BS ISO/IEC 15962:2022

BSI Standards Publication

Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 15962:2022

National foreword

This British Standard is the UK implementation of ISO/IEC 15962:2022. It supersedes BS ISO/IEC 15962:2013, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/34, Automatic identification and data capture techniques.

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

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2023
Published by BSI Standards Limited 2023

ISBN 978 0 539 02168 4

ICS 35.040.50

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 April 2023.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 15962:2022

INTERNATIONAL STANDARD

ISO/IEC 15962

Third edition
2022-01

Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions

Technologies de l'information — Identification par radiofréquence (RFID) pour la gestion d'objets — Protocole de données: règles d'encodage des données et fonctions logiques de mémoire

Reference number
ISO/IEC 15962:2022(E)

Documents
Document Sharing Hub

© ISO/IEC 2022

Full document details: BS ISO/IEC 15962:2022

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2022

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 15962:2022

Contents

Page

Foreword ... **ix**

Introduction ... **x**

1 Scope ... **1**

2 Normative references ... **1**

3 Terms and definitions and abbreviated terms ... **1**
3.1 Terms and definitions ... 1
3.2 Abbreviated terms ... 2

4 Conformance ... **2**
4.1 Conformance with the air interface ... 2
4.2 Conformance with the application interface ... 2
4.2.1 Encoders and the application interface ... 2
4.2.2 Decoders and the application interface ... 3
4.2.3 Comprehensive encoder/decoder devices and the application interface ... 3
4.3 Conformance with the Access-Method ... 3
4.3.1 Encoders and the Access-Method ... 3
4.3.2 Decoders and the Access-Method ... 3
4.3.3 Comprehensive encoder/decoder devices and the Access-Method ... 3

5 Protocol model ... **3**
5.1 Overview ... 3
5.2 Layered protocol ... 4
5.2.1 Layers ... 4
5.2.2 Application layer as defined in the ISO/IEC 15961 series ... 4
5.2.3 Application interface as defined in ISO/IEC 15961-1 ... 5
5.2.4 Data Protocol processing ... 5
5.2.5 Data Protocol interface ... 5
5.3 Flexible implementation configurations ... 6
5.4 Functional processes — Interrogator implementation ... 6
5.4.1 Logical functions and interfaces ... 6
5.4.2 Functional processes — Application interface ... 7
5.4.3 Functional processes — Interrogator ... 7
5.4.4 RFID tag ... 8
5.5 ISO/IEC 15962 and the Data Processor ... 9

6 Data and presentation conventions ... **9**
6.1 Data types in ISO/IEC 15961-1 commands and responses ... 9
6.2 Extensible bit vector (EBV) ... 9
6.3 Object Identifier presentation in the application interface ... 10
6.3.1 Object identifier structure to ISO/IEC 8824-1 ... 10
6.3.2 Presenting the Object-Identifier in accordance with ISO/IEC 8824-1 ... 11
6.3.3 Presenting the Object-Identifier as a Uniform Resource Name (URN) ... 11
6.4 The Object ... 11
6.5 8-bit byte ... 11
6.6 N-bit encoding ... 11

7 Data Processor — High level processing ... **11**

8 Data Processor and the application interface ... **12**
8.1 Application commands — Overview ... 12
8.2 Application commands and responses — Write ... 14
8.2.1 Configure-AFI ... 14
8.2.2 Configure-DSFID ... 15
8.2.3 Write-Objects ... 15
8.2.4 Write-Objects-Segmented-Memory-Tag ... 19
8.2.5 Write-EPC-UII ... 22

AVA Documents
Document Sharing Hub

© ISO/IEC 2022 – All rights reserved

Full document details: BS ISO/IEC 15962:2022

8.2.6 Write-Password-Segmented-Memory-Tag 23
8.2.7 Write-Segments-6TypeD-Tag 24
8.2.8 Write-Monomorphic-UII 27
8.2.9 Configure-Extended-DSFID 30
8.2.10 Configure-Multiple-Records-Header 31
8.3 Application commands and responses — Read 33
8.3.1 Read-Object-Identifiers 33
8.3.2 Read-Logical-Memory-Map 34
8.3.3 Read-Objects 35
8.3.4 Inventory-ISO-UIImemory 36
8.3.5 Inventory-EPC-UIImemory 37
8.3.6 Read-Words-Segmented-Memory-Tag 38
8.3.7 Read-Segments-6TypeD-Tag 39
8.3.8 Read-Multiple-Records 40
8.4 Application commands and responses — Other 44
8.4.1 Inventory-Tags 44
8.4.2 Delete-Object 44
8.4.3 Modify-Object 46
8.4.4 Erase-Memory 48
8.4.5 Get-App-based-System-Info 49
8.4.6 Kill-Segmented-Memory-Tag 49
8.4.7 Delete-Packed-Object 50
8.4.8 Modify-Packed-Object-Structure 51
8.4.9 Delete-Multiple-Record 52
8.5 Air interface support for application commands 53

9 Data Processor and the air interface 53
9.1 Use 53
9.2 Air interface services 53
9.3 Defining the system information 54
9.3.1 System information elements 54
9.3.2 Singulation-Id 54
9.3.3 Physical block size 55
9.3.4 Number of blocks 55
9.3.5 AFI 55
9.3.6 DSFID 55
9.3.7 Encoding the Extended-Data-Format 56
9.3.8 Other extensions using the Extended Syntax indicator bit 56
9.3.9 Extended Syntax flag byte 1 56
9.3.10 Memory length indicator bits 56
9.3.11 Procedure for length encoding 58
9.3.12 Data CRC indicators 58
9.3.13 Data CRC 58
9.3.14 Extended Syntax flag byte 2 58
9.3.15 Simple Sensor indicator 59
9.3.16 Battery Assist indicator 59
9.3.17 Full-Function Sensor indicator 59
9.3.18 DSFID and Extended Syntax 59
9.4 Configuring the Logical Memory 63

10 Command/Response Unit — Processing of command and response arguments 63
10.1 Function 63
10.2 Process arguments 64
10.2.1 Access-Password 64
10.2.2 Additional-App-bits 64
10.2.3 AFI 64
10.2.4 AFI-Lock 64
10.2.5 Append-To-Existing-Multiple-Record 65
10.2.6 Application-Defined-Record-Capacity 65

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

10.2.7 Avoid-Duplicate ... 65
10.2.8 Check-Duplicate ... 65
10.2.9 Compact-Parameter ... 66
10.2.10 Data-Length-Of-Record ... 66
10.2.11 DSFID ... 67
10.2.12 DSFID-Lock ... 67
10.2.13 Directory-Length-EBV8-Indicator ... 67
10.2.14 Encoded-Memory-Capacity ... 67
10.2.15 EPC-Code ... 67
10.2.16 Hierarchical-Identifier-Arc ... 67
10.2.17 Identifier-Of-My-Parent ... 68
10.2.18 Identify-Method and Number-Of-Tags ... 68
10.2.19 Instance-Of-Arc ... 68
10.2.20 Item-Related-DSFID ... 69
10.2.21 Item-Related-Segment-Map ... 69
10.2.22 Kill-Password ... 69
10.2.23 Length-Of-Mask ... 69
10.2.24 Lock-Directory-Entry ... 69
10.2.25 Lock-Multiple-Records-Header ... 69
10.2.26 Lock-Record-Preamble ... 69
10.2.27 Lock-UII-Segment-Arguments ... 70
10.2.28 Max-App-Length ... 70
10.2.29 Memory-Bank ... 70
10.2.30 Memory-Bank-Lock ... 70
10.2.31 Memory-Segment ... 70
10.2.32 Memory-Type ... 70
10.2.33 Multiple-Records-Directory-Length ... 70
10.2.34 Multiple-Records-Features-Indicator ... 71
10.2.35 NSI-bits ... 71
10.2.36 Number-In-Data-Element-List ... 71

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

10.2.37 Number-Of-Records 71
10.2.38 Object-Lock 71
10.2.39 Packed-Object-Directory-Type 71
10.2.40 Password 72
10.2.41 Password-Type 72
10.2.42 Pointer 72
10.2.43 Pointer-To-Multiple-Records-Directory 72
10.2.44 Read-Record-Type 73
10.2.45 Read-Type 73
10.2.46 Record-Memory-Capacity 73
10.2.47 Record-Type-Arc 74
10.2.48 Record-Type-Classification 74
10.2.49 Sector-Identifier 74
10.2.50 Segment-Read-Type 74
10.2.51 Simple-Sensor-Data-Block 75
10.2.52 Start-Address-Of-Record 75
10.2.53 Tag-Data-Profile-ID-Table 75
10.2.54 Tag-Mask 75
10.2.55 TID-Segment-Map 75
10.2.56 UII-DSFID 75
10.2.57 UII-Segment-Map 75
10.2.58 Update-Multiple-Records-Directory 75
10.2.59 Word-Count 76
10.2.60 Word-Pointer 76
10.3 Completion-Codes 76
10.4 Execution-Codes 79

11 Access-Method 80
11.1 Methods 80
11.2 No-Directory structure 81
11.2.1 Structure 81
11.2.2 Restrictions to air interfaces 82
11.2.3 The dataset 82
11.2.4 Encoding rules 82
11.3 Directory structure 83

© ISO/IEC 2022 – All rights reserved
AVA Documents Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

11.3.1 Structure ... 83
11.3.2 Restrictions to air interfaces ... 83
11.3.3 Directory structure for Data-Format = "3 ...287" ... 84
11.3.4 Directory structure for Data-Format = 2 ... 84
11.3.5 Encoding the address of the dataset ... 84
11.3.6 Encoding example ... 84
11.4 Packed-Objects structure ... 84
11.5 Tag Data Profile ... 85
11.5.1 Use ... 85
11.5.2 Restrictions to air interfaces ... 86
11.5.3 Defining the Tag-Data-Profile ... 86
11.5.4 Encoding Rules ... 86
11.6 Multiple-Records ... 86
11.6.1 Structure ... 86
11.6.2 Categories of multiple records ... 87
11.6.3 Object-Identifier structure ... 89
11.6.4 Sector identifier ... 91
11.6.5 Restrictions to air interfaces ... 91
11.6.6 Encoding rules ... 91

12 ISO/IEC 15434 direct encoding and transmission method using Access-Method 0 and Data-Format 3 ... 91
12.1 Use ... 91
12.2 General rules for ISO/IEC 15434 direct encoding ... 92
12.3 Specific support for ISO 17364, ISO 17365, ISO 17366 and ISO 17367 ... 92

13 Monomorphic-UII encoding ... 92
13.1 Use ... 92
13.2 6-bit encoding ... 93
13.3 7-bit encoding ... 93
13.4 URN Code 40 encoding ... 94
13.5 8859-1 octet encoding ... 94
13.6 Application-defined 8-bit coding ... 94

Annex A (informative) **Air interface support for application commands ... 95**
Annex B (normative) **Pro forma description for the Tag Driver ... 101**
Annex C (normative) **ISO/IEC 18000 Series Tag Driver Descriptions ... 103**
Annex D (normative) **Encoding rules for No-Directory Access-Method ... 115**
Annex E (normative) **Basic data compaction schemes ... 128**
Annex F (normative) **ISO/IEC 646 characters supported by the compaction schemes ... 133**
Annex G (informative) **Encoding example for No-Directory structure ... 137**
Annex H (informative) **Encoding example for a directory structure ... 140**
Annex I (normative) **Packed-Objects structure ... 143**
Annex J (normative) **Packed Objects ID tables ... 165**
Annex K (normative) **Packed Objects Encoding tables ... 175**
Annex L (informative) **Encoding example for Packed Objects ... 180**
Annex M (informative) **Decoding Packed Objects ... 184**
Annex N (normative) **Tag Data Profile encoding ... 188**
Annex O (normative) **Tag Data Profile ID tables ... 193**
Annex P (informative) **Encoding example for Tag Data Profile ... 197**
Annex Q (normative) **Basic encoding rules for Multiple-Records Access-Method ... 201**
Annex R (normative) **Multiple-Records encoding rules for hierarchical records ... 218**

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Annex S (informative) **Encoding example for the Multiple-Records Access-Method** **225**
Annex T (normative) **ISO/IEC 15434 direct encoding and transmission** **237**
Annex U (informative) **ISO/IEC 15434 direct DI encoding and transmission for ISO 17364, ISO 17365, ISO 17366 and ISO 17367** **243**
Annex V (normative) **URN Code 40 encoding** **248**
Bibliography **251**

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

This third edition cancels and replaces the second edition (ISO/IEC 15962:2013), which has been technically revised.

