INTERNATIONAL STANDARD

ISO/IEC 24791-3

Second edition
2022-12

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

Part 3: Device management

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

Partie 3: Gestion de dispositif

Reference number
ISO/IEC 24791-3:2022(E)

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

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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

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

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

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

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

This second edition cancels and replaces the first edition (ISO/IEC 24791-3:2014), which has been technically revised.

The main changes compared to the previous edition are: the references have been updated to the latest standards.

A list of all parts in the ISO ISO/IEC 24791 series can be found on the ISO website.

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

Introduction

Radio frequency identification (RFID) air interface technology is based on non-contact electromagnetic 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 have to provide secure and interoperable services, interfaces, and technologies. This is the goal of the ISO/IEC 24791 series, created for RFID software system infrastructure (SSI).

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

Part 3: Device management

1 Scope

1.1 General

This document defines interfaces for device management of RFID systems. Interfaces are defined that provide for discovery, configuration, initialization and monitoring of RFID systems within the software system infrastructure (SSI).

This document only deals with devices that provide RFID related services. It does not distinguish the form factor of such RFID devices.

This document provides two distinct *interface sets*, one based on the GS1 EPCglobal DCI standard and the IETF SNMP RFCs and the other based on the Organization for the Advancement of Structured Information Standards (OASIS) DPWS standard. The definition of the Device Profile for RFID is referred to in this document as the RFID Device Management Profile, or RDMP.

Each interface option set provides interface definitions that provide ISO/IEC 24791-3 Client Endpoints and Services Endpoints with the mechanisms for:

— the discovery of the RFID devices and services on a local or remote subnet;

— a firmware upgrade service;

— a management service that implements configuration related functions;

— a monitoring service for reporting alerts, diagnostics, and performance information.

The two interface set definitions provided by this document allow for clients and services endpoints to implement and provide the services based on the specific characteristics of the RFID system to be implemented. Subclause 1.2 defines the Conformance requirements for systems that implement components of one or both of the interface sets.

1.2 Conformance

This document provides two interface sets; the DCI and SNMP Interface Set and the RDMP interface Set. If a certain implementation conforms to the mandatory functions of at least one of the interface sets, that implementation is conformant to this document.

1.3 DCI and SNMP interface set

This document divides the DCI capabilities into two *Conformance Groups:*

— Discovery, Configuration, and Initialization Conformance Group: this Conformance Group is defined in Clause 7. It specifies the protocols and operational procedures that are required for conforming Interrogator Implementations and Device Management Implementations, as defined in this document as well as in ISO/IEC 24791-1.

— Performance Monitoring and Diagnostics Conformance Group: this Conformance Group is defined in Clause 8. It specifies the SNMP MIBs that can be implemented by Interrogator Implementations and Data Management Implementations as defined in this document as well as in ISO/IEC 24791-1. Conforming implementations claim conformance to the MODULE_COMPLIANCE statements in the SNMP MIBs appropriate for the particular implementation.

A conforming implementation has to implement all of the requirements of each Conformance Group for its particular function in the SSI, but an implementation is not required to claim conformance to either group.

1.4 RDMP interface set

This document specifies the following device management capabilities in RDMP:

— discovery of devices and hosted services in devices;

— a Firmware Upgrade Service to initialize and manage firmware on devices;

— a Management service to set and get device configuration and to perform specific device operations, such as reboot;

— a monitoring service to monitor the health of a device using events and statistics.

RDMP interface set is defined in Clause 9.

2 Normative references

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

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

Devices Profile for Web Services (DPWS) Version 1.1, OASIS Standard July 2009. http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf.

GS1 Discovery, Configuration, & Initialisation (DCI) Standard for Reader Operations, https://www.gs1.org/standards/epc-rfid

GS1 Reader Management (RM v1.0.1), Ratified Standard, https://www.gs1.org/standards/epc-rfid

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

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

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

3.1
component
identifiable part of a service that provides specific functionality

3.2
data management
device functionality that includes or is a combination of reading, writing, collection, filtering, grouping, and event subscription and notification of RFID tag data to higher level applications and interfaces

3.3
device management
functionality that includes or is a combination of monitoring and control of discovery, configuration, performance and diagnosis of one or more RFID interrogators

3.4
endpoint
component (3.1) that implements or exposes an interface to other components or uses the interface of another component

3.5
implementation
software and hardware that provides the reduction to practice of particular functionality

3.6
interrogator controller
software capability possibly embodied in a distinct physical device, within the *data management* (3.2) *implementation* (3.5) of the architecture in ISO/IEC 24791-1 and capable of exercising the data, control and management of interrogators over the device interface defined in ISO/IEC 24791-5

3.7
client
network *endpoint* (3.4) that sends messages to and/or receives messages from a service

3.8
hosted service
service with lifecycle under the control of another service

4 Abbreviated terms

For the purposes of this document, the abbreviated terms given in ISO/IEC 19762 and the following shall apply.

AC	Access controller
CAPWAP	Control and provisioning of wireless access pints
DCI	Discovery, configuration, initialization
DPWS	Devices profile for web services standard
IETF	Internet engineering task force
LLRP	Low level reader protocol
MIB	Management information base
MIB-II	Management information base version 2
RDMP	RFID device management profile
RFC	Request for comment
RM	Reader management
SNMP	Simple network management protocol
SOAP	Simple object access protocol
SSI	Software system infrastructure

UML	Unified modelling language
URI	Uniform resource identifier
URL	Uniform resource locator
WTP	Wireless termination point
FUS	RDMP firmware update service
MS	Management service
MNS	Monitoring service

5 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.

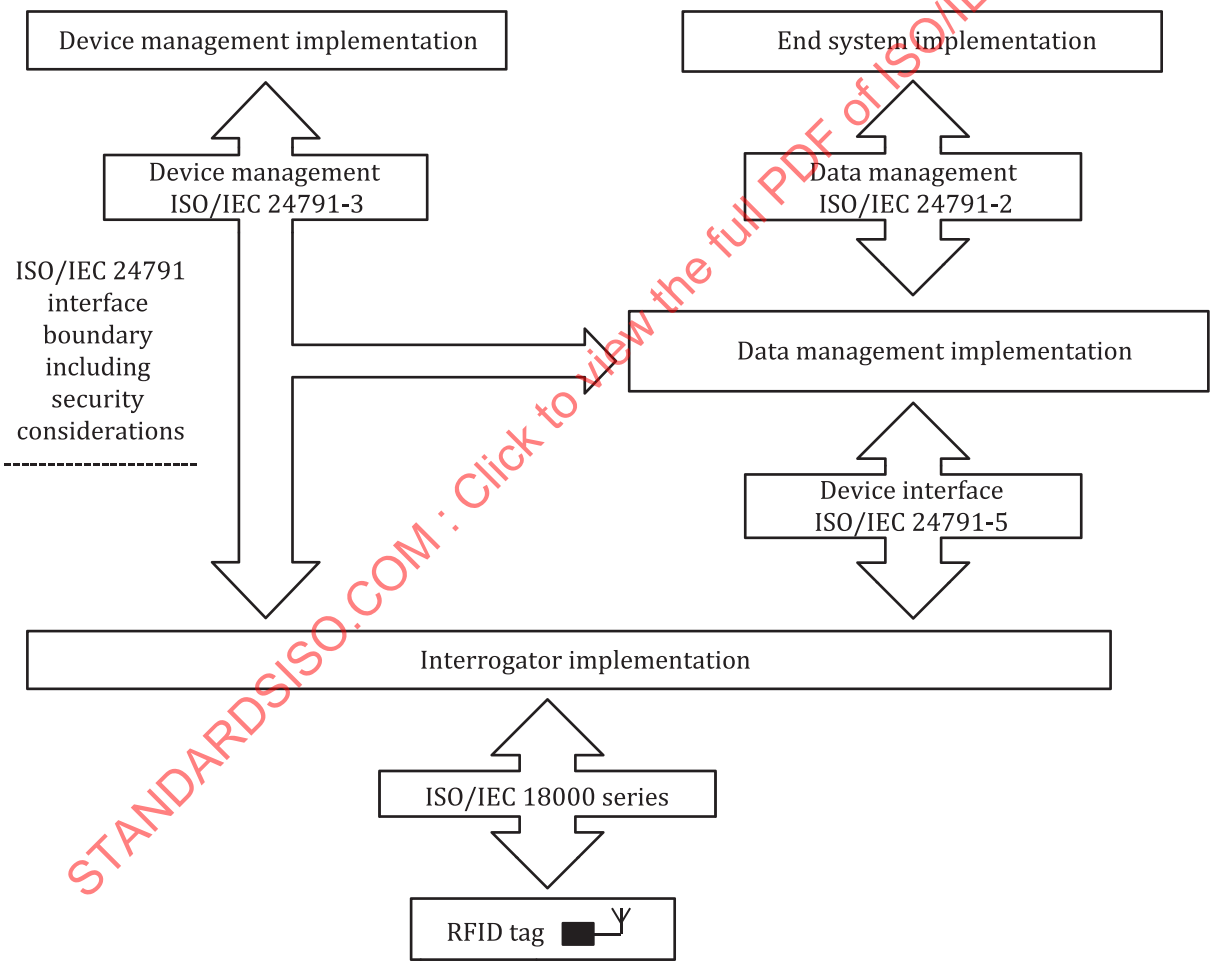

Figure 1 — Architecture overview including relationships to other RFID standards

The parts of the ISO/IEC 24791 series that define Data Management (i.e. ISO/IEC 24791-2), Device Interface (i.e. ISO/IEC 24791-5), and Device Management (i.e. this document) each provide one or more interfaces which 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.

Device management is concerned with providing discovery, configuration, initialization, performance monitoring and diagnostics of software system infrastructure components and interrogators. As shown in Figure 1, device management defines *interfaces* that provide pairwise communications between interrogator implementations, data management implementations and device management implementations.

