

---

---

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

**Part 5:  
Application program service interface**

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

*Partie 5: Interface de service des programmes d'application*



STANDARDSISO.COM : Click to view the full PDF of ISO 20242-5:2020



**COPYRIGHT PROTECTED DOCUMENT**

© ISO 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland



| <b>Contents</b>                                                                                                             | <b>Page</b> |
|-----------------------------------------------------------------------------------------------------------------------------|-------------|
| Foreword.....                                                                                                               | iv          |
| Introduction.....                                                                                                           | v           |
| <b>1 Scope .....</b>                                                                                                        | <b>1</b>    |
| <b>2 Normative references .....</b>                                                                                         | <b>1</b>    |
| <b>3 Terms and definitions.....</b>                                                                                         | <b>1</b>    |
| <b>4 Symbols and abbreviated terms.....</b>                                                                                 | <b>3</b>    |
| <b>5 Application Program Service Interface .....</b>                                                                        | <b>3</b>    |
| 5.1 Introduction .....                                                                                                      | 3           |
| 5.2 Parameterization.....                                                                                                   | 5           |
| 5.3 Configuration .....                                                                                                     | 6           |
| 5.3.1 PID based configuration.....                                                                                          | 6           |
| 5.3.2 Service based and mixed configuration .....                                                                           | 6           |
| 5.4 Coordinator structure.....                                                                                              | 7           |
| 5.4.1 Workspace.....                                                                                                        | 7           |
| 5.4.2 Interface variants .....                                                                                              | 9           |
| 5.5 Details for using APSI .....                                                                                            | 12          |
| 5.5.1 Workspace structure.....                                                                                              | 12          |
| 5.5.2 Callback methods and references.....                                                                                  | 12          |
| 5.5.3 Workspace extension .....                                                                                             | 13          |
| 5.5.4 Workspace transfer .....                                                                                              | 14          |
| 5.5.5 Data monitoring.....                                                                                                  | 14          |
| 5.5.6 Data types.....                                                                                                       | 15          |
| 5.5.7 Streaming .....                                                                                                       | 16          |
| 5.6 Error handling .....                                                                                                    | 18          |
| <b>Annex A (normative) Programmer's reference guide — Smart Access Interface .....</b>                                      | <b>19</b>   |
| <b>Annex B (normative) Programmer's reference guide — Extended Access Interface .....</b>                                   | <b>79</b>   |
| <b>Annex C (normative) Programmer's reference guide — Full Access Interface.....</b>                                        | <b>125</b>  |
| <b>Annex D (normative) Reference guide — Enums and status/ident informations .....</b>                                      | <b>222</b>  |
| <b>Annex E (informative) Sequence diagrams .....</b>                                                                        | <b>235</b>  |
| <b>Annex F (informative) Streaming .....</b>                                                                                | <b>243</b>  |
| <b>Annex G (informative) Data alignment and byte order .....</b>                                                            | <b>247</b>  |
| <b>Annex H (informative) Testing applications with the ISO 20242 (APSI) series and the<br/>ISO 13209 (OTX) series .....</b> | <b>250</b>  |
| <b>Annex I (normative) Conformance test methods, criteria and reports .....</b>                                             | <b>279</b>  |
| <b>Bibliography .....</b>                                                                                                   | <b>283</b>  |

## Foreword

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

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

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

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

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

This document 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*.

A list of all parts in the ISO 20242 series can be found on the ISO website.

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

## Introduction

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

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

- independence from the computer operating system;
- independence from the device connection technology (device interface/network);
- independence from device suppliers;
- the ability to certify device drivers with connected devices and their behavior in the context of a given computer platform;
- independence from the technological device development in the future.

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

STANDARDSISO.COM : Click to view the full PDF of ISO 2042-5:2020

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

## Part 5: Application program service interface

### 1 Scope

This document defines the formatting, syntax and semantic rules for describing

- an object oriented interface for using services provided by a coordinator and
- the configuration of virtual devices and the environment for their use.

### 2 Normative references

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

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

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

ISO 20242-3, *Industrial automation and systems integration — Service interface for testing applications — Part 3: Virtual Device Service Interface*

ISO 20242-4, *Industrial automation and systems integration — Service interface for testing applications — Part 4: Device Capability Profile Template*

ISO 13209-1, *Road vehicles — Open Test sequence eXchange format (OTX) — Part 1: General information and use cases*

ISO 13209-2, *Road vehicles — Open Test sequence eXchange format (OTX) — Part 2: Core data model specification and requirements*

### 3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 20242-1, ISO 20242-2, ISO 20242-3, ISO 20242-4, ISO 13209-1, ISO 13209-2 and the following apply.

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

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <http://www.electropedia.org/>

### 3.1

#### **communication object**

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

[SOURCE: ISO 20242-1:2005, 2.3]

### 3.2

#### **coordinator**

program with a specified interface to handle the access of an application program to one or more *device drivers* (3.5) and to manage real-time application aspects, synchronization and events

[SOURCE: ISO 20242-1:2005, 2.4]

### 3.3

#### **coordinator capability description**

text file containing information about the capabilities of *coordinators* (3.2) in a defined format (i.e. structure, syntax)

[SOURCE: ISO 20242-4:2011, 3.3]

### 3.4

#### **coordinator services**

services of a *coordinator* (3.2) for the exchange of data with application programs

### 3.4

#### **device capability description**

text file containing information about the capabilities of *virtual devices* (3.9) in a defined format (i.e. structure, syntax)

[SOURCE: ISO 20242-1:2005, 2.5, modified — Note 1 to entry deleted.]

### 3.5

#### **device driver**

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

[SOURCE: ISO 20242-2:2010, 3.1]

### 3.6

#### **function object**

instance describing one capability of a *virtual device* (3.9)

[SOURCE: ISO 20242-3:2011, 3.4]

### 3.7

#### **operation**

instance describing one complete procedure

[SOURCE: ISO 20242-3:2011, 3.5]

### 3.8

#### **parameterization instance description**

information about the configuration of a *coordinator* (3.2) and of *virtual devices* (3.9)

[SOURCE: ISO 20242-4:2011, 3.8]

**3.9****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

[SOURCE: ISO 20242-3:2011, 3.7]

**3.10****workspace**

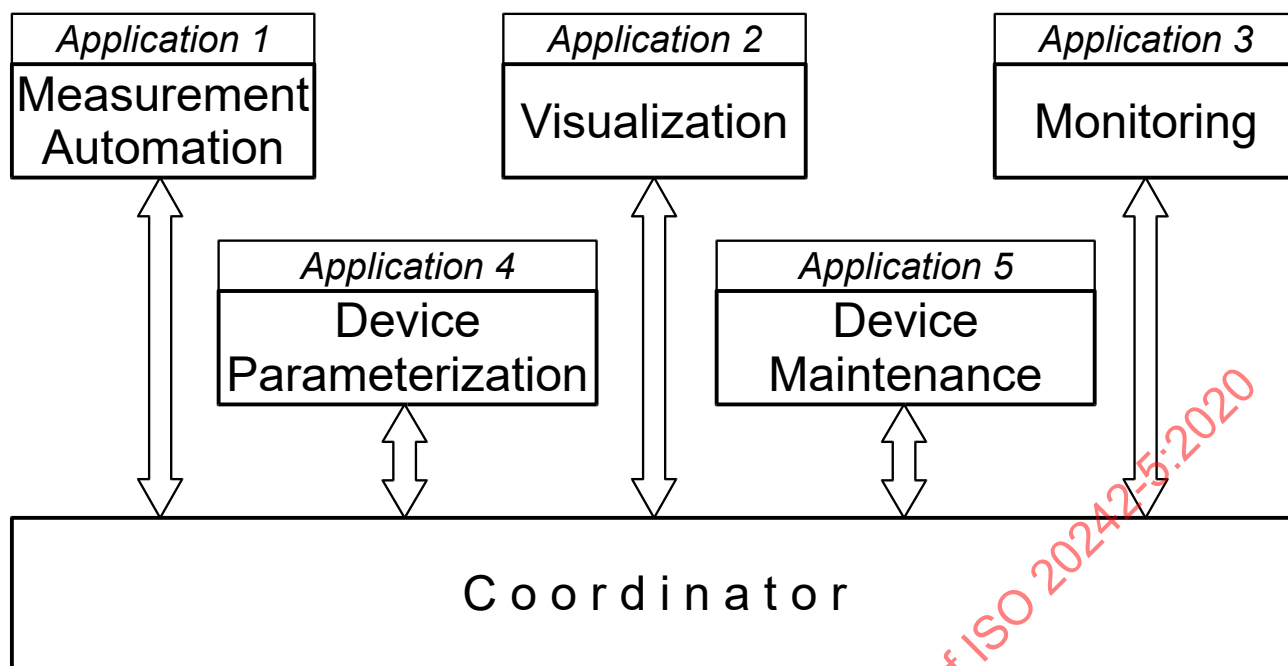
grouping of *coordinators* (3.2) resources providing an access point for an application

**4 Symbols and abbreviated terms**

|       |                                                                  |
|-------|------------------------------------------------------------------|
| APSI  | Application Program Service Interface                            |
| ASCII | American Standard Code for Information Interchange               |
| CCD   | Coordinator Capability Description                               |
| CO    | Communication Object                                             |
| DCD   | Device Capability Description                                    |
| DCPT  | Device Capability Profile Template                               |
| DSL   | Domain Specific Language                                         |
| EAI   | Extended Access Interface                                        |
| FAI   | Full Access Interface                                            |
| FO    | Function Object                                                  |
| NIL   | Null pointer or null reference, does not refer to a valid object |
| OP    | Operation                                                        |
| OTX   | Open Test sequence eXchange                                      |
| PID   | Parameterization Instance Description                            |
| SAI   | Smart Access Interface                                           |
| VD    | Virtual Device                                                   |
| VDSI  | Virtual Device Service Interface                                 |
| XML   | eXtensible Markup Language (see REC-xml-20040204)                |

**5 Application Program Service Interface****5.1 Introduction**

This clause describes a set of services that shall be provided by a coordinator to be used by several application programs (Figure 1).



**Figure 1 — Exemplary application programs using a coordinator**

The concept for the Application Program Service Interface (APSI) is to be device neutral. It shall be possible to access application-relevant data presented by any device without device-specific services or parameters (Applications 1, 2 and 3 in Figure 1). Nevertheless, for the configuration of devices and other device-specific tasks it should also be possible to transfer device-specific information (Applications 4 and 5 in Figure 1). Therefore, APSI defines different interfaces (see 5.4.2), whereas the Smart Access Interface (SAI) is mandatory and shall not handle any device-specific access, and the Full Access Interface (FAI) is optional and shall enable an access to virtual devices in device drivers (see ISO 20242-3). In a typical full stack usage of the ISO 20242 series, the coordinator accesses device drivers via the Virtual Device Service Interface (see Annex I and ISO 20142-1:2005).

The application should use abstract data sources and sinks identified by instance names. Depending on the application environment, data types of sources and sinks are known before or have to be identified via the interface. Applications using APSI typically will not have information about the physical devices providing the requested information processing. In addition, the coordinator shall implement a memory separation between application data and device data (see 5.5.7). Therefore, device-specific data elements shall be encapsulated and not be visible for the application.

This document describes the basics of coordinator services and their usage in typical applications. An object-oriented model is used with a fundamental description of classes, attributes and methods. Services are executable synchronous by default. The asynchronous operation is optional. Corresponding capabilities of a coordinator shall be described in a datasheet delivered with the coordinator (see I.2). Asynchronous operation is controlled via callback functions. For global event handling in a workspace (see 5.4.1), the callback functions (e.g. see `refCBInformationReport` in A.2.2.7) shall be defined with opening the workspace. Additionally local callbacks should be registered at each attribute (see 5.5.2). Coordinator services will typically be mapped to the applied technology, platform and programming language (e.g. CORBA, JAVA, C++). Therefore, this document defines services and not an interface for a specific programming language. Details for a specific programming language shall be explained for the respective mapping in a datasheet (see I.2). This also applies for the error handling, because technological restrictions of data processing have to be considered for the corresponding implementation.

Annex H describes the usage of this standard with applications based on ISO 13209 (OTX). Conformance tests for this document shall be carried out as specified in Annex I.



## 5.2 Parameterization

The coordinator shall be able to link and release applications and device drivers dynamically and to create and delete data objects in device drivers (virtual devices, function objects and communication objects as defined in ISO 20242-3). This procedure shall be identical for all device drivers. The sequence of creating and deleting data objects shall not be fixed. It shall be possible to use a parameter instance description (PID, as defined in ISO 20242-4) to control the procedure of instantiation.

NOTE 1 The coordinator does not know, which device capabilities are necessary for a specific application, but it can communicate with each device on the basis of its device capability description (DCD, as defined in ISO 20242-4).

NOTE 2 To achieve a balance between the functionalities needed by the application and the device capabilities provided by the device drivers, a configuration is usually set up by means of a parameterization procedure. The result is a parameterization instance description (PID) which contains the information for the coordinator to create all necessary data objects. This procedure is the key for device independence, as the application does not need to know the specific capabilities of the used devices. A parameterization tool can create the PID, which can be part of the application itself or a stand-alone software.

The parameterization shall be based on the DCD available for the device drivers and should be supported by multilingual text elements as defined in ISO 20242-4:2011, Clause 8.

NOTE 3 The PID can be created without a coordinator and it is not necessary to connect any device.

The PID shall not be altered by a coordinator.

The following information shall be included within the parameter instance description:

- workspace name,
- drivers (location) with version number,
- device capability descriptions (DCD) of drivers,
- virtual devices (VD) including identification and parameters for creation,
- function objects (FO) including identification and parameters for creation,
- communication objects (CO) and their default values,
- operations (OP) and their identification,
- in/out-structures of operations with values for input parameter,
- initializing order, and
- application references (instance names) for the abstract data sources and sinks.

The parameter instance description (PID) shall be based on the description of structured data types. Thus, the typedef sections of the DCDs shall be resolved. See ISO 20242-4 for details.

NOTE 4 With this parameterization, the initialized values for virtual devices, function objects and communication objects are established for an application-related configuration.

The generated PID shall be stored separately. The read configurations shall be reproducible by the coordinator at any time.

## 5.3 Configuration

### 5.3.1 PID based configuration

With a PID based configuration, an XML file created using the specification in ISO 20242-4 will be presented to the coordinator by the application at the creation of a workspace. This file shall be processed automatically by the coordinator. When an application logs in at the coordinator, the instance description to be used shall be identified by name.

If the application knows the instance names and types of the abstract data sources and sinks, it could iterate over the object references created by the coordinator. This is a typical use case for the Smart Access Interface.

If the application does not know the instance names and types of the abstract data sources and sinks, it should iterate over the object references created by the coordinator and request the names and types of the abstract data sources and sinks. This is a typical use case for the Extended Access Interface.

The coordinator shall instantiate the objects given with the instance description, and thus creates abstract data sources and inside to be used by the application. The data objects of the device driver assigned to these abstract data sources and sinks shall be instantiated within the corresponding virtual devices.

**NOTE** During the life time of the device driver these abstract data sources and sinks can be updated by the application via reading and writing. The coordinator can update the abstract data sources and sinks accessing the data objects instantiated within the device drivers, as the coordinator knows the relations between the abstract data sources and sinks inside and the function objects of the virtual devices.

The operating state of a virtual device will be "Working" (ISO 20242-3:2011, Clause 7) after running a parameterization with the SAI or EAI. In this state, the modification of parameters is not possible. The method `GdiCoWorkspace::gdiSetAllowChangeParameter(boolean bEnable)` resets this prohibition and enables the application to write a parameter (see A.2.29.12). For this, the coordinator shall transit the respective VDs to the appropriate states.

### 5.3.2 Service based and mixed configuration

If the full access interface is implemented in the coordinator, an application itself can do the configuration without an accordingly prepared PID.

In a service based configuration, the structure (classes and data types) and the content (instances and parameters) of the interface can be created. See Annex C for details of this step by step procedure of configuration.

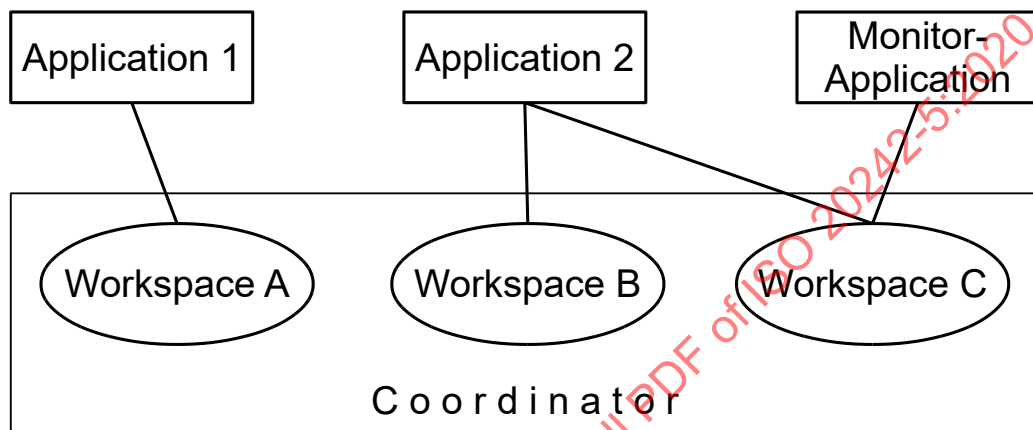
In a mixed configuration, a PID without instances and parameters (called empty PID in this document) can be presented at the interface. The coordinator shall build the structure based on the DCD contained in the empty PID. Instances and parameters can be created afterwards.

For both cases, the operating state of a VD can be controlled by the application.

## 5.4 Coordinator structure

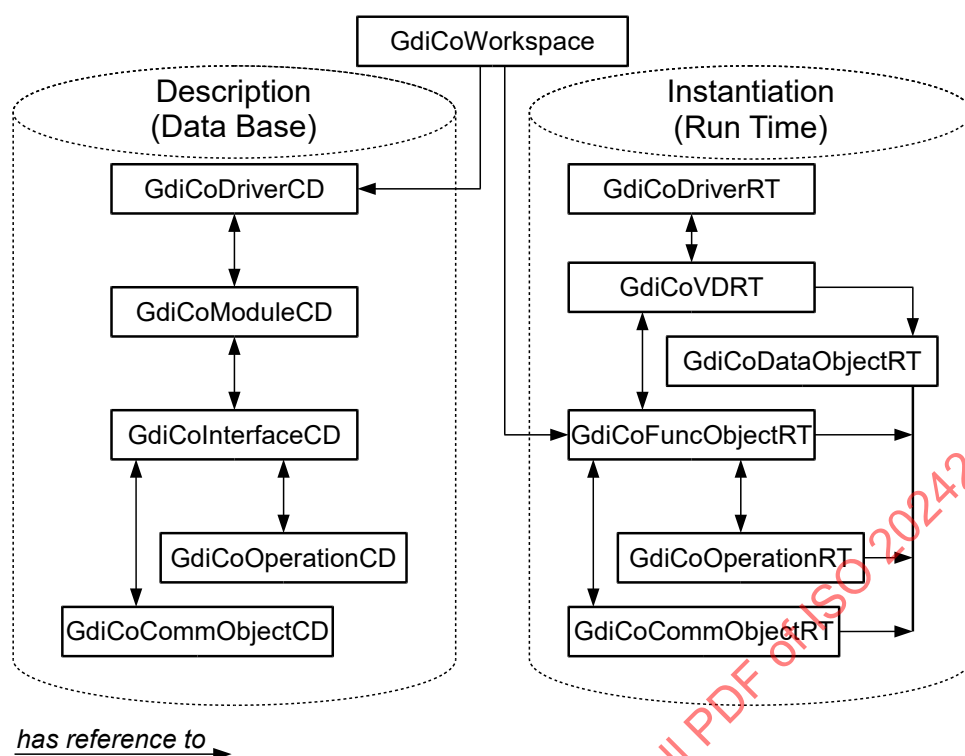
### 5.4.1 Workspace

The workspace in this document is defined as a grouping of coordinator resources and provides an access point for an application. The coordinator shall provide several workspaces which shall be managed separately. An application can use one or more workspaces. Workspaces shall be identified by name and shall connect only to one application (Figure 2) and an optional monitoring application (see 5.5.5 for details).



**Figure 2 — Applications and workspaces**

APSI shall be a multi client interface, as several workspaces can be used at the same time. The content of a PID file shall exactly describe one workspace, independent from the number of devices, device drivers and DCDS.



**Figure 3 — Workspace view for service organisation**

With connecting to a workspace, an application shall get a handle for its further identification. Therefore any interface access after connecting takes this application handle as parameter.

A coordinator shall be able to handle several applications simultaneously. The objects set up by the coordinator for each application shall remain reserved for this application and shall be invisible for other applications. If several applications need access to identical coordinator objects, this should be organized by the application connected to the workspace.

Access services defined in this document shall be carried out as specified in Annex A for the Smart Access Interface (SAI), Annex B for the Extended Access Interface (EAI) and Annex C for the Full Access Interface (FAI). Further services providing information about enums, status and identification shall be carried out as specified in Annex D. The services are organized with a view on two sides of a workspace. One side is called Description and the other side is called Instantiation (see Figure 3).

The Description side is also referred to as Data Base side, because the contained information is static and represents the class descriptions from the DCDs of the corresponding device drivers. The Instantiation side is referred to as RunTime side, because it represents the realized instances including default initial values used at RunTime. If an application uses only the services from the RunTime side, the allocation of the separate device drivers and VDs shall be not visible. Instead a pool of abstract data sources and sinks shall be provided.

Device driver and DCD form a unit. For each device driver there is exactly one Description side, which represents the respective DCD, and exactly one RunTime side. The classes of Description side and the objects of RunTime have a tree structure. Each element is assigned to an unambiguous position in the tree. The GdiCoDriverCD (see C.2.7) is the root element in the Description side and GdiCoDriverRT (see C.2.9) is the root element in the RunTime side.

Instance names of function objects within a workspace (RunTime side) shall be unique over all used drivers.

If an object from RunTime side is set up, there shall be an implicit set up of the corresponding instance within the device driver. All parameters for set up shall be handed over directly by streaming (see Annex F) when the service for the creation of a VD or FO is used.

Annex E describes typical use cases to get more grip on the services and to facilitate the development of coordinator software.

### 5.4.2 Interface variants

This document defines three kinds of interfaces.

For PID based configuration there shall be the

- mandatory Smart Access Interface

and there should be the

- optional Extended Access Interface.

For service based or mixed parameterization there should be the

- optional Full Access Interface.

With view to the structure of APSI, these different interfaces are based upon each other. There is additional functionality within the Extended Access Interface compared to the Smart Access Interface. The classes and methods of individual objects are always identical for each interface variant (e.g. `GdiCoFuncObjectRT`). An application can use any of the interface variants, if they all are implemented. With creating a workspace, the parameter `eRequiredCoordinatorInterface` shall select the required functionality.

With the Extended Access Interface, only the object `GdiCoExtendedObjectNavigator` (see B.2.6, it contains methods for navigation over the objects and data) is defined additional to the Smart Access Interface. With the Full Access Interface the classes `GdiCoObjectNavigator` (see C.2.18, it contains methods for navigation over the objects and data) and `GdiCoFactory` (see C.2.11, it contains methods for adding and deleting of objects) exist in addition to the Extended Access Interface. The Full Access Interfaces shall be able to alter (extend, modify, delete) the RunTime side (Instantiation) and to also alter (extend, modify) the Data Base side (Description). The capabilities of the different interfaces shall be independent from the kind of configuration (PID based, service based or mixed) done after creating the workspace.

To maintain the compatibility between the separate interfaces, the methods

- `gdiGetFactory`
- `gdiGetObjectNavigator` and
- `gdiGetExtendedObjectNavigator`

are included in the class `GdiCoWorkspace` (see A.2.29). They return the references to the objects corresponding to the capabilities of the coordinator, or NIL, if the Coordinator does not support the desired functionality.

