INTERNATIONAL STANDARD

ISO 20242-2

First edition
2010-09-01

Industrial automation systems and integration — Service interface for testing applications —

Part 2: Resource management service interface

Systèmes d'automatisation industrielle et intégration — Interface de service pour contrôler les applications —
Partie 2: Interface de service pour la gestion de ressource

Reference number
ISO 20242-2:2010(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) 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.

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

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member 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 shall not be held responsible for identifying any or all such patent rights.

ISO 20242-2 was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 5, *Architecture, communications and integration frameworks*.

ISO 20242 consists of the following parts, under the general title *Industrial automation systems and integration — Service interface for testing applications*:

— *Part 1: Overview*

— *Part 2: Resource management service interface*

The following parts are planned:

— *Part 3: Virtual device service interface*

— *Part 4: Device capability profile template*

— *Part 5: Application program service interface*

— *Part 6: Conformance test methods, criteria and reports*

Introduction

The motivation for ISO 20242 stems from international automotive industries and their suppliers to facilitate the integration of automation and measurement devices, and other peripheral components for this purpose, into computer-based applications. It defines rules for the construction of device drivers and their behaviour in the context of an automation application, or a measurement application, or an automation and measurement application.

The main goal of ISO 20242 is to provide users with:

— independence from the computer operating system;

— independence from the device connection technology (device interface/network);

— independence from device suppliers;

— the ability to certify device drivers with connected devices and their behaviour in the context of a given computer platform;

— independence from the technological device development in the future.

ISO 20242 will not force the development of new device families or the use of special interface technologies (networks). It encapsulates a device and its communication interface to make it compatible with other devices of that kind for a given application.

Industrial automation systems and integration — Service interface for testing applications —

Part 2:
Resource management service interface

1 Scope

This part of ISO 20242 defines a service interface that provides a generic service access point for managing and operating the resources supported by the operating system of a computer and its peripherals, including special hardware on plug-in boards that are used in computer-assisted testing applications. The resource management service interface is intended to be implemented in a manner that offers the exposed services of a computing platform adapter to be generic and independent of the operating system and its communication interfaces.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 20242-1, *Industrial automation systems and integration — Service interface for testing applications — Part 1: Overview*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 20242-1 and the following apply.

3.1
device driver
software module providing an ISO 20242-specified interface with service functions to call a platform adapter to access physical devices

3.2
platform adapter
software module providing a resource management service interface as defined in this part of ISO 20242, which encapsulates the computer platform, including the operating system, the hardware and its peripherals

4 Symbols and abbreviated terms

CNF, Cnf	Confirm (service primitive)
IND, Ind	Indication (service primitive)
REQ, Req	Request (service primitive)

RMS	Resource Management Services
RMSI	Resource Management Service Interface
RSP, Rsp	Response (service primitive)
SAP	Service Access Point

5 Conventions for service definitions and procedures

5.1 General

This part of ISO 20242 uses the descriptive conventions given in ISO/IEC 10731.

The interface between the user of RMS and the provider of RMS is described by service primitives that convey parameters. Since data transmission aspects are outside the scope of ISO 20242, only the request and confirm primitives are used to describe events occurring at the RMS service provider. Indication and response primitives are used to handle events occurring at the RMS service provider. The service model, service primitives and sequence diagrams are abstract descriptions; they do not represent a specification for implementation.

Annex A contains rules for example implementations.

5.2 Parameters

Service primitives, used to represent service user/provider interactions (see ISO/IEC 10731), convey parameters that indicate information used and exchanged in these interactions.

This part of ISO 20242 uses a tabular format to describe the component parameters of the RMS primitives, as shown in Table 1. The parameters that apply to each group of RMS primitives are set out in tables throughout the remainder of this part of ISO 20242. Each table consists of three columns, where the first column contains the name of the service parameter, the second column contains the input parameters of either the request or indication primitives, and the third column contains the output parameters of either the confirm or response primitives.

One parameter (or part of it) is listed in each row of each table. Under the appropriate service primitive columns, the following codes are used to specify the type of usage of the parameter on the primitive and parameter direction specified in the column:

a) M: parameter is mandatory for the primitive;

b) C: parameter is conditional upon other parameters or upon RMS capabilities;

c) S: parameter is a selected item;

d) (blank): parameter is not conveyed by the RMS user or the RMS provider.

