INTERNATIONAL STANDARD

ISO 20242-3

First edition
2011-12-15

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

Part 3: Virtual device service interface

Systèmes d'automatisation industrielle et intégration — Interface de service pour contrôler les applications —

Partie 3: Interface de service de dispositif virtuel

Reference number
ISO 20242-3:2011(E)

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

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) 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 20242-3 was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 5, *Interoperability, integration, and architectures for enterprise systems and automation applications*.

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*

— *Part 3: Virtual device service interface*

— *Part 4: Device capability profile template*

The following parts are under preparation:

— *Part 5: Application program service interface*

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

Introduction

The motivation for ISO 20242 stems from the desire of 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. ISO 20242 defines rules for the construction of device drivers and their behaviour in the context of an automation and/or 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 ensure compatibility between device drivers and connected devices, and their behaviour in the context of a given computer platform;

— independence from the technological device development in the future.

ISO 20242 does not necessitate 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 3:
Virtual device service interface

1 Scope

This part of ISO 20242 defines a service interface for the communication with virtual devices comprising capabilities of software modules and physical devices, accessed via resource management services as defined in ISO 20242-2.

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*

ISO 20242-2, *Industrial automation systems and integration — Service interface for testing applications — Part 2: Resource management service interface*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 20242-1, ISO 20242-2 and the following apply.

3.1
communication object
existing object which may be accessed with a communication function to read or write a value

[ISO 20242-1:2005, definition 2.3]

3.2
device capability description
text file containing information about the capabilities of virtual devices in a defined format (i.e. structure, syntax)

[ISO 20242-1:2005, definition 2.5]

3.3
device driver
software module providing an ISO 20242-specified interface with service functions to call a platform adapter to access physical devices

[ISO 20242-2:2010, definition 3.1]

3.4
function object
instance describing one capability of a virtual device

3.5
operation
instance describing one complete procedure

3.6
platform adapter
software module providing a resource management service interface as defined in ISO 20242-2, which encapsulates the computer platform, including the operating system, the hardware and its peripherals

NOTE Adapted from ISO 20242-2:2010, definition 3.2.

3.7
virtual device
representation of one or more physical devices and/or stand-alone software modules that provide an unambiguous view of the resources of a communication interface

4 Abbreviated terms

RMS	Resource Management Services
RMSI	Resource Management Service Interface
SAP	Service Access Point
VD	Virtual Device
VDS	Virtual Device Services
VDSI	Virtual Device Service Interface

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 VDS and the provider of VDS is described by service primitives that convey parameters. Because it is not in the scope of this part of ISO 20242 to deal with data transmission aspects, only the request and confirm primitives apply for confirmed services. To handle events occurring at the service provider of VDS, indication and response primitives are used.

The service model, service primitives and sequence diagrams are descriptions for an interface and its usage. They do not represent a specification for a software implementation by using a specific programming language.

Annex A contains rules for using specific programming languages.

5.2 Parameters

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

This part of ISO 20242 uses a tabular format to describe the component parameters of the VDS primitives. The parameters that apply to each group of VDS primitives are set out in tables throughout the remainder of this part of ISO 20242. Each table consists of three columns, containing the name of the service parameter, and a column each for those primitives and parameter-transfer directions used by the VDS:

— the request primitive's input parameters (Req) or the indication primitive's input parameters (Ind), and

— the confirm primitive's output parameters (Cnf) or the response primitive's output parameters (Rsp).

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

M parameter is mandatory for the primitive.

I parameter is an implementation option, and may or may not be provided depending on the implementation of the VDS provider.

C parameter is conditional upon other parameters or upon the environment of the VDS user.

S parameter is a selected item.

O parameter is optional for the service, its presence depends on the contents of the device capability description in accordance with ISO 20242-4.

(blank) parameter is never present.

5.3 Service procedures

5.3.1 VDS confirmed services

The requesting user submits a request primitive to the VDSI. 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 has occurred.

