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)

COPYRIGHT PROTECTED DOCUMENT

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).

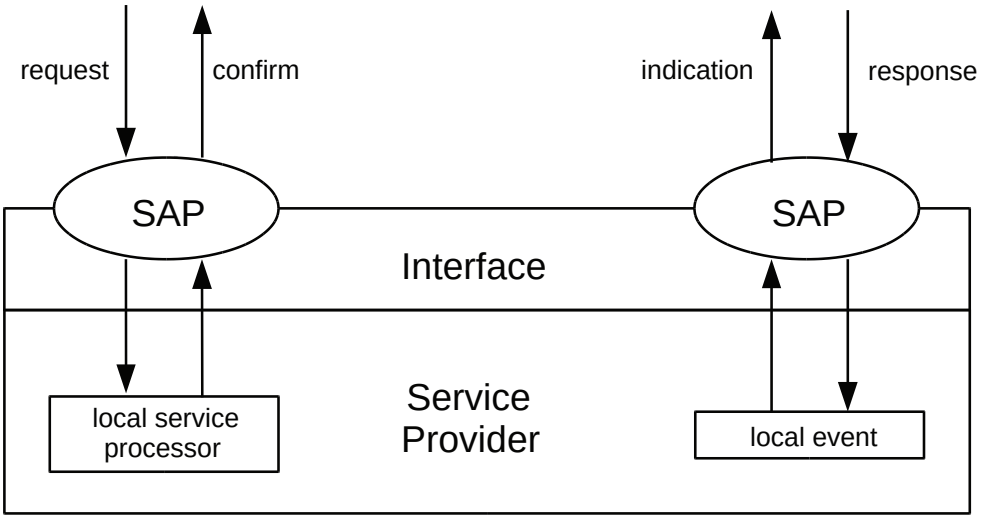

Figure 1 — Handling local events with VDS

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.

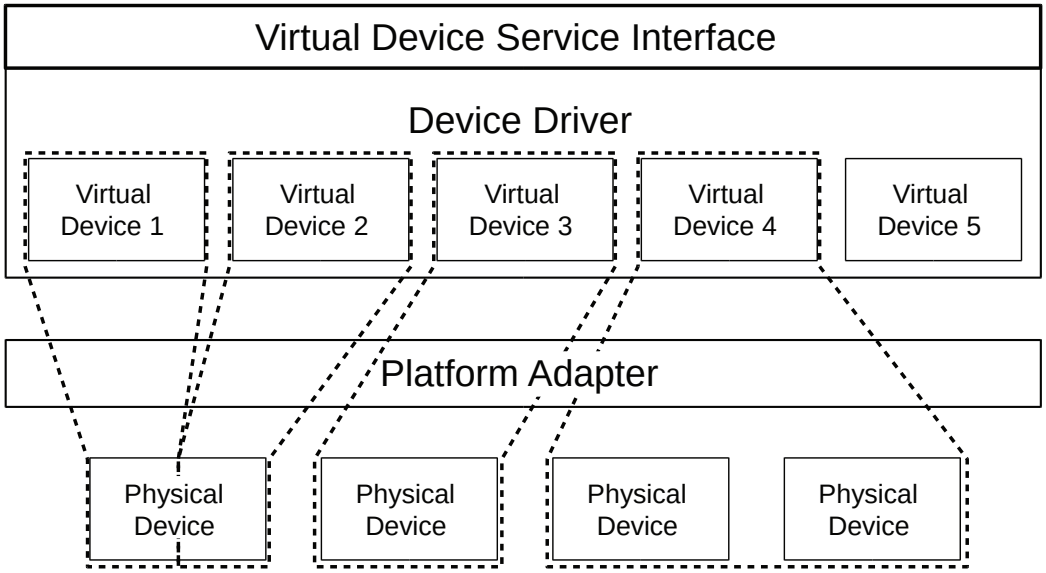

Figure 2 — Mapping of virtual devices to physical devices and software modules

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.

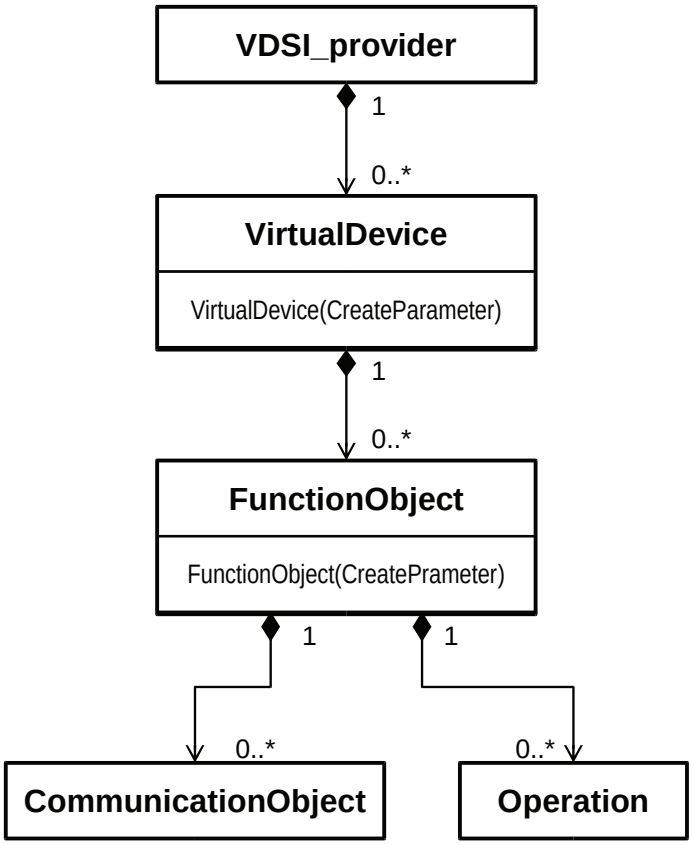

Figure 3 — UML class diagram of VDSI model

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.

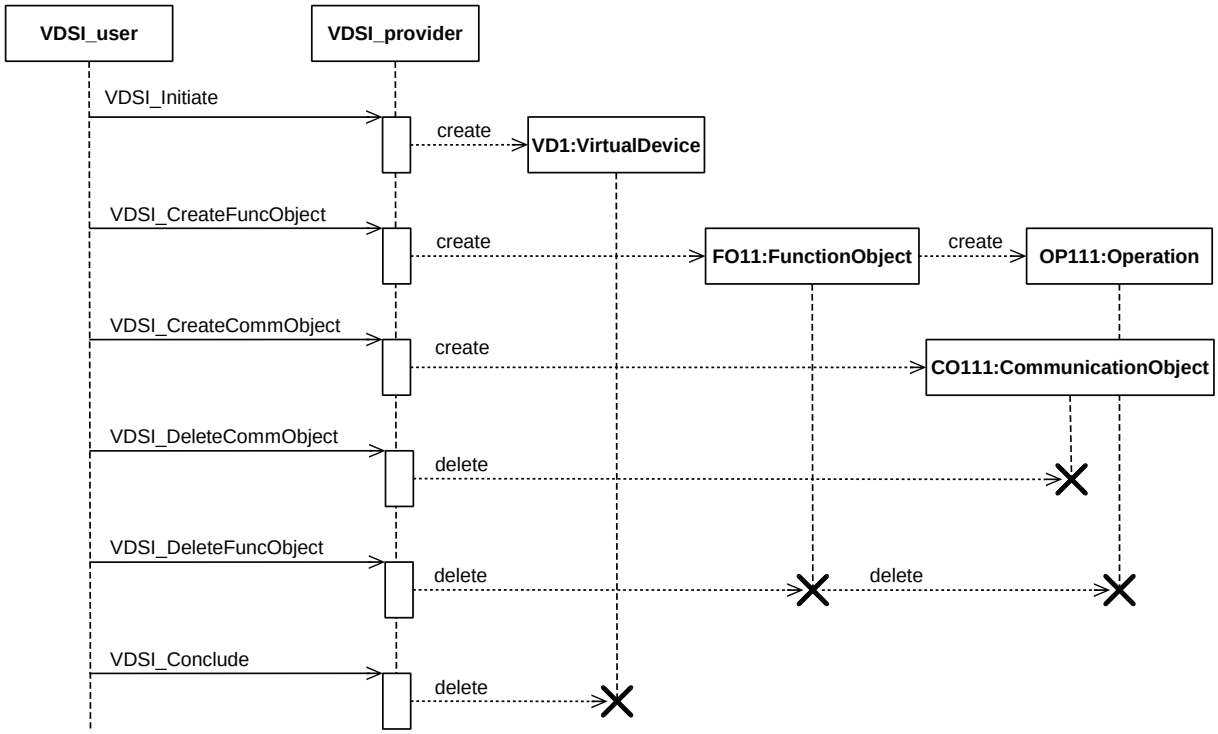

Figure 4 — UML sequence diagram for lifetime of VDSI objects

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).

