

Australian Standard[®]

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



This Australian Standard® was prepared by Committee IT-034, Automatic Identification and Data Capture Techniques. It was approved on behalf of the Council of Standards Australia on 15 November 2006.

This Standard was published on 27 December 2006.

The following are represented on Committee IT-034:

- Australian Custom Service
 - Australian Data Capture Association
 - Australian Electrical and Electronic Manufacturers Association
 - Australian Retailers Association
 - Australian Veterinary Association
 - Department of Communications, Information Technology and the Arts
 - Department of Defence
 - Department of Primary Industries, Vic
 - RFID Association of Australia
 - The University of Adelaide
 - GS1
-

This Standard was issued in draft form for comment as DR 06534.

Standards Australia wishes to acknowledge the participation of the expert individuals that contributed to the development of this Standard through their representation on the Committee and through public comment period.

Keeping Standards up-to-date

Australian Standards® are living documents that reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued.

Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments that may have been published since the Standard was published.

Detailed information about Australian Standards, drafts, amendments and new projects can be found by visiting **www.standards.org.au**

Standards Australia welcomes suggestions for improvements, and encourages readers to notify us immediately of any apparent inaccuracies or ambiguities. Contact us via email at **mail@standards.org.au**, or write to Standards Australia, GPO Box 476, Sydney, NSW 2001.

Australian Standard[®]

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

First published as AS ISO/IEC 15962—2006.

COPYRIGHT

© Standards Australia

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

Published by Standards Australia, GPO Box 476, Sydney, NSW 2001, Australia

ISBN 0 7337 7930 1

PREFACE

This Standard was prepared by the Standards Australia Committee IT-034, Automatic Identification and Data Capture Techniques.

The objective of this Standard is to provide the overall process and the methodologies developed to format the application data into a structure to store on an RF tag and to define the encoded structure of RF object identifiers and specify formatting rules for data.

This Standard is identical with, and has been reproduced from ISO/IEC 15962:2004, *Information technology—Radio frequency identification (RFID) for item management—Data protocol: data encoding rules and logical memory functions*.

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

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover.
- (b) In the source text ‘this International Standard’ should read ‘this Australian Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australia Standard</i>	
ISO		AS ISO/IEC	
15961	Information technology—Radio frequency identification (RFID) for item management—Data protocol: application interface	15961	Information technology—Radio frequency identification (RFID) for item management—Data protocol: application interface
18000	Information technology—Radio frequency identification for item management	18000	Information technology—Radio frequency identification for item management
19762	Information technology—Automatic identification and data capture (AIDC) techniques—Harmonized vocabulary	19762	Information technology—Automatic identification and data capture (AIDC) techniques—Harmonized vocabulary –
19762-1	Part 1: General terms relating to AIDC	19762.1	Part 1: General terms relating to AIDC
19762-3	Part 3: Radio frequency identification (RFID)	19762.3	Part 3: Radio frequency identification (RFID)

Only international references that have been adopted as Australian or Australian/New Zealand Standards have been listed.

The terms ‘normative’ and ‘informative’ are used to define the application of the annex to which they apply. A normative annex is an integral part of a standard, whereas an informative annex is only for information and guidance.

CONTENTS

Page

1	Scope.....	1
2	Normative references	1
3	Terms, definitions and abbreviated terms.....	2
3.1	Terms and definitions.....	2
3.2	Abbreviated terms.....	4
4	Protocol model	4
4.1	Overview	4
4.2	Layered protocol	5
4.3	Functional processes	6
5	Data structure	8
5.1.1	The 8-bit byte	8
5.1.2	N-bit encoding	8
6	Data Protocol Processor and the application interface	8
6.1	Processing transfers from ISO/IEC 15961	9
6.2	Universal Types.....	9
6.3	Length encoding and decoding.....	10
6.4	Decoding of Type values.....	10
6.4.1	Decoding of a BOOLEAN value	10
6.4.2	Decoding an INTEGER value	10
6.4.3	Decoding an OBJECT IDENTIFIER value.....	11
6.4.4	Decoding the OCTET STRING value	11
6.4.5	Decoding the SEQUENCE or SEQUENCE OF value.....	11
7	Data Protocol Processor and the air interface.....	12
7.1	Air interface services	12
7.2	Defining the system information	13
7.2.1	Tag Identifier.....	13
7.2.2	Physical block size	14
7.2.3	Number of blocks.....	14
7.2.4	Application Family Identifier	14
7.2.5	Storage Format.....	14
7.3	Configuring the Logical Memory.....	15
7.3.1	Non- directory structure of the Logical Memory	15
7.3.2	Directory structure of the Logical Memory	16
7.3.3	Self mapping RF tags	16
8	Data flows and processes	17
8.1	Application data	17
8.1.1	Processing data transferred from ISO/IEC 15961	17
8.1.2	Processing example	18
8.2	Data object processing.....	19
8.2.1	Compaction process.....	19
8.2.2	Compaction Schemes.....	20
8.2.3	Compaction Type codes.....	21
8.2.4	Encoding the length of the compacted object.....	21
8.2.5	Example of encoding for the Logical Memory after compaction.....	21
8.3	Data formatting.....	22
8.3.1	Data Formatter Functions	22

