BS ISO/IEC 15962:2013

BSI Standards Publication

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

bsi.

...making excellence a habit.™

Document Sharing Hub

National foreword

This British Standard is the UK implementation of ISO/IEC 15962:2013. It supersedes BS ISO/IEC 15962:2004 which is withdrawn.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 secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2013.

Published by BSI Standards Limited 2013

ISBN 978 0 580 61907 6

ICS 35.040

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 31 March 2013.

Amendments issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 15962:2013

INTERNATIONAL STANDARD

ISO/IEC 15962

Second edition
2013-03-15

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'ojects — Protocole de données: règles d'encodage des données et fonctions logiques de mémoire

Reference number
ISO/IEC 15962:2013(E)

Documents
Document Sharing Hub

© ISO/IEC 2013

Full document details: BS ISO/IEC 15962:2013

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2013

All rights reserved. Unless otherwise specified, 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
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 15962:2013

Contents

Page

Foreword vii

Introduction viii

1 Scope 1

2 Normative references 1

3 Terms, definitions and conventions 2
3.1 Terms and definitions 2
3.2 Conventions 2

4 Conformance 2
4.1 Conformance with the air interface 2
4.2 Conformance with the application interface 2
4.3 Conformance with the Access-Method 3

5 Protocol model 4
5.1 Overview 4
5.2 Layered protocol 4
5.3 Flexible implementation configurations 6
5.4 Functional processes – interrogator implementation 6
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 10
6.2 Extensible bit vector (EBV) 10
6.3 Object Identifier presentation in the application interface 10
6.4 The Object 12
6.5 The 8-bit byte 12
6.6 N-bit encoding 12

7 Data Processor – high level processing 12

8 Data Processor and the application interface 13
8.1 Application commands – overview 13
8.2 Application commands and responses– write 15
8.3 Application commands and responses– read 31
8.4 Application commands and responses– other 40
8.5 Air interface support for application commands 49

9 Data Processor and the air interface 49
9.1 Air interface services 49
9.2 Defining the system information 50
9.3 Configuring the Logical Memory 58

10 The Command/Response Unit: processing of command and response arguments 58
10.1 Process arguments 59
10.2 Completion-Codes 71
10.3 Execution-Codes 74

11 Access-Method 74
11.1 No-Directory structure 75
11.2 Directory structure 77
11.3 Packed-Objects structure 79
11.4 Tag Data Profile 80
11.5 Multiple-Records 80

Documents
Document Sharing Hub

© ISO/IEC 2013 – All rights reserved

Full document details: BS ISO/IEC 15962:2013

12 ISO/IEC 15434 direct encoding and transmission method using Access-Method 0 and Data-Format 3 86
12.1 General rules for ISO/IEC 15434 direct encoding 86
12.2 Specific support for ISO TC122 standards 87

13 Monomorphic-UII encoding 87
13.1 6-bit encoding 88
13.2 7-bit encoding 88
13.3 URN Code 40 encoding 88
13.4 8859-1 octet encoding 89
13.5 Application-defined 8-bit coding 89

Annex A (informative) **Air interface support for application commands** 90
A.1 Overview 90
A.2 ISO/IEC 18000-3 Mode 1 support 90
A.3 ISO/IEC 18000-6 Type C support 91
A.4 ISO/IEC 18000-6 Type D support 93

Annex B (normative) **Pro forma description for the Tag Driver** 96
B.1 Defining the Singulation-Id 96
B.2 System information : AFI 96
B.3 System information: DSFID 96
B.4 Memory-related parameters 96
B.5 Support for commands 97

Annex C (normative) **ISO/IEC 18000 Tag Driver Descriptions** 98
C.1 Tag Driver for ISO/IEC 18000-2: Parameters for air interface communications below 135 kHz 98
C.2 Tag Driver for Mode 1 of ISO/IEC 18000-3: Parameters for air interface communications at 13,56 MHz 99
C.3 Tag Driver for Mode 2 of ISO/IEC 18000-3: Parameters for air interface communications at 13,56 MHz 101
C.4 Tag Driver for ISO/IEC 18000-4: Parameters for air interface communications at 2,45 GHz - Mode 1 102
C.5 Tag Driver for ISO/IEC 18000-4: Parameters for air interface communications at 2,45 GHz - Mode 2 104
C.6 Tag Driver for ISO/IEC 18000-6 Type A: Parameters for air Interface Communications at 860 MHz to 960 MHz 104
C.7 Tag Driver for ISO/IEC 18000-6 Type B: Parameters for air Interface Communications at 860 MHz to 960 MHz 105
C.8 Tag Driver for ISO/IEC 18000-6 Type C: Parameters for air Interface Communications at 860 MHz to 960 MHz 107
C.9 Tag Driver for ISO/IEC 18000-6 Type D: Parameters for air Interface Communications at 860 MHz to 960 MHz 108

Annex D (normative) **Encoding rules for No-Directory Access-Method** 112
D.1 Object processing 112
D.2 Encoding the length of the compacted Object 114
D.3 Processing the Object-Identifier 114
D.4 Processing the Relative-OID 116
D.5 Encoding the length and Object-Identifier or Relative-OID 119
D.6 The Precursor 120
D.7 The Offset byte 121
D.8 The Precursor expansion byte 121
D.9 Decoding the Logical Memory 122

Annex E (normative) **Basic Data Compaction Schemes** 125
E.1 Integer compaction 125
E.2 Numeric compaction 125
E.3 5-bit compaction 126
E.4 6-bit compaction 126
E.5 7-bit compaction 127
E.6 Octet encodation 129

Documents
Document Sharing Hub

© ISO/IEC 2013 – All rights reserved

Full document details: BS ISO/IEC 15962:2013

Annex F (normative) **ISO/IEC 646 Characters Supported by the Compaction Schemes** 130

Annex G (informative) **Encoding example for No-Directory structure** 133
G.1 Starting position 133
G.2 Encoding the Object-Identifiers 133
G.3 The initial state of the entry for the Logical Memory 133
G.4 The Logical Memory after data compaction 134
G.5 The Logical Memory after formatting for a No-Directory Access-Method 134

Annex H (informative) **Encoding example for Directory structure** 136
H.1 The base data 136
H.2 Encoding the first Directory entry 136
H.3 Encoding the second Directory entry 137
H.4 Encoding the remaining Directory entries 137
H.5 Decoding the Directory and reading the target Object-Identifier 138

Annex I (normative) **Packed-Objects structure** 139
I.1 Overview 139
I.2 Overview of associated Annexes 139
I.3 High-level Packed-Objects format design 139
I.4 Format Flags section 142
I.5 Object Info section 144
I.6 Secondary ID Bits section 150
I.7 Aux Format section 150
I.8 Data section 152
I.9 ID Map and Directory encoding options 155