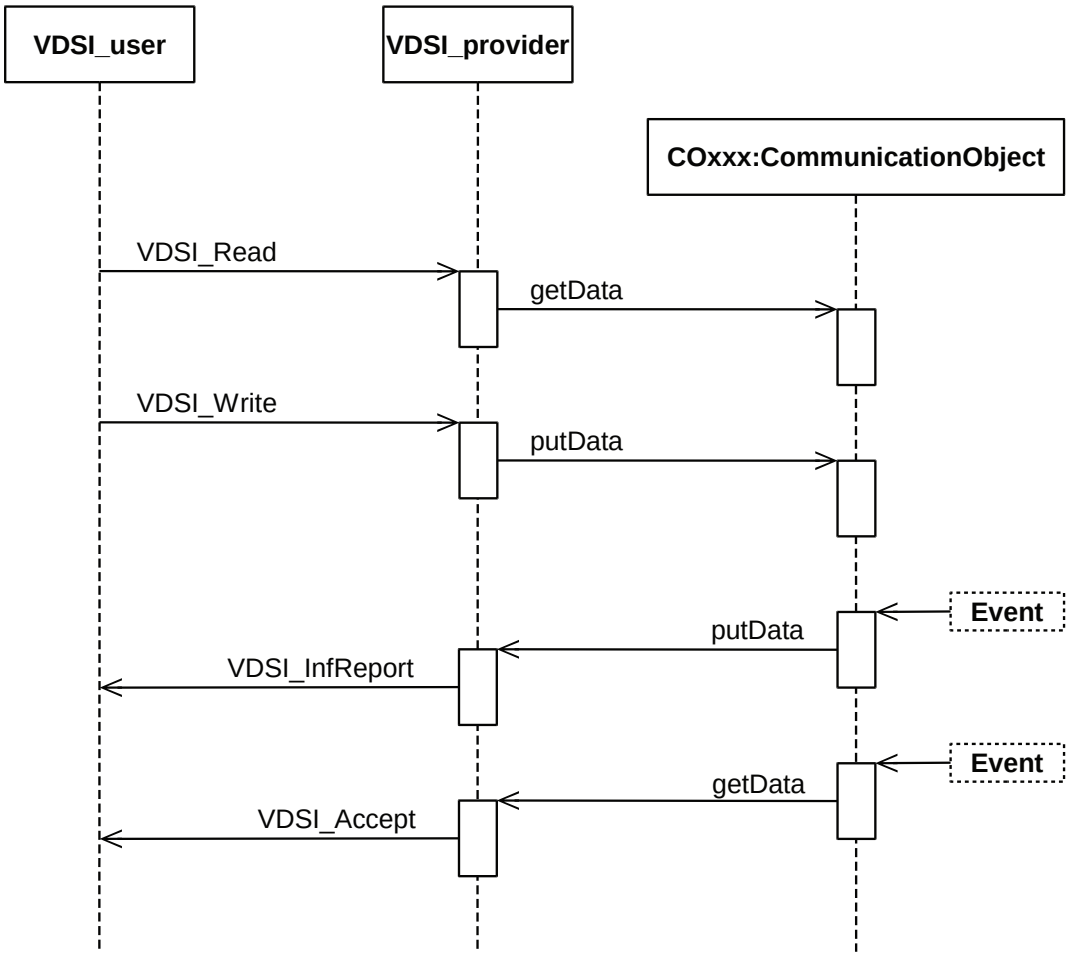

Figure 5 — UML sequence diagram for data exchange with communication objects

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.

6.3.1.2.4.2 Invocation Error

This parameter shall indicate an error among the following choices:

— VDSI entity already opened

— Other

6.3.1.3 Service Procedure

If the VDSI is not already open, it is prepared for executing services, otherwise an error is returned. If service access points for local events are given, VDSI is enabled for unsolicited data transfer via indication and response service primitives.

6.3.2 Cancel Service

6.3.2.1 Service overview

This service cancels any other open service except Attach VDSI Entity, Initiate Virtual Device or another open Cancel Service.

6.3.2.2 Service parameters

6.3.2.2.1 General

The service parameters for this service are shown in Table 6.

Table 6 — Cancel Service

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
User Service Handle	M	
User Handle of cancelled Service	M	
Result (+)		S
User Service Handle		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		M

6.3.2.2.2 Argument

6.3.2.2.2.1 General

The argument contains the parameters of the service request.

6.3.2.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.2.2.2.3 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.2.2.2.4 User Handle of cancelled Service

This parameter is the user defined identifier for the service to be cancelled.

6.3.2.2.3 Result (+)

6.3.2.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.2.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.2.2.3.3 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.2.2.4 Result (–)

6.3.2.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.2.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.2.2.4.3 Result Error

This parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.2.3 Service Procedure

If the VDSI is not already open, it is prepared for executing services, otherwise an error is returned. If service access points for local events are given, VDSI is enabled for unsolicited data transfer via indication and response service primitives.

6.3.3 Initiate Virtual Device

6.3.3.1 Service overview

This service creates an empty virtual device. The contents of the device are defined by further services. The type of the virtual device is defined by a device capability description in accordance with ISO 20242-4.

6.3.3.2 Service parameters

6.3.3.2.1 General

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

Table 7 — Initiate Virtual Device

Parameter name	Req	Cnf
Argument	M	
Virtual Device Type Identifier	M	
User Service Handle	M	
Create Parameter	O	
Result (+)		S
User Service Handle		M
Virtual Device Handle		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.3.2.2 Argument

6.3.3.2.2.1 General

The argument contains the parameters of the service request.

6.3.3.2.2.2 Virtual Device Type Identifier

This parameter identifies the virtual device type defined in the device capability description in accordance with ISO 20242-4.

6.3.3.2.2.3 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.3.2.2.4 Create Parameter

This parameter is optional. Its presence and type depends on relevant definitions in the device capability description for this VDSI entity.

6.3.3.2.3 Result (+)

6.3.3.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.3.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.3.2.3.3 Virtual Device Handle

This parameter is the handle for the instantiated virtual device. It is used with other services for routing into this virtual device.

6.3.3.2.3.4 Result Information

This parameter is an implementation option and defined in detail in 8.1. It may provide additional information about the service execution.

6.3.3.2.4 Result (–)

6.3.3.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.3.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.3.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.3.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— VDSI entity not opened

— Invalid virtual device type identifier

— Other

6.3.3.3 Service Procedure

If the VDSI is open and the virtual device template identifier is valid, the virtual device is created and a handle is defined for further access to this virtual device. Any actions may be done to guarantee the availability of this virtual device. The actions may depend on the mapping of this virtual device to physical devices, as described in 6.1.

There are multiple possible instances for the same virtual device template identifier. The number of instances may be limited by software or hardware resources and by definition in the device capability description.

6.3.4 Conclude Virtual Device

6.3.4.1 Service overview

This service removes a virtual device, which was created by service Initiate Virtual Device of 6.3.3, if there are no local conditions to keep the instance.

6.3.4.2 Service parameters

6.3.4.2.1 General

The service parameters for this service are shown in Table 8.

Table 8 — Conclude Virtual Device

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
User Service Handle	M	
Result (+)		S
User Service Handle		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.4.2.2 Argument

6.3.4.2.2.1 General

The argument contains the parameters of the service request.