The main changes are as follows:

— editorial changes were made;

— references have been updated.

A list of all parts in the ISO/IEC 15962 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Introduction

The technology of radio frequency identification (RFID) is based on non-contact electronic communication across an air interface. The structure of the bits stored on the memory of the RFID tag is invisible and accessible between the RFID tag and the interrogator only by the use of an air interface protocol, as specified in the appropriate part of the ISO/IEC 18000 series. The result of the transfer of data between an application and an interrogator in open systems requires data to be encoded in a consistent manner on any RFID tag that is part of that open system. This is not only to allow equipment to be interoperable, but in the special case of data carriers, for the data to be encoded on the RFID tag in one system's implementation for it to be read at a later time in a completely different and unknown system's implementation. The data bits stored on each RFID tag must be formatted in such a way as to be reliably read at the point of use if the RFID tag is to fulfil its basic objective. This reliability is achieved through the specification of a Data Protocol using the application-defined arguments defined in ISO/IEC 15961-1 and the data encoding rules of this document. Additionally, ISO/IEC 24791-1 specifies a software system infrastructure architecture that enables RFID system operations between business applications and RFID interrogators. Specific parts of the ISO/IEC 24791 series address data management requirements (ISO/IEC 24791-2) and device interface requirements (ISO/IEC 24791-5). These support defined implementations that incorporate the encoding rules of this document and the functional rules of the commands and responses in ISO/IEC 15961-1.

Manufacturers of RFID equipment (e.g. interrogators, RFID tags) and the users of RFID technology require a standards-based Data Protocol for RFID for item management. ISO/IEC 15961-1, ISO/IEC 15961-2, ISO/IEC 15961-3, this document and the ISO/IEC 24791 series specify this protocol, which is layered above the air interface standards defined in the ISO/IEC 18000 series.

The Data Protocol used to exchange information in an RFID system for item management is specified in ISO/IEC 15961-1 and in this document. Both documents are required for a complete understanding of the Data Protocol in its entirety, but each focuses on one particular interface:

— ISO/IEC 15961-1 addresses the interface with the application system.

— This document deals with the processing of data and its presentation to the RF tag, and the initial processing of data captured from the RF tag.

The transfer of data to and from an application, supported by appropriate application commands, is the subject of ISO/IEC 15961-1. This document specifies the overall process and the methodologies developed to format the application data into a structure to store on the RFID tag.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions

1 Scope

This document focuses on encoding the transfer syntax of the application commands defined in ISO/IEC 15961-1. The encodation is in a Logical Memory as a software analogue of the physical memory of the RFID tag being addressed by the interrogator.

This document

— defines the encoded structure of object identifiers;

— specifies the data compaction rules that apply to the encoded data;

— specifies a Precursor for encoding syntax features efficiently;

— specifies formatting rules for the data, e.g. depending on whether a directory is used or not;

— defines how application commands, e.g. to lock data, are transferred to the Tag Driver;

— specifies processes associated with sensory information and the transfers to the Tag Driver;

— defines other communication to the application.

2 Normative references

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

ISO/IEC 15961-1, *Information technology — Radio frequency identification (RFID) for item management — Data Protocol — Part 1: Application interface*

ISO/IEC 19762, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary*

3 Terms and definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19762 and the following apply. ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1.1
data compaction
mechanism, or algorithm, to process the original data so that it is represented efficiently in fewer bytes in a data carrier than in the original presentation

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

3.1.2
data processor
implementation of the processes defined in this document, including the Data Compactor, Formatter, Logical Memory, and Command/Response Unit

3.1.3
precursor
byte, sometimes a sequence of bytes, used in the Directory and No-Directory Access-Methods that acts as metadata for the subsequent Object-Identifier and Object

3.2 Abbreviated terms

FQIDV Fully-qualified ID value

IDLPO ID List Packed Object

IDMPO ID Map Packed Object

PO Packed Object

TBD To be determined

TEI Text element identifier

URN uniform resource name

Since "TDS", "Type A" to "Type D" are commonly used in the industry to refer to the technology components as specified by these standards, the following designations are used in this document:

Type A refers to ISO/IEC 18000-61

Type B refers to ISO/IEC 18000-62

Type C refers to ISO/IEC 18000-63

Type D refers to ISO/IEC 18000-64

TDS refers to GS1 EPC, *Tag Data Standard*

4 Conformance

4.1 Conformance with the air interface

A conformant implementation of this document shall support one or more air interface protocols through the Tag Drivers defined in Annex C. Declarations of conformance shall refer to the specific air interface protocol(s). This applies to encoders, decoders, or more comprehensive devices.

4.2 Conformance with the application interface

4.2.1 Encoders and the application interface

Within the constraints of the air interface protocol supported, a conformant implementation of this document on an encoder shall support the application commands defined in 8.2 and the associated process argument, as defined in Clause 10.

A conformant RFID tag shall have its encoding in a state that can be properly decoded by a conformant decoder (see 4.2.2).

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

4.2.2 Decoders and the application interface

Within the constraints of the air interface protocol supported, a conformant implementation of this document on a decoder shall support the application commands defined in 8.3 and the associated process argument, as defined in Clause 10.

4.2.3 Comprehensive encoder/decoder devices and the application interface

Within the constraints of the air interface protocol supported, a conformant implementation of this document on an encoder/decoder shall support the application commands defined in 8.2 and 8.3 and the associated process argument, as defined in Clause 10. In addition, the Delete-Object (see 8.4.2) and Modify-Object (see 8.4.3) commands shall be supported. Other commands defined in 8.4 may be supported, and each command that is supported shall be declared.

4.3 Conformance with the Access-Method

4.3.1 Encoders and the Access-Method

A conformant implementation of this document on an encoder shall support the encoding rules and formatting rules of one or more Access-Methods as defined in Clause 11 and associated Annexes. Declarations of conformance shall refer to the specific Access-Method(s) supported.

4.3.2 Decoders and the Access-Method

A conformant implementation of this document on a decoder shall support the decoding rules and formatting rules of all the Access-Methods as defined in Clause 11 and associated Annexes.

An interrogator is not expected to fully support the decoding functions of all the Access-Methods, and the following shall apply to achieve conformance:

— For full conformance, the decoder process on the interrogator shall output the Object-Identifier, Object and other arguments as required in the responses to the commands.

— For partial conformance, the decoder process on the interrogator shall output the byte string that represents the encoded package (depending on the Access-Method) containing the requested Object-Identifier. The encoded package then shall be fully decoded by a decoder process, external to the interrogator that is fully in conformity with the rules defined in this document.

Declarations of conformance shall refer to the specific Access-Method(s) supported.

4.3.3 Comprehensive encoder/decoder devices and the Access-Method

A conformant implementation of this document on an encoder/decoder shall support the encoding rules and formatting rules of one or more Access-Methods as defined in Clause 11 and associated Annexes. Declarations of conformance shall refer to the specific Access-Method(s) supported. The decoding function shall be as defined in 4.3.2.

5 Protocol model

5.1 Overview

RFID supports bit encodation in the RFID tag memory. Unlike other data carrier standards which require encodation schemes that are specific to the individual data carrier technology, the ISO/IEC 18000 series does not specify the interpretation of bits or bytes encoded on the RFID tag memory. However, as an RFID tag is a relay in a communication system, each tag used for open systems item management shall have data encoded in a consistent manner. The prime function of ISO/IEC 15961-1 is to specify a common interface between the application programs and the RFID interrogator. The prime function of this document is to specify the common encoding rules and Logical Memory functions

© ISO/IEC 2022 – All rights reserved

AVA Documents Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

RFID tags utilise electronic memory, which is typically capable of increasing data capacity as new generations of product are introduced. Differences in data capacity of each RFID tag type, whether similar or dissimilar, are recognised by the Data Protocol defined in these two documents.

Different application standards may have their own datasets or data dictionaries. Each major application standard for item management shall have its data treated in an unambiguous manner, avoiding confusion with data from other applications and even with data from closed systems. The Data Protocol specified in these documents ensures the unambiguous identification of data.

5.2 Layered protocol

5.2.1 Layers

The protocol layers of an implementation of RFID for item management are illustrated schematically in Figure 1.

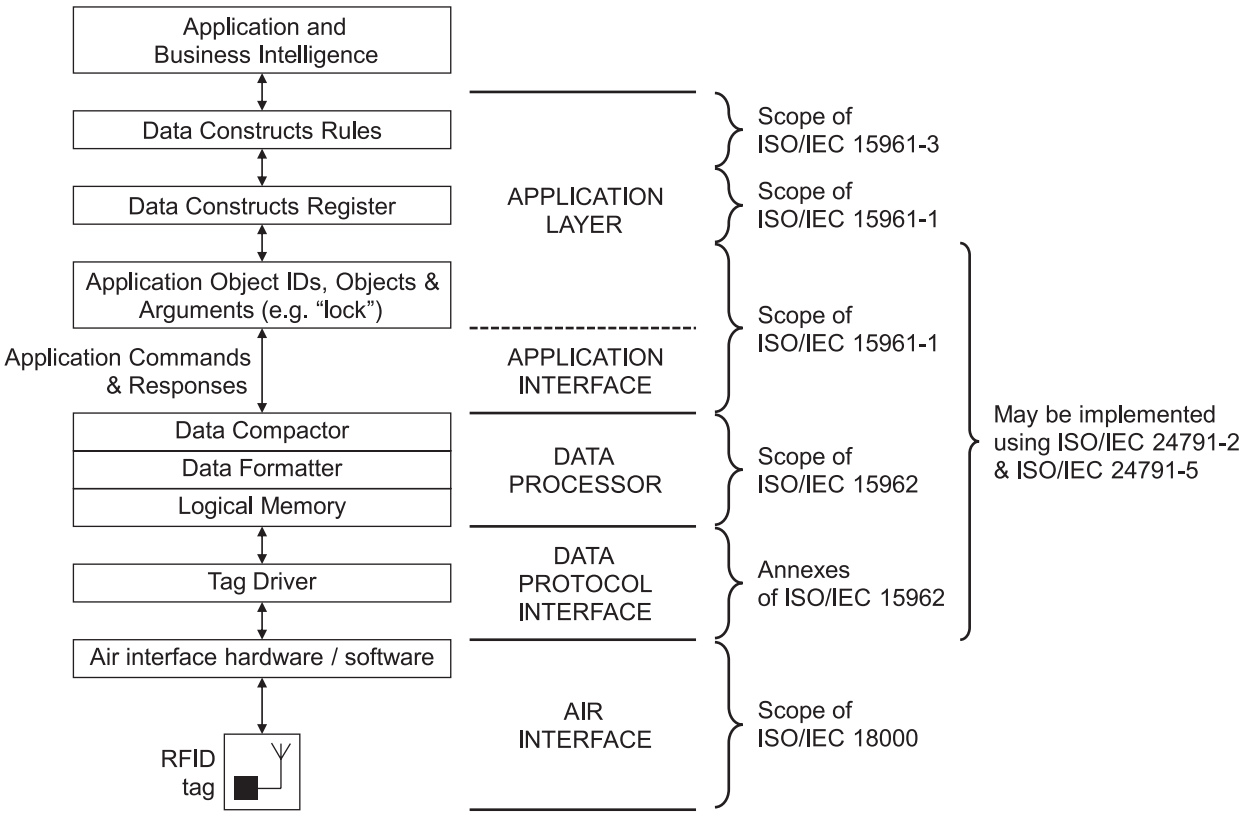


Figure 1 — Schematic of protocol layers for an implementation of RFID for item management

5.2.2 Application layer as defined in the ISO/IEC 15961 series

The RFID Data Protocol specifies how data is presented as objects, each uniquely identified with an object identifier, which are meaningful to the application and can be encoded on the RFID tag. ISO/IEC 15961-3 specifies the data construct rules for the AFI, DSFID, object identifier for the unique item identifier, and object identifier structure for other item-related data. This ensures that each piece of data can be uniquely identified, both within the scope of a particular application and between applications.