With the description of interfaces in this document, the stereotypes explained in Table 1 are used to mark the affiliation of classes and methods to an interface variant.

Table 1 — Classes of interface variants (without data types)

| Stereotype    | Applies to                                                                                                               |
|---------------|--------------------------------------------------------------------------------------------------------------------------|
| << S, E, F >> | classes and methods of Smart Access Interface.<br>Also available in Extended Access Interface and Full Access Interface. |
| << E, F >>    | classes and methods of Extended Access Interface.<br>Also available in Full Access Interface.                            |
| << F >>       | classes and methods of Full Access Interface.<br>Only available in Full Access Interface.                                |

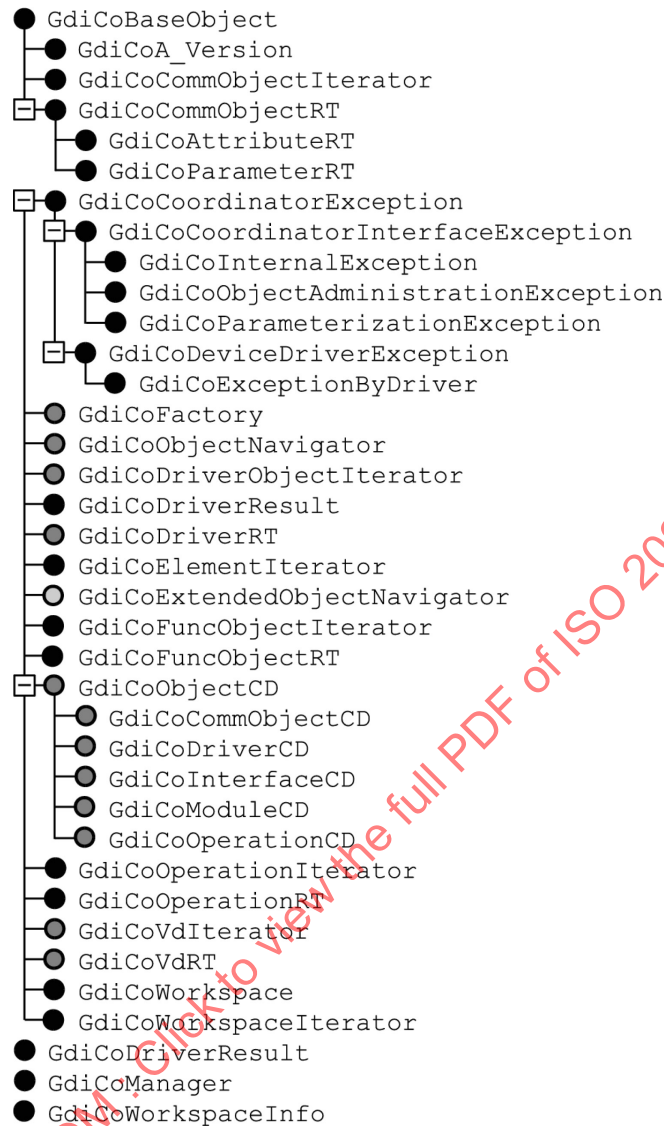
In figures, stereotypes are marked with graphical elements (Figure 4).

- <<abstract>>
- <<E,F>>
- <<enum>>
- <<F>>
- <<S,E,F>>
- <<struct>>
- <<typedef>>

Figure 4 — Graphical marks for stereotypes in figures

Figure 5 shows the classes for all interface variants. If implemented, the coordinator shall provide the interface variant requested by an application. A workspace created for Smart Access Interface shall be open for switching to Full Access Interface, if that is implemented. After using a workspace for Smart Access Interface with configuration by PID, it shall be possible to transfer the workspace to another application for being used with its Full Access Interface to extend the capabilities by additional service based configuration. See 5.5.4 for details.

The method `GdiCoManager::gdiGetCoordinatorType` shall return a bit mask, which visualizes the support of the interface variants (<S>, <E>, <F>).



**Figure 5 — Classes of interface variants (without data types)**

The Application Programm Service Interface with implemented Full Access Interface shall support all services described in Annex C and therefore provides capabilities to

- set up application areas within the coordinator and to generate instances via parameterization data or via services,
- obtain references to function objects and communication objects,
- exchange data with the coordinator,
- initiate the data transport within a device driver,
- obtain descriptions of data (types and names),
- load device drivers,
- install and control virtual devices,

- generate instances (function objects and communication objects),
- set up data elements (assign symbolic names and define types) and
- delete instances.

## 5.5 Details for using APSI

### 5.5.1 Workspace structure

For each device driver and DCD exactly one Description side and one Instantiation side exists in the workspace. These are structured as trees. The application access to the individual objects depends on the used interface variant. See Figure 5 for an overview of the classes and objects which can be used by an application. The names of the classes in Description side end with CD (for capability description) to show their affiliation as the names in RunTime side end with RT.

RunTime objects can be instantiated using the reference to the Description objects, the class name (e.g. name of an interface in the DCD) or the class Id (DCD Id). A coordinator shall provide the following services for the instantiation of RunTime objects:

```
gdiCreateDriverRTByDescription(),  
gdiCreateVdRTByDescription(),  
gdiCreateFuncObjectRTByDescription(),  
gdiCreateCommObjectByDescription() and  
gdiCreateOperationRTByDescription() in GdiCoFactory (see C.2.11).
```

From RunTime side, the respective Description objects may be accessed via methods of the RunTime objects. That is:

```
gdiGetDriverCD() in GdiCoDriverRT (see C.2.9),  
gdiGetModuleCD() in GdiCoVdRT (see C.2.29),  
gdiGetInterfaceCD() in GdiCoCommObjectCD (see C.2.4),  
gdiGetCommObjectCD() in GdiCoObjectNavigator (see C.2.18) and  
gdiGetOperationCD() in GdiCoObjectNavigator (see C.2.18).
```

If the Full Access Interface is used, the Description elements as well as the RunTime objects may be modified.

### 5.5.2 Callback methods and references

An asynchronous execution of services should be implemented, as it provides an easy way for applications to handle concurrent procedures for data transfer.

If asynchronous execution is implemented, it shall be realized via callback methods.

With creating a workspace by method `gdiCreateWorkspace()` in `GdiCoManager` (see A.2.20), the arguments `refCBException`, `refCBAccept` and `refCBInformationReport` shall take callback references to methods within the application for handling event based errors, unsolicited data requests by devices and unsolicited data transmissions from devices. With a NIL reference, the callbacks shall be switched off.

This global assignment of callback references shall be mandatory for the whole life cycle of the workspace. It shall be possible to overwrite the callbacks locally.



The local callback registration shall be done with the

- create methods (VD, FuncObject, ComObject), and the
- asynchronous method calls  
(GdiCoCommObjectRT::gdiReadValue, GdiCoCommObjectRT::gdiWriteValue,  
GdiCoOperationRT::gdiExecuteOperation,  
GdiCoExtendedObjectNavigator::gdiReadCommObjectData,  
GdiCoExtendedObjectNavigator::gdiWriteCommObjectData and  
GdiCoExtendedObjectNavigator::gdiExecuteOperation).

The local callback reference is handed over with this registration and temporarily overwrites the global callback reference for the respective application (identified by its handle). If more applications are registered (see data monitoring 5.5.5), the callbacks shall be distributed to all registered applications. For deregister, the callback reference NIL shall be given. It shall be possible to change local references for InformationReport and Accept by multiple registrations.

With `gdiReleaseWorkspace` or `gdiReleaseMonitorAccess` the registered callbacks shall be made invalid. If an application connects to an existing workspace, the callback references shall be registered again.

The same applies for InformationReport and Accept. The global callback references shall be set at creating the workspace and the local callback references at the `GdiCoAttributeRT` (`gdiSetLocalCBAccept`, `gdiSetLocalCBInfReport`).

### 5.5.3 Workspace extension

It shall be possible to extend existing configurations by using a (new) PID file. This shall not depend on the used interface variant.

For this, the following rules apply:

- The existing configuration shall be maintained;
- No information shall be deleted;
- Configuration elements shall only be added;
- The PID file has to belong to the same devices.

At first the correct syntax and semantic of the PID file shall be checked with respect to the definitions of ISO 20242-4.

For the following configuration check, the PID shall be aligned with the existing configuration of the workspace. The new PID file shall be tested for already existing objects. Instance names and the DCD class descriptions for objects (OP, CO, FO) shall be compared. For already existing objects, the application reference name and the DCD class name for the object and the parent (FO and/or VD) shall be in accordance. If there is any CO or OP with an application reference name that has not been instantiated and the application reference name and DCD class name for the parent (FO/VD) are identical, a re-instantiation shall be done. Any creation parameters for already existing VDs and FOs shall be ignored.

If application reference names for identical class names of CO and/or OP are not in accordance, an error exception shall be thrown (see 5.6). If application reference names for identical class names of FOs are not in accordance but the parent object has an identical application reference name and DCD class name for the VD, this FO shall be added.

If the new PID file holds values for existing COs, these COs shall be written for a value update. Create parameters of existing FOs generally shall not be updated.

All other COs shall not be modified. Also, no implicit default values shall be set (e.g. minimum of a value).

If the new PID file marks execution of operations, they shall be executed.

When method `gdiExtendWorkspace` is finished, the state of the corresponding VD shall be Working (see ISO 20242-4).

If any error concerning accordance is registered for the given PID file, the processing is stopped. The Workspace remains in the VD state Preparation for all VDs concerned. The same action will take place, if any error occurs while running the PID file.

If there is an error, the processing is stopped. All concerned VDs shall be in state Preparation (see ISO 20242-4). If the error is with the syntax and semantic check, the method `gdiExtendWorkspace` shall throw the exception

`eINT_PID_FILE_MISMATCH` (`GdiCoInternalException`, see A.2.19)

and if it is with applying the new configuration, the method shall throw the exception

`ePAR_INCORRECT_PARAMETERIZATION` (`GdiCoParameterizationException`, see A.2.24).

#### 5.5.4 Workspace transfer

It shall be possible that different applications can use the same workspace successively. When an application disconnects from the workspace via method `gdiReleaseWorkspace` (see A.2.20.11), another application may connect to the released workspace using the workspace name and method `gdiGetWorkspaceByName` (see A.2.20.7).

**NOTE** The workspace transfer enables the user of APSI to split big applications into smaller, independent modules which take less resources. Examples are modules for basic configuration, extend configuration, maintain devices and run automation.

Because of the sequential use of the workspace by different applications, there is always a short period of time, where no application is connected. When executing `gdiReleaseWorkspace()`, the coordinator shall set all connected VDs to the state Preparation, and release the workspace afterwards. Release of workspace shall be rejected, if errors hinder the transition to Preparation.

With `gdiGetWorkspaceByName()` the VDs shall be set to Working if SAI or EAI is used (`eSMART` or `eEXTENDED` in parameter `eRequiredCoordinatorInterface`) or shall remain in Preparation if FAI is used (`eFULL` in the parameter `eRequiredCoordinatorInterface`).

The state of a workspace, e.g. if it is just used or not, shall be provided for any application via `GdiWorkspaceIterator` (see A.2.31) and `GdiCoWorkspaceInfo` (see A.2.30).

After release of a workspace, all references shall become invalid. Any access with an invalid handle shall throw the exception `eINT_INVALID_ACCESS` (see A.2.19).

#### 5.5.5 Data monitoring

If an application connects to a workspace via `gdiGetMonitorAccessByWorkspaceName` (see A.2.20.6), it shall have only rights for reading data access (`gdiGetDataValue`, see A.2.9.2, and

InformationReport via callback in refCBInformationReport, see A.2.20.6) and there shall be no influence to the state of the workspace (eNOT\_USED, eUSED). This application is called monitor application.

A monitor application shall have access to the existing trees of the RunTime side and to the data types of the communication parameters and operation parameters. An access to the Database side should be rejected, any access changing state or configuration shall be rejected with error eOAD\_OBJECT\_ACCESS.

For problems with data access, e.g. caused by access of the main application, the error eOAD\_DATAINUSE\_OR\_INCONSISTENT shall be notified.

Exceptions shall be limited to errors occurring at the call of methods by the monitor application and method gdiReleaseMonitorAccess shall release the connection.

InformationReports for communication objects shall be sent to the monitor application only for those objects, which are processed for the main application. The object administration for sending events shall be specific for the main application. Therefore, monitor applications shall receive information reports only for objects,

- which have valid callbacks from the monitor application within the object tree and
- for which the InformationReport has been enabled by the main application.

An additional InformationReport is sent from the coordinator to the monitor, if the main application has processed a successful write or read of an attribute.

The global callback reference for InformationReports (parameter in refCBInformationReport of gdiGetMonitorAccessByWorkspaceName), shall be overwritten related to the attribute when method GdiCoAttributeRT::gdiSetLocalCBInfReport is used to define callbacks.

The number of connected monitor applications shall not be limited. Monitor applications shall stay valid if a workshop transfer is done with main applications.

If the main application wants to delete the workspace via gdiDeleteWorkspace while a monitor application is connected the deletion is rejected with error eINT\_INVALID\_ACCESS.

### 5.5.6 Data types

All defined data types, including FO references, shall be supported. On the Description side, data type descriptions exist from the DCD. On RunTime side, data objects shall exist, which correspond to defined types. Complex types shall be created step-by-step in a tree structure.

The data exchange between application and device driver shall be mediated by the coordinator with providing formatted and allocated communication memory corresponding to the data type descriptions. Data exchange shall be done in two steps:

1. Data is copied from the source (application or driver) to the communication memory of the abstract data sink.
2. The data destination object reads the data from this communication memory.

APSI shall provide services for writing and reading of communication memory within the coordinator.

Methods `gdiGet...Value` and `gdiSet...Value` (see e.g. B.2.4.1) shall handle the formatted access to the communication memory of the coordinator via streaming without launching any activity at the device driver. Further on, each sub-element of a complex data type should set the sub-element values separately (step by step).

After the communication memory has been updated, the data exchange with the device driver (via `GDI_Read` and `GDI_Write`) shall start using

`GdiCoExtendedObjectNavigator::gdiReadCommObjectData` and  
`GdiCoExtendedObjectNavigator::gdiWriteCommObjectData`.

Alternatively, the interface `GdiCoCommObject` shall provide the execution of these two communication steps with only one method `gdiReadValue` or `gdiWriteValue` (i.e. update of the communication memory with application data and data exchange with the device driver). Figure 6 gives an overview on the data type handling of APSI.

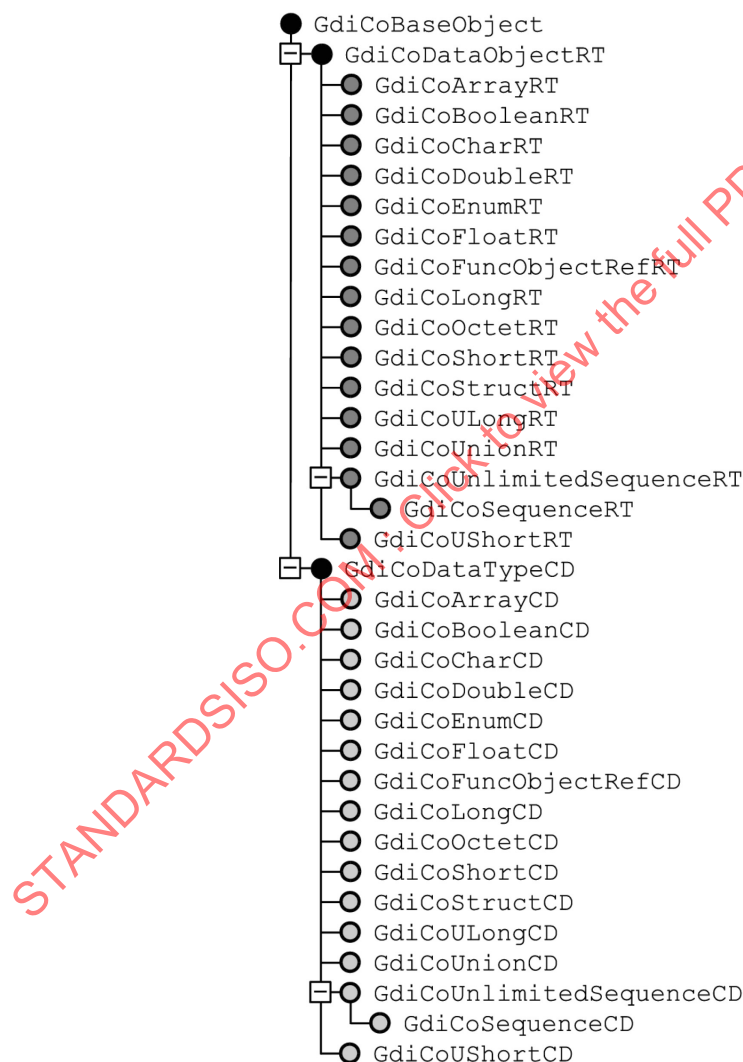


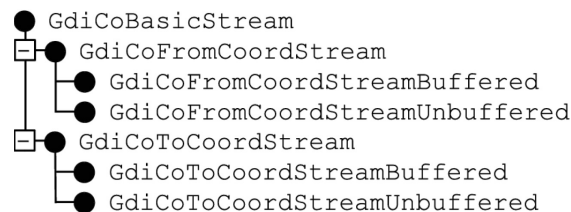
Figure 6 — Data type classes and instantiated data objects

### 5.5.7 Streaming

To enable the formatted application access also for complex data types, streaming shall be used for the data exchange between coordinator and application. Device driver specific data types defined in the DCD shall not be visible for the application.

Stream objects shall be used for the data exchange between application and coordinator. The stream shall be a BLOB (Binary Large Object), which is a sequence of bytes. A stream shall be set up according to the type descriptions in the DCD (see ISO 20242-4). The stream BLOB shall meet the alignment and byte order agreed upon for the respective data exchange (see coordinator datasheet in Annex I).

A non-typified streaming shall be used. No type description shall precede nor shall be transmitted separately or included.



**Figure 7 — Objects for streaming**

Figure 7 shows objects defined for streaming in APSI. A stream object shall enable the stream based data exchange between application and coordinator. Possible implementations are, that the stream object either sets up the connection between two points (data source and sink) and thus provides for an unbuffered data exchange between these two points, or provides a container, which makes it possible to fill data into (package data) and hand over the package to any destination. The addressee may take the data from the package for further processing. This is a buffered data exchange. Buffered streaming is optional for this document.

Stream objects shall enable the platform independent data exchange between application and coordinator. The view of application to data therefore may be arbitrary.

With APSI, two kinds of stream objects are defined (see A.2.14 and A.2.26),

- GdiCoFromCoordStream and
- GdiCoToCoordStream,

which are derived from GdiCoBasicStream.

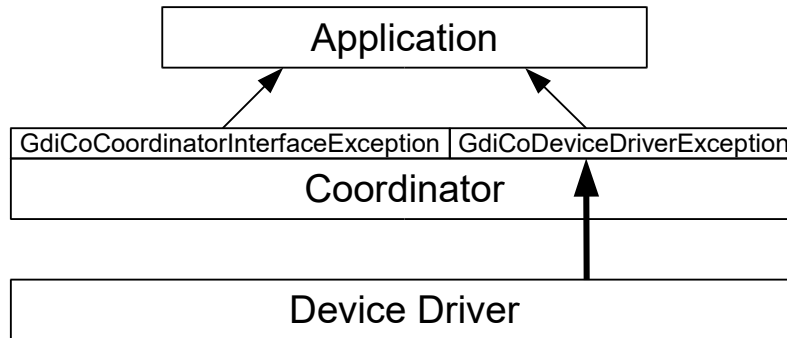
There are additional objects to distinguish between buffered and unbuffered data exchange (see A.2.15, A.2.16 and A.2.27, A.2.28).

An unbuffered streaming is only efficient, if alignment and byte order are identical between application and coordinator. If coordinator and application have different alignments and/or byte order, buffered data exchange should be used.

Using stream objects, the most different alignments and byte orders (see Annex G) may be used. The order of the serialized data within the stream object is determined by the kind of application. The coordinator has to be enabled to adjust to this order by passing on information on the order of the data in the stream object handed over. These context parameters are arguments for the construction of stream objects. The parameters shall precede with constants `cAppAlignment` and `cAppByteOrder`, defined in respective reference implementations (default constants as literal in the header). With the streaming the used alignment and byte order shall be decoded. Further rules for the application of stream objects are defined in Annex F.

## 5.6 Error handling

Errors shall be reported with exceptions. It shall be distinguished between device driver errors (including device exceptions), which are passed to the application via coordinator, and coordinator errors. Device driver errors shall not be reflected in coordinator errors (see Figure 8).



**Figure 8 — Sources for coordinator and device driver errors**

For details see A.2.7 to A.2.10.

Coordinator errors shall be assigned to one of the following classes:

- parameterization (faulty PID file),
- object administration (faulty use of APSI) and
- internal (coordinator, operating system, version, forbidden monitor access).

It is near at hand, that implementations of APSI contain additional internal data check features with respect to specific resources or internal limits. Accordingly support is not defined in APSI, but if a coordinator detects such an internal error, the error code `eINT_PRACTICAL_DATA_OUT_OF_RANGE` shall be used.

## Annex A (normative)

### Programmer's reference guide — Smart Access Interface

#### A.1 General

Single interfaces (respectively classes and their methods and attributes) of the Smart Access Interface, marked with «S», also shall belong to the Extended Access Interface, marked with «E» and the Full Access Interface, marked with «F». Therefore all single interfaces are marked with «S,E,F». The following detailed description is completed by Figures A.1 to A.31 for an overview on classes and their methods.

#### A.2 Detailed description of Small Access Interface

##### A.2.1.1 «S,E,F» GdiCoA\_Version

This interface contains version information of the Application Program Service Interface.

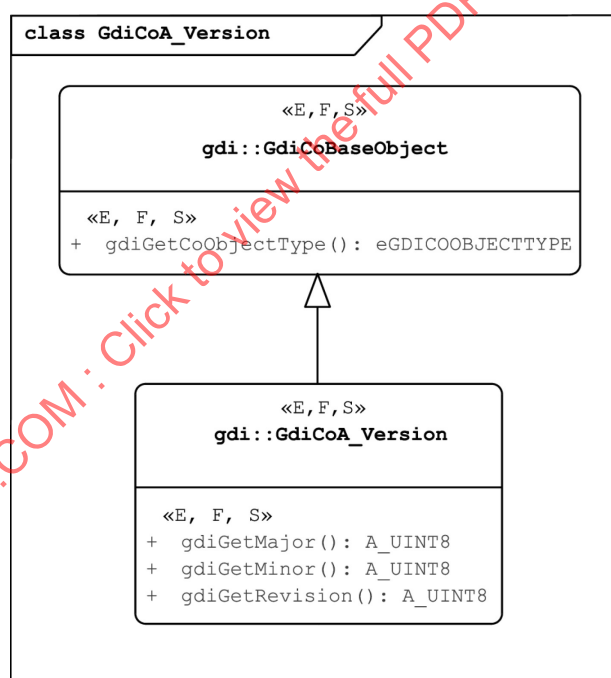


Figure A.1 — Hierarchical diagram of GdiCoA\_Version

##### A.2.1.2 GdiCoA\_Version :: «S,E,F» gdiGetMajor()

###### Function Call

```

gdiGetMajor
(
) : A_UINT8
  
```

Function Description

Returns the Major Version Number.

Return Value

Major Version Number.

**A.2.1.3 GdiCoA\_Version :: «S,E,F» gdiGetMinor()**

Function Call

```
gdiGetMinor  
(  
    ) : A_UINT8
```

Function Description

Returns the Minor Version number.