8.3.2	Formatting the objectId	23
8.3.3	The Precursor for dataFormat not equal 2	24
8.3.4	The Precursor for the root-OID for dataFormat = 2	24
8.3.5	Encoding the RELATIVE-OID	24
8.3.6	Encoding the OBJECT IDENTIFIER	26
8.3.7	Encoding the root-OID for dataFormat = rootOidEncoded (2).....	26
8.3.8	Encoding the object and its length.....	27
8.3.9	The offset byte	27
8.3.10	The Precursor expansion byte	27
8.3.11	The directory structure	27
8.3.12	Addressing from the directory	28
8.3.13	Structures of Logical Memory	28
8.4	Decoding the Logical Memory	28
8.4.1	Overall decode strategy	28
8.4.2	Decoding the storageFormat	28
8.4.3	Decoding the Precursor	29
8.4.4	Decoding the leading byte(s) of the encoded objectId.....	30
9	The Command / Response unit.....	31
9.1	Commands	31
9.1.1	Configure Application Family Identifier command	31
9.1.2	Configure Storage Format command	31
9.1.3	Inventory Tags command	32
9.1.4	Add Single Object command	32
9.1.5	Delete Object command	32
9.1.6	Modify Object command	32
9.1.7	Read Single Object command.....	33
9.1.8	Read ObjectIds command	33
9.1.9	Read All Objects command.....	34
9.1.10	Read Logical Memory Map command	34
9.1.11	Inventory And Read Objects command	34
9.1.12	Erase Memory command	35
9.1.13	Get Application-based System Information command	35
9.1.14	Add Multiple Objects command	35
9.1.15	Read Multiple Objects command	35
9.1.16	Read First Object command.....	36
9.2	Processing arguments	36
9.2.1	afiLock	36
9.2.2	avoidDuplicate	36
9.2.3	checkDuplicate	37
9.2.4	identifyMethod and numberOfTags.....	37
9.2.5	lockStatus	38
9.2.6	maxAppLength	38
9.2.7	objectLock.....	38
9.2.8	storageFormatLock	38
9.3	Completion codes	39
9.4	Execution codes	40
10	Communications between the Data Protocol and the RF tag.....	41
11	Compliance, or classes of compliance, to this International Standard.....	41
11.1	Compliance of the Data Protocol Processor	41
11.2	Compliance of the Tag Driver	41
Annex A	(normative) Pro Forma Description for the Tag Driver	42
A.1	Defining the tagId	42
A.2	System information : applicationFamilyId	42
A.3	System information : storageFormat	42
A.4	Memory-related parameters	42
A.5	Support for commands	42

Annex B (normative) ISO/IEC 18000 Tag Driver Descriptions	44
B.1 Tag Driver for ISO/IEC 18000-2: Parameters for Air Interface Communications below 135 kHz.....	44
B.2 Tag Driver for Mode 1 of ISO/IEC 18000-3: Parameters for Air Interface Communications at 13,56 MHz.....	45
B.3 Tag Driver for Mode 2 of ISO/IEC 18000-3: Parameters for Air Interface Communications at 13,56 MHz.....	46
B.4 Tag Driver for ISO/IEC 18000-4: Parameters for Air Interface Communications at 2,45 GHz - Mode 1	47
B.5 Tag Driver for ISO/IEC 18000-4: Parameters for Air Interface Communications at 2,45 GHz - Mode 2	48
B.6 Tag Driver for ISO/IEC 18000-6: Parameters for Air Interface Communications at 860 MHz to 960 MHz	49
Annex C (normative) Data Compaction Schemes.....	51
C.1 Integer compaction	51
C.2 Numeric compaction.....	51
C.3 5-bit compaction	52
C.4 6-bit compaction	52
C.5 7-bit compaction	53
C.6 Octet encodation.....	54
Annex D (normative) ISO/IEC 646 Characters Supported by the Compaction Schemes	55
Annex E (informative) Encoding Example	59
E.1 Starting position.....	59
E.2 The initial state of the entry for the Logical Memory	59
E.3 The Logical Memory after data compaction.....	59
E.4 The Logical Memory after formatting with a noDirectory accessMethod	60
Annex F (informative) Logical Memory Structures	62
F.1 Notation.....	62
F.2 Non-directory structured Logical Memory with root-OID implicitly encoded.....	62
F.3 Non-directory structured Logical Memory with root-OID explicitly encoded.....	63
F.4 Directory structured Logical Memory with root-OID implicitly encoded	63
F.5 Directory structured Logical Memory with root-OID explicitly encoded.....	63

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 RF tag is invisible and accessible between the RF tag and the interrogator only by the use of the appropriate air interface protocol, as specified in the different ISO/IEC 18000 parts. The transfer of data between the application and the interrogator in open systems requires data to be presented in a consistent manner on any RF tag that is part of that open system. Functional commands from the application and responses from the interrogator also require being processed in a standard way. This is not only to allow equipment to be interoperable, but in the special case of data carrier, for the data to be encoded on the RF 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 RF tag must be formatted in such a way as to be reliably read at the point of use if the RF tag is to fulfil its basic objective. The integrity of this is achieved through the use of a data protocol as specified in ISO/IEC 15961 and this International Standard.

Manufacturers of radio frequency identification equipment (interrogators, RF tags, etc) and the users of RFID technology require a publicly available data protocol for RFID for item management. ISO/IEC 15961 and this International Standard specify this data protocol, which is independent of any of the air interface standards defined in the various parts of ISO/IEC 18000. As such, the data protocol is a consistent component in the RFID system that may independently evolve to include additional air interface protocols

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

AUSTRALIAN STANDARD

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 that International Standard. The encodation is in a Logical Memory as a software analogue of the physical memory of the RF 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;
- defines other communication to the application.

NOTE 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.

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 8824-1, *Information technology — Abstract Syntax Notation One (ASN.1) — Specification of basic notation* (equivalent to ITU-T Recommendation X.680)

ISO/IEC 8825-1, *Information technology — ASN.1 encoding rules — Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)* (equivalent to ITU-T Recommendation X.690)

ISO/IEC 15961:2004, *Information technology — Radio frequency identification (RFID) for item management — Data protocol: application interface*

ISO/IEC 18000 (all parts), *Information technology — Radio frequency identification for item management*

ISO/IEC 19762-1, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 1: General terms for AIDC*¹⁾

ISO/IEC 19762-3, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 3: Radio frequency identification (RFID)*¹⁾

3 Terms, definitions and abbreviated terms

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 Application commands

The instruction issued from the application to the Data Protocol Processor in order to initiate an action or operation with the RF tag(s) via the interrogator.