In addition to defining interfaces for providing configuration and control of the implementations in the network, Device Management may also define requirements for basic initial operation of interrogators, particularly related to initialization in networked environments. This is necessary in order to achieve the SSI goal of providing scalable deployment and management of large numbers of interrogators in a system.

Although Figure 1 depicts the Device Management Implementation residing outside of the boundary of the SSI, the Device Management Implementation may be implemented within any device in a system. For example, it may reside within a standalone network management application or it may be just one component within a device that is also providing a Data Management Implementation. It may also be one component of an application that is also providing the End System Implementation. As with all other components of the SSI as defined in ISO/IEC 24791-1, the platform on which the standard interfaces are implemented is not important; it is conformance to the interfaces and procedures defined in the ISO/IEC 24791 series that is important. Examples of different deployment models of this document are provided in Annex A.

6 UML modelling

Although Figure 1 provides a general overview of the relationship between the interfaces and implementations in the SSI, UML is used for the figures in this document to graphically represent the organization and operation of the device 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 can exist in standards compliant solutions, respectively. Refer to ISO/IEC 24791-1 for a more complete description of how UML is used in other parts of the ISO/IEC 24791 series.

7 Device management

7.1 Architecture

Device management defines the *interface(s)* that provide discovery, configuration, initialization, performance monitoring and diagnostics of software system infrastructure components and interrogators. Device management also defines a set of standardized operational procedures that must be executed by conforming devices, typically related to the initial operation of a device in a networked environment.

Specific device management interface capabilities are provided by a device management services endpoint. A device management client endpoint accesses the services endpoint in a component that provides the desired service(s). Figure 2 provides the representation of the device management interface in a component.

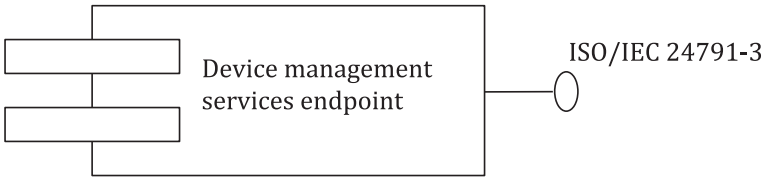

Figure 2 — Device management representation

The software programs that provide device management client and services endpoints can reside within any of the Implementations that can exist in the SSI, as shown in Figure 1. This document does define requirements on how the implementations are developed or packaged within computing or network platforms; requirements are only defined for the operation that is provided.

Device management is distinct from the data and control interfaces provided by ISO/IEC 24791-5 and ISO/IEC 24791-2, respectively. It is possible that the implementation of the device management interface utilizes the same network interface as the implementation of one of the data and/or control interfaces in the implementation. It is also possible that for a specific operation or interface, a component can be both a client and services endpoint, essentially resulting in peer-to-peer operation or a negotiated client/server relationship. This does not change the architecture defined in this document or in ISO/IEC 24791-1.

The functions covered by device management can be grouped and defined as follows:

— Discovery: the process of automatically finding components and devices in a system as well as dynamically identifying service endpoints and enabling connections between the components and services.

— Configuration: the process of setting operational parameters for components that are loaded at system initialization and that change relatively infrequently, primarily through user interaction.

— Initialization: the process of providing initial deployment of network and operating parameters for interrogators as well as installing, updating, maintaining software images at desired versions through a dynamic, potentially automated process.

— Monitoring: the gathering of statistics and state data useful for determining the historic and current operational state of a component, in particular an interrogator or an SSI component that provides a data management implementation function, such as an interrogator controller within the data management implementation depicted in Figure 1.

— Diagnostics: the mechanism to aid in the detection and isolation of faults or abnormal operation within a component of the software system infrastructure. Where the diagnostics involve the computing platform, they are applicable to an interrogator only. Diagnostic capabilities can be defined for other SSI software components, but diagnostic capabilities for general purpose computing platforms will not be defined.

The interfaces defined by this document will provide extension mechanisms to allow implementations to expose management services beyond those specifically defined in this document. This is consistent with standards-based approaches currently used in the management of telecommunication devices.

It is important to note that not all of the above capabilities are required to be deployed in all implementations of a device management services endpoint. For example, interrogators may implement and expose a different set of ISO/IEC 24791-3 capabilities from data management implementations. Furthermore, different classes of interrogators may implement and expose different sets of ISO/IEC 24791-3 capabilities. Conformance requirements for implementations of the device management services endpoint are defined in Clause 6.

8 DCI and SNMP interface set

8.1 Discovery, configuration and initialization conformance group

8.1.1 General

Conforming devices implement discovery, configuration and initialization capabilities through the implementation of the protocols and procedures defined in this conformance group. This subclause of this document references the GS1 EPCglobal DCI for reader operations standard for the normative requirements for this SSI capability. The GS1 EPCglobal DCI standard, references the IETF CAPWAP standard for the core network protocol, security, and communication operations and interfaces.

8.1.2 Interrogator implementations

Interrogators that conform to this document for discovery, configuration and initialization capabilities shall implement all requirements, indicated with "shall", for the *Reader* function as defined in the GS1 EPCglobal DCI standard. Conforming implementations may implement any requirements for the Reader function indicated with "may" in the GS1 EPCglobal DCI standard.

8.1.3 Device management implementations

Device management implementations that conform to this document shall implement all requirements, indicated with "shall" for the AC function as specified in the GS1 EPCglobal DCI standard. Conforming implementations may implement any requirements indicated with "may" in the GS1 EPCglobal DCI standard.

It is not required that implementations of the access controller also implement the *RO Client* function, which is equivalent to the ISO/IEC 24791-5 device interface client functionality, although it is possible and likely that the implementations will be co-resident in computing or network systems. Note that in such cases, the device management implementation and data management implementation from Figure 1 will coexist in the same device. This example is demonstrated in Annex A.

8.2 Performance monitoring and diagnostics conformance group

8.2.1 General

Performance monitoring and diagnostic information access within of SSI components is provided by device management services endpoints that expose SNMP MIBs within one or more of the implementations defined in ISO/IEC 24791-1 and illustrated in Figure 1. SNMP clients (client endpoints in the SSI architecture) access the exposed device management information using the SNMP. Implementations claim conformance to one or more MODULE_COMPLIANCE statements within the specific SNMP MIBs normatively referenced in the following subclauses.

Conformance requirements for implementations that expose an SNMP MIB for performance monitoring and diagnostic information access according to this document are defined in the following subclauses. It is not required that an implementation implement or claim conformance to both of the following subclauses if it claims conformance to one of them.

8.2.2 Interrogator implementations

The GS1 EPCglobal reader management specification Version 1.0.1 defines an SNMP MIB for performance monitoring and diagnostic information access for Interrogator Implementations.

The MIB groups specified as MANDATORY-GROUPS in the SNMP MODULE-COMPLIANCE statement referenced in the GS1 EPCglobal Reader Management Version 1.0.1 specification shall be implemented by interrogators that claim conformance to this subclause.

Implementation of non-SNMP bindings or transports described in the GS1 EPCglobal Reader Management standard is not required by this document.

In addition, the network-attached devices in which the interrogator implementations execute shall implement:

a) the MIB-II System Group, defined in the SNMPv2-MIB module in RFC 3418;

b) the MIB-II IP Group, defined in the IP-MIB module in RFC 2011;

c) the MIB-II Interfaces Group, defined in the IF-MIB in RFC 2863.

8.2.3 Data management implementations providing interrogator controller functionality

Annex B provides an SNMP MIB for performance monitoring and diagnostic information access of data management implementations that implement a device interface (see ISO/IEC 24791-5) client endpoint for the control and data access of interrogators. These implementations have been defined as *interrogator controllers*.

The MIB groups specified as MANDATORY-GROUPS in the SNMP MODULE-COMPLIANCE statement in Annex B shall be implemented by interrogator controller functions within data management implementations that claim conformance to this subclause. Note that other functions that may be implemented by a data management implementation, such as the data management services endpoint may provide performance monitoring and diagnostic information access by additions the MIB in Annex B in a future version of this document.

In addition, the network-attached devices in which the data management implementations execute shall implement:

a) the MIB-II System Group, defined in the SNMPv2-MIB module in RFC 3418;

b) the MIB-II IP Group, defined in the IP-MIB module in RFC 2011;

c) the MIB-II Interfaces Group, defined in the IF-MIB in RFC 2863.

9 RDMP interface set

9.1 General

The following subclauses define the RDMP interface set.

A conforming RDMP implementation shall implement DEVICE as defined in DPWS

A conforming RDMP implementation may implement the FUS. If it does implement FUS, it shall implement the mandatory requirements of the firmware update service

A conforming RDMP implementation may implement the MS. If it does implement MS, it shall implement the mandatory requirements of the management service.

A conforming RDMP implementation may implement the MNS. If it does implement MNS, it shall implement the mandatory requirements of the monitoring service.

9.2 XML namespace

In addition to the namespaces defined in DPWS, this document defines the following XML namespace:

https://standards.iso.org/iso/24791/-3/2013/01/rdmp

Table 1 lists XML namespaces that are used in this document. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 1 — XML namespaces

Prefix	XML namespace	Specification
Rdmp	https://standards.iso.org/iso-iec/24791/-3/ed-2/en/rdmp/	ISO/IEC 24791-3
Dpws	http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01	DPWS
Soap	http://www.w3.org/2003/05/soap-envelope	DPWS
Was	http://www.w3.org/2005/08/addressing	DPWS

9.3 Device discovery

A conformant RDMP device shall implement DEVICE, as defined in DPWS.

A conformant RDMP device shall advertise the rdmp: ISO/IEC 24791-3 type in discovery Hello and Probe Match messages.

A conformant RDMP device will have dpws:device and rdmp:ISO24791-3 in the Types section of discovery Hello and Probe Match message.

A transport address may be sent by an RDMP device in the Hello and Probe Match messages defined in WS-Discovery.

