INTERNATIONAL STANDARD

ISO/IEC 24791-2

First edition
2011-10-15

Information technology — Radio frequency identification (RFID) for item management — Software system infrastructure —

Part 2: Data management

Technologies de l'information — Identification de radiofréquence (RFID) pour la gestion d'élément — Infrastructure de système de logiciel —

Partie 2: Gestion de données

Reference number
ISO/IEC 24791-2:2011(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 24791-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

ISO/IEC 24791 consists of the following parts, under the general title *Information technology — Radio frequency identification (RFID) for item management — Software system infrastructure*:

— *Part 1: Architecture*

— *Part 2: Data management*

— *Part 3: Device management*

— *Part 5: Device interface*

Introduction

Radio frequency identification (RFID) air interface technology is based on non-contact electro-magnetic communication among interrogators and tags. RFID software systems are composed of RFID interrogators, intermediate software systems, and applications that provide control and coordination of air interface operation, tag information exchange, and health and performance management of system components. RFID technology is expected to increase effectiveness in many aspects of business by further advancing the capabilities of automatic identification and data capture (AIDC). To achieve this goal through the successful adoption of RFID technology into real business environments, RFID devices, software systems, and business applications must provide secure and interoperable services, interfaces, and technologies. This is the goal of ISO/IEC 24791.

Information technology — Radio frequency identification (RFID) for item management — Software system infrastructure —

Part 2: Data management

1 Scope

This part of ISO/IEC 24791 defines the interface(s) that provide operations on RFID tag data including, but not limited to, reading, writing, collection, filtering, grouping, and event subscription and notification within the Software System Infrastructure (SSI).

Specifically, the interface(s) defined by this part of ISO/IEC 24791 provide the following features:

— full support for the commands and responses for air protocols supported by this part of ISO/IEC 24791 at an abstraction level appropriate for Data Management's position in the SSI architecture defined in ISO/IEC 24791-1;

— an abstract definition of commands and operations that can be applied to different network bindings and encoding mechanisms;

— support for the encoding mechanisms defined in ISO/IEC 15962;

— volume reduction, format or structure modification, data analysis, and data access appropriate for Data Management's position in the SSI architecture defined in ISO/IEC 24791-1;

— reporting of data to support application or data managing in formats controlled by the client, either inside or outside of SSI.

This part of ISO/IEC 24791 is composed of the EPCglobal *Application Level Events Standard*, in its entirety, with extensions to further support operation with ISO/IEC 15962 and the air protocols defined by ISO/IEC 18000.

2 Conformance

Conformance for this part of ISO/IEC 24791 shall satisfy the conformance requirements of the EPCglobal ALE Standard and the requirements defined in Clause 9 of this part of ISO/IEC 24791, which defines the required interpretation and extension of ALE to fully support the SSI architecture and its interaction with ISO/IEC 15962 and ISO/IEC 18000.

3 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 15962, *Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions*

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

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

ISO/IEC 24791-1, *Information technology — Radio frequency identification (RFID) for item management — Software system infrastructure — Part 1: Architecture*

The Application Level Events Standard (latest version), EPCglobal, http://www.epcglobalinc.org/standards/ale

4 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.

4.1
data management
function and its interfaces that provide reading, writing, collection, filtering, grouping, and event subscription and notification of RFID tag data to higher level applications and interfaces

4.2
endpoint
one of two components that either implements and exposes an interface to other components or uses the interface of another component

5 Symbols and abbreviated terms

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

AIDC automatic identification and data capture

ALE EPCglobal Application Level Events Standard

DSFID Data Storage Format Identifier

PO Packed Object

SSI Software System Infrastructure

UML Unified Modeling Language

6 Software System Infrastructure Architecture Overview

ISO/IEC 24791-1 defines the architecture for the Software System Infrastructure. The basic relationship among the interfaces and implementations of the Software System Infrastructure is depicted in Figure 1.