Table 1 — Tabular format for service primitive parameters

Parameter name	REQ or IND	CNF or RSP
Argument	M	
Parameter 1	M	
Parameter 2	C	
Result (+)		S
Parameter 3		M
Parameter 4		C
Result (-)		S
Parameter 5		M

5.3 Service procedures

5.3.1 RMS confirmed services

An RMS user submits a request primitive to the RMSI. It is implied that the service access point (SAP) exists. The corresponding service processing entity delivers a confirmation primitive to the user after all necessary interactions are finished or an error occurred.

5.3.2 RMS event handling

The user creates a service access point (SAP) at RMSI for handling events. An event is signalled with an indication primitive at this access point. The user of RMSI issues a response primitive after all necessary interactions are finished or an error occurred (see Figure 1).

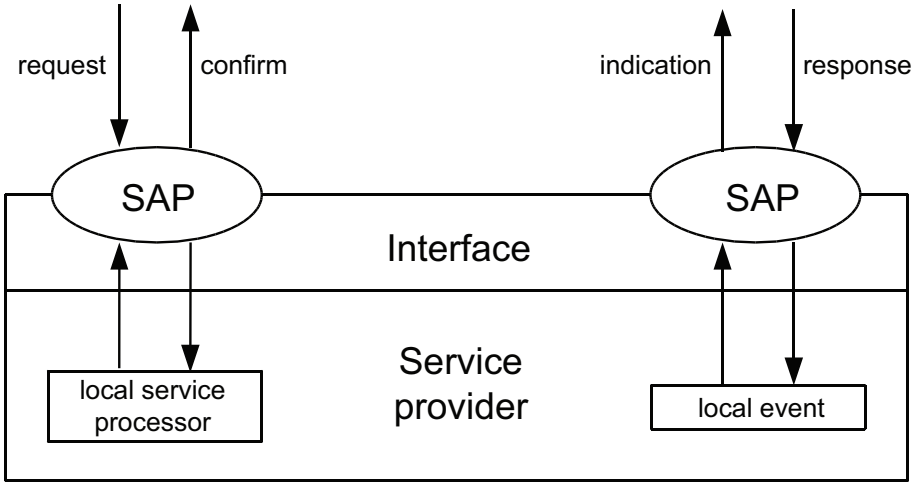

Figure 1 — Handling local events with RMS

5.4 Service primitives and state diagrams

If needed, UML state diagrams are used to describe the behaviour of RMS. In such diagrams, only the service name is used to describe a state transition where no explicit state between request and confirm primitives is necessary [see Figure 2 b)]. Otherwise an extra state of processing the service is denoted [see Figure 2 a)].

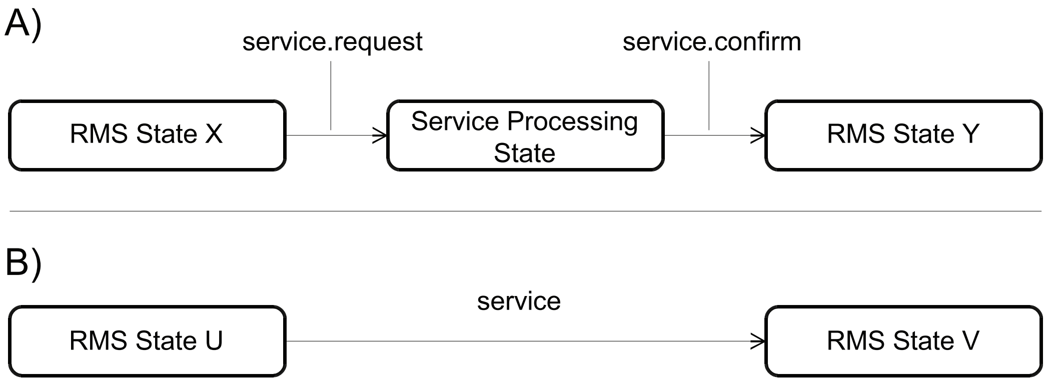

Figure 2 — State transitions caused by services

6 Resource Management Services

6.1 Overview

The RMSI shall provide generic management support services, generic operating support services, and generic input/output services.

The input/output services access another subjacent layer providing extended services. Extended services are introduced to describe the structure of loadable resources for different kinds of periphery interfaces (see Figure 3).

NOTE 1 ISO 20242 does not define the methods for integrating entities with extended services into the RMS provider, as that will depend on the computer operating system and the programming language used for implementing service providers. However, the extended services need to be described to enable the extension of input/output services for different peripheral interfaces without changing the RMS provider. See Annex A for an implementation example.

NOTE 2 There are additional cascading methods described in Annex B for using the RMSI in more complex structures of device and equipment integration.