NOTE DPWS uses WS-Discovery as the device discovery protocol. WS-Discovery defines a multicast discovery protocol. Clients discovering RDMP devices joins the multicast group defined in WS-Discovery and discovers RDMP devices on the network from Hello multicast message or a Probe Match unicast response message in response to a Probe multicast message from the client, where the RDMP devices advertises dpws: device and rdmp:ISO24791-3 in the Types section of the Hello or Probe Match message.

9.4 Device metadata

9.4.1 General

This document does not specify requirements for exchanging device metadata in addition to those already specified in DPWS.

NOTE 1 DPWS defines a standard mechanism for retrieving device metadata from a device. Metadata includes information such as manufacturer name, model name, firmware version, etc. This mechanism is documented in the Description Section in the DPWS spec. This document describes it briefly for illustration in 9.4.2 EXAMPLES 1 and 2.

NOTE 2 An RDMP client interested in getting metadata about a RDMP device would send a SOAP envelope containing a WS-Transfer Get message to the transport address of the chosen device. The RDMP device then sends a WS-Transfer GetResponse message containing the device metadata.

NOTE 3 Refer to Devices Profile for Web Services Version 1.1 section titled "Description" for more details and requirements.

9.4.2 Service discovery

An RDMP conformant device may advertise services that are not specified in this document in the dpws:Relationship/dpws:Host/dpws:Types.

NOTE 1 In addition to device discovery, DPWS specifies mechanisms for discovery of hosted services on a device. Some examples of hosted services are a stock quote service, firmware update service, a print service, a calendar service etc. An RDMP client discovers a hosted service by parsing the Metadata section of the WS-Transfer GetResponse.

EXAMPLE 1 An example response that advertises a printer service is:

```
<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
```

```
    xmlns:mex="https://schemas.xmlsoap.org/ws/2004/09/mex"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01"
>
...
<mex:MetadataSection
Dialect="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/Relationship">
      <dpws:Relationship Type="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/host">
        <dpws:Hosted>
            <wsa:EndpointReference>
                <wsa:Address>http://192.168.0.101:80/SamplePrintService0</wsa:Address>
            </wsa:EndpointReference>
            <dpws:Types
                xmlns:spt="http://example.com/wsdp/sample/print">
                spt:PrinterServiceType
            </dpws:Types>
            <dpws:ServiceId>
                http://example.com/sample/print/PrintService
            </dpws:ServiceId>
        </dpws:Hosted>
    </dpws:Relationship>
</mex:MetadataSection>
</wsoap12:Envelope>
```

EXAMPLE 2 An example for advertising ISOIEC24791-5 (LLRP) service by RDMP devices is:

```
<mex:MetadataSection
Dialect="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/Relationship">
      <dpws:Relationship Type="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01/host">
        <dpws:Hosted>
            <wsa:EndpointReference>
                <wsa:Address>ISOIEC24791-5://192.168.0.101:5084</<wsa:Address>
            </wsa:EndpointReference>
            <dpws:Types
                ISOIEC24791-5
            </dpws:Types>
            <dpws:ServiceId>
                http://example.com/sample/ISO24791-5/
            </dpws:ServiceId>
        </dpws:Hosted>
    </dpws:Relationship>
</mex:MetadataSection>
```

NOTE 2 The types sent in the dpws:Relationship/dpws:Host/dpws:Types element are the portTypes of services that are supported by this device. In contrast, the types element in WS-Discovery messages contains discovery-layer portTypes that are implemented by the device.

9.5 Firmware update service

9.5.1 General

This document describes a firmware update service that is used to initialize and update firmware for RFID devices. The client is responsible for initiating a firmware update. The device downloads the firmware from the location presented by the client.

9.5.2 Firmware update service state machine

The firmware update service on the device shall maintain the states as described in the firmware update state machine. Figure 3 illustrates the state machine for a firmware update service.

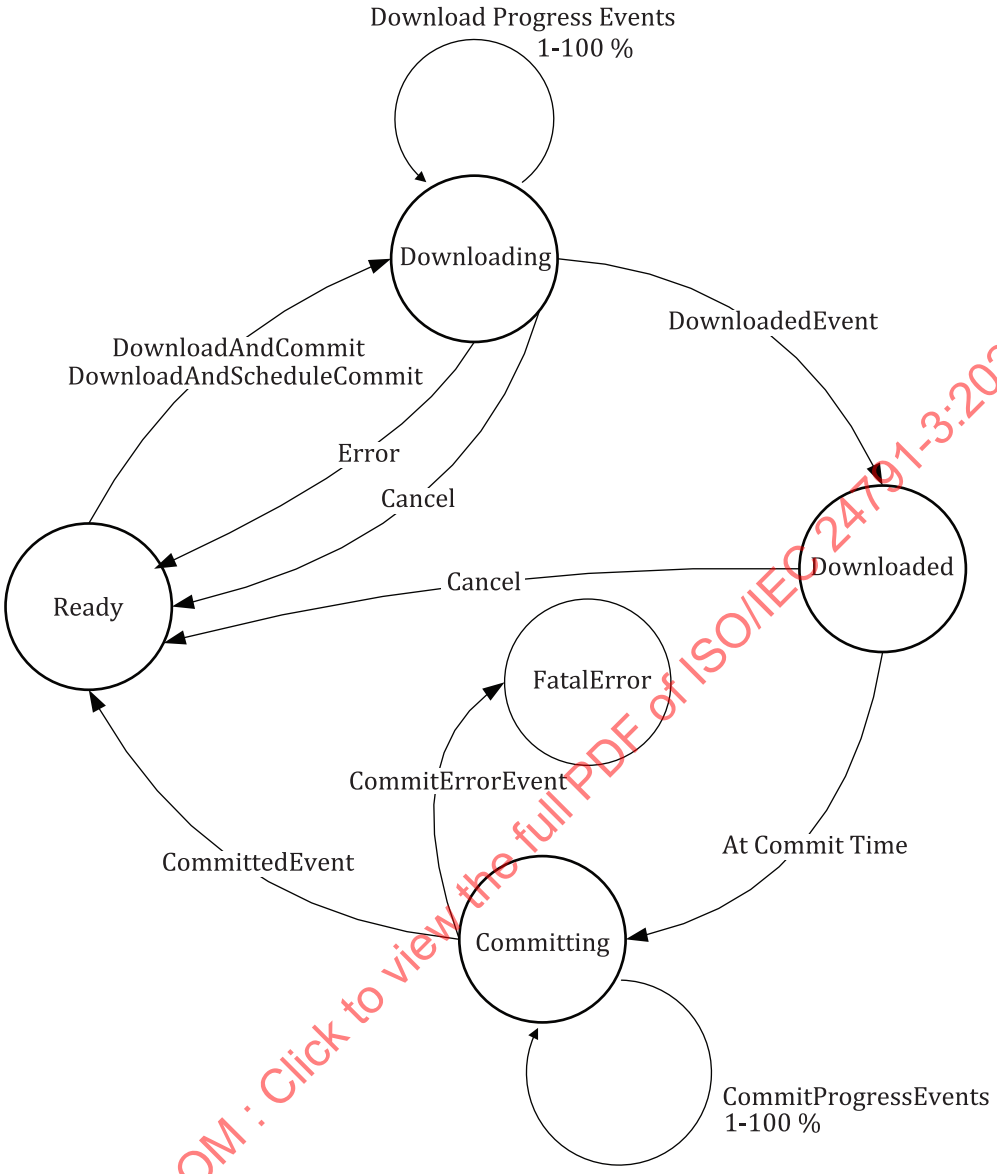

Figure 3 — Service state machine

The firmware states in Figure 3 are explained as follows:

— Ready: The ready state is the initial state. The firmware service state is ready if the service is not downloading or committing firmware currently.

— Downloading: The service is currently downloading firmware to the device.

— Downloaded: Firmware was successfully downloaded to the device. The firmware may be immediately committed or scheduled for a commit based on the client request.

— Committing: Firmware is currently being applied to the target.

— Fatal error: There was an error during commit of firmware and the RF module is now unusable.

9.5.3 FUS Operations

9.5.3.1 OperationNotPermittedInCurrentState

9.5.3.1.1 General

Any of the operations defined in FUS can return the **OperationNotPermittedInCurrentState** fault. This fault is sent when the FUS is not in one of the allowed states for this operation.

Table 2 lists the details for an operation fail in current state.

Table 2 — Fault details for operation fail

Field	Value
[Code]	soap:Receiver
[Subcode]	rdmp:OperationNotPermittedInCurrentState
[Reason]	The service is not in one of the allowed states for this operation
[Detail]	Current state of the service. Allowed states for this operation

9.5.3.1.2 DownloadandCommit

This method shall be supported by the FUS.

If this operation is invoked in a state other than Ready, the FUS shall return OperationNotPermittedinCurrentState error. Once the FUS responds successfully to this operation, it shall move into the Downloading state.

NOTE It is best practice to invoke CheckFirmwareApplicability before invoking the DownloadandCommit method.

9.5.3.1.3 Request Elements [Method Parameters for DownloadandCommit] — FirmwareLocation

The URL from where the firmware is available for download.

The FUS shall support downloading firmware using HTTP. The FUS can support downloading firmware using HTTPS.

If the URL has an unsupported scheme, the FUS shall return an UnsupportedFirmwareLocationScheme fault. If the URL is invalid for other reasons, the FUS may return the InvalidRequestElement fault. The [Reason] element may have additional text to explain why the URL is invalid. Table 3 lists the details for the Firmware location.

Table 3 — Details for FirmwareLocation

Field	Value
[Code]	soap:Sender
[Subcode]	rdmp:UnsupportedFirmwareLocationScheme
[Reason]	The supplied FirmwareLocation has an unsupported URI scheme.
[Detail]	*<SupportedUriSchemes>* *<!-- One or more supported URL schemes-->* *<SupportedUriScheme>HTTP</SupportedUriScheme>* *<SupportedUriScheme>HTTPS</SupportedUriScheme>* *...* *</SupportedUriSchemes>*

Faults defined in this method can be generated by the FUS just by examining the FirmwareLocation URL syntactically. FUS provides another mechanism to address other errors such as the FirmwareLocation being unreachable. Since the firmware download is attempted in the Downloading state, if download fails, the FUS shall move into the Ready state. The error may be recorded by the FUS and reported in a GetStatus operation.