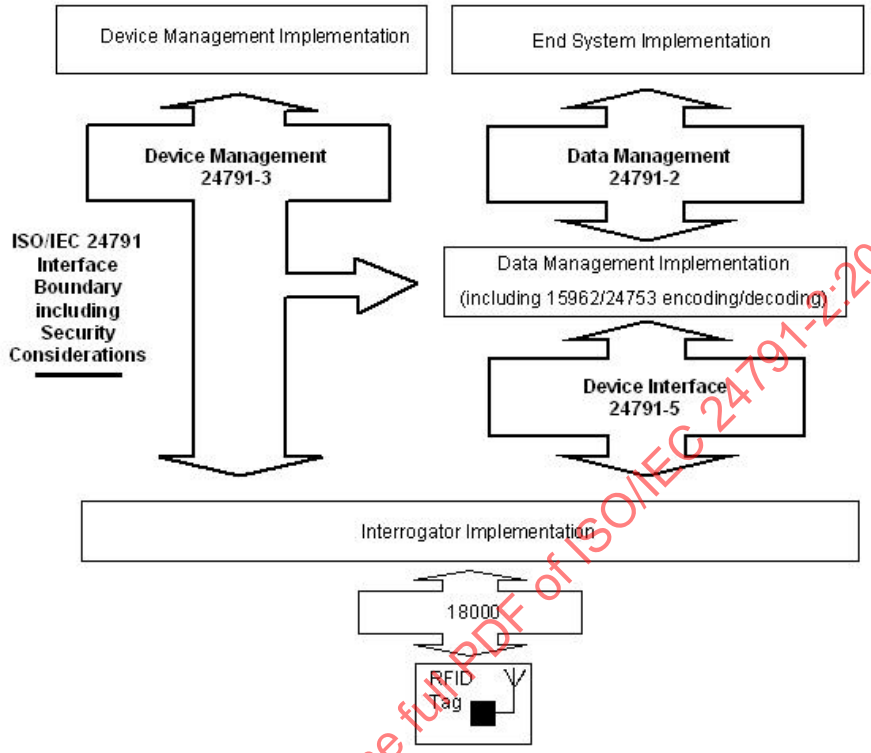

Figure 1 — Architecture Overview including Relationships to other RFID Standards

The Device Interface, Data Management, and Device Management each provide one or more interfaces that allow a client to communicate with a service-providing implementation, either within the same computing device or across a network. These client and service implementations are consistently referred to as Client Endpoints and Services Endpoints, respectively, and in general, the Client Endpoint accesses the capabilities provided by the Services Endpoint. It is the responsibility of the specific standard to define the formats, procedures, operations, and conformance requirements of each interface.

7 UML Modelling

Although Figure 1 provides a general overview of the relationship between the interfaces and implementations in the SSI, Unified Modeling Language (UML) is used for the figures in this document to graphically represent the organization and operation of the Data Management interfaces and implementations so that a precise and common understanding of the relationships among the components can be defined.

UML is a very rich language, but for simplicity only the Physical Diagram subset of the language is used to represent the architecture of the Software System Infrastructure. Physical diagrams, comprised of Component Diagrams and Deployment Diagrams, represent the relationships among the functions and the interfaces provided by the SSI architectural elements as well as how these functions might exist in standards compliant solutions, respectively. Refer to ISO/EC 24791-1 for the normative description of the UML diagrams used in this part of ISO/IEC 24791.

8 Data Management

This clause describes Data Management in terms of the overall Software System Infrastructure described in ISO/IEC 24791-1.

8.1 Architecture

As defined in ISO/IEC 24791-1, Data Management in the Software System Infrastructure provides operations on tag and sensor data including, but not limited to, reading, writing, collection, filtering, grouping, and event subscription and notification. A Data Management component exposes an interface as represented in Figure 2 whereby a Data Management Services Endpoint provides services to one or more Data Management Client Endpoints over an interface binding. The Client and Services Endpoints may exist in a single device or may be accessed across a network.

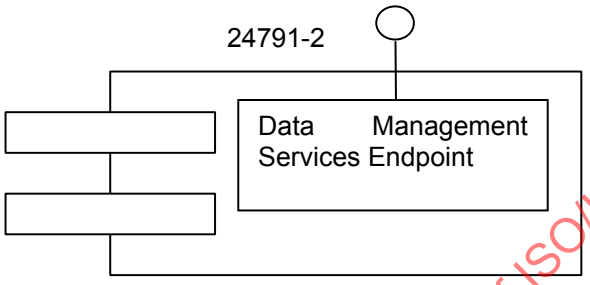

Figure 2 — Data Management Representation

Data Management provides for transfer of RFID tag information from a Services Endpoint to a Client Endpoint that requested the specific operation or data. The request for services is considered to be part of the *data* function as defined in ISO/IEC 24791-1, and the request is communicated over the interface between the Client and the Services Endpoints. The Services Endpoint, through a Data Management Implementation as shown in Figure 1, processes the Client requests, tag data, and possibly other external inputs (such as sensor, time or external logic signal) and delivers the results in the requested form to the Client. In the process of collecting and preparing tag data for delivery to a Client, the Services Endpoint may group, filter, decode, and possibly augment the tag data with additional data.

Data Management provides a control function that can be utilized by Clients to modify tag access parameters for requested data operations. When requested by a Client, the Data Management Implementation communicates the tag access parameters to the relevant interrogators using either the standard Device Interface or proprietary interrogator interface mechanisms, depending on the specific implementation. There is no architectural requirement for the control functionality to be communicated between Client and Services Endpoints through a single or different communication channel.

The Data Management Interface supports the semantics of tag encoding information as defined in ISO/IEC 15962 including the concepts of the unique identifier, AFI, DSFID, as well as the actual data to be encoded. This interface also supports the semantics of the parameters specified in ISO/IEC 15962 that serve to further define the state of the tag after writing or command execution. Examples of these parameters include those that specify the locking of portions of the tag memory as well as setting the encoding method that is reflected in the DSFID. It is the responsibility of the Data Management Implementation, as shown in Figure 1, to understand a request received from a Data Management Client for a tag data operation, create the ISO/IEC 15962-compliant encoding or tag state request, and communicate this request to the interface or implementation below it in the system.

Data Management will continue to support enhancements to ISO/IEC 15962 that provide additional capacity, performance, and tag format functionality, such as Packed Objects that minimize the tag user memory usage and profiles of tag data that minimize the tag operations required to achieve a specific result. Additionally, Data Management will also attempt to provide facilities to support other standardized tag formats in use in RFID software systems. The data format for a particular operation is defined by the Client for an operation

requested over the interface between the Data Management Client and the provider of the Data Management Services.