Annex J (normative) **Packed Objects ID Tables** 161
J.1 Packed Objects Data Format registration file structure 161
J.2 Mandatory and Optional ID Table columns 163
J.3 Syntax of OIDs, IDString, and FormatString columns 166
J.4 OID input/output representation 168

Annex K (normative) **Packed Objects Encoding tables** 170

Annex L (informative) **Encoding example for Packed Objects** 175

Annex M (informative) **Decoding Packed Objects** 179
M.1 Overview 179
M.2 Decoding Alphanumeric data 180

Annex N (normative) **Tag Data Profile encoding** 183
N.1 Scope 183
N.2 The Registered Table 183
N.3 Encoding the Tag Data Profile on the RFID tag 184
N.4 Decoding the Tag Data Profile 186
N.5 Modifying Data 187

Annex O (normative) **Tag Data Profile ID tables** 188
O.1 Tag-Data-Profile Data-Format registration file structure 188
O.2 File Header section 189
O.3 Table Header section 189
O.4 Table Trailer section 190
O.5 Mandatory ID Table columns 190

Annex P (informative) **Encoding example for Tag Data Profile** 192
P.1 Encoded data segment 192
P.2 Encoding the header segment 195

Annex Q (normative) **Basic encoding rules for Multiple-Records Access-Method** 196
Q.1 Overview 196
Q.2 Encoding the Multiple-Records header 196
Q.3 Encoding the preamble of an individual record that is not part of a hierarchical structure 200
Q.4 The record 203
Q.5 The directory 203

Documents
Document Sharing Hub

© ISO/IEC 2013 – All rights reserved

Full document details: BS ISO/IEC 15962:2013

Q.6 Appending a new record ... 207
Q.7 Modifying an existing record ... 208
Q.8 Deleting an existing record ... 208
Q.9 Constructing the Object-Identifier from the MR-header, preamble and individual record ... 209

Annex R (normative) **Multiple-Records encoding rules for hierarchical records** ... 212
R.1 Overview ... 212
R.2 Encoding the Multiple-Records header ... 213
R.3 Encoding the preamble of hierarchical record ... 214
R.4 The hierarchical record ... 216
R.5 Data element list ... 216
R.6 The directory ... 218
R.7 Appending a new record ... 218
R.8 Modifying an existing record ... 218
R.9 Deleting a record ... 218

Annex S (informative) **Encoding example for the Multiple-Records Access-Method** ... 219
S.1 The heterogeneous multiple record example ... 219
S.2 An encoding example of a homogeneous multiple record ... 226
S.3 An encoding example of a hierarchical multiple record ... 229

Annex T (normative) **ISO/IEC 15434 Direct Encoding and Transmission** ... 232
T.1 DSFID ... 232
T.2 Precursor byte ... 232
T.3 Data byte-count indicator ... 232
T.4 Encoding and Decoding ... 233
T.5 Encoding and Decoding Example using Data Identifiers ... 234
T.6 Additional Code Values and other Precursor features ... 236

Annex U (informative) **ISO/IEC 15434 Direct DI Encoding and Transmission for ISO TC122 Standards** ... 238
U.1 DSFID ... 238
U.2 Precursor byte ... 238
U.3 Data byte-count indicator ... 238
U.4 Encoding and Decoding ... 239
U.5 Encoding and Decoding Example ... 241

Annex V (normative) **URN Code 40 encoding** ... 243
V.1 Basic Character Set ... 243
V.2 Extended Encoding ... 244
V.3 Encoding Example ... 245
V.4 Resolver Example ... 245

Bibliography ... 246

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 15962 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 15962:2004), which has been technically revised.

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

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 ISO/IEC 18000. 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 systems implementation for it to be read at a later time in a completely different and unknown systems 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 International Standard. 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 ISO/IEC 24791 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 International Standard and the functional rules of the commands and responses in ISO/IEC 15961-1.

Manufacturers of RFID equipment (interrogators, RFID tags, etc.) and the users of RFID technology require a standards-based data protocol for RFID for item management. ISO/IEC 15961-1 to ISO/IEC 15961-3, this International Standard, and ISO/IEC 24791 specify this protocol, which is layered above the air interface standards defined in ISO/IEC 18000.

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

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

INTERNATIONAL STANDARD **ISO/IEC 15962:2013(E)**

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

1 Scope

The data protocol used to exchange information in an RFID system for item management is specified in ISO/IEC 15961 and in this International Standard. Both International Standards are required for a complete understanding of the data protocol in its entirety; but each focuses on one particular interface:

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

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

This International Standard focuses on encoding the transfer syntax, as defined in ISO/IEC 15961 according to the application commands defined in ISO/IEC 15961. 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 International Standard

— 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 referenced documents are indispensable for the application 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 19762-1, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 1: General terms relating to AIDC*

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

ISO/IEC 19762-3, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 3: Radio frequency identification (RFID)*

3 Terms, definitions and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19762-1, ISO/IEC 19762-3 and the following apply.

NOTE For terms defined below and in ISO/IEC 19762-1 or ISO/IEC 19762-3, the definitions given below apply.

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

3.1.2
Data Processor
implementation of the processes defined in this International Standard, 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

3.1.4
Relative-OID
particular object identifier where a common root-OID (for the first and subsequent arcs) is implied, and remaining arcs after the Root-OID are defined by the Relative-OID

3.2 Conventions

Conventionally in International Standards, long numbers are separated by a space character as a "thousands separator". This convention has not been followed in this International Standard, because the arcs of an object identifier are defined by a space separator (according to ISO/IEC 8824 and ISO/IEC 8825). As the correct representation of these arcs is vital to this International Standard, all numeric values have no space separators except to denote a node between two arcs of an object identifier.

4 Conformance

Conformance to this International Standard shall depend on the functional capability of the device as defined in the following three sub-clauses.

4.1 Conformance with the air interface

A conformant implementation of this International Standard 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

The conformance requirements depend on the type of device as follows:

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

4.2.1 Encoders and the application interface

Within the constraints of the air interface protocol supported, a conformant implementation of this International Standard 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).

4.2.2 Decoders and the application interface

Within the constraints of the air interface protocol supported, a conformant implementation of this International Standard 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 International Standard 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

The conformance requirements depend on the type of implementation as follows:

4.3.1 Encoders and the Access-Method

A conformant implementation of this International Standard 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 International Standard 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 needs to be fully decoded by a decoder process, external to the interrogator, that is fully compliant with the rules defined in this International Standard.

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 International Standard 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.

AVA Documents
Document Sharing Hub

© ISO/IEC 2013 – All rights reserved

Full document details: BS ISO/IEC 15962:2013

5 Protocol model

5.1 Overview

RFID supports bit encodation in the RFID tag memory. Unlike other data carrier standards prepared by ISO/IEC JTC1 SC31 which require encodation schemes that are specific to the individual data carrier technology, ISO/IEC 18000 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 needs to 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 International Standard is to specify the common encoding rules and logical memory functions.

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 International Standards.