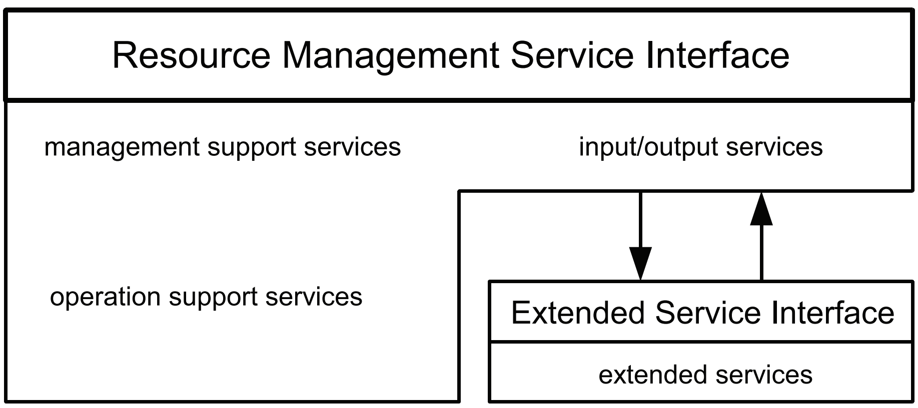

Figure 3 — Service users and providers at the RMSI

6.2 List of services

6.2.1 Generic management support services

Generic management support services are used for handling the access to other services and for initiating (and loading, if necessary) extended service providers. Table 2 gives an overview of these services.

Table 2 — Generic management support services

Service	Name for identification	Remarks
Get Service Reference	getFuncAddress	Get the reference for a service by identifier (name and/or number) and version number.
Initiate Periphery Interface Type	io_initiate	Get the identifier for a specified interface type and load an extended service provider for this type (if necessary).
Conclude Periphery Interface Type	io_conclude	Release a type identifier and close the extended service provider for this type (if existing).

6.2.2 Generic input/output services

Generic input/output services are used for communication with real devices and for configuration and control of peripheral interfaces. Table 3 gives an overview of these services.

Table 3 — Generic input/output services

Service	Name for identification	Remarks
Open Periphery Interface Channel	io_open	Open a peripheral interface for data transmission and configure the interface.
Reconfigure Periphery Interface Channel	io_config	Change the configuration of an interface without closing it, e.g. change transmission parameters.
Read Data	io_read	Fetch received data at a peripheral interface.
Write Data	io_write	Deliver data to a peripheral interface for transmitting.
Execute Operation	io_execute	Execute an operation belonging to a peripheral interface channel. This is comparable to handling both read and write data with one service (data exchange).
Cancel Communication	io_cancel	Cancel a Read Data, Write Data or Execute Operation service and prepare the interface for new requests.
Get Periphery Interface Channel Status	io_stat	Investigate the status of a peripheral interface.
Clear Read Buffer	io_clear	Delete the contents of the input buffer of a peripheral interface.
Close Peripheral Interface Channel	io_close	Close a peripheral interface.
Signal Event	io_event	Indicating a local event and responding to the event source.

Generic input/output services are transferred to corresponding extended services (see Table 4) if an extended service provider is loaded for the specified type of interface.

6.2.3 Extended services

Extended services are not visible to the user of the RMSI; they are defined in this part of ISO 20242 to enable a hierarchical modular structure of RMS implementation by using extended service providers. These extended services are substantially the same as the generic input/output services of RMS.

Table 4 — Extended services for peripheral interfaces

Service	Name for identification	Remarks
Initiate Extended Interface Type	ext_initiate	Set the identifier for a specified peripheral interface type.
Conclude Extended Interface Type	ext_conclude	Release the type identifier of ext_initiate.
Open Extended Interface	ext_open	Open a peripheral interface for data transmission and configure the interface.
Reconfigure Extended Interface	ext_config	Change the configuration of a peripheral interface without closing it, e.g. change transmission parameters.
Read Extended Interface Data	ext_read	Fetch received data at a peripheral interface.
Write Extended Interface Data	ext_write	Deliver data to a peripheral interface for transmitting.
Execute Extended Interface Operation	ext_execute	Execute an operation belonging to a peripheral interface. This is comparable to handling read and write data with one service (data exchange).
Cancel Extended Communication	ext_cancel	Cancel a Read Extended Interface Data, Write Extended Interface Data or Execute Extended Interface Operation service and prepare the interface for new requests.
Get Extended Interface Status	ext_stat	Investigate the status of an interface.
Clear Extended Interface Read Buffer	ext_clear	Delete the contents of the input buffer of a peripheral interface.
Close Extended Interface	ext_close	Close a peripheral interface.
Signal Extended Event	ext_event	Indicating an extended event and awaiting a response.