Return Value

Returns Minor Version number.

**A.2.1.4 GdiCoA\_Version :: «S,E,F» gdiGetRevision()**

Function Call

```
gdiGetRevision  
(  
    ) : A_UINT8
```

Function Description

Returns the Revision Number.

Return Value

Returns Revision Number.

**A.2.2 «S,E,F» GdiCoAttributeRT**

This interface contains services for the handling of attributes. These Communication Objects are distinguished by parameters, so that attributes may additionally exchange data by means of InformationReport and Accept, initiated by the driver.



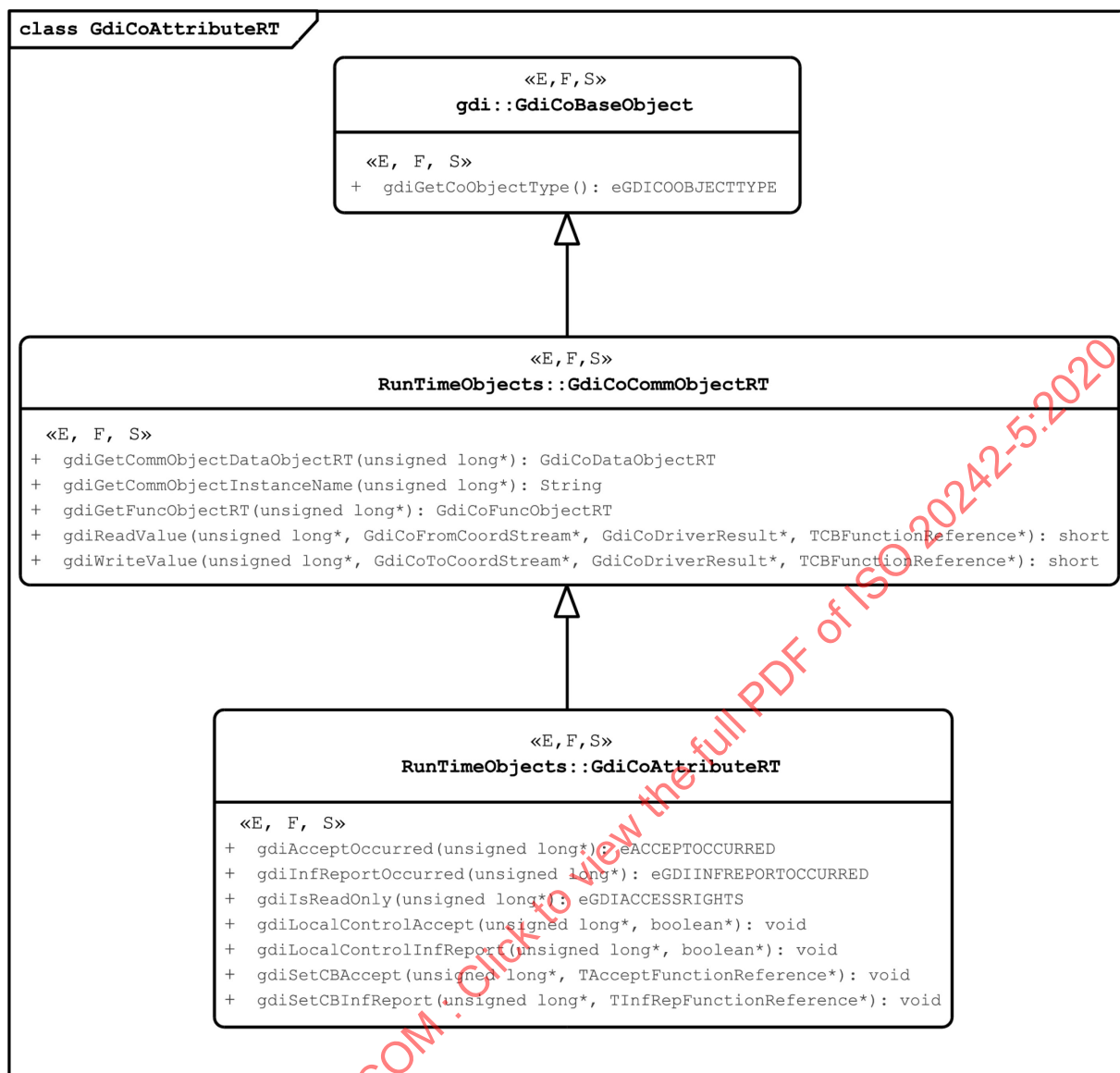


Figure A.2 — Hierarchical diagram of GdiCoAttributeRT

### A.2.2.1 GdiCoAttributeRT :: «S,E,F» gdiAcceptOccurred()

#### Function Call

```

gdiAcceptOccurred
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : eACCEPTOCCURRED
  
```

#### Function Description

Returns the information, if an Accept has been carried out since the last test or **gdiWriteValue**. The service is used, if no call back is supported by the application.

### Return Value

Returns eNOACCEPT (0) if no Accept occurred.  
Returns eACCEPT (1) if Accept occurred.

### **A.2.2.2 GdiCoAttributeRT :: «S,E,F» gdiInfReportOccurred()**

#### Function Call

```
gdiInfReportOccurred
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : eGDIINFREPORTOCCURRED
```

#### Function Description

Returns the information, if an InformationReport has been carried out since the last test or gdiReadValue. The service is used, if no call back is supported by the application.

### Return Value

Returns eNOINFREPORT (0) if no InformationReport occurred.  
Returns eINFREPORT (1) if InformationReport occurred.

### **A.2.2.3 GdiCoAttributeRT :: «S,E,F» gdiIsReadOnly()**

#### Function Call

```
gdiIsReadOnly
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : eGDIACCESSRIGHTS
```

#### Function Description

Returns the information, if a Communication object is read only.

### Return Value

Returns eREADWRITE (0) Read Write.  
Returns eREADONLY (1) Read only.

### **A.2.2.4 GdiCoAttributeRT :: «S,E,F» gdiLocalControlAccept()**

#### Function Call

```
gdiLocalControlAccept
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                boolean bEnableControl
```

```

/* in parameter
true = enable Accept Callbacks
false = disable Accept Callbacks */
) : void

```

### Function Description

Enable/disable the Accept Callback for this Communication Object.

### Return Value

None

## **A.2.2.5 GdiCoAttributeRT :: «S,E,F» gdiLocalControlInfReport()**

### Function Call

```

gdiLocalControlInfReport
(
    inout                unsigned long ulAppHnd
    /* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
    inout                boolean bEnableControl
    /* in parameter
true = enable InformationReport
false = disable InformationReport */
) : void

```

### Function Description

Enable or disable the InformationReport CallBack for this attribute.

If the InformationReport flag is enabled at the GdiCoAttributeRT and the device driver does not support information reports for this attribute, the coordinator shall send an InformationReport to the monitoring client any time the attribute is changed by another client.

If the device driver sends an InformationReport to this attribute, the information shall be delivered to all monitoring clients which have enabled the InformationReport flag.

### Return Value

None

## **A.2.2.6 GdiCoAttributeRT :: «S,E,F» gdiSetCBAccept()**

### Function Call

```

gdiSetCBAccept
(
    inout                unsigned long ulAppHnd
    /* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
    inout                TAcceptFunctionReference refCBAccept
    /* in parameter
Reference to a call back method which handles the accept mechanism. A
NIL reference signals that no call back is supported or the global
callback reference is to use.

```

```
At the coordinator call back to the application, an reference to the
specific GdiCoCommObjectRT is passed. */
) : void
```

#### Function Description

Register a local Accept callback for this communication object. The global call back (given by `gdiCreateWorkspace`) shall be deactivated.

#### Return Value

None

### **A.2.2.7 GdiCoAttributeRT :: «S,E,F» gdiSetCBInfReport()**

#### Function Call

```
gdiSetCBInfReport
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                TInfRepFunctionReference refCBInformationReport
    /* in parameter
    Reference to a call back method which handles the informationreport
    mechanism. A NIL reference signals that no call back is supported or the
    global callback reference is to use.
    At the coordinator call back to the application, an reference to the
    specific GdiCoCommObjectRT is passed. */
) : void
```

#### Function Description

Register a local InformationReport callback for this communication object. The global call back (given by `gdiCreateWorkspace`) shall be deactivated.

If the InformationReport flag is enabled at the GdiCoAttributRT and the device driver does not support information reports for this attribute, the coordinator shall send an InformationReport to the monitoring client any time the attribute is changed by another client.

If the device driver sends an InformationReport to this attribute, the information will be delivered to all monitoring clients, which have enabled the InformationReport flag.

#### Return Value

None

### **A.2.3 «S,E,F» GdiCoBaseObject**

This interface is the base interface of all coordinator objects. This services occurs in inheritance only.



Figure A.3 — Hierarchical diagram of GdiCoBaseObject

### A.2.3.1 GdiCoBaseObject :: «S,E,F» gdiGetCoObjectType()

#### Function Call

```

gdiGetCoObjectType
(
) : eGDICOBJECTTYPE
  
```

#### Function Description

Returns the enumeration for detecting the type of the Coordinator Object reference.

#### Return Value

Returns a value of GDICOBJECTTYPE.

### A.2.4 «S,E,F» GdiCoBasicStream

This interface is the base interface of the stream objects. These services only occur in inheritance.

For monitor access the data is duplicated to all clients.

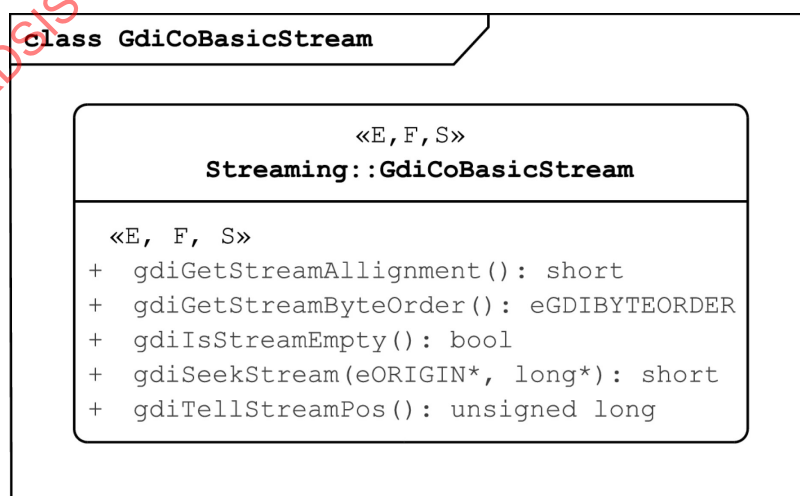


Figure A.4 — Hierarchical diagram of GdiCoBasicStream

#### A.2.4.1 GdiCoBasicStream :: «S,E,F» gdiGetStreamAllignment()

##### Function Call

```
gdiGetStreamAllignment  
(  
    ) : short
```

##### Function Description

Returns the Allignment of the Data in the Stream Object.

##### Return Value

Returns the Allignment of the Data in the Stream Object.

#### A.2.4.2 GdiCoBasicStream :: «S,E,F» gdiGetStreamByteOrder()

##### Function Call

```
gdiGetStreamByteOrder  
(  
    ) : eGDIBYTEORDER
```

##### Function Description

Returns the ByteOrder of the Data in the Stream.

##### Return Value

Returns the ByteOrder of the Data in the Stream.

#### A.2.4.3 GdiCoBasicStream :: «S,E,F» gdiIsStreamEmpty()

##### Function Call

```
gdiIsStreamEmpty  
(  
    ) : bool
```

##### Function Description

This method is used to check the type compatibly of the Date in stream against the Data in the Application.

If the return value is true after a communication, the type is with a high probability correctly interpreted.

(similar to "End Of File" Flag)

##### Return Value

true = Stream is empty.

false = Stream is not empty.

**A.2.4.4 GdiCoBasicStream :: «S,E,F» gdiSeekStream()****Function Call**

```

gdiSeekStream
(
    inout                eORIGIN eOriginator
    /* in parameter
    contains the initial position. */
    inout                long lPos
    /* in parameter
    contains the position in which byte to move the stream pointer from
    origin. */
) : short

```

**Function Description**

Moves the stream pointer in a stream to a specific location.

**Return Value**

0 : sucessful seeked  
 != 0 : error

**A.2.4.5 GdiCoBasicStream :: «S,E,F» gdiTellStreamPos()****Function Call**

```

gdiTellStreamPos
(
) : unsigned long

```

**Function Description**

Report current position in output stream.

**Return Value**

Returns the current position in the stream.

**A.2.5 «S,E,F» GdiCoCommObjectIterator**

This interface contains the service for the application of a CommObject iterator, sequential output of all Communication Objects which have been set up within a Function Object.

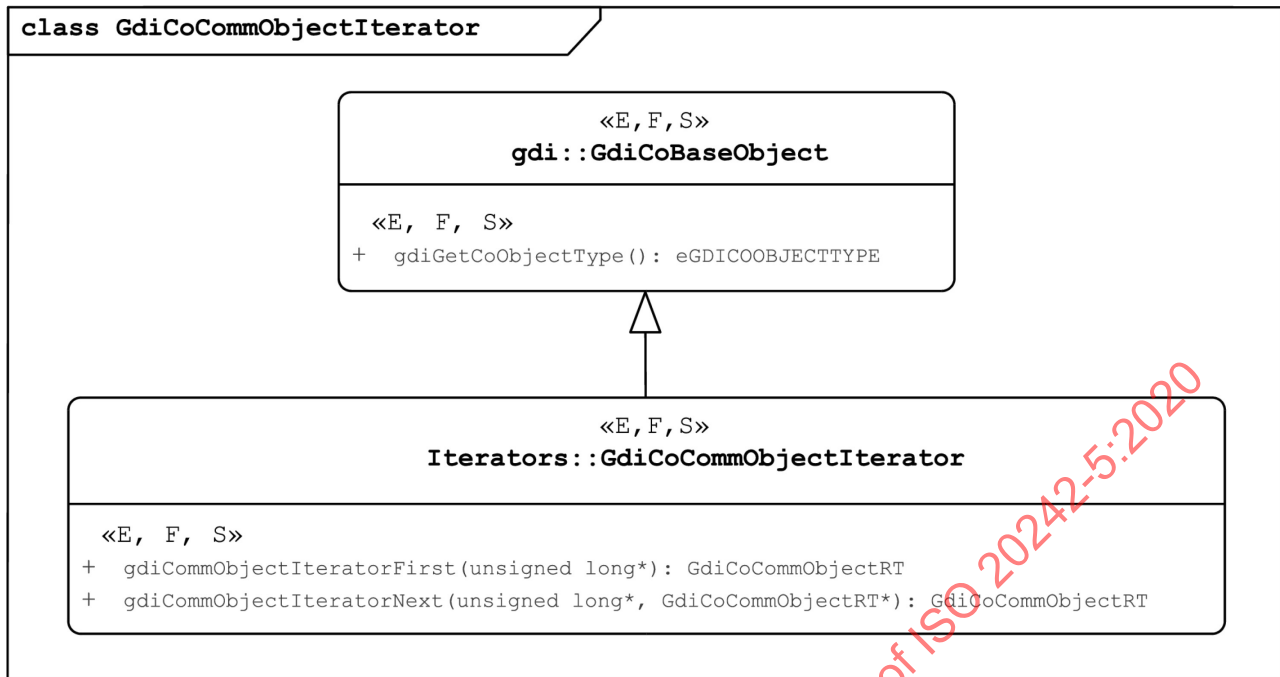


Figure A.5 — Hierarchical diagram of GdiCoCommObjectIterator

#### A.2.5.1 GdiCoCommObjectIterator :: «S,E,F» gdiCommObjectIteratorFirst()

##### Function Call

```

gdiCommObjectIteratorFirst
(
    inout          unsigned long ulAppHnd
    /* in parameter
       Identifier of the application returned by gdiCreateWorkspace(...) or by
       gdiGetWorkspaceByName() */
) : GdiCoCommObjectRT
  
```

##### Function Description

Sets the internal pointer to the first ComObject within the Workspace.

##### Return Value

Returns a reference to the first GdiCoCommObject if it exists, otherwise a NIL Reference returns.

#### A.2.5.2 GdiCoCommObjectIterator :: «S,E,F» gdiCommObjectIteratorNext()

##### Function Call

```

gdiCommObjectIteratorNext
(
    inout          unsigned long ulAppHnd
    /* in parameter
       Identifier of the application returned by gdiCreateWorkspace(...) or by
       gdiGetWorkspaceByName() */
    inout          GdiCoCommObjectRT refLastElement
    /* in parameter
  
```



```

A reference to a element returned by gdiComObjectIteratorFirst or
gdiComObjectIteratorNext. */
) : GdiCoCommObjectRT

```

### Function Description

Detects the reference to the current ComObject (Parameter) within the function object. Sets the internal pointer to the next ComObject within the FO (if existing) and returns the reference.

### Return Value

Returns reference to a coordinator CommObject (Administration unit within the coordinator).

## A.2.6 «S,E,F» GdiCoCommObjectRT

This interface contains services for the use of Communication Objects. These services only occur in inheritance.

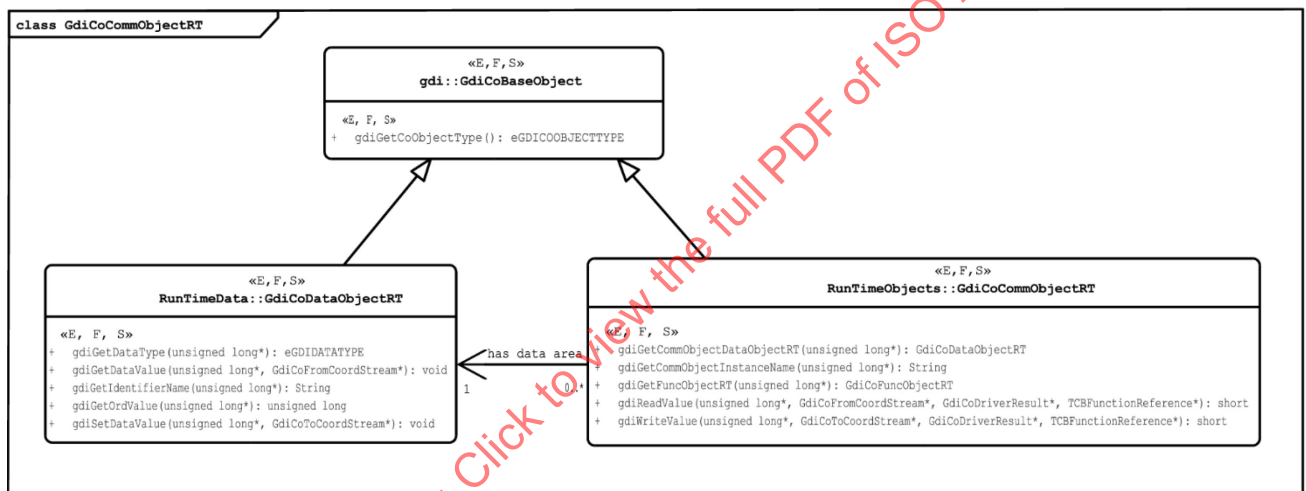


Figure A.6 — Hierarchical diagram of GdiCoCommObjectRT

### A.2.6.1 GdiCoCommObjectRT :: «S,E,F» gdiGetCommObjectDataObjectRT()

#### Function Call

```

gdiGetCommObjectDataObjectRT
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataObjectRT

```

### Function Description

Returns the reference to the corresponding data object of the given communication object.

### Return Value

Returns the reference to the corresponding data object.

### A.2.6.2 GdiCoCommObjectRT :: «S,E,F» gdiGetCommObjectInstanceName()

#### Function Call

```
gdiGetCommObjectInstanceName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String
```

#### Function Description

Returns the instance name of the object defined during parameterization.

Only one ComObject instance is possible so the attribute/parameter name inside the interface and object name can be identical.

#### Return Value

Returns the name of the object.

### A.2.6.3 GdiCoCommObjectRT :: «S,E,F» gdiGetFuncObjectRT()

#### Function Call

```
gdiGetFuncObjectRT
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoFuncObjectRT
```

#### Function Description

Creates reference to the existing Function Object.

The referenced Function Object contains this Communication Object. (used to evaluate the corresponding Function Object to the Communication Object delivered by an Information Report).

#### Return Value

Returns the reference to the GdiCoFuncObjectRT if succesful.

### A.2.6.4 GdiCoCommObjectRT :: «S,E,F» gdiReadValue()

#### Function Call

```
gdiReadValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoFromCoordStream refDestValue
    /* out parameter
```

```

Reference to a stream object, the type is identical to the type of the
Communication object. */
inout          GdiCoDriverResult refDestInfo
/* out parameter
contains the reference to the Information Object, warnings and
informations of the Driver will be stored in it. */
inout          TCBCFunctionReference refCBCCompleteRead
/* in parameter
Reference to a call back method which handles the read mechanism. A NIL
reference signals a synchronous call. */
) : short

```

### Function Description

Reads a Communication object from the driver and returns a data stream by reference. The data in the stream is a sequence of bytes.

### Return Value

Describes the executed communication by the GDI driver:

0 = synchronous return;

1 = asynchronous return.

## **A.2.6.5 GdiCoCommObjectRT :: «S,E,F» gdiWriteValue()**

### Function Call

```

gdiWriteValue
(
inout          unsigned long ulAppHnd
/* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          GdiCoToCoordStream refSourceValue
/* in parameter
Reference to a stream object, the type is identical to the type of the
Communication object. */
inout          GdiCoDriverResult refDestInfo
/* out parameter
contains the reference to the Information Object, warnings and
informations of the Driver will be stored in it. */
inout          TCBCFunctionReference refCBCCompleteWrite
/* in parameter
Reference to a call back method which handles the write mechanism. A NIL
reference signals synchronous call. */
) : short

```

### Function Description

Writes a data value (by reference) of a communication object to the coordinator and starts the transfer to the driver.

### Return Value

Describes the executed communication by the GDI driver:

0 = synchronous return;

1 = asynchronous return.

### A.2.7 «S,E,F» GdiCoCoordinatorException

This is the base error class and shall be used for error handling. See also 5.6 for the separated processing of coordinator internal errors and device driver errors.