It is not required that an implementation of the Data Management component exist in an implementation of the Software System Infrastructure. The architecture is modular, and it is possible for systems to utilize proprietary or purpose-built applications to provide this function, or for systems to not provide this function at all.

8.2 Application Level Events (ALE) Overview

The EPCglobal Application Level Events Standard (ALE) defines interfaces that provide the Data Management capabilities defined in ISO/IEC 24791-1 and this part of ISO/IEC 24791.

As summarized in ALE, these interfaces are:

— The Reading API – an interface through which clients may obtain filtered, consolidated tag data from a variety of sources. In particular, clients may read RFID tags using RFID readers.

— The Writing API – an interface through which clients may cause operations to be performed on relevant RFID tags through a variety of actuators. In particular, clients may write RFID tags using RFID "readers" (capable of writing tags) and printers.

— The Tag Memory Specification API – an interface through which clients may define symbolic names that refer to data fields of tags.

— The Logical Reader Configuration API – an interface through which clients may define logical reader names for use with the Reading API and the Writing API, each of which maps to one or more sources/actuators provided by the implementation.

— The Access Control API – an interface through which clients may define the access rights of other clients to use the facilities provided by the other APIs.

ALE defines both synchronous request/response and asynchronous specification of desired tag operations requested by Data Management Client Endpoints to Data Management Services Endpoints. Clients can specify the desired parameters of *Event Cycles* or *Command Cycles* that request read or tag access (e.g. write) operations, respectively. Clients are also able to specify the desired format of the resulting output.

9 Data Management Use of ALE

9.1 Overview

This part of ISO/IEC 24791 is composed of ALE, in its entirety, with extensions and additional conformance requirements to further support operation with ISO/IEC 15962 and the air protocols defined by ISO/IEC 18000. The remainder of this document clarifies the usage of and defines the extensions to ALE that are required for conformance to this part of ISO/IEC 24791.

9.1.1 Terminology Mapping

The following terms used in ALE shall be interpreted in this part of ISO/IEC 24791 as defined in this subclause.

1) ALE Implementation – This is the Data Management implementation that provides the support for, and possibly the implementation of, the Data Management Services Endpoint.

2) Gen2 – This is equivalent to ISO/IEC 18000-6C except where noted.

3) EPC Memory Bank – This is bank 01 as defined in ISO/IEC 18000-6C.

9.1.2 Support for ISO/IEC 18000 Tag Types

It is a goal of the ISO/IEC 24791 to support operations with the air interface protocols defined in ISO/IEC 18000. As defined in the SSI architecture, the higher level operations and interfaces, such as those provided by Data Management, exercise less of a control function than those at lower levels in the architecture, such as the Device Interface. However, one of the lower level parameters exposed at the Data Management level of the architecture is the awareness of the air protocol.

Data Management Clients may request operations with parameters and capabilities that depend on the specific tag architecture. These include operations such as locking of memory and even the accessing of logical partitions of tag memory commonly referred to as *memory banks.* ALE, and therefore Data Management, provides commands for access to memory banks using symbolic references. It is these symbolic reference definitions for various ISO/IEC 18000 air interface specifications that will be the primary extension mechanism of ALE for ISO/IEC 24791-2 Data Management.

The support for the many different air interface standards defined in ISO/IEC 18000 will be specified over time in revisions to this part of ISO/IEC 24791. In this revision, ISO/IEC 18000-6C, ISO/IEC 18000-3 Mode 1, and ISO/IEC 18000-3 Mode 3 are specified.

Because ALE was written specifically to support the EPCglobal Gen2 air protocol, there is language in the ALE specification that states that when interacting with any other type of tag, the operation is implementation dependent and should be carefully documented. This part of ISO/IEC 24791 specifically supersedes this statement with the following requirement:

Data Management Implementations shall recognize the Pre-Defined Fieldnames and Data Types defined for the air protocols specified in subclause 9.2. However, SSI implementations are not required to implement support for all ISO/IEC 18000 air protocols defined in this part of ISO/IEC 24791. Because not all Pre-Defined Fieldnames and Data Types are exclusive to a particular air protocol, it is the responsibility of the implementation to provide the mechanism to inform users of requests for air protocol operations on tags that will not be encountered.

9.1.3 ALE API Implementation Requirements

ALE defines the requirements for implementation of the five APIs. As a summary, not all APIs are mandatory in an implementation. However, if an API is implemented, it must fully implement the API according to the specification and any additional conformance requirements imposed by this part of ISO/IEC 24791. Note that there are no additional implementation or conformance requirements imposed by this part of ISO/IEC 24791 for the ALE Tag Memory, Logical Reader, or Access Control APIs.

Data Management Implementations shall conform to the API implementation requirements defined in ALE.

9.2 Pre-Defined Fieldnames and Data Types

ALE defines specific fieldnames, datatypes and formats that are required to be supported by ALE implementations. Data Management Implementations shall conform to the ALE specification by recognizing and processing accordingly these pre-defined values and types.

This subclause of this part of ISO/IEC 24791 provides additions to and clarifications of the ALE specification that provide the capability necessary for Data Management Implementations to support ISO/IEC 18000, ISO/IEC 15962, and ISO/IEC 24753 in the SSI.

9.2.1 Gen2 Fieldnames

In ALE, several fieldnames have been predefined for the Gen2 air protocol. Data Management Implementations shall recognize these predefined fieldnames. The requirements for the actions of Data Management implementations are specific to the air protocol and are thus provided in the following separate subclauses.

9.2.1.1 ISO/IEC 18000-6C

Data Management implementations shall apply these predefined fieldnames to the equivalent logical memory banks when interacting with ISO/IEC 18000-6C tags, except where noted in this subclause.

When the afi fieldname is specified for a reading operation and the value of the bit at offset 17h is not 1, the Data Management Implementation shall raise a “field not found” condition.

When the afi fieldname is specified for a writing operation and the value of the bit at offset 17h is not 1, the Data Management Implementation shall raise a “field not found” condition. Bit 17h should be initialized to 1 using the [ALE] writing API with the UII Memory Bank INITIALIZE Operation described in [ALE].

9.2.1.2 ISO/IEC 18000-3 Mode 1

When interacting with ISO/IEC 18000-3 Mode 1 tags, Data Management Implementations shall respond to commands using the fieldnames defined in ALE according to Table 1.

Table 1 — ISO 18000-3 Mode 1 Fieldname Interpretation

Fieldname	Response
epc	The implementation shall raise a “field not found” exception
killPwd	The implementation shall raise a “field not found” exception
accessPwd	The implementation shall raise a “field not found” exception
epcBank	The implementation shall raise a “field not found” exception
tidBank	The implementation shall interpret as referring to the specific region of memory on the tag that holds the unique identifying information for the tag. The only CCOpType permitted for this fieldname is READ. The implementation shall raise an “operation not possible” condition if any other CCOpType is requested for this fieldname.
userBank	The implementation shall interpret as referring to the total encoding of the memory space on the tag.
afi	The implementation shall interpret as referring to the specific region of memory on the tag that holds the ISO/IEC 15962 Application Family Identifier. The only CCOpTypes permitted for this fieldname are READ, WRITE, and LOCK. The implementation shall raise an “operation not possible” condition if any other CCOpType is requested for this fieldname.
nsi	The implementation shall raise a “field not found” exception

9.2.1.3 ISO/IEC 18000-3 Mode 3

The structure and formats of tags conforming to ISO/IEC 18000-3 Mode 3 are equivalent, from the data field perspective, to ISO/IEC 18000-6C. As such, conforming Data Management implementations shall implement the field name requirements specified for ISO/IEC 18000-6C in 9.2.1.1.

9.2.2 The dsfidUii fieldname

A Data Management implementation shall recognize the string dsfidUii as valid fieldname as specified in this subclause. Requirements on the usage of this fieldname that are common across air protocols are provide in this subclause, and air protocol-specific requirements are provided in the following subclauses.

Data Management Implementations shall support operations on tag memory contents encoded as either a single or multiple byte DSFID, as defined in ISO/IEC 15962.

As defined in ISO/IEC 15962, the DSFID field includes bits indicating the Data Format and the Access Method for the subsequent memory on a tag.

If a WRITE operation is requested that indicates the dsfidUii fieldname, the Data Management implementation shall raise an “operation not possible” condition.

The default datatype for the dsfidUii fieldname shall be uint (as defined in ALE); the default format shall be hex. The Data Management implementation shall not permit any other datatypes in this specification to be used for the dsfidUii field.

9.2.2.1 ISO/IEC 18000-6C

This subclause contains requirements on conforming implementations when the requested operation is to be executed on an ISO/IEC 18000-6C tag.

Data Management Implementations shall interpret the dsfidUii fieldname as a synonym for the fieldname @1.8.32, that is, for bits 20_h through 27_h (inclusive) in the UII memory bank. If an extended DSFID is present in the UII memory bank, the Data Management implementation will need to use the syntax for absolute addressing in order to indicate the extended DSFID data.

Data Management Implementations shall raise an “operation not possible” condition if a LOCK command is requested on the dsfidUii field.

9.2.2.2 ISO/IEC 18000-3 Mode 1

This subclause contains requirements on conforming implementations when the requested operation is to be executed on an ISO/IEC 18000-3 Mode 1 tag.

Data Management Implementations shall interpret the dsfidUii fieldname as a synonym for the logical storage location for the DSFID field because two different implementations exist in current tags, *hardcoded*, which is fixed in memory, and *softcoded*, which is dynamically managed. ALE does not distinguish between the different types of memory implementations so it is the responsibility of the Data Management Implementation to respond appropriately based on the type of tag being accessed. If an extended DSFID is present in the UII memory bank, the Data Management implementation will need to use the syntax for absolute addressing in order to indicate the extended DSFID data.

The following CCOpTypes shall be supported for hardcoded DSFID tag implementations: READ, CHECK, INITIALIZE, WRITE, and LOCK. Any other CCOpType with dsfidUii for the fieldspec shall result in an “operation not possible” condition.

The following CCOpTypes shall be supported for softcoded DSFID tag implementations: READ, CHECK, INITIALIZE, WRITE, and LOCK. Any other CCOpType with dsfidUii for the fieldspec shall result in an “operation not possible” condition.

9.2.2.3 ISO/IEC 18000-3 Mode 3

The structure and formats of tags conforming to ISO/IEC 18000-3 Mode 3 are equivalent to ISO/IEC 18000-6C. As such, conforming Data Management implementations shall implement the requirements specified for ISO/IEC 18000-6C.

9.2.3 The dsfidUm fieldname

A Data Management implementation shall recognize the string dsfidUm as valid fieldname as specified in this subclause. Requirements on the usage of this fieldname that are common across air protocols are provide in this subclause, and air protocol-specific requirements are provided in the following subclauses.

Data Management Implementations shall support operations on tag memory contents encoded as either a single or multiple byte DSFID, as defined in ISO/IEC 15962.

As defined in ISO/IEC 15962, the DSFID field includes bits indicating the Data Format and the Access Method for the subsequent memory on a tag.

If a write operation is requested that indicates the dsfidUm fieldname, the Data Management implementation shall raise an “operation not possible” condition.

The default datatype for the dsfidUm fieldname shall be uint (as defined in ALE); the default format shall be hex. The Data Management implementation shall not permit any other datatypes in this specification to be used for the dsfidUm field.

9.2.3.1 ISO/IEC 18000-6C

This subclause contains requirements on conforming implementations when the requested operation is to be executed on an ISO/IEC 18000-6C tag.

Data Management Implementations shall interpret the dsfidUm fieldname as a synonym for the fieldname @3.8.0, that is, for bits 00_h through 07_h (inclusive) in the User Memory bank. If an extended DSFID is present in the User Memory bank, the Data Management implementation will need to use the syntax for absolute addressing in order to indicate the extended DSFID data.

Data Management Implementations shall raise an “operation not possible” condition if a LOCK command is requested on the dsfidUm field.

9.2.3.2 ISO/IEC 18000-3 Mode 1

ISO/IEC 18000-3 Mode 1 tags do not support multiple memory banks as defined in ALE. If an operation is requested on an ISO/IEC 18000-3 Mode 1 tag that indicates the dsfidUm fieldname, the Data Management implementation shall raise an “operation not possible” condition.

9.2.3.3 ISO/IEC 18000-3 Mode 3

The structure and formats of tags conforming to ISO/IEC 18000-3 Mode 3 are equivalent to ISO/IEC 18000-6C. As such, conforming Data Management implementations shall implement the requirements specified for ISO/IEC 18000-6C.

9.2.4 The tid fieldname

A Data Management implementation shall recognize the string tid as valid fieldname as specified in this subclause. This fieldname is only applicable to ISO/IEC 18000-3 Mode 1 tags. Data Management Implementations shall generate a “field not found” condition if a command is requested with tid as a fieldname on any tag family other than ISO/IEC 18000-3 Mode 1. Data Management implementations shall interpret the tid fieldname as a synonym for the physical or logical storage location for the unique tag identifying information. This memory is read-only; therefore, the only CCOpType permitted for this fieldname is READ. The implementation shall raise an “operation not possible” condition if any other CCOpType is requested for this fieldname.

The default datatype for the tid fieldname shall be uint (as defined in ALE); the default format shall be hex. The Data Management implementation shall not permit any other datatypes in this specification to be used for the dsfid field.

9.3 Absolute Address Fieldnames

This subclause defines how ALE's description of absolute address fieldnames is to be interpreted and extended in this part of ISO/IEC 24791. Requirements on the usage of absolute address fieldnames that are common across air protocols are provide in this subclause, and air protocol-specific requirements are provided in the following subclauses.

9.3.1 ISO/IEC 18000-6C

Data Management Implementations shall conform to ALE when the requested operation is to be performed on an ISO/IEC 18000-6C tag, including the mapping of *bank* values for absolute address fieldnames to the logical memory banks on the ISO/IEC 18000-6C tag.

9.3.2 ISO/IEC 18000-3 Mode 1

This subclause contains requirements on conforming implementations when the requested operation is to be executed on an ISO/IEC 18000-3 Mode 1 tag. Data Management Implementations shall interpret bank 0 as the beginning of the user memory bank. The implementation shall raise a “field not found” condition if any other value for bank is requested for this tag type.

9.3.3 ISO/IEC 18000-3 Mode 3

The structure and formats of tags conforming to ISO/IEC 18000-3 Mode 3 are equivalent to ISO/IEC 18000-6C. As such, conforming Data Management implementations shall implement the requirements specified for ISO/IEC 18000-6C.

9.4 Variable Fieldnames

This subclause defines how ALE's description of variable fieldnames is to be interpreted and extended in this part of ISO/IEC 24791. Requirements on the usage of this variable address fieldname, in the form *@bank.oid*, that are common across air protocols are provided in this subclause, and air protocol-specific requirements are provided in the following subclauses.

Note that although this interface expresses a variable fieldname in a *@bank.oid* format, the encoding of the information on the tag will be dependent on other inputs to the encoding mechanism such as data format, access method, and air protocol.

Conforming implementations shall support variable fieldnames as defined in ALE for both read and write operations, even though this capability is defined as “MAY” in ALE.

9.4.1 ISO/IEC 18000-6C

Data Management Implementations shall conform to ALE when the requested operation is to be performed on an ISO/IEC 18000-6C, referred to as “Gen2” in ALE, tag, including the mapping of *bank* values for variable fieldnames to the logical memory banks on the ISO/IEC 18000-6C tag.