Each application shall be registered according to ISO/IEC 15961-2 so that the data constructs can be declared and used in an unambiguous manner.

The RFID Data Protocol in ISO/IEC 15961-1 defines functions and arguments used to construct application commands and responses. This is so that application programs can specify what data to

© ISO/IEC 2022 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

transfer to and from the RFID tag and to append, update, selectively lock, delete data, or perform other functions on the RFID tag.

NOTE To illustrate how the functions and arguments are assembled into a structured format, several commands and responses have been constructed using an abstract syntax. This is independent of the host application, operating system, and programming language and independent of the specific command structures between the interrogator and Tag Driver. The abstract syntax used in ISO/IEC 15961-1 is like that used in ISO/IEC 24791-5, and is intended to enable closer integration with that standard. The original version of ISO/IEC 15961:2004[1)] included commands defined using ASN.1 abstract syntax, see ISO/IEC 8825-1. For backward compatibility, the commands that were originally defined in this manner have been included in an annex of ISO/IEC 15961-1.

This RFID Data Protocol also defines arguments and codes to support responses of data that is read from an RFID tag, including error messages, which are returned to the application.

The abstract syntax may be used as a basis to prepare commands in different programme languages, supporting the functionality and arguments of the abstract commands.

5.2.3 Application interface as defined in ISO/IEC 15961-1

The application interface may be implemented in several different ways that are not explicitly defined in this document, nor in ISO/IEC 15961-1. The basic requirement is to identify data objects distinctly from all others using object identifiers, even to enable different data formats to be intermixed on the same RFID tag. The application interface also shall define command and response arguments unambiguously, so that they can be intermixed with data on the same wired or wireless network.

One major class of implementation, described as a *straight-through process*, is appropriate where the functions and arguments used to construct commands and the arguments and codes used to construct responses, as specified in this document, are directly input to the encoding processes of ISO/IEC 15962. Such input can be from computer screens or forms, or more direct transfers from host systems. The advantage of this process is that it avoids the creation of the transfer encoding (see below) but requires more rigorous adherence to the functional requirements of the commands and responses. ISO/IEC 15961-1 imposes no constraints on the application interface process to be adopted, other than the requirement that it be integrated with the encoding rules of ISO/IEC 15962.

Whichever approach is used, the encoding rules of ISO/IEC 15962 shall be followed, and the encoding on the RFID tag shall be in conformity with all the arguments in the commands specified in ISO/IEC 15961-1.

5.2.4 Data Protocol processing

The RFID Data Protocol specifies how data is encoded, compacted, and formatted on the RFID tag and how this data is retrieved from the RFID tag to be meaningful to the application.

This RFID Data Protocol provides for a set of schemes that compact the data to make more efficient use of the memory space.

This RFID Data Protocol also supports various storage formats to enable efficient use of memory and efficient access procedures.

5.2.5 Data Protocol interface

Each air interface protocol standard in the ISO/IEC 18000 series has its own specific rules for defining commands and responses. Furthermore, some air interface protocols can support different tag architectures with different memory sizes, and possibly support optional commands. The Data Protocol provides a mechanism to interface with these rules through specific Tag Drivers. These allow the basic application commands and responses of ISO/IEC 15961-1 to be applied independently of the air interface protocol and specific tag architecture.

---

1) Withdrawn standard. Revised by ISO/IEC 15961-1:2013, ISO/IEC 15961-2:2019, ISO/IEC 15961-3:2019, ISO/IEC 15961-4:2016.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

The Tag Driver component of the Data Protocol provides the mapping rule from the generic processes to the specific tag requirements. These mapping rules are used to write data and to read data.

NOTE Additional Tag Drivers can be specified as new air interface protocols are introduced in the ISO/IEC 18000 series of standards.

5.3 Flexible implementation configurations

This RFID Data Protocol specifies the application level communication and the RFID tag interrogator level rules for data encoding, compaction, and storage formats. This protocol may be implemented:

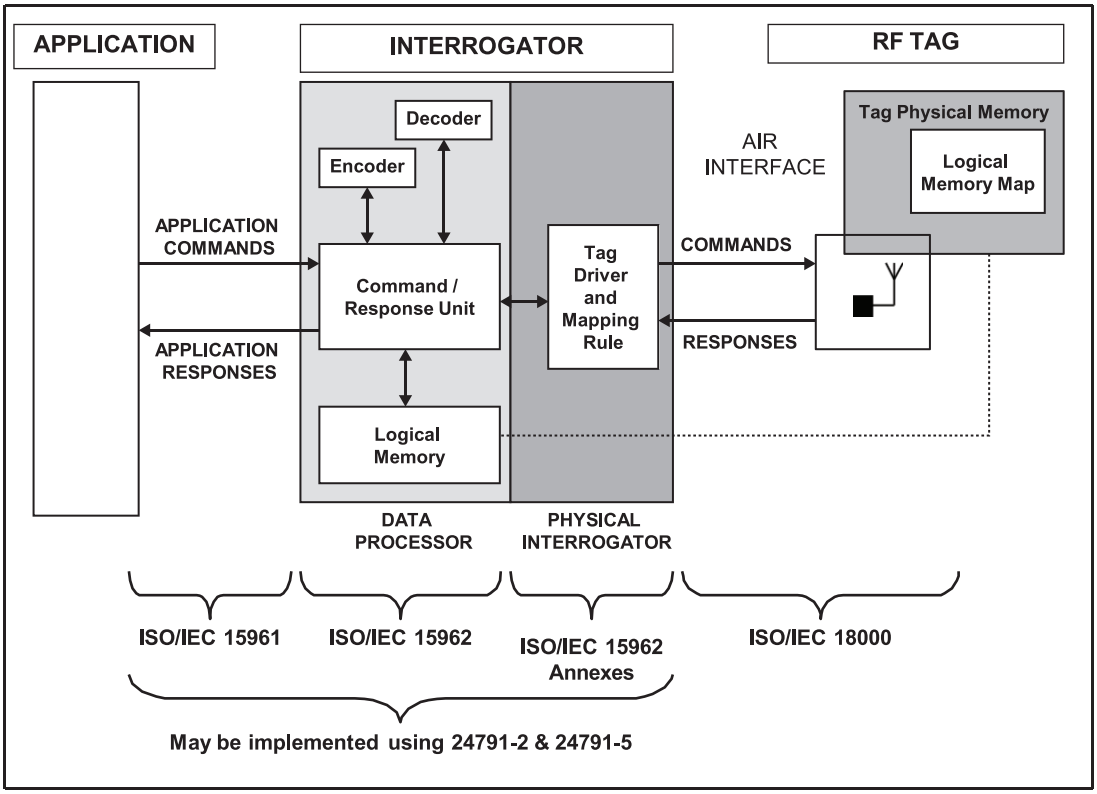
— with the document incorporated into the software system infrastructure architecture defined in the ISO/IEC 24791 series. This is the recommended approach for any networked application.

— with ISO/IEC 15961-1 and this document incorporated into stand-alone software or devices that have as its output conformant encoding and/or conformant decoding with responses in conformity with the responses of ISO/IEC 15961-1.

5.4 Functional processes — Interrogator implementation

5.4.1 Logical functions and interfaces

There are various functional processes that need to take place to write data to an RFID tag and to read data from it. Figure 2 shows a schematic of an example implementation where the processing of the Data Protocol resides in the interrogator. This illustration is provided to help with the understanding of the processes, and although a typical implementation, many others are possibly in conformity with this Data Protocol.



NOTE The Logical Memory Map in the Tag Physical Memory is given by the Tag architecture and the mapping rules in the Tag Driver. All the information in the Logical Memory is represented in the Logical Memory Map.

Figure 2 — Logical functions and interfaces of ISO/IEC 15962 with other RFID system components

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

5.4.2 Functional processes — Application interface

The data flows between the application and the Data Processor are formatted according to this document and are not compacted. However, there are numerous established systems where data is formatted to be in conformance, for example, with a bar code related syntax. It is therefore reasonable to insert interface modules in the data flow to convert from and to existing application formats.

Careful consideration should be given to the extent that established systems need to be supported relative to the potential benefits to be gained from adopting the Data Protocol specified in this document and ISO/IEC 15961-1. This is because this protocol has been developed around the features of RFID, such as selective read/write and the ability to lock data. Older protocols are unlikely to support such features.

5.4.3 Functional processes — Interrogator

5.4.3.1 General

In the process illustrated in Figure 2, the interrogator is the module in which all the basic processing of the Data Protocol takes place and there is an interface to the RFID tag. Different implementations can separate some of the functions described below and have an interface between the application and the physical interrogator.

5.4.3.2 Data processor

5.4.3.2.1 General

The Data Processor provides all the processing, which is as specified in this document and is required for handling application data. It consists of the following components, all of which are described more fully below: Command/Response Unit, Logical Memory, Encoder (which supports a Data Compactor and Formatter function) and Decoder (which supports the inverse functions of the Encoder).

The Data Processor can physically reside anywhere between the application software and the Tag Driver but shall contain all the components. Some, or all, of the Data Processor functions may be implemented using processes defined in ISO/IEC 24791-2 and ISO/IEC 24791-5.

5.4.3.2.2 Command/Response Unit

The Command/Response Unit receives the application commands from the application in a format specified in this document, acting upon these commands where appropriate and converting to the specific RFID tag lower-level command codes.

EXAMPLE An application command of *write Data Object "name"* is application related. The Data Protocol recognises this and can format the data onto the Logical Memory in the Data Processor. Information from the particular RFID tag is required to set the parameters of the Logical Memory Map (e.g. number of bytes, whether a directory is in use) on the RFID Tag. The Tag Driver converts the application command into a tag-specific command.

It can be seen from this example that there is a distinct logical boundary between the Data Processor and the Tag Driver.

5.4.3.2.3 Logical Memory

This is an array of contiguous bytes of memory acting as a common software representation of the RFID tag memory accessible by an application and to which the object identifiers and data objects are mapped in bytes. The Logical Memory considers some parameters of the real RFID tag, for example the block size, the number of blocks and the storage format. The Logical Memory ignores any detailed tag architecture.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

The use of the Logical Memory means that an application can interface with an application-conformant RFID tag, but that individual RFID tags can have completely different memory capacities and architectures. This enables an implementation to benefit from new technological developments permitted within the framework of the ISO/IEC 18000 series, such as larger capacity or faster access RFID tags, without changing the application.

5.4.3.2.4 Encoder

The Encoder controls the process of writing data through the functional processes performed by the Data Compactor Module and Formatter Module.

5.4.3.2.5 Data Compactor

The Data Compactor provides the standard compaction rules to reduce the number of bytes stored on the RFID tag and transferred across the air interface. Numeric data, for example, is byte based to some coded character set for the application but can be encoded in a compact form on the RFID tag memory.

5.4.3.2.6 Formatter

The Formatter provides the processes to place the object identifier and object (data) into an appropriate and efficient format for storing on the Logical Memory.

NOTE The physical mapping of bits to conform with the RFID tag architecture is performed based on information provided by the Tag Driver.

5.4.3.2.7 Decoder

The Decoder controls the process of reading and interpreting data through the functional processes performed by the Data de-compactor Module and de-formatter Module.

5.4.3.3 Tag Driver

The Tag Driver provides two main functions:

— It provides mapping rules on the structure of the RFID tag to enable the contents of the Logical Memory in the Data Processor to be exchanged with the Logical Memory Map of the RFID tag in use.

— It provides facilities that accept the application commands of this Data Protocol and converts them to a format that results in calls to command codes supported by the particular RFID Tag. For example, an application command write Data Object "name" could result in the RFID tag related command of write (block #, data).

The description of the Tag Driver for RFID tags is provided in the annexes of this document, specifically Annexes B and C. For this document, a Tag Driver is unique to a particular air interface type of RFID tag as specified in the appropriate part of the ISO/IEC 18000 series. This is a logical representation; physical implementation could combine features of different logical Tag Drivers. An interrogator may support one or many Tag Drivers.

5.4.3.4 Transfer mechanisms

The transfer mechanisms for transferring data, commands, and responses between the Data Processor (i.e. the implementation of this document) and the RFID tag shall be done through the interrogator or incorporated within the functions of the interrogator.

5.4.4 RFID tag

Although the RFID tag is beyond the scope of this document, the tag is shown in Figure 2 to complete the flow of data and commands. Within the RFID tag, the Logical Memory Map represents all the data in

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

the Logical Memory of the Data Processor converted (or mapped) to a location structure determined by the mapping rules in the Tag Driver and the architecture of the RFID tag.

5.5 ISO/IEC 15962 and the Data Processor

This document defines all the rules for encoding data on an RFID tag. The implementation of these rules is described in this document as the Data Processor. As described in 5.4, various processes are undertaken to achieve successful encoding. The rules ensure the encoded bytes are "self-declaring" in a reading operation without any previous or independent knowledge of what is encoded on the tag. This allows the Data Protocol to be used in open systems where the organisation encoding the data on the RFID tag can be completely unaware of what organisation can eventually read the data from the RFID tag. This is achieved by:

— Having clearly defined rules for compacting data that are applied independently of the application and the type of RFID tag.

— Defining fundamental structuring rules for the encoding of data largely determined by a user-selectable feature called the Access-Method.

— Having precise structuring rules within each Access-Method, including a clearly defined syntax that allows reading systems to selectively identify data required for the application.

Clause 11 provides a description and a definition of the Access-Methods currently supported. The associated processing is fully specified in annexes referred to in Clause 11.

6 Data and presentation conventions

6.1 Data types in ISO/IEC 15961-1 commands and responses

The following data types are used in the commands and responses:

— BOOLEAN: An argument that can have the values TRUE or FALSE.

— BIT STRING: A sequence of bits.

— BYTE: An integer with the possible values 0 to 255, usually expressed as a hexadecimal value 00h to FFh.

— BYTE STRING: A sequence of bytes (equivalent to OCTET STRING).

— HEXADECIMAL ADDRESS: A location (on the memory of an RFID tag), expressed as a hexadecimal value

— INTEGER: An integer can take any whole number. In the context of this document, the values are all positive.

— OBJECT IDENTIFIER: An Object Identifier as defined in 6.3.1.

6.2 Extensible bit vector (EBV)

The EBV block structure is used as a self-declaring code to determine the length, position, or size of some characteristic. Various EBV-n structures are used in this document, where n refers to the number of bits in each block. EBV-8 is identical to the length encoding (see 9.3.11), which has been used since the original edition of this document.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

6.3 Object Identifier presentation in the application interface

6.3.1 Object identifier structure to ISO/IEC 8824-1

This document uses the OBJECT IDENTIFIER type as defined in ISO/IEC 8824-1 and with identifiers assigned as specified in ISO/IEC 9834-1. This uses a registration tree with a common implied root node (ISO/IEC 9834-1), a series of arcs from each node, with new arcs added as required to define a particular object. Thus, the body responsible for a particular node (see Figure 3):

— has a defined set of arcs to identify itself;

— can manage the allocation of arcs under its node, independently of other bodies;

— is assured of uniqueness from all other arcs in the registration tree.

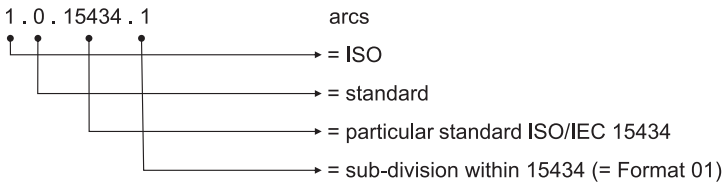


Figure 3 — Example of arcs

The only top arcs permitted for all object identifiers are shown in Table 1.

Table 1 — Object Identifier top arcs

Identifier arc name	Numeric value	
itu-t	0	
iso	1	
joint-iso-itu-t	2	

The second arc is administered by the relevant organisation named for the top arc. The current list of top and second arcs is given in ISO/IEC 15961-3.

The third arc is controlled by the system or body defined for the second arc; sometimes this is a registration authority. The hierarchical structure continues until the object is identified uniquely. The procedure of naming object identifiers ensures that each object is unique within its "parent" arc and that each parent arc is unique within its previous level, right back to the top 3 arcs.

NOTE This structure enables object identifiers from different domains (e.g. open and closed systems) to be encoded unambiguously on an RFID tag memory.

Three forms of object identifier are used with the RFID Data Protocol:

— Object-Identifier: This full structure is used for communications between the application and the Data Processor defined by the scope of this document. The full structure is suitable for use in this document where the set of object identifiers to be encoded on the RFID tag have different higher-level arcs, or where a generic system is in place.

— Relative-OID: This structure is used in conjunction with the Root-OID (see below) for communication between the application and the Data Processor. These structures are applied in situations where a common root applies to the set of object identifiers to be encoded on the RFID tag. For example, if all the object identifiers have the common root 1 0 15961 12 1, encoding space can be saved on the RFID tag if this common Root-OID does not have to be encoded for each object identifier. The Relative-OID is a suffix to a common Root-OID, which is either encoded or declared in some other way.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

— Root-OID: The Root-OID is the common part of a set of encoded object identifiers. It acts as a common prefix to the Relative-OID values encoded on the RFID tag. This structure is particularly important in applications that require a variety of data from a common data dictionary to be encoded on an RFID tag. The Root-OID is either explicitly encoded or declared in some other way, according to the encoding rules of this document.

6.3.2 Presenting the Object-Identifier in accordance with ISO/IEC 8824-1

When the Object-Identifier is presented in application commands and responses in accordance with ISO/IEC 8824-1, spaces are inserted between each arc as follows:

1 0 15961 12 1

6.3.3 Presenting the Object-Identifier as a Uniform Resource Name (URN)

The Object-Identifier may also be presented in the URN in the following format, based on IETF RFC 3061, with the decimal point character between each arc:

urn:oid:1.0.15961.12.1

6.4 The Object

On the interface with the application, all data that is intended to be encoded on the RFID is described as the Object in this document. The Object shall be qualified using an Object-Identifier to ensure unambiguous communication.

When the Object-Identifier is that of a unique item identifier then its combination with the Object confers domain uniqueness. Other Object-Identifier plus Object pairs (e.g. to identify a particular batch number) are not unique but represent attribute data that is associated with the unique item identifier.

6.5 8-bit byte

This document supports the encoding of data and data objects aligned as 8-bit bytes (sometimes referred to as octets to be more compatible with communications standards). Within each 8-bit byte (hereafter "byte"), the most significant bit is bit 8 and the least significant is bit 1. Accordingly, the weight allocated to each bit is defined in Table 2.

Table 2 — Bit sequence in the 8-bit byte

Bit value	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1
Weight	128	64	32	16	8	4	2	1

NOTE Although other bit position conventions are possible and more common, this bit sequence is used here for backward compatibility with the original version of this document, which in turn was in conformity with ASN.1 rules (ISO/IEC 8825-1).

6.6 N-bit encoding

This document supports the encoding of bit fields that are not necessarily aligned on a byte boundary. The most significant bit is bit N and the least significant bit is bit 1. When a sequence of bit fields is defined, the first bit field occupies the most significant bits.

7 Data Processor — High level processing

The main flows of information to and from the Command/Response Unit are illustrated in Figure 4, which focuses on the encoding process.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

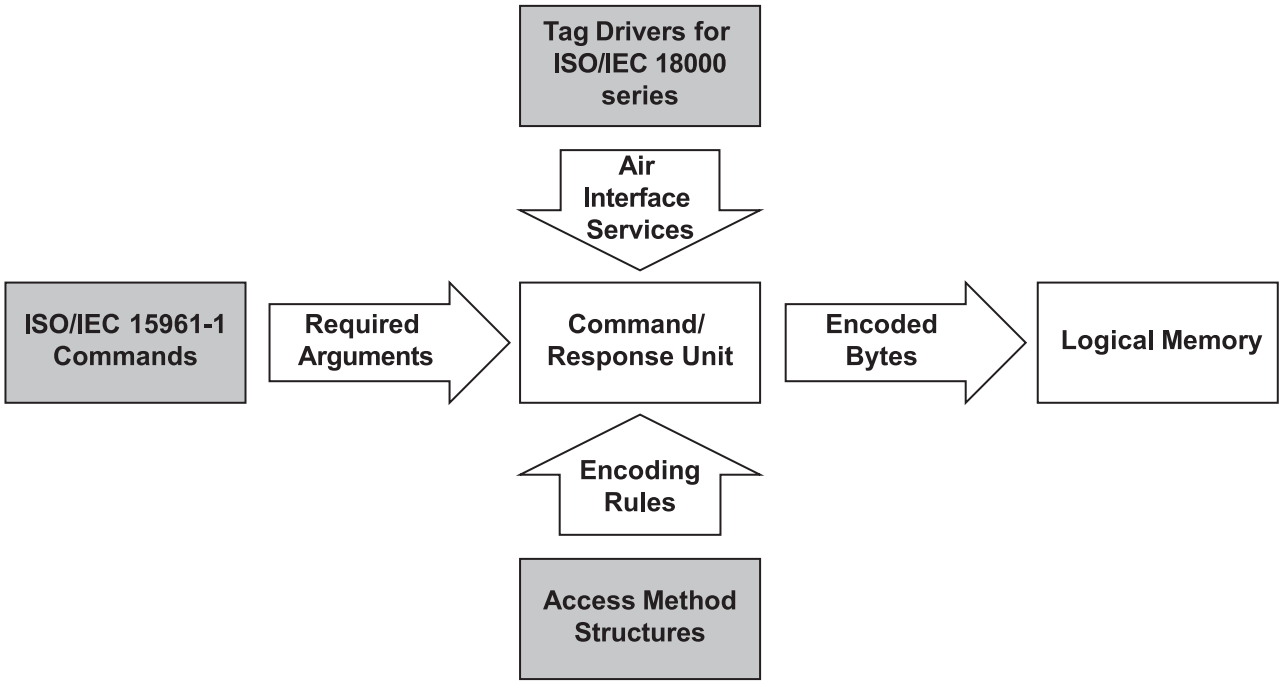


Figure 4 — High level information flows

The Data Processor is not required to support all application commands. For the commands that are supported, all the set of incorporated arguments shall be processed correctly to achieve the encoding required by the application. These requirements are defined in Clause 8. The method of transferring the commands to the command response unit is not specified in this document.

Applications should define that the data is to be encoded on a particular type of RFID tag. The Tag Driver for the particular tag provides the air interface services as defined in Clause 9. These are used by the Command/Response Unit to correctly encode data on, and decode data from, the type of tag required by the application. The Data Processor is not required to support all tag types but shall support the encoding and decoding capabilities of any type of tag that is supported. As some types of RFID tag that are defined in the ISO/IEC 18000 series of air interface protocols have optional components, the precise capability of the Data Processor shall be clearly defined against optional components and revisions to the ISO/IEC 18000 series.

8 Data Processor and the application interface

8.1 Application commands — Overview

ISO/IEC 15961-1 defines several commands and associated responses to provide instructions of how data is to be encoded on the RFID tag. Table 3 lists these commands. The column headed "Code" identifies the final arc of the object identifier that identifies the command module and response module. These code values have been retained to provide a structure that is compatible with the previous, withdrawn edition (ISO/IEC 15961:2004).

EXAMPLE The inventory-tag command has the object identifier: 1 0 15961 126 3. Its response has the object identifier: 1 0 15961 127 3.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Table 3 — ISO/IEC 15961-1 commands

Code	Command name	Write	Read	Other
1	Configure-AFI	[a]		
2	Configure-DSFID	[a]		
3	Inventory-Tags			YES
5	Delete-Object			YES
6	Modify-Object			YES
8	ReadObject-Identifiers		YES	
10	Read-Logical-Memory-Map		YES	
12	Erase-Memory			[b]
13	Get-App-based-System-Info			[c]
17	Write-Objects	YES		
18	Read-Objects		YES	
19	Write-Objects-Segmented-Memory-Tag	YES		
20	Write EPC-UII	YES		
21	Inventory-ISO-UIImemory		YES	
22	Inventory-EPC-UIImemory		YES	
23	Write-Password-Segmented-Memory-Tag	YES		
24	Read-Words-Segmented-Memory-Tag		YES	
25	Kill-Segmented-Memory-Tag			YES
26	Delete-Packed-Object			YES
27	Modify-Packed-Object			YES
28	Write-Segments-6TypeD-Tag	YES		
29	Read-Segments-6TypeD-Tag		YES	
30	Write-Monomorphic-UII	YES		
31	Configure-Extended-DSFID	YES		
32	Configure-Multiple-Records-Header	YES		
33	Read-Multiple-Records		YES	
34	Delete-Multiple-Record			YES

[a] The Configure-AFI and Configure-DSFID commands are implemented in different ways, depending on the air interface protocol. Some support specific air interface commands that are distinctly different from the generic write commands (e.g. ISO/IEC 18000-3 Mode 1). Others use the write command but require the data to be written into a specific location (e.g. Type B and Type C).

[b] Some air interface protocols support a specific Erase-Memory command, whereas others require the data to be overwritten with a null-byte value (typically 0016).

[c] The Get-App-based-System-Info command is one that is explicitly supported in some air interface protocols but in others is completely redundant because the information is returned as part of another command.

Table 3 has three columns: write, read, other. The write column shows the commands that are most likely to be implemented in a device such as a printer-encoder, and in an application that simply provides source marking for data capture at a later stage and at a different point. The read column identifies commands that need to be implemented in the simplest of data capture devices. The commands highlighted in the other column either require more complex processing (e.g. Modify-Object) or require a special process (e.g. Kill- Segmented-Memory-Tag).

The commands associated with code values not included in Table 3 were originally specified in ISO/IEC 15961:2004. This document no longer shall support these commands, mainly because their functionality has been absorbed in other commands.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

To implement the encoding rules of this document it is necessary to understand the detail of the commands and responses in ISO/IEC 15961-1. The subclauses that follow, provide a listing of commands under the three main headings of: read, write, and other. The arguments for each command are listed in the subclauses.

8.2 Application commands and responses — Write

8.2.1 Configure-AFI

8.2.1.1 Use

The AFI is a single byte code and is used as part of the selection process in an application. Depending on the air interface protocol, it may write into a particular location of memory using an explicit air interface protocol command. This application command is intended to support this process. It may also be used to support the encoding of the AFI using a generic air interface write command.

8.2.1.2 Process requirements

To correctly implement this command in the Data Processor, in addition to processing the AFI in the range 90_{16} to CF_{16}, the following process argument shall be supported:

AFI (see 10.2.3)

AFI-Lock (see 10.2.4)

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

The Completion-Codes for this command are:

0 No-Error

1 AFI-Not-Configured

2 AFI-Not-Configured-Locked

3 AFI-Configured-Lock-Failed

8 Singulation-Id-Not-Found

255 Execution-Error

8.2.1.3 Conformance requirements

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— That the RFID tag(s) supported by the encoder have the AFI in a memory location requiring an explicit air interface command, or that a generic air interface write command supports writing the AFI to a specific location determined by the Tag Driver with a single byte write transaction.

— If the encoder supports writing the AFI, then it shall also support the transfer of any AFI-Lock argument so that it can be implemented in the air interface.

— The encoder shall support the associated Completion-Codes and Execution-Codes.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

8.2.2 Configure-DSFID

8.2.2.1 Use

The DSFID is a single byte or multiple byte code that is used to reduce the encoding of Object-Identifiers, based on encoding rules in this document. In particular, the DSFID specifies the Access-Method and the Data-Format assigned to particular applications. Depending on the air interface protocol, the DSFID is written into a particular location of memory using an explicit air interface protocol command. This application command is intended to support this process. It may also be used to support the encoding of the DSFID using a generic air interface write command.

8.2.2.2 Process requirements

To correctly implement this command in the Data Processor, in addition to processing the DSFID, the following process argument shall be supported:

DSFID (see 10.2.11)

DSFID-Lock (see 10.2.12)

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

The Completion-Codes for this command are:

0 No-Error

4 DSFID-Not-Configured

5 DSFID-Not-Configured-Locked

6 DSFID-Configured-Lock-Failed

8 Singulation-Id-Not-Found

255 Execution-Error

8.2.2.3 Conformance requirements

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— That the RFID tag(s) supported by the encoder have the DSFID in a memory location requiring an explicit air interface command, or that a generic air interface write command supports writing the DSFID to a specific location determined by the Tag Driver with a single byte write transaction.

— If the encoder supports writing the DSFID, then it shall also support the transfer of any DSFID-Lock argument so that it can be implemented in the air interface.

— The encoder shall support the associated Completion-Codes and Execution-Codes.

8.2.3 Write-Objects

8.2.3.1 Use

This command shall not be used to write a Monomorphic-UII. If the AFI on the RFID tag declares that it is registered for a Monomorphic-UII, the appropriate error shall be returned, and the encoding process aborted. The correct command to use is defined in 8.2.8.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

The Write-Objects command is used to write one or more Object-Identifiers and associated Objects to an RFID tag. This command may be implemented to write the initial data to the RFID tag, or to add data to the tag. The command is supported by the following compound arguments, for which individual arguments are listed below:

Add-Objects-List

DSFID-Constructs

Ext-DSFID-Constructs

Multiple-Records-Constructs

Packed-Object-Constructs

8.2.3.2 Process requirements

To correctly implement this command in the Data Processor, the process arguments and compound arguments defined for the command in ISO/IEC 15961-1 shall be supported. For Multiple Records they are applied once per record, otherwise they are applied once per Logical Memory.

Data-CRC-Indicator (see 9.3.12)

DSFID (see 10.2.11)

DSFID-Lock (see 10.2.12)

DSFID-Pad-bytes (see 9.3.18)

Length-Of-Encoded-Data (see 9.3.11)

Memory-Capacity (see 9.3.11)

Tag-Data-Profile-ID-Table (see 10.2.53)

The main determinant for conformant encoding is the Access-Method, which is incorporated as part of the DSFID. The DSFID is provided as an optional argument for use in one of the following ways:

— If data is being written to a blank RFID tag, then the DSFID is provided as part of this command to minimise communications.

— If data is being added to the RFID tag, then the DSFID in the command should match the DSFID already encoded on the RFID tag, otherwise there is an error and the encoding process can cease before significant amounts of data have been processed.

To correctly implement this command in the Data Processor, the following process arguments shall be supported for each Object-Identifier and Object pair, irrespective of the Access-Method:

Avoid-Duplicate (see 10.2.7)

Compact-Parameter (see 10.2.9)

Object-Lock (see 10.2.38)

The Avoid-Duplicate, Compact-Parameter and Object-Lock arguments shall be applied in a manner consistent with the rules of the Access-Method (see Clause 11) to ensure reliable encoding. If there is an inconsistency within the application command, then it shall be in error and the encoding process shall not be implemented.

If the Access-Method = Packed-Objects, the input states are all defined in the Packed-Objects-Constructs argument specified in ISO/IEC 15961-1 and apply to a single Packed Object. That argument in turn has the following arguments, for which the processes are defined in Annex I:

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Block-Align-Packed-Objects

Editable-Pointer-Size

ID-Type

Object-Offsets-Multiplier

Packed-Object-Directory-Type

PO-Directory-Size

PO-ID-Table

PO-Index-Length

If the Access-Method = Tag-Data-Profiles, the input states are all defined by the Tag-Data-Profile-ID-Table and encoded as a single Tag Data Profile using the processes defined in Annex N.

The following tag-related arguments, if declared by the Extended DSFID are used as follows to complete the encoding process:

— If the memory Memory-Capacity (see 9.3.11) is encoded, or the air interface has its own means of providing this information, it can be used to assess if there is sufficient memory for the command to be fully implemented.

— If one or both Data-CRC-Indicators (see 9.3.12) is set, then the CRC-16 shall be applied as defined in 9.3.13.

— If the Length-Of-Encoded-Data (see 9.3.11) is signalled to be encoded, then this shall be calculated after all encoded bytes have been determined. This field is then written, or overwritten, taking into account the availability of DSFID-Pad-bytes (see 9.3.18).

If the compound argument Multiple-Records-Constructs is used in the command, then the Object-Identifier structure as defined in 11.6 shall be used for identifying each data Object in the add-Objects-List argument. If this is not the case, then the command shall not be processed. In addition to writing the Multiple-Record itself, this command can require other areas of the Logical Memory to be encoded: The MR-header if the number of records is maintained and the directory if already encoded or called for by the command arguments. If any part of the memory required to implement the command is locked, then the command shall be aborted.

Encoding shall be as defined in Annex Q if the first four arcs are 1.0.15961.401, or as defined in Annex R if the first four arcs are 1.0.15961.402 or 1.0.15961.403. The compound argument in turn has the following arguments, for which the processes are defined in Annex Q and Annex R.

Append-To-Existing-Multiple-Record (see 10.2.5)

Application-Defined-Record-Capacity (see 10.2.6)

Identifier-Of-My-Parent (see 10.2.17)

Lock-Directory-Entry (see 10.2.24)

Lock-Record-Preamble (see 10.2.26)

Number-In-Data-Element-List (see 10.2.36)

Record-Memory-Capacity (see 10.2.46)

Record-Type-Classification (see 10.2.48)

Update-Multiple-Records-Directory (see 10.2.58)

The following arguments shall be supported for the response:

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

Object-Identifier (see 6.3)

The response arguments Completion-Code and Object-Identifier are associated with each data element.

The Completion-Codes for this command will vary based on the associated compound arguments. The complete list is:

0 No-Error

8 Singulation-Id-Not-Found

9 Object-Not-Added

10 Duplicate-Object

11 Object-Added-But-Not-Locked

29 Object-Not-Editable

31 Packed-Object-ID-Table-Not-Recognised-No-Encoding

32 Tag-Data-Profile-ID-Table-Not-Recognised

33 Insufficient-Tag-Memory

36 Command-Cannot-Process-Monomorphic-UII

37 Data-CRC-Not-Applied

38 Length-Not-Encoded-In-DSFID 255 Execution-Error

43 Data-Format-Not-Compatible-Multiple-Records-Header

44 Access-Method-Not-Compatible-Multiple-Records-Header

45 Sector-Identifier-Not-Compatible-Multiple-Records-Header

46 Record-Preamble-Not-Configured

47 Record-Preamble-Not-Locked

255 Execution-Error

8.2.3.3 Conformance requirements

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— The encoder supports all aspects of the Access-Method.

— The encoder supports all the processes defined by the Extended DSFID.

— The encoder supports the associated Completion-Codes and Execution-Codes.

— If any of the first two conditions are not supported, an error is signalled by returning the Execution-Code = 4 Command-Not-Supported.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

8.2.3.4 Guidance for appending data

To maximize reliability when adding additional data or modifying data, it is recommended that the interrogator performs the following procedure in the order presented below:

a) write all the new bytes except the first word or block of data (i.e. the data that will overwrite the existing 00_{16} terminator), including the new terminator at the end of the new data;

b) optionally read all the new encoded data to ensure that it was stored correctly;

c) write the first word/block, which overwrites the original terminator.

NOTE Writing the new data in this sequence preserves the ability to parse the pre-existing data in an event of a failure to write of the new data.

8.2.4 Write-Objects-Segmented-Memory-Tag

8.2.4.1 Use

This command shall not be used to write a Monomorphic-UII. If the AFI on the RFID tag declares that it is registered for a Monomorphic-UII, the appropriate error shall be returned, and the encoding process aborted. The correct command to use is defined in 8.2.8.

The Write-Objects-Segmented-Memory-Tag command is like the Write-Objects command except that it is intended to write data to a selected memory bank in a segmented memory tag. The command may be implemented to write initial data to the RFID tag, or to add data to the tag. The command is supported by the following compound arguments, for which individual arguments are listed below:

Add-Objects-List

DSFID-Constructs

Ext-DSFID-Constructs

Multiple-Records-Constructs

Packed-Object-Constructs

8.2.4.2 Process requirements

To correctly implement this command in the Data Processor, the process arguments and compound arguments defined for the command in ISO/IEC 15961-1 shall be supported. For Multiple Records they are applied once per record, otherwise they are applied once per Logical Memory.

Access-Password (see 10.2.1)

AFI (see 10.2.3)

Data-CRC-Indicator (see 9.3.12)

DSFID (see 10.2.11)

DSFID-Lock (see 10.2.12)

DSFID-Pad-bytes (see 9.3.18)

Length-Of-Encoded-Data (see 9.3.11)

Memory-Bank (see 10.2.29)

Memory-Capacity (see 9.3.11)

Tag-Data-Profile-ID-Table (see 10.2.53)

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

The Access-Password in the application command shall match that encoded on the RFID tag to continue processing. If there is a mismatch, then an error shall be reported.

The main determinant for conformant encoding is the Access-Method, which is incorporated as part of the DSFID. The AFI and DSFID are provided as arguments for use in one of the following ways:

— If data is being written to a blank segmented memory RFID tag, then these arguments are provided as part of this command to minimise communications.

— Depending on the memory bank concerned, if data is being added to the segmented memory RFID tag, then the AFI and DSFID in the command should match the AFI and DSFID already encoded on the RFID tag, otherwise there is an error and the encoding process can cease before significant amounts of data have been processed.

To correctly implement this command in the Data Processor, the following process arguments shall be supported for each Object-Identifier and Object pair, irrespective of the Access-Method:

Avoid-Duplicate (see 10.2.7)

Compact-Parameter (see 10.2.9)

Object-Lock (see 10.2.38)

The Avoid-Duplicate, Compact-Parameter and Object-Lock arguments shall be applied in a manner consistent with the rules of the Access-Method (see Clause 11) to ensure reliable encoding. If there is an inconsistency within the application command, then it shall be in error and the encoding process shall not be implemented.

If the Access-Method = Packed-Objects, the input states are all defined in the Packed-Objects-Constructs argument specified in ISO/IEC 15961-1 and apply to a single Packed Object. That argument in turn has the following arguments, for which the processes are defined in Annex I:

Block-Align-Packed-Objects

Editable-Pointer-Size

ID-Type

Object-Offsets-Multiplier

Packed-Object-Directory-Type

PO-Directory-Size

PO-ID-Table

PO-Index-Length

If the Access-Method = Tag-Data-Profiles, the input states are all defined by the Tag-Data-Profile-ID-Table and encoded as a single Tag Data Profile using the processes defined in Annex N.

The following tag-related arguments, if declared by the Extended DSFID are used as follows to complete the encoding process:

— If the memory Memory-Capacity (see 9.3.11) is encoded, or the air interface has its own means of providing this information, it can be used to assess if there is sufficient memory for the command to be fully implemented.

— If one or both Data-CRC-Indicators (see 9.3.12) are set, then the CRC-16 shall be applied as defined in 9.3.13.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

— If the Length-Of-Encoded-Data (see 9.3.11) is signalled to be encoded, then this shall be calculated after all encoded bytes have been determined. This field is then written, or overwritten, taking into account the availability of DSFID-Pad-bytes (see 9.3.18).

If the compound argument Multiple-Records-Constructs is used in the command, then the Object-Identifier structure as defined in 11.6 shall be used for identifying each data Object in the add-Objects-List argument. If this is not the case, then the command shall not be processed. In addition to writing the Multiple-Record itself, this command can require other areas of the Logical Memory to be encoded: The MR-header if the number of records is maintained and the directory if already encoded or called for by the command arguments. If any part of the memory required to implement the command is locked, then the command shall be aborted.

Encoding shall be as defined in Annex Q if the first four arcs are 1.0.15961.401, or as defined in Annex R if the first four arcs are 1.0.15961.402 or 1.0.15961.403. The compound argument in turn has the following arguments, for which the processes are defined in Annex Q and Annex R.

Append-To-Existing-Multiple-Record (see 10.2.5)

Application-Defined-Record-Capacity (see 10.2.6)

Identifier-Of-My-Parent (see 10.2.17)

Lock-Directory-Entry (see 10.2.24)

Lock-Record-Preamble (see 10.2.26)

Number-In-Data-Element-List (see 10.2.36)

Record-Memory-Capacity (see 10.2.46)

Record-Type-Classification (see 10.2.48)

Update-Multiple-Records-Directory (see 10.2.58)

The data shall only be encoded in memory bank 11 of the segmented memory tag.

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

Object-Identifier (see 6.3)

The Completion-Codes for this command will vary based on the associated compound arguments. The complete list is:

0 No-Error

8 Singulation-Id-Not-Found

9 Object-Not-Added

10 Duplicate-Object

11 Object-Added-But-Not-Locked

25 Password-Mismatch

26 AFI-Mismatch

27 DSFID-Mismatch

29 Object-Not-Editable

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

31 Packed-Object-ID-Table-Not-Recognised-No-Encoding

32 Tag-Data-Profile-ID-Table-Not-Recognised

33 Insufficient-Tag-Memory

36 Command-Cannot-Process-Monomorphic-UII

37 Data-CRC-Not-Applied

38 Length-Not-Encoded-In-DSFID

43 Data-Format-Not-Compatible-Multiple-Records-Header

44 Access-Method-Not-Compatible-Multiple-Records-Header

45 Sector-Identifier-Not-Compatible-Multiple-Records-Header

46 Record-Preamble-Not-Configured

47 Record-Preamble-Not-Locked

255 Execution-Error

8.2.4.3 Conformance requirements

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— The encoder supports all aspects of the Access-Method.

— The encoder supports encoding to a segmented memory tag.

— The encoder supports a process to match the Access-Password in the command with that encoded on the RFID tag.

— The encoder supports all the processed defined by the Extended DSFID.

— The encoder supports the associated Completion-Codes and Execution-Codes.

— If any of the first four conditions are not supported an error is signalled by returning the Execution-Code = 4 Command-Not-Supported.

8.2.4.4 Guidance for appending data

The procedure described in 8.2.3.4 applies to this command.

8.2.5 Write-EPC-UII

8.2.5.1 Use

The Write-EPC-UII command specifies an EPC code to be written to the UII memory of a segmented memory tag (e.g. Memory Bank 01 of a Type C tag).

8.2.5.2 Process requirements

To correctly implement this command in the Data Processor, the following process arguments shall be supported:

Access-Password (see 10.2.1)

EPC-Code (see 10.2.15)

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Memory-Bank-Lock (see 10.2.30)

NSI-bits (see 10.2.35)

The Access-Password in the command is required to match that on the RFID tag to promote writing data to the RFID tag.

This command can be used to initially write the EPC-Code to the RFID tag, or to overwrite the code value. If the new code is of a shorter length, the interrogator shall ensure that bytes representing part of the older code are removed.

The interrogator shall calculate the length bits in the Protocol Control word. If the bits are provided by the application, the interrogator shall use its calculation of the length bits in preference.

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

The Completion-Codes for this command are:

0 No-Error

8 Singulation-Id-Not-Found

25 Password-Mismatch

33 Insufficient-Tag-Memory

255 Execution-Error

8.2.5.3 Conformance requirements

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— The encoder supports encoding to a segmented memory tag.

— The encoder supports a process to match the Access-Password in the command with that encoded on the RFID tag.

— The encoder supports the associated Completion-Codes and Execution-Codes.

8.2.6 Write-Password-Segmented-Memory-Tag

8.2.6.1 Use

The Write-Password-Segmented-Memory-Tag command does not require any encoding, other than a transfer using the device interface (e.g. as in ISO/IEC 24791-5) to the interrogator. If it is incorporated in the Data Processor, then the output interface will simply carry through the command arguments.

8.2.6.2 Process requirements

The following process arguments shall be supported:

Password (see 10.2.40)

Password-Type (see 10.2.41)

The following arguments shall be supported for the response:

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

The Completion-Codes for this command are:

0 No-Error

8 Singulation-Id-Not-Found

26 Password-Not-Written

255 Execution-Error

8.2.6.3 Conformance requirements

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— The encoder supports encoding to a segmented memory tag.

— The encoder supports a process to write the password in the application command to the appropriate memory location on the RFID tag.

— The encoder supports the associated Completion-Codes and Execution-Codes.

8.2.7 Write-Segments-6TypeD-Tag

8.2.7.1 Use

This command shall not be used to write a Monomorphic-UII. If the AFI on the RFID tag declares that it is registered for a Monomorphic-UII, the appropriate error shall be returned, and the encoding process aborted. The correct command to use is defined in 8.2.8.

The Write-Segments-6TypeD-Tag is used to write data to the UII segment, the item-related segment or both segments. The command may be implemented to write initial data to the Type D tag, or to add data to the tag.

The Write-Segments-6TypeD-Tag command is similar to the Write-Objects-Segmented-Memory-Tag command except that it is intended to write data to the Type D tag that has its segmented memory boundaries determined dynamically by the encoder. The command is supported by the following compound arguments, for which individual arguments are listed below:

Ext-DSFID-Constructs

Item-Related-Add-Objects-List

Item-Related-DSFID-Constructs

Multiple-Records-Constructs

Packed-Object-Constructs

UII-Add-Objects-List

UII-DSFID-Constructs

8.2.7.2 Process requirements

To correctly implement this command in the Data Processor, the process arguments and compound arguments defined for the command in ISO/IEC 15961-1 shall be supported. For Multiple Records they are applied once per record, otherwise they are applied once per Logical Memory.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

AFI (see 10.2.3)

DSFID-Lock (see 10.2.12)

Item-Related-DSFID (see 10.2.20)

Lock-UII-Segment-Arguments (see 10.2.27)

Memory-Segment (see 10.2.31)

Tag-Data-Profile-ID-Table (see 10.2.53)

UII-DSFID (see 10.2.56)

The Type D does not require implementing any of the features of the Extended DSFID that have an impact on encoding processes. The tag has its own methods for declaring memory capacity and length of the encoded data. It also calculates a CRC-16 for each segment. Given these hardware features, this command defines no processes associated with the Extended DSFID. The Extended DSFID is still required to encode an Access- Method code value greater than 3.

The main determinant for conformant encoding is the Access-Method, which is incorporated as part of the DSFID. The AFI and DSFID are provided as arguments for use in one of the following ways:

— If data is being written to a blank segment, then these arguments are provided as part of this command to minimise communications.

— Depending on the memory segment concerned, if data is being added to the segmented memory, then the AFI and DSFID in the command should match the AFI and DSFID already encoded on the RFID tag, else there is an error and the encoding process can cease before significant amounts of data have been processed.

To correctly implement this command in the Data Processor, the following process arguments shall be supported for each Object-Identifier and Object pair, irrespective of the Access-Method:

Avoid-Duplicate (see 10.2.7)

Compact-Parameter (see 10.2.9)

Object-Lock (see 10.2.38)

The Avoid-Duplicate, Compact-Parameter and Object-Lock arguments shall be applied in a manner consistent with the rules of the Access-Method (see Clause 11) to ensure reliable encoding. If there is an inconsistency within the application command, then it shall be in error and the encoding process shall not be implemented.

If the Access-Method = Packed-Objects, the input states are all defined in the Packed-Objects-Constructs argument specified in ISO/IEC 15961-1 and apply to a single Packed Object. That argument in turn has the following arguments, for which the processes are defined in Annex I:

Block-Align-Packed-Objects

Editable-Pointer-Size

ID-Type

Object-Offsets-Multiplier

Packed-Object-Directory-Type

PO-Directory-Size

PO-ID-Table

PO-Index-Length

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

If the Access-Method = Tag-Data-Profiles, the input states are all defined by the Tag-Data-Profile-ID-Table and encoded as a single Tag Data Profile using the processes defined in Annex N.

If the compound argument Multiple-Records-Constructs is used in the command, then the Object-Identifier structure as defined in 11.6 shall be used for identifying each data Object in the add-Objects-List argument. If this is not the case, then the command shall not be processed. In addition to writing the Multiple-Record itself, this command can require other areas of the Logical Memory to be encoded: The MR-header if the number of records is maintained and the directory if already encoded or called for by the command arguments. If any part of the memory required to implement the command is locked, then the command shall be aborted.

Encoding shall be as defined in Annex Q if the first four arcs are 1.0.15961.401, or as defined in Annex R if the first four arcs are 1.0.15961.402 or 1.0.15961.403. The compound argument in turn has the following arguments, for which the processes are defined in Annex Q and Annex R.

Append-To-Existing-Multiple-Record (see 10.2.5)

Application-Defined-Record-Capacity (see 10.2.6)

Identifier-Of-My-Parent (see 10.2.17)

Lock-Directory-Entry (see 10.2.24)

Lock-Record-Preamble (see 10.2.26)

Number-In-Data-Element-List (see 10.2.36)

Record-Memory-Capacity (see 10.2.46)

Record-Type-Classification (see 10.2.48)

Update-Multiple-Records-Directory (see 10.2.58)

The data shall only be encoded in the item-related segment of the tag memory.

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

The Completion-Codes for this command will vary based on the associated compound arguments. The complete list is:

0 No-Error

7 Object-Locked-Could-Not-Modify

8 Singulation-Id-Not-Found

9 Object-Not-Added

10 Duplicate-Object

11 Object-Added-But-Not-Locked

13 Object-Identifier-Not-Found

21 Object-Not-Modified

22 Object-Modified-But-Not-Locked

26 AFI-Mismatch

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

27 DSFID-Mismatch

29 Object-Not-Editable

31 Packed-Object-ID-Table-Not-Recognised-No-Encoding

32 Tag-Data-Profile-ID-Table-Not-Recognised

33 Insufficient-Tag-Memory

36 Command-Cannot-Process-Monomorphic-UII

37 Data-CRC-Not-Applied

38 Length-Not-Encoded-In-DSFID

43 Data-Format-Not-Compatible-Multiple-Records-Header

44 Access-Method-Not-Compatible-Multiple-Records-Header

45 Sector-Identifier-Not-Compatible-Multiple-Records-Header

46 Record-Preamble-Not-Configured

47 Record-Preamble-Not-Locked

255 Execution-Error

8.2.7.3 Conformance requirements

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— The encoder supports all aspects of the Access-Method.

— The encoder supports encoding to a Type D segmented memory tag.

— The encoder supports the associated Completion-Codes and Execution-Codes.

— If any of the first two conditions are not supported, an error is signalled by returning the Execution-Code = 4 Command-Not-Supported.

8.2.7.4 Guidance for appending data

The procedure described in 8.2.3.4 applies to this command.

8.2.8 Write-Monomorphic-UII

8.2.8.1 Use

The Write-Monomorphic-UII command instructs the Data Processor to write a Monomorphic-UII, either initially or to modify an existing Monomorphic-UII. Arguments are applied selectively depending on the type of RFID tag being addressed.

The generic encoding process calls for the Data Processor to check that the AFI in the command matches with the one for a Monomorphic-UII on the ISO/IEC 15961-2 Data Constructs register. If a match is not possible, either because the AFI is not registered for a Monomorphic-UII or that no matching AFI can be found on the register, then the encoding process is aborted.

If the AFI matches, the Object-Identifier is also checked for a match with that on the ISO/IEC 15961-2 Data Constructs register. A mismatch generates an appropriate Completion-Code, but this should only be treated as a warning about constructing the command. The process continues because the Object-

© ISO/IEC 2022 – All rights reserved

AVA Documents Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Identifier is not encoded. The Data Processor uses the explicitly defined compaction scheme associated with the AFI on the ISO/IEC 15961-2 Data Constructs register to encode the Monomorphic-UII.

8.2.8.2 Process requirements

To correctly implement this command in the Data Processor, the following process arguments shall be supported once per tag:

Access-Password (see 10.2.1)

AFI (see 10.2.3)

AFI-Lock (see 10.2.4)

Compact-Parameter (see 10.2.9)

Lock-UII-Segment-Arguments (see 10.2.27)

Memory-Bank-Lock (see 10.2.30)

Memory-Type (see 10.2.32)

Object-Lock (see 10.2.38)

The command argument Memory-Type (see 10.2.32) should match the type of RFID tag being encoded. If there is a match, then one of the following processes is applied.

For a Type C tag:

— If the Access-Password is in the command it is used to match that on the RFID tag to protect against unauthorised writing of data to the RFID tag.

— Next, read the content of MB 01 to establish any existing encoding. If this is locked the process is aborted, otherwise the process continues.

— The encoding process compacts the Object, and if this results in an odd number of bytes appends the terminator byte 00_{16}.

— The Protocol Control word is constructed, incorporating the AFI and the length bits.

— If this command is used to over-write MB01 with a new Monomorphic-UII it is necessary to compare the length of the current and the new byte string. If the new code is of a shorter length, the interrogator shall ensure that bytes representing part of the older code are over-written with zero bytes.

— If the command calls for the Monomorphic-UII to be locked, the entire MB01 is locked.

For a Type D tag:

— Read the content of the UII segment to establish any existing encoding. If this is locked the process is aborted, otherwise the process continues.

— The encoding process compacts the Object, and if this results in an odd number of bytes appends the terminator byte 00_{16}.

— The Protocol Control word is constructed, incorporating the AFI and the length bits. If the tag has data encoded in the item related segment, or has a simple sensor, these characteristics shall be encoded and maintained in the Protocol control word.

— If this command is used to over-write the UII segment with a new Monomorphic-UII it is necessary to compare the length of the current and the new byte string. If the new code is of a different length (i.e. shorter or longer) to an existing Monomorphic-UII, the interrogator shall check if an item related segment is already encoded. If the encoding can still be encoded in the UII segment without re-

© ISO/IEC 2022 – All rights reserved
AVA Documents Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

writing the item-related segment, the encoded bytes are transferred to the interrogator. If the UII segment shall be increased in size and none of the item-related bytes are locked, the entire memory can be re-written, otherwise the process shall be aborted.

NOTE The interrogator calculates a CRC-16 and places this at the end of the encoded words of the UII segment. Depending on the end position of the UII segment and the lock boundary, one or more pad words can be added by the interrogator to achieve alignment.

— If the command calls for the Monomorphic-UII to be locked, then the locking shall be as defined in the Lock-UII-Segment-Arguments.

If the RFID tag has a single memory for all data:

— The first requirement is to read the AFI on the RFID tag (which can be encoded in a separate memory area). If this matches, continue, otherwise report an AFI mismatch.

— Read at least 16 bytes of the content of the user memory to establish any existing encoding. If encoded bytes are found, then the process continues until a string of four zero bytes is found. If either the AFI or any part of the user memory is locked the process is aborted, otherwise the process continues.

— The encoding process compacts the Object and adds the length of the compacted Monomorphic-UII as a prefix.

— If this command is used to over-write a new Monomorphic-UII it is necessary to compare the length of the current and the new byte string. If the new code is of a shorter length, the interrogator shall ensure that bytes representing part of the older code are over-written with zero bytes.

— If the correct AFI for the Monomorphic-UII is not already encoded on the RFID tag, then this is encoded.

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

The Completion-Codes for this command are:

0 No-Error

7 Object-Locked-Could-Not-Modify

8 Singulation-Id-Not-Found

22 Object-Modified-But-Not-Locked

25 Password-Mismatch

26 AFI-Mismatch

33 Insufficient-Tag-Memory

34 AFI-Not-For-Monomorphic-UII

35 Monomorphic-UII-OID-Mismatch

255 Execution-Error

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

8.2.8.3 Conformance requirements

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— The Monomorphic-UII shall be compacted to the rules defined for the AFI on the ISO/IEC 15961-2 Data constructs register.

— The encoding on the tag shall be as defined in the processes above for the Memory-Type defined in the command.

8.2.9 Configure-Extended-DSFID

8.2.9.1 Use

Whereas the Configure-DSFID command (see 8.2.2) provides the Data Processor with the basic encoding rules to follow when encoding data, there have been additional requirements that call for an extended set of encoding instructions. These include providing the Data Processor with information about the memory capacity and length of encoded data where this cannot be provided by a mechanism in the tag architecture or air interface protocol. Information is also provided about the requirement to encode a data-related CRC, and to declare to the decoder that this process has been applied.

The Configure-Extended-DSFID command is used prior to encoding the set these encoding parameters and to also encode the base DSFID.

8.2.9.2 Process requirements

To correctly implement this command in the Data Processor, the following process arguments shall be supported once per tag:

Battery-Assist-Indicator (see 9.3.16)

Data-CRC-Indicator (see 9.3.12)

DSFID-Lock (see 10.2.12)

DSFID-Pad-bytes (see 9.3.18)

Full-Function-Sensor-Indicator (see 9.3.17)

Length-Of-Encoded-Data (see 9.3.11)

Memory-Capacity (see 9.3.11)

Memory-Length-Encoding (see 9.3.11)

Simple-Sensor-Indicator (see 9.3.15)

The Extended DSFID is encoded in a slightly different ways depending on the tag architecture:

— For tags where the DSFID is encoded as the first byte of memory, the entire Extended DSFID is encoded starting at the position of the single DSFID.

— For tags that have a dedicated DSFID memory, the first byte(s) of the Extended DSFID are encoded in the dedicated memory location. The remaining bytes are then encoded from the first position of the data memory.

If there is an intention to encode some extended DSFID parameters initially and add to this later, then sufficient pad bytes shall be reserved for this purpose.

The following arguments shall be supported for the response:

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

The Completion-Codes for this command are:

0 No-Error

4 DSFID-Not-Configured

5 DSFID-Not-Configured-Locked

6 DSFID-Configured-Lock-Failed

8 Singulation-Id-Not-Found

37 Data-CRC-Not-Applied

38 Length-Not-Encoded-In-DSFID

255 Execution-Error

8.2.9.3 Conformance requirements

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— If the RFID tag(s) supported by the encoder have the DSFID in a memory location requiring an explicit air interface command, then the Extended DSFID shall be split with the first byte(s) transported by the explicit air interface command, and the remaining bytes transferred with a single write transaction.

— If the RFID tag(s) supported by the encoder have the DSFID in a memory location requiring a generic air interface write command, then the Extended DSFID shall be written with a single write transaction to a specific location starting with the DSFID position determined by the Tag Driver.

— If the encoder supports writing the DSFID, then it shall also support the transfer of any DSFID-Lock argument so that it can be implemented in the air interface.

— The encoder shall support the associated Completion-Codes and Execution-Codes.

8.2.10 Configure-Multiple-Records-Header

8.2.10.1 Use

Encoding multiple records shall be undertaken in a precise sequence of stages. The first stage is to configure the MR-header, as defined in this clause. The Configure-Multiple-Records-Header command is used to create all the fields on the RFID tag within the MR-header that provides information about the types of record encoded on the RFID tag, including the directory. The command is supported by the following compound arguments defined in ISO/IEC 15961-1, for which additional inherent arguments are also defined.