Different application standards may have their own particular data sets or data dictionaries. Each major application standard for item management needs to 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 International Standards ensures the unambiguous identification of data.

5.2 Layered protocol

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

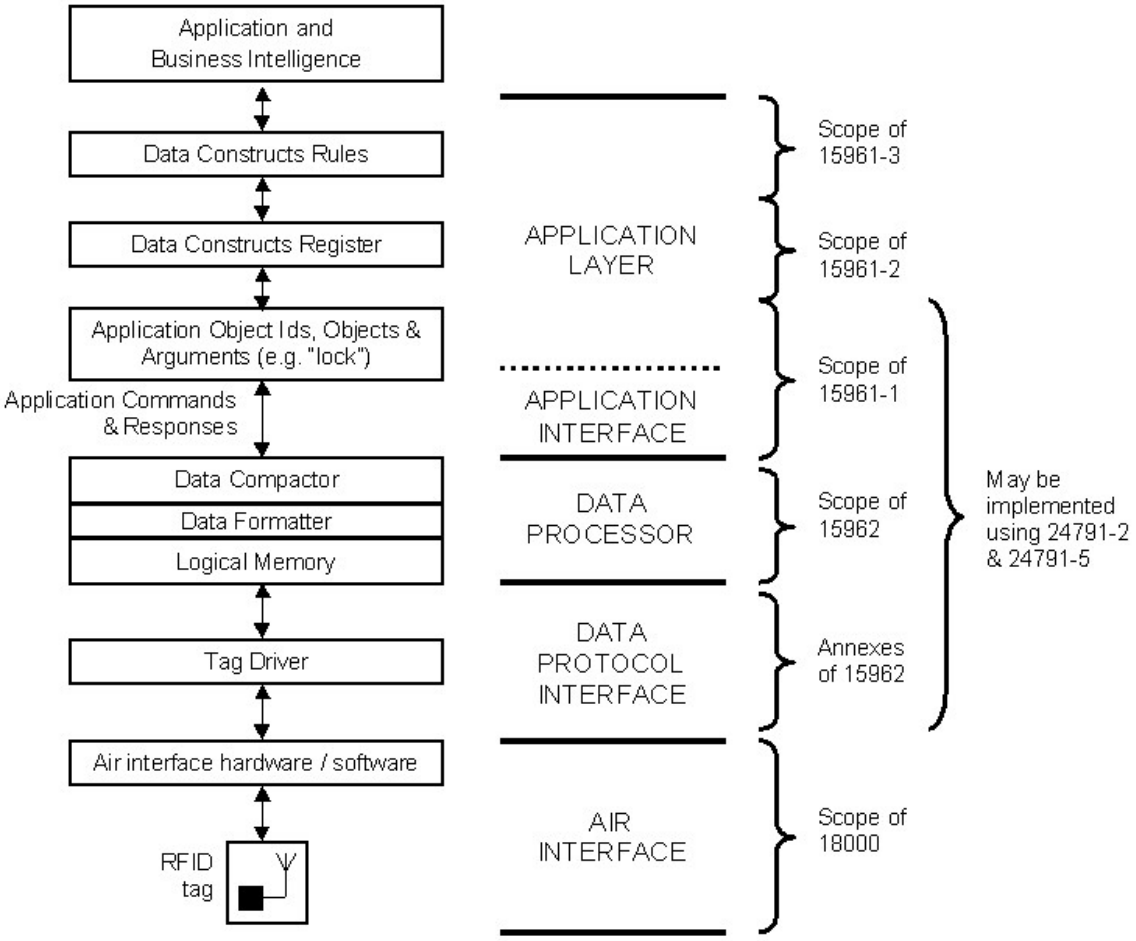


Figure 1 — Schematic of protocol layers for an implementation of RFID for item management

© ISO/IEC 2013 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

5.2.1 Application layer - as defined in the various parts of ISO/IEC 15961

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 needs to be registered according to the rules of 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 transfer to and from the RFID tag and to append, update, selectively lock, delete data, or perform other functions on the RFID tag.

To illustrate how the functions and arguments are assembled into a structured format, a number of commands and responses have been constructed using an abstract syntax. This is independent of the host application, operating system, and programming language and also independent of the specific command structures between the interrogator and tag driver. The abstract syntax used in ISO/IEC 15961-1 is similar to 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 included commands defined using ASN.1 abstract syntax. 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.

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

The application interface may be implemented in a number of different ways that are not explicitly defined in this International Standard, 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 needs to 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 International Standard, 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 particular application interface process to be adopted, other than the requirement that it be integrated with the encoding rules of ISO/IEC 15962.

An alternative process, consistent with the first edition of ISO/IEC 15961, is to use the abstract syntax for defining the commands and responses in a structured, consistent and verifiable manner. It is then necessary to generate the transfer encoding that defines the byte stream transferred between the processes of this International Standard and those 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 has to be complaint with all the arguments in the commands specified in ISO/IEC 15961-1.

5.2.3 Data Protocol Processing - as defined in this International Standard

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.

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

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.

5.2.4 Data Protocol Interface - as defined in this International Standard

Each air interface protocol standard in ISO/IEC 18000 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.

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.

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 International Standard incorporated into the software system infrastructure architecture defined in ISO/IEC 24791. This is the recommended approach for any networked application.

— with ISO/IEC 15961 Part 1 and this International Standard incorporated into stand-alone software or devices that have as its output conformant encoding and / or conformant decoding with responses compliant to the responses of ISO/IEC 15961-1.

5.4 Functional processes – interrogator implementation

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 — Logical functions and interfaces of ISO/IEC 15962 with other RFID system components 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 compliant with this data protocol.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