9.4.2 ISO/IEC 18000-3 Mode 1

This subclause contains requirements on conforming implementations when the requested operation is to be executed on an ISO/IEC 18000-3 Mode 1 tag.

Data Management Implementations shall interpret bank 0 as the beginning of the user memory bank. The implementation shall raise a “field not found” condition if any other value for bank is requested for this tag type.

9.4.3 ISO/IEC 18000-3 Mode 3

The structure and formats of tags conforming to ISO/IEC 18000-3 Mode 3 are equivalent to ISO/IEC 18000-6C. As such, conforming Data Management implementations shall implement the requirements specified for ISO/IEC 18000-6C.

9.5 Variable Pattern Fieldnames

This subclause defines how ALE's description of variable pattern fieldnames is to be interpreted and extended in this part of ISO/IEC 24791. Requirements on the usage of variable pattern fieldnames, in the form *@bank.oid-prefix.*,* that are common across air protocols are provide in this subclause, and air protocol-specific requirements are provided in the following subclauses.

9.5.1 ISO/IEC 18000-6C

Data Management Implementations shall conform to ALE when the requested operation is to be performed on an ISO/IEC 18000-6C tag.

9.5.2 ISO/IEC 18000-3 Mode 1

This subclause contains requirements on conforming implementations when the requested operation is to be executed on an ISO/IEC 18000-3 Mode 1 tag.

Data Management Implementations shall interpret bank 0 as the beginning of the user memory bank. The implementation shall raise a “field not found” condition if any other value for bank is requested for this tag type.

9.5.3 ISO/IEC 18000-3 Mode 3

The structure and formats of tags conforming to ISO/IEC 18000-3 Mode 3 are equivalent to ISO/IEC 18000-6C. As such, conforming Data Management implementations shall implement the requirements specified for ISO/IEC 18000-6C.

9.6 Extensions to the CCOpType Values

In order to support the directory capabilities defined in ISO/IEC 15962, the following additions and clarifications to the ALE CCOpType formats and commands are defined.

9.6.1 INITIALIZE (User Memory Bank)

The INITIALIZE CCOpType for the User Memory Bank is specified in ALE. The following subclauses further clarify the operation described in ALE.

9.6.1.1 Change of DSFID from the No-Directory Access Method to the Directory Access Method

If a dataSpec of urn:epcglobal:ale:init:iso15962:xDD.force is specified and DD has a value that indicates initialization to Access Method 1, the current DSFID on the tag indicates Access Method 0, and the other bits in DD are the same as those on the tag, then the Data Management Implementation shall interpret this command as a request to change the Access Method and add a directory to the existing data on the tag as defined in ISO/IEC 15962. If the other bits in DD are not the same, then the entire user memory shall be cleared and the new value of DD shall be written to the tag.

9.6.2 Writing and Adding

The following extensions to the processing of the ALE Writing API CCOpType comands of WRITE and ADD are specified. These extensions are dependent on the specific Access Method of the data encoded on the tag.

9.6.2.1 Directory, No Directory, and Tag Data Profiles Access Methods

The following rules apply when writing to a tag memory bank that is encoded according to the ISO/IEC Directory, No Directory, and Tag Data Profiles Access Methods.

9.6.2.1.1 iso-15962-string datatype

If the datatype specified for the WRITE or ADD operation is *iso-15962-string,* then the dataSpec field shall be encoded using ISO/IEC 15962 encoding rules and this encoded data shall be written to the tag. The encoding process should use the most efficient compaction method defined by ISO/IEC 15962 for the specified Access Method in use on the tag. For instance, if the dataSpec field consists only of uppercase ASCII letters and digits, then compaction type 4 (6-bit code) should be used; if the dataSpec field contains only US ASCII, then compaction type 5 (US ASCII) should be used, if the dataSpec field contains only characters belonging to ISO/IEC 8859-1 (assigned values equivalent to Unicode code points 0020 – 00FF hex) then compaction type 6 (octet string) should be used, and if the dataSpec field contains an ISO/IEC 10646 character that does not belong to ISO/IEC 8859-1 then compaction type 7 (UTF-8) should be used.

Note that Annex B provides an informative reference algorithm for translating a sequence of Unicode code points to a sequence of UTF-8 octets if such a conversion is necessary (i.e., if compaction type 7 (UTF-8) is selected by the implementation).

9.6.2.1.2 bits datatype

If the datatype specified for the WRITE or ADD operation is *bits*, then the dataSpec field must be a value that specifies a multiple of eight bits in length, otherwise the implementation shall return an OUT_OF_RANGE_ERROR condition as specified in ALE. The value of the dataSpec field shall be interpreted as a sequence of octets, which are directly encoded according to ISO/IEC 15962.

9.6.2.2 Packed Objects Access Method

The following rules apply when writing to a tag memory bank that is encoded according to the ISO/IEC 15962 Packed Objects Access Method.

9.6.2.2.1 iso-15962-string datatype

The dataSpec field data string provided by the client shall be translated according to the character set specified by the K-interpretation field of the data table prior to encoding on the tag. If the input contains a character that is not supported by the character set specified by the K-interpretation field of the Packed Objects data format, the implementation shall return an OUT_OF_RANGE_ERROR condition as specified in ALE.