The device can use HTTP Range headers to control the rate at which the download happens. The heuristics used to determine the size of the range and timing of (partial) HTTP(S) GET requests is left to the device.

In case of error prone connections, range headers may be used to resume a download. It is the device's responsibility to determine the resume location in the firmware update file.

When the FUS does a partial HTTP GET, it shall use “bytes" as the range unit.

As a best practice, this document recommends the following: the HTTP server that serves firmware download requests should support partial HTTP GET using the Range header. It should support “bytes” as a Range Unit. It is the responsibility of the HTTP Server serving the firmware download request to specify and implement any policies with respect to the valid duration of the URL.

9.5.3.2 Post download

Once download is completed successfully, the FUS may raise a DownloadedEvent.

Once download is completed successfully, the FUS shall move into the Downloaded state.

The FUS is recommended to validate the firmware binary prior to committing. The mechanism used for validation is out of scope for this document. If the binary is found to be invalid, the FUS may record the error and report this in a GetStatus operation.

The FUS is expected to commit the firmware immediately once the download is complete.

NOTE The HTTP Server for FUS can be implemented in the same host as the RDMP client or hosted on a remote server.

9.5.3.3 DownloadAndScheduleCommit

9.5.3.3.1 General

This method may be supported by the FUS.

If this operation is invoked in a state other than Ready, the FUS shall return OperationNotPermittedinCurrentState error. Once the FUS responds successfully to this operation, it shall move into the Downloading state.

9.5.3.3.2 Request elements [Input Parameters]

The request elements are as follow.

— FirmwareLocation: Please see the DownloadandCommit section for description of requirements.

— AtTime: This is the date & time at which to commit the loaded firmware. If the AtTime is in the past, the FUS shall return an InvalidRequestElement error. Additional text may be added to the [Reason] element of the SOAP fault.

— PostDownload: Once teh download is completed successfully, the FUS may raise a DownloadedEvent.

  Once the download is completed successfully, the FUS shall move into the Downloaded state.

The FUS is recommended to validate the firmware binary prior to committing. The mechanism used for validation is out of scope for this document. If the binary is found to be invalid, the FUS may record the error and report this in a GetStatus operation.

9.5.3.4 CheckFirmwareApplicability

9.5.3.4.1 General

This method requests the device firmware update service to checks if specified firmware is applicable for the device. This method shall be supported by the FUS.

9.5.3.4.2 Request elements [Method Parameters] — FirmwareVersionToCompare

The FirmwareVersion is the version of the firmware that the client wishes to compare with the committed firmware version on the FUS.

This document does not mandate any specific requirements for the FirmwareVersion string. Implementations are recommended to construct a string that enables the device to determine:

a) if the firmware being supplied is applicable to the FUS;

b) if the firmware being supplied is newer or older.

9.5.3.4.3 Response elements

The method returns one of the following:

— **Newer:** firmware is applicable to FUS and is more recent than the currently installed version;

— **Older**: firmware is applicable to FUS and is less recent than the currently installed version;

— **SameVersion:** firmware is applicable to FUS and is the same version as the currently installed version;

— **NotApplicable:** firmware is not applicable for the device or the supplied string is invalid or the FUS is unable to parse the string.

9.5.3.5 GetStatus

9.5.3.5.1 General

This method returns current state of firmware update service on the device. This method shall be supported by the FUS.

9.5.3.5.2 Response elements

The response shall contain the field **State,** one of Ready, Downloading, Downloaded, Committing or FatalError.

The response can contain the following fields:

a) % downloaded: if the state is Downloading, the FUS may return the percentage of firmware downloaded.

b) % committed: if the state is committing, the FUS may return the percentage committed.

c) Error: the FUS may persist an error string from a previous operation or error situation.

d) TimeofError: the time at which the last error occurred.

e) Description: a localizable string, it applies to any state.

NOTE If, there is an error while committing firmware and the service goes into the Ready state, the error can be stored in the service to be retrieved later by the client. The error string can also be used to diagnose errors encountered by the FUS.

9.5.3.6 Cancel

9.5.3.6.1 General

This method may be supported by the FUS.

9.5.3.6.2 Cancel description

If this operation is invoked in the Ready state, it may be ignored. If the FUS is in the Downloading or Downloaded states, it shall move to the Ready state. In the Committing state, the FUS may reject the Cancel operation by returning the CancelNotPermittedNow error. Table 4 lists the details of the cancel method.

Table 4 — Details of the cancel method

Field	Value
[Code]	soap:Receiver
[Subcode]	rdmp: CancelNotPermittedNow
[Reason]	Cancel operation not permitted at this point.
[Detail]	None

9.5.3.7 Events from FUS

9.5.3.7.1 General

The FUS shall implement eventing as defined in the DPWS requirements for eventing (see section titled "Eventing" in Devices Profile for Web Services Version 1.1).

9.5.3.7.2 DownloadProgressEvent

DownloadProgressEvents may be generated only in the downloading state. The parameter that describes DownloadProgressEvent is: PercentCompleted. The percentage of download completed may be provided with the DownloadProgressEvent by the service. If provided, this shall be an integer between 0 and 100.

9.5.3.7.3 DownloadedEvent

DownloadedEvent may be generated when the service transitions from the Downloading state to the Downloaded state.

9.5.3.7.4 CommitProgressEvent

CommitProgressEvents may be generated only in the Committing state. The parameters that describe CommitProgressEvent are:

— PercentCompleted: The percentage of update completed may be provided with the CommitProgressEvent by the service. If provided, this shall be an integer between 0 and 100.

— Description: This is a descriptive localizable string about the status of the commit.

9.5.3.7.5 CommittedEvent

CommittedEvent may be generated when the service transitions from the Committing state to the Ready state.

9.5.3.7.6 CommitErrorEvent

If there is an error during firmware commit, it may be reported using the CommitErrorEvent. The parameters that describe CommitErrorEvent are:

— Description: This is a localizable string containing a description of what went wrong during the firmware commit.

— Time: This is the time at which this particular error took place.

9.5.3.7.7 DownloadErrorEvent

If there is an error during firmware download, it may be reported using the DownloadErrorEvent.

9.6 Management service

9.6.1 General

The MS provides the following functionality:

a) configuring a device with standard properties; in this document, the ability to configure ISO/IEC 24791-5 properties on the device is included;

b) implementing common operations like device reboot and reset to factory settings.

The MS specifies a set of configuration properties that are commonly understood among RFID device implementations. The MS also allows for custom configuration properties to be retrieved and applied on devices. The MS also provides a mechanism to advertise metadata for configuration properties.

Each configuration property has a unique identifier and a value associated with the identifier.

9.6.2 Property identifier

A property identifier has two parts:

a) group URI (e.g. https://standards.iso.org/24791-3/rdmp/Configuration/RF), which acts as a namespace;

b) property name (e.g. "power level").

A property name shall be of type token as defined in XML Schema Part 2.

NOTE A token represents tokenized strings. The value space of token is the set of strings that do not contain the carriage return (#xD), line feed (#xA) nor tab (#x9) characters, that have no leading or trailing spaces (#x20) and that have no internal sequences of two or more spaces. The lexical space of token is the set of strings that do not contain the carriage return (#xD), line feed (#xA) nor tab (#x9) characters, that have no leading or trailing spaces (#x20) and that have no internal sequences of two or more spaces.

A property identifier used by MS shall NOT have null value for either the group URI or the property name.

9.6.3 Property value metadata

Metadata of the property value is made up of the following.

— **Description:** Friendly description of the property. It is of type string. A MS shall include description in the metadata.

— **Writeable**: A property is writeable if it may be modified on the device. It is of type Boolean. MS may include this field in the metadata. If this is not specified, the client may assume the property to be writeable.

— **Persisted**: A property is persisted if the device stores the property in permanent storage. It is specified as a Boolean. The MS may include this in the metadata. If this is not specified, the client may assume the property to be persisted.

— **RestartRequired:** The device requires a restart for the change in the property to take effect; this is specified as a Boolean. The MS may include this in the metadata. If this is not specified, the client may assume that restart is not required.

— **Type**: The type of the property, indicating if the property is of the type string, int, boolean or some other complex type. The MS shall include the type in the metadata. The type shall be a XML QName, as defined in Reference [18].

— **DefaultValue:** The value that the device uses if the client has not modified. It may be included. If it is included, it shall conform to the Type specified in the metadata.

NOTE The type metadata is critical for clients to interpret and process configuration data sent by the device.

9.6.4 Device property profile

A property profile is a collection of properties.

NOTE Using a collection of properties is useful in many scenarios. Management software can wish to bring the device to a base configuration when it is first discovered. When a device changes role, role specific configuration can be required to be applied to the device. Using a property profile enables device and clients to store a collection of properties together.

9.6.5 Sources and source types

Sources in RDMP may refer to the device or the antennas or physical ports connected to the device. The standard source types are antenna, GPI, GPO and barcode.

9.6.6 MS operations

9.6.6.1 GetSources

9.6.6.1.1 General

This method is invoked by the client to get the list of sources on the device and the types of the sources. This method shall be supported by MS.

9.6.6.1.2 Request elements

There are no request elements.

9.6.6.1.3 Response elements

This method returns a collection of the following tuples. Each tuple shall have:

— **SourceName,** of type token, as defined in Reference [18];

— **Type:** source type.

9.6.6.2 GetAllPropertyMetadata

9.6.6.2.1 General

This method shall be supported by MS. This is the mechanism for a client to obtain a list of supported properties and property metadata.

GetAllPropertyMetadata is the key method for a client to discover device and source related configuration. It is recommended that the Management Service provide comprehensive metadata information about all the properties configurable via RDMP. Metadata enables the management software to correctly display configuration (e.g. a property can be shown as disabled in the user interface if it is not writeable). It also enables management software to warn the user to reboot the device if the property change requires a restart to take effect.