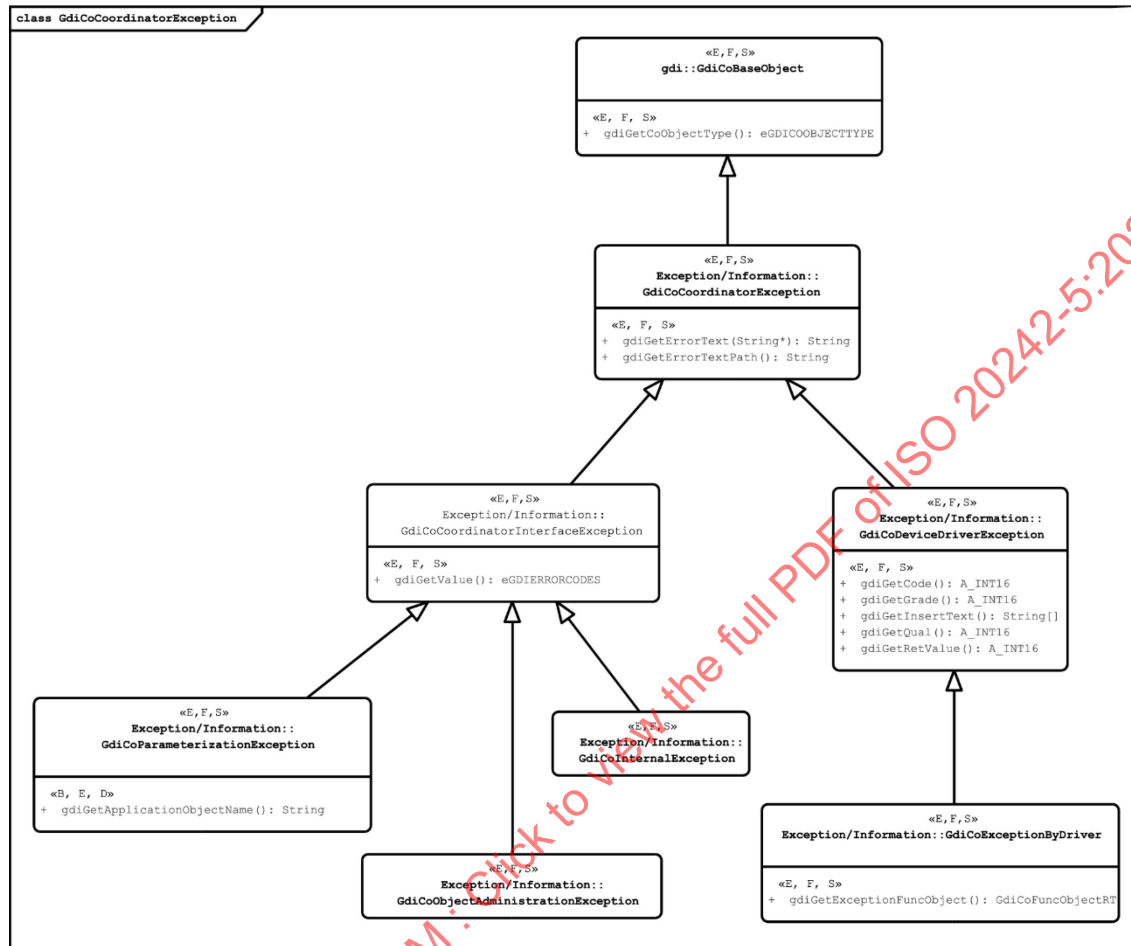


Figure A.7 — Hierarchical diagram of GdiCoCoordinatorException

#### A.2.7.1 GdiCoCoordinatorException :: «S,E,F» gdiGetErrorText()

##### Function Call

```

gdiGetErrorText
    inout          String strLanguage
    /* in parameter
    requested language token according to ISO 639. */
    ) : String
  
```

##### Function Description

Returns the corresponding String in the corresponding Language including the replace text from the standard DIT if exist.

Otherwise an empty String is returned.

Return Value

Returns the corresponding String in the corresponding Language including the replace text from the standard DIT if exist.

Otherwise an empty String is returned.

**A.2.7.2 GdiCoCoordinatorException :: «S,E,F» gdiGetErrorTextPath()**Function Call

```
gdiGetErrorTextPath
(
) : String
```

Function Description

Returns a path reference as String to a DIT or DII to the corresponding error message.

This path can be used to ask an external DIT/DII error server for the language specific error message.

Return Value

Returns a path reference as String to a DIT or DII to the corresponding error message.

This path can be used to ask an external DIT/DII error server for the language specific error message.

**A.2.8 «S,E,F» GdiCoCoordinatorInterfaceException**

This exception class shall be thrown if coordinator specific errors occur. Details of errors shall be described with child classes.

Figure A.8 — Hierarchical diagram of `GdiCoCoordinatorInterfaceException`A.2.8.1 `GdiCoCoordinatorInterfaceException :: <<S,E,F>> gdiGetValue()`Function Call

```

gdiGetValue
(
) : eGDIERRORCODES

```

Function Description

Returns the error Value as a number.

See the `eGDIERRORCODES` enumeration for details.

Return Value

Returns the error codes as defined in the `eGDIERRORCODES` enumeration.

A.2.8.2 `GdiCoDataObjectRT :: <<S,E,F>> gdiGetOrdValue()`Function Call

```

gdiGetOrdValue
(
    inout                unsigned long ulAppHnd
)

```

```

/* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
) : unsigned long

```

### Function Description

Returns the internal position value of the type (sub type) in a structure or union. Is used for the detection of the switch value of a sub type within a union. Within a structure this value is used for information only. If the type is a sub type in sequences or arrays, this service returns the value 0.

### Return Value

Returns the internal position of this type as sub type within a structure or union.

## **A.2.8.3 GdiCoDataObjectRT :: «S,E,F» gdiSetDataValue()**

### Function Call

```

gdiSetDataValue
(
    inout                unsigned long ulAppHnd
/* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
    inout                GdiCoToCoordStream refSourceDataStream
/* in parameter
Contains the reference to the source Data Stream. */
) : void

```

### Function Description

Sets the value of a data area within the Coordinator with the value handed over by the stream reference. The data are handled as a block of Bytes. The number of Bytes is determined by the type of the datum.

### Return Value

None

## **A.2.9 «S,E,F» GdiCoDataObjectRT**

This interface contains services for the access of data within the coordinator.



Figure A.9 — Hierarchical diagram of GdiCoDataObjectRT

#### A.2.9.1 GdiCoDataObjectRT :: «S,E,F» gdiGetDataType()

##### Function Call

```

gdiGetDataType
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : eGDIDATATYPE
  
```

##### Function Description

Returns the type as a numeric value (enum).

##### Return Value

Returns types as a numeric value (described in Semantics).

#### A.2.9.2 GdiCoDataObjectRT :: «S,E,F» gdiGetDataValue()

##### Function Call

```

gdiGetDataValue
  
```



```

(
  inout                unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout                GdiCoFromCoordStream refDestDataStream
  /* in parameter
  Contains the reference to the destination Data Stream in the application.
  */
) : void

```

### Function Description

Returns the value of a data area within the Coordinator with as stream reference. The data are handled as a block of Bytes. The number of Bytes is determined by the type of the datum.

### Return Value

None

### **A.2.9.3 GdiCoDataObjectRT :: «S,E,F» gdiGetIdentifierName()**

#### Function Call

```

gdiGetIdentifierName
(
  inout                unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
) : String

```

### Function Description

Returns the name of the type. Is used for the identification of derived types or sub-types.

### Return Value

Returns a text string (Pointer, class, address of sz area).

### **A.2.10 «S,E,F» GdiCoDeviceDriverException**

This shall be used for throwing exceptions caused by device driver errors.

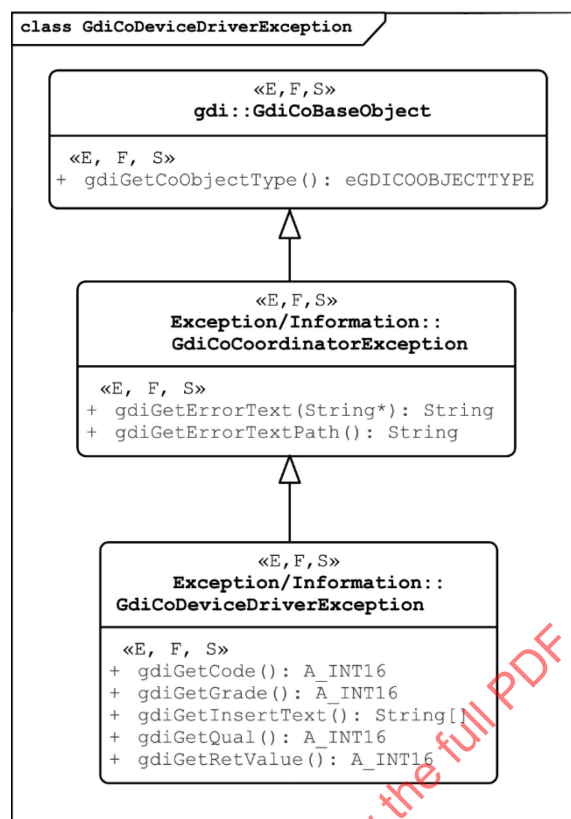


Figure A.10 — Hierarchical diagram of GdiCoDeviceDriverException

#### A.2.10.1 GdiCoDeviceDriverException :: «S,E,F» gdiGetCode()

##### Function Call

```

gdiGetCode
(
) : A_INT16

```

##### Function Description

Returns the corresponding value of the GDI API error.

##### Return Value

Returns the errorcode of a gdi device driver api function.

#### A.2.10.2 GdiCoDeviceDriverException :: «S,E,F» gdiGetGrade()

##### Function Call

```

gdiGetGrade
(
) : A_INT16

```

##### Function Description

Returns the corresponding value of the GDI API error.

Return Value

Returns the errorcode of a GDI device driver API function.

**A.2.10.3 GdiCoDeviceDriverException :: «S,E,F» gdiGetInsertText()**Function Call

```
gdiGetInsertText
(
) : String[]
```

Function Description

Returns Replace text from GDI Driver as list of strings.

Return Value

Returns a list of String representing the replace text from GDI Driver.

**A.2.10.4 GdiCoDeviceDriverException :: «S,E,F» gdiGetQual()**Function Call

```
gdiGetQual
(
) : A_INT16
```

Function Description

Returns the corresponding value of the Device Driver API error.

Return Value

Returns the errorcode of a GDI device driver API function.

**A.2.10.5 GdiCoDeviceDriverException :: «S,E,F» gdiGetRetVal()**Function Call

```
gdiGetRetVal
(
) : A_INT16
```

Function Description

Returns the corresponding value of the GDI API error.

Return Value

Returns the errorcode of a GDI device driver API function.

**A.2.11 «S,E,F» GdiCoDriverResult**

This interface shall transport warnings and informations from the device driver.

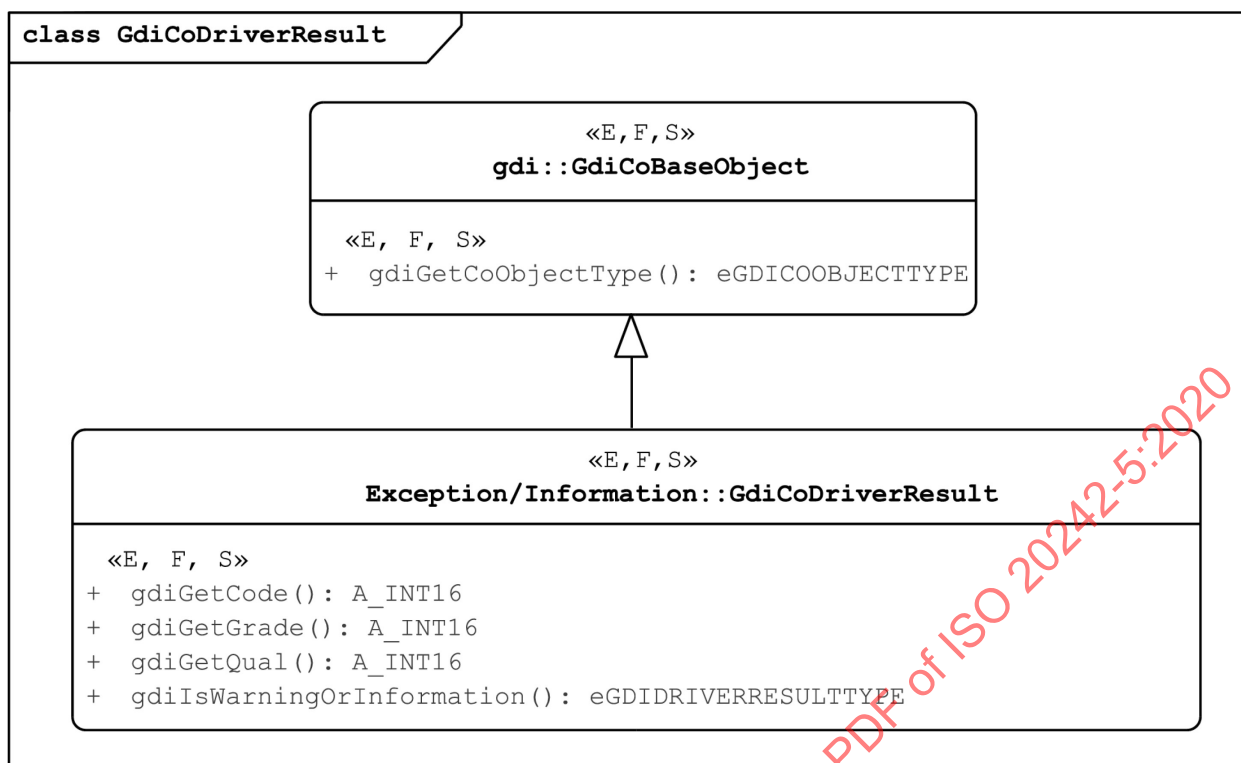


Figure A.11 — Hierarchical diagram of GdiCoDriverResult

#### A.2.11.1 GdiCoDriverResult :: «S,E,F» gdiGetCode()

##### Function Call

```

gdiGetCode
(
) : A_INT16

```

##### Function Description

Returns the corresponding value of the GDI API information.

##### Return Value

Returns the information of a GDI device driver API function.

#### A.2.11.2 GdiCoDriverResult :: «S,E,F» gdiGetGrade()

##### Function Call

```

gdiGetGrade
(
) : A_INT16

```

##### Function Description

Returns the corresponding value of the GDI API information.

Return Value

Returns the information code of a GDI device driver API function.

**A.2.11.3 GdiCoDriverResult :: «S,E,F» gdiGetQual()**Function Call

```
gdiGetQual
(
    ) : A_INT16
```

Function Description

Returns the corresponding value of the GDI API information.

Return Value

Returns the information code of a GDI device driver API function.

**A.2.11.4 GdiCoDriverResult :: «S,E,F» gdiIsWarningOrInformation()**Function Call

```
gdiIsWarningOrInformation
(
    ) : eGDIDRIVERRESULTTYPE
```

Function Description

Returns the enumeration about the information, if a warning or a information has occurred.

Return Value

Returns the enumeration about the information, if a warning or a information has occurred.

**A.2.12 «S,E,F» GdiCoElementIterator**

This interface shall contain the service for the application of a type iterator, sequential output of all sub types which have been set up within a structure or union.

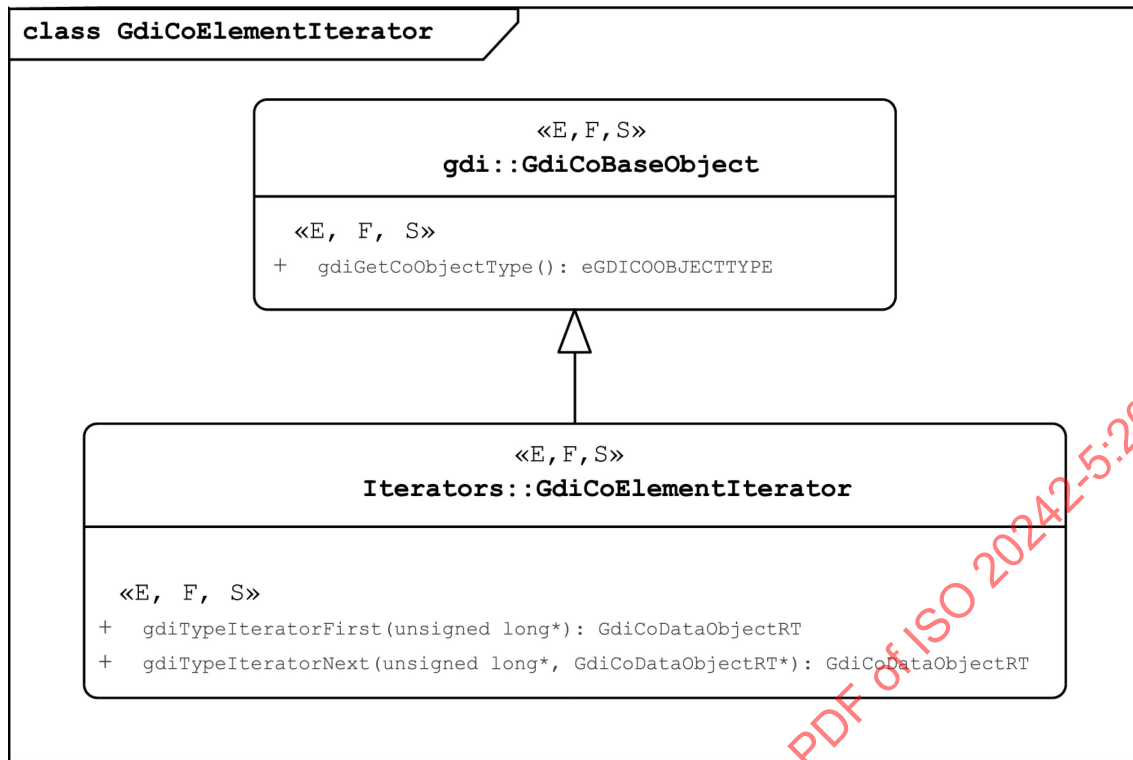


Figure A.12 — Hierarchical diagram of GdiCoElementIterator

#### A.2.12.1 GdiCoElementIterator :: «S,E,F» gdiTypeIteratorFirst()

##### Function Call

```

gdiTypeIteratorFirst
(
    inout          unsigned long ulAppHnd
    /* in parameter
       Identifier of the application returned by gdiCreateWorkspace(...) or by
       gdiGetWorkspaceByName() */
) : GdiCoDataObjectRT
  
```

##### Function Description

Sets the internal pointer to the first Type within the Control interface.

##### Return Value

Returns a reference to the first GdiCoBaseType if it exists, otherwise a NIL Reference returns.

#### A.2.12.2 GdiCoElementIterator :: «S,E,F» gdiTypeIteratorNext()

##### Function Call

```

gdiTypeIteratorNext
(
    inout          unsigned long ulAppHnd
    /* in parameter
  
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          GdiCoDataObjectRT refLastElement
/* in parameter
A reference to a element returned by gdiTypeIteratorFirst or
gdiTypeIteratorNext. */
) : GdiCoDataObjectRT

```

### Function Description

Detects the reference to the current Type (Parameter) within the Control interface. Sets the internal pointer to the next Type (if existing) and returns the reference.

### Return Value

Returns the reference to a coordinator Base Type (Administration unit within the coordinator).

## A.2.13 «S,E,F» GdiCoExceptionByDriver

This interface shall throw exceptions for user-defined function objects. The device driver may throw one or more FuncObject exceptions described in the DCD. The coordinator shall generate one dynamic function object and shall throw a GdiExceptionByDriver with reference to the FO.

After releasing the GdiExceptionByDriver object, the coordinator shall delete the dynamic function objects.

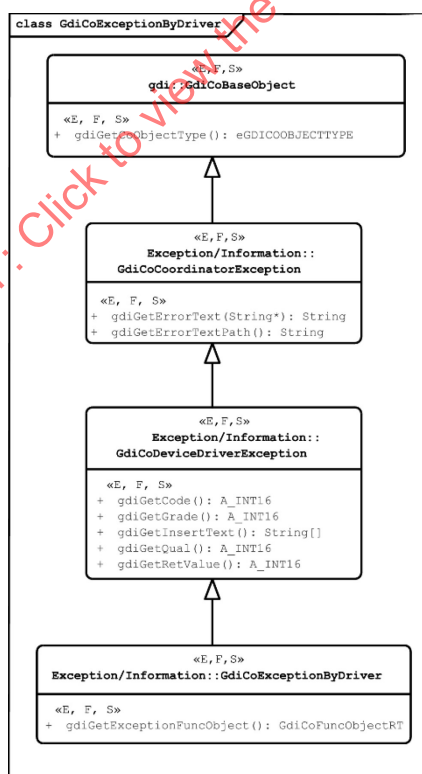


Figure A.13 — Hierarchical diagram of GdiCoExceptionByDriver

A.2.13.1 GdiCoExceptionByDriver :: «S,E,F» gdiGetExceptionFuncObject()

Function Call

```
gdiGetExceptionFuncObject
(
    ) : GdiCoFuncObjectRT
```

Function Description

Returns the reference to a FuncObject at the given position, which the GDI Driver has thrown as exception.

Return Value

Returns the reference to an object of GdiCoFuncObject. In case of an error, the function returns NIL.

A.2.14 «S,E,F» GdiCoFromCoordStream

This interface is used to read an unformatted byte sequence from the coordinator.

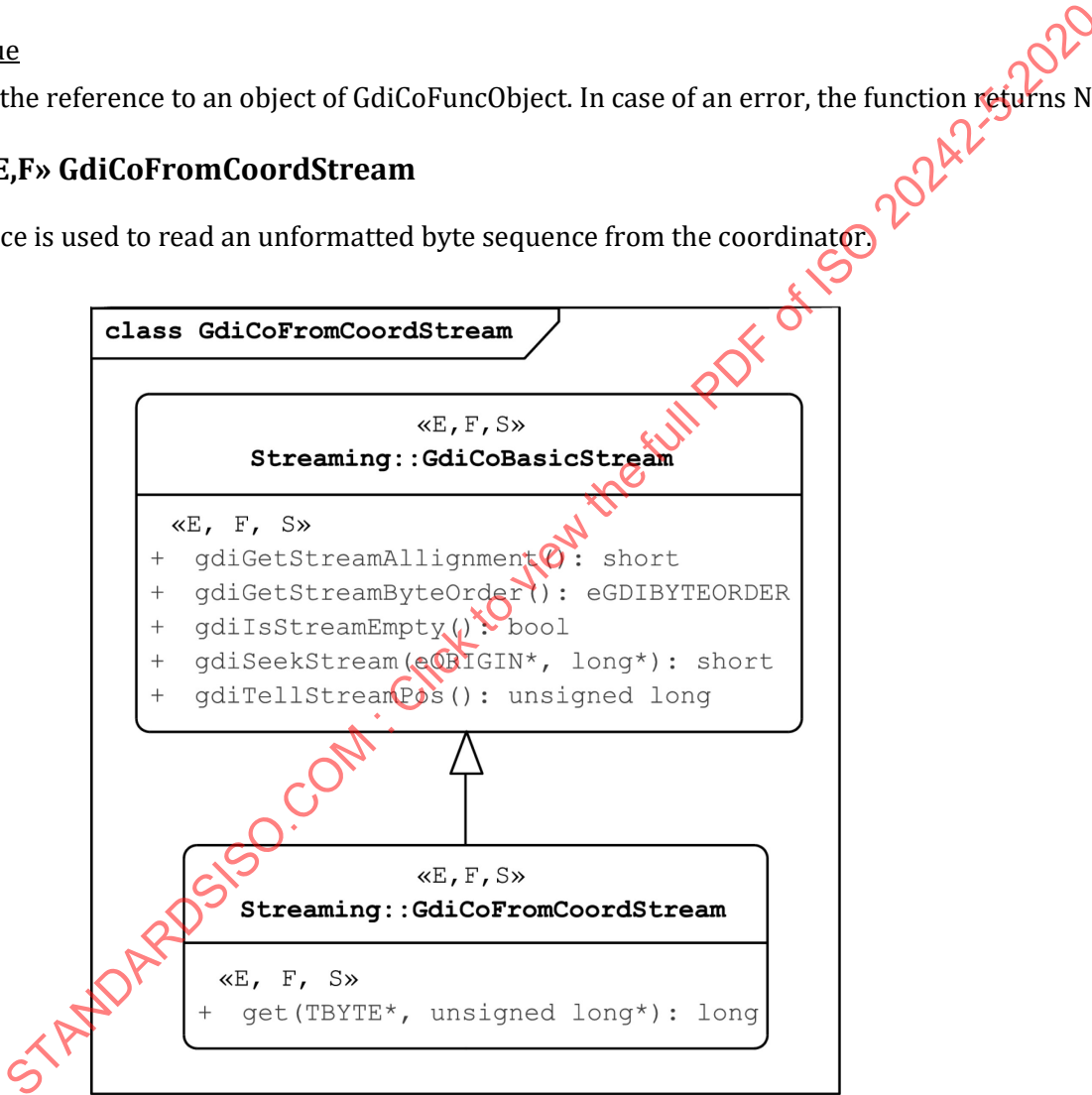


Figure A.14 — Hierarchical diagram of GdiCoFromCoordStream

A.2.14.1 GdiCoFromCoordStream :: «S,E,F» get()

Function Call

```
get
(
```



```

inout          TBYTE refDestination
/* in parameter
contains the reference to the destination buffer in the
application. */
inout          unsigned long ulnByteCount
/* in parameter
contains the requested number of bytes to be read. */
) : long

```

### Function Description

Gets the data from the stream to the application.