If the K-interpretation field is absent, or has the value “UNSPECIFIED” as defined in ISO/IEC 15962, then the tag contains a sequence of octets whose mapping to “characters” is not specified. In this case, the input character string should only include Unicode code points 00 – FF (hex) that are mapped directly into octets prior to encoding according to ISO/IEC 15962. If the input character string contains a Unicode character >= 100 (hex), then the implementation shall return an OUT_OF_RANGE_ERROR condition as specified in ALE.

9.6.2.2.2 bits datatype

If the datatype specified for the WRITE or ADD operation is *bits*, then the dataSpec field must be a value that specifies a multiple of eight bits in length, otherwise the implementation shall return an OUT_OF_RANGE_ERROR condition as specified in ALE. The value of the dataSpec field shall be interpreted as a sequence of octets, which are directly encoded according to ISO/IEC 15962. Note that the K-interpretation field of the data table does not affect processing in this case.

9.6.3 READ

The following extensions to ALE tag reading operations are defined. The extensions are relevant to the processing of the READ CCOpType in the Writing API well as to the processing of the ALE Reading API. These extensions are dependent on the specific Access Method of the data encoded on the tag.

9.6.3.1 Directory, No Directory Access, and Tag Data Profiles Method

The following rules apply when reading from a tag memory bank that is encoded according to the ISO/IEC Directory, No Directory, and Tag Data Profiles Access Methods.

9.6.3.1.1 iso-15962-string datatype

If the datatype specified for the read operation is *iso-15962-string,* then the octets decoded according to ISO/IEC 15962 shall be mapped to a string of Unicode characters and delivered to the client through the processes defined in ALE.

Note that Annex B provides an informative reference algorithm for translating a sequence of UTF-8 octets to a sequence of Unicode code points if such a conversion is necessary (I.e., when the data on the tag uses compaction type 7 (UTF-8).

9.6.3.1.2 bits datatype

If the datatype specified for the read operation is *bits*, then the octets decoded according to ISO/IEC 15962 shall be reported as a bits value, whose length is a multiple of eight bits, without further translation or modification.

9.6.3.2 Packed Objects Access Method

The following rules apply when reading from a tag memory bank that is encoded according to the ISO/IEC 15962 Packed Objects Access Method.

9.6.3.2.1 iso-15962-string datatype

If the datatype specified for the reading operation is *iso-15962-string*, the octets decoded according to ISO/IEC 15962 shall be interpreted according to the character set specified by the K-interpretation field of the data table and delivered to the client through the processes defined in ALE. If the K-interpretation field of a data format is absent, or has the value "UNSPECIFIED" as defined in ISO/IEC 15962, then the tag contains a sequence of octets whose mapping to “characters” is not specified. In this case, the octets 00-FF (hex) at the tag level shall be mapped to Unicode code points 00-FF (hex) in the iso-15962-string.

NOTE Although this mapping may or may not correspond to the interpretation the application wants to put on those octets, the application can reliably retrieve the octet values and then interpret them according to its requirements.

9.6.3.2.2 bits datatype

If the datatype specified for the read operation is *bits*, then the octets decoded according to ISO/IEC 15962 shall be reported as a bits value, whose length is a multiple of eight bits, without further translation or modification.

9.6.4 PO_CREATE CCOpType

Data Management implementations shall support the PO_CREATE CCOpType as defined in the text and Table 2 of this subclause. The PO_CREATE will result in the creation of a new Packed Object in memory when the Access Method for the memory bank is Packed Objects. If Packed Objects is not the Access Method for the memory bank, this command shall have no effect and shall be ignored.

The Data Management implementation shall consider all WRITE and ADD operations in a single CCCmdSpec after a PO_CREATE and before another PO_CREATE or PO_OPTIONS to pertain to the prior PO_CREATE. As such, the PO_CREATE sets the context for subsequent WRITE and ADD operations. When other CCOpType operations are specified with the CCCmdSpec, the Data Management implementation shall follow the requirement for the order of execution as specified for opSpecs in ALE.

If the DSFID specifies Packed Objects as the Access Method and a PO_CREATE CCOpType operation is not included in the CCCmdSpec, then the implementation shall create the new Packed Object with the default operation specified in subclause 9.6.4.1.

Table 2 — PO_CREATE Defintion

CCOpType Value	Description	Fieldspec	Dataspec
PO_CREATE	Provides the initial control parameters required for the creation of a new Packed Object	The memory bank in which to create the Packed Object; one of the values specified in ALE.	A LITERAL dataspec whose value specifies additional information that guides the creation of the directory. See subclause 9.6.4.1.

9.6.4.1 Values for the PO_CREATE Operation

PO_CREATE values of the following forms shall be recognized:

urn:iso:iso24791-2:po_create:[.editablePointerSize=D][.IDMap]
[.directoryType=pointer|basic|index|offset][.poDirectorySize=D][.poIndexLength=D][.objectOffsetsMultiplier=D]

The optional parameters in the po_create uri shall be interpreted to represent requested differences from the default behavior so the parameters need only be included if a difference from default behaviour is required. Table 3 describes each initialization parameter.

If any of the parameters or options in the uri is not recognized, the Data Management implementation shall raise a CCSpecValidationException.