9.6.6.2.2 Request elements

SourceName is of type string. This field is optional.

If the SourceName parameter is not specified, this method shall return the metadata associated with all the properties of the device.

If the SourceName parameter is specified, this method shall return the metadata associated with all the properties of the specified source.

9.6.6.2.3 Response elements

The method returns a collection of property identifiers and property value metadata.

9.6.6.3 GetPropertyMetadata

9.6.6.3.1 General

This method may be supported by MS.

9.6.6.3.2 Request elements

This method takes property identifier and a source name. The SourceName is an optional parameter.

9.6.6.3.3 Response elements

If the SourceName parameter is not specified, this method shall return the property metadata associated with the specified property of the device.

If the SourceName parameter is specified, this method shall return the property metadata associated with the specified property of the source.

9.6.6.4 GetAllPropertyValues

9.6.6.4.1 General

This method may be supported by MS.

9.6.6.4.2 Request elements

SourceName is an optional parameter.

9.6.6.4.3 Response elements

If the SourceName parameter is not specified, this method shall return a property profile of the device.

If the SourceName parameter is specified, this method shall return a property profile of the source.

9.6.6.5 GetPropertyValue

9.6.6.5.1 General

This method may be supported by the MS.

9.6.6.5.2 Request elements

This method takes an optional SourceName and a PropertyIdentifier as input.

9.6.6.5.3 Response elements

This method returns a PropertyValue.

If the SourceName parameter is not specified, this method shall return the property value associated with the specified property of the device.

If the SourceName parameter is specified, this method shall return the property value associated with the specified property of the source.

9.6.6.5.4 Errors

If the property is not supported by the MS, it shall return rdmp:PropertyNotSupported fault as defined Table 5. Table 5 lists the details of an error.

Table 5 — Error details

Field	Value
[Code]	soap:Sender
[Subcode]	rdmp:PropertyNotsupported
[Reason]	The requested property is not supported.
[Detail]	PropertyIdentifier

9.6.6.6 SetPropertyProfile

9.6.6.6.1 General

This method shall be supported by MS.

9.6.6.6.2 Request elements

This method takes a PropertyProfile as an input and an optional SourceName parameter.

If the SourceName parameter is not specified, this method shall set the property profile on the device.

If the SourceName parameter is specified, this method shall set the property profile on the source.

9.6.6.6.3 Response elements

There are no response elements.

9.6.6.6.4 Errors

If the property is not writeable, the MS shall return rdmp:PropertyNotWriteable fault as defined Table 6. Table 6 lists the details of an error.

Table 6 — Error details

Field	Value
[Code]	soap:Sender
[Subcode]	rdmp:PropertyNotWriteable
[Reason]	The requested property is not writeable.
[Detail]	*PropertyIdentifier*

If the MS service was unable to apply all properties, the MS shall return rdmp:UnableToApplyAll Properties fault as defined in Table 7. MS may include more details about the error in the reason. The detail may contain the PropertyProfile of the actual Properties the MS was unable to apply.

Table 7 — Error details

Field	Value
[Code]	soap:Sender
[Subcode]	rdmp:UnableToApplyAllProperties
[Reason]	*The reason for being unable to apply the property.*
[Detail]	*PropertyProfile*

9.6.6.7 GetPropertyValuesByGroup

9.6.6.7.1 General

This method may be implemented by MS.

9.6.6.7.2 Request elements

This method takes an xs:anyURI as GroupURI and an optional parameter SourceName.

9.6.6.7.3 Response elements

If the SourceName parameter is not specified, this method shall return a property profile for the specified property group of the device.

If the SourceName parameter is specified, this method shall return a property profile for the specified property group of the source.

9.6.7 Standard properties

The following group names and properties shown in Table 8 shall be supported by the management service.

GroupName URI: https://standards.iso.org/24791-3/rdmp/configuration/General

Table 8 — General configuration properties

Property name	Type	Property editability	Applicable target	Notes
Name	Token	W	A, D	Friendly name of the source/device
ReaderRegulatoryRegion	Token	R	D	Legal regulatory region of the device
Key				
R : Read only				
W : Read-write				
A : Antenna				
D : Device				

The RF properties shown in Table 9 may be supported by the management service.

GroupName URI: https://standards.iso.org/24791-3/rdmp/configuration/RF

Table 9 — RF configuration properties

Property name	Type	Property editability	Applicable target	Notes
AirProtocolsSupported	Tokens	R	A, D	Protocols (tag types) supported by this device/source
PowerLevel	Float	W	A	Percentage that designates the antenna's power setting for reading tags
OperationEnvironment	Token	R	A, D	Reader operation mode. Example values are: single, multiple and dense.
Key				
R : Read only				
W : Read-write				
A : Antenna				
D : Device				

The group names and properties shown in Table 10 may be supported by the management service.

GroupName URI: http://standards.iso.org/24791-3/rdmp/configuration/ISO24791-5

NOTE RDMP clients are allowed to configure certain properties and capabilities of 24791-5 conformant devices. The reason this is allowed is to enable management software to set default configuration values that are not specific to a LLRP client session. One cannot expect LLRP clients to set all default values every time they connect. In terms of capabilities, the ability to get this information by RDMP clients helps enterprise management software provide data for reporting applications. It is noteworthy to mention that when a LLRP client sets a value during a session and the RDMP client sets the same value, the new value does not take effect until a reboot is performed.

The ISO 24791-5 properties (capabilities and configuration) shown in Table 10 may be supported by the management service.

Table 10 — ISO/IEC 24791-5 properties

Property name	Type	Property editability	Applicable target	Notes
LLRPStatus	LLRPStatus	R	D	—
GeneralDeviceCapabilities	GeneralDeviceCapabilities	R	D	—
RegulatoryCapabilities	RegulatoryCapabilities	R	D	—
Antenna Properties	AntennaProperties	W	A	—
Antenna Configuration	AntennaConfiguration	W	A	—
Key				
R : Read only				
W : Read-write				
A : Antenna				
D : Device				

Table 10 *(continued)*

Property name	Type	Property editability	Applicable target	Notes
KeepaliveSpec	KeepaliveSpec	W	D	—
EventsAndReports	EventsAndReports	W	D	—
Identification	Identification	R	D	—
LLRPServers	anyURI	W	D	The URI specified used this property should have the URI scheme ISOIEC24791-5. EXAMPLE **ISOIEC24791-5://192.168.0.101:5555/**

Key
R : Read only
W : Read-write
A : Antenna
D : Device

The RDMP properties shown in Table 11 may be supported by the management service.

GroupName URI: https://standards.iso.org/24791-3/rdmp/Configuration/General

Table 11 — General properties

Property name	Type	Property editability	Applicable target	Notes
Location	String	W	A, D	Location of the source
Description	String	W	A, D	
TimeServer	String	W	D	Time server used by the device for time syncs
Role	String	W	D	Role of the device
DeviceEPC	String	R	D	Manufacturer assigned EPC URI
TimeServers	tokens	W	D	Time servers used by the device for time syncs
DHCPServers	Tokens	W	D	DHCP server currently used by the device for DHCP.
Contact	String	W	D	The textual identification of the contact person for this device, together with information on how to contact this person.

Key
R : Read only
W : Read-write
A : Antenna
D : Device

The RDMP properties shown in Table 12 may be supported by the management service.

GroupName URI: https://standards.iso.org/24791-3/rdmp/Configuration/RF

Table 12 — RF properties

Property name	Type	Property editability	Applicable target	Notes
Frequency	Double	W	A	Frequency used by the source in MHz.
EffectiveRange	Float	R	A	Range of antenna in m.

Key
R : Read only
W : Read-write
A : Antenna
D : Device

Table 12 *(continued)*

Property name	Type	Property editability	Applicable target	Notes
AntennaSequence	Tokens	W	D	Sequence of antennas for reading. The order is important.
Key				
R : Read only				
W : Read-write				
A : Antenna				
D : Device				

The RDMP property hown in Table 13 may be supported by the management service.

GroupName URI: https://standards.iso.org/24791-3/rdmp/Configuration/Notification

Table 13 — Notification property

Property name	Type	Property editability	Applicable target	Notes
EventMode	Boolean	W	D	Boolean to indicate if the device can send notifications to the host
Key				
R : Read only				
W : Read-write				
A : Antenna				
D : Device				

9.6.8 Other management operations

9.6.8.1 Reboot

9.6.8.1.1 General

By invoking the reboot method, a client initiates the reboot of a device. This method shall be supported by MS.

9.6.8.1.2 Request elements

There are no request elements.

9.6.8.1.3 Response elements

There are no response elements.

9.6.8.2 ResetToFactorySettings

9.6.8.2.1 General

Using this method, the client instructs the device to reset any settings such as configuration to factory specified defaults. This method shall be supported.

9.6.8.2.2 Request elements

There are no request elements.

9.6.8.2.3 Response elements

There are no response elements.

9.6.8.3 ResetToFactorySettingsExceptNetwork

9.6.8.3.1 General

Using this method, the client instructs the device to reset any settings such as configuration to factory specified defaults EXCEPT the network settings.

This method may be supported.

9.6.8.3.2 Request elements

There are no request elements.

9.6.8.3.3 Response elements

There are no response elements.

9.7 Operation error reporting

9.7.1 General

Mechanisms and artifacts defined in 9.7 are used by operations in all the services defined in this document to communicate errors to the client.

SOAP faults are the defined mechanism to communicate errors. Fault messages are correlated as replies using the [relationship] property as defined in Reference [20].

9.7.2 Common operation error codes

9.7.2.1 General

Operations defined in the specification may return any of the faults defined in Reference [21]. In addition, operations may define additional SOAP faults.

9.7.2.2 Action not supported

When an optional operation of any service defined in this specification is invoked by a client but is not supported by a service it shall generate a wsa:ActionNotSupported fault. The fault is defined in Reference [21].