6.3.4.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.4.2.2.3 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.4.2.3 Result (+)

6.3.4.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.4.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.4.2.3.3 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.4.2.4 Result (–)

6.3.4.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.4.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.4.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.4.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Other

6.3.4.3 Service Procedure

If the virtual device handle is valid and there is no condition to keep the instance of the virtual device, it is removed with all its contents. An example for a condition to keep the instance is an open communication procedure with physical devices, which cannot be broken at this state.

6.3.5 Abort Virtual Device

6.3.5.1 Service overview

This service removes a virtual device, which was created by service Initiate Virtual Device of 6.3.3, unconditionally.

6.3.5.2 Service parameters

6.3.5.2.1 General

The service parameters for this service are shown in Table 9.

Table 9 — Abort Virtual Device

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
User Service Handle	M	
Result (+)		S
User Service Handle		M
Result (–)		S
User Service Handle		M
Invocation Error		M

6.3.5.2.2 Argument

6.3.5.2.2.1 General

The argument contains the parameters of the service request.

6.3.5.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.5.2.2.3 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.5.2.3 Result (+)

6.3.5.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.5.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.5.2.4 Result (–)

6.3.5.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.5.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.5.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.5.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Other

6.3.5.3 Service Procedure

If the handle is valid, the identified virtual device is removed with all its contents. No conditions apply to keep the instance.

6.3.6 Get Virtual Device Status

6.3.6.1 Service overview

This service indicates the status of a virtual device.

6.3.6.2 Service parameters

6.3.6.2.1 General

The service parameters for this service are shown in Table 10.

Table 10 — Get Virtual Device Status

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
User Service Handle	M	
Result (+)		S
User Service Handle		M
Logical State		M
Physical State		M
Operating State		M
Additional State Info		I
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.6.2.2 Argument

6.3.6.2.2.1 General

The argument contains the parameters of the service request.

6.3.6.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.6.2.2.3 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.6.2.3 Result (+)

6.3.6.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.6.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.6.2.3.3 Logical State

This parameter describes the internal status of the virtual device among the following choices:

— All services may be used for this virtual device without any restriction

— Services that cause a change of data or state in this virtual device are rejected

— Only virtual device services listed in Table 2 will be executed

— Other

6.3.6.2.3.4 Physical State

This parameter describes the physical status of the virtual device among the following choices:

— This virtual device is operational without any restriction

— This virtual device is partly operational, the one or other service may fail

— This virtual device is inoperable

— Maintenance is needed, the virtual device cannot be put to operation by service execution

— Configuration check is active, the structure and the contents of the virtual device are evaluated. The virtual device is in operating state Check (see 6.3.6.2.3.5) and may not be driven to another operating state.

— Other

6.3.6.2.3.5 Operating State

This parameter describes the operating state of the virtual device among the following choices:

— Initialized

— Preparation

— Check

— Working

— Evaluation

— Revise

See Clause 7 for more details on operating states.

6.3.6.2.3.6 Additional State Info

This parameter is an implementation option and may be of type Result Information as described in 8.1 or of type Result Error as described in 8.2. It may provide additional information about the status of the virtual device.

6.3.6.2.3.7 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.6.2.4 Result (–)

6.3.6.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.6.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.6.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.6.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Other

6.3.6.3 Service Procedure

If the virtual device handle is valid, the status of the virtual device is examined and transferred. The status is not defined for the Control VD (see 7.1). If this service is applied to the Control VD, an Invocation Error is returned.

6.3.7 Identify Virtual Device

6.3.7.1 Service overview

This service identifies the virtual device.

6.3.7.2 Service parameters

6.3.7.2.1 General

The service parameters for this service are shown in Table 11.

Table 11 — Identify Virtual Device

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
User Service Handle	M	
Result (+)		S
User Service Handle		M
Virtual Device Version		M
Virtual Device Type Description		M
Version of VDSI		M
Virtual Device Vendor		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.7.2.2 Argument

6.3.7.2.2.1 General

The argument contains the parameters of the service request.

6.3.7.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.7.2.2.3 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.7.2.3 Result (+)

6.3.7.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.7.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.7.2.3.3 Virtual Device Version

This parameter describes the version of the virtual device identified by the virtual device handle. This version may be different for each instance of a virtual device type. This may depend on the mapping of virtual devices to physical devices and information provided by physical devices.

If the Control VD (see 7.1) is addressed, this parameter describes a special version for all VDSI entity virtual devices.

6.3.7.2.3.4 Virtual Device Type Description

This parameter contains a description of the virtual device type and thus corresponds to the virtual device templates in the device capability description in accordance with ISO 20242-4.

If the Control VD (see 7.1) is addressed, this parameter provides a special description for VDSI entity and corresponds to the header information of the device capability description.

6.3.7.2.3.5 Version of VDSI

This parameter provides the version of the VDSI entity and may be checked against the relevant version in the device capability description to ensure that the assumed templates are really provided by this VDSI.

6.3.7.2.3.6 Virtual Device Vendor

This parameter provides information for the individualization of the addressed virtual device. This parameter should only be used if the device capability description is not sufficient to operate the virtual device correctly. Contents of this parameter are a matter for the vendor.

If the Control VD (see 7.1) is addressed, this parameter provides information to identify the vendor of VDSI entity.

6.3.7.2.3.7 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.7.2.4 Result (–)

6.3.7.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.7.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.7.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.7.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Other

6.3.7.3 Service Procedure

If the virtual device handle is valid, the identification information of the virtual device is transferred.

6.3.8 Instantiate Function Object

6.3.8.1 Service overview

This service creates a function object, including its operations. The communication objects belonging to a function object are created with extra services.

6.3.8.2 Service parameters

6.3.8.2.1 General

The service parameters for this service are shown in Table 12.

Table 12 — Instantiate Function Object

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
Function Object Template Identifier	M	
User Service Handle	M	
Create Parameter	O	
Result (+)		S
User Service Handle		M
Function Object Handle		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.8.2.2 Argument

6.3.8.2.2.1 General

The argument contains the parameters of the service request.

6.3.8.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.8.2.2.3 Function Object Template Identifier

This parameter identifies the function object template defined in the device capability description in accordance with ISO 20242-4.

6.3.8.2.2.4 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.8.2.2.5 Create Parameter

This parameter is optional. Its presence and type depends on relevant definitions in the device capability description for this VDSI entity.

6.3.8.2.3 Result (+)

6.3.8.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.8.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.8.2.3.3 Function Object Handle

This parameter is the handle for the instantiated function object. It is used with other services for routing into this function object.

6.3.8.2.3.4 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.8.2.4 Result (–)

6.3.8.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.8.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.8.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.8.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— VDSI entity not opened

— Invalid function object template identifier

— Other

6.3.8.3 Service Procedure

If the virtual device handle and the function object template identifier are valid, the function object is created and a handle is defined for further access to this function object. Any actions may be done to guarantee the availability of this function object.

There are multiple possible instances for the same function object template identifier. The number of instances may be limited by software or hardware resources and by definition in the device capability description.

6.3.9 Remove Function Object

6.3.9.1 Service overview

This service removes a function object, which was created by the service Instantiate Function Object of 6.3.8, if there are no local conditions to keep the instance.

6.3.9.2 Service parameters

6.3.9.2.1 General

The service parameters for this service are shown in Table 13.

Table 13 — Remove Function Object

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
Function Object Handle	M	
User Service Handle	M	
Result (+)		S
User Service Handle		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.9.2.2 Argument

6.3.9.2.2.1 General

The argument contains the parameters of the service request.

6.3.9.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.9.2.2.3 Function Object Handle

This parameter identifies the function object by a handle defined at VDSI with service Instantiate Function Object as described in 6.3.8.

6.3.9.2.2.4 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.9.2.3 Result (+)

6.3.9.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.9.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.9.2.3.3 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.9.2.4 Result (–)

6.3.9.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.9.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.9.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.9.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Invalid function object handle

— Communication object still exists

— Other

6.3.9.3 Service Procedure

If the virtual device handle and the function object handle are valid and there is no condition to keep the instance of the function object, it is removed with all its contents. An example for a condition to keep the instance is an open communication procedure with physical devices, which cannot be broken at this state.