3.1.2 Application memory

The area of the RF tag available for storing data written to it. Sometimes known as *user memory*.

3.1.3 Arc

A specific branch of an object identifier tree, with new arcs added as required to define a particular object. The top three arcs of all object identifiers compliant with ISO/IEC 9834-1 are defined in Annex A of ISO/IEC 15961.

3.1.4 Block

The minimum number of bytes on an RF tag that can be in a write transaction, or read transaction, across the air interface.

3.1.5 Command / Response Unit

That part of the Data Protocol Processor that processes application commands and sends responses to control encoding, decoding, structuring of the Logical Memory and transfer to the Tag Driver.

1) To be published.

3.1.6 Data compaction

A 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.

3.1.7 Data Compactor

The implementation of the data compaction process defined in this International Standard.

3.1.8 Data Protocol Processor

The implementation of the processes defined in this International Standard, including the Data Compactor, Formatter, Logical Memory, and Command/Response Unit.

3.1.9 elementName

A component of a ReferenceType or enumerated list in ASN.1 Syntax.

3.1.10 Formatter

The implementation of the data formatting process defined in this International Standard.

3.1.11 Logical Memory

A software analogue on the Data Protocol Processor of the Logical Memory Map.

3.1.12 Logical Memory Map

An array of contiguous bytes of memory on the RF tag, representing the application (or user) memory to be used exclusively for the encoding of objects, objectIds, and their associated Precursor on the RF tag. The system information shall be defined by different means or stored in a separate area on the RF tag. This can be achieved by partitioning memory, partly for system information and mainly for the Logical Memory Map purpose.

3.1.13 Object

A well-defined piece of information, definition, or specification which requires a name in order to identify its use in an instance of communication.

3.1.14 Object identifier

A value (distinguishable from all other such values) which is associated with an object.

3.1.15 OBJECT IDENTIFIER type

A simple ASN.1 type whose distinguished values are the set of all object identifiers allocated in accordance with the rules of ISO/IEC 8824-1 (ITU-T X.680).

3.1.16 Precursor

A byte, sometimes a sequence of bytes, within the encodation on the Logical Memory and Logical Memory Map that acts as metadata for the subsequent objectId and object.

3.1.17 RELATIVE-OID type

A 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.

3.1.18 Response

The feedback received by the application from an application command sent to the Data Protocol Processor.

3.1.19 root-OID

That part of a set of OBJECT IDENTIFIERS that has a common first, second, and subsequent arcs. The root-OID is a prefix to a RELATIVE-OID to construct a complete OBJECT IDENTIFIER.

3.1.20 Tag Driver

The implementation of the process to transfer data between the Data Protocol Processor and the RF tag.

3.1.21 Transfer syntax

The abstract syntax and concrete syntax used in the transfer of data between open systems.

NOTE: The term "transfer syntax" is sometimes used to mean encoding rules, and sometimes used to mean the representation of bits in data while in transit.

3.2 Abbreviated terms

BER	Basic Encoding Rules (of ASN.1)
EAN.UCC	EAN International & Uniform Code Council, Inc
IATA	International Air Transport Association
UPU	Universal Postal Union

4 Protocol model

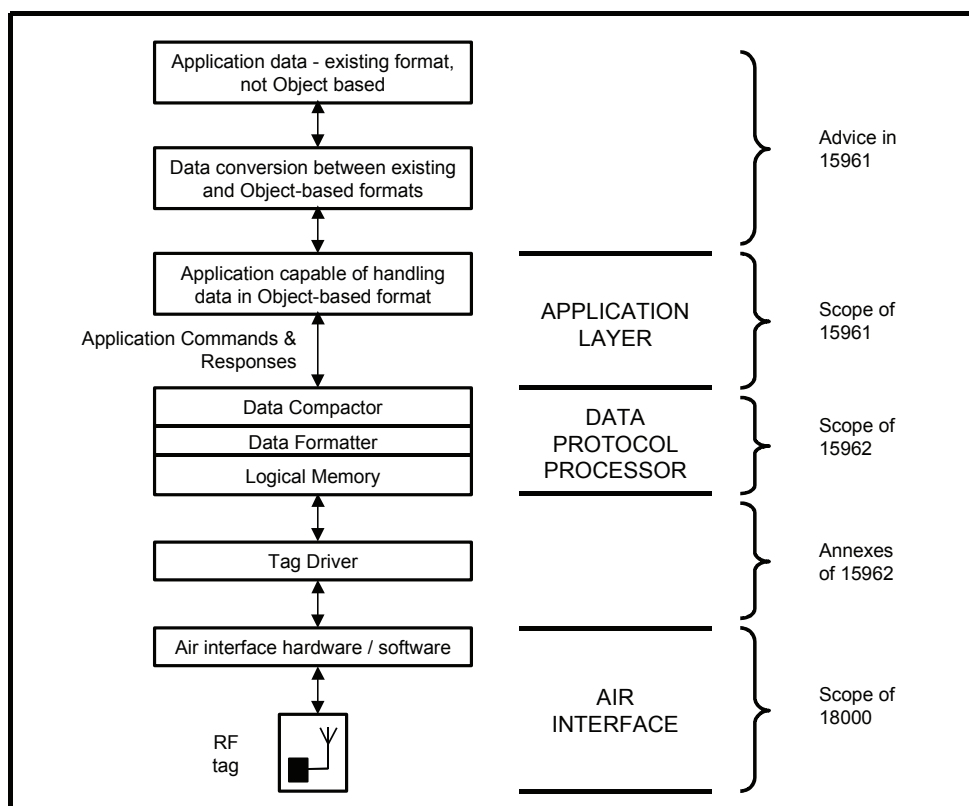


Figure 1 — Schematic of Protocol Layers for an Implementation of RFID for Item Management

4.1 Overview

RFID supports bit encodation in the RF 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 RF tag memory. However, as an RF 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 is to specify a common interface between the application programs and the RF interrogator. The prime function of this International Standard is to specify the common encoding rules and logical memory functions.