9.7.2.3 InvalidRequestElement

This fault is sent when a client sends an invalid request element as part of an operation. Table 14 lists the details of the error message.

Table 14 — InvalidRequestElement description

Field	Value
[Code]	soap:Sender
[Subcode]	rdmp: InvalidRequestElement
[Reason]	At least one request elements is invalid
[Detail]	*Name of the invalid request element.*

9.7.2.4 InternalError

This fault is sent when the service encounters an unexpected condition such as out of memory that prevented it from fulfilling the request. Additional text may be appended to the [Reason] element to provide more information about the internal error. Table 15 lists the details of the error message.

Table 15 — InternalError description

Field	Value
[Code]	soap:Receiver
[Subcode]	rdmp: InternalError
[Reason]	The service had an unexpected error
[Detail]	*None*

9.8 Monitoring service

9.8.1 General

The MNS is used to monitor the health of the device. The monitoring service publishes events and statistics to enable RDMP monitoring clients to actively monitor the device health.

9.8.2 Monitoring event structure

Each monitoring event has the following structure.

Fields: Each event contains these standard fields.

— **TimeofOccurrence**: Date and time that the event was generated. This is of type xs:datetime. The TimeofOccurrence field shall be implemented by MNS.

— **Level**: Critical, Error, Warning or Informational. It is of type string that allows only the above values. Level indicates severity of the event. Level shall be implemented by the MNS.

— **Description:** A localizable string that describes the event in friendly terms to the user. This may be implemented.

An event may have additional fields. Additional fields are described in specific events in 9.8.3.

9.8.3 Events

The following events listed in Table 16 may be supported and sent by MNS.

Table 16 — Event details

EventName	Additional fields	Type	Description
BatteryEvent	The percentage of battery remaining field shall be included for the battery event.	Integer	The service shall send a battery event at Informational level upto 15 % of battery left. Between 14 % to 10 % remaining, it shall send a warning level event. Between 9 % to 5 %, it shall send an error level event. Less than 5 % it shall raise a critical level event. Within the same battery level, the frequency of sending events is not defined in this document.

Table 16 *(continued)*

EventName	Additional fields	Type	Description
MemoryEvent	The percentage of free memory remaining on device field shall be included for the memory event.	Integer	The service shall send a memory event at Informational level upto 15 % of memory left. Between 14 % to 10 % remaining, it shall send a warning level event. Between 9 % to 5 %, it shall send an error level event. Less than 5 % it shall send a critical level event. Within the same memory level, the frequency of sending events is not defined in this document.
SourceStateEvent	The SourceName field shall be included.	String	Indicates the state of the source. The level shall be error when down and shall be informational when up.
	The state values are Up or Down. This field shall be included.	enum	
SourceNoiseLevelEvent	The SourceName field shall be included.	String	—
	The Noiselevel field shall be included.	integer	
PowerSupplyEvent (applies to devices that support battery backup)	The state values are Up or Down. This field shall be included.	enum	The down event is raised when power supply to device is cut. The up event when power is restored. The level shall be warning when down, shall be informational when up.
PrinterEvent A printer event, in addition to the standard fields contains an error code and an error string. The possible error codes are:	ErrorCode	—	—
	rdmp:CutterFault rdmp:GapNotFound rdmp:Jam rdmp:OutOfMedia rdmp:NoCurrentTemplate rdmp:RibbonFault	QName	
TagOperationalEvents A tag operation event in addition to the standard fields may contain an error code a TagInfo structure and an error string. The possible error codes are:	ErrorCode	Type	—
	rdmp:TagLocked	QName	Locked when accessing the tag
	rdmp:TagLockFailed		—
	rdmp:InvalidPasscode		
	rdmp:PartiallyLocked		
	rdmp:ReadOnly		Trying to write to a region that is read-only
	rdmp:CorruptTag		—
	rdmp:WriteFailed		
	rdmp:KillFailed		
	rdmp:FailedLock		
	rdmp:FailedErase		
	rdmp:FailedMemoryRead		
	rdmp:FailedRead		

Table 16 *(continued)*

EventName	Additional fields	Type	Description
HardwareErrorEvent This event consists of an error code in addition to the standard fields. The error code is vendor dependent. The error code shall be a QName.	—		

9.8.4 Statistics

9.8.4.1 General

The MNS supports reporting of the following statistics information.

9.8.4.2 GetStatisticsMetadata

9.8.4.2.1 General

The GetStatisticsMetadata method shall be supported by the MNS.

9.8.4.2.2 Request elements

The SourceName is of type string. This field is optional.

If the SourceName parameter is not specified, this method shall return the metadata associated with all the statistics of the device.

If the SourceName parameter is specified, this GetStatisticsMetadata method shall return the metadata associated with all the statistics of the specified source.

9.8.4.2.3 Response elements

This method shall return a collection of StatisticsIdentifier and StatisticsMetadata.

9.8.4.3 GetStatistics

9.8.4.3.1 General

The GetStatistics method shall be supported by the MNS.

9.8.4.3.2 Request elements

The GetStatistics method takes an optional SourceName as input.

9.8.4.3.3 Response elements

This method shall return a StatisticsProfile, which is a collection of StatisticsIdentifier and StatisticsValue.

The GroupName URI for the identifiers used in this service shall be: https://standards.iso.org/24791-3/rdmp/Monitoring/Statistics

Vendors extending statistics shall not use this group name for custom statistics.

9.8.4.4 ResetStatistics

9.8.4.4.1 General

The ResetStatistics method may be supported by the MNS.

9.8.4.4.2 Request Elements

The ResetStatistics method takes an optional SourceName as input.

9.8.4.4.3 Response elements

There are no response elements.

9.8.4.5 Standard statistics

Table 17 lists the standard statistics details.

Table 17 — Standard statistics

Name	Property value type	Notes
BatteryPowerRemaining	Positive integer in the range of 0 to 100	Percentage of battery power remaining. Not resettable.
BatteryTimeRemaining	Positive integer in the range of 0 to 100	Remaining time remaining in minutes. Not resettable.
DeviceUptimeTicks	Positive double	Time in ticks for which the device has been up. Not resettable.
FailedKillCount	Positive integer	—
FailedLockCount	Positive integer	—
FailedReadCount	Positive integer	—
FailedWriteCount	Positive integer	—
KillCount	Positive integer	—
LockCount	Positive integer	—
ReadCount	Positive integer	—
WriteCount	Positive integer	—
MemoryReadCount	Positive integer	—
EraseCount	Positive integer	—
FailedEraseCount	Positive integer	—
PrintedCount	Positive integer	—
FailedPrintedCount	Positive integer	—
TimeSinceLastTagNotification	Integer, in minutes	Enables troubleshooting of why no tag reads for a duration.

9.9 Security

Parties involved in message exchanges defined in this specification fall into the following categories:

a) CLIENT: To elaborate this category covers both clients that invoke operations on the device and clients that act as an Event Sink, i.e. receive events from the device;

b) DEVICE;

c) Firmware Server.

Security between the CLIENT and the DEVICE is covered in the Security section of Devices Profile for Web Services Version 1.1. Firmware is downloaded from the Firmware Server as explained in FUS. This communication may be secured by using a secure transport such as HTTPS.

NOTE Security encompasses the following.

a) Authentication of device and client: Authentication is the process by which the identity of the sender is determined by the recipient. A client can be required to authenticate the devices it can communicate with or vice versa.

b) Integrity and confidentiality of messages exchanged between device and client.

Briefly, transport layer security may be used to provide mutual authentication of client and device as well as the establishment of a secure channel over which messages are exchanged. Integrity and confidentiality are achieved by using the secure channel. The policy assertions carried in the messages exchanged during discovery may contain the client security requirements as well as the security protocols supported by client and device for authentication and establishment of a secure channel. Since discovery precedes setting up of a secure channel, the integrity of discovery messages is protected using message-level signatures, while the integrity of other messages is protected using a secure channel.

9.10 Extensibility

9.10.1 General

Messages defined by this specification may include constructs to be extensible by conforming implementations. Clients or devices that do not understand a custom extension to a type may choose to ignore it.

The XML/XSD notation for types shall be:

```
<xs:complexType name="PropertyMetadata">
        <xs:sequence>
            <xs:element name="Type" type="xs:QName" minOccurs="1" maxOccurs="1"
nillable="false"/>
            <xs:element name="Description" …/>
            …
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
```

The following example shows an extension to PropertyMetadata that conforms to the above schema. The extension adds a new element called lowerBound.

```
<PropertyMetadata xmlns="https://schemas.xmlsoap.org/ws/2008/rdmp" xmlns:xs="http://www.
w3.org/2001/XMLSchema">
  <Type>xs:int</Type>
  <Description>some description</Description>
  …
  <lowerBound xmlns="http://www.example.org/rdmpextensions">5</lowerBound>
</PropertyMetadata>
```

9.10.2 Extending monitoring events

RDMP implementations of the monitoring service may choose to implement custom events outside of the standard specification. This document provides a way to describe a custom event in order for any client to understand and interpret the event.

The XML/XSD notation for extension events shall be:

```
<xs:complexType name="VendorDefinedEvent">
<xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
       <xs:anyAttribute namespace="##other" processContents="lax" />
```

```
</xs:complexType>
```

A similar notation shall be followed for extending statistics for the monitoring service.

Management service and monitoring service can be extended to support custom types as values for properties and statistics. The following example shows how the management service can be extended to support a custom type for properties. Similar changes can be done to monitoring service schema to support custom types for Statistics. In the management service schema, within the PropertyValue complex Type, add the text that is in bold:

```
<xs:complexType name="PropertyValue">
    <xs:choice>
      <xs:element name="String" type="xs:string" />
      <xs:element name="Integer" type="xs:int" />
      <xs:element name="TagDataSelector" type="tns:TagDataSelector" />
      <xs:element name="QName" type="xs:QName" />
      <xs:element name="Boolean" type="xs:boolean"/>
      <xs:element name="Float" type="xs:float"/>
      <xs:element name="Double" type="xs:double"/>
      <xs:element name="Uri" type="xs:anyURI"/>
      <xs:element name="DateTime" type="xs:dateTime"/>
   <xs:element name="NewComplexType" type="myns:NewComplexType"/>
    </xs:choice>
 </xs:complexType>
```

Where “myns” is the prefix defined for the namespace containing the “NewComplexType” xml type.