### Return Value

Returns the count of Elements, which was read.

A negative number describes an error.

## A.2.15 «S,E,F» GdiCoFromCoordStreamBuffered

This interface is used to read an unformatted byte sequence from the coordinator. The data in the stream object shall be updated by the coordinator with the next `gdiRead`, `gdiExecute` and `gdiGetDataValue` (in extended access). The data in the stream shall be a copy of the coordinator internal memory.

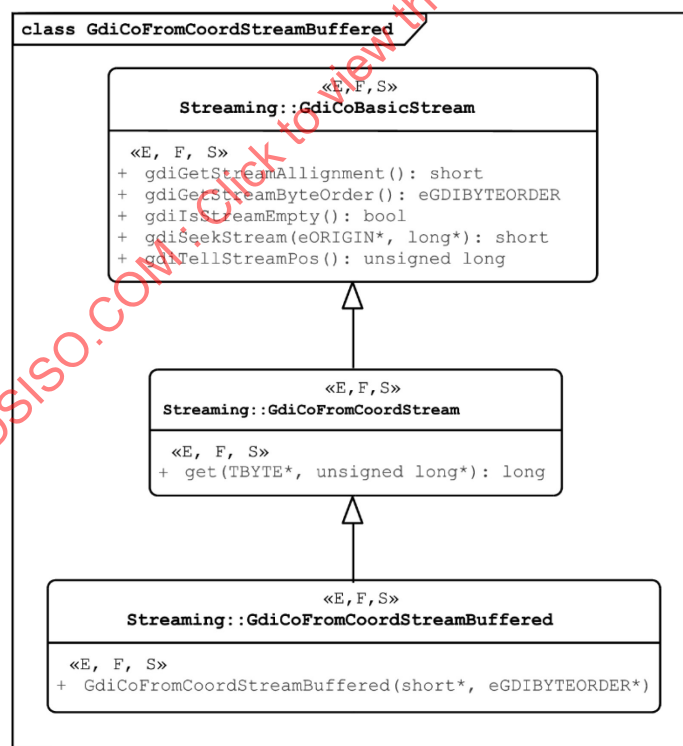


Figure A.15 — Hierarchical diagram of `GdiCoFromCoordStreamBuffered`

**A.2.15.1 GdiCoFromCoordStreamBuffered :: «S,E,F» GdiCoFromCoordStreamBuffered()****Function Call**

```

GdiCoFromCoordStreamBuffered
(
    inout          short nAlignment
    /* in parameter
    defines the requested Alignment of the Application.
    The default Constant is defined in the manufacturer specific Coordinator
    environment. */
    inout          eGDI_BYTEORDER eByteOrder
    /* in parameter
    defines the requested ByteOrder of the Application.
    The default Constant is defined in the manufacturer specific Coordinator
    environment. */
) :

```

**Function Description**

Constructor of the Streamobject, defines Alignment and Byteorder.

The Data in the Stream Object will be updated by the Coordinator, if a communication (gdiRead, gdiExecute) to the Device Driver will be performed.

**A.2.16 «S,E,F» GdiCoFromCoordStreamUnbuffered**

This interface is used to read an unformatted byte sequence from the coordinator. The stream object shall refer to the coordinator internal memory. A modification of the coordinator internal memory shall do a modification of the data in the stream.

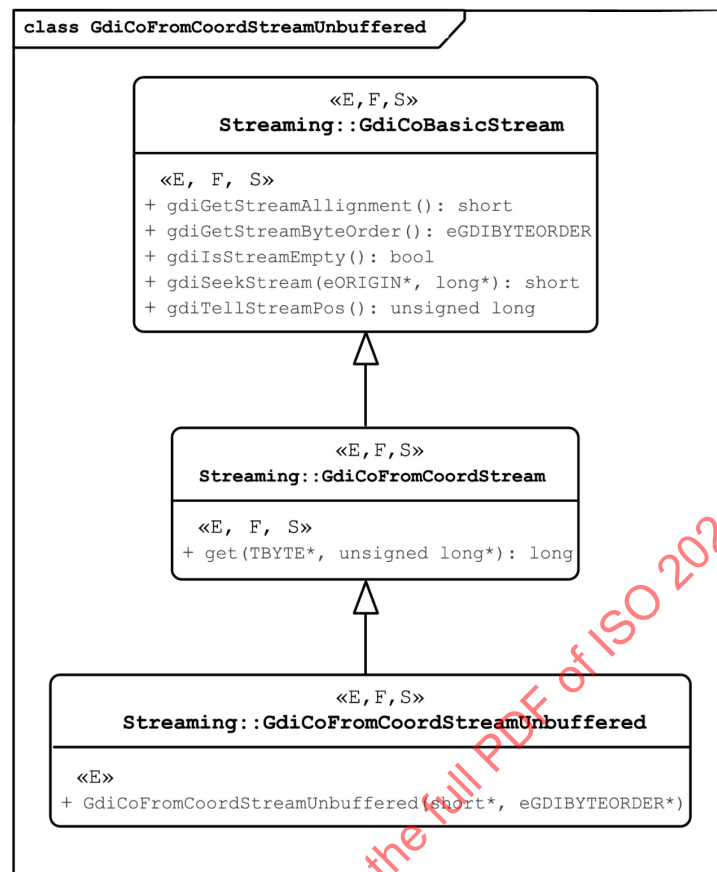


Figure A.16 — Hierarchical diagram of GdiCoFromCoordStreamUnbuffered

### A.2.16.1 GdiCoFromCoordStreamUnbuffered :: «E» GdiCoFromCoordStreamUnbuffered()

#### Function Call

```

GdiCoFromCoordStreamUnbuffered
(
    inout          short nAlignment
    /* in parameter
    defines the requested Alignment of the Application.
    The default Constant is defined in the manufacturer specific Coordinator
    environment. */
    inout          eGDIBYTEORDER eByteOrder
    /* in parameter
    defines the requested ByteOrder of the Application.
    The default Constant is defined in the manufacturer specific Coordinator
    environment. */
) :
  
```

#### Function Description

Constructor of the unbuffered Streamobject, defines Alignment and Byteorder.

The Stream Object is used as unbuffered Out Reference in connection with gdiReadValue or the out Parameter of a gdiExecute.

The Data in the Stream Object will be updated if a communication (gdiRead, gdiExecute) to the Device Driver is performed.

### A.2.17 «S,E,F» GdiCoFuncObjectIterator

This interface contains the service for the application of a FuncObject iterator. It shall present a sequential output of all application function objects which have been set up within a VD.

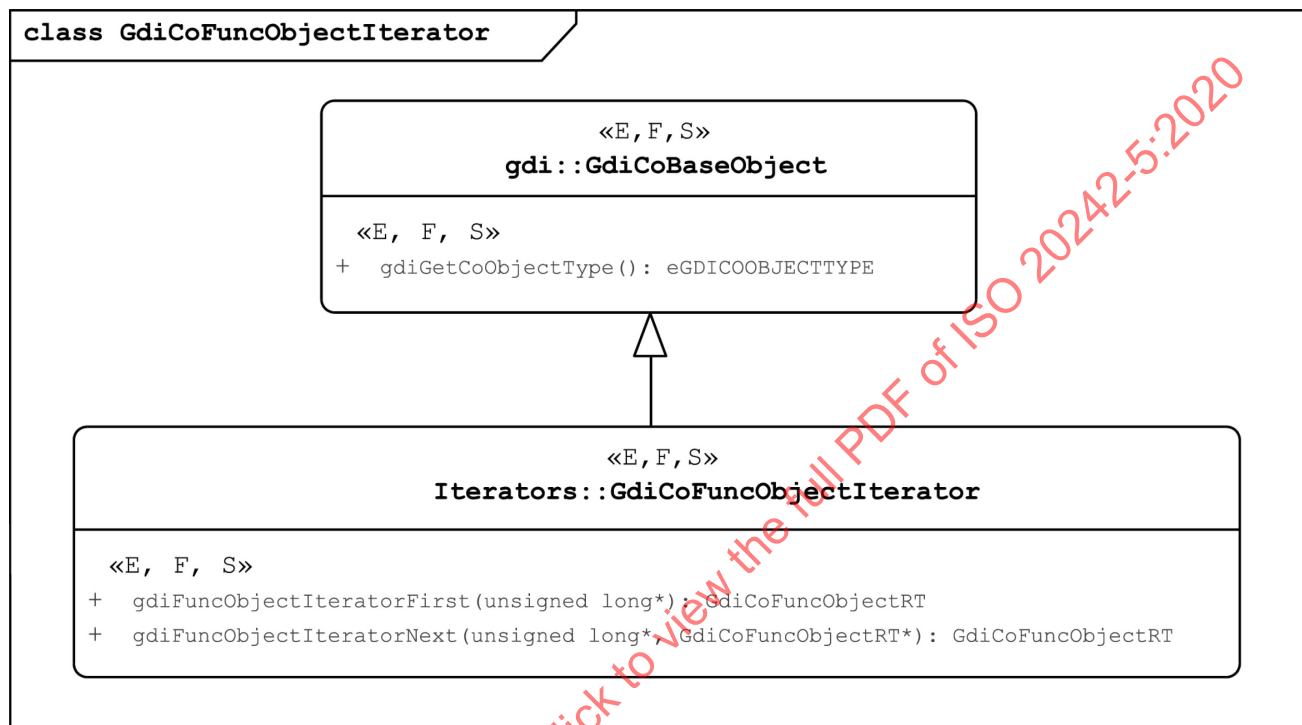


Figure A.17 — Hierarchical diagram of GdiCoFuncObjectIterator

#### A.2.17.1 GdiCoFuncObjectIterator :: «S,E,F» gdiFuncObjectIteratorFirst()

##### Function Call

```

gdiFuncObjectIteratorFirst
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoFuncObjectRT
  
```

##### Function Description

Sets the internal pointer to the first VD within the Workspace.

##### Return Value

Returns a reference to the first GdiCoFuncObject if it exists, otherwise a NIL Reference returns.

**A.2.17.2 GdiCoFuncObjectIterator :: «S,E,F» gdiFuncObjectIteratorNext()****Function Call**

```

gdiFuncObjectIteratorNext
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoFuncObjectRT refLastElement
    /* in parameter
    A reference to a element returned by gdiFuncObjectIteratorFirst or
    gdiFuncObjectIteratorNext. */
) : GdiCoFuncObjectRT

```

**Function Description**

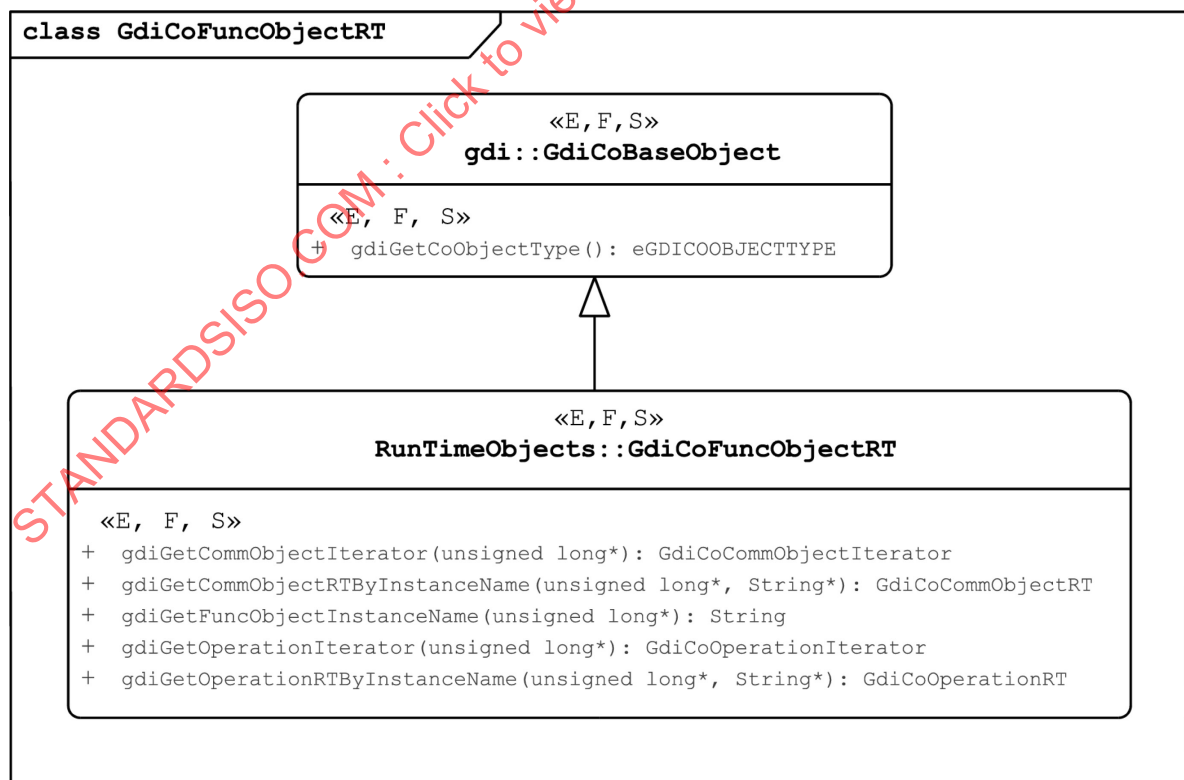
Detects the reference to the current FuncObject (Parameter) within the workspace. Sets the internal pointer to the next FuncObject (if existing) and returns the reference.

**Return Value**

Returns a reference to the next GdiCoFuncObject if it exists, otherwise a NIL Reference returns.

**A.2.18 «S,E,F» GdiCoFuncObjectRT**

This interface contains all services for referencing communication objects and operations of a function object.



**Figure A.18 — Hierarchical diagram of GdiCoFuncObjectRT**

### A.2.18.1 GdiCoFuncObjectRT :: «S,E,F» gdiGetCommObjectIterator()

#### Function Call

```
gdiGetCommObjectIterator
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoCommObjectIterator
```

#### Function Description

It is used for the output of all set up Communication objects of a Function object.

#### Return Value

Returns the reference to GdiCoComObjectIterator if successful.

### A.2.18.2 GdiCoFuncObjectRT :: «S,E,F» gdiGetCommObjectRTByName()

#### Function Call

```
gdiGetCommObjectRTByName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sCommObjectName
    /* in parameter
    Name of expected Instance; defined during parameterization */
) : GdiCoCommObjectRT
```

#### Function Description

Is used for the detecting of a reference to a Communication object instance.

#### Return Value

Returns the reference to GdiCoCommObjectRT if successful.

### A.2.18.3 GdiCoFuncObjectRT :: «S,E,F» gdiGetFuncObjectName()

#### Function Call

```
gdiGetFuncObjectName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String
```

#### Function Description

Returns the instance name of the object defined during parameterization.

Must be unique inside a VD, so the name is different to the name of the corresponding interface defined in the DCD.

If this Object is a dynamic FO, the Instance Name has the following Syntax:

<InterfaceName>\_<InstanceCounter>

InterfaceName: Class Name in DCD

InstanceCounter: Counter is increasing by the number of  
Instances of all dynamic FOs.  
Deleted instances are not considered

#### Return Value

Returns the Instance Name of the FO instance.

### **A.2.18.4 GdiCoFuncObjectRT :: «S,E,F» gdiGetOperationIterator()**

#### Function Call

```
gdiGetOperationIterator
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoOperationIterator
```

#### Function Description

It is used for the output of all available operations of a Function object.

#### Return Value

Returns the reference to the GdiCoOperationIterator if successful.

### **A.2.18.5 GdiCoFuncObjectRT :: «S,E,F» gdiGetOperationRTByInstanceName()**

#### Function Call

```
gdiGetOperationRTByInstanceName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sOperationObjectInstanceName
    /* in parameter
    Name of expected Operation; Name defined in DCD or redefined during
    parameterization. */
) : GdiCoOperationRT
```

#### Function Description

It is used for the detecting of a reference to an executable operation.

#### Return Value

Returns references GdiCoOperation if successful.

### A.2.19 «S,E,F» GdiCoInternalException

This exception class shall be thrown if an error in the coordinator implementation occurs.

The error codes

- eINT\_INTERNAL\_ERROR (coordinator failure, operation system failure) and
- eINT\_VERSION\_NOT\_SUPPORTED (the requested interface version is not supported)

shall be linked to this exception.

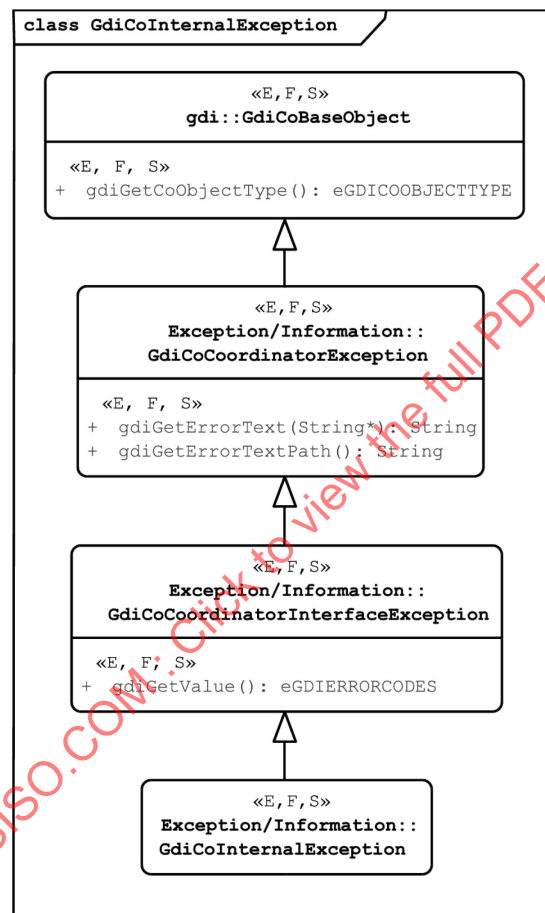


Figure A.19 — Hierarchical diagram of GdiCoInternalException

### A.2.20 «S,E,F» GdiCoManager

Via the services of this interface an application may connect and disconnect. The services of this interface shall be permanently available. In case of an object oriented implementation, this instance shall be created with the start of the coordinator.



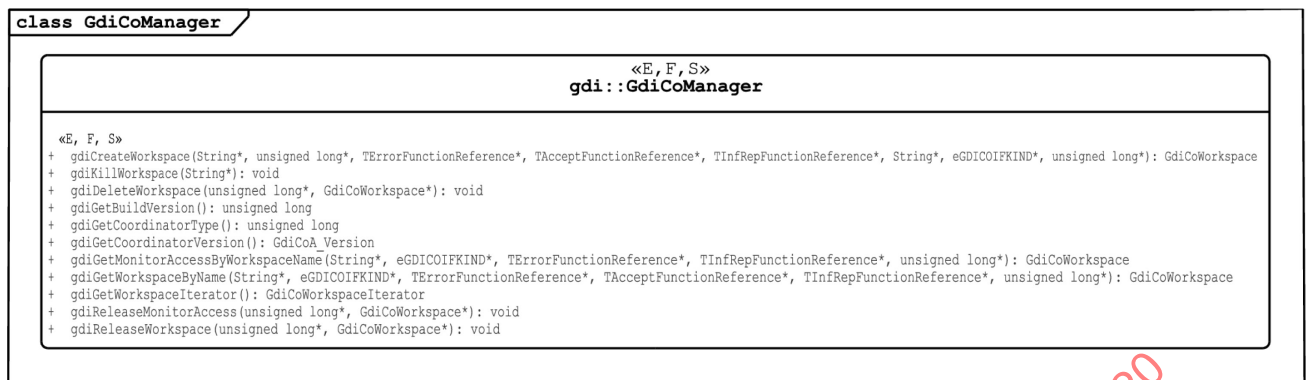


Figure A.20 — Hierarchical diagram of GdiCoManager

### A.2.20.1 GdiCoManager :: «S,E,F» gdiCreateWorkspace()

#### Function Call

```

gdiCreateWorkspace
(
    inout          String sInstanceDescription
    /* in parameter
    Contains parameterization data or the name of a parameterization file
    */
    inout          unsigned long ulTimeOut
    /* in parameter
    Maximum duration time for the execution of a coordinator service in
    milliseconds (the maximum is about 50 days) */
    inout          TErrorFunctionReference refCBException
    /* in parameter
    Reference to a method which handles errors. If a exception mechanism
    will be used (e.g. C++ exception) use NIL reference for this parameter.
    */
    inout          TAcceptFunctionReference refCBAccept
    /* in parameter
    Reference to a call back method which handles the accept mechanism. A
    NIL reference signals that no call back is supported.
    At the coordinator call back to the application, an reference to the
    specific GdiCoCommObjectRT is passed. */
    inout          TInfRepFunctionReference refCBInformationReport
    /* in parameter
    Reference to a call back method which handles the Information Report
    mechanism. A NIL reference signals that no call back is supported.
    At the coordinator call back to the application, an reference to the
    specific GdiCoCommObjectRT is passed. */
    inout          String sWorkspaceName
    /* in parameter
    Instance Name of the new Workspace */
    inout          eGDICOFKIND eRequiredCoordinatorInterface
    /* in parameter
    contains the expected Coordinator Interface Type Smart, Extended, Full
    as bit mask. (see semantics) */
    inout          unsigned long ulAppHnd
    /* out parameter
    Delivers the Identifier of the application. */
) : GdiCoWorkspace
  
```

Function Description

This method shall create a new workspace, containing a group of function instances. These function instances may be members of several VDs. The instance name shall be mandatory and may be defined by the user, independent from the workspace name given in the PID. An empty string as argument shall be refused.

Callback pointers shall be entry points for asynchronous completion or signal entries for events (InformationReport, Accept).

The parameter eRequiredCoordinatorInterface defines the requested coordinator interface type:

0 = eSMART1 = eEXTENDED  
2 = eFULL.

If the expected interface is not supported by the coordinator, the workspace creation shall be refused.

In case of any error during the instance creation process, the reference to GdiCoWorkspace shall be NIL and the workspace shall be deleted immediately.

Return Value

Returns GdiCoWorkspace if successful, otherwise NIL returns.

**A.2.20.2 GdiCoManager :: «S,E,F» gdiDeleteWorkspace()**Function Call

```
gdiDeleteWorkspace
(
    inout                unsigned Long ulAppHand
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoWorkspace refWorkspace
    /* Reference to a existing Workspace */
) : void
```

Function Description

This service shall disconnect an application from the coordinator. With this, all instances within the coordinator and device driver(s) shall be deleted.

The coordinator behaviour is dependent from the required coordinator access at the creation time (gdiCreateWorkspace(), gdiGetWorkspaceByName()).

In case of a required Smart or Extended access, the coordinator shall delete all Communication Objects, Function Objects and Virtual Devices. Because errors will be ignored, this shall be simply realized with the Abort functionality of the Device Driver API. Also the device drivers themselves shall be decoupled.

In case of a required Full Access Interface all activities for destroying of Communication Objects, Function Objects, VDs and Device Driver shall be directly controlled by the Application via the Coordinator Full Interface. If any error occurs, the application is responsible for the following steps. Only if the workspace is empty, gdiDeleteWorkspace() shall be successful in case of a full access. This is required for Description- and Run-Time side.

Return Value

None

**A.2.20.3 GdiCoManager :: «S,E,F» gdiGetBuildVersion()****Function Call**

```
gdiGetBuildVersion
(
    ) : unsigned long
```

**Function Description**

Returns the Build Version of the coordinator.

**Return Value**

Returns the Build Version.