6.2.4 Operating support services

Operating support services (see Table 5) provide access to memory, timer control, semaphores and other resources of the computer operating system.

Table 5 — Operating support services

Service	Name for identification	Remarks
Allocate Memory	os_allocate	Allocate coherent data space of specified size.
Reallocate Memory	os_reallocate	Change size of allocated data space.
Free Memory	os_free	Release allocated data space.
Get Time	os_time	Investigate the local time.
Get Process Time	os_clock	Investigate the CPU-time for a process.
Wait	os_delay	Temporize a specified amount of time.
Create Timer	os_settimer	Create and start a timer.
Signal Timer Event	os_timerEvent	Indicating that a timer elapsed and awaiting a response.
Remove Timer	os_killtimer	Stop and remove a timer.
Create Light Process Timer	os_setLPtimer	Create and start a light process timer; resolution and accuracy depend on the light process.
Signal Light Process Timer Event	os_LPtimerEvent	Indicating that a light process timer elapsed and awaiting a response.
Remove Light Process Timer	os_killLPtimer	Stop and delete a light process timer.
Identify Light Process	os_getLPnumber	Identify the actual light process.
Create Counted Semaphore	os_createSem	Create a counted semaphore to control multiple concurrent use of resources.
Wait for Counted Semaphore	os_waitSem	Wait for a free access to a protected resource.
Release Counted Semaphore	os_releaseSem	Release the access to a protected resource.
Delete Counted Semaphore	os_deleteSem	Delete a counted semaphore.
Create Private Semaphore	os_createMutex	Create a private semaphore to control access to resources by different light processes with mutual exclusion.
Wait for Private Semaphore	os_waitMutex	Wait for a free access to a protected resource.
Release Private Semaphore	os_releaseMutex	Release the access to a protected resource.
Delete Private Semaphore	os_deleteMutex	Delete a private semaphore.
Open Debug Log	os_openDebug	Open a text log for debug messages.
Write Debug Message	os_writeDebug	Send message to text log.
Close Debug Log	os_closeDebug	Close a text log.

6.3 Management support services

6.3.1 Get Service Reference service

6.3.1.1 Service overview

The Get Service Reference service is used to get a reference for other version-dependent resource management services. This service is requested by the RMS user for each resource management service that is needed for an application.

6.3.1.2 Service parameter structure

The service parameters for the Get Service Reference service are shown in Table 6.

Table 6 — Get Service Reference parameter structure

Parameter name	Req	Cnf
Argument	M	
Service identifier (name)	M	
Proposed version number	M	
Result (+)		S
Service reference		M
Result (-)		S

6.3.1.3 Service parameters

6.3.1.3.1 Argument

The argument contains the parameters of the service request.

6.3.1.3.2 Service identifier

This parameter identifies the service for which the reference is requested.

6.3.1.3.3 Proposed version number

This parameter specifies the version which the RMS user requests for this service.

6.3.1.3.4 Result (+)

This selection type parameter indicates that the service request succeeded.

6.3.1.3.5 Service reference

This parameter contains a reference to identify the service of the proposed version number.

6.3.1.3.6 Result (-)

This selection type parameter indicates that the service request failed.

6.3.1.4 Service procedure

If a service of the specified name and with the specified version number is available, a reference to it is created and submitted to the requester.

6.3.2 Initiate Peripheral Interface Type service

6.3.2.1 Service overview

This service requests the availability of an interface with the specified type name. If a name for an extended service provider is specified with this request, the extended provider will be loaded and the availability of the specified interface is requested at this provider.

6.3.2.2 Service parameter structure

The service parameters for this service are shown in Table 7.

Table 7 — Initiate Peripheral Interface Type parameter structure

Parameter name	Req	Cnf
Argument	M	
Interface type name	M	
Extended services provider name	C	
Result (+)		S
Interface type identifier		M
Result (-)		S
Error		M

6.3.2.3 Service parameters

6.3.2.3.1 Argument

The argument contains the parameters of the service request.

6.3.2.3.2 Interface type name

This parameter contains the name of the interface type.

6.3.2.3.3 Extended services provider name

This conditional parameter, if specified, contains the name of an extended services provider that handles the input/output services for this type of interface.

6.3.2.3.4 Result (+)

This selection type parameter indicates that the service request succeeded.

6.3.2.3.5 Interface type identifier

This parameter contains a number identifying this interface type for other service requests.