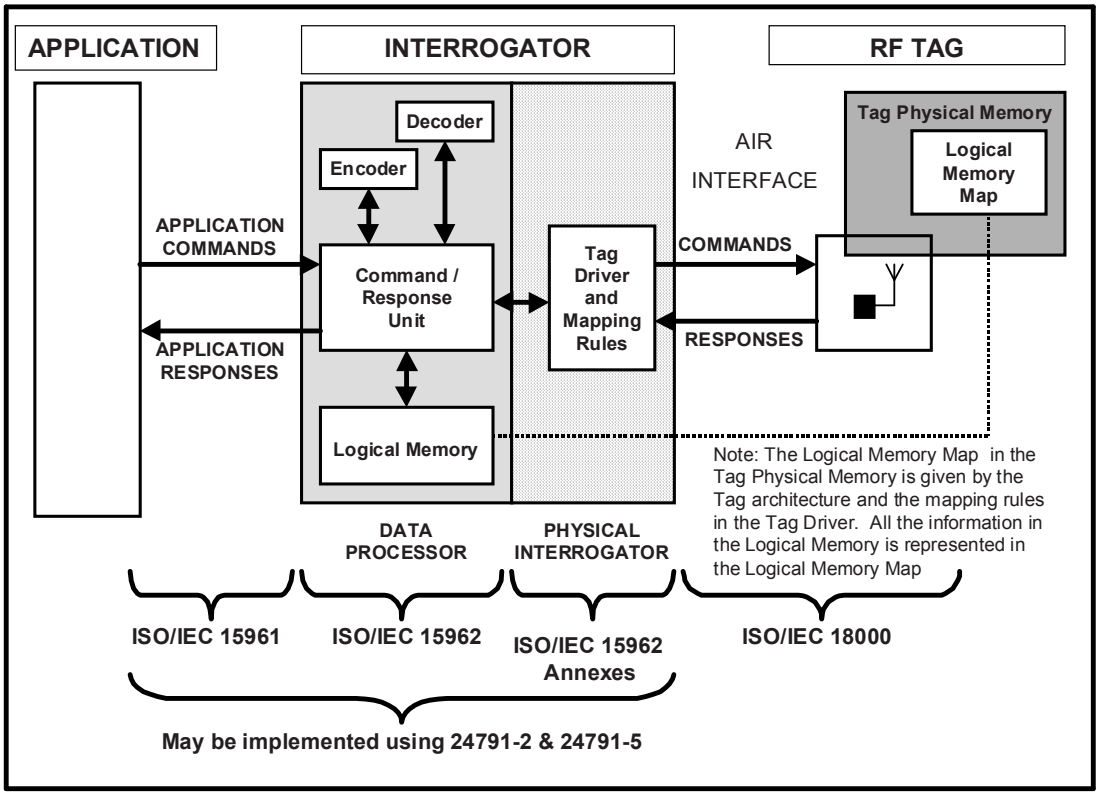


Figure 2 — Logical functions and interfaces of ISO/IEC 15962 with other RFID system components

5.4.1 Functional processes – application interface

The data flows between the application and the Data Processor are formatted according to this International Standard and are not compacted. However, there are numerous established systems where data is formatted to be compliant, 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.

NOTE 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 International Standard 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.

5.4.2 Functional processes – interrogator

In the process illustrated in Figure 2 — Logical functions and interfaces of ISO/IEC 15962 with other RFID system components, 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 might separate some of the functions described below and have an interface between the application and the physical interrogator.

5.4.2.1 Data processor

The Data Processor provides all the processing, which is as specified in this International Standard 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.

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

5.4.2.1.1 Command/Response unit

The Command/Response Unit receives the application commands from the application in a format specified in this International Standard, 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, etc) 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.

5.4.2.1.2 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 takes into account 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.

The use of the Logical Memory means that an application can interface with an application-compliant 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 ISO/IEC 18000, such as larger capacity or faster access RFID tags, without changing the application.

5.4.2.1.3 Encoder

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

5.4.2.1.4 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.

5.4.2.1.5 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 comply with the RFID tag architecture is performed based on information provided by the Tag Driver.

5.4.2.1.6 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.

5.4.2.2 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.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

— 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 particular RFID tags is provided in annexes of this International Standard. For the purpose of this International Standard, a tag driver is unique to a particular air interface type of RFID tag as specified in the appropriate part of ISO/IEC 18000. This is a logical representation; physical implementation could combine features of different logical tag drivers. An interrogator may support one or many tag drivers.

5.4.2.3 Transfer mechanisms

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

5.4.3 RFID tag

Although the RFID tag is beyond the scope of this International Standard, the tag is shown in Figure 2 — Logical functions and interfaces of ISO/IEC 15962 with other RFID system components to complete the flow of data and commands. Within the RFID tag, the Logical Memory Map represents all the data in 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 International Standard defines all the rules for encoding data on an RFID tag. The implementation of these rules is described in this International Standard 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 might be completely unaware of what organisation might 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

The following conventions are used:

— The field/argument names are shown in **bold** type face using this format: **Access-Method**.

— Basic data types are shown in UPPERCASE type face using this format: BOOLEAN

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

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 International Standard, 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 International Standard, where n refers to the number of bits in each block. EBV-8 is identical to the length encoding (see 9.2.10), which has been used since the original edition of this International Standard.

6.3 Object Identifier presentation in the application interface

6.3.1 Object identifier structure to ISO/IEC 8824-1

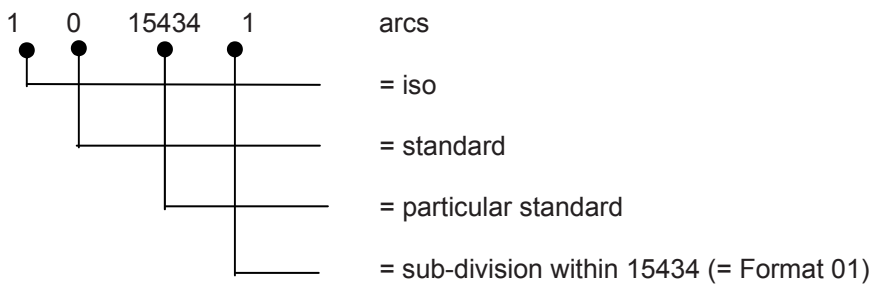
This International Standard uses the OBJECT IDENTIFIER type as defined in ISO/IEC 8824-1 and with identifiers assigned as specified in ISO/IEC 9834-1. This uses a registration tree with a common implied root node (ISO/IEC 9834-1), a series of arcs from each node, with new arcs added as required to define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

— 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

EXAMPLE



The only top arcs permitted for all object identifiers are shown in Table 1 — Object Identifier Top Arcs.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

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 Part 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 International Standard. The full structure is suitable for use in this International Standard 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 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.

— **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 International Standard.

6.3.2 Presenting the Object-Identifier in the style of ISO/IEC 8824-1

When the **Object-Identifier** is presented in application commands and responses in the style of 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

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

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 International Standard. The **Object** shall be qualified by the use of 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 The 8-bit byte

This International Standard 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 — Bit sequence in the 8-bit byte.

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 International Standard, which in turn complied with ASN.1 rules.