**A.2.20.4 GdiCoManager :: «S,E,F» gdiGetCoordinatorType()****Function Call**

```
gdiGetCoordinatorType
(
    ) : unsigned long
```

**Function Description**

Returns the supported Interface access types as enumeration.  
Interface Types:

0 = eSMART  
1 = eEXTENDED  
2 = eFULL.

**Return Value**

Returns Supported Interface types as Bit combination.

**A.2.20.5 GdiCoManager :: «S,E,F» gdiGetCoordinatorVersion()****Function Call**

```
gdiGetCoordinatorVersion
(
    ) : GdiCoA_Version
```

**Function Description**

Returns the GDI Version object. The Interface contains the highest supported GDI Version.

**Return Value**

Returns the reference to the GDIVersion Interface.

**A.2.20.6 GdiCoManager :: «S,E,F» gdiGetMonitorAccessByWorkspaceName()**

## Function Call

```
gdiGetMonitorAccessByWorkspaceName
(
    inout          String sWorkspaceName
    /* in parameter
    contains the name of the expected workspace. */
    inout          eGDICOIFKIND eRequiredCoordinatorInterface
    /* in parameter
    contains the expected Coordinator Interface Type Smart, Extended, Full
    as bit mask. (see semantics) */
    inout          TErrorFunctionReference refCBException
    /* in parameter
    Reference to a method which handles errors. If a exception mechanism
    will be used (e.g. C++ exception) use NIL reference for this parameter.
    */
    inout          TInfRepFunctionReference refCBInformationReport
    /* in parameter
    Reference to a call back method which handles the Information Report
    mechanism. A NIL reference signals that no call back is supported.
    At the coordinator call back to the application, an reference to the
    specific GdiCoCommObjectRT is passed. */
    inout          unsigned long ulAppHnd
    /* out parameter
    Delivers the Identifier of the application. */
) : GdiCoWorkspace
```

## Function Description

Returns the reference of the workspace, corresponding to the given Workspace name.

The parameter eRequiredCoordinatorInterface describes the expected Coordinator Interface type by the application.

0 = eSMART  
1 = eEXTENDED  
2 = eFULL

If the expected Interface is not supported by the coordinator, the workspace cannot be returned.

If the informationreport flag is enabled at the GdiCoAttributRT and the device driver does not support information reports for this attribute, the coordinator will send a Information Report to the monitoring client any time the attribute is changed by another client.

If the device driver sends a Information Report to this attribute, the Information will be delivered to all monitoring clients, enabling the Information Report flag.

## Return Value

Returns GdiCoWorkspace for monitor access if successful, otherwise NIL.

## A.2.20.7 GdiCoManager :: «S,E,F» gdiGetWorkspaceByName()

**Function Call**

```

gdiGetWorkspaceByName
(
    inout                String sWorkspaceName
    /* in parameter
    contains the name of the expected workspace. */
    inout                eGDICOIFKIND eRequiredCoordinatorInterface
    /* in parameter
    contains the expected Coordinator Interface Type (Smart, Extended, Full
    as bit mask. (see semantics) */
    inout                TErrorFunctionReference refCBException
    /* in parameter
    Reference to a method which handles errors. If a exception mechanism
    will be used (e.g. C++ exception) use NIL reference for this parameter.
    */
    inout                TAcceptFunctionReference refCBAccept
    /* in parameter
    Reference to a call back method which handles the accept mechanism. A
    NIL reference signals that no call back is supported.
    At the coordinator call back to the application, an reference to the
    specific GdiCoCommObjectRT is passed. */
    inout                TInfRepFunctionReference refCBInformationReport
    /* in parameter
    Reference to a call back method which handles the Information Report
    mechanism. A NIL reference signals that no call back is supported.
    At the coordinator call back to the application, an reference to the
    specific GdiCoCommObjectRT is passed. */
    inout                unsigned long ulAppHnd
    /* out parameter
    Delivers the Identifier of the application. */
) : GdiCoWorkspace

```

**Function Description**

Returns the reference of the workspace, corresponding to the given Workspace name.

The parameter eRequiredCoordinatorInterface describes the expected Coordinator Interface type by the application.

0 = eSMART

1 = eEXTENDED

2 = eFULL.

If the expected Interface is not supported by the coordinator, the workspace cannot be returned.

In case the parameter eRequiredCoordinatorInterface is eSMART or eEXTENDED all VDs are transformed to the state working.

In case the parameter eRequiredCoordinatorInterface is eFULL all VDs remain in the state preperation.

**Return Value**

Returns GdiCoWorkspace if it exists, otherwise NIL.

#### A.2.20.8 GdiCoManager :: «S,E,F» gdiGetWorkspaceIterator()

##### Function Call

```
gdiGetWorkspaceIterator  
(  
    ) : GdiCoWorkspaceIterator
```

##### Function Description

Service for the output of all existing Workspace. It is used by Diagnosis Tools.

The parameter eRequiredCoordinatorInterface describes the expected Coordinator Interface type by the application.

##### Return Value

Returns GdiCoWorkspaceIterator (Handle, Pointer, Class) if successful, otherwise 0.

#### A.2.20.9 GdiCoManager :: «S,E,F» gdiKillWorkspace()

##### Function Call

```
gdiKillWorkspace  
(  
    inout          String sWorkspaceName  
    /* in parameter  
    Name of the Workspace to kill. */  
    ) : void
```

##### Function Description

Kills the workspace with the given workspace name.

##### Return Value

None

#### A.2.20.10 GdiCoManager :: «S,E,F» gdiReleaseMonitorAccess()

##### Function Call

```
gdiReleaseMonitorAccess  
(  
    inout          unsigned long ulAppHnd  
    /* in parameter  
    Identifier of the application returned by gdiCreateWorkspace(...) or by  
    gdiGetWorkspaceByName() */  
    inout          GdiCoWorkspace refWorkspace  
    /* in parameter  
    reference to the workspace. */  
    ) : void
```

##### Function Description

This method shall release a workspace without closing it. The registered application shall be invalid.

##### Return Value

None

**A.2.20.11 GdiCoManager :: «S,E,F» gdiReleaseWorkspace()****Function Call**

```

gdiReleaseWorkspace
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoWorkspace refWorkspace
    /* in parameter
    reference to the workspace. */
) : void

```

**Function Description**

This method shall release a workspace without closing it. The access of other applications to the workspace is now possible.

The coordinator shall initiate a state change to Preparation to all VDs of the workspace independent from their initial state, and then release the workspace handle. If a VD does not allow the state transition because of an error, the release of the workspace shall be rejected.

**Return Value**

None

**A.2.21 «S,E,F» GdiCoObjectAdministrationException**

This exception class is thrown if an error in the coordinator internal Object administration occurs, because of a handling error of the Coordinator API:

The error codes

- eOAD\_OUT\_OF\_RANGE (Sequence, Array access)
- eOAD\_OPEN\_SERVICE (GDI Object in use)
- eOAD\_TYPE\_NOT\_ALLOWED (if the new element is a sequence, and part of a Union)
- eOAD\_OBJECT\_ACCESS (Data Source / Sink not available)

are connected with this exception.

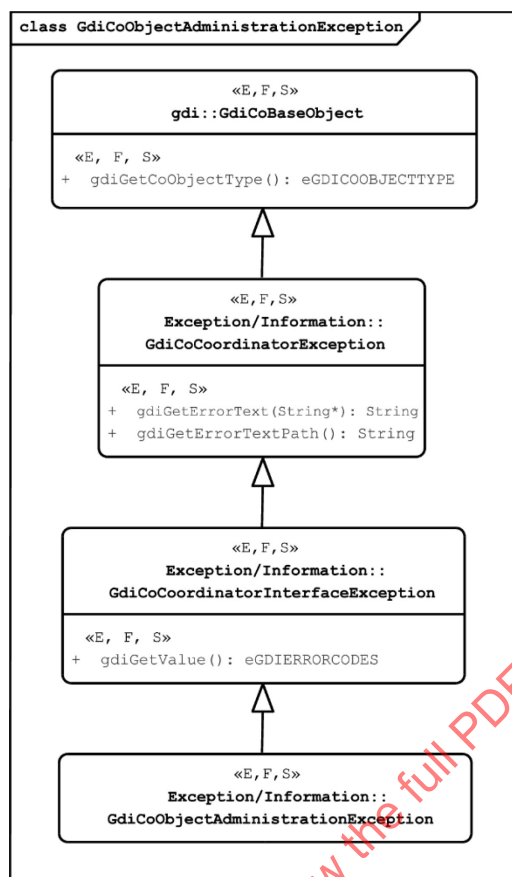


Figure A.21 — Hierarchical diagram of GdiCoObjectAdministrationException

#### A.2.22 «S,E,F» GdiCoOperationIterator

This interface contains the service for the application of an operation iterator, sequential output of all operations which have been set up within Function object.





Figure A.22 — Hierarchical diagram of GdiCoOperationIterator

#### A.2.22.1 GdiCoOperationIterator :: «S,E,F» gdiOpIteratorFirst()

##### Function Call

```

gdiOpIteratorFirst
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoOperationRT
  
```

##### Function Description

Sets the internal pointer to the first Operation within the FO Interface.

##### Return Value

Returns a reference to the first GdiCoOperation if it exists, otherwise a NIL reference returns.

#### A.2.22.2 GdiCoOperationIterator :: «S,E,F» gdiOpIteratorNext()

##### Function Call

```

gdiOpIteratorNext
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
  
```

```

inout          GdiCoOperationRT refLastElement
/* in parameter
A reference to a element returned by gdiOPIteratorFirst or
gdiOPIteratorNext. */
) : GdiCoOperationRT

```

### Function Description

Detects the reference to the current OP (Parameter) within the function object. Sets the internal pointer to the next OP (if existing) and returns the reference.

### Return Value

Returns the reference to a coordinator operation (administration unit within the coordinator).

## A.2.23 «S,E,F» GdiCoOperationRT

This interface contains services for the use of operations.

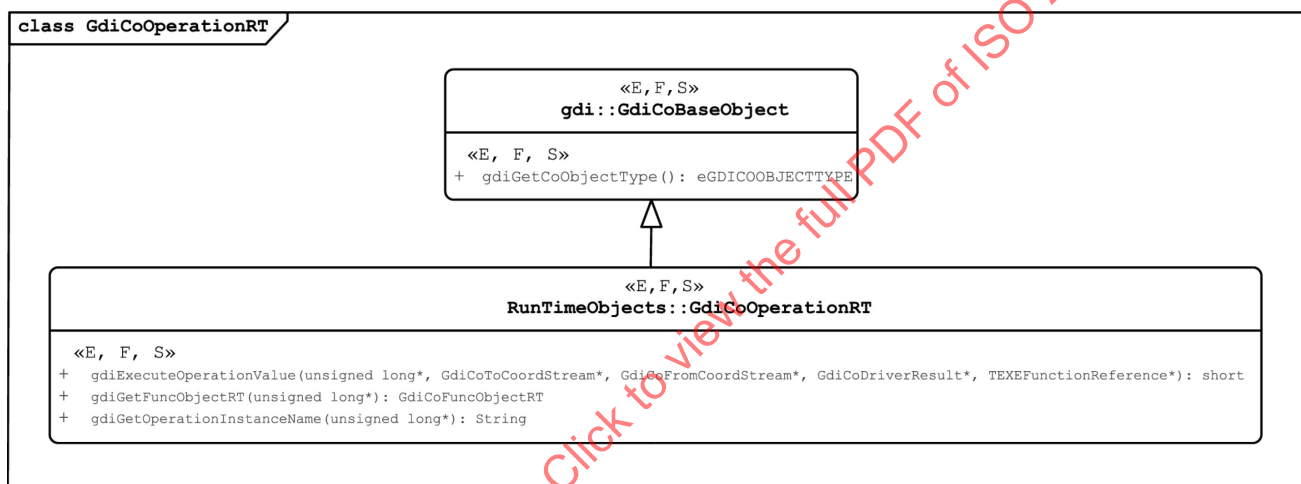


Figure A.23 — Hierarchical diagram of GdiCoOperationRT

### A.2.23.1 GdiCoOperationRT :: «S,E,F» gdiExecuteOperationValue()

#### Function Call

```

gdiExecuteOperationValue
(
inout          unsigned long ulAppHnd
/* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          GdiCoToCoordStream refInValue
/* in parameter
Reference to a stream object, the type is identical to the type of the
operation in parameter object. */
inout          GdiCoFromCoordStream refOutValue
/* out parameter
Reference to a stream object, the type is identical to the type of the
out paramter object. */
inout          GdiCoDriverResult refDestInfo

```

```

/* out parameter
contains the reference to the Information Object, warnings and
informations of the Driver will be stored in it. */
inout          TEXEFunctionReference refCBCompleteExec
/* in parameter
Reference to a call back method which handles the execution mechanism.
A NIL reference signals a synchronous call. */
) : short

```

### Function Description

This service takes over the data into the In area of an operation, starts the operation and returns the return values.

### Return Value

Describes the executed communication by the GDI driver:

0 = synchronous return;  
 1 = asynchronous return;  
 less than 0 = error.

## **A.2.23.2 GdiCoOperationRT :: «S,E,F» gdiGetFuncObjectRT()**

### Function Call

```

gdiGetFuncObjectRT
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
) : GdiCoFuncObjectRT

```

### Function Description

Creates reference to the existing Function Object.  
 The referenced Function Object contains this Operation Object.

### Return Value

Returns a reference to the GdiCoFuncObjectRT if succesful.

## **A.2.23.3 GdiCoOperationRT :: «S,E,F» gdiGetOperationInstanceName()**

### Function Call

```

gdiGetOperationInstanceName
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
) : String

```

### Function Description

Returns the instance name of the object defined during parameterization.

The Instance of an operation is implicit (no API activity is necessary). The operation name of a interface and operation name of a FuncObject can be identical.

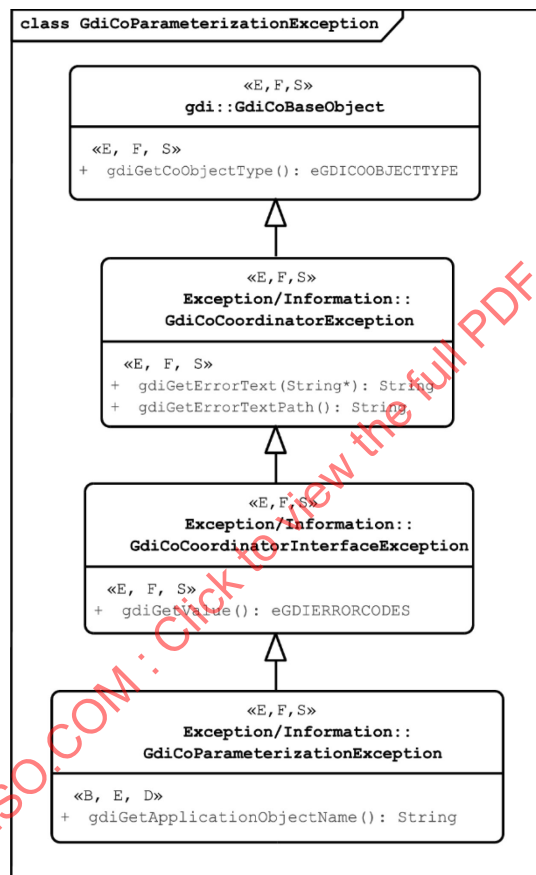
### Return Value

Returns Instance name as string.

## **A.2.24 «S,E,F» GdiCoParameterizationException**

This exception class is thrown if the PID File is incorrect or invalid.

The error code ePAR\_INCORRECT\_PARAMETERIZATION is connected with this exception.



**Figure A.24 — Hierarchical diagram of GdiCoParameterizationException**

### **A.2.24.1 GdiCoParameterizationException :: «B,E,D» gdiGetApplicationObjectName()**

#### Function Call

```

gdiGetApplicationObjectName
(
) : String
  
```

#### Function Description

Returns the name of the abstract data source/sink that could not be established. The PID process fails on this location.

Return Value

Returns the name of the abstract data source/sink that could not be established. (FO,CO,OP).

**A.2.25 «S,E,F» GdiCoParameterRT**

No additional service compared to the Communication object available. However, it is Read only within the state working of the respective VD.

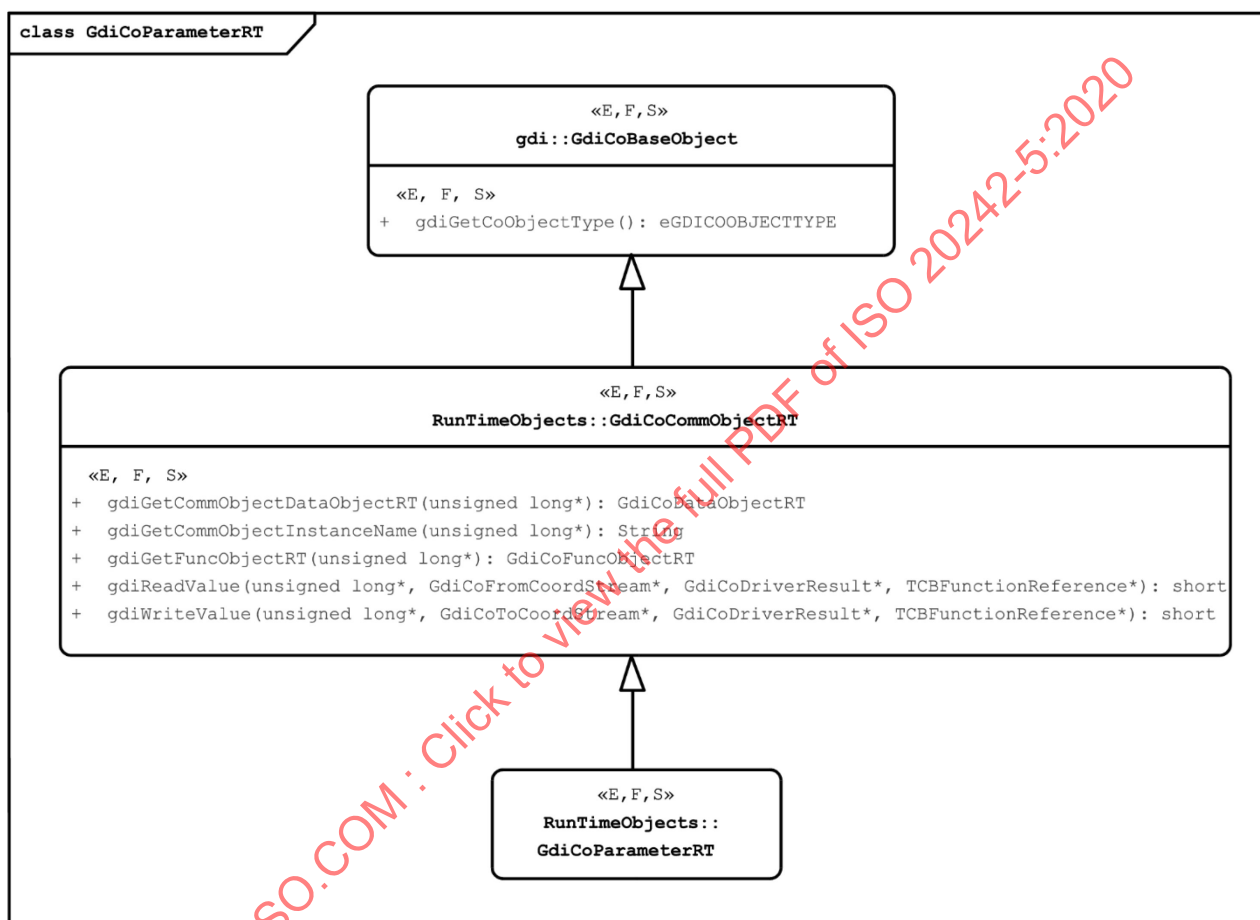


Figure A.25 — Hierarchical diagram of GdiCoParameterRT

**A.2.26 «S,E,F» GdiCoToCoordStream**

This interface is used to write an unformatted byte sequence from the coordinator.



Figure A.26 — Hierarchical diagram of GdiCoToCoordStream

## A.2.26.1 GdiCoToCoordStream :: «S,E,F» put()

Function Call

```

put
(
    inout          TBYTE refSource
    /* in parameter
    contains a reference to the elements in the application which to
    write to the stream. */
    inout          unsigned long ulnByteCount
    /* in parameter
    contains the requested number of bytes to be read. */
) : long
  
```

Function Description

Puts data from the application to the stream.

Return Value

Returns the count of Elements, which was put to the stream.

A negative number describes an error.

### A.2.27 «S,E,F» GdiCoToCoordStreamBuffered

This interface is used to write an unformatted byte sequence to the coordinator.

The client is responsible for creating and deleting this object.

Updating the Data in the Stream Object will not affect a direct modification of the Coordinator internal memory.

The Coordinator Memory will be updated by the next gdiWrite, gdiExecute, gdiSetDataValue (in extended access) action.

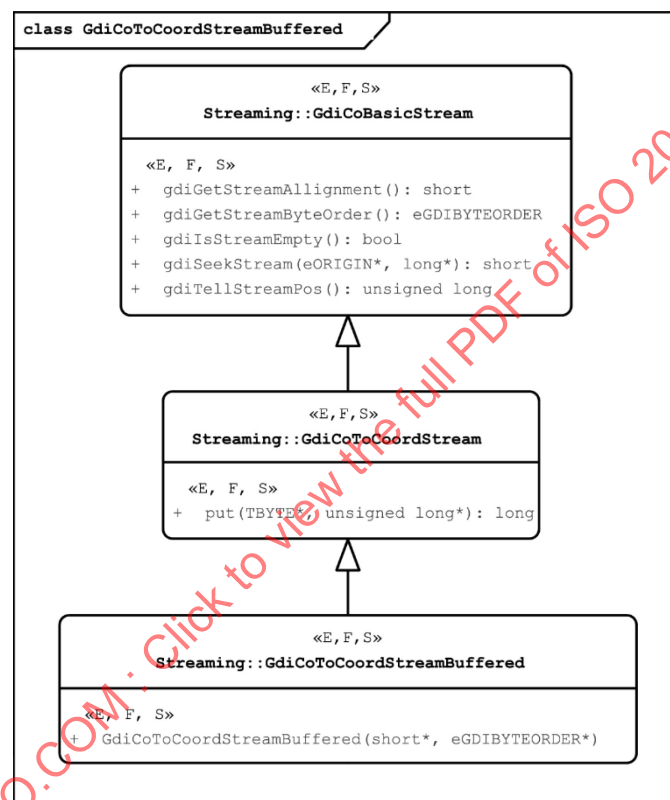


Figure A.27 — Hierarchical diagram of GdiCoToCoordStreamBuffered

### A.2.27.1 GdiCoToCoordStreamBuffered :: «S,E,F» GdiCoToCoordStreamBuffered()

#### Function Call

```

GdiCoToCoordStreamBuffered
(
    inout          short nAlignment
    /* in parameter
    defines the requested Alignment of the Application.
    The default Constant is defined in the manufacturer specific Coordinator
    environment. */
    inout          eGDIBYTEORDER eByteOrder
    /* in parameter
    defines the requested ByteOrder of the Application.
  
```

```
The default Constant is defined in the manufacturer specific Coordinator
environment. */
) :
```

Function Description

Constructor of the unbuffered Streamobject. Defines Allignment and Byteorder.

Updating the Data in the Stream Object will not affect a direct modification of the Coordinator internal memory.

The Coordinator Memory will be updated by the next gdiWrite, gdiExecute, gdiSetDataValue (in extended access) action.

A.2.28 «S,E,F» GdiCoToCoordStreamUnbuffered

This interface is used to write an unformatted byte sequence to the coordinator.

The client is responsible for creating and deleting this object.

Updating the Data in the Stream Object will affect a direct modification of the Coordinator internal memory.