5.3.2 VDS event handling

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

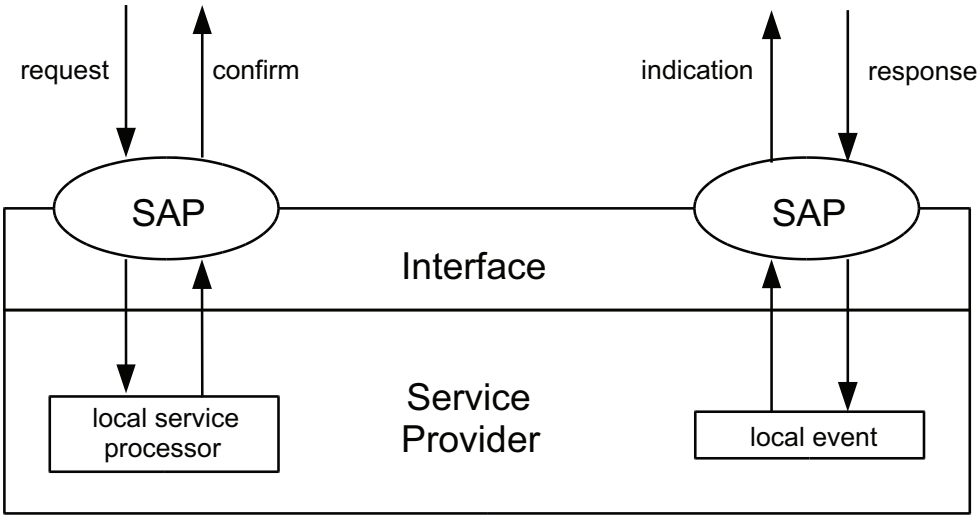

Figure 1 — Handling local events with VDS

6 VDSI model

6.1 Virtual devices and physical devices

ISO 20242 virtual devices are defined from the needs of the VDSI user. The application requires functionalities that may be presented by one or more physical devices and/or software modules (see Figure 2). The functionalities are grouped in accordance with virtual devices that may belong to the following:

— a dedicated physical device,

— a number of physical devices,

— a part of a physical device, or

— a software module inside the device driver or inside a platform adapter.