6.6 N-bit encoding

This International Standard 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 3 — High level information flows, which focuses on the encoding process.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

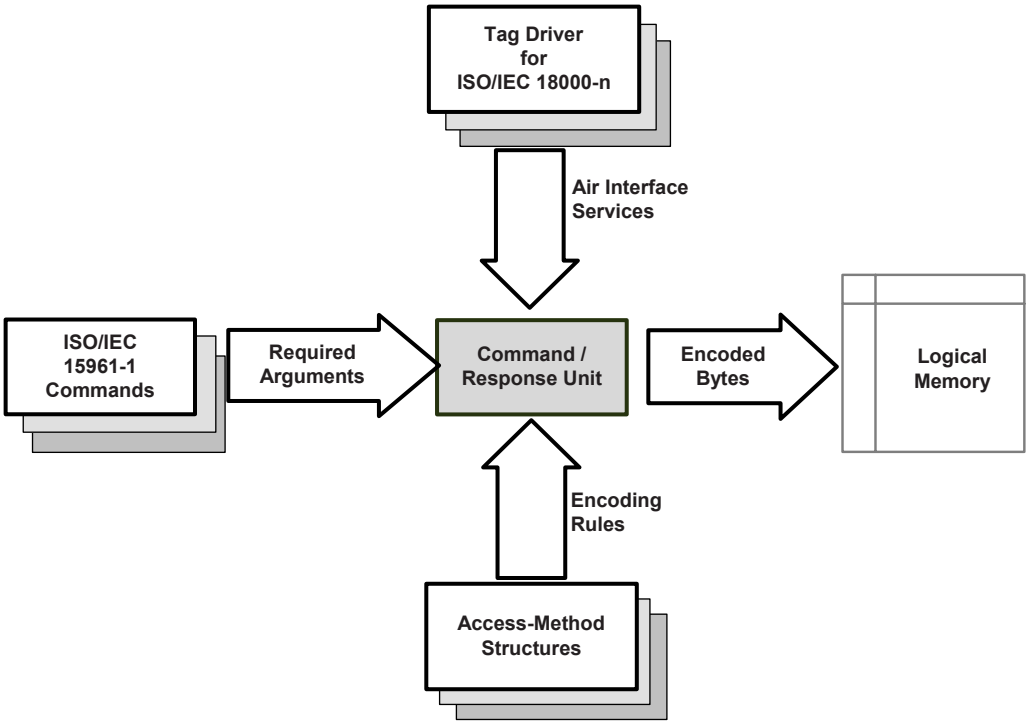


Figure 3 — High level information flows

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 International Standard.

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 needs to be clearly defined against optional components and Revisions to the ISO/IEC 18000 series of standards.

8 Data Processor and the application interface

8.1 Application commands – overview

ISO/IEC 15961-1 defines a number of commands and associated responses to provide instructions of how data is to be encoded on the RFID tag. Table 3 — ISO/IEC 15961-1 commands 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 original publication (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 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

Table 3 — ISO/IEC 15961-1 commands

Code	Command Name	Write	Read	Other
1	Configure-AFI	Note 1		
2	Configure-DSFID	Note 1		
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			Note 2
13	Get-App-based-System-Info			Note 3
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

NOTE 1. 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 18000-3M1). Others use the write command, but require the data to be written into a specific location (e.g. ISO/IEC 18000-6B and ISO/IEC 18000-6C).

2. Some air interface protocols support a specific Erase-Memory command, whereas others require the data to be overwritten with a null-byte value (typically 00_{16}).

3. 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 — ISO/IEC 15961-1 commands 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 — ISO/IEC 15961-1 commands were originally specified in ISO/IEC 15961:2004. This International Standard no longer needs to support these commands, mainly because their functionality has been absorbed in other commands. The original commands may continue to be supported by encoding systems compliant with ISO/IEC 15962:2004.

To implement the encoding rules of this International Standard it is necessary to understand the detail of the commands and responses in ISO/IEC 15961-1. The sub-clauses 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 sub-clauses.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.2 Application commands and responses– write

8.2.1 Configure-AFI

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 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 **AFI** using a generic air interface write command.

8.2.1.1 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 10.1.3)
AFI-Lock (see 10.1.4)

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)

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

8.2.1.2 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**.

8.2.2 Configure-DSFID

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 International Standard. 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.

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.2.2.1 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 10.1.11)
DSFID-Lock (see 10.1.12)

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)

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

8.2.2.2 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

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** 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

8.2.3.1 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 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

Data-CRC-Indicator (see 9.2.11)
DSFID (see 10.1.11)
DSFID-Lock (see 10.1.12)
DSFID-Pad-Bytes (see 9.2.17)
Length-Of-Encoded-Data (see 9.2.10)
Memory-Capacity (see 9.2.10)
Tag-Data-Profile-ID-Table (see 10.1.53)

The main determinant for compliant 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, 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 10.1.7)
Compact-Parameter (see 10.1.9)
Object-Lock (see 10.1.38)

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 considered to 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 9.2.10) is encoded, or the air interface has its own means of providing this information, it can be used to assess in there is sufficient memory for the command to be fully implemented.

— If one or both **Data-CRC-Indicators** (see 9.2.11) is set, then the CRC-16 shall be applied as defined in 9.2.12.

— If the **Length-Of-Encoded-Data** (see 9.2.10) 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 9.2.17).

Documents
Document Sharing Hub

© ISO/IEC 2013 – All rights reserved

Full document details: BS ISO/IEC 15962:2013

If the compound argument **Multiple-Records-Constructs** is used in the command, then the **Object-Identifier** structure as define in 11.5 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 10.1.5)
Application-Defined-Record-Capacity (see 10.1.6)
Identifier-Of-My-Parent (see 10.1.17)
Lock-Directory-Entry (see 10.1.24)
Lock-Record-Preamble (see 10.1.26)
Number-In-Data-Element-List (see 10.1.36)
Record-Memory-Capacity (see 10.1.46)
Record-Type-Classification (see 10.1.48)
Update-Multiple-Records-Directory (see 10.1.58)

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)
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

8.2.3.2 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 processed defined by the Extended DSFID

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

— 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**.

8.2.3.3 Guidance for appending data

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

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 of the new encoded data to ensure that it was stored correctly, then

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

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 similar to 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

8.2.4.1 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 10.1.1)
AFI (see 10.1.3)
Data-CRC-Indicator (see9.2.11)
DSFID (see 10.1.11)
DSFID-Lock (see 10.1.12)
DSFID-Pad-Bytes (see 9.2.17)
Length-Of-Encoded-Data (see 9.2.10)
Memory-Bank (see 10.1.29)
Memory-Capacity (see 9.2.10)
Tag-Data-Profile-ID-Table (see 10.1.53)

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.

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

The main determinant for compliant 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, 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 10.1.7)
Compact-Parameter (see 10.1.9)
Object-Lock (see 10.1.38)

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 considered to 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 9.2.10) is encoded, or the air interface has its own means of providing this information, it can be used to assess in there is sufficient memory for the command to be fully implemented.

— If one or both **Data-CRC-Indicators** (see 9.2.11) are set, then the CRC-16 shall be applied as defined in 9.2.12.

— If the **Length-Of-Encoded-Data** (see 9.2.10) 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 9.2.17).

If the compound argument **Multiple-Records-Constructs** is used in the command, then the **Object-Identifier** structure as define in 11.5 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.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

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 10.1.5)
Application-Defined-Record-Capacity (see 10.1.6)
Identifier-Of-My-Parent (see 10.1.17)
Lock-Directory-Entry (see 10.1.24)
Lock-Record-Preamble (see 10.1.26)
Number-In-Data-Element-List (see 10.1.36)
Record-Memory-Capacity (see 10.1.46)
Record-Type-Classification (see 10.1.48)
Update-Multiple-Records-Directory (see 10.1.58)

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 10.2 and below)
Execution-Code (see 10.3)
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
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

8.2.4.2 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

Documents
Document Sharing Hub

© ISO/IEC 2013 – All rights reserved

Full document details: BS ISO/IEC 15962:2013

— 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**.