RF tags utilise electronic memory that is typically capable of increasing data capacity as new generations of product are introduced. Differences in data capacity of each RF 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.

4.2 Layered protocol

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

The data protocol specified in this International Standard is independent of the different RF tag technologies specified in ISO/IEC 18000, which is concerned with different air interface protocols that function between the interrogator and the RF tag. This independence is achieved by implementing the standards at different levels in the protocol hierarchy. The RFID data protocol defined in this International Standard is primarily concerned with the upper layers as described below:

Application layer - as defined in 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 RF tag.
- This RFID data protocol defines application commands and responses so that application programs can specify what data to transfer to and from the RF tag and to append, update or delete data on the RF tag.
- This RFID data protocol also defines error messages as responses to the application.

The application interface of this RFID data protocol is based on ASN.1, which:

- provides a means of defining the protocol which 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.
- identifies any data object distinctly from all others using object identifiers, even to enable different data formats to be intermixed on the same RF tag.
- defines unambiguous commands and responses, so that they can be intermixed with data on the same wired or wireless network.
- provides the abstract syntax for defining the commands and responses in a structured and consistent and verifiable manner, and provides the transfer syntax that defines the byte stream transferred between the processes of ISO/IEC 15961 and those of this International Standard.
- enables implementation in a variety of computer languages through the use of compilers, alternatively programs can be written from the specification. In either case there is a vital need for the transfer syntax to be fully consistent and compliant to function in open systems where the sender and recipient can be unknown to one another.

Data Protocol Processing - as defined in this International Standard

- The RFID data protocol specifies how data is encoded, compacted and formatted on the RF tag and how this data is retrieved from the RF tag to be meaningful to the application.
- This RFID data protocol provides for a set of schemes that compact the data to make more use of the memory space.
- This RFID data protocol also supports various storage formats to enable efficient use of memory and efficient access procedures.

All these features are described and specified later in this International Standard and its companion standard. Figure 1, and the outline description above, applies to a general process. Different rules may apply to RF tags that are capable of executing commands (see 7.3.3).

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

- on the same platform as the application.
- on a separate platform linked to the application platform e.g. linked via a serial link, LAN or internet connection.
- on an embedded platform e.g. in a bar code printer/RFID encoder, in a bar code/RFID scanner, or a dedicated RFID interrogator.

This RFID data protocol has been designed such that the actual platform on which it is implemented is transparent to the application. It is also independent of the programming language used by the application. If both standards are not implemented, care will need to be taken to maintain the functionality between the two standards. The compliance clauses of both standards address these points in greater detail.

The rules specified in these International Standards create a complete independence between the application and the technology of the air interface and RF tag. The type of RF tag used in an implementation can be changed without requiring the application to change.

4.3 Functional processes

There are various functional processes that need to take place to write data to an RF tag and to read data from it. Figure 2 shows a schematic of an 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.

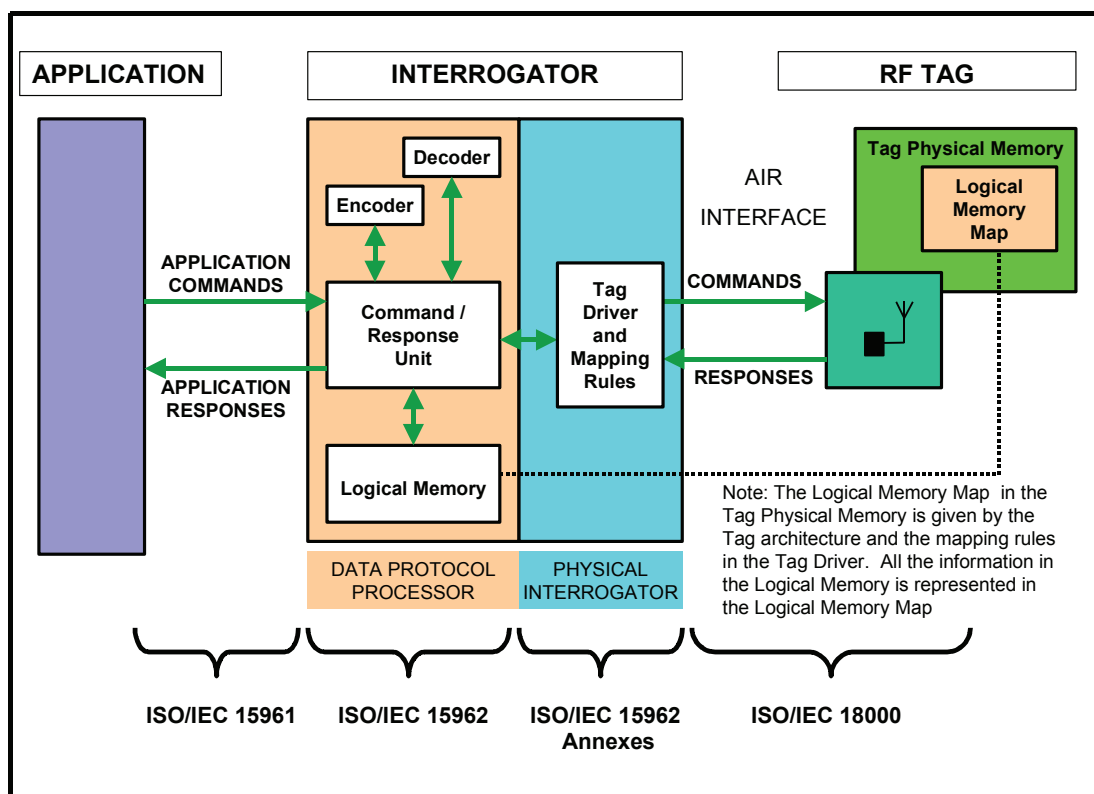


Figure 2 — Logical Functions and Interfaces

Application is the user application database and software.

The data flows between the application and the Data Protocol Processor are formatted according to ISO/IEC 15961 and are uncompact. 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 ISO/IEC 15961 and this International Standard. 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.

Interrogator is the module in which all the basic processing of the data protocol takes place and there is an interface to the RF tag.

Data Protocol 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 Protocol Processor can physically reside anywhere between the application software and the tag driver but shall contain all the components.

Command/Response Unit for receiving the application commands from the application in a format specified in ISO/IEC 15961, acting upon these commands where appropriate and converting to the specific RF 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 Protocol Processor. The information from the particular RF tag is required to set the parameters of the Logical Memory Map (e.g. number of octets, whether a directory is in use, etc) on the RF 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 boundary between the Data Protocol Processor and the Tag Driver.

Logical Memory. This is an array of contiguous octets (or bytes) of memory acting as a common representation of the Logical Memory Map in the user memory of the RF tag to which the object identifiers and data objects are mapped in octets. The Logical Memory takes into account some parameters of the real RF 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 RF tag, but that individual RF 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 RF tags, without changing the application.

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

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

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 RF tag architecture is performed by the Tag Driver.

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

Tag Driver provides two main functions:

- It maps the contents of the Logical Memory to the Logical Memory Map of the RF tag in use.
- It provides facilities that accept the application commands of this data protocol, and converts them to a format that results in calls to command codes supported by the particular RF Tag. For example, an application command **write Data Object {name}** could result in the RF tag related command of write (block #, data).

The description of the tag driver for particular RF 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 RF 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.

RF Tag, although beyond the scope of these International Standards, is shown to complete the flow of data and commands.

Logical Memory Map represents all the data in the Logical Memory of the Data Protocol Processor converted (or mapped) to a location structure determined by the mapping rules in the Tag Driver and the architecture of the RF tag.

5 Data structure

5.1.1 The 8-bit byte

This International Standard supports the encoding of data and data objects aligned as 8-bit bytes (referred to as octets in ISO/IEC 15961 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:

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

5.1.2 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.

6 Data Protocol Processor and the application interface

The application shall provide transfers of commands, which may contain embedded data, to the Data Protocol Processor in the format that is compliant with the transfer syntax specified in ISO/IEC 15961.

NOTE: For the remainder of this International Standard, it will be assumed that the transfer mechanism from the application is fully compliant with the Basic Encoding Rules used in ISO/IEC 15961. Implementations that do not use all the specifications of ISO/IEC 15961 need to ensure that transfers are achieved in a completely reliable manner.

The following subclauses describe, in an overview, how the transfer syntax is presented and processed.

6.1 Processing transfers from ISO/IEC 15961

The transfers across the application interface shall be compliant with the transfer syntax of ISO/IEC 15961. This results in an octet aligned stream that intermixes commands and data on the transfer from the application to the Data Protocol Processor, and intermixes response codes and data in communications in the reverse direction. These transfers require a precisely defined syntax that basically has a structure of three components of a Type, Length and Value (or Content), sometimes known as a **TLV** format.

Two formats are permitted:

- Primitive encoding, that is a simple sequence of TLV elements
- Constructed encoding (e.g. TL, TLV, TLV...), where the first V is replaced by triplets of TLV elements

The Data Protocol of ISO/IEC 15961 and this International Standard uses the simpler Primitive encoding, except where the rules of ISO/IEC 8825-1 require that constructed encoding shall be used. Further details are given in 6.2.1 of ISO/IEC 15961.

The **Type** identifies a code that uniquely distinguishes the context of the associated value. The structure of the Type code is defined in 6.2.2 of ISO/IEC 15961 and has a length of one byte.

The **Length** identifies the length in octets (bytes) of the associated value. The rules for length encoding are defined in 6.2.3 of ISO/IEC 15961.

All the **Value** (contents) used in the Data Protocol standards of ISO/IEC 15961 and this International Standard have been specified to reduce to a Universal Type defined in ISO/IEC 8824-1. These are fully defined in subsequent clauses of this International Standard where their significance is relevant.

6.2 Universal Types

The transfer syntax supports a number of universal types that are fundamental to the syntax; sometimes called "built-in types". Each has been assigned a class tag in ISO/IEC 8824-1 to unambiguously identify each type of data. Universal types are shown in capital (uppercase) letters e.g. **UNIVERSAL**. The Universal Types used in this International Standard, together with their Class Tags, are shown in Table 1. This also shows whether Primitive or Constructed encoding shall be used with this Data Protocol and the resultant Type code (as defined in 6.2.2 of ISO/IEC 15961).

Table 1 — Universal Types Used in this Standard

Universal Type	Class Tag	Primitive/ Constructed	Type Code (binary)
BOOLEAN	1	Primitive	00000001
INTEGER	2	Primitive	00000010
OBJECT IDENTIFIER	6	Primitive	00000110
OCTET STRING	4	Primitive	00000100
RELATIVE-OID (reserved for future commands)	13	Primitive	00001101
SEQUENCE & SEQUENCE OF	16	Constructed	00110000

As the Type code is at the beginning of each TLV sequence, it is possible to use the encoded value to determine, in conjunction with the command and response syntax, how to process the associated value.

6.3 Length encoding and decoding

The rules (defined in Clause 6.2.3 of ISO/IEC 15961) encode the length field (n) into an octet-aligned-bit-field. The rules that follow are an inverse of the ISO/IEC 15961 rules, but based on the byte values of the resultant encoding. Typically one or more bytes defines the length of the following value element.

- a. If the lead byte value is in the range 01_{HEX} to 7F_{HEX}, the length is encoded in this single byte. Convert to a bit string, 0 [////////], strip off the leading zero bit and read the length as a binary value from the remaining 7 bits. The length of the value is <128.
- b. If the lead byte value is in the range 80_{HEX} to FF_{HEX}, the length is encoded in this and subsequent byte(s), and is decoded using the following procedure:
 1. Convert the lead byte to a bit string, 1 [nnnnnnn], strip off the leading 1 bit and read the number of subsequent length bytes as a binary value from the remaining 7 bits. The number of subsequent bytes is <127, because the lead byte 01111111_{BIN} is not permitted by the current rules of ISO/IEC 8825-1.

NOTE: It is extremely unlikely that the value nnnnnnn will be large with respect to this Data Protocol for RFID for item management.

2. Count the number of bytes from step b1 to determine the bytes that define the length encoding.
3. Convert this byte string as a binary value of the length of the associated value.

EXAMPLE:

This example is of a length ≥ 128 bytes

Length bytes	82 01 65 _{HEX}
Step b1	82 _{HEX} = 10000010 0000010 indicates that there are 2 bytes to determine length
Step b2	01 65 _{HEX} determines the length
Step b3	01 65 _{HEX} = 1 01100101 = 357 bytes = length of value

6.4 Decoding of Type values

The value (or content) component shall have its Universal Class determined by its preceding Type code; and its length determined by its preceding length bytes. The Type value can then be decoded using the specific rule for the Type, as defined in the following subclauses. This is the first stage of processing and encoding the application data into the Logical Memory. A number of the Type values are not encoded, for example; command codes and arguments; while others are subject to further processing before being encoded more efficiently in the Logical Memory. The sub-clauses that follow only address the decode rules.

6.4.1 Decoding of a BOOLEAN value

The Boolean value is encoded in a single byte. If the value of this is 00_{HEX}, the Boolean value is FALSE. All other values convert to a Boolean value of TRUE.

6.4.2 Decoding an INTEGER value

The Integer value is encoded in one or more bytes, as determined by the preceding length component.

If bit 8 of the lead byte of the Integer value has the value 0, the Integer value is positive. Strip out the lead bit and convert the remaining bit string as the Integer value. To avoid the possibility of bit 8 of the lead byte being 1, pad octets may be introduced into the encoding process of ISO/IEC 15961. These extend the length of the encoded Integer value but do not affect the decode process.

EXAMPLES:

01000101		= 69	The pad byte is present to avoid the first bit being 1, which would wrongly signal a negative integer
00000000	10000000	= 128	
00000001	00000000	= 256	
01111111	11111111	= 32767	

If bit 8 of the lead byte of the Integer value has the value 1, the Integer value is negative. Conversion takes place on the remaining bits with the least significant bit being in position 0. The first stage is to create a decimal integer value as the sum of the values of 2^n . The second stage takes this as a positive decimal integer from which is subtracted the value 2^p , where p is the position number of the lead bit that identifies this as a negative integer. As an equation, this is:

$$\sum_{n=0}^{p-1} 2^n - 2^p$$

EXAMPLE:

10010110	01000110		
1			indicates -ve
0010110	01000110	=	5702
$2^p = 2^{15} =$			32768
$5702 - 32768 = -27066$			

6.4.3 Decoding an OBJECT IDENTIFIER value

The Object Identifier value is encoded over 2 or more bytes. No decode process is invoked, but there are specific conversion rules that apply to encoding this value onto the Logical Memory, using one of the following two options:

- The Object Identifier is separated into two constituent parts of a root-OID and a RELATIVE-OID, based on the dataFormat (see 8.3).
- If Option A is not possible to invoke, it is encoded as a full OBJECT IDENTIFIER. In this case, the OBJECT IDENTIFIER is encoded as presented by the byte string of the transfer encoding from the application.

6.4.4 Decoding the OCTET STRING value

The Octet String value is encoded over one or more bytes. No decode process is invoked, but there are specific compaction rules (see 8.2) that apply to encoding this value onto the Logical Memory.

6.4.5 Decoding the SEQUENCE or SEQUENCE OF value

For decode purposes, Sequence and Sequence Of are treated in a similar manner. The first Length component (shown as L_s below) of the TLV structure determines the length of the complete Sequence (or Sequence Of).

$$T L_s \{TLV TLV TLV \dots\}$$

Therefore, it is advisable, in the decode process, to keep a reducing count of the number of bytes still remaining.

The first byte after the Length component L_s is the Type code of the first TLV triplet in the Sequence, or Sequence Of. The next byte(s) are the Length component of the first TLV triplet. Decoding this to the rules defined in 6.3 yields the length of the Value component of the first TLV triplet. The Value component,

identified by its Type and Length is decoded by the relevant process defined in 6.4.1 to 6.4.4. The process is repeated until the number of bytes following, and defined by, the Length component L_S of the Sequence, or Sequence Of, value has been processed.

EXAMPLE:

Transfer encoding:

- 30 14 04 07 {ABC1234} 04 06 {widget} 02 01 0C_{HEX}
NOTE: Values in { } are literal printable characters, presented in this way for ease of illustration. In a real transfer these would be encoded as a string of byte values.
- 30 = SEQUENCE or SEQUENCE OF as per Table 1
- 14 = length L_S = 20 bytes of the complete sequence that follows
- 04 = T of TLV triplet = OCTET STRING
- 07 = L of TLV triplet = 7 bytes
- ABC1234 = literal translation of the 7 bytes of the OCTET STRING value
- At this stage, 9 bytes of the overall length L_S have been processed, with 11 bytes remaining
- 04 = T of next TLV triplet = OCTET STRING
- 06 = L of TLV triplet = 6 bytes
- widget = literal translation of the 6 bytes of the OCTET STRING value
- At this stage, a further 8 bytes of the overall length L_S have been processed, with 3 bytes remaining
- 02 = T of next TLV triplet = INTEGER
- 01 = L of TLV triplet = 1 byte
- 0C = 12

7 Data Protocol Processor and the air interface

The Data Protocol Processor receives communications across the air interface, via the Tag Driver. This clause defines the basic requirements that enable the Data Protocol Processor and RF tag to transfer and share information.

7.1 Air interface services

This International standard is open-ended with respect to the fact that new types of RF tag may be added to the ISO/IEC 18000 multi-part standard and leave the Data Protocol Processor unaltered. To achieve this, some basic presumptions are made about the types of RF tag in ISO/IEC 18000.

- Application memory is an integer number of bytes.
- Application memory shall be organised in blocks. These shall be fixed size and be of one or more bytes.
- The individual blocks shall each be accessible by read and/or write.

NOTE: This applies to the basic function, additional features may be used to restrict access to authorised users.

- In addition to the requirements (above) relating to the memory, there shall be a reliable mechanism for writing and reading to and from the application memory.

Each type of RF tag in ISO/IEC 18000 shall provide the following air interface services of the physical characteristics and data capabilities of the RF tag:

1. Provide a byte location addressing mechanism from the beginning to the end of the application memory, starting from byte 0. This shall map to the Logical Memory.
2. Provide a mechanism to address a specific RF tag using a permanent or virtual tag identifier.

3. Provide the block size (in bytes).
4. Provide the value of the number of blocks in the application memory.
5. Optionally support selection RF tag inventory and/or addressing of groups of RF tags according to the Application Family Identification mechanism.
6. Provide a mechanism to write and read the storage format.
7. Return a positive or negative acknowledgement (error code) to read and to write commands.
8. Optionally provide the writing capabilities supported by the RF tag.
9. Optionally provide the locking capabilities supported by the RF tag.
10. Support the ability to query the status (locked or not locked) of the memory blocks.

These air interface services should be provided by the Tag Driver. A description of a generic Tag Driver is provided in Annex A. Details of specific Tag Drivers are provided in Annex B.

7.2 Defining the system information

The systems information is a set of elements that is encoded, or provided by other means, from the RF tag to the Data Protocol Processor when communications are established. The systems information shall consist of the following elements and shall be as defined in the following subclasses:

tagId	up to 256 bytes
physical block size	1 byte
number of blocks	1 or more bytes
applicationFamilyId , which itself consists of:	
applicationFamily	4 bits
applicationSubFamily	4 bits
storageFormat , which itself consists of:	
accessMethod	2 bits
reserved for future use	1 bit
dataFormat	5 bits

NOTE: Those elements shown in bold are in the style of the elementNames of ISO/IEC 15961, and are transferred across the application interface by the commands and responses. This style is used in the remainder of this International Standard.

7.2.1 Tag Identifier

The Tag Identifier (**TagId**) is the means of ensuring reliable communication between the application and the RF tag throughout the transaction process via the Data Protocol Processor. The Tag Identifier shall be up to 256 bytes long. It is communicated across the air interface and the application interface. The format of the Tag Identifier shall be as defined for each Tag Driver (see Annex B) and shall be based on one of the following:

- a. A completely unique ID programmed in the RF tag, as specified in the ISO/IEC 18000 series).
- b. A data related identifier (e.g. like the unique identifier of transport units as specified in ISO 15459-1), that provides for uniqueness within the particular domain of item management or logistics. This requires the data to be read to establish the Tag Id.
- c. A virtual or session ID based on a time slot or other feature managed by the air interface protocol.
- d. Combinations of (b) and (c), e.g. a virtual ID across the air interface, but requiring the data related identifier to be returned as a response.

7.2.2 Physical block size

The physical block size is defined as the number of 8-bit bytes that are capable of being written to, read from, or locked on the particular RF tag. For the purposes of this International Standard, the value is constrained to between 1 and 256 bytes.

NOTE: The block size relates to memory units on the RF tag and not necessarily to any constraint, such as a frame, on the air interface.

The physical block size shall be identified by any reasonable means using the Tag Driver and air interface features and is communicated to the Data Protocol Processor. The physical block size is not communicated across the application interface.

7.2.3 Number of blocks

The number of blocks is defined as the number of physical blocks in the user memory of the particular RF tag.

The number of blocks shall be identified by any reasonable means using the Tag Driver and air interface features and is communicated to the Data Protocol Processor. The number of blocks is not communicated across the application interface.

7.2.4 Application Family Identifier

The Application Family Identifier (**applicationFamilyId**) shall be defined by the application and be a single byte value with the following structure:

bits 8 to 5 shall define the **applicationFamily**
bits 4 to 1 shall define the **applicationSubFamily**

The full list of Application Family Identifiers are defined in ISO/IEC 15961.

The Application Family Identifier shall be used as a low level selection device to separate RF tags with the particular application features or classes of data content, based on the code values specified in ISO/IEC15961. This feature is co-managed with JTC1 SC17, so currently 60 of the byte values are available for RFID for item management.