RDMP services can be extended to support custom defined events. The following example shows how to add a custom event to Firmware Service.

To add a new event for FUS called FirmwareUpdate_CustomEvent, do the following:

a) In the FirmwareUpdate wsdl message definition section, add the following:

```
<wsdl:message name="FirmwareUpdate_CustomEvent_OutputMessage">
    <wsdl:part name="parameters" element="myns:CustomEvent"/>
</wsdl:message>
```

b) In the port binding section of the FirmwareUpdate wsdl, add the following:

```
<wsdl:operation name="CustomEvent">
<wsdl:output wsaw:Action="<Custom Event Namespace>/CustomEvent"
    message="tns:FirmwareUpdate_ CustomEvent _OutputMessage"/>
</wsdl:operation>
```

c) In the soap binding section of the FirmwareUpdate wsdl, add the following:

```
<wsdl:operation name=" CustomEvent ">
            <soap12:operation soapAction="http://customnanespace/CustomEvent"
style="document"/>
     <wsdl:output>
        <soap12:body use="literal"/>
     </wsdl:output>
</wsdl:operation>
```

Where myns is the prefix of the namespace containing the xml schema definition of “CustomEvent”.

Refer Annex C for WSDL and XSD files for the RDMP interface set.

Annex A
(informative)

Implementation examples

A.1 General

There are many possible organizations of the implementations of the software components referenced in this document, including how the components are distributed among different physical computing platforms. This Annex provides informative examples of the different organizations utilizing UML deployment diagrams.

The figures in this Annex do not show any of the data or control interfaces of ISO/IEC 24791-5 and ISO/IEC 24791-2, respectively, that may be implemented on the devices because the implementation-specific logic provides any interaction between the interfaces, making them functionally independent. In certain deployment scenarios, the instances of the device interface or data management services endpoint could receive commands or requests from the device management implementation logic in response to commands or requests from the device management services endpoint. These requests would be for configuration, provisioning, performance monitoring and/or diagnostics, and not for tag or sensor data as defined in this document.

A.2 DCI and SNMP interface set implementation examples

Figure A.1 illustrates a node that implements the device management services endpoint in an interrogator. In this example, services endpoint would conform to Clause 7 by implementing discovery, configuration and initialization capabilities as defined in this document, which references IETF RFC 5415 (CAPWAP)[19] and GS1 EPCglobal DCI. Also illustrated in this example, the services endpoint would conform to Clause 8 by implementing monitoring and diagnostics capabilities as defined in this document which references the GS1 EPCglobal reader management SNMP MIB and the IETF MIBII MIBs.

The separation between the services endpoint and the implementation in Figure A.1 illustrates how the standards-based interfaces are logically distinct from the implementation that provides the management data or achieves the specified device management result.

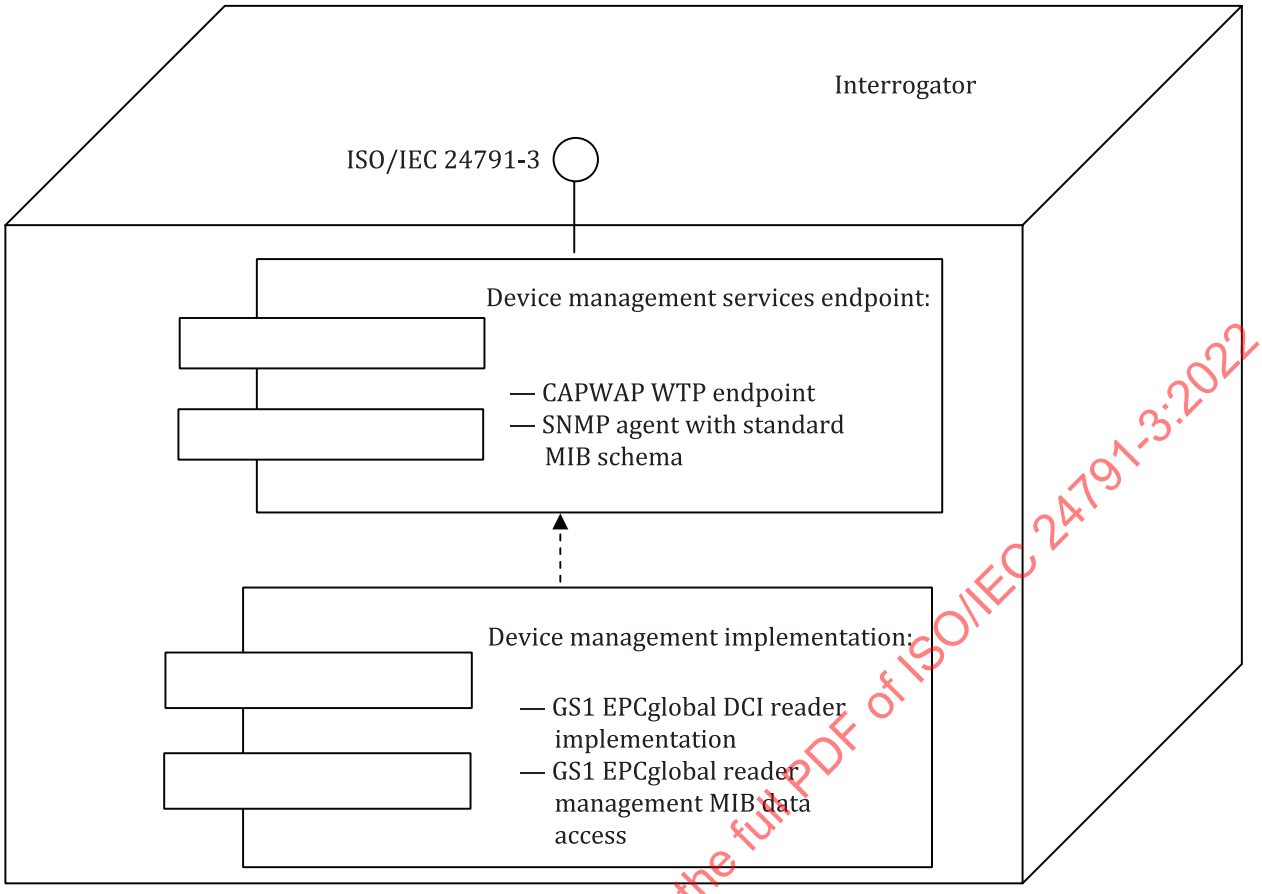

Figure A.1 — Node diagram with DCI and SNMP interface set implementation on interrogator

Figure A.2 illustrates nodes in a system that implements the device management endpoints on an interrogator controller as well as on interrogators. The interrogator controller is one example of a data management implementation as shown in Figure 1. Other examples of deployment models providing conforming implementations are possible.

In this example, the interrogators provide the same services as in the example of Figure A.1.

The services endpoint on the interrogator would conform to Clause 7 by implementing discovery, configuration and initialization capabilities as defined in this document, which references IETF RFC 5415 (CAPWAP)[19] and the GS1 EPCglobal DCI specification. The DCI access controller provides a services endpoint to the interrogators, which, as defined in the specific standards, provide peer-level services back to the interrogator controller.

Also illustrated in this example, the services endpoint would conform to Clause 8 by implementing monitoring and diagnostics capabilities as defined in this document and the referenced IETF MIBII MIBs.

Figure A.2 shows the node diagram with ISO/IEC 24791-3 endpoints on interrogator controller and interrogators. In this deployment diagram (see Figure A.2), there is no device management client endpoint on the interrogator controller to request monitoring and diagnostic device management services from the interrogators. While this is possible, in this deployment example, a management application would access the performance monitoring and diagnostic data directly using the SNMP MIBs referenced in Annex B.