8.2.4.3 Guidance for appending data

The procedure described in 8.2.3.3 applies to this command.

8.2.5 Write-EPC-UII

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 an ISO/IEC 18000-6C tag).

8.2.5.1 Process requirements

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

Access-Password (see 10.1.1)
EPC-Code (see 10.1.15)
Memory-Bank-Lock (see 10.1.30)
NSI-Bits (see 10.1.35)

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 needs to ensure that bytes representing part of the older code are removed.

NOTE 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 10.2 and below)
Execution-Code (see 10.3)

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

8.2.5.2 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**.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.2.6 Write-Password-Segmented-Memory-Tag

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.

8.2.6.1 Process requirements

The following process arguments shall be supported:

Password (see 10.1.40)
Password-Type (see 10.1.41)

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)

The **Completion-Codes** for this command are:

0 No-Error
8 Singulation-Id-Not-Found
26 Password-Not-Written
255 Execution-Error

8.2.6.2 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

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 ISO/IEC 18000-6 Type D RFID 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 ISO/IEC 18000-6 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

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.2.7.1 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.

AFI (see 10.1.3)
DSFID-Lock (see 10.1.12)
Item-Related-DSFID (see 10.1.20)
Lock-UII-Segment-Arguments (see 10.1.27)
Memory-Segment (see 10.1.31)
Tag-Data-Profile-ID-Table (see 10.1.53)
UII-DSFID (see 10.1.56)

The ISO/IEC 18000-6 Type D does not require to implement 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 amd lemgth 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 compliant 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 10.1.7)
Compact-Parameter (see 10.1.9)
Object-Lock (see 10.1.38)

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 considered to 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.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

If the compound argument **Multiple-Records-Constructs** is used in the command, then the **Object-Identifier** structure as define in 11.5 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 10.1.5)
Application-Defined-Record-Capacity (see 10.1.6)
Identifier-Of-My-Parent (see 10.1.17)
Lock-Directory-Entry (see 10.1.24)
Lock-Record-Preamble (see 10.1.26)
Number-In-Data-Element-List (see 10.1.36)
Record-Memory-Capacity (see 10.1.46)
Record-Type-Classification (see 10.1.48)
Update-Multiple-Records-Directory (see 10.1.58)

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 10.2 and below)
Execution-Code (see 10.3)

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
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

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.2.7.2 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 an ISO/IEC 18000-6 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**.

8.2.7.3 Guidance for appending data

The procedure described in 8.2.3.3 applies to this command.

8.2.8 Write-Monomorphic-UII

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 an 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 treated as a warning about constructing the command. The process continues because the **Object-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**.

8.2.8.1 Process requirements

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

Access-Password (see 10.1.1)
AFI (see 10.1.3)
AFI-Lock (see 10.1.4)
Compact-Parameter (see 10.1.9)
Lock-UII-Segment-Arguments (see 10.1.27)
Memory-Bank-Lock (see10.1.30)
Memory-Type (see 10.1.32)
Object-Lock (see 10.1.38)

The command argument **Memory-Type** (see 10.1.32) should match the type of RFID tag being encoded. If there is a match, then one of the following processes is applied.

For an **ISO/IEC 18000-6 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.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

— 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 needs to 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 an **ISO/IEC 18000-6 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 need to 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 needs to check if an item related segment is already encoded. If the encoding can still be encoded in the UII segment without re-wrting the item-related segment, the encoded bytes are transferred to the interrogator. If the UII segment needs to be increased in size and none of the item-related bytes are locked, the entire memory can be re-written, otherwise the process has to 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 otherwies 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 needs to 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.

Documents
Document Sharing Hub

© ISO/IEC 2013 – All rights reserved

Full document details: BS ISO/IEC 15962:2013

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)

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

8.2.8.2 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

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 am extended set of encoding instructions. These include providing the Data Processor with information about the memory capacity and length of encoded data where this cammot 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. Other signal are also included and the feature can be expanded in the future.

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

8.2.9.1 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 9.2.15)
Data-CRC-Indicator (see 9.2.11)
DSFID-Lock (see 10.1.12)
DSFID-Pad-Bytes (see 9.2.17)
Full-Function-Sensor-Indicator (see 9.2.16)
Length-Of-Encoded-Data (see 9.2.10)
Memory-Capacity (see 9.2.10)
Memory-Length-Encoding (see 9.2.10)
Simple-Sensor-Indicator (see 9.2.14)

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

The Extended DSFID is encodedin slighly 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 need to be reserved for this purpose.

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)

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

8.2.9.2 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 tramsferred with a single write transactiom.

— 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

Encoding multiple records needs to 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

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.2.10.1 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:

Access-Password (see 10.1.1)
Directory-Length-EBV8-Indicator (see 10.1.13)
Lock-Multiple-Records-Header (see 10.1.25)
Multiple-Records-Features-Indicator (see 10.1.34)
Pointer-To-Multiple-Records-Directory (see 10.1.43)
Sector-Identifier (see10.1.49)
Singulation-Id (see 9.2.1)

For RFID tags with segmented memory, the multiple records shall only be encoded in the user memory. Specifically, for ISO/IEC 18000-6 Type C this is Memory Bank 11; and for ISO/IEC 18000-6 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, ISO/IEC 18000-6 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
255 Execution-Error

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.2.10.2 Conformance requirements

8.2.10.2.1 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.

8.2.10.2.2 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

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)

8.3.1.1 Process requirements

To correctly implement this command in the Data Processor, the following points need to be taken into account:

— 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.

— 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.

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)
Object-Identifier (see 6.3)

The **Completion-Codes** for this command are:

0 No-Error
8 Singulation-Id-Not-Found
255 Execution-Error

8.3.1.2 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

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.

8.3.2.1 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 10.2 and below)
Execution-Code (see 10.3)
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

8.3.2.2 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

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)

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.3.3.1 Process requirements

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

Check-Duplicate (see 10.1.8)
Max-App-Length (see 10.1.28)
Read-Type (see 10.1.45)

The following arguments shall be supported for the response:

Compact-Parameter (see 10.1.9)
Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)
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

8.3.3.2 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 similar to the **Read-Object-Identifiers** command could be used.

— If the **Max-App-Length** argument is invoked, the interrogator and decoder only need to 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

The **Inventory-ISO-UIImemory** command is intended to return the contents of the UII memory from a number of 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.

8.3.4.1 Process requirements

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

Additional-App-Bits (see 10.1.2)
AFI (see 10.1.3)
DSFID (see 10.1.11)

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

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 **Data-Set** 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 10.1.9)
Completion-Code (see 10.2 and below)
DSFID (see 10.1.11)
Execution-Code (see 10.3)
Lock-Status (see list in Clause 10)
Object (see 6.4)
Object-Identifier (see 6.3)
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

8.3.4.2 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

The **Inventory-EPC-UIImemory** command is intended to return the contents of the UII memory from a number of 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.