DSFID-Constructs

Ext-DSFID-Constructs

8.2.10.2 Process requirements

To correctly implement this command in the Data Processor, the process arguments and compound arguments defined for the command in ISO/IEC 15961-1 shall be supported. The application command has the following arguments:

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Access-Password (see 10.2.1)

Directory-Length-EBV8-Indicator (see 10.2.13)

Lock-Multiple-Records-Header (see 10.2.25)

Multiple-Records-Features-Indicator (see 10.2.34)

Pointer-To-Multiple-Records-Directory (see 10.2.43)

Sector-Identifier (see 10.2.49)

Singulation-Id (see 9.3.2)

For RFID tags with segmented memory, the multiple records shall only be encoded in the user memory. Specifically, for Type C this is Memory Bank 11 and for Type D this is the item-related segment.

The bit setting for sensors and batteries shall only be applied if the tag has no hardware means of defining that a sensor is attached. Currently, Type C and Type D are the only RFID tags that support sensors. Both use hardware mechanisms to declare the presence of sensors and batteries.

The Extended DSFID component of the header is encoded in slightly different ways depending on the tag architecture:

— For tags where the DSFID is encoded as the first byte of memory, the entire Extended DSFID is encoded starting at the position of the single DSFID.

— For tags that have a dedicated DSFID memory, the first byte(s) of the Extended DSFID are encoded in the dedicated memory location. The remaining bytes are then encoded from the first position of the data memory.

The Data Processor shall accept the address at the start of the directory provided by the command, unless the address is non-existent in the tag memory. In this case, an error shall be reported.

Because the memory length encoding on the data CRC indicator refers to the requirement of these in the directory, no processing can take place whilst creating the header record.

The header shall be locked according to the instructions provided in the command.

The response arguments, as defined for the command response in ISO/IEC 15961-1, shall be supported by the Data Processor when responding to the application.

The Completion-Codes for this command will vary based on the associated compound arguments. The complete list is:

0 No-Error

4 DSFID-Not-Configured

8 Singulation-Id-Not-Found

25 Password-Mismatch

37 Data-CRC-Not-Applied

38 Length-Not-Encoded-In-DSFID

39 Multiple-Records-Header-Not-Configured

40 Multiple-Records-Header-Not-Locked

41 File-Support-Indicators-Not-Configured

42 File-Support-Indicators-Not-Locked

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

255 Execution-Error

8.2.10.3 Conformance requirements

8.2.10.3.1 DSFID conformance

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— If the RFID tag(s) supported by the encoder have the DSFID in a memory location requiring an explicit air interface command, then the Extended DSFID shall be split with the first byte(s) encoded by the explicit air interface command in the dedicated DSFID memory on the tag, and the remaining bytes encoded in the user memory.

— If the RFID tag(s) supported by the encoder have the DSFID in a memory location requiring a generic air interface write command, then the Extended DSFID shall be written to a specific location starting with the DSFID position determined by the Tag Driver.

— If the encoder supports writing the DSFID, then it shall also support the transfer of any DSFID-Lock argument so that it can be implemented in the air interface.

8.2.10.3.2 Other conformance

To conform, an encoder shall support this command and its arguments, subject to the following conditions:

— The encoder supports all aspects of encoding the header for the Multiple-Records Access-Method.

— The Data-CRC-Indicator shall be ignored for the header record itself.

— The total encoded data length (part of the Extended DSFID) refers to the size of the directory and shall be ignored for the header record itself.

— The encoder supports the associated Completion-Codes and Execution-Codes.

8.3 Application commands and responses — Read

8.3.1 Read-Object-Identifiers

8.3.1.1 Use

The Read-Object-Identifiers command instructs the interrogator to read all the Object-Identifiers from the RFID tag. This module can be used in advance of a more selective command to read a specific Object, or to identify duplicate Object-Identifiers so that a housekeeping procedure can be invoked. A valid response, if the RFID tag Logical Memory Map has no Object-Identifiers stored, is to return an empty Object-Identifiers list. Only one RFID tag shall be programmed per command to ensure that the read process is robust.

This command is not used to read Object-Identifiers from a Multiple Record. The functionality is incorporated in the Read-Multiple-Records command (see 8.3.8)

8.3.1.2 Process requirements

To correctly implement this command in the Data Processor, the following points shall be considered:

— An interrogator may provide all the functionality to achieve this application command. Alternatively, the interrogator may invoke a read command to capture all the encoded bytes a transfer the decode process to a higher-level device.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

— The different Access-Methods each provide a means of a faster delivery of the Object-Identifiers than reading the entire data on the tag. For example, the basic No-Directory syntax enables a decoder to "step over" the de-compaction process. Achieving this is an implementation issue for each decoder manufacturer.

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

Object-Identifier (see 6.3)

The Completion-Codes for this command are:

0 No-Error

8 Singulation-Id-Not-Found

255 Execution-Error

8.3.1.3 Conformance requirements

To conform, a decoder shall support this command and its arguments. In addition, it shall support the associated Completion-Codes and Execution-Codes.

8.3.2 Read-Logical-Memory-Map

8.3.2.1 Use

The Read-Logical-Memory-Map command is intended to read the entire memory content of the RFID tag and respond with this in a completely unstructured way (i.e. by returning the encoded byte values). No processing takes place through the Data Processor as part of this read command, so it is not possible to identify the encoding structure.

8.3.2.2 Process requirements

The command applies equally to all Access-Methods, but if a directory structure has been defined by the Access-Method, this shall be included in the response, but shall not be distinguished from other bytes in the Logical Memory Map.

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

Logical-Memory-Map (see list in Clause 10)

The Completion-Codes for this command are:

0 No-Error

8 Singulation-Id-Not-Found

19 Read-Incomplete

255 Execution-Error

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

8.3.2.3 Conformance requirements

To conform, a decoder shall support this command. In addition, it shall support the associated Completion-Codes and Execution-Codes.

8.3.3 Read-Objects

8.3.3.1 Use

The Read-Objects command is intended to read one or more data objects from the RFID tag, as defined by their Object-Identifiers. The argument Read-Type determines whether the first, selected, or all Object-Identifier(s) are read from the RFID tag. The command also supports an argument, Max-App-Length, which enables the application to prescribe an upper address point on the RFID tag beyond which reading is discontinued.

This command is not used to read Object-Identifiers from a Multiple Record. The functionality is incorporated in the Read-Multiple-Records command (see 8.3.8)

8.3.3.2 Process requirements

To correctly implement this command in the Data Processor, the following process arguments shall be supported:

Check-Duplicate (see 10.2.8)

Max-App-Length (see 10.2.28)

Read-Type (see 10.2.45)

The following arguments shall be supported for the response:

Compact-Parameter (see 10.2.9)

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

Lock-Status (see list in Clause 10)

Object (see 6.4)

Object-Identifier (see 6.3)

Except for Execution-Code, the response arguments are associated with each data element.

The Completion-Codes for this command are:

0 No-Error

8 Singulation-Id-Not-Found

10 Duplicate-Object

13 Object-Identifier-Not-Found

15 Object-Not-Read

255 Execution-Error

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

8.3.3.3 Conformance requirements

To conform, a decoder shall support this command and its arguments, subject to the following:

— If the Check-Duplicate argument is set to TRUE, then the entire encoded data shall be read to establish that the subject Object-Identifier is not duplicated. If the Access-Method is Multiple-Records, then the Check-Duplicate argument is only applied to the specific record being accessed. As there is no requirement to decode the data Object for Object-Identifiers not called out in the application command, a process like the Read-Object-Identifiers command could be used.

— If the Max-App-Length argument is invoked, the interrogator and decoder shall only process the bytes up to the value for that argument.

— The decoder shall support the associated Completion-Codes and Execution-Codes.

8.3.4 Inventory-ISO-UIImemory

8.3.4.1 Use

The Inventory-ISO-UIImemory command is intended to return the contents of the UII memory from several segmented memory tags, given the expectation that an Object-Identifier for a non-EPC-Code is encoded. The response returns the content of the UII memory for all tags whose encoded bit string matches the arguments of the command.

8.3.4.2 Process requirements

To correctly implement this command in the Data Processor, the following arguments shall be supported:

Additional-App-bits (see 10.2.2)

AFI (see 10.2.3)

DSFID (see 10.2.11)

The arguments provided in the command enable a bit mask to be incorporated into appropriate air interface protocol commands to select only tags that match the bit mask. For tags with encoding based on Object-Identifier, the AFI is included as a minimum component of the mask. The DSFID may be added, particularly where its value is constant for an application. Additional bits may be added that, for example, identify the Precursor of the first (sometimes only) encoded dataset and any additional bits that a certain to create a tag selection criterion.

The Data Processor concatenates these values into a contiguous bit stream and prepends this with a bit = "1" and identifies this bit stream as beginning at bit 17h of Memory Bank 01. An error occurs if bit 17h of Memory Bank 01 = "0".

The following arguments shall be supported for the response:

Compact-Parameter (see 10.2.9)

Completion-Code (see 10.3 and below)

DSFID (see 10.2.11)

Execution-Code (see 10.4)

Lock-Status (see list in Clause 10)

Object (see 6.4)

Object-Identifier (see 6.3)

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

Protocol-Control-Word (see list in Clause 10)

If the UII memory encodes more than one Object-Identifier, then these arguments shall be returned for each data element: Compact-Parameter, Completion-Code (some values), Lock-Status, Object, and Object- Identifier.

The Completion-Codes for this command are:

0 No-Error

10 Duplicate-Object

13 Object-Identifier-Not-Found

15 Object-Not-Read

255 Execution-Error

8.3.4.3 Conformance requirements

To conform, a decoder shall support this command and its arguments, subject to the following conditions:

— The decoder supports decoding from a segmented memory tag.

— The decoder supports the associated Completion-Codes and Execution-Codes.

8.3.5 Inventory-EPC-UIImemory

8.3.5.1 Use

The Inventory-EPC-UIImemory command is intended to return the contents of the UII memory from several segmented memory tags, given the expectation that an EPC-Code is encoded. The response returns the content of the UII memory for all tags whose encoded bit string matches the arguments of the command.

8.3.5.2 Process requirements

To correctly implement this command in the Data Processor, the following arguments shall be supported:

Length-of-Mask (see 10.2.23)

Pointer (see 10.2.42)

Tag-Mask (see 10.2.54)

If the value of the pointer is less than 18h, an error occurs if bit 17h of Memory Bank 01 = "1".

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

EPC-Code (see 10.2.15)

Execution-Code (see 10.4)

Protocol-Control-Word (see list in Clause 10)

The Completion-Codes for this command are:

0 No-Error

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022

255 Execution-Error

8.3.5.3 Conformance requirements

To conform, a decoder shall support this command and its arguments, subject to the following conditions:

— The decoder supports decoding from a segmented memory tag.

— The decoder supports the associated Completion-Codes and Execution-Codes.

8.3.6 Read-Words-Segmented-Memory-Tag

8.3.6.1 Use

The Read-Words-Segmented-Memory-Tag command instructs the interrogator to read a contiguous sequence of words from one of the memory banks of a segmented memory RFID tag. This command can be used to extract encoded bytes, which might not be object-based such as the unique Singulation-Id or a password. It can also be useful for diagnostic purposes.

8.3.6.2 Process requirements

To correctly implement this command in the Data Processor, the following arguments shall be supported:

Access-Password (see 10.2.1)

Memory-Bank (see 10.2.29)

Word-Count (see 10.2.59)

Word-Pointer (see 10.2.60)

The following arguments shall be supported for the response:

Completion-Code (see 10.3 and below)

Execution-Code (see 10.4)

Read-Data (see list in Clause 10)

The Completion-Codes for this command are:

0 No-Error

25 Password-Mismatch

255 Execution-Error

8.3.6.3 Conformance requirements

To conform, a decoder shall support this command and its arguments, subject to the following conditions:

— The decoder supports decoding from a segmented memory tag.

— The decoder supports a process to match the Access-Password in the command with that encoded on the RFID tag.

— The decoder supports the associated Completion-Codes and Execution-Codes.

© ISO/IEC 2022 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2022