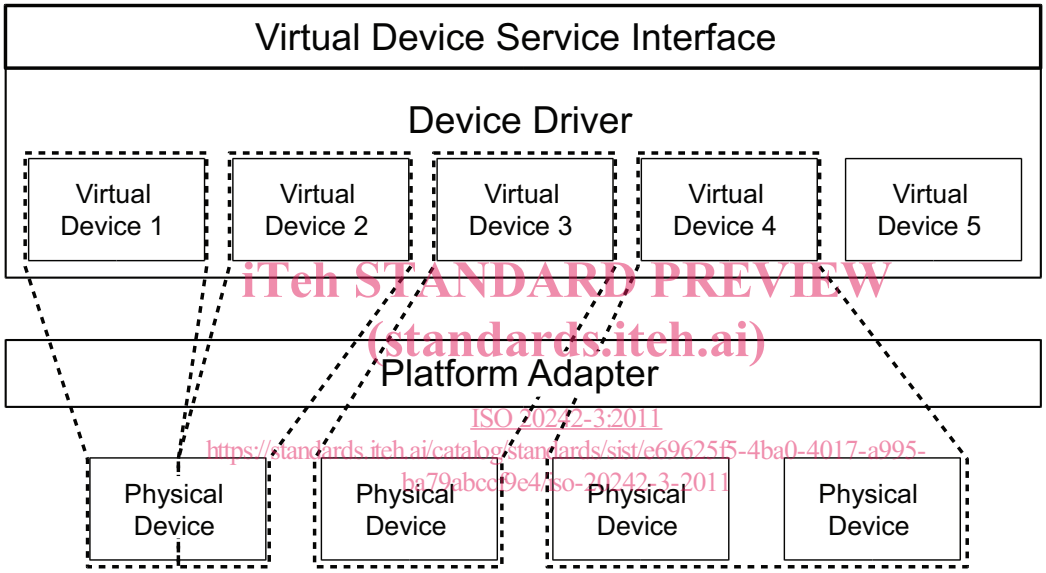

Figure 2 — Mapping of virtual devices to physical devices and software modules

6.2 Structure of VDSI

6.2.1 General

An entity of virtual device services (VDS) contains an access point for services to construct and use virtual devices and provides a virtual device service interface (VDSI). Virtual devices contain function objects with operations and communication objects. Figure 3 shows the structure in a UML class diagram.

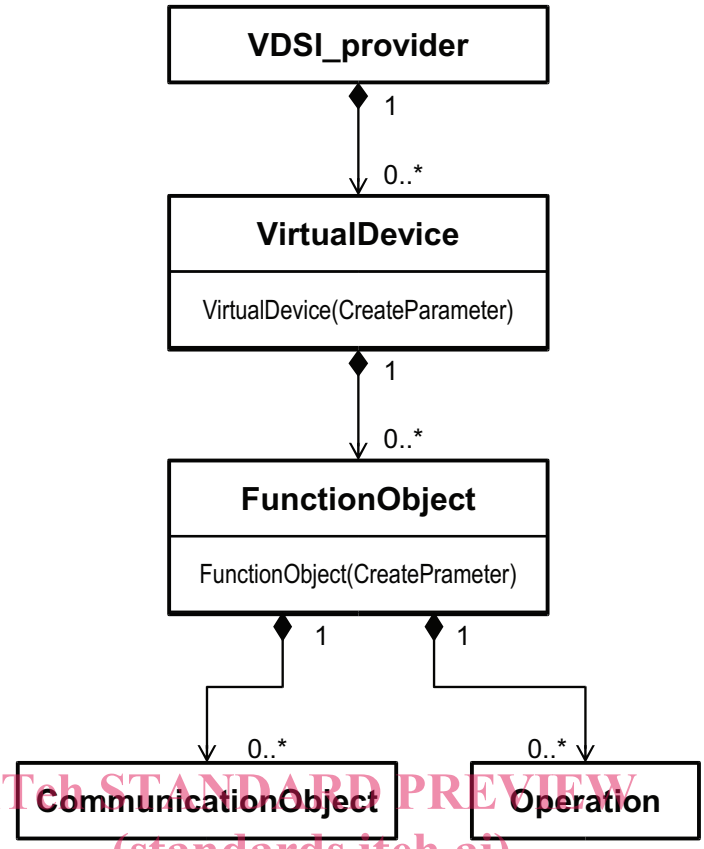

Figure 3 — UML class diagram of VDSI model

6.2.2 Basic management

The services for basic management open a VDSI for use and to control other services. Table 1 gives an overview of these services.

Table 1 — Basic management services

Service	Name for Identification	Remarks
Attach VDSI Entity	VDSI_Attach	Opens an entity, which has a VDSI, to use all its services and creates the service access points for event handling (see 5.3.2).
Cancel Service	VDSI_Cancel	Cancels the execution of a selected service.

6.2.3 Virtual device handling

A virtual device shall be instantiated before it can be used (see 6.2.6). The class of a virtual device is defined in a device capability description. The template for the device capability description is defined in ISO 20242-4. The number of instances of a virtual device depends on the requirements of the application and can be limited by physical or software resources. After instantiation of a virtual device, the associated function objects can be created. Table 2 gives an overview of services to be used with virtual devices.

Table 2 — Virtual device services

Service	Name for Identification	Remarks
Initiate Virtual Device	VDSI_Initiate	Creates an instance of a virtual device and opens it for using other services with it.
Conclude Virtual Device	VDSI_Conclude	Concludes a virtual device and removes the instance, if there are no conditions to keep the instance.
Abort Virtual Device	VDSI_Abort	Aborts a virtual device and removes the instance, if there are no conditions to keep the instance.
Get Virtual Device Status	VDSI_Status	Gets the status of a virtual device.
Identify Virtual Device	VDSI_Identify	Gets the version and an indisputable identification for the vendor of this virtual device.

6.2.4 Function object handling

Function objects are the capabilities of virtual devices needed by an application to fulfil its tasks. Examples of function objects are software entities described as classes or measurement channels with specific signal processing and parameters. A function object shall be instantiated before it is active, or can be used. The class of a function object is defined in the device capability description. The template for the device capability description is defined in ISO 20242-4. The number of instances of a function object depends on the requirements of the application and can be limited by physical or software resources. After instantiation of a function object the associated operations can be used and the associated communication objects can be created. Table 3 gives an overview of services to be used with function objects and their operations.

Table 3 — Function object services

Service	Name for Identification	Remarks
Instantiate Function Object	VDSI_Create-FuncObject	Creates an instance of a function object and opens it for using other services with it.
Remove Function Object	VDSI_Delete-FuncObject	Removes the instance of a function object, if there are no conditions to keep the instance.
Execute Operation	VDSI_Execute	Starts the execution of a procedure associated with the function object.

6.2.5 Communication object handling

Communication objects are the sources and destinations for application data exchange. Examples for communication objects are the parameters for signal processors or measurement results in real devices or the class attributes of software entities described by classes. A communication object shall be instantiated before it can be accessed. The data type of a communication object is defined in the device capability description. The template for the device capability description is defined in ISO 20242-4. Only one instance of a communication object can be created. After instantiation, a communication object can use unsolicited messages to send data to the application or to request data from the application. The management of such unsolicited actions may be done by extra function objects defined in the device capability description. Table 4 gives an overview of services to be used for communication objects and by communication objects.

Table 4 — Communication object services

Service	Name for Identification	Remarks
Instantiate Communication Object	VDSI_Create-CommObject	Creates an instance of a communication object and opens it for using other services with it.
Remove Communication Object	VDSI_Delete-CommObject	Removes the instance of a communication object, if there are no conditions to keep the instance.
Write Data to Communication Object	VDSI_Write	Starts data transmission from application to communication object by request of application.
Read Data from Communication Object	VDSI_Read	Starts data transmission from the communication object to the application by request of the application.
Report Data to Application	VDSI_InfReport	Starts data transmission from the communication object to the application by request of the communication object.
Request Data from Application	VDSI_Accept	Starts data transmission from application to communication object by request of the communication object.

6.2.6 Lifetime of VDSI objects

Figure 4 shows the procedures for creating and deleting VDSI objects. Function objects cannot be created before the associated virtual device exists. Operations are implicitly created with function objects. Communication objects cannot be created before the associated function object exists.

Function objects cannot be deleted before the associated communication objects are deleted. Operations are implicitly deleted with their function object. Virtual devices cannot be deleted before the associated function objects are deleted. An exception to this rule is described in 7.1.3.8, where a special virtual device is used for control procedures.

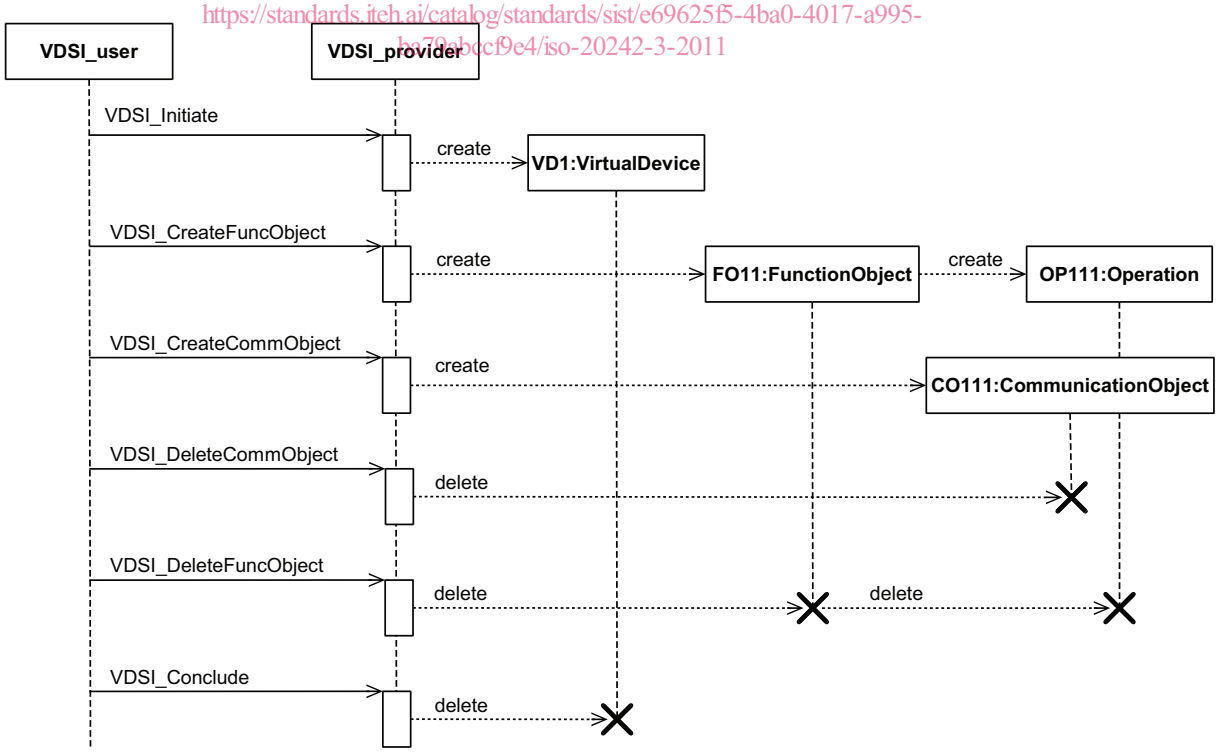

Figure 4 — UML sequence diagram for lifetime of VDSI objects

6.2.7 Executing operations

The execution of an operation is started with the request primitive of service VDSI_Execute. Operations belong to function objects. There may be input parameters for the operation and a result of the execution with output parameters.

6.2.8 Data exchange with communication objects

Data exchange with communication objects may be triggered by VDSI users or by a local event at the VDSI provider. Figure 5 shows the procedures for data exchange.

If the user of VDSI launches service VDSI_Read, then the VDSI provider gets the data out of the addressed communication object (getData in Figure 5) and delivers it to the VDSI user.

If the user of VDSI launches service VDSI_Write, then the VDSI provider puts the data carried by the service into the addressed communication object (getData in Figure 5).

If a local event occurs for a communication object to send, then data is put to the VDSI provider (putData in Figure 5) for delivering it to the VDSI user with service VDSI_InfReport.

If a local event occurs for a communication object to receive, then data is requested from the VDSI provider, who fetches data from the VDSI user with service VDSI_Accept and delivers it to the communication object (getData in Figure 5).

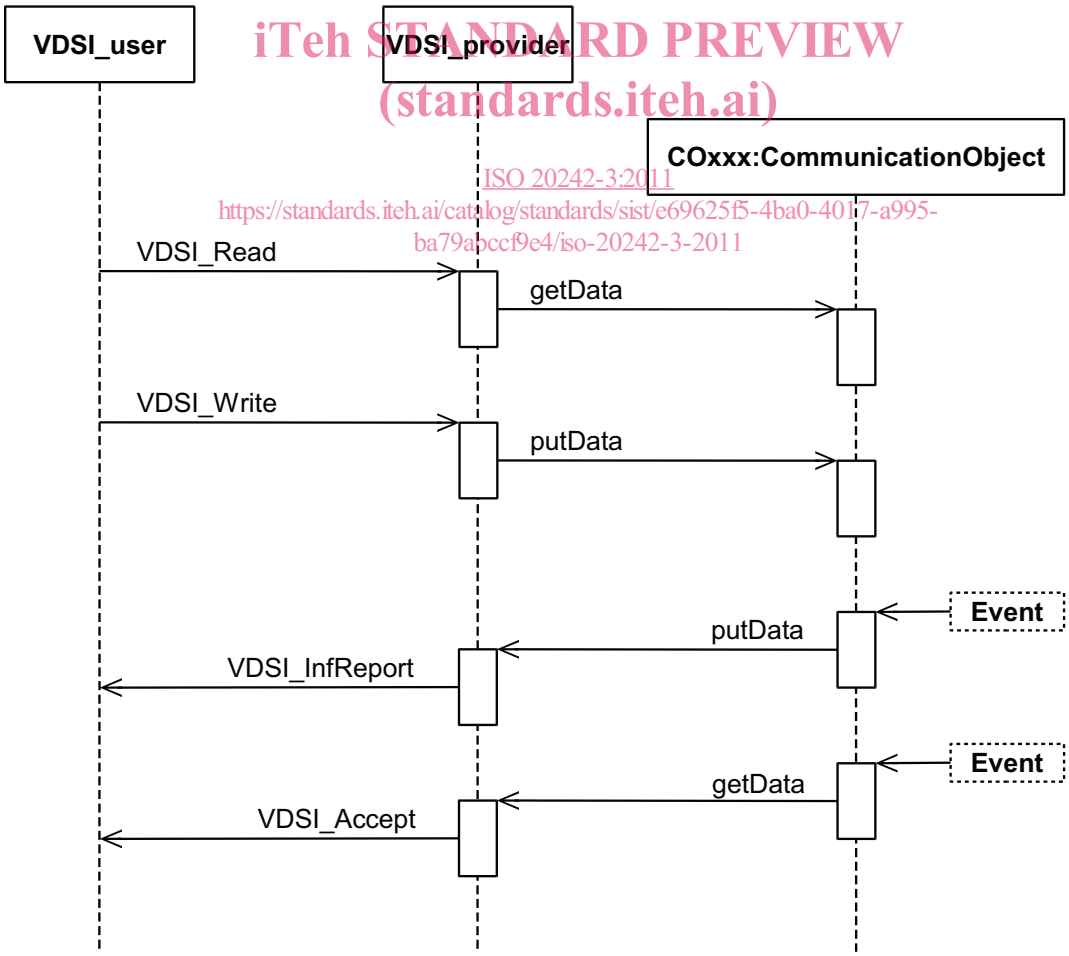

Figure 5 — UML sequence diagram for data exchange with communication objects

6.3 Description of VDSI services

6.3.1 Attach VDSI Entity

6.3.1.1 Service overview

This service is used to open a VDSI entity for using other services and to install the service access points for local events.

6.3.1.2 Service parameters

6.3.1.2.1 General

The service parameters for this service are shown in Table 5.

Table 5 — Attach VDSI Entity

Parameter name	Req	Cnf
Argument	C	
Fetch service access point	C	
Report service access point	C	
Result (+)		S
Result (–)		S
Invocation Error		M

6.3.1.2.2 Argument

6.3.1.2.2.1 General

The argument contains the parameters of the service request.

6.3.1.2.2.2 Fetch service access point

This parameter identifies the service access point which is used by a virtual device to fetch data unsolicited from the user of VDSI as described in 6.3.16.

6.3.1.2.2.3 Report service access point

This parameter identifies the service access point which is used by a virtual device to send data unsolicited to the user of VDSI as described in 6.3.7.

6.3.1.2.3 Result (+)

This selection type parameter indicates that the service request succeeded.

6.3.1.2.4 Result (–)

6.3.1.2.4.1 General

This selection type parameter indicates that the service request failed.