8.3.5.1 Process requirements

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

Length-of-Mask (see 10.1.23)
Pointer (see 10.1.42)
Tag-Mask (see 10.1.54)

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

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
EPC-Code (see 10.1.15)
Execution-Code (see 10.3)
Protocol-Control-Word (see list in Clause 10)

The **Completion-Codes** for this command are:

0 No-Error
255 Execution-Error

8.3.5.2 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

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.

8.3.6.1 Process requirements

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

Access-Password (see 10.1.1)
Memory-Bank (see 10.1.29)
Word-Count (see 10.1.59)
Word-Pointer (see 10.1.60)

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)
Read-Data (see list in Clause 10)

The **Completion-Codes** for this command are:

0 No-Error
25 Password-Mismatch
255 Execution-Error

8.3.6.2 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 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.3.7 Read-Segments-6TypeD-Tag

The **Read-Segments-6TypeD-Tag** command instructs the interrogator to read all the data from an ISO/IEC 18000-6 Type D tag and instructs the Data Processor to provide the decoded data according to the arguments in the command. This command can be used to show the content of each segment as a raw byte string, or to provide more detailed analysis of the content of one or more segments.

This command is not used to read **Objects** or **Object-Identifiers** from a Multiple Record encoded on an RFID tag compliant with ISO/IEC 18000-6 Type D. The functionality is incorporated in the **Read-Multiple-Records** command (see 8.3.8)

8.3.7.1 Process requirements

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

Segment-Read-Type (see 10.1.50)

The ISO/IEC 18000-6 type D air protocol delivers the entire data payload of the RFID. Therefore, the command argument codes specify the processes that the Data Processor undertakes, as defined in 10.1.50.

The following arguments shall be supported for the response, and be included depending on the command arguments:

Compact-Parameter (see 10.1.9)
Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)
Item-Related-DSFID (see 10.1.20)
Item-Related-Segment-Map (see 10.1.21)
Lock-Status (see list in Clause 10)
Object (see 6.4)
Object-Identifier (see 6.3)
Simple-Sensor-Data-Block (see 10.1.51)
TID-Segment-Map (see 10.1.55)
UII-Segment-Map (see 10.1.57)

The **Completion-Codes** for this command are:

0 No-Error
19 Read-Incomplete
255 Execution-Error

8.3.7.2 Conformance requirements

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

— The decoder supports decoding from to a segmented memory tag compliant with ISO/IEC 18000-6 Type D.

— The decoder supports the processing into segments and further into the constituent parts of the UII segment and the Item-related segment

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

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.3.8 Read-Multiple-Records

The **Read-Multiple-Records** command instructs the interrogator to read various logically structured components from an RFID tag configured to encode multiple records. This includes performing relevant functions defined in 8.3.1 to read a set of one or more **Object-Identifiers**; and performing the relevant functions defined in 8.3.3 to read data **Objects** from an individual record.

The command is applied to memory bank 11 of the ISO/IEC 18000-6 Type C tag, and to the Item-related data segment of the ISO/IEC 18000-6 Type D tag. Only one RFID tag shall be programmed per command to ensure that the reading process is robust.

The command is supported by the following compound argument defined in ISO/IEC 15961-1, for which the additional inherent **Read-Objects** argument is also defined.

8.3.8.1 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:

Check-Duplicate (see 10.1.8)
Max-App-Length (see 10.1.28)
Object-Identifier (see 6.3)
Read-Record-Type (see 10.1.44)
Singulation-Id (see 9.2.1)

The **Read-Record-Type** codes 3 to 9 (as defined in 10.1.44) require the use of one or more **Object-Identifiers** in the command for the Data Processor to invoke the relevant processes. Three very specific formats of **Object-Identifier** are applied to multiple records:

— For a multiple record that is not part of a hierarchy, the structure is: 1.0.15961.401.{Data-Format}.{sector identifier}.{record type}.{instance-of}.{Relative-OID of data element}

— For a multiple record that is part of a hierarchy, but not a data element list, the structure is: **1.0.15961.402.{Data-Format}.{sector identifier}.{record type}.{hierarchical id}.{Relative-OID of data element}**

— For a multiple record that is a data element list, the structure is: 1.0.15961.403.{Data-Format}.{sector identifier}.{record type}.{hierarchical id}.{Relative-OID of data element}

NOTE This object identifier calls for response to include all the list element number.

The first two **Object-Identifier** structures apply to **Read-Record-Type** codes 3 to 8. The third listed **Object-Identifier** structure only applies to **Read-Record-Type** codes 3, 4 and 9.

If **Read-Multiple-Records-Header** (**Read-Record-Type** code 0) is selected, the Data Processor returns the interpretation of the MR-header in the **Multiple-Records-Header-Structure** argument. The response contains the following compound arguments, for which individual arguments are listed below:

— **DSFID-Constructs** with the following inherent arguments:

Access-Method (see Clause 11)

Data-Format (see 9.2.5)

— **Ext-DSFID-Constructs** with the following inherent arguments:

Battery-Assist-Indicator (see 9.2.15)
Data-CRC-Indicator (see 9.2.11)
DSFID-Pad-Bytes (see 9.2.17)
Full-Function-Sensor-Indicator (see 9.2.16)

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

Length-Of-Encoded-Data (see 9.2.10)
Memory-Capacity (see 9.2.10)
Memory-Length-Encoding (see 9.2.10)
Simple-Sensor-Indicator (see 9.2.14)

The following arguments are also in the **Multiple-Records-Header-Structure** argument:

Multiple-Records-Directory-Length (see 10.1.33)
Multiple-Records-Features-Indicator (see 10.1.34)
Number-Of-Records (see 10.1.37)
Pointer-To-Multiple-Records-Directory (see 10.1.43)
Sector-Identifier (see 10.1.49)

If **Read-Multiple-Records-Header-Plus-1st-Preamble** (**Read-Record-Type** code 1) is selected, the Data Processor returns the interpretation of the MR-header in the **Multiple-Records-Header-Structure** argument and the interpretation of the first record's preamble in the **Multiple-Records-Preamble-Structure** argument.

The response is a combination of the responses for **Read-Record-Type** code 0 and 3). It contains the specific responses for the compound arguments **DSFID-Constructs**, **Ext-DSFID-Constructs** and **Multiple-Records-Header-Structure** that are all listed above, and the specific responses for the compound argument **Multiple-Records-Header-Structure** that are listed for the **Read-Preamble-Specific-Multiple-Record** (below).