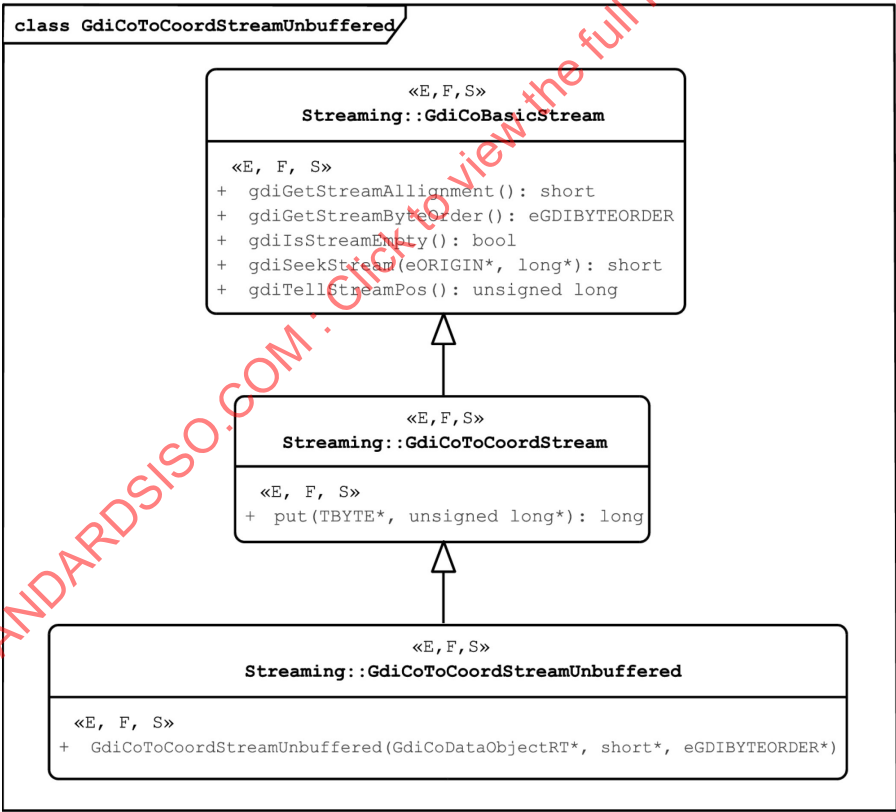


Figure A.28 — Hierarchical diagram of GdiCoToCoordStreamUnbuffered



**A.2.28.1 GdiCoToCoordStreamUnbuffered :: «S,E,F» GdiCoToCoordStreamUnbuffered()****Function Call**

```

GdiCoToCoordStreamUnbuffered
(
    inout                GdiCoDataObjectRT refGdiDestination
    /* in parameter
    defines the Destination Data Area in the Coordinator.
    This Parameter must not be NIL. The Stream Object is used as In Reference
    in connection with gdiWriteValue or the in Parameter of a gdiExecute.
    */
    inout                short nAlignment
    /* in parameter
    defines the requested Alignment of the Application.
    The default Constant is defined in the manufacturer specific Coordinator
    environment. */
    inout                eGDIBYTEORDER eByteOrder
    /* in parameter
    defines the requested ByteOrder of the Application.
    The default Constant is defined in the manufacturer specific Coordinator
    environment. */
) :

```

**Function Description**

Constructor of the unbuffered Streamobject. Defines Alignment and Byteorder and the Destination Data Area in the Coordinator.

This Parameter shall not be NIL.

The Stream Object is used as unbuffered In Reference in connection with gdiWriteValue or the in Parameter of a gdiExecute.

Updating the Data in the Stream Object will affect a direct modification of the Coordinator internal memory.

**A.2.29 «S,E,F» GdiCoWorkspace**

This interface contains services for the referencing of existing Function objects within application areas. A Function object may be detected via its name or via iteration of all existing Function objects of the Workspace.

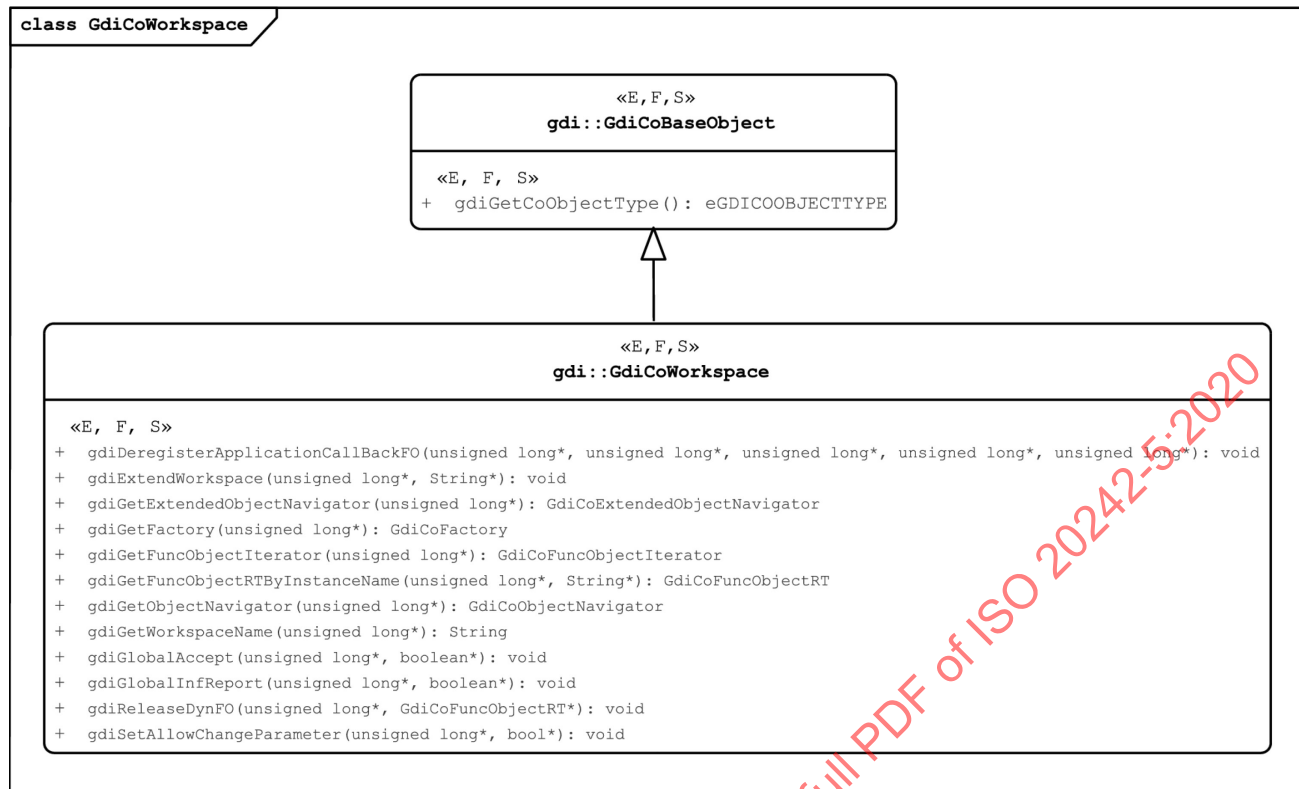


Figure A.29 — Hierarchical diagram of GdiCoWorkspace

### A.2.29.1 GdiCoWorkspace :: «S,E,F» gdiDeregisterApplicationCallBackFO()

#### Function Call

```

gdiDeregisterApplicationCallBackFO
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout          unsigned long ulFuncMem
    /* in parameter
    The FO value has to be specified within the DCD of the driver. */
    inout          unsigned long refApplicationFO
    /* in parameter
    Reference or handle to the application FO. This reference is given by
    the Application. */
    inout          unsigned long ulFuMemId
    /* in parameter
    Id of the Application Call Back Communication object according to DCD.
    */
    inout          unsigned long refApplicationCO
    /* in parameter
    Reference or Handle of the CO which is used for a Information Report.
    */
) : void
  
```

Function Description

Deregisters the indicated Application Function Object with all its attributes.

Return Value

None

**A.2.29.2 GdiCoWorkspace :: «S,E,F» gdiExtendWorkspace()**Function Call

```
gdiExtendWorkspace
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sInstanceDescription
    /* in parameter
    Contains parameterization data or the name of a parameterization file.
    */
) : void
```

Function Description

The instances given in the function description are used to extend and update the existing workspace. Already created objects may be changed, but they are never deleted.

If any error concerning accordance is registered for the given PID file or an error occurs while running the PID file, the workspace will not be extended. All affected VDs still remain in state preparation. In the first case the method gdiExtendWorkspace throws the exception eINT\_PID\_FILE\_MISMATCH (GdiCoInternalException) and in the second case the Exception ePAR\_INCORRECT\_PARAMETERIZATION (GdiCoParameterizationException).

In case of non-accordance of the keys, the PID File is processed and the existing Workspace is extended. In case of PID error the behaviour is as above mentioned, that means the process is aborted and the existing Workspace is still valid, but all affected VDs still remain in state preparation.

Return Value

None

**A.2.29.3 GdiCoWorkspace :: «S,E,F» gdiGetExtendedObjectNavigator()**Function Call

```
gdiGetExtendedObjectNavigator
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoExtendedObjectNavigator
```

Function Description

Returns the reference to the GdiCoExtendedObjectNavigator if the given Coordinator supports the Extended access functionality.

Return Value

Returns reference to the GdiCoExtendedObjectNavigator if it exists.

**A.2.29.4 GdiCoWorkspace :: «S,E,F» gdiGetFactory()**Function Call

```
gdiGetFactory
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoFactory
```

Function Description

Returns the reference to the GdiCoFactory if the given Coordinator support the Full access functionality.

Return Value

Returns reference to the GdiCoDescriptionFactory if it exists.

**A.2.29.5 GdiCoWorkspace :: «S,E,F» gdiGetFuncObjectIterator()**Function Call

```
gdiGetFuncObjectIterator
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoFuncObjectIterator
```

Function Description

Creates reference to an iterator, it is used for the output of all set up Function objects of the Workspace.

Return Value

Returns reference to GdiFuncObjectIterator if successful.

**A.2.29.6 GdiCoWorkspace :: «S,E,F» gdiGetFuncObjectRTByInstanceName()**Function Call

```
gdiGetFuncObjectRTByInstanceName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sFOInstanceName
    /* in parameter
    Identifier of expected Instance */
) : GdiCoFuncObjectRT
```

Function Description

Creates reference to an existing Function object.

Return Value

Returns reference to a GdiCoFuncObjectRT if it exists.

**A.2.29.7 GdiCoWorkspace :: «S,E,F» gdiGetObjectNavigator()**Function Call

```
gdiGetObjectNavigator
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...)
    or by gdiGetWorkspaceByName() */
) : GdiCoObjectNavigator
```

Function Description

Returns the reference to the GdiCoObjectNavigator if the given Coordinator supports the Full access functionality.

Return Value

Returns reference to the GdiCoObjectNavigator if it exists.

**A.2.29.8 GdiCoWorkspace :: «S,E,F» gdiGetWorkspaceName()**Function Call

```
gdiGetWorkspaceName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String
```

Function Description

Returns the identifier of the workspace.

Return Value

Returns Workspacename as a string.

**A.2.29.9 GdiCoWorkspace :: «S,E,F» gdiGlobalAccept()**Function Call

```
gdiGlobalAccept
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          boolean bEnableControl
/* true = enable Accept Callbacks of this Communication Objects wish was
local enabled.
false = disable all Accept Callbacks */
) : void

```

### Function Description

Enables/disables the Accept Callbacks for all Communication Objects in all VDs inside the Workspace.

### Return Value

None

## **A.2.29.10 GdiCoWorkspace :: «S,E,F» gdiGlobalInfReport()**

### Function Call

```

gdiGlobalInfReport
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          boolean bEnableControl
  /* true = enable Information Reports of this Communication Objects wish
  was local enabled.
  false = disable all Information Reports */
) : void

```

### Function Description

Enables/disables the Information Reports Callbacks for all Communication Objects in all VDs inside the Workspace.

### Return Value

None

## **A.2.29.11 GdiCoWorkspace :: «S,E,F» gdiReleaseDynFO()**

### Function Call

```

gdiReleaseDynFO
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          GdiCoFuncObjectRT refFuncObject
  /* in parameter
  Reference to a Function object within the VD */
) : void

```

### Function Description

In case the application does not need a dynamic Function Object anymore, this Operation releases the Function Object in the Coordinator.

The Coordinator deletes the corresponding Function Object in the Device Driver with all its Communication objects without any transitions. If any activities to the operations, attributes or parameters of the Function Object are in progress, an Exception will be generated.

In case of a repeated access from application an Error will be generated.

#### Return Value

None

### **A.2.29.12 GdiCoWorkspace :: «S,E,F» gdiSetAllowChangeParameter()**

#### Function Call

```
gdiSetAllowChangeParameter
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                bool bEnable
    /* in parameter
    controls the ability of the application to modify parameter after the
    instantiation phase. */
) : void
```

#### Function Description

Announces that the Application is going to change Parameters (Communication Object) after the instantiation phase.

If bEnable = TRUE the application is allowed to write on parameters. The affected VDs will be transferred to state "Revise" during the write Process on the parameter at least.

If bEnable = FALSE all affected VDs which are in state "Revise" will be transferred to state "Working". If the transition is impossible, a GdiCoDriverException will be thrown. The Parameter cannot be changed anymore.

#### Return Value

None

### **A.2.30 «S,E,F» GdiCoWorkspaceInfo**

This interface contains information about the workspace. Each wokspace is connected to a GdiCoWorkspaceInfo object. The Workspace Information can be determined via the GdiCoWorkspaceInfoIterator.

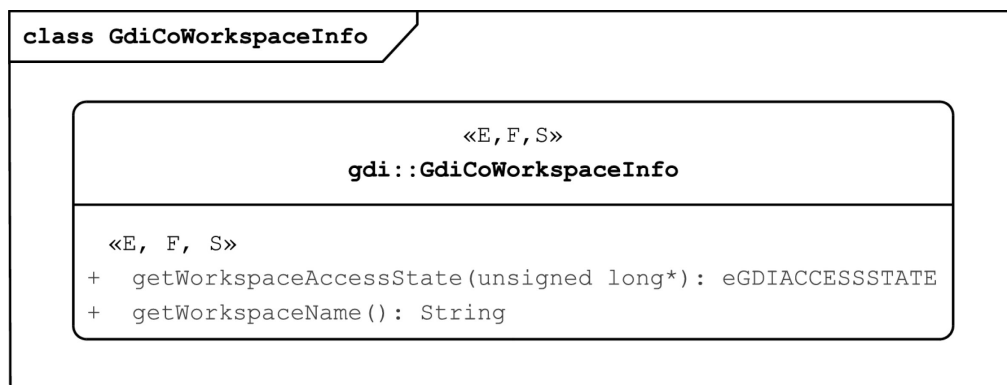


Figure A.30 — Hierarchical diagram of GdiCoWorkspaceInfo

### A.2.30.1 GdiCoWorkspaceInfo :: «S,E,F» getWorkspaceAccessState()

#### Function Call

```

getWorkspaceAccessState
(
    inout                unsigned long ulSec
    /* out parameter
       contains the elapsed time of the returned workspace state. */
) : eGDIACCESSSTATE
  
```

#### Function Description

This method returns the current access state of the workspace.

NOTE The access state is only a snapshot. It is not guaranteed that the Workspace is accessible in the next seconds, because another application gets the access to the workspace.

#### Return Value

Returns the current access state of the workspace:

eNOT\_USED, if workspace is available;  
eUSED, if workspace is in use by another client.

### A.2.30.2 GdiCoWorkspaceInfo :: «S,E,F» getWorkspaceName()

#### Function Call

```

getWorkspaceName
(
) : String
  
```

#### Function Description

This method returns the Instance Name of the Workspace.

#### Return Value

Returns the name of the workspace instance.



### A.2.31 «S,E,F» GdiCoWorkspaceIterator

This interface contains the service for the application of a Workspace iterator, sequential output of all Workspace which have been set up within a coordinator. Please note, by means of this service, a second application may gain sovereignty over Function objects within a coordinator, without letting the owner (application) of the Workspace see this. This service should be used by diagnosis applications only.

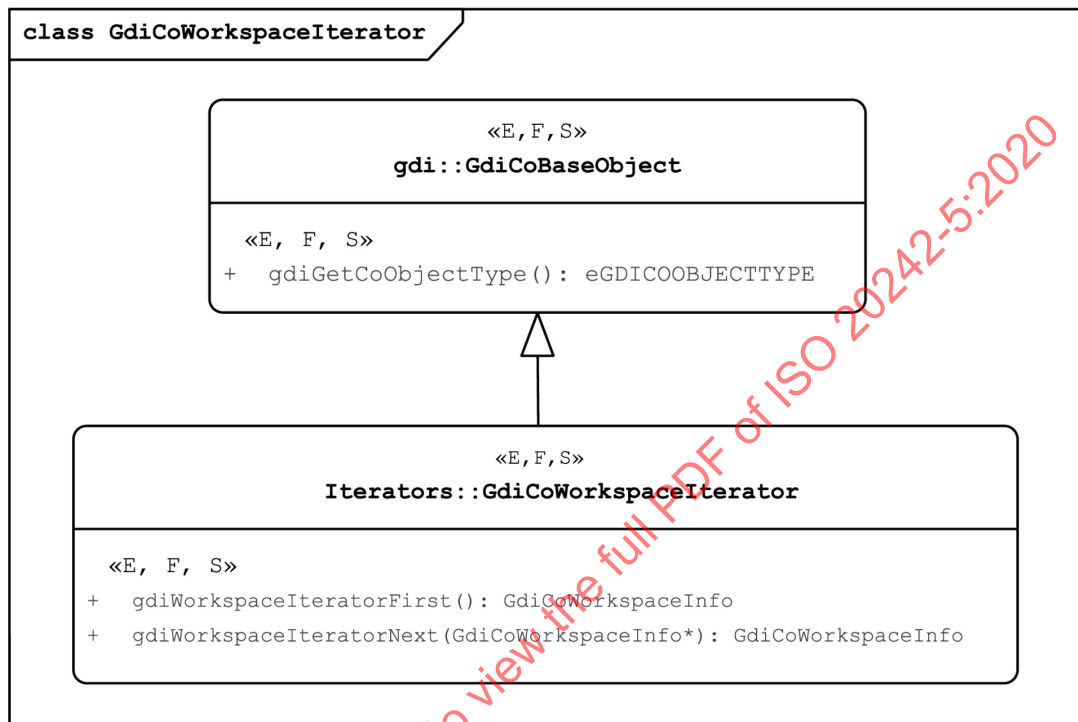


Figure A.31 — Hierarchical diagram of GdiCoWorkspaceIterator

#### A.2.31.1 GdiCoWorkspaceIterator :: «S,E,F» gdiWorkspaceIteratorFirst()

##### Function Call

```

gdiWorkspaceIteratorFirst
(
): GdiCoWorkspaceInfo
  
```

##### Function Description

Sets the internal pointer to the first Workspace within the Coordinator.

##### Return Value

Returns a reference to the Iterator which points to the first GdiCoWorkspaceInfo if it exists, otherwise a NIL Reference returns.

#### A.2.31.2 GdiCoWorkspaceIterator :: «S,E,F» gdiWorkspaceIteratorNext()

##### Function Call

```

gdiWorkspaceIteratorNext
(
  
```

```
inout          GdiCoWorkspaceInfo refLastElement
/* in parameter
A reference to a element returned by gdiWorkspaceIteratorFirst or
gdiWorkspaceIteratorNext. */
) : GdiCoWorkspaceInfo
```

#### Function Description

Detects the reference to the current Workspace (Parameter) within the Coordinator. Sets the internal pointer to the next Workspace (if existing) and returns the reference.

#### Return Value

Returns a reference to the GdiCoWorkspaceInfo which points to the next GdiCoWorkspace if it exists, otherwise a NIL Reference returns.

STANDARDSISO.COM : Click to view the full PDF of ISO 2042-5:2020

## **Annex B**

(normative)

### **Programmer's reference guide — Extended Access Interface**

#### **B.1 General Information**

Single interfaces (respectively classes and their methods and attributes) of the Extended Access Interface, marked with «E», also shall belong to the Full Access Interface, marked with «F». Therefore all single interfaces are marked with «E,F». All single interfaces of the Smart Access Interface described in Annex A shall be contained in Extended Access Interface and are not repeated in this annex. The following detailed description is completed by Figures B.1 to B.17 for an overview on classes and their methods

#### **B.2 Detailed description of Extended Access Interface**

##### **B.2.1 «E,F» GdiCoArrayRT**

This interface contains services for the handling of the type Array.



Figure B.1 — Hierarchical diagram of GdiCoArrayRT

### B.2.1.1 GdiCoArrayRT :: «E,F» gdiGetNumberOfElements()

#### Function Call

```

gdiGetNumberOfElements
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : unsigned long
  
```

#### Function Description

It is used for the detection of the number of elements of an Array.

Return Value

Returns the number of elements of an Array

**B.2.1.2 GdiCoArrayRT :: «E,F» gdiGetSubValueObject()**Function Call

```
gdiGetSubValueObject
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                unsigned long ulIndex
    /* in parameter
    Index value of the data element */
) : GdiCoDataObjectRT
```

Function Description

A sub value object will be created by this activity.

Return Value

Returns a reference to a sub value object of the array.

**B.2.2 «E,F» GdiCoBooleanRT**

This interface contains services for the handling of data of the type boolean.

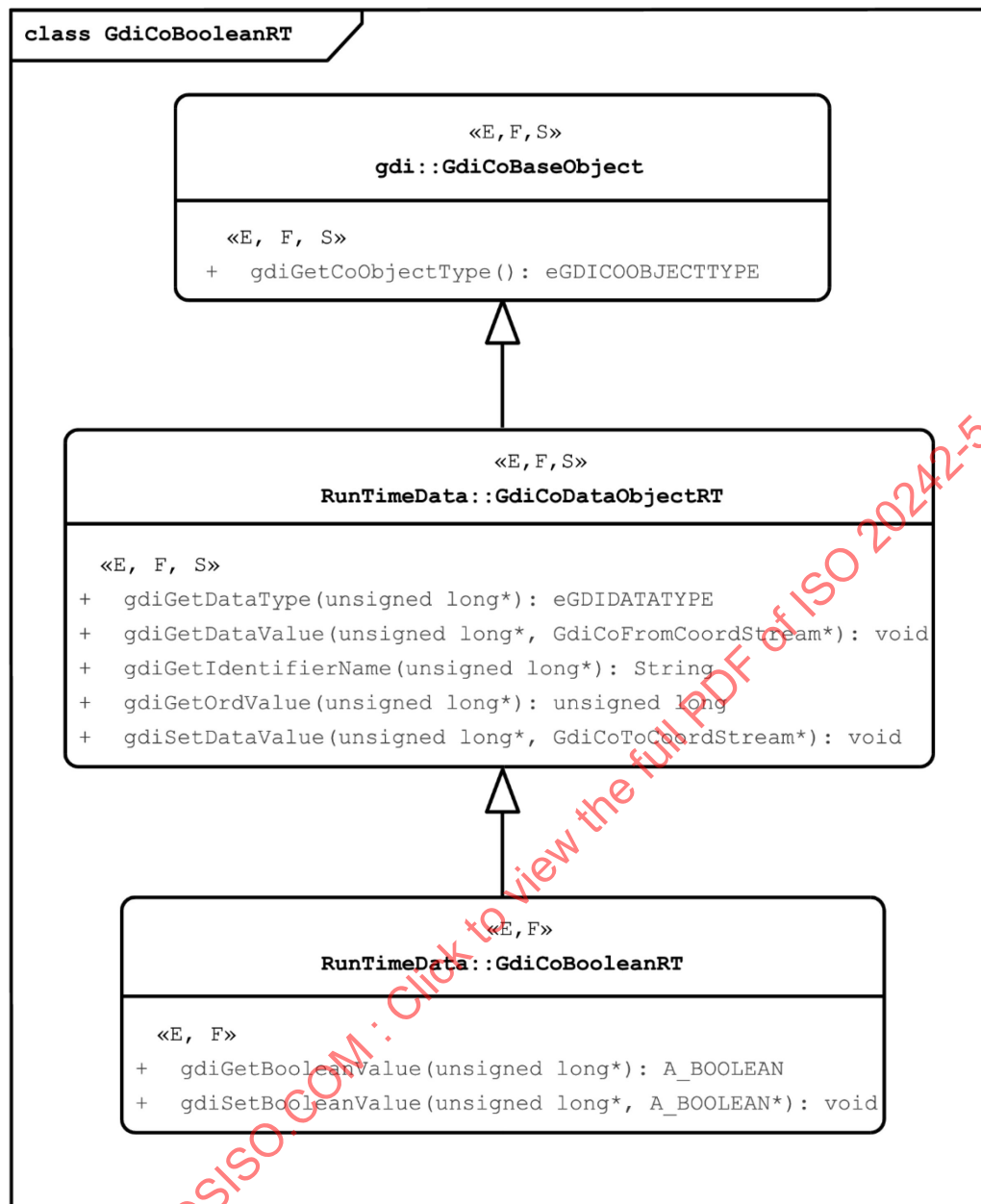


Figure B.2 — Hierarchical diagram of GdiCoBooleanRT

### B.2.2.1 GdiCoBooleanRT :: <<E,F>> gdiGetBooleanValue()

#### Function Call

```

gdiGetBooleanValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_BOOLEAN
  
```

Function Description

Returns the value of a Boolean datum. This might also be a sub-element of the type Boolean of a datum (Array, Sequence, Structure, Union).

