

Australian Standard[®]

**Information technology—Radio
frequency identification (RFID) for item
management—Data protocol:
application interface**



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 06535.

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Standards may also be withdrawn.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:
application interface**

First published as AS ISO/IEC 15961—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 7932 8

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 guidelines on how data shall be presented as objects during the interface between a RFID item management application and the data protocol processor.

This Standard is identical with, and has been reproduced from ISO/IEC 15961:2004, *Information technology—Radio frequency identification (RFID) for item management—Data protocol: application interface*.

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>Australian Standard</i>
ISO	AS ISO/IEC
15962 Information technology—Radio frequency identification (RFID) for item management—Data protocol: data encoding rules and logical memory functions	15962 Information technology—Radio frequency identification (RFID) for item management—Data protocol: data encoding rules and logical memory functions
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	2
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	9
5.1	Notation.....	9
5.2	Structure of the transfer between ISO/IEC 15961 and ISO/IEC 15962	9
6	Abstract and transfer syntax	9
6.1	Abstract syntax	9
6.2	Transfer syntax	14
7	Data flows and processes	19
7.1	Establishing communications between the application and the RF tag.....	19
7.2	Preparing the basic objects	24
7.3	Application system services.....	27
7.4	Data security.....	29
8	Application commands and responses	30
8.1	Final arc values of the command and response modules.....	30
8.2	completionCode (elementName)	31
8.3	executionCode (elementName)	32
8.4	Command-related elementNames	33
8.5	ConfigureAfiModules.....	35
8.6	ConfigureStorageFormatModules.....	37
8.7	InventoryTagsModules	39
8.8	AddSingleObjectModules.....	41
8.9	DeleteObjectModules.....	43
8.10	ModifyObjectModules.....	44
8.11	ReadSingleObjectModules.....	46
8.12	ReadObjectIdsModules	48
8.13	ReadAllObjectsModules	49
8.14	ReadLogicalMemoryMapModules	51
8.15	InventoryAndReadObjectsModules	52
8.16	EraseMemoryModules.....	55
8.17	GetApplication-basedSystemInformationModules	56
8.18	AddMultipleObjectsModules.....	57
8.19	ReadMultipleObjectsModules.....	59
8.20	ReadFirstObjectModules.....	61
8.21	Development commands.....	63

9	Compliance, or classes of compliance, to this standard	63
9.1	Application compliance	63
9.2	Compliance of the Data Protocol Processor	63
9.3	Compliance of the RF tag and RF interrogator	63
Annex A (normative)	First, Second and Third Arcs of Object Identifier Tree.....	65
Annex B (normative)	Code Assignments for ApplicationFamilyId.....	67
Annex C (informative)	Accommodating established data formats	69
Annex D (informative)	Contact Addresses for Managers of Main Application Data Dictionaries	71
D.1	EAN.UCC System	71
D.2	Data Identifiers.....	71
D.3	IATA data elements	71
D.4	UPU data elements	71
Annex E (normative)	Converting alphanumeric Data Identifiers to the final arc of the Object Identifier	72
Annex F (informative)	Relating data objects	73
F.1	Concatenation technique	73
F.2	Object identifier extension technique	73
Annex G (informative)	Data security issues.....	75
G.1	Object identifier issues	75
G.2	The data object	75
G.3	Using the TagId.....	75
G.4	Advice on public key methods of encryption.....	76
Annex H (informative)	Example of a transfer encoding	77
H.1	Functional description of the command	77
H.2	The abstract syntax for the AddMultipleObjects command	77
H.3	The AddMultipleObjects command with the data values.....	78
H.4	The transfer encoding for the example command.....	78
H.5	Functional description of the response	79
H.6	The abstract syntax for the AddMultipleObjects response	79
H.7	The AddMultipleObjects response with the data values.....	80
H.8	The transfer encoding for the example response.....	80
Annex I (informative)	Guidance to implementers of development commands	81

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 appropriate part of ISO/IEC 18000. 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. Application 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 this International Standard and ISO/IEC 15962.

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. This International Standard and ISO/IEC 15962 specify this data protocol, which is independent of any of the air interface standards defined in 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 application commands is the subject of this International Standard. The companion International Standard, ISO/IEC 15962, 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: application interface

1 Scope

The data protocol used to exchange information in an RFID system for item management is specified in this International Standard and in ISO/IEC 15962. Both International Standards are required for a complete understanding of the data protocol in its entirety; but each focuses on one particular interface:

- This International Standard addresses the information interface with the application system.
- ISO/IEC 15962 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 the interface between the application and the data protocol processor, and includes the specification of the transfer syntax and definition of application commands and responses. It allows data and commands to be specified in a standardised way, independent of the particular air interface of ISO/IEC 18000.