If **Read-Multiple-Records-Directory** (**Read-Record-Type** code 2) is selected, the Data Processor returns the interpretation of the multiple records directory in the **Multiple-Records-Directory-Structure** argument. The response contains the following compound arguments, for which individual arguments are listed below:

— **DSFID-Constructs** with the inherent arguments as defined for the **Read-Multiple-Records-Header**

— **Ext-DSFID-Constructs** with the inherent arguments as defined for the **Read-Multiple-Records-Header**

The following arguments are also in the **Multiple-Records-Directory-Structure** argument:

Hierarchical-Identifier-Arc (see 10.1.16)
Identifier-Of-My-Parent (see 10.1.17)
Instance-Of-Arc (see 10.1.19)
Record-Type-Arc (see 10.1.47)
Record-Type-Classification (see 10.1.48)
Sector-Identifier (see 10.1.49)
Start-Address-Of-Record (see 10.1.52)

If **Read-Preamble-Specific-Multiple-Record** (**Read-Record-Type** code 3) is selected, the command shall include a single **Object-Identifier** in the **Read-Objects List** that is certainly defined down to the record type arc, and the instance-of arc (if applicable) or the hierarchical id arc (if applicable). he Data Processor returns the interpretation of the record's preamble in the **Multiple-Records-Preamble-Structure** argument. The response contains the following compound arguments, for which individual arguments are listed below:

— **DSFID-Constructs** with the inherent arguments as defined for the **Read-Multiple-Records-Header**

— **Ext-DSFID-Constructs** with the inherent arguments as defined for the **Read-Multiple-Records-Header**

The following arguments are also in the **Multiple-Records-Preamble-Structure** argument:

Data-Length-Of-Record (see 10.1.10)
Encoded-Memory-Capacity (see 10.1.13)
Hierarchical-Identifier-Arc (see 10.1.16)
Identifier-Of-My-Parent (see 10.1.17)
Instance-Of-Arc (see 10.1.19)
Record-Type-Arc (see 10.1.47)

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

Record-Type-Classification (see 10.1.48)
Sector-Identifier (see 10.1.49)
Start-Address-Of-Record (see 10.1.52)

The **Read-All-Record-OIDs-Specific-Record-Type** (**Read-Record-Type** code 4) is used for identifying a series of history records of the same type or a set of records in a hierarchy of the same type. The command shall include a single **Object-Identifier** in the **Read-Objects List** that is only defined down to the record type arc. The Data Processor returns the list of **Object-Identifiers** one layer lower in the **Read-OIDs-Response-List**, i.e. either with the set of instance-of arcs or with the set of hierarchical id arcs. To achieve this, the Data Processor needs to parse the entire Multiple Records directory; or if this is not encoded, the Data Processor read and parse the preamble of each Multiple Record. During this procedure, the Data Processor shall only retain and report on records whose record type arc matches that in the **Object-Identifier** in the command.

If **Read-OIDs-Specific-Multiple-Record** (**Read-Record-Type** code 5) is selected, the command shall include a single **Object-Identifier** in the **Read-Objects List** that is certainly defined down to the record type arc, and the instance-of arc (if applicable) or the hierarchical id arc (if applicable). The Data Processor returns the list of **Object-Identifiers** encoded within the record in the **Read-OIDs-Response-List**. This does not apply to Data Element lists.

If **Read-All-Objects-Specific-Multiple-Record** (**Read-Record-Type** code 6) is selected, the command shall include a single **Object-Identifier** in the **Read-Objects List** that is certainly defined down to the record type arc, and the instance-of arc (if applicable) or the hierarchical id arc (if applicable). The Data Processor returns the list of **Object-Identifiers** and **Objects** encoded within the record in the **Read-Objects-Response-List**. This does not apply to Data Element lists.

If **Read-Multiple-Objects-Specific-Multiple-Record** (**Read-Record-Type** code 7) is selected, the command shall include the nominated **Object-Identifiers** that are defined down to the specific data element in the **Read-Objects List**. The Data Processor returns the list of nominated **Object-Identifiers** and **Objects** encoded within the record in the **Read-Objects-Response-List**. This does not apply to Data Element lists.

If **Read-1st-Objects-Specific-Multiple-Record** (**Read-Record-Type** code 8) is selected, the command shall include the nominated **Object-Identifier(s)** that are defined down to the specific data element(s) in the **Read-Objects List**. The Data Processor returns the list of nominated **Object-Identifiers** and **Objects** encoded up to the **Max-App-Length** within the record in the **Read-Objects-Response-List**. This does not apply to Data Element lists.

If **Read-Data-Element-List-Specific-Multiple-Record** (**Read-Record-Type** code 9) is selected, the command shall include a single **Object-Identifier** in the **Read-Objects List** that is defined down to the data element. The Data Processor first checks that the record is a Data Element list. If so, it uses the rules of the **Access-Method** to re-construct the **Object-Identifier** down to the list element number as encoded in the data element list and returns this and associated **Objects** encoded within the record in the **Read-Objects-Response-List**. The first four arcs of the **root-OID** for the application command and response is **1.0.15961.403**. If this is not in the command, than the Data Processor shall not implement the command.

The **Completion-Codes** for this command will vary based on the **Read-Record-Type** code value set in the command. The complete list is:

0 No-Error
8 Singulation-Id-Not-Found
10 Duplicate-Object
13 Object-Identifier-Not-found
15 Object-Not-Read
35 Monomorphic-UII-OID-Mismatch
48 Multiple-Records-Directory-Not-Present
255 Execution-Error

© ISO/IEC 2013 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013

8.3.8.2 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**.

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

— All the **Read-Record-Types** shall be supported.

— For **Read-Record-Type** codes 5, 6, 7,and 8, if the **Check-Duplicate** argument is set to TRUE, then the entire record shall be read to establish that the subject **Object-Identifier** is not duplicated.

— For **Read-Record-Type** codes 4 and 9, if the **Check-Duplicate** argument is set to TRUE, then the argument shall be ignored, because the intention is to identify multiple occurrences of the **Object-Identifier** in the command and return any additional arc.

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

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

The decoder needs to support the following, depending on the requirements of the application:

— At least one of the **Access-Methods** defined in this International Standard

— The decoding from a segmented memory tag. compliant with ISO/IEC 18000-6 Type C.

— The decoder supports decoding from to a segmented memory tag compliant with ISO/IEC 18000-6 Type D.

8.4 Application commands and responses– other

8.4.1 Inventory-Tags

The **Inventory-Tags** command is intended to read a set of **Singulation-Ids** from RFID tags that have a particular **AFI**. It is only applicable where an air interface command supports an inventory process using the **AFI** as a named argument and, generally, where a unique chip identifier is used in the arbitration process.

8.4.1.1 Process requirements

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

Identify-Method (see 10.1.18)
Number-Of-Tags (see 10.1.18)

The following arguments shall be supported for the response:

Completion-Code (see 10.2 and below)
Execution-Code (see 10.3)
Identities (see list in Clause 10)
Number-of-Tags-Found (see list in Clause 10)

Typically, the response should include the argument Identities, which is a list of **Singulation-Ids** each of which has met the selection criteria.

© ISO/IEC 2013 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 15962:2013