Table 3 — PO_CREATE Parameters

Parameter	Description	Operation
.editablePointerSize	Specifies that the Packed Object shall be created as editable and provides the size in bits for the null-pointer link that will later point to an addendum Packed Object where edits can occur. Packed Objects are made editable by adding an optional Addendum subsection to the end of the Object Info section, which includes a pointer to an “Addendum Packed Object” where additions and/or deletions are made. The parameter, D, is an integer number of bits to reserve for the pointer.	The implementation shall mark the Packed Object as editable if this parameter is present and shall allocate D bits for a pointer to an Addendum Packed Object, otherwise, the Packet Object shall be created as not editable. If the implementation is not able to allocate D bits for a pointer to the Addendum Packed Object, the implementation shall raise an “operation not possible” condition.

Parameter	Description	Operation
.IDMap	This parameter specifies that a Packed Object shall be created to use the IDMap Object Info encoding	The implementation shall create the Packed Object using the IDMap Object Info encoding if this parameter is present, otherwise the Packed Object shall be created to use the IDList Object Info encoding
.directoryType	This parameter specifies the type of directory that the Packed Object shall be created to use	If this parameter is not present, no directory shall be created for the Packed Object. If the parameter is present, the directory as specified in parameter shall be created.
.poDirectorySize	This parameter specifies the size of the directory pointer (D) in bits to be created for the Packed Object when the .directoryType = pointer option is also present.	If this parameter is present but .directoryType=pointer is not also present, this parameter is ignored. Otherwise, the implementation shall use this parameter for appropriate sizing of the (null) directory pointer created with the Packed Object. If the implementation is not able to allocate the requested number of bits for the Packed Object, the implementation shall raise an “operation not possible” condition.
.poIndexLength	This parameter specifies the size of the POindex Length (D) to be used for the AuxMap structure to be created when the .directoryType=index option is also present.	If this parameter is present but .directoryType=index is not also present or does not contain an integer value in the range from 1 to 7, this parameter is ignored. Otherwise, the implementation shall use this parameter in the POIndex Length parameter created for the PO Index Field for this Packed Object in a PO index directory.
.objectsOffsetsMultiplier	This parameter specifies size of the memory (D) in bits that is requested for the storage of object offsets as defined in ISO/IEC 15962 when the .directoryType=offset and .poIndexLength options are also present. The implementation shall use this information for proper sizing of the AuxMap structure in the Packed Object.	If this parameter is present but .directoryType=offset and .poIndexLength are not also present, this parameter is ignored. Otherwise, the implementation shall use this parameter to reserve D bits of storage for object offsets in an AuxMap section of the directory Packed Object. If the implementation is not able to allocate N bits for the AuxMap section of the directory Packed Object, the implementation shall raise an “operation not possible” condition.

9.6.4.2 directoryType Options

The following options for .directoryType shall be recognized and the implementation shall create the Packed Objects directory for the items in the CCCmdSpec that will be created as a single Packed Object.

9.6.4.2.1 pointer

If the *pointer* option is indicated, the implementation shall create the pointer to a directory Packed Object as described in ISO/IEC 15962, recognizing that a subsequent PO_MODIFY command, most likely in a subsequent CCCmdSpec, would be used to specify the actual directory type.

9.6.4.2.2 basic

If the *basic* option is indicted, the implementation shall create the directory as a Presence/Absence directory as defined in ISO/IEC 15962.

9.6.4.2.3 index

If the *index* option is indicted, the implementation shall create the directory as a POIndex directory as defined in ISO/IEC 15962.

9.6.4.2.4 offset

If the *offsets* option is indicted, the implementation shall create the directory as a ObjectOffsets directory as defined in ISO/IEC 15962.

9.6.5 PO_OPTIONS CCOpType

Data Management implementations shall support the PO_OPTIONS CCOpType as defined in the text and Table 4 of this subclause. The PO_OPTIONS operation is used to modify the behaviour of a Packed Object after it has been created.

The fieldspec defines either a memory bank fieldname or a variable fieldname as defined in ALE. The fieldspec has the effect of setting the context for the scope of the dataspec in this CCOpSpec. Additionally, it sets the PO context for subsequent ADD and/or WRITE operations in the CCCmdSpec.

If a memory bank is specified, the PO_OPTIONS operation shall be executed for all Packed Objects in the memory bank. If the memory bank fieldname is not recognized or does not support Packed Object operations, the implementation shall raise an "operation not possible" condition.

If a variable fieldname is specified, then the operation shall be executed on the Packed Object that contains the specified field. If the object specified in the fieldspec is not found in the specified bank, the implementation shall raise a "field not found" condition.

The dataspec in this CCOpSpec is a LITERAL that defines the options to be applied to Packed Objects specified by the fieldspec.

Table 4 — PO_OPTIONS Defintion

CCOpType Value	Description	fieldspec	dataspec
PO_OPTIONS	Sets the defined set of options in a Packed Object that was previously created with the PO_OPTIONS command.	The field that determines the Packed Object context for this operation, either all Packed Objects in a memory bank if a memory bank fieldname or a specific PO if a variable fieldname	A LITERAL dataspec whose value specifies additional information that modifies a previously created PO directory structure. See subclause 9.6.5.1.