This International Standard

- provides guidelines on how data shall be presented as objects;
- defines the structure of object identifiers, based on ISO/IEC 9834-1;
- specifies the commands that are supported for transferring data between the application and the RF tag;
- specifies the responses that are supported for transferring data between the RF tag and the application;
- provides a formal description of all the processes using ASN.1, as specified in ISO/IEC 8824-1;
- specifies the transfer syntax, based on the Basic Encoding Rules of ISO/IEC 8825-1, for data to be transferred from and to the application.

It is expected that this International Standard will be used as a reference to develop software appropriate for particular applications, or for particular RF equipment.

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 9834-1, *Information technology — Open Systems Interconnection — Procedures for the operation of OSI Registration Authorities: General procedures* (equivalent to ITU-T Recommendation X.660)

ISO/IEC 15962:2004, *Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions*

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 relating to AIDC¹⁾*

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, 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 command

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 on 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.

3.1.4 BER = Basic Encoding Rules

An ASN.1 encoding method.

3.1.5 Block

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

1) To be published.

3.1.6 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.

3.1.7 Data carrier

A device or medium used to store data as a relay mechanism in an AIDC system, e.g. bar code, OCR character string, RF tag.

3.1.8 Data compaction

A mechanism, or algorithm, to process the original data so that it is represented efficiently in fewer octets in a data carrier than in the original presentation.

3.1.9 Data Compactor

The implementation of the data compaction process defined in ISO/IEC 15962.

3.1.10 Data Protocol Processor

The implementation of the processes defined in ISO/IEC 15962, including the Data Compactor, Formatter, Logical Memory, and Command / Response Unit.

3.1.11 Element name

A component of a Reference Type or enumerated list in ASN.1 syntax.

3.1.12 Formatter

The implementation of the data formatting process defined in ISO/IEC 15962.

3.1.13 Logical Memory

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

3.1.14 Logical Memory Map

An array of contiguous octets 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 (see 7.1.2) 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.

3.1.15 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.

3.1.16 Object identifier

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

3.1.17 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).

3.1.18 Octet

An ordered sequence of eight bits considered as a unit, equivalent to an 8-bit byte.

NOTE: The term is used in preference to "byte" in this International Standard and in the ASN.1 standards to avoid confusion in cases where there is a hardware association e.g. 7-bit byte, 16-bit byte.

3.1.19 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.

3.1.20 Response

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

3.1.21 System information

Information held on the RF tag, or generated by unique features of the air interface, that specify Data Protocol parameters to establish the Logical Memory and other formatting rules.

3.1.22 Tag Driver

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

3.1.23 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.

3.1.24 Type reference

A name, in ASN.1 syntax, associated uniquely with a characteristic e.g. ObjectId.

3.1.25 Unique item identifier

A code assigned to an item (for example a product, transport unit, returnable asset) that is unique within the domain and scope of a code system. When used with this Data Protocol, the particular ObjectId that defines the unique item identifier shall rely on the fact that each instance of its object shall be unique and unambiguous with all others related objects. As the object is unique, its use in the RF tag confers uniqueness to the RF tag itself.

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

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 octets 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 this International Standard is to specify a common interface between the application programs and the RF interrogator. The prime function of ISO/IEC 15962 is to specify the common encoding rules and logical memory functions.

RF 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 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.

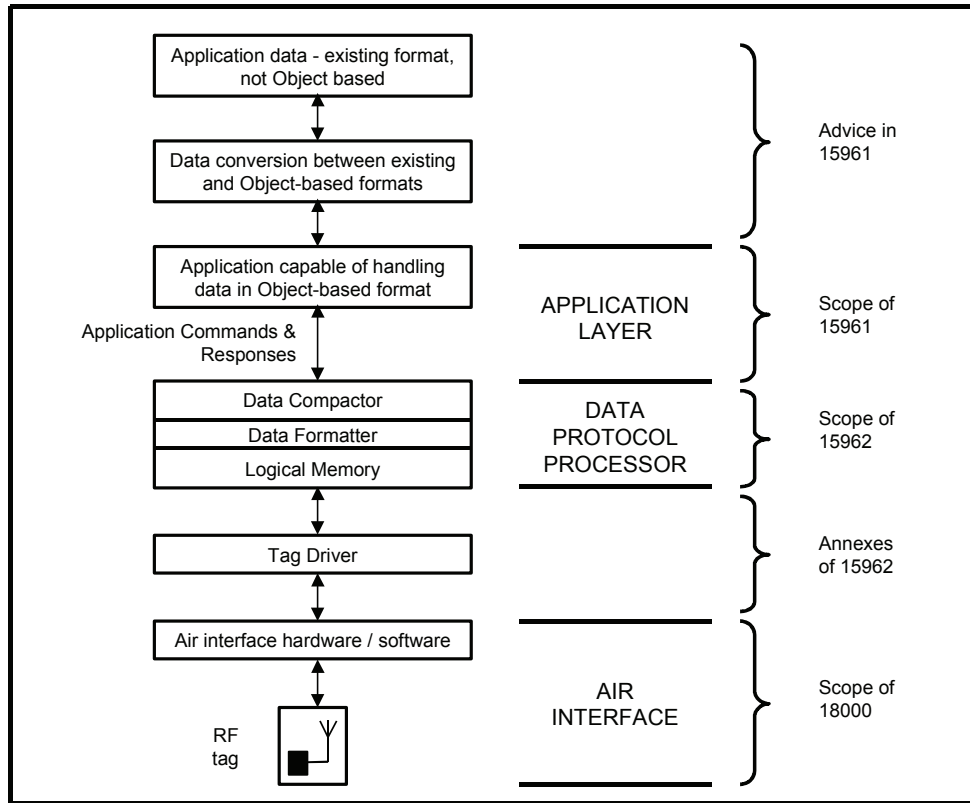


Figure 1 — Schematic of Protocol Layers for an Implementation of RFID for Item Management

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 this International Standard

- 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 this International Standard and those of ISO/IEC 15962.
- 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 ISO/IEC 15962

- 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 of ISO/IEC 15962).

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.

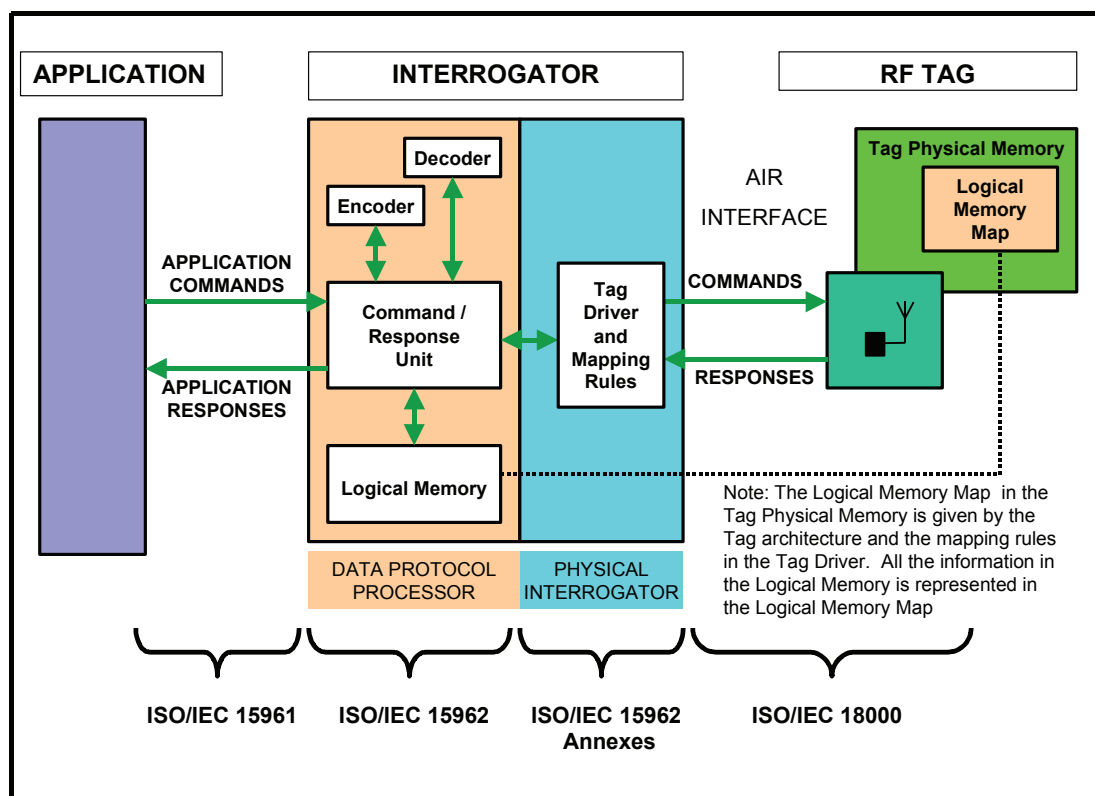


Figure 2 — Logical Functions and Interfaces

Application is the user application database and software.

The data flows between the application and the Data Protocol Processor are formatted according to this International Standard and are uncompacted. 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 15962. 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 ISO/IEC 15962 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 this International Standard, 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 ISO/IEC 15962. For the purpose of ISO/IEC 15962, 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 Notation

5.1.1 The octet: the basic unit for 8-bit coding

This International Standard supports binary, 7-bit, 8-bit and user data that may exceed 8-bits per character. The common unit of coding is the octet (also known as the 8-bit byte). Binary data shall be padded with leading zero bits until the binary value is octet aligned; 7-bit data shall be represented as octets with bit 8 (see 5.1.2) set to a zero value. Data exceeding 8-bits shall be encoded in multiple octets.

An octet is represented as 2 hexadecimal characters with the values 0-9, A-F.

5.1.2 Bit ordering

Within each octet, 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

5.1.3 Octet conversion

The 8-bit value is converted into the two hexadecimal characters with bit 8, bit 7, bit 6 and bit 5 having the weights 8, 4, 2 and 1 respectively to define the first hexadecimal character. Bit 4, bit 3, bit 2 and bit 1 retain the weights 8, 4, 2 and 1 respectively to define the second hexadecimal character.

5.2 Structure of the transfer between ISO/IEC 15961 and ISO/IEC 15962

All object identifiers, data, commands, and responses transferred between the application and the Data Protocol Processor shall be octet aligned. This simplifies the construction of the transfer and aids parsing the octets. It has no significance on the encoding efficiency on the RF tag itself, because the process by the Data Protocol Processor (as specified in ISO/IEC 15962) controls the final encodation.

6 Abstract and transfer syntax

ASN.1 defines:

- an **abstract syntax**, which is effectively a data definition language used to define repetitive and optional structures using a number of primitive data types. It is equivalent to high level programming languages, but is independent (hence "abstract") from any of these.
- a **transfer syntax** from which the encoding rules are derived. These rules determine the bit pattern representation during the transfer of data structures created using the abstract syntax.

6.1 Abstract syntax

The abstract syntax for this International Standard shall be ASN.1 as defined in ISO/IEC 8824-1. The notation shall be as specified in that standard.

The syntax required for RFID for item management are specified in Clause 7.

6.1.1 Character set

The character set used to define an ASN.1 item shall consist of:

A to Z
a to z
0 to 9
:=, { } < . @ () [] - ' " | & ^ * ; !

This character set is identical to that defined in ISO/IEC 8824-1.

6.1.2 Universal Types

ASN.1 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.

Table 1 — Universal Types Used in this Standard

Universal Type	Class Tag
BOOLEAN	1
INTEGER	2
OBJECT IDENTIFIER	6
OCTET STRING	4
RELATIVE-OID (reserved for future commands)	13
SEQUENCE & SEQUENCE OF	16

6.1.3 Type references

In addition to the Universal Types, ASN.1 enables application specific types to be defined. When a type is defined, it is given a name to reference it in another type assignment. Type references begin with an uppercase letter and the complete name is shown without spaces. There are a few variants to the presentation of the subsequent characters. This International Standard uses the convention of mixed upper and lowercase characters e.g. **TypeReference**.

The **TypeReference** name is followed by the three character sequence ": : =" to separate it from its definition.

Examples of **TypeReference** names that are used in this International Standard are:

ApplicationFamilyId
ObjectId
StorageFormat
TagId

All the **TypeReference** names used in this International Standard are defined in the appropriate sub-clause.

6.1.4 Element names

The components or elements of a TypeReference or enumerated list are named using a lowercase letter at the beginning, e.g. **elementName**. For some elements, further typing is required either to a TypeReference or a Universal Type.

Examples of **elementNames** that are used in this International Standard are:

```
accessMethod
applicationFamilyId
applicationSubFamily
commandCode
compactParameter
object
objectId
tagId
```

NOTE: In this International Standard, some **elementNames** and **TypeReference** names are often the same with the exception that the first letter is lowercase for the **elementName** and uppercase for the **TypeReference**.

All the **elementNames** used in this International Standard are defined in the appropriate sub-clauses.

6.1.5 Other ASN.1 conventions illustrated

By using a simple example of ASN.1 syntax, unrelated to the purpose of this International Standard, it is possible to illustrate the aspects covered in 6.1.2 to 6.1.4. Other aspects will also be identified.

EXAMPLE:

```
CustomerOrder ::= SEQUENCE {
    orderNumber      INTEGER
    name             OCTET STRING
    address          CustomerAddress
    productDetails   SEQUENCE OF SEQUENCE {
        productCode   OBJECT IDENTIFIER
        quantity      INTEGER (1..999)
    },
    urgency          ENUMERATED {
        nextDay (0),
        -- excludes Saturday and Sunday
        firstClass (1),
        roadTransport (2),
        -- typically three days
    }
}
```

NOTE:

- In this example, `::=` separates the named TypeReference **CustomerOrder** from the definition.
- The curly brackets `{ }` following **SEQUENCE** and the end, specify that **orderNumber**, **name**, **address**, **productDetails** and **urgency** are all elements of the CustomerOrder type reference.
- The element name **address** is further specified in the **CustomerAddress** type reference (excluded from the example for brevity).
- The element **productDetails** consists of two further elements **productCode** and **quantity**. This pair of elements is repeated n times, based on the **SEQUENCE OF SEQUENCE**. The `{ }` define the boundary of the Type.
- The element **urgency** offers one of three codes: 0, 1 or 2 by the use of the **ENUMERATED** type. The `{ }` define the boundary of the Type.
- Comments in this example "excludes Saturday and Sunday" and "typically three days" are preceded by the double dash "--".

- The **INTEGER** value for the element **quantity** is constrained by the "(1..999)" to be any value in the range 1 to 999, making it an error to order 1000 or more items of a **productCode**.

6.1.6 Modular structure of ASN.1 syntax

In keeping with the ASN.1 standards, the syntax in this International Standard has been presented in a modular format. Separate modules are used for the commands and for the responses. Each module contains:

- A unique name.
- A unique object identifier that refers to this particular standard (in accordance with ISO/IEC 8824-1). The penultimate arc is either "commandModules (126)" or "responseModules (127)" to distinguish the data flows. The final arc of each command/response pair has the same name and value to link these together.
- The use of the key words DEFINITIONS, BEGIN and END to be compliant with ISO/IEC 8824-1 and to allow the modules to be processed through compiler tools.
- A statement that this standard uses "EXPLICIT TAGS", which indicates that all of the elements ultimately encoded as UNIVERSAL TYPES.

The structure of a command module follows the following common format:

```
Module Name
{ISO(1) standard(0) rfid-data-protocol (15961) commandModules (126) moduleName(n)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

CommandName
-- assignments

END
```

The responseModule follows a similar format.

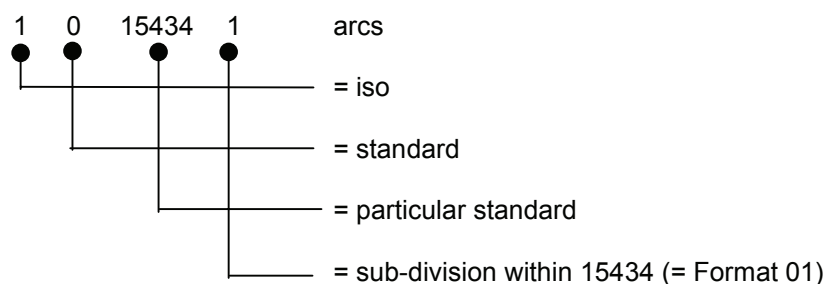
Within each module, all of the elements are defined in such a way as to reduce these to UNIVERSAL TYPES. This avoids the need to implement any import function within the module.

6.1.7 Object identifiers

This International Standard shall use 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 2.

Table 2 — 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 Annex A.

The third arc is controlled by the system or body defined for the second arc (see Annex A); 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 RF tag memory.

Two forms of object identifier can be used with the RFID Data Protocol:

- **OBJECT IDENTIFIER:** This full structure is used for communications between the application and the Data Protocol Processor defined by the scope of this International Standard. The OBJECT IDENTIFIER type is fully specified in 6.2.2. The full structure is suitable for use in ISO/IEC 15962 where the set of object identifiers to be encoded on the RF tag have different higher level arcs. In such cases all the arcs of the OBJECT IDENTIFIER shall be encoded in the RF tag.
- **RELATIVE-OID:** This structure is not used for communication between the application and the Data Protocol Processor defined by the scope of this International Standard. However, the RELATIVE-OID structure is applied in ISO/IEC 15962 in situations where a common root applies to the set of object identifiers to be encoded on the RF tag. For example, if all the object identifiers have the common root 1 0 15434 1 (as in the example in above) encoding space can be saved on the RF tag if this common root does not have to be encoded for each object identifier. The RELATIVE-OID assumes a common root-OID, which is either encoded or declared in some other way.

NOTE: The specification of the RELATIVE-OID is included in this International Standard to illustrate for those mainly concerned with the application interface how it is used for encoding by the Data Protocol Processor of ISO/IEC 15962.

6.2 Transfer syntax

The transfer syntax of this International Standard is based on the Basic Encoding Rules (BER) of ASN.1, as specified in ISO/IEC 8825-1.

6.2.1 Structure of the transfer encoding

The structure of the transfer encoding for the data protocol for RFID for Item Management is described below:

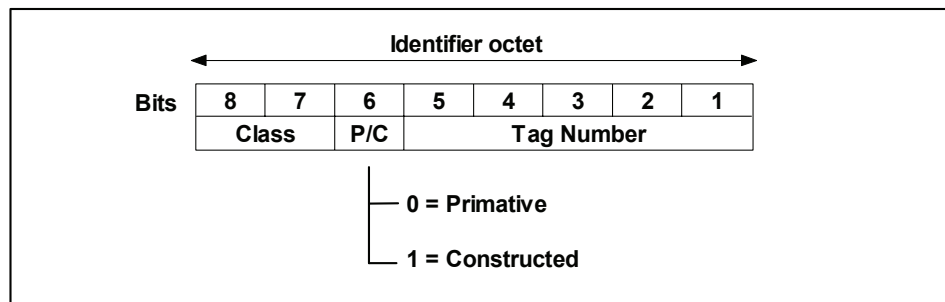
1. type identifier octet(s) that encode the ASN.1 tag (class and number) of the Type used to qualify the data value (see 6.2.2);
2. length octets that define a count of the octets that make up the contents (see 6.2.3);
3. content (or value) octets (see 6.2.4).

This is sometimes known as Type, Length, Value (**TLV**). When the encoding is based on a sequence of **TLV**, it is known as Primitive encoding. The value V can be a triplet of **TLV**, and when this structure is used it is known as Constructed encoding, for example **TL TLV TLV TLV**. The choice between Primitive and Constructed encoding is largely determined by the Basic Encoding Rules of ISO/IEC 8825-1. The constructor types of SEQUENCE and SEQUENCE OF that shall use a Constructed encoding to be fully compliant the ASN.1 standards. Otherwise, the rules of ISO/IEC 8825-1 require that Primitive encoding is used; or offer a choice, in which case only Primitive encoding shall be used in this International Standard. The option is clearly defined for each of the Universal Types (see 6.1.2) that have their BER encoding rules defined in subsequent clauses.

The module OBJECT IDENTIFIER shall be encoded in a TLV structure at the beginning of the transfer.

6.2.2 Encoding the ASN.1 type identifier

The ASN.1 type identifier shall be encoded as a single octet for the Types defined in this International Standard, as illustrated in Figure 3.



where: bits 8 and 7 shall encode the class of ASN.1 tag as defined in Table 3

bit 6 shall encode whether this is a Primitive or Constructed tag (see below)

bits 5 to 1 shall encode the number of the ASN.1 tag

Figure 3 — Type Identifier Octet Structure

The 2-bit value for Class shall be one of the values defined in Table 3. For the Universal Types defined in this International Standard, the value shall be '00₂'.

Table 3 — Encoding of ASN.1 Class of Tag

Class	Bit 8	Bit 7
Universal	0	0
Application	0	1
Context-specific	1	0
Private	1	1

The single bit value for the 'P/C' component shall be set to '0₂' to indicate Primitive encoding structures, or shall be set to '1₂' to indicate constructed encoding structures.

The 5-bit value for the ASN.1 tag shall encode the Class tag number as a binary integer with bit 5 as the most significant bit. The Class tags specified for this International Standard are defined in Table 1.

EXAMPLE:

Universal Type = OBJECT IDENTIFIER
 ASN.1 Type identifier = 00 0 00110

6.2.3 Length encoding

Although ISO/IEC 8825-1 allows other forms of length encoding, only the definite form shall be used in this International Standard because this applies to both primitive and constructed encoding. For the definite form, the length octets shall consist of one or more octets, depending on the number of octets in the contents. If the number of octets in the contents is less than, or equal to, 127, then a single length octet shall be used. Bit 8 shall be '0₂' and bits 7 to 1 shall encode the number of octets in the content (which may be zero) as an unsigned binary integer with bit 7 as the most significant bit.

EXAMPLE:

L = 38 is encoded as 00100110₂

If the number of octets in the contents is more than 127, then two or more length octets shall be used. The length shall be converted to an octet aligned value, for example, a length of 201 octets converts to C9₁₆ (or 11001001). This value is encoded in the second and subsequent octets. The first octet shall be encoded as follows:

- Bit 8 shall be 1₂.
- Bits 7 to 1 shall encode the number of subsequent octets in the length octets, as an unsigned binary integer with bit 7 as the most significant bit.
- The value 11111111₂ shall not be used to allow for future extension:

EXAMPLE:

Length of content = 357
 Convert to HEX = 01 65₁₆
 = 00000001₂ 01100101₂
 As this is 2 octets, the first octet = 100000010₂

The complete length encoding is:

10000010 00000001 01100101₂
 = 82 01 65₁₆

6.2.4 Contents Octets

The contents octets encode the data value, which can be zero, one or more octets, depending on the Universal Type as specified in subsequent subclauses.

6.2.5 Encoding of a BOOLEAN value

The encoding of a BOOLEAN value shall be primitive to comply with ISO/IEC 8825-1. The Boolean value shall be encoded in a single octet. If the Boolean value is FALSE, the octet shall be zero. If the Boolean value is TRUE, the octet shall have any non-zero value, at the sender's option.

6.2.6 Encoding an INTEGER value

The encoding of an INTEGER value shall be primitive to comply with ISO/IEC 8825-1. The integer shall be encoded in one or more octets using the following procedures.

For positive integers and zero:

1. The whole number is converted to a binary integer number in a bit field with the most significant bit first.
2. The bit field is aligned to octet boundaries by adding leading zero bits.
3. If the high order bit is 1, add a pad octet 00₁₆ as a prefix.

NOTE: The high order bit of 0 denotes that the encodation is of a positive integer value.

EXAMPLE:

Integer 128		
Step 1:		10000000
Step 2:		10000000
Step 3:	00000000	10000000

For negative integers, the encodation is to a twos-complement rule:

1. The whole number is converted to a binary integer number in a bit field with the most significant bit first.
2. The bit field is aligned to octet boundaries by adding leading zero bits.
3. The binary value from Step 2 is bit complemented (ie 0 to 1, 1 to 0).
4. The twos-complement rule is applied adding 1₂ to the bit string of Step 2.
5. If the high order bit is 0, add a pad octet FF₁₆ as a prefix.

NOTE: The high order bit of 1 denotes that the encodation is of a negative integer value.

EXAMPLE:

Integer -27066		
Step 1:	1101001	10111010
Step 2:	01101001	10111010
Step 3:	10010110	01000101
Step 4:	10010110	01000110

For decoding, the lead bit of the encoded integer value identifies whether the value is positive or negative.

If it is a positive value, conversion takes place on the remaining bits with the least significant bit being in position 0. The decimal integer value is the sum of the values 2^n , where n is the position number, or:

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

If it is a negative value, 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:

$$\begin{array}{rcl} 10010110 & 01000110 & \\ 1 & & \text{indicates -ve} \\ 0010110 & 01000110 & = 5702 \\ 2^p = 2^{15} = & & 32768 \\ 5702 - 32768 = & & -27066 \end{array}$$

6.2.7 Encoding the OBJECT IDENTIFIER value

The encoding of an OBJECT IDENTIFIER value shall be primitive to comply with ISO/IEC 8825-1. The object identifier value is encoded as a series of octet aligned values as follows:

1. The first two arcs of the registration tree are encoded as a single integer using the formula:

$$\begin{array}{lcl} 40f + s & & \\ \text{where } f & = & \text{the value of the first arc} \\ s & = & \text{the value of the second arc} \end{array}$$

2. The value "n" of each additional arc is encoded into an octet-aligned-bit-field. This is done as follows for values of "n":
 - a. For $n < 128$:

the decimal value is converted to binary and encoded in a single octet; thus bit 8 is set to zero
 - b. For $128 \leq n < 16384$:

the decimal value is converted to binary and subdivided into two 7-bit strings: bit 7 to bit 1, bit 14 to bit 8. Each of these new bit strings is encoded in an octet; with bit 8 of the first octet set to 1, bit 8 of the last octet set to 0.
 - c. For $n \geq 16384$:

the decimal value is converted to binary and subdivided into 7-bit strings: bit 7 to bit 1, bit 14 to bit 8, bit 21 to bit 15, and so on. Each of these new bit strings is encoded in an octet; with bit 8 of the first octet set to 1, bit 8 of the last octet set to 0 and bit 8 of the intervening octet(s) set to 1. The example below shows the process.

EXAMPLE:

1. Value = 91234_{10}
 = 1 01100100 01100010₂
2. Split into 7-bit strings
 0000101 1001000 1100010
3. Add prefix bits 0 for last octet
 1 for preceding octet(s)
 10000101 11001000 01100010

Using this technique, the length of each component arc of the OBJECT IDENTIFIER is self declaring. The first octet always defines the first two arcs. Each subsequent arc is defined by one octet if the lead bit of the next octet is 0; and multiple octets if the lead bit is 1, the group of octets ends with the octet with its lead bit equal 0. The arc value is encoded in the sequence of 7-bit values.

EXAMPLE:

[00101000]	1[1111000]	0[1001010]	0[0000001]
(1 x 40) + 0	15434	1	1
1 0	15434	1	1

Although the number of arcs allows for an OBJECT IDENTIFIER of any length, this International Standard limits the length of the encoded value to be no more than 127 octets. This is a constraint placed to meet encoding requirements on the RF tag and the structure of the Logical Memory.

NOTE: The constraint is on the encoded length of the OBJECT IDENTIFIER and not the number of arcs. It should also be understood that an OBJECT IDENTIFIER encoded in 127 octets is highly unlikely.

6.2.8 Encoding an OCTET STRING value

Although the Basic Encoding Rules of ISO/IEC 8825-1 permit both forms of encoding, this International Standard only supports primitive encoding of an OCTET STRING value.

The primitive encoding contains zero, one or more octets equal in value to the octets in the application data value. The encoded octets appear in the same order as they appear in the data value and with the most significant bit of an octet being aligned in both the encoded and data presentations.

NOTE: This means that, for open systems, the octet sequence and bit ordering shall be output by the receiving system exactly as input by the sending system. It is the responsibility of the application standards to define the requirements for the sequence.

6.2.9 Encoding a SEQUENCE value

The encoding of a SEQUENCE value shall be constructed to comply with ISO/IEC 8825-1. The contents octets shall consist of the complete TLV encoding of one data value from each of the Types listed in the ASN.1 definition of the particular SEQUENCE Type. The data values shall be in the order of their appearance in the definition. Although ISO/IEC 8825-1 allows optional rules for Types with the keywords 'OPTIONAL' or 'DEFAULT' in the ASN.1 definition, this International Standard requires all Types in the SEQUENCE to appear in the constructed encoding.

EXAMPLE:

ASN.1 definition

SEQUENCE {orderNumber OCTET STRING, product OCTET STRING, quantity INTEGER}

with the values:

{orderNumber "ABC1234", product "widget", quantity "12"}

is encoded as:

T = SEQUENCE	L	
30 ₁₆	14 ₁₆	
T = OCTET STRING	L	V
04 ₁₆	07 ₁₆	"ABC1234"
T = OCTET STRING	L	V
04 ₁₆	06 ₁₆	"widget"
T = INTEGER	L	V
02 ₁₆	01 ₁₆	0C ₁₆

6.2.10 Encoding a SEQUENCE OF value

The SEQUENCE OF type has the same ASN.1 tag (UNIVERSAL 16) as the SEQUENCE type; therefore, adopting the same encoding rules. The encoding of a SEQUENCE OF value shall be constructed to comply with ISO/IEC 8825-1. The contents octets shall consist of the complete TLV encoding of each value, including the encoding of the repeated UNIVERSAL class tag of the encoded elements.

EXAMPLE:

ASN.1 definition

sequence of {productCode OCTET STRING}

with the three values:

{productCode "ABC1234", "X6789Y", "PQR12345"}

is encoded as:

T = SEQUENCE	L	
30 ₁₆	1B ₁₆	
T = OCTET STRING	L	V
04 ₁₆	07 ₁₆	"ABC1234"
T = OCTET STRING	L	V
04 ₁₆	06 ₁₆	"X6789Y"
T = OCTET STRING	L	V
04 ₁₆	08 ₁₆	"PQR12345"

7 Data flows and processes

Various processes are required to format the RF tag, to write data to it, to read from it, to modify data etc. These are defined in the sub-clauses that follow. All the processes to write and add data will be described. Where the read process is the inverse of the write process, this will be described briefly, otherwise a further description will be provided. Within this clause and subsequent clauses, the appropriate transfer encoding values will be shown.

7.1 Establishing communications between the application and the RF tag

The Data Protocol Processor does not communicate directly with the RF tag (see Figures 1 and 2), but does this through the Tag Driver. The Data Protocol Processor requires specific system information based on the configuration of the RF tag (see 7.1.2). This is to set parameters of the Logical Memory to represent correctly the

RF tag memory and to enable communication to and from the Tag Driver. To achieve this, air interface services shall be provided to the Data Protocol Processor via the Tag Driver to establish communications (see 7.1.1).

Various of these parameters need to be known to, or requested by, the Data Protocol Processor. Effectively, this procedure is used to configure the RF tag initially, to re-configure it if required, and to establish a communications link while the data transaction is open.

7.1.1 Air interface services

This International Standard is open-ended with respect to the fact that new types of RF tag may be added to ISO/IEC 18000 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 octets.

NOTE: the term application memory is used in this sub-clause as a generic name for the area of RF tag memory available for user data; this is sometimes called user memory in ISO/IEC 18000.

- Application memory shall be organised in blocks. These shall be fixed size and be of one or more octets.
- 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.

The RF tag shall have a mechanism for storing the system information (see 7.1.2), including the ability to write and read the component elements.

The technical details of the air interface services are provided in ISO/IEC 15962.

7.1.2 System information

The system information shall consist of the following elements that need to be transferred across the application interface and air interface, and are therefore part of this International Standard and ISO/IEC 15962:

```
tagId
applicationFamilyId
storageFormat, which itself consists of:
    accessMethod
    dataFormat
```

The terms and functions are described in the sub-clauses below.

The system information shall also consist of the following elements that only need to transfer between the RF tag and the Data Protocol Processor to configure the Logical Memory Map, and are therefore only part of ISO/IEC 15962:

```
physical block size
number of blocks
```