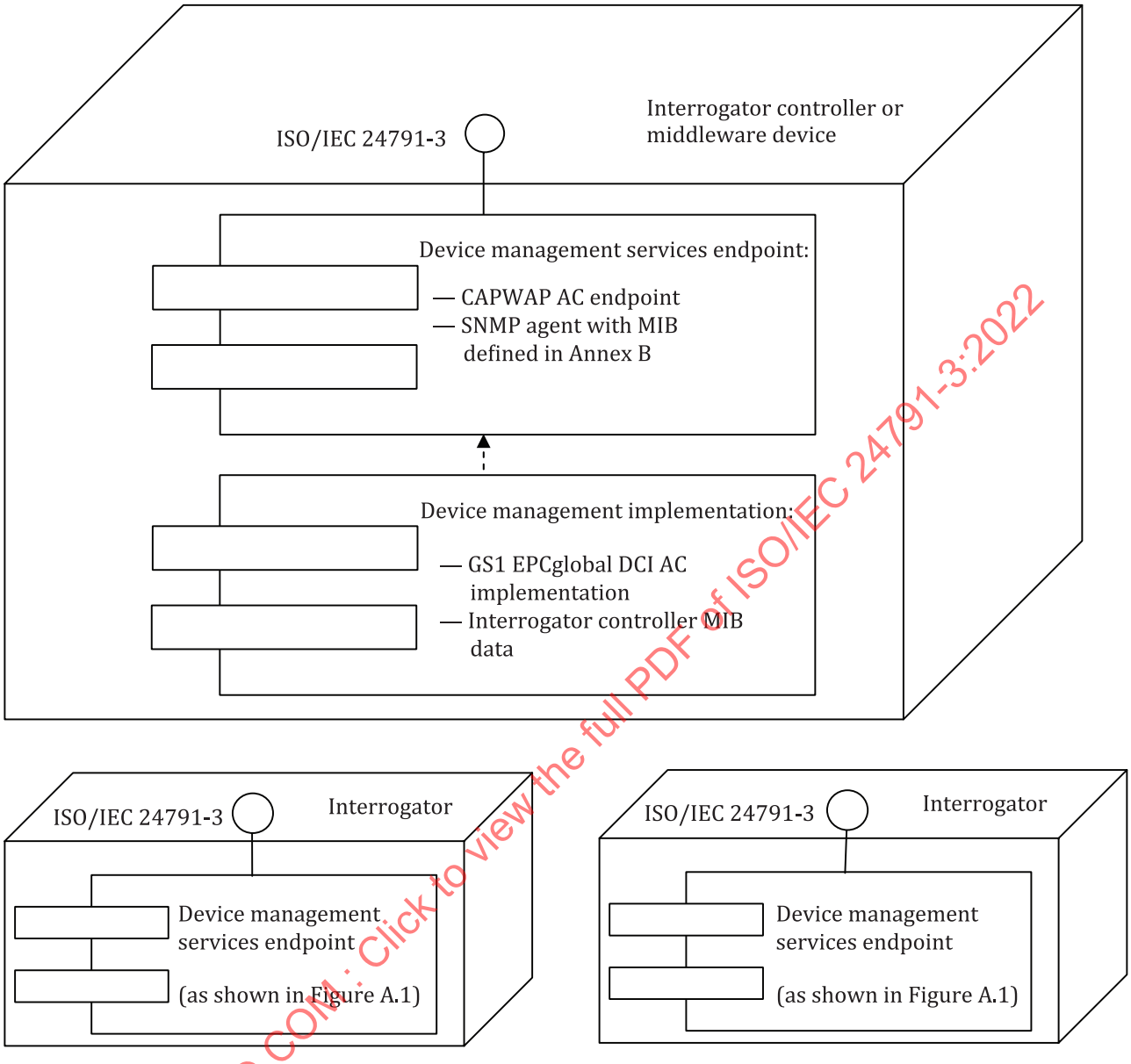

Figure A.2 — Node diagram with ISO/IEC 24791-3 endpoints on interrogator controller and interrogators

Annex B
(normative)

SSI Device Management MIB

```
ISO-24791-3-MIB DEFINITIONS ::= BEGIN
--
-- ISO/IEC 24791-3 MIB for Software System Infrastructure Device Management
--

IMPORTS
    Counter32, IpAddress
        FROM SNMPv2-SMI
    DisplayString, TimeStamp
        FROM SNMPv2-TC
    MODULE-IDENTITY
        FROM SNMPv2-SMI
    iso
      FROM SNMPv2-SMI;

rfidSsiDeviceManagementMIB MODULE-IDENTITY
    LAST-UPDATED "201212122100Z"
    ORGANIZATION "ISO"
    CONTACT-INFO
        "Convener, ISO WG4/SG1"
    DESCRIPTION
        "This module defines a MIB that represents that manageable entities
         described in the RFID Software System Infrastructure International
         Standard ISO/IEC 24791, specific Part 3, Device Management"
    ::= { iso(1) standard(0) iso24791(24791) part3(3) }


rfidSsiDeviceManagementObjects       OBJECT IDENTIFIER   ::= { rfidSsiDeviceManagementMIB
0 }
rfidSsiDeviceManagementConformance   OBJECT IDENTIFIER   ::= { rfidSsiDeviceManagementMIB
1 }


rfidSsiDeviceManagementDescription OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A textual description of this RFID Network."
    ::= { rfidSsiDeviceManagementObjects 1 }

rfidSsiDeviceManagementLocation OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A textual identification of the location of this RFID Network.  This may
        be a GLN or other structured identifier represented in the URI format."
    ::= { rfidSsiDeviceManagementObjects 2 }


--
-- RFID Interrogator Controller (RIC) infrastructure representation MIB in
-- the SSI Device Management MIB tree.  The RFID Interrogator Controller
-- is a function that requests ISO/IEC 24791-5 Device Interface services
-- to one or more interrogators.
--
-- It is expected that more than one RIC will be deployed in an RFID Network to
-- provide for redundancy, load balancing, and scaling.  This is represented
-- by the ricControllerTable that may contain more than one RIC.
```

```
--

ricMIB OBJECT IDENTIFIER ::= { rfidSsiDeviceManagementObjects 10 }

ricNotifications OBJECT IDENTIFIER ::= { ricMIB 0 }
ricObjects       OBJECT IDENTIFIER ::= { ricMIB 1 }


ricControllerTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF RicControllerEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "A table of information about the RICs in the RFID Network."
    ::= { ricObjects 1 }


ricControllerEntry OBJECT-TYPE
    SYNTAX  RicControllerEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "An entry containing information about a particular RIC."
    INDEX   { ricControllerName }
    ::= { ricControllerTable 1 }

RicControllerEntry ::=
    SEQUENCE {
        ricControllerName
            DisplayString,
        ricControllerIpAddr
            IpAddress,
        ricControllerDescription
            DisplayString
    }

ricControllerName OBJECT-TYPE
    SYNTAX  DisplayString
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A administrative name or IP hostname for this RIC. Because this is used as the
    index for this table, it is mandatory, but it is not necessary that the value is
    a proper IP hostname to be resolved.  If it is not, another mechanism must be
    used to provide a valid value for ricControllerIpAddr"
    ::= { ricControllerEntry 1 }

ricControllerIpAddr OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The main IP address of this RIC that will be used for
        communication from outside the RFID Network, possibly resolved from
    ricControllerName. The interrogators in the RFID Network will connect to this address,
        but it is also possible that the readers will connect to another
        address on the RIC.  The IF-MIB on the RIC should be used
        for access to the full set of interfaces."
    ::= { ricControllerEntry 2 }

ricControllerDescription OBJECT-TYPE
    SYNTAX  DisplayString
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A textual description for this RIC."
    ::= { ricControllerEntry 3 }


--
--  Notifications for the ricMIB
```

```
--

ricStartup NOTIFICATION-TYPE
    STATUS    current
    DESCRIPTION
    "A notification for an RFID Network Controller startup event"
    ::= { ricNotifications 1 }


ricShutdown NOTIFICATION-TYPE
    STATUS    current
    DESCRIPTION
    "A notification for an RFID Network Controller shutdown event "
    ::= { ricNotifications 2 }

--
-- RFID RIC Interrogator representation MIB in the SSI Device Management MIB
--
-- This MIB provides the representation of the interrogator from the
-- the viewpoint of an RFID Interrogator Controller.  The MIB for an
-- interrogator within SSI Device Management is provided by the GS1 EPCglobal
-- Reader Management MIB as described in ISO/IEC 24791-3.
--

ricInterrogatorMIB OBJECT IDENTIFIER ::= { rfidSsiDeviceManagementObjects 11 }

ricInterrogatorNotifications OBJECT IDENTIFIER ::= { ricInterrogatorMIB 0 }
ricInterrogatorObjects    OBJECT IDENTIFIER    ::= { ricInterrogatorMIB 1 }


ricInterrogatorTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF RicInterrogatorEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "A table of information about the Interrogators in an RFID Network as
         viewed by an Interrogator Controller."
    ::= { ricInterrogatorObjects 1 }

ricInterrogatorEntry OBJECT-TYPE
    SYNTAX  RicInterrogatorEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "An entry containing information about a particular interrogator."
    INDEX   { ricInterrogatorName }
    ::= { ricInterrogatorTable 1 }

RicInterrogatorEntry ::=
    SEQUENCE {
        ricInterrogatorName
            DisplayString,
        ricInterrogatorIpAddr
            IpAddress,
        ricInterrogatorRicIpAddr
            IpAddress,
        ricInterrogatorDescription
            DisplayString,
        ricInterrogatorConnectionAdminStatus
            INTEGER,
        ricInterrogatorConnectionOperStatus
            INTEGER,
        ricInterrogatorConnectionTime
            TimeStamp,
        ricInterrogatorFirmwareVersion
            DisplayString,
        ricInterrogatorAggregateReadReqs
            Counter32,
        ricInterrogatorAggregateTagReads
            Counter32,
    ricInterrogatorLastTagReadTime
```

```
            Gauge32,
        ricInterrogatorAggregateGpiEvents
            Counter32,
        ricInterrogatorLastGpiEventTime
            Gauge32,
        ricInterrogatorErrorMessage
            DisplayString

    }

ricInterrogatorName OBJECT-TYPE
    SYNTAX  DisplayString
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A administrative name or IP hostname for this interrogator. Because this is used
as the
    index for this table, it is mandatory, but it is not necessary that the value is
    a proper IP hostname to be resolved.  If it is not, another mechanism must be
    used to provide a valid value for ricInterrogatorIpAddr."
    ::= { ricInterrogatorEntry 1 }

ricInterrogatorIpAddr OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The IP address of the interrogator that is used for the network connection
        between the RIC and the interrogator."
    ::= { ricInterrogatorEntry 2 }

ricInterrogatorRicIpAddr OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The IP address of the RIC that is connected to this interrogator and is
        responsible for control of the interrogator in the RFID Network.  This
        RIC IP address may not be the primary RIC IP address in the
        ricControllerEntry if the controlling RIC has more than one interface."
    ::= { ricInterrogatorEntry 3 }

ricInterrogatorDescription OBJECT-TYPE
    SYNTAX  DisplayString
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A textual description for this interrogator."
    ::= { ricInterrogatorEntry 4 }

ricInterrogatorConnectionAdminStatus OBJECT-TYPE
  SYNTAX      INTEGER {
                  unknown(0),
                  enabled(1),
                  disabled(2)
                  }
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The desired administrative status of the connection between
        the RIC and the interrogator."
    ::= { ricInterrogatorEntry 5 }

ricInterrogatorConnectionOperStatus OBJECT-TYPE
    SYNTAX     INTEGER {
                    unknown(0),
                    connected(1),
                    disconnected(2)
                    }
    ACCESS  read-only
    STATUS  mandatory
```