Return Value

Returns the Boolean value of the (Sub)datum as A\_BOOLEAN:

0 = FALSE;

>0 = TRUE.

**B.2.2.2 GdiCoBooleanRT :: «E,F» gdiSetBooleanValue()**Function Call

```
gdiSetBooleanValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_BOOLEAN ucValue
    /* in parameter
    reference to the value to be set. */
) : void
```

Function Description

Copies a boolean value to the data area of the coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Union).

Return Value

None

**B.2.3 «E,F» GdiCoCharRT**

This interface contains the services for the detection of restrictions of the type Char. All other scalar types contain equivalent services with return values according to the basic type.

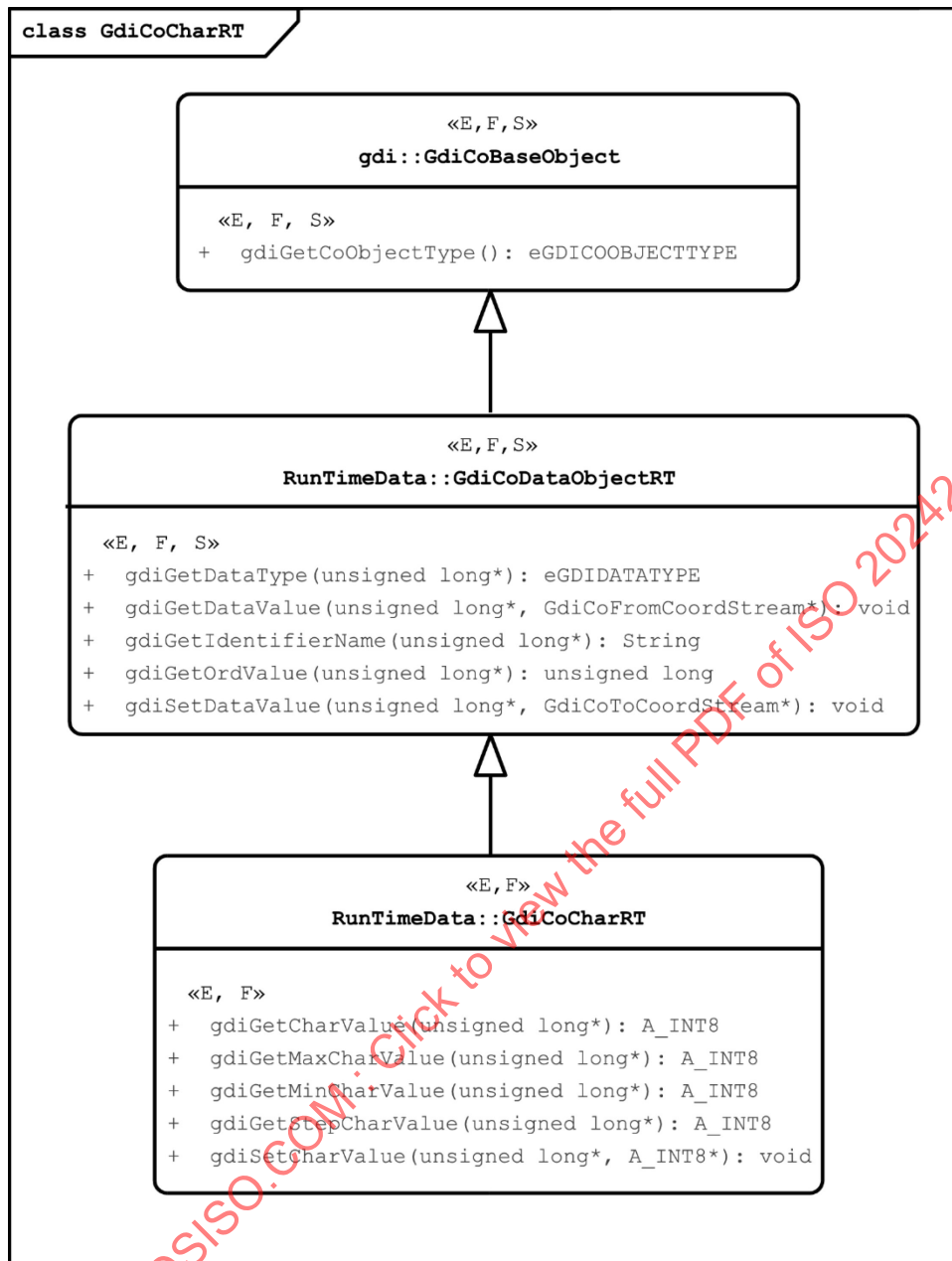


Figure B.3 — Hierarchical diagram of GdiCoCharRT

### B.2.3.1 GdiCoCharRT :: «E,F» gdiGetCharValue()

#### Function Call

```

gdiGetCharValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT8
  
```



Function Description

Returns the value of a Char datum. This might also be a sub-element of the type Char of a datum (Array, Sequence, Structure, Union).

Return Value

Returns Char value of the (Sub)datum.

**B.2.3.2 GdiCoCharRT :: «E,F» gdiGetMaxCharValue()**Function Call

```
gdiGetMaxCharValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT8
```

Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 127.

Return Value

Returns the maximum value of a datum, based upon this type.

**B.2.3.3 GdiCoCharRT :: «E,F» gdiGetMinCharValue()**Function Call

```
gdiGetMinCharValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT8
```

Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value -128.

Return Value

Returns the minimum value of a datum, based upon this type.

**B.2.3.4 GdiCoCharRT :: «E,F» gdiGetStepCharValue()**Function Call

```
gdiGetStepCharValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```
Identifier of the application returned by gdiCreateWorkspace(...) or by  
gdiGetWorkspaceByName() */  
) : A_INT8
```

#### Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

#### Return Value

Returns the step width of a datum, based upon this type.

### **B.2.3.5 GdiCoCharRT :: «E,F» gdiSetCharValue()**

#### Function Call

```
gdiSetCharValue  
(  
    inout                unsigned long ulAppHnd  
    /* in parameter  
    Identifier of the application returned by gdiCreateWorkspace(...) or by  
    gdiGetWorkspaceByName() */  
    inout                A_INT8 cValue  
    /* in parameter  
    reference to the value to be set. */  
) : void
```

#### Function Description

Copies a Char value to the data area of the Coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

#### Return Value

None

### **B.2.4 «E,F» GdiCoDoubleRT**

This interface contains services for the handling of the type Double.

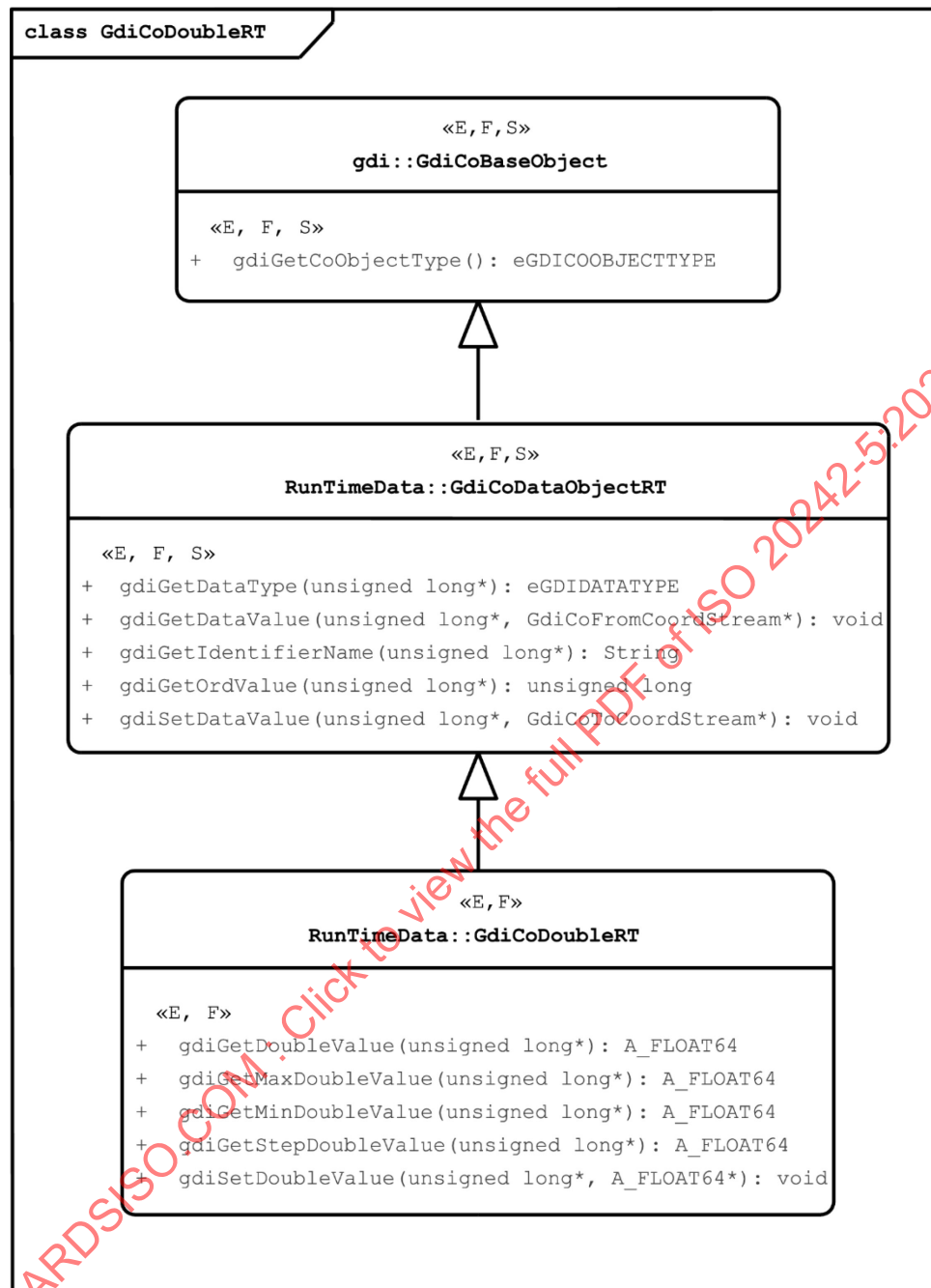


Figure B.4 — Hierarchical diagram of GdiCoDoubleRT

#### B.2.4.1 GdiCoDoubleRT :: <<E,F>> gdiGetDoubleValue()

##### Function Call

```

gdiGetDoubleValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT64
  
```

### Function Description

Returns the value of a Double datum. This might also be a sub-element of the type Char of a datum (Array, Sequence, Structure, Union).

### Return Value

Returns the double value of the (Sub)datum.

## **B.2.4.2 GdiCoDoubleRT :: «E,F» gdiGetMaxDoubleValue()**

### Function Call

```
gdiGetMaxDoubleValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT64
```

### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 1,7976931348623158 E + 308.

### Return Value

Returns the maximum value of a datum, based upon this type.

## **B.2.4.3 GdiCoDoubleRT :: «E,F» gdiGetMinDoubleValue()**

### Function Call

```
gdiGetMinDoubleValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT64
```

### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 2,2250738585072014 E - 308.

### Return Value

Returns the minimum value of a datum, based upon this type.

## **B.2.4.4 GdiCoDoubleRT :: «E,F» gdiGetStepDoubleValue()**

**Function Call**

```

gdiGetStepDoubleValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT64

```

**Function Description**

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

**Return Value**

Returns the step width of a datum, based upon this type.

**B.2.4.5 GdiCoDoubleRT :: «E,F» gdiSetDoubleValue()****Function Call**

```

gdiSetDoubleValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_FLOAT64 dValue
    /* in parameter
    reference to the value to be set. */
) : void

```

**Function Description**

Copies a Double value to the data area of the Coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

**Return Value**

None

**B.2.5 «E,F» GdiCoEnumRT**

This interface contains services for the handling of the set up type Enumeration.

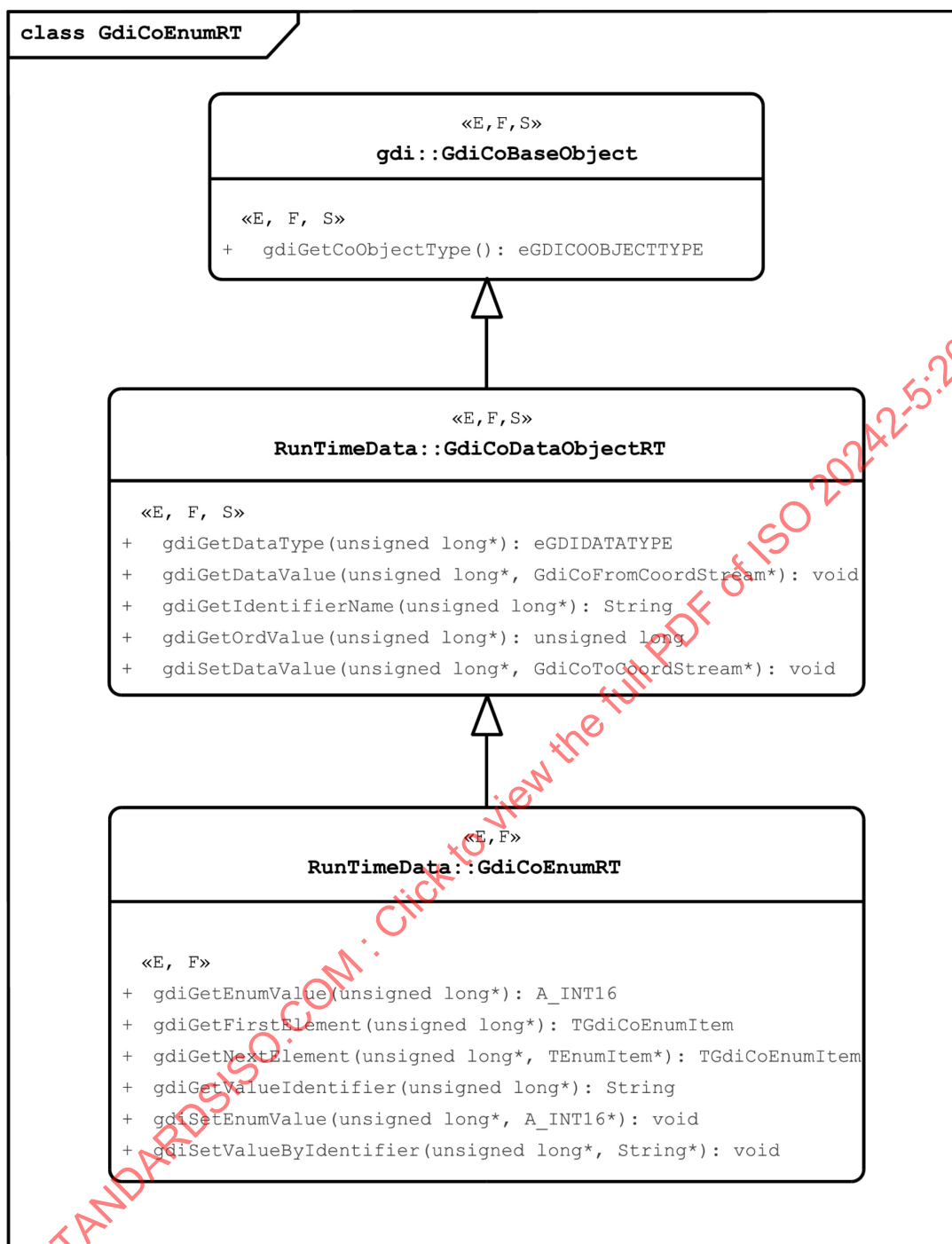


Figure B.5 — Hierarchical diagram of GdiCoEnumRT

## B.2.5.1 GdiCoEnumRT :: «E,F» gdiGetEnumValue()

Function Call

```

gdiGetEnumValue
(
    inout          unsigned long ulAppHnd
    /* in parameter
  
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
) : A_INT16

```

### Function Description

Returns the value of a Enumeration datum. This might also be a sub-element of the type enumeration of a datum (Array, Sequence, Structure, Union).

### Return Value

Returns the short value of the (Sub)datum.

## **B.2.5.2 GdiCoEnumRT :: «E,F» gdiGetFirstElement()**

### Function Call

```

gdiGetFirstElement
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : TGdiCoEnumItem

```

### Function Description

Returns the first element of an enumeration of single elements. Usually this is the element with the lowest value. The return element is a structure which consists of the sub elements String and short.

### Return Value

Returns the structure consisting of the sub elements String and short. If no element is available, the value "noGDIElement" as name and the numeric value 32768 is returned.

## **B.2.5.3 GdiCoEnumRT :: «E,F» gdiGetNextElement()**

### Function Call

```

gdiGetNextElement
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...)
    or by gdiGetWorkspaceByName() */
    inout                TEnumItem refLastElement
    /* in parameter
    contains the reference to the last Enumeration Description. */
) : TGdiCoEnumItem

```

### Function Description

Returns the next element of an enumeration of single elements. The return element is a structure which consists of the sub elements String and short.

### Return Value

If no element is available, the value "noGDIElement" as name and the numeric value 32768 is returned.

**B.2.5.4 GdiCoEnumRT :: «E,F» gdiGetValueIdentifier()****Function Call**

```

gdiGetValueIdentifier
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String

```

**Function Description**

Returns the value Identifier of the Enumeration datum as string. This might also be a sub-element of the type short of a datum (Array, Sequence, Structure, Union).

**Return Value**

Returns the Current Value as String (enum member) of the (Sub)datum.

If no element is available, the string "noGDIElement" is returned.

**B.2.5.5 GdiCoEnumRT :: «E,F» gdiSetEnumValue()****Function Call**

```

gdiSetEnumValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT16 nValue
    /* in parameter
    reference to the value to be set. */
) : void

```

**Function Description**

Copies a Short value to the data area of the coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

**Return Value**

None

**B.2.5.6 GdiCoEnumRT :: «E,F» gdiSetValueByIdentifier()****Function Call**

```

gdiSetValueByIdentifier
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String szValueIdentifier
    /* in parameter
    Identifier of the enumeration element. */

```



) : void

### Function Description

Copies the corresponding Short value to the data area of the coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

### Return Value

None

## B.2.6 «E,F» GdiCoExtendedObjectNavigator

This static interface contains services for iterating over GDI Objects and GDI types.



Figure B.6 — Hierarchical diagram of GdiCoExtendedObjectNavigator

### B.2.6.1 GdiCoExtendedObjectNavigator :: «E,F» gdiExecuteOperationData()

#### Function Call

```

gdiExecuteOperationData
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoOperationRT refOperationRT
    /* in parameter
    contains the reference to the Operation, to which the communication
    shall be performed. */
    inout                GdiCoDriverResult refDestInfo
    /* out parameter
    contains the reference to the Information Object, warnings and
    informations of the Driver will be stored in it. */
    inout                TEXEFunctionReference refCBCompleteExec

```

```

/* in parameter
Reference to a call back method which handles the execution mechanism.
A NIL reference signals a synchronous call. */
) : short

```

### Function Description

This service starts an operation. The Input and Return values remain within the area of the coordinators.

### Return Value

Describes the executed communication by the GDI driver:

0 = synchronous return;  
1 = asynchronous return;  
less than 0 = error.

## **B.2.6.2 GdiCoExtendedObjectNavigator :: «E,F» gdiGetOperationInDataObjectRT()**

### Function Call

```

gdiGetOperationInDataObjectRT
(
    inout                unsigned long ulAppHnd
    /* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
    inout                GdiCoOperationRT refOperationRT
    /* in parameter
contains the reference to the GdiCoOperationRT object. */
) : GdiCoDataObjectRT

```

### Function Description

This service returns a reference to the type of the In Data of the given operation.

### Return Value

Returns GdiCoDataObjectRT if In Data exists, 0 if Operation has no In Data.

## **B.2.6.3 GdiCoExtendedObjectNavigator :: «E,F» gdiGetOperationOutDataObjectRT()**

### Function Call

```

gdiGetOperationOutDataObjectRT
(
    inout                unsigned long ulAppHnd
    /* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
    inout                GdiCoOperationRT refOperationRT
    /* in parameter
contains the reference to the GdiCoOperationRT object. */
) : GdiCoDataObjectRT

```

### Function Description

This service returns a reference to the type of the Out Data of the given operation.

Return Value

Returns the reference to GdiCoDataObjectRT if out Data exists, 0 if Operation has no Out Data.

**B.2.6.4 GdiCoExtendedObjectNavigator :: «E,F» gdiReadCommObjectData()**Function Call

```

gdiReadCommObjectData
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoCommObjectRT refCommObjectRT
    /* in parameter
    contains the reference to the Communication Object to which the
    communication will be executed. */
    inout                GdiCoDriverResult refDestInfo
    /* out parameter
    contains the reference to the Information Object, warnings and
    informations of the Driver will be stored in it. */
    inout                TCBFunctionReference refCBCompleteRead
    /* in parameter
    Reference to a call back method which handles the read mechanism. A NIL
    reference signals a synchronous call. */
) : short

```

Function Description

Start the data exchange with the driver (GdiRead).

Uses stored data inside the GdiDataObject for write or stores data to the GdiDataObject during read process.

Causes a data exchange between coordinator and driver.

Return Value

Describes the executed communication by the GDI driver:

0 = synchronous return;

1 = asynchronous return.

**B.2.6.5 GdiCoExtendedObjectNavigator :: «E,F» gdiWriteCommObjectData()**Function Call

```

gdiWriteCommObjectData
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoCommObjectRT refCommObjectRT
    /* in parameter
    contains the reference to the Communication Object to which the
    communication will be executed. */
    inout                GdiCoDriverResult refDestInfo
    /* out parameter

```

```
contains the reference to the Information Object, warnings and
informations of the Driver will be stored in it. */
inout          TCBFunctionReference refCBCompleteWrite
/* in parameter
Reference to a call back method which handles the write mechanism. A NIL
reference signals a synchronous call. */
) : short
```

### Function Description

Start the data exchange with the driver (GdiWrite).

Uses stored data inside the GdiDataObject for write or stores data to the GdiDataObject during read process.

Causes a data exchange between coordinