response shall be as shown in Table E.9 :

**Table E.9 - Identity get\_attributes\_all response, instance level**

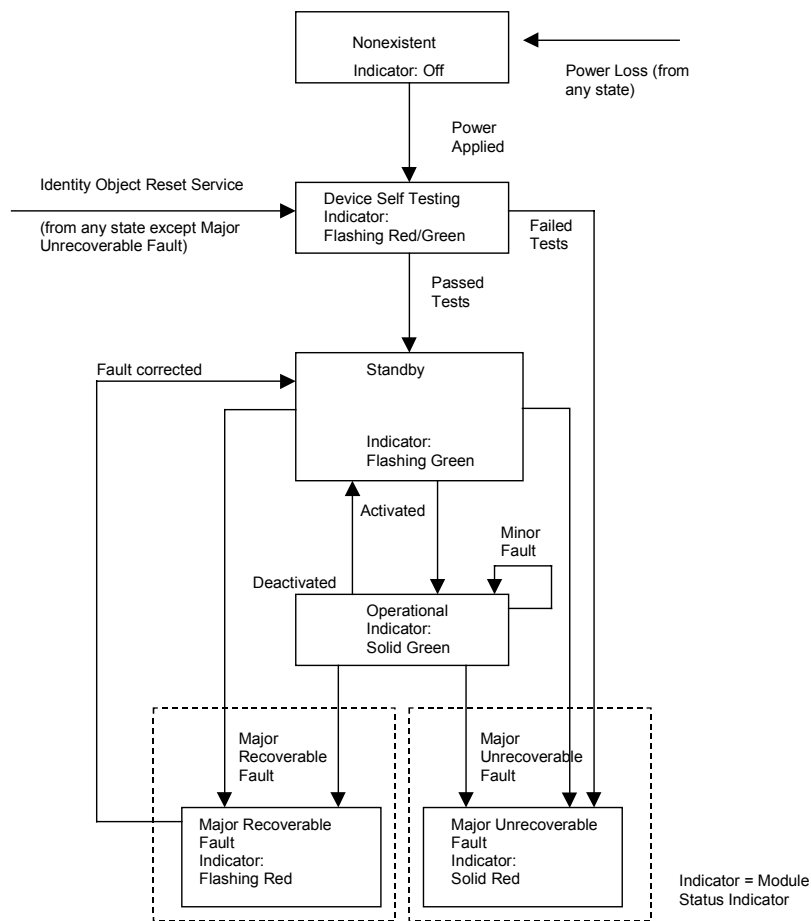
Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Manufacturer (low byte)							
1	Manufacturer (high byte)							
2	Device type (low byte)							
3	Device type (high byte)							
4	Product code (low byte)							
5	Product code (high byte)							
6	Major revision							
7	Minor revision							
8	Status (low byte)							
9	Status (high byte)							
10	Serial number (low byte)							
11	Serial number							
12	Serial number							
13	Serial number (high byte)							
14	Product name length							
15	Product name (1st character)							
16	Product name (2nd character)							
n	Product name (last character)							
n+1	State default = 255							
NOTE: Default values shall be inserted for all unsupported attributes.								

### E.3.5 Object-specific services

The identity object provides no object-specific services.

### E.3.6 Behaviour

The behaviour of the identity object is shown in the state transition diagram in Figure E.1. This state transition diagram associates the state of the device with the status reported by the status attribute and the state of the module status indicator.



**Figure E.1 - State transition diagram for identity object**

The state transition diagram for the identity object contains the following events:

- Power applied: the device has power applied;
- Power loss: the device is no longer powered;
- Passed tests: the node has successfully passed all self tests;
- Activated: the device's configuration is valid and the application for which the device was designed is now capable of executing (communications channels may or may not yet be established);
- Deactivated: the device's configuration is no longer valid and the application for which the device was designed is no longer capable of executing (communication channels may or may not still be established);
- Minor fault: an event classified as either a minor unrecoverable fault or a minor recoverable fault has occurred;
- Major recoverable fault: an event classified as major recoverable fault has occurred;
- Major unrecoverable fault: an event classified as a major unrecoverable fault has occurred;
- Fault Corrected: major recoverable fault corrected or cleared;
- Failed Test(s): a major unrecoverable fault;
- Reset Service: See E.3.4.2

The state event matrix for the identity object (see Table E. 10) contains the following states:

- **Non-existent:** power is not applied to the device
- **Device self testing:** the device is executing its self tests
- **Standby:** the device needs commissioning due to an incorrect or incomplete configuration
- **Operational:** the device is operating in a fashion that is normal for the device
- **Major recoverable fault:** the device has experienced a recoverable fault
- **Major unrecoverable fault:** the device has experienced an unrecoverable fault

**Table E. 10 - State event matrix for identity object**

Event	Non-existent	Device self testing	Standby	Operational	Major unrecoverable fault	Major recoverable fault
Power loss	Not applicable	Transition to non-existent	Transition to non-existent	Transition to non-existent	Transition to non-existent	Transition to non-existent
Power applied	Transition to device self testing	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable
Failed tests	Not applicable	Transition to major unrecoverable fault	Not applicable	Not applicable	Not applicable	Not applicable
Passed tests	Not applicable	Transition to standby	Not applicable	Not applicable	Not applicable	Not applicable
Deactivated	Not applicable	Ignore event	Ignore event	Transition to standby	Ignore event	Ignore event
Activated	Not applicable	Ignore event	Transition to operational	Ignore event	Ignore event	Ignore event
Major recoverable fault	Not applicable	Not applicable	Transition to major recoverable fault	Transition to major recoverable fault	Ignore event	Ignore event
Major unrecoverable fault	Not applicable	Not applicable	Transition to major unrecoverable fault	Transition to major unrecoverable fault	Ignore event	Ignore event
Minor fault	Not applicable	Ignore event	Ignore event	Ignore event	Ignore event	Ignore event
Fault corrected	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	Transition to Standby
Reset	Not applicable	Restart self tests	Transition to device self testing	Transition to device self testing	Ignore event	Transition to device self testing

#### E.4 Message router object (class code: 02<sub>hex</sub>)

##### E.4.1 General

The message router object provides a messaging connection point through which a client may address a service to any object class or instance residing in the node.

##### E.4.2 Class attributes

The message router class attributes are described in Table E.11 :

**Table E.11 - Message router class attributes**

Attribute ID	Need in implementation	Access rule	Name	DeviceNet data type	Description of attribute	Meanings of values
1	Optional	Get	Revision	UINT	Revision of this object	The value assigned to this attribute is one (1).

NOTE: All other public attribute IDs are reserved.

### E.4.3 Instance attributes

The message router instance attributes are described in Table E.12 :

**Table E.12 - Message router instance attributes**

Attribute ID	Need in implementation	Access rule	Name	DeviceNet data type	Description of attribute	Meanings of values
1	Optional	Get	Object_list	STRUCT	A list of supported objects	Structure with an array of class codes supported by the device
			Number	UINT	Number of members in the classes array	The number of class codes in the classes array
			Classes	Array of UINT	List of supported class codes	The class codes supported by the device
2	Optional	Get	Number available	UINT	Maximum number of connections supported	Count of the max. number of connections supported
3	Optional	Get	Number active	UINT	Number of connections currently used by system components	Current count of the number of connections allocated to system communication
4	Optional	Get	Active connections	Array of UINT	A list of the connection IDs of the currently active connections	Array of system connection IDs

NOTE: All other public attribute IDs are reserved.

### E.4.4 Common services

#### E.4.4.1 General

The message router object shall provide the following common services (see Table E.13) :

**Table E.13 - Message router common services**

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0E <sub>hex</sub>	Optional <sup>1)</sup>	Optional <sup>1)</sup>	Get_attribute_single	Returns the contents of the specified attribute.
01 <sub>hex</sub>	Optional	Optional	Get_attributes_all	Returns the contents of all attributes

1) The get\_attribute\_single service is required if any attributes are implemented.

#### E.4.4.2 Get\_attributes\_all response

At the class level, the order of the attributes returned in the “object/service specific reply data” portion of the get\_attributes\_all response shall be as shown in Table E.14 :

**Table E.14 - Get\_attributes\_all response, class level**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Revision (low byte); default = 1							
1	Revision (high byte); default = 0							

NOTE: Default values shall be used for all unsupported attributes.

At the instance level, the order of the attributes returned in the “object/service specific reply data” portion of the get\_attributes\_all response shall be as shown in Table E.15 :



**Table E.15 - Get\_attributes\_all response, instance level**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Object_list : number (low byte); default = 0							
1	Object_list : number (high byte); default = 0							
2	Object_list : class #1 (low byte)							
3	Object_list : class #1 (high byte)							
2n	Object_list : class #n (low byte)							
2n+1	Object_list : class #n (high byte)							
2n+2	Number available (low byte); default = 0							
2n+3	Number available (high byte); default = 0							
2n+4	Number active (low byte); default = 0							
2n+5	Number active (high byte); default = 0							
2n+6	Active connections #1 (low byte)							
2n+7	Active connections #1 (high byte)							
2n+2 m+4	Active connections #m (low byte)							
2n+2 m+5	Active connections #m (high byte)							
NOTE 1: Default values shall be used for all unsupported attributes.								
NOTE 2: If the instance attribute "object_list" is not supported, the default value of zero shall be inserted into the response byte array without any object_list class numbers.								
NOTE 3: If the Instance attribute "number active" is not supported, the default value of zero shall be inserted into the response byte array without any active connection numbers.								

#### E.4.5 Object-specific services

The message router object provides no object-specific services.

#### E.4.6 Behaviour

##### E.4.6.1 General

The message router object shall receive explicit messages and perform the following functions:

- interpret the class instance specified in a message;
- route a service to the specified object;
- interpret services directed to it;
- route a response to the correct service source.

#### E.4.6.2 Service request

Interpretation of the class instance shall be performed on every service received by the message router. Any class instance that cannot be interpreted by a node's implementation of a message router shall report the object\_not\_found error. If no error is detected, the service shall then be routed to a target object.

#### E.4.6.3 Service response

All service responses shall be routed to the explicit messaging connection across which the service request was received.

#### E.5 DeviceNet object (class code: 03<sub>hex</sub>)

The DeviceNet object (see 5.3.3) provides the configuration and status of a DeviceNet port. Each DeviceNet device shall support only one DeviceNet object per physical connection to the DeviceNet communication link.

#### E.6 DeviceNet connection object (class code: 05<sub>hex</sub>)

The DeviceNet connection object manages the characteristics of a communication connection (see 5.3.2).

#### E.7 Acknowledge handler object (class code: 2B<sub>hex</sub>)

##### E.7.1 General

The acknowledge handler object manages the reception of message acknowledgements. This object communicates with a message-producing application object within a device. The acknowledge handler object notifies the producing application of acknowledge reception, acknowledge timeouts, and production retry limit.

##### E.7.2 Class attributes

The acknowledge handler object class attributes are described in Table E.16 :

**Table E.16 - Acknowledge handler object class attributes**

Attribute ID	Need in implementation	Access rule	Name	DeviceNet Data type	Description of attribute	Value
1	Optional	Get	Revision	UINT	Revision of this object.	The value assigned to this attribute shall be one (1).
2	Optional	Get	Max. instance	UINT	Maximum instance number of an object currently created in this class level of the node.	The largest instance number of a created object at this class hierarchy level.

NOTE: All other public attribute IDs are reserved.

##### E.7.3 Instance attributes

The acknowledge handler object instance attributes are described in Table E.17 :

**Table E.17 - Acknowledge handler instance attributes**

Attribute ID	Need in implementation	Access Rule	Name	DeviceNet data type	Description of attribute	Value
1	Required	Set	Acknowledge timer	UINT	Time to wait for acknowledge before resending	Range 1-65 535 ms (0 invalid); default = 16
2	Required (Get) Optional (Set)	Get/Set	Retry Limit	USINT	Number of Ack Timeouts to wait before informing the producing application of a RetryLimit_Reached event.	Range 0-255; default = 1
3	Required	Get/Set (Inactive) Get (Active)	COS producing connection instance	UINT	Connection instance which contains the path of the producing I/O application object which shall be notified of Ack Handler events.	Connection instance ID
4	Optional	Get	Ack List Size	BYTE	Maximum number of members in Ack List	0 = Dynamic; >0 = Max number of members
5	Optional	Get	Ack List	BYTE Array of UINT	List of active connection instances which are receiving Acks.	Number of members followed by list of: Connection Instance ID
6	Optional	Get	Data with Ack Path List Size	BYTE	Maximum number of members in Data with Ack Path List	0 = Dynamic; >0 = Max number of members
6	Optional	Get	Data with Ack Path List	BYTE Array of UINT USINT Array of USINT	List of connection instance/consuming application object pairs. This attribute shall be used to forward data received with acknowledgement	Number of members followed by list of: Connection Instance ID/DeviceNet path length/DeviceNet path
<p>NOTE 1 : If the specified value for the Acknowledge Timer attribute is not equal to an increment of the available clock resolution, then the value shall be rounded up to the next serviceable value.</p> <p>NOTE 2 : The value that shall be loaded into the Acknowledge Timer attribute shall be reported in the Service Data Field of a set_attribute_single response message associated with a request to modify this attribute.</p> <p>NOTE 3 : A successful set attribute to the Retry Limit attribute shall reset the Retry Counter.</p> <p>NOTE 4 : The Ack List attribute shall be updated when an associated connection transitions between configuring, established, timed-out, and non-existent. Refer to the state event matrix for details.</p> <p>NOTE 5 : If the acknowledge handler object is active (at least one member in the Ack List), the COS Producing Connection Instance attribute access shall be get only.</p> <p>NOTE 6 : The default value loaded into the Acknowledge Timer attribute at time of instantiation shall be 16 ms.</p> <p>NOTE 7 : The default value loaded into the Retry Limit attribute at time of instantiation shall be 1.</p> <p>NOTE 8 : All other public attribute IDs are reserved.</p>						

### E.7.4 Common services

The acknowledge handler object shall provide the common services shown in Table E.18 :

**Table E.18 - Acknowledge handler object common services**

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0E <sub>hex</sub>	Optional	Required	Get_attribute_single	Returns the contents of the specified attribute.
10 <sub>hex</sub>	Not applicable	Required	Set_attribute_single	Modifies an attribute value
08 <sub>hex</sub>	Optional	Not applicable	Create	Creates an acknowledge handler object
09 <sub>hex</sub>	Optional	Optional	Delete	Deletes an acknowledge handler object
NOTE: If the Delete service is addressed to the class, all instances shall be deleted. If it is addressed to a specific instance, only this one shall be deleted.				

### E.7.5 Object-specific services

The acknowledge handler object shall provide the object-specific services shown in Table E.19 :

**Table E.19 - Acknowledge handler object-specific services**

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
4B <sub>hex</sub>	Not applicable	Optional	Add_AckData_Path	Adds a path for data returned with acknowledgement for a connected consumer.
4C <sub>hex</sub>	Not applicable	Optional	Remove_AckData_Path	Removes a path for data returned with acknowledgement for the given connected consumer.

These services are used to add and remove paths for each of the connected acknowledge consumers when data is sent with the acknowledgement. See Table E.20 and Table E.21 :

**Table E.20 – Acknowledge handler Add\_AckData\_Path request**

Name	Type	Description of request parameters
Connection Instance ID	UINT	The integer value assigned to identify the producer connection
DeviceNet Path Length	UINT	Number of bytes in the DeviceNet Path attribute
DeviceNet Path	Array of USINT	Specifies the application object(s) which shall receive the data sent with the acknowledgement

**Table E.21 - Acknowledge handler Remove\_AckData\_Path request**

Name	Type	Description of request parameters
Connection Instance ID	UINT	The integer value assigned to identify the producer connection

### E.7.6 Behaviour and configuration of acknowledged data production

#### E.7.6.1 General

The following rules shall be used to configure and determine the behaviour of an acknowledged Change of State or Cyclic I/O connection using the acknowledge handler object. In the following examples, COS Producer shall be used to reference the device producing change of state or cyclic data and consuming an acknowledgement (client). COS Consumer shall be used to reference the device consuming the change of state or cyclic data and producing an acknowledgement (server).

### E.7.6.2 Acknowledged data production

The following rules shall be used:

- The COS Producer's consumed connection path shall be set to an available acknowledge handler object. The path shall consist of class and instance. If an acknowledge handler object is not available, then the acknowledge handler class create service shall be used to obtain a new one.
- The COS Producer's producing I/O application shall inform the acknowledge handler object of new data production (data sent event message) or data production retries (data resent event message).
- The COS Producer's acknowledgement reception shall be performed by the acknowledge handler object. The acknowledge handler object informs the producing I/O application when one or more acknowledgements have not been received within the acknowledge timeout (using the Ack List and Ack Timeout attributes).
- The acknowledge message requires no data. The COS producing device's acknowledge handler object shall consider valid message reception as an acknowledgement. However, a change of state or cyclic producing device may be configured to consume data along with the acknowledgement. In this case the data shall be forwarded to the application object in the "Data with Ack Path List" attribute, based on the connection which received the data.
- The COS Consumer acknowledge producing application shall be configured to send either a zero length message or a valid response (output) message when a valid input message may be consumed.
- An acknowledge timer shall be started each time production occurs. The acknowledge handler object shall be notified of this event by a data sent or data resent event message from the producing application.
- Expiry of the acknowledge timer shall cause an Acknowledge Timeout message to be sent to the producing application object. That object shall resend the last message if the Retry Limit has not been reached. It may also take an application specific action.
- The retry count shall be incremented each time an Acknowledge Timeout message is sent to the producing application. When the retry limit has been reached, a Retry Limit Reached message shall be sent to the producing application object.
- The retry count shall be cleared on each data sent message. A data resent message shall not clear the retry counter.
- The acknowledge timer value shall be configurable within the acknowledge handler object.
- The number of retries may be configurable within the acknowledge handler object.

### E.7.6.3 Acknowledged Change of State (using one connection object and one COS consumer)

The producer shall either perform or be aware of these steps:

- The producer shall create a connection object.
- The producer transportClass\_trigger attribute shall be set to Class 2/3, Change-Of-State, Client (13<sub>hex</sub> for Class 3) or Class 2/3, Cyclic, Client (03<sub>hex</sub> for Class 3).
- The produced connection path shall be set to the producing application, which shall support Change of State, or Cyclic production.
- The producer shall create an acknowledge handler object, and configure the "COS Producing Connection Instance" attribute to the instance of the connection object which specifies the producing I/O application object. The Ack List attribute shall be updated with the connection instance of the I/O connection object which shall consume an acknowledge as that connection transitions to the established state. Optionally, the producer may configure the Acknowledge Timer and Retry Count attributes.

- The consumed connection path shall be set to the acknowledge handler object created and configured for handling the acknowledge. This path contains class and instance only.
- The consumed connection size shall be set to the size of the data expected to be delivered to the object specified in the consumed path of the acknowledge handler object, or zero if no path is configured.
- The producer shall configure all other attributes in the same way as any other peer to peer connection.

The consumer shall either perform or be aware of these steps:

- The consumer transportClass\_trigger attribute is set to Class 2/3, Server (83<sub>hex</sub> for Class 3).
- The produced connection path shall be set to the consuming application (or some other application as this is product specific) for generating the acknowledge.
- The produced connection size shall be set to the size of the data to be delivered to the object specified in the consumed path of the change of state producing device's acknowledge handler object, or zero if no path is configured.
- The consumer shall configure all other attributes in the same way as any other peer to peer connection.

#### **E.7.6.4 Acknowledged Change of State (using multiple connection objects and COS consumers)**

The producer shall either perform or be aware of these steps:

- The producer shall create a Connection Object.
- The producer transportClass\_trigger attribute shall be set to Class 0, Change-Of-State, Client (10<sub>hex</sub>) or Class 0, Cyclic, Client (00<sub>hex</sub>).
- The produced connection path shall be set to the producing application, which shall support Change of State, or Cyclic production.
- The consumed connection path and consumed connection size shall not be configured by the producer.
- The producer shall configure all other attributes in the same way as any other peer to peer connection.
- The producer shall create a Connection Object for each COS consumer.
- The consumer transportClass\_trigger attribute shall be set to Class 0, Server (80<sub>hex</sub>) for each consuming connection object.
- The producer shall create an acknowledge handler object, and configure the "COS Producing Connection Instance" attribute to the instance of the connection object which specifies the producing I/O application object. The Ack List attribute shall be updated with the connection instance of each I/O connection object which shall consume an acknowledge as those connections transition to the established state. Optionally, the producer may configure the Acknowledge Timer and Retry Count attributes.
- The consumed connection path for each consuming connection object shall be set to the acknowledge handler object just created and configured for handling the acknowledge. The path contains class and instance only.
- The consumed connection sizes shall be set to the size of the data expected to be delivered to the consumed path set in the acknowledge handler object, or zero if no path is configured.
- The producer shall configure all other attributes in the same way as any other peer to peer connection.

Each consumer shall either perform or be aware of these steps:

- The consumer transportClass\_trigger attribute shall be set to Class 2/3, Server (83<sub>hex</sub> for Class 3).

- The produced connection path shall be set to the consuming application (or some other application as this is product specific) for generating the acknowledge.
- The produced connection size shall be set to the size of the data to be delivered to the consumed path set in the change of state producing device's acknowledge handler object, or zero if no path is configured.
- The consumer shall configure all other attributes in the same way as any other peer to peer connection.

Once the connection object(s) are configured, an `apply_attributes` service shall be sent to each configured connection object to transition the connection(s) to the `Established` state. During the `apply`, the connection object shall be 'delivered' to both the consuming and producing application objects for validation of the attribute information. At this time, the producing I/O application shall check for a change of state configuration by examining the `transportClass_trigger` attribute. If the Production Trigger bits are set to `change-of-state` or `cyclic`, the application shall configure itself for change of state or cyclic production. If change of state or cyclic production are not supported by the producing application object an error shall be returned to the `apply_attributes` service.

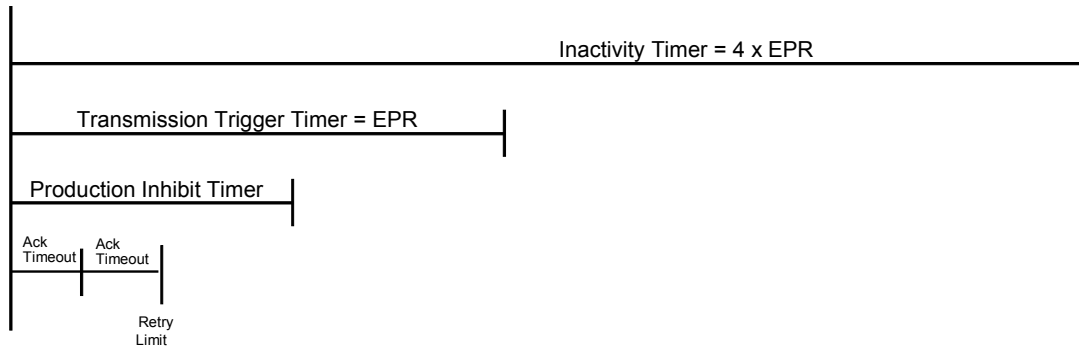
#### **E.7.6.5 Use of timers with acknowledged data production**

The following rules shall be observed when sending acknowledged data:

- New data shall not be sent while the Inhibit Timer is active (running).
- New data shall be sent when no acknowledge is pending, subject to the above rule. An acknowledge shall be pending after a send of new data or a retry of old data and until an Ack Timeout or Ack Received.
- Retrying old data shall occur at Ack Timeout if new data is not available or the Inhibit Timer is active.
- Sending new data (or old data on transmission trigger timeout) shall start the Ack Timer, Inhibit Timer, and the Transmission Trigger Timer. The Retry Counter shall be also cleared.
- Retrying old data shall start the Ack Timer.

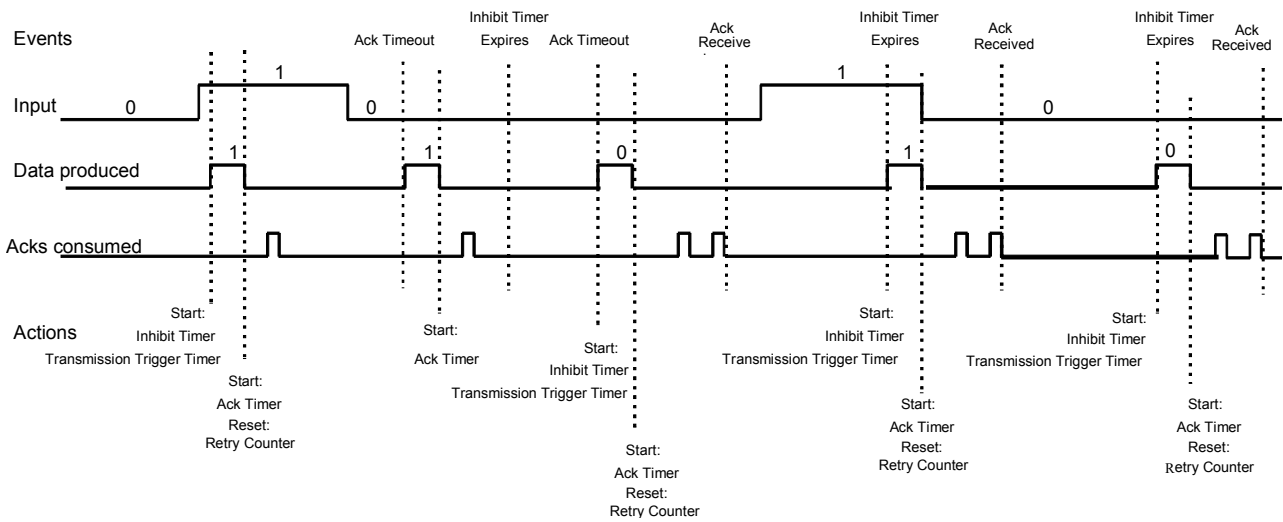
**E.7.6.6 Timing**

Figure E.2 shows typical timing relationships for acknowledged data production :



**Figure E.2 - Typical Timing Relationships for Acknowledged Data Production**

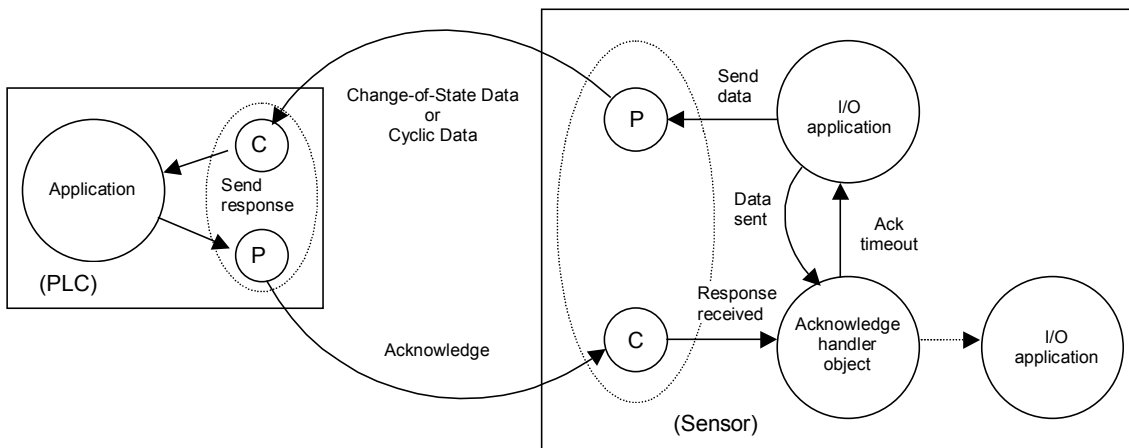
Figure E.3 shows a typical timing diagram for a Change of State system :



**Figure E.3 – Typical timing diagram for a Change of State system**

**E.7.6.7 Message flow**

Figure E.4 and Figure E.5 illustrate the message flow in a Change of State connection for both single and multi-consumer configurations :



**Figure E.4 - Message flow - one connection object, one consumer**

Licensed Copy: :FULLNAME, : DATE, Uncontrolled Copy, (c) BSI



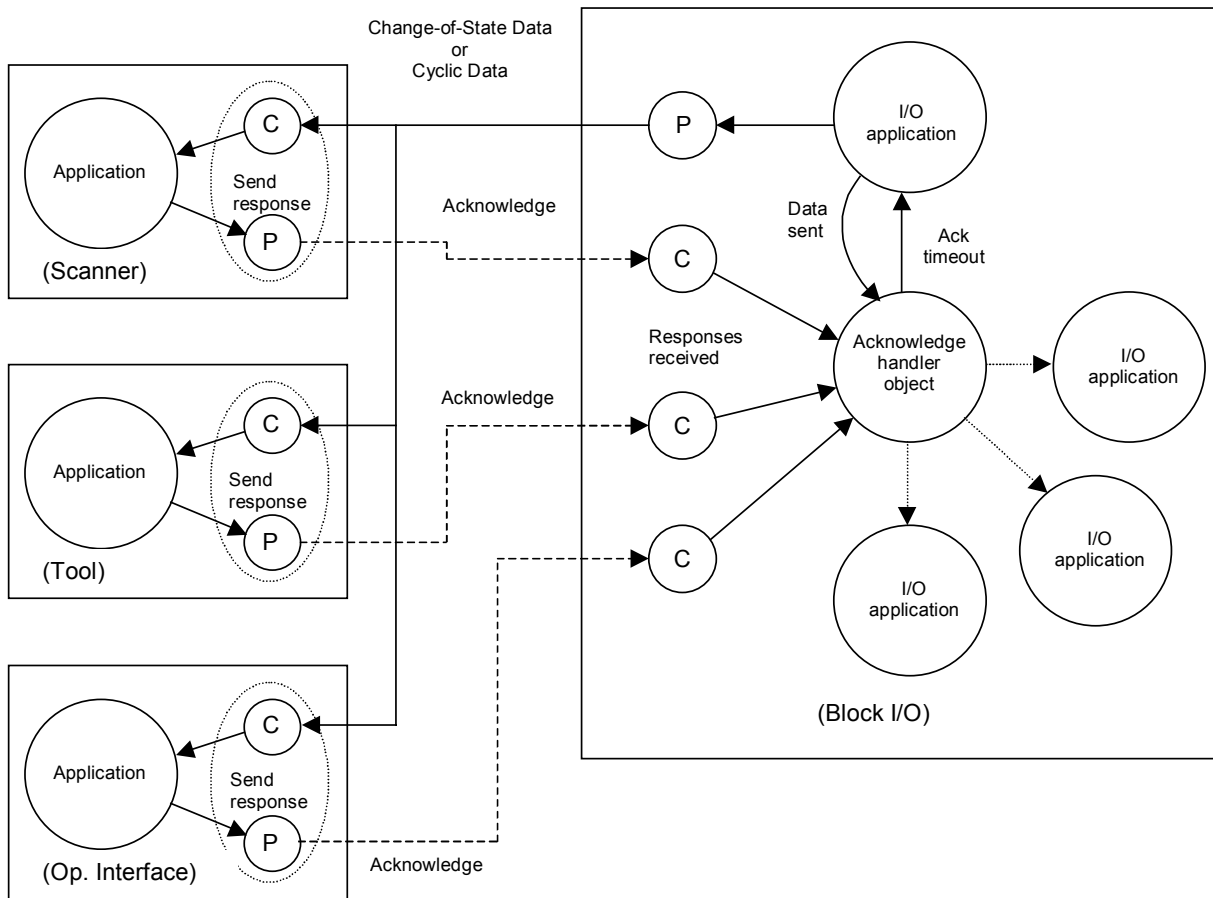


Figure E.5 - Message flow - multi-consumer configuration

**E.7.6.8 State event matrices**

Table E.22 and Table E.23 are the state event matrices for the producing I/O application object and acknowledge handler object associated with a Change of State connection.

**Table E.22 - State event matrix for producing I/O application object**

Event	State			
	Not Running	Running	Running with Acknowledgement	Prohibited
Change of State detected	Ignore event	Inform link producer to send data. If inhibit time configured: Start inhibit timer Transition to Prohibited	Inform link producer to send data. Send data_sent event message to acknowledge handler object. If inhibit time configured: Start inhibit timer Transition to Prohibited	Queue event
Acknowledge_received	Not applicable	Not applicable	Ignore event	If Ack_active set if inhibit timer not running Transition to Running with Acknowledgement else set Ack_received flag Else ignore event
Acknowledge_timeout	Ignore event	Not applicable	Inform link producer to send data Send data_resent event message to acknowledge handler object	Inform link producer to send data Send data_resent event message to acknowledge handler object
Transmission timer expires	Not applicable	Inform link producer to send data	Inform link producer to send data Send data_sent event message to acknowledge handler object	Inform link producer to send LAST data sent Send data_sent event message to acknowledge handler object
Retry limit reached	Ignore event	Not applicable	Product specific	Transition to Running with Acknowledgement
Inhibit timer expires	Ignore event	Not applicable	Not applicable	If Ack_active set if Ack_received Transition to Running with Acknowledgement Clear Ack_received flag else ignore event else Transition to Running
Connection deleted	Not applicable	Transition to Not Running	Transition to Not Running	Transition to Not Running
Acknowledge_Active	Set Ack_Active flag	Transition to Running with Acknowledgement Set Ack_Active flag	Ignore event	Set Ack_Active flag
Acknowledge_Inactive	Reset Ack_Active flag	Ignore event	Transition to Running Reset Ack_Active flag	Reset Ack_Active flag
Connection transitions to established	If Ack_Active Transition to Running with Acknowledgement If Ack_Inactive Transition to Running	Ignore event	Ignore event	Ignore event

NOTE : This is a partial state event matrix for a producing I/O application object. Only those states and events associated with data acknowledgement are defined. Other states and events are likely to be associated with a producing I/O application object.

Licensed Copy: :FULLNAME, : DATE, Uncontrolled Copy, (c) BSI

**Table E.23 - State event matrix for acknowledge handler object**

Event	State		
	Non-Existent	Inactive	Active
Receive Acknowledge	Not applicable	Ignore event	Clear Ack flag Forward any data to application object If all Acknowledges received Clear Ack timer and retry counter Send Acknowledge_received event message to producing application object
Acknowledge timer expires	Not applicable	Ignore event	Send Ack-Timeout event message to producing application object
Data_sent	Not applicable	Ignore event	Set Ack required flag and Ack timer Clear retry counter
Data_resent	Not applicable	Ignore event	Set Ack required flag and Ack timer If retry limit <>0 Increment retry counter If retry counter = retry_limit Send Retry_limit_reached event message to producing application object
Delete	Not applicable	Transition to Non-Existent	Transition to Non-Existent
Create	Transition to Inactive	Not applicable	Not applicable
Apply_attributes	Not applicable	Verify new connection may be added to list Pass this message to the consuming application object, if one is configured for this connection	Verify new connection may be added to list Pass this message to the consuming application object, if one is configured for this connection
Connection transitions to established	Not applicable	Add this connection instance to the connection list (or internally flag as "Acking") Pass this message to the consuming application object, if one is configured for this connection Send Acknowledge_active event message to producing application Transition to Active	Add this connection instance to the connection list Pass this message to the consuming application object, if one is configured for this connection
Inactivity/watchdog timer expires	Not applicable	Not applicable	Internally flag as "Not Acking" An acknowledge will no longer be monitored for this connection, however, it remains in the connection list Pass this event to the consuming application object, if one is configured for this connection If no "Acking" connections in list, send Acknowledge_Inactive event to producing application and transition to Inactive
Connection deleted	Not applicable	Ignore event	Remove this connection instance from the connection list Pass this event to the consuming application object, if one is configured for this connection If no "Acking" connections in list, send Acknowledge_Inactive event to producing application and transition to Inactive

## Annex F (normative)

### Value ranges

The following terms are used when defining value ranges:

- **Open:** A value whose meaning shall be the same for all DeviceNet users.
- **Manufacturer specific:** A range of values that are specific to the manufacturer of a device and which may be used to extend a device beyond the available open options.
- **Object class specific:** A range of values whose meaning is defined by an object class. This range applies to service code definitions.

Class ID value ranges and meanings are given in Table F.1 :

**Table F.1 - Class ID ranges**

Range	Meaning
00 - 63 <sub>hex</sub>	Open. These are referred to as DeviceNet public class codes and are defined in Annex E.
64 <sub>hex</sub> - C7 <sub>hex</sub>	Manufacturer specific
C8 <sub>hex</sub> - FF <sub>hex</sub>	Reserved
100 <sub>hex</sub> - 2FF <sub>hex</sub>	Open
300 <sub>hex</sub> - 4FF <sub>hex</sub>	Manufacturer specific
500 <sub>hex</sub> - FFFF <sub>hex</sub>	Reserved

Service code value ranges and meanings are given in Table F.2 :

**Table F.2 – Service code ranges**

Range	Meaning
00 - 31 <sub>hex</sub>	Open. These are referred to as DeviceNet common services and are defined in Annex A.
32 <sub>hex</sub> - 4A <sub>hex</sub>	Manufacturer specific
4B <sub>hex</sub> - 63 <sub>hex</sub>	Object class specific
64 <sub>hex</sub> - 7F <sub>hex</sub>	Reserved
80 <sub>hex</sub> - FF <sub>hex</sub>	Invalid

Attribute ID value ranges and meanings are given in Table F.3 :

**Table F.3 - Attribute ID ranges**

Range	Meaning
00 - 63 <sub>hex</sub>	Open
64 <sub>hex</sub> - C7 <sub>hex</sub>	Manufacturer specific
C8 <sub>hex</sub> - FF <sub>hex</sub>	Reserved

MAC ID value ranges and meanings are given in Table F.4 :

**Table F.4 - MAC ID range**

Range	Meaning
00 - 63 <sub>decimal</sub>	The MAC ID. The value 63 (decimal) shall be utilised upon initialisation of a device unless another value has been assigned.

Device type value ranges and meanings are given in Table F.5 :

**Table F.5 – Device type ranges**

Range	Meaning
00 <sub>hex</sub> - 63 <sub>hex</sub>	Open
64 <sub>hex</sub> - C7 <sub>hex</sub>	Manufacturer specific
C8 <sub>hex</sub> - FF <sub>hex</sub>	Reserved
100 <sub>hex</sub> - 2FF <sub>hex</sub>	Open
300 <sub>hex</sub> - 4FF <sub>hex</sub>	Manufacturer specific
500 <sub>hex</sub> - FFFF <sub>hex</sub>	Reserved

---

---

## BSI — British Standards Institution

BSI is the independent national body responsible for preparing British Standards. It presents the UK view on standards in Europe and at the international level. It is incorporated by Royal Charter.

### Revisions

British Standards are updated by amendment or revision. Users of British Standards should make sure that they possess the latest amendments or editions.

It is the constant aim of BSI to improve the quality of our products and services. We would be grateful if anyone finding an inaccuracy or ambiguity while using this British Standard would inform the Secretary of the technical committee responsible, the identity of which can be found on the inside front cover. Tel: 020 8996 9000. Fax: 020 8996 7400.

BSI offers members an individual updating service called PLUS which ensures that subscribers automatically receive the latest editions of standards.

### Buying standards

Orders for all BSI, international and foreign standards publications should be addressed to Customer Services. Tel: 020 8996 9001. Fax: 020 8996 7001. Standards are also available from the BSI website at <http://www.bsi-global.com>.

In response to orders for international standards, it is BSI policy to supply the BSI implementation of those that have been published as British Standards, unless otherwise requested.

### Information on standards

BSI provides a wide range of information on national, European and international standards through its Library and its Technical Help to Exporters Service. Various BSI electronic information services are also available which give details on all its products and services. Contact the Information Centre. Tel: 020 8996 7111. Fax: 020 8996 7048.

Subscribing members of BSI are kept up to date with standards developments and receive substantial discounts on the purchase price of standards. For details of these and other benefits contact Membership Administration. Tel: 020 8996 7002. Fax: 020 8996 7001. Further information about BSI is available on the BSI website at <http://www.bsi-global.com>.

### Copyright

Copyright subsists in all BSI publications. BSI also holds the copyright, in the UK, of the publications of the international standardization bodies. Except as permitted under the Copyright, Designs and Patents Act 1988 no extract may be reproduced, stored in a retrieval system or transmitted in any form or by any means – electronic, photocopying, recording or otherwise – without prior written permission from BSI.

This does not preclude the free use, in the course of implementing the standard, of necessary details such as symbols, and size, type or grade designations. If these details are to be used for any other purpose than implementation then the prior written permission of BSI must be obtained.

If permission is granted, the terms may include royalty payments or a licensing agreement. Details and advice can be obtained from the Copyright Manager. Tel: 020 8996 7070.