7.2.5 Storage Format

The storage Format (**storageFormat**) shall be defined by the application and be a single byte value with the following structure:

bits 8 and 7 shall determine the **accessMethod** (see 7.1.2.4 of ISO/IEC 15961)
bit 6 is reserved
bits 5 to 1 shall determine the **dataFormat** (see 7.1.2.5 of ISO/IEC 15961)

Once an **accessMethod** has been specified for an RF tag, all subsequent additional data shall be stored using the same **accessMethod**, whether the **storageFormat** is locked or not locked.

NOTE: An **accessMethod** may be changed from a noDirectory structure to directory structure (see 7.3.2).

Once a **dataFormat** has been specified for an RF tag, all subsequent additional data shall be stored in one of two ways:

- If the new data has the same root-OID implied by the **dataFormat**, or if this is explicitly stored on the RF tag as a root-OID, then only the RELATIVE-OID needs to be encoded.
- Otherwise, the full OBJECT IDENTIFIER shall be encoded.

The size of the Logical Memory shall be defined using the physical block size and number of blocks from the system information (see 7.2). This can be perceived as a matrix with the physical block size determining the X-axis and the number of blocks determining the Y-axis. This matrix concept is used in the remainder of this International Standard. This configuration process creates an empty Logical Memory.

The **accessMethod**, part of the system information, also determines the structure and processing through the Logical Memory (see sub-clauses below).

- Precursor - This is typically 1 byte long and provides a bit-based meta-data structure of the encoded **objectId** and **object**.
- The length of the **objectId**.
- **ObjectId** - This is a formatted **objectId** that makes use of the RELATIVE-OID and **dataFormat** to reduce the encoding, but still require only a simple process to reconstruct the complete OBJECT IDENTIFIER.
- The length of the **object**
- **Object** - This is the compacted **object**.

The Data Protocol Processor needs to support a number of Logical Memories to enable transactions to take place with the required number of RF tags. The Logical Memory of each RF tag is referenced by its tag identifier.

7.3.1 Non- directory structure of the Logical Memory

The diagram shows a B-tree leaf node divided into two horizontal sections:

- Top Section (Orange background):** Contains the first three entries of a set of pointers. Each entry consists of a precursor pointer (P), an object identifier (OID), and a pointer to the object (O). The format is shown as $\langle P \mid OID \mid O \rangle$. The first entry is followed by a right arrow ($>$), and subsequent entries are preceded by left arrows ($<$) and followed by right arrows ($>$). Ellipses indicate more entries.
- Bottom Section (Green background):** Indicated by a large downward-pointing blue arrow from the top section. It contains the text: "Free space is used for additional sets of ($P + OID + O$)".

Legend:

- P = Precursor
- OID = ObjectId
- O = (data) object

AVA
Documents
Document Sharing Hub

7.3.2 Directory structure of the Logical Memory

The **accessMethod = directory** shall determine that the Logical Memory has a two part structure:

- The lower blocks are identical to the non-directory structure.
- The higher blocks contain the directory.

Directory entries shall be stored in the lowest byte address to the highest byte address sequence within a block, but in reverse block sequence starting from the last block. Figure 4 illustrates this.

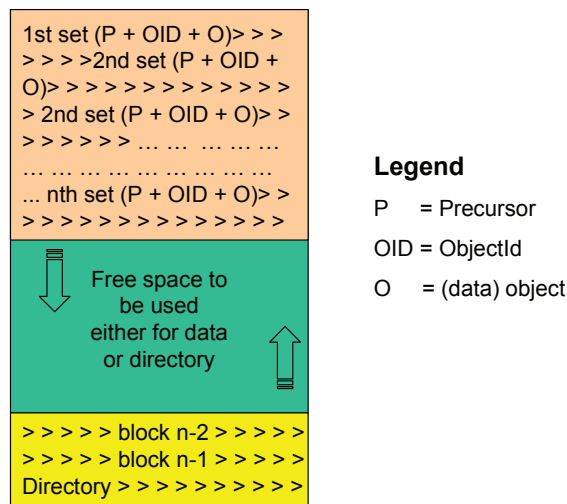


Figure 4 — Logical Memory Schematic - Directory Structure

Structuring the directory in this reverse sequence of blocks leaves unused blocks available either to encode additional data or directory until there are too few blocks remaining to encode new data. This flexibility allows for a small directory (i.e. few **objectIds**) with large data **objects**; or a large directory (i.e. many **objectIds**), each of which could have a small data **object**; or any other combination.

It is possible to begin with a non-directory Logical Memory structure, and some time later convert to a directory structure. This first requires all of the content of the RF tag's application memory to be transferred to the Logical Memory of the Data Protocol Processor. Then the **accessMethod** needs to be changed to indicate that a directory structure is now invoked, and a directory created. Finally the system info needs to be updated on the RF tag, and the content of the directory transferred to the RF tag.

7.3.3 Self mapping RF tags

The **accessMethod = selfMappingTag** indicates that integral processing capability to organise the data mapping is embedded within the RF tag. As such, it shall perform similar functions to that of the Data Formatter (see 8.3) of the Data Protocol Processor.

Data compaction shall be performed on the data **object** if required by a write command argument, and de-compaction shall be performed if the encodation for the **object** on the RF tag signals that it is compacted.

Processing by the Data Formatter shall be by-passed for both read and write processes. As such, the self mapping tag shall process and respond directly to application commands with the exception of data compaction.

Self mapping tags may use other processes than a precursor and directory to structure their encoded **objectIds** and **objects**.

NOTE: At the date of first publication of this International Standard, none of the artefacts demonstrated for ISO/IEC 18000 support such a sophisticated level of processing.