If there are instantiated communication objects associated with this function object, these communication objects shall be removed before the function object may be removed.

6.3.10 Execute Operation

6.3.10.1 Service overview

This service executes a procedure associated with a function object.

6.3.10.2 Service parameters

6.3.10.2.1 General

The service parameters for this service are shown in Table 14.

Table 14 — Execute Operation

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
Function Object Handle	M	
Operation Identifier	M	
User Service Handle	M	
Operation Input Data	O	
Result (+)		S
User Service Handle		M
Operation Output Data		O
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.10.2.2 Argument

6.3.10.2.2.1 General

The argument contains the parameters of the service request.

6.3.10.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.10.2.2.3 Function Object Handle

This parameter identifies the function object by a handle defined at VDSI with service Instantiate Function Object as described in 6.3.8.

6.3.10.2.2.4 Operation Identifier

This parameter identifies the operation of the addressed function object.

6.3.10.2.2.5 Operation Input Data

This parameter is optional and contains the input data for the operation. Its presence and type depends on relevant definitions in the device capability description for this VDSI entity.

6.3.10.2.2.6 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.10.2.3 Result (+)

6.3.10.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.10.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.10.2.3.3 Operation Output Data

This parameter is optional and contains the output data of the operation. Its presence and type depends on relevant definitions in the device capability description for this VDSI entity.

6.3.10.2.3.4 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.10.2.4 Result (–)

6.3.10.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.10.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.10.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.10.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Invalid function object handle

— Invalid operation identifier

— Other

6.3.10.3 Service Procedure

If the virtual device handle, the function object handle and the operation identifier are valid, the addressed operation is executed.

6.3.11 Instantiate Communication Object

6.3.11.1 Service overview

This service creates a communication object within a function object.

6.3.11.2 Service parameters

6.3.11.2.1 General

The service parameters for this service are shown in Table 15.

Table 15 — Instantiate Communication Object

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
Function Object Handle	M	
Communication Object Identifier	M	
User Object Handle	M	
User Service Handle	M	
Result (+)		S
User Service Handle		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.11.2.2 Argument

6.3.11.2.2.1 General

The argument contains the parameters of the service request.

6.3.11.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.11.2.2.3 Function Object Handle

This parameter identifies the function object by a handle defined at VDSI with service Instantiate Function Object as described in 6.3.8.

6.3.11.2.2.4 Communication Object Identifier

This parameter identifies the communication object as defined in the device capability description in accordance with ISO 20242-4.

6.3.11.2.2.5 User Object Handle

This parameter identifies the communication object defined by the user. This parameter will be used when unsolicited data transfer for this communication object is requested at the user. See also services Report Data to Application (6.3.15) and Request Data from Application (6.3.16).

6.3.11.2.2.6 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.11.2.3 Result (+)

6.3.11.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.11.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.11.2.3.3 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.11.2.4 Result (–)

6.3.11.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.11.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.11.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.11.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Invalid function object handle

— Invalid communication object identifier

— Other

6.3.11.3 Service Procedure

If the virtual device handle, the function object handle and the communication object identifier are valid, the communication object is created and the user object handle is attached. When a local event occurs to transfer data between communication object and VDS user, the user object handle will be used to identify the communication object at VDS user.

There is only one instance for one communication object identifier possible.

6.3.12 Remove Communication Object

6.3.12.1 Service overview

This service removes a communication object, which was created by the service Instantiate Communication Object of 6.3.11, if there are no local conditions to keep the instance.

6.3.12.2 Service parameters

6.3.12.2.1 General

The service parameters for this service are shown in Table 16.

Table 16 — Remove Communication Object

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
Function Object Handle	M	
Communication Object Identifier	M	
User Service Handle	M	
Result (+)		S
User Service Handle		M
User Object Handle		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.12.2.2 Argument

6.3.12.2.2.1 General

The argument contains the parameters of the service request.

6.3.12.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.12.2.2.3 Function Object Handle

This parameter identifies the function object by a handle defined at VDSI with service Instantiate Function Object as described in 6.3.8.

6.3.12.2.2.4 Communication Object Identifier

This parameter identifies the communication object as defined in the device capability description in accordance with ISO 20242-4.

6.3.12.2.2.5 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.12.2.3 Result (+)

6.3.12.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.12.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.12.2.3.3 User Object Handle

This parameter is the identifier of the communication object defined by the user and was enclosed with the service Instantiate Communication Object of 6.3.11.

6.3.12.2.3.4 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.12.2.4 Result (–)

6.3.12.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.12.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.12.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.12.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Invalid function object handle

— Other

6.3.12.3 Service Procedure

If the virtual device handle, the function object handle and the communication object identifier are valid and there is no local condition to keep the instance of the communication object, it is removed. An example for a condition to keep the instance is an open communication procedure with physical devices, which cannot be broken at this state.

6.3.13 Write Data to Communication Object

6.3.13.1 Service overview

This service transfers data from VDSI user to a communication object.

6.3.13.2 Service parameters

6.3.13.2.1 General

The service parameters for this service are shown in Table 17.

Table 17 — Write Data to Communication Object

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
Function Object Handle	M	
Communication Object Identifier	M	
User Data for Communication Object	M	
User Service Handle	M	
Result (+)		S
User Service Handle		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.13.2.2 Argument

6.3.13.2.2.1 General

The argument contains the parameters of the service request.

6.3.13.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.13.2.2.3 Function Object Handle

This parameter identifies the function object by a handle defined at VDSI with service Instantiate Function Object as described in 6.3.8.

6.3.13.2.2.4 Communication Object Identifier

This parameter identifies the communication object as defined in the device capability description in accordance with ISO 20242-4.

6.3.13.2.2.5 User Data for Communication Object

This parameter holds the user data which shall be written to the communication object. The data type is defined in the device capability description for this VDSI entity in accordance with ISO 20242-4.

6.3.13.2.2.6 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.13.2.3 Result (+)

6.3.13.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.13.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.13.2.3.3 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.13.2.4 Result (–)

6.3.13.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.13.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.13.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.13.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Invalid function object handle

— Invalid communication object identifier

— Invalid user data

— Other

6.3.13.3 Service Procedure

If the virtual device handle, the function object handle, the communication object identifier and user data are valid, user data is written to the communication object. If the communication object is located inside a physical device, all necessary communication is done to transport user data to the physical device via RMSI.

6.3.14 Read Data from Communication Object

6.3.14.1 Service overview

This service transfers data from a communication object to the VDSI user.

6.3.14.2 Service parameters

6.3.14.2.1 General

The service parameters for this service are shown in Table 18.

Table 18 — Read Data from Communication Object

Parameter name	Req	Cnf
Argument	M	
Virtual Device Handle	M	
Function Object Handle	M	
Communication Object Identifier	M	
User Service Handle	M	
Result (+)		S
User Service Handle		M
Data from Communication Object		M
Result Information		I
Result (–)		S
User Service Handle		M
Result Error		S
Invocation Error		S

6.3.14.2.2 Argument

6.3.14.2.2.1 General

The argument contains the parameters of the service request.

6.3.14.2.2.2 Virtual Device Handle

This parameter identifies the virtual device by a handle defined at VDSI with service Initiate Virtual Device as described in 6.3.3.

6.3.14.2.2.3 Function Object Handle

This parameter identifies the function object by a handle defined at VDSI with service Instantiate Function Object as described in 6.3.8.

6.3.14.2.2.4 Communication Object Identifier

This parameter identifies the communication object as defined in the device capability description in accordance with ISO 20242-4.

6.3.14.2.2.5 User Service Handle

This parameter is a user defined identifier for this service. It shall be unique for each open service.

6.3.14.2.3 Result (+)

6.3.14.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.14.2.3.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.14.2.3.3 Data from Communication Object

This parameter holds the data which is read from the communication object. The data type is defined in the device capability description for this VDSI entity in accordance with ISO 20242-4.

6.3.14.2.3.4 Result Information

This parameter is an implementation option and is defined in detail in 8.1. It may provide additional information about the service execution.

6.3.14.2.4 Result (–)

6.3.14.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.14.2.4.2 User Service Handle

This parameter is a copy of the user defined identifier for this service delivered with the service request.

6.3.14.2.4.3 Result Error

This selection type parameter, which is defined in detail in 8.2, provides the reason for failure.

6.3.14.2.4.4 Invocation Error

This selection type parameter shall indicate an error among the following choices:

— Invalid virtual device handle

— Invalid function object handle

— Invalid communication object identifier

— Other

6.3.14.3 Service Procedure

If the virtual device handle, the function object handle and the communication object identifier are valid, data is read from the communication object. If the communication object is located inside a physical device, all necessary communication is done to fetch data from the physical device via RMSI.

6.3.15 Report Data to Application

6.3.15.1 Service overview

This service is used by a communication object to send data to the user of VDSI.

6.3.15.2 Service parameters

6.3.15.2.1 General

The service parameters for this service are shown in Table 19.

Table 19 — Report Data to Application

Parameter name	Ind	Rsp
Argument	M	
User Object Identifier	M	
Data	M	
Result (+)		S
Result (–)		S
Error		M

6.3.15.2.2 Argument

6.3.15.2.2.1 General

The argument contains the parameters of the service request.

6.3.15.2.2.2 User Object Identifier

This parameter identifies the communication object for the application. This identifier is defined with service Instantiate Communication Object (see 6.3.11).

6.3.15.2.2.3 Data

The data transferred to the user of VDSI. The type of this data is defined in the device capability description of the VDSI entity.

6.3.15.2.3 Result (+)

This selection type parameter indicates that the service request succeeded.

6.3.15.2.4 Result (–)

6.3.15.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.15.2.4.2 Error

This parameter shall indicate an error among the following choices:

— User object identifier is not valid

— Data access temporarily not possible

— Other

6.3.15.3 Service Procedure

If the given object identifier is valid and data access is possible, VDSI user accepts data. Otherwise an error is returned.

6.3.16 Request Data from Application

6.3.16.1 Service overview

This service is used by a communication object to request data from the user of VDSI.

6.3.16.2 Service parameters

6.3.16.2.1 General

The service parameters for this service are shown in Table 20.

Table 20 — Request Data from Application

Parameter name	Ind	Rsp
Argument	M	
User Object Identifier	M	
Result (+)		S
Data		M
Result (–)		S
Error		M

6.3.16.2.2 Argument

6.3.16.2.2.1 General

The argument contains the parameters of the service request.

6.3.16.2.2.2 User Object Identifier

This parameter identifies the communication object for the application. This identifier is defined with service Instantiate Communication Object (see 6.3.11).

6.3.16.2.3 Result (+)

6.3.16.2.3.1 General

This selection type parameter indicates that the service request succeeded.

6.3.16.2.3.2 Data

The data is transferred to the communication object. The type of this data is defined in the device capability description of the VDSI entity.

6.3.16.2.4 Result (–)

6.3.16.2.4.1 General

This selection type parameter indicates that the service request failed.

6.3.16.2.4.2 Error

This parameter shall indicate an error among the following choices:

— User object identifier is not valid

— Data access temporarily not possible

— Other

6.3.16.3 Service Procedure

If the given object identifier is valid and data access is possible, VDSI user responds with data. Otherwise an error is returned.

7 Operating status of a virtual device

7.1 Control VD

7.1.1 Overview

Virtual devices are constructed for a dedicated application. The instantiation of the needed virtual devices and their function objects and communication objects, plus writing initial data to the communication objects, is called configuration. The operating states of virtual devices are oriented to the procedure of configuration. The operating status of a virtual device is not a matter of internal events but of control by the user of VDSI. For this purpose, a special virtual device for local control, called control VD, is defined, which shall be instantiated before any other virtual device. There are mandatory function objects associated with the control VD.

7.1.2 Device Base Function Object

This function object has only one instance in the control VD and contains one operation, which provides the actual version of the VDSI entity as operation output data (see 6.3.10).

7.1.3 Transition Function Object

7.1.3.1 General

This function object has only one instance in the control VD and contains operations to switch the operating status of an addressed virtual device. The handle of that virtual device is the operation input parameter (see 6.3.10). If the transition to a new state is not possible, a corresponding Result Error is provided by service Execute Operation.

7.1.3.2 Operation StartDefinition

With the execution of this operation, the addressed virtual device is switched from status Initialized to status Preparation (see 7.2).

7.1.3.3 Operation EndDefinition

With the execution of this operation, the addressed virtual device is switched from status Preparation to status Check (see 7.2).

7.1.3.4 Operation StartWorking

With the execution of this operation, the addressed virtual device is switched from status Check or Revise to status Working (see 7.2).

7.1.3.5 Operation AddDefinition

With the execution of this operation, the addressed virtual device is switched from status Working to status Revise (see 7.2).

7.1.3.6 Operation EndWorking

With the execution of this operation, the addressed virtual device is switched from status Working or Check to status Evaluation (see 7.2).

7.1.3.7 Operation ChangeDefinition

With the execution of this operation, the addressed virtual device is switched from status Evaluation to status Preparation (see 7.2).

7.1.3.8 Operation ClearAllObjects

With the execution of this operation, the addressed virtual device is switched from status Evaluation to status Initialized (see 7.2). All communication objects and function objects are removed, the virtual device is empty.

7.2 Operating states of virtual devices

7.2.1 Overview

The possible operating states of virtual devices are shown in Figure 6. The transitions between different states are caused by operations of the Transition Function Object (see 7.1.3) or by services shown in Table 21.

Table 21 — Transitions caused by service request

Transition	Associated Service
Initiate	Initiate Virtual Device
Conclude	Conclude Virtual Device
Abort	Abort Virtual Device

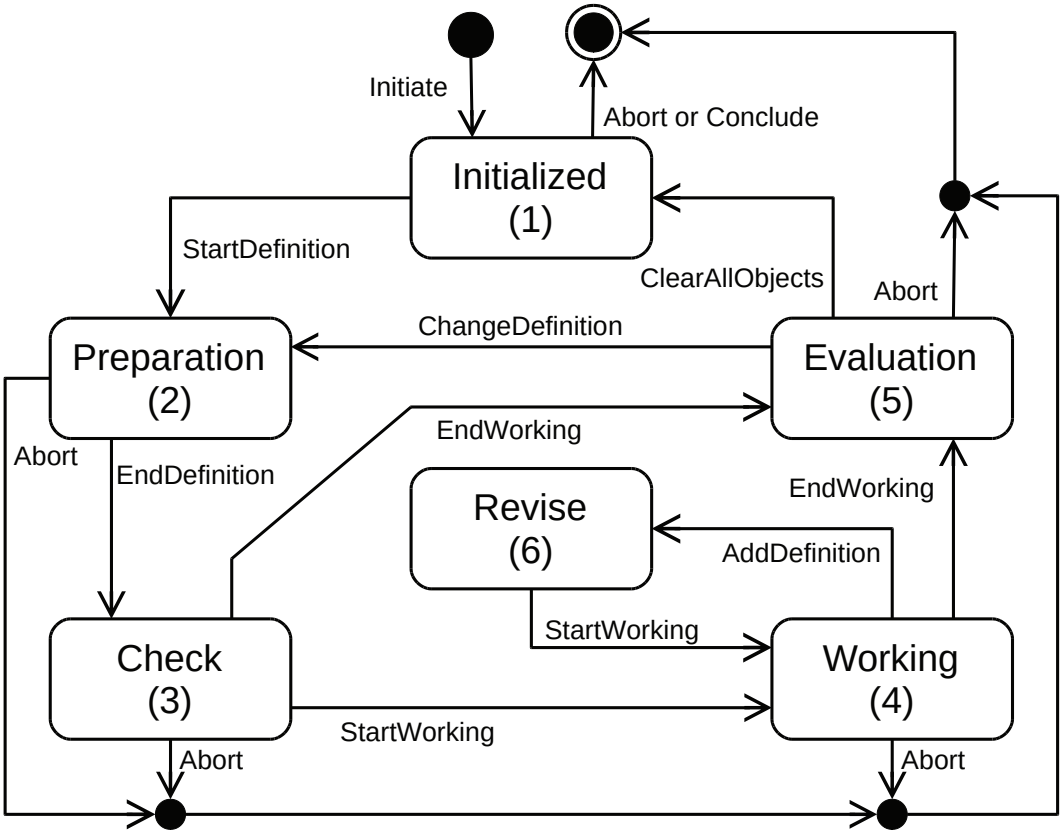

Figure 6 — Operating states of virtual devices

7.2.2 Status Initialized

This status is for a simple test of the virtual device. The VDSI entity may check the availability of the needed fundamental resources, e.g. by initializing the communication to associated physical devices.

The status Initialized is reached by the instantiation of a virtual device or by executing the operation ClearAllObjects in Transition Function Object of Control VD. In status Initialized only services for handling the virtual device itself are possible. See Table 22.

Table 22 — Properties of status Initialized

In-Transitions	Out-Transitions	Possible Services
Initiate ClearAllObjects	Conclude Abort StartDefinition	Conclude Virtual Device Abort Virtual Device Get Virtual Device Status Identify Virtual Device

7.2.3 Status Preparation

This status is for configuration, that is the construction of the virtual device with all it contents to meet a given application. All needed device capabilities are installed and parameterized.

The status Preparation is reached by executing operations StartDefinition or ChangeDefinition in Transition Function Object of Control VD. In status Preparation all services requested by VDSI user dealing with this virtual device and its contents are possible. Local events of VDS are not performed. See Table 23.

Table 23 — Properties of status Preparation

In-Transitions	Out-Transitions	Possible Services
StartDefinition ChangeDefinition	Abort EndDefinition	All services for this virtual device except — Conclude Virtual Device — Report Data to Application — Request Data from Application

7.2.4 Status Check

This status is for a complete test of the virtual device by VDSI entity without being disturbed by VDSI user. The VDSI entity checks the validity of the configuration and the availability of all needed resources, e.g. by testing associated physical devices. The actual state of the check procedure itself may be examined with service Get Virtual Device Status.

The status Check is reached by executing operation EndDefinition in Transition Function Object of Control VD. In status Check only services for handling the virtual device itself are possible. See Table 24.

Table 24 — Properties of status Check

In-Transitions	Out-Transitions	Possible Services
EndDefinition	Abort StartWorking EndWorking	Abort Virtual Device Get Virtual Device Status Identify Virtual Device

7.2.5 Status Working

This status is for running the application, the configuration is finished and all resources are available. It is not possible to change the configuration in this state. Communication objects, marked as parameters in the device capability description (see ISO 20242-4 for more details), may not be written.

The status Working is reached by executing operation StartWorking in Transition Function Object of Control VD. In status Check all services dealing with this virtual device and its contents are possible, except those that change the configuration. Local events of VDS will be performed. See Table 25.

Table 25 — Properties of status Working

In-Transitions	Out-Transitions	Possible Services
StartDefinition ChangeDefinition	Abort EndWorking AddDefinition	All services for this virtual device except — Conclude Virtual Device — Instantiate Function Object — Remove Function Object — Instantiate Communication Object — Remove Communication Object

7.2.6 Status Revise (optional)

This optional status is for optimizing the configuration while running the application, but only minor changes are possible by adding or deleting communication objects and writing parameter values. It is not possible to change the capabilities of the virtual device by adding or deleting function objects.

The status Revise is reached by executing operation AddDefinition in Transition Function Object of Control VD. In status Revise all services dealing with this virtual device and its contents are possible, except those that change the capabilities of the virtual device. Local events of VDS will be performed. See Table 26.

Table 26 — Properties of status Revise

In-Transitions	Out-Transitions	Possible Services
AddDefinition	Abort StartWorking	All services for this virtual device except — Conclude Virtual Device — Instantiate Function Object — Remove Function Object

7.2.7 Status Evaluation

This status is for releasing the configuration used in status Working. The virtual device is prepared for a new configuration. Only those objects will be removed which are not needed for the next application.

The status Evaluation is reached by executing operation EndWorking in Transition Function Object of Control VD. In status Evaluation only services for removing objects and for handling the virtual device itself are possible. See Table 27.

Table 27 — Properties of status Evaluation

In-Transitions	Out-Transitions	Possible Services
EndWorking	Abort ChangeDefinition ClearAllObjects	Abort Virtual Device Get Virtual Device Status Identify Virtual Device Remove Function Object Remove Communication Object

8 Service results

8.1 Additional information

8.1.1 Structure Result Information

If a service is executed successfully, a confirmation of type Result(+) is presented which contains result data in many cases. There may be additional information provided with the confirmation in a structure Result Information as shown in Table 28.

Table 28 — Structure of Result Information

Result Information	Cnf
Info Group	M
Info Grade	M
Info Code	M
Info Description	M

Info Group, Info Grade and Info Code are integer numbers. Info Description is a textual description.

8.1.2 Result Information

The different types of information provided with a successful service are shown in Table 29.

Table 29 — Result Information

Info Group	Info Grade	Info Code	Info Description	Annotation
0	0	0	empty	No extra information provided
0	1	local defined	local defined	Warnings
0	2	local defined	local defined	VDSI Entity Information
not 0	not 0	local defined	local defined	implementation specific information, see Annex A for an example

8.2 Service errors

8.2.1 Invocation Errors

Invocation errors refer to the internal status of a requested service of VDSI entity without any external service involved via RMSI. Invocation errors are described with the Invocation Error parameter at Result(–) within the service descriptions of 6.3.

8.2.2 Structure Result Error

If a service is not executed successfully, including all external services involved via RMSI, a result error is returned. The result error is of type structure with elements described in Table 30.

Table 30 — Structure of Result Error

Result Error	Cnf
Error Group	M
Error Grade	M
Error Code	M
Error Description	M

Error Group, Error Grade and Error Code are integer numbers. Error Description is a textual description.

Error Group shall indicate an error classification among the following choices:

— Periphery Error Group

— Execution Error Group

— Access Error Group

— Application Error Group

— GDI/DIP error Group

— MICX error Group

— Other

8.2.3 Periphery Error Group

The periphery error group contains errors dealing with the communication to physical devices via RMSI. The error is specified in Error Grade as described in Table 31. Error Code is always 0 and Error Description may be used for more details describing the error.

Table 31 — Periphery Errors

Error Grade	Annotation
Per_1	The connection to the physical device is broken, data transmission is no longer possible
Per_2	Confirmation of RMSI not usable
Per_3	Unknown data from RMSI service
Per_4	Invalid User Handle with RMSI confirmation
Per_5	Required periphery interface could not be opened
Per_6	Sending data via RMSI with service Write Data rejected
Per_7	Receiving data via RMSI with service Read Data rejected
Per_8	Service Execute Operation of RMSI failed
Per_9	Any other error

8.2.4 Execution Error Group

8.2.4.1 Overview

The execution error group contains errors dealing with VDSI service execution. Error Grade contains error classes as described in Table 31. Error Code identifies the error itself. Error Description may be used for more details describing the error.

Table 32 — Execution Errors

Error Grade	Annotation
VDstate	Errors concerning the status of a virtual device
AppRef	Errors concerning the application
Definition	Errors concerning definition of objects
Resource	Errors concerning configuration and resources
Preemptive	Errors concerning deadlocks and time control
Access	Errors concerning data access and state transitions
Remove	Errors concerning the release of objects
Cancel	Errors concerning service cancellation

8.2.4.2 VDstate Errors

Error Code shall indicate an error among the following choices:

— Service execution not possible in this operating state

— Other

8.2.4.3 AppRef Errors

Error Code shall indicate an error among the following choices:

— Semaphore error, program controlled resources stay locked

— Other

8.2.4.4 Definition Errors

Error Code shall indicate an error among the following choices:

— Virtual device template identifier not valid

— Function object template identifier not valid

— Communication object identifier not valid

— Data not valid

— Communication object identifier in use

— Communication object instantiation rejected because of configuration mismatch

— Other

8.2.4.5 Resource Errors

Error Code shall indicate an error among the following choices:

— Memory allocation problem

— Processing time problem

— Number of possible instances exhausted

— Configuration error, state Working not possible

— Configuration check in progress, state Working not possible now

— Function object of Control VD not removable because another VD exists

— Other

8.2.4.6 Preemptive Errors

Error Code shall indicate an error among the following choices:

— Execution time for this service exhausted

— Deadlock detected with execution of this service

— Other

8.2.4.7 Access Errors

Error Code shall indicate an error among the following choices:

— Invalid virtual device handle

— Invalid function object handle

— Accessed communication object does not exist in this function object

— Accessed operation does not exist in this function object

— Write Access not allowed because of operating state or read only object

— Data range violation or pointer mismatch access

— Operating state transition not possible

— Hardware error of associated device

— Other

8.2.4.8 Remove Errors

Error Code shall indicate an error among the following choices:

— Object remove not possible because other service is open at this object

— Remove of Control VD not possible because another VD exists

— Other

8.2.4.9 Cancel Errors

Error Code shall indicate an error among the following choices:

— Unknown user service handle

— Service may not be cancelled at this time

— Other

8.2.5 Application Error Group

Application errors cover VDSI implementation specific errors and are associated to multilingual text elements as defined in ISO 20242-4. The XML-element SubAreaError described in ISO 20242-4 is used for this.

Error Group contains the number of the text area specified in multilingual text elements in accordance with ISO 20242-4. Error Code contains the number of the text in that area. Error Description contains a list of text elements which may be inserted at defined placeholders in the associated text element.

Annex A
(informative)

Implementation guidelines for VDSI

A.1 Compatibility

Using this implementation guide should guarantee that implementations of VDSI for the same operating system are compatible in respect to their interfaces and service procedures. Furthermore they will be compatible with implementations of so-called device drivers of ASAM GDI Version 4.5. The names of the functions described in this annex are the same as used in ASAM GDI specification. For the purposes of this part of ISO 20242, it is not necessary to access ASAM GDI specification.

A.2 Using these mapping rules

These mapping rules are not sufficient for creating a valid implementation of VDSI. The service descriptions of 6.3 shall also be considered.

A.3 C and C++ standards

The C programming language is standardized in ISO/IEC 9899.

NOTE The C programming language has been standardized in ISO/IEC 9899:1990 (C90) and in ISO/IEC 9899:1999 (C99).

The C++ programming language is standardized in ISO/IEC 14882.

A.4 Conventions for simple data types

If not otherwise specified, the C/C++ simple data types used in this annex reflect to the 32 bit environment and thus are in most cases operating system independent. If other environments are used (e.g. 64 bit), some simple data types may be operating system dependent. For these cases, the implementation shall be documented with respect to relevant specifications of ASAM GDI.

Strings are fields of 8 bit values (octets) without the value 0. The value 0 marks the end of a string.

A.5 Special simple data types

A.5.1 Operating system independent types

For the service primitive parameters, the operating system independent definitions of special simple data types specified in Table A.1 shall be used.

Table A.1 — OS independent Parameter Data Types

Data Type	C/C++ definition	Remarks
APICHAR	signed char	range –128 to 127
APIBYTE	unsigned char	range 0 to 2exp8
APIRET	signed short	range –65536 to 65535
APIHND	unsigned long	range 0 to 2exp32

A.5.2 Operating system dependent types

For the service primitive parameters, the operating system dependent definitions of special simple data types specified in Table A.2 shall be used.

Table A.2 — OS dependent function calls

Data Type	WIN32 C definition	LINUX C definition
GDI_CALL	__stdcall	no special function type
GDI_CB	__cdecl	no special function type

A.6 Special complex data types

A.6.1 Compiling conventions

Complex data types should be compiled with an alignment of 8 bytes (default alignment). If other alignments are used, this shall be marked with the delivery of the driver.

A.6.2 C data structures

For some service primitive parameters, the operating system independent data structures specified in Table A.3 shall be used.

Table A.3 — OS independent Data Structures

Data Type	C/C++ definition	Remarks

GDIRESULT	struct
	{
	short qual;
	short grade;
	short code;
	void *addInfo;
	};
		This data type describes the result of a function call, in case of errors, warnings or any other information.
		See Tables 28 and 30 for element description

GDIIDENT	struct
	{
	unsigned long deviceVersion;
	char *driverName;
	unsigned long driverVersion;
	char *vendor;
	};
		deviceVersion:
		this number handles the versioning for virtual devices and, if applicable, that of associated physical devices.
		driverName:
		an identification for the device driver or part of the device driver to which this VD belongs
		driverVersion:
		this number handles the versioning for the driver at all and may be checked against a relevant version number in the device capability description
		vendor:
		identification for the manufacturer (developer) of this virtual device
		NOTE Used with the Control VD, these elements address the device driver at all, or information that is valid for all types of virtual devices in a device driver.

GDISTATUS	struct
	{
	short log;
	short phys;
	short phase;
	GDIRESULT detail;
	};
		This data type describes the status of a virtual device. It is not applicable for the Control VD.
		See the description of service primitive parameters in 6.3.6, Get Virtual Device Status, for details.

A.7 Conventions for predefined constants

If constants are used by name, they shall be defined in header files. The names of header files are specified for ASAM GDI in the relevant specifications. Because the contents of the header files may be extended by future implementations and the combination of device drivers of different versions and platform adapters of different versions is also handled inside header files, they are not described in this part of ISO 20242.

Predefined constants used in this annex are described in Table A.4.

Table A.4 — Predefined Constants

Name	Value	Description
SYNC	(APIHND) 0	Identifier for synchronous function calling.

NOTE SYNC is used as an alternative to a user defined handle for asynchronous function calling. Asynchronous communication is requested by a value not equal SYNC. The synonym ASYNC is used for this handle in the function descriptions below. In case of asynchronous communication, this handle is a parameter of the confirmation via callback.

A.8 Conventions for function prototypes

Function prototypes will be described as

returnType callType functionName (list of argument types).

In the following descriptions the returned value of type returnType is called return. Arguments are called arg and numbered beginning with 1 as arg1, arg2, etc.

A.9 Return values

The numbers specified in Table A.5 are used for return values of functions.

Table A.5 — Return value numbers

Number	Name	Description
< –1	Invocation Error	A function could not be executed correctly, the return value describes the error in accordance with the list of invocation errors in. The elements of the structure GDIRESULT are set to zero.
–1	COM_ERR	A function could not be executed correctly, the error is described in structure VDSIRESULT.
0	COM_FIN	A function was completed successfully in synchronous operation mode.
1	COM_BUSY	A function has been started in asynchronous operation mode.

A.10 Invocation error numbers

The negative numbers specified in Table A.6 are used to indicate errors in the return value of a function.

Table A.6 — Error numbers

Error number	Description
–2	VDSI_Attach was requested although VDSI is already open
–3	A service request occurred before VDSI was opened by VDSI_Attach
–4 … –8	Reserved for future use
–9	No resources to execute further asynchronous communication
–10, –11	Reserved for future use
–12	Asynchronous function call not supported
–13	VDSI does not support creating instances of the addressed class
–14	Reserved for future use
–15	Function could not be executed because of a sequence violation or because of invalid parameters

A.11 Basic management services

The functions for basic management services are described in Table A.7.

Table A.7 — Basic management services

Service	Function prototype	Relation to service parameters	
Attach VDSI Entity	short GDI_CALL GDI_Attach (GDI_CB, GDI_CB, GDI_CB)	arg1:	Pointer of the function that is called by VDSI provider for confirmation
		arg2:	Pointer of the function that is called by VDSI provider for indication of service VDSI_InfReport
		arg3:	Pointer of the function that is called by VDSI provider for indication of service VDSI_Accept
		return:	COM_FIN, if VDSI may be used, otherwise an error (see Table A.6)
Cancel Service	short GDI_CALL GDI_Cancel (APIHND, APIHND, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device
		arg2:	SYNC or ASYNC, see note to Clause A.7
		arg3:	VDSI user defined handle of the service that shall be cancelled
		arg4:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5

A.12 Virtual device handling services

The functions for virtual device handling services are described in Table A.8.

Table A.8 — Virtual device services

Service	Function prototype	Relation to service parameters	
Initiate Virtual Device	short GDI_CALL GDI_Initiate (APIHND, APIHND *, void *, APIHND, GDIRESULT *)	arg1:	Identification number for the addressed type of virtual device. This number is defined in the device capability description.
		arg2:	Pointer to a storage for a VDSI provider defined handle of this instance of virtual device.
		arg3:	Pointer to a structure containing the create parameters for the virtual device as described in the device capability description.
		arg4:	SYNC or ASYNC, see note to Clause A.7
		arg5:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5
Conclude Virtual Device	short GDI_CALL GDI_Conclude (APIHND, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	SYNC or ASYNC, see note to Clause A.7
		arg3:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5
Abort Virtual Device	short GDI_CALL GDI_Abort (APIHND)	arg1:	Handle of the addressed VD, defined by VDSI provider at service VDSI_Initiate.
		return:	COM_FIN, if VD is destroyed, otherwise an error (see Table A.6)
Get Virtual Device Status	short GDI_CALL GDI_Status (APIHND, GDISTATUS *, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	Pointer to structure GDISTATUS for information about the status of the VD.
		arg3:	SYNC or ASYNC, see note to Clause A.7
		arg4:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5
Identify Virtual Device	short GDI_CALL GDI_Identify (APIHND, GDIIDENT *, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	Pointer to structure GDIIDENT for identification information about the VD.
		arg3:	SYNC or ASYNC, see note to Clause A.7
		arg4:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5

A.13 Function object handling services

The functions for function object handling services are described in Table A.9.

Table A.9 — Function object services

Service	Function prototype	Relation to service parameters	
Instantiate Function Object	short GDI_CALL GDI_CreateFuncObject (APIHND, APIHND, void *, APIHND *, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	Identification number for the addressed type of function object. This number is defined in the device capability description.
		arg3:	Pointer to a structure containing the create parameters for the function object as described in the device capability description.
		arg4:	Pointer to a storage for a VDSI provider defined handle of this instance of function object.
		arg5:	SYNC or ASYNC, see note to Clause A.7
		arg6:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5
Remove Function Object	short GDI_CALL GDI_DeleteFuncObject (APIHND, APIHND, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	Handle of the addressed FO, defined by VDSI provider at service Instantiate Function Object.
		arg3:	SYNC or ASYNC, see note to Clause A.7
		arg4:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5
Execute Operation	short GDI_CALL GDI_Execute (APIHND, APIHND, APIHND, void *, void *, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	Handle of the addressed FO, defined by VDSI provider at service Instantiate Function Object.
		arg3:	Identification number for the addressed operation. This number is defined in the device capability description.
		arg4:	Pointer to a structure containing the input parameters for the operation as described in the device capability description. This shall be NULL, if there are no input parameters.
		arg5:	Pointer to a structure containing the output parameters of the operation as described in the device capability description. This shall be NULL, if there are no output parameters.
		arg6:	SYNC or ASYNC, see note to Clause A.7
		arg7:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5

A.14 Communication object handling services

A.14.1 Functions related to service descriptions in 6.3.

The functions for communication object handling services are described in Table A.10.

Table A.10 — Communication object services

Service	Function prototype	Relation to service parameters	
Instantiate Communication Object	short GDI_CALL GDI_CreateCommObject (APIHND, APIHND, APIHND, APIHND, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	Handle of the addressed FO, defined by VDSI provider at service Instantiate Function Object.
		arg3:	Identification number for the addressed communication object. This number is equal to the position of this CO in the list of COs inside an FO, starting with 1.
		arg4:	User defined global object identifier for unsolicited services started for this object.
		arg5:	SYNC or ASYNC, see note to Clause A.7
		arg6:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5
Remove Communication Object	short GDI_CALL GDI_DeleteCommObject (APIHND, APIHND, APIHND, APIHND *, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	Handle of the addressed FO, defined by VDSI provider at service Instantiate Function Object.
		arg3:	Identification number for the addressed communication object.
		arg4:	Pointer to a user storage for the global object identifier delivered with service Create Communication Object. This service writes the identifier to the storage.
		arg5:	SYNC or ASYNC, see note to Clause A.7
		arg6:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5

Table A.10 (*continued*)

Service	Function prototype	Relation to service parameters	
Write Data to Communication Object	short GDI_CALL GDI_Write (APIHND, APIHND, APIHND, void *, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	Handle of the addressed FO, defined by VDSI provider at service Instantiate Function Object.
		arg3:	Identification number for the addressed communication object.
		arg4:	Pointer to a user storage containing the data for this communication object.
		arg5:	SYNC or ASYNC, see note to Clause A.7
		arg6:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5
Read Data from Communication Object	short GDI_CALL GDI_Read (APIHND, APIHND, APIHND, void *, APIHND, GDIRESULT *)	arg1:	Handle of the addressed VD, defined by VDSI provider at service Initiate Virtual Device.
		arg2:	Handle of the addressed FO, defined by VDSI provider at service Instantiate Function Object.
		arg3:	Identification number for the addressed communication object.
		arg4:	Pointer to a user storage receiving the data for this communication object.
		arg5:	SYNC or ASYNC, see note to Clause A.7
		arg6:	Pointer to structure GDIRESULT for error messages or other information about this service
		return:	see Table A.5
Report Data to Application	short GDI_CB GDI_InfReport (APIHND, void *)	arg1:	User defined global object identifier for unsolicited services started for this object. The identifier was assigned with service Instantiate Communication Object
		arg2:	Pointer to data for the application
		return:	COM_FIN, if data transport was successful, –1, if global object identifier is not valid, –2, if data transport is temporarily not possible
Request Data from Application	short GDI_CB GDI_Accept (APIHND, void *)	arg1:	User defined global object identifier for unsolicited services started for this object. The identifier was assigned with service Instantiate Communication Object
		arg2:	Pointer to data from the application
		return:	COM_FIN, if data transport was successful, –1, if global object identifier is not valid, –2, if data transport is temporarily not possible

The functions GDI_InfReport and GDI_Accept shall be provided by the user of VDSI to be called by VDS provider in case of local events for unsolicited data transport. The implementation in C/C++ shall use callback-

functions for this task. The address of the callback-functions is presented at VDS provider with calling GDI_ Attach.

A.14.2 Extra function to handle asynchronous communication

The description of Virtual Device Services in 6.3 is neutral to special communication scenarios like synchronous and asynchronous communication. The implementation in C/C++ shall use callback-functions to handle asynchronous communication.

The user of VDSI provides a callback-function as described in Table A.11 to be called by RMS provider in case of completing a communication process started with a service function call. The address of the callback-function is presented at VDS provider with calling GDI_Attach.

Table A.11 — Extra function for asynchronous communication

Service	Function prototype	Relation to service parameters	
no explicit service specified for handling asynchronous communication	short PA_CB io_complete (APIHND, IO_STAT *)	arg1:	communication process identifier provided by RMS user with calling io_read, io_write or io_execute and handed back by RMS provider with calling io_complete
		arg2:	see Table A.3 for the data structure
		return:	COM_FIN

Bibliography

[1] ISO 9506-1, *Industrial automation systems — Manufacturing Message Specification — Part 1: Service definition*

[2] ISO/IEC 9899, *Programming languages — C*

[3] ISO/IEC 10731, *Information technology — Open Systems Interconnection — Basic Reference Model — Conventions for the definition of OSI services*

[4] ISO/IEC 14882, *Information technology — Programming languages — C++*

[5] ISO/IEC 19501, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

[6] ISO 20242-4, *Industrial automation systems and integration — Service interface for testing applications — Part 4: Device capability profile template*

[7] Generic Device Interface Version 4.5 — Association for Standardization of Automation and Measuring Systems (ASAM)

ICS 25.040.40

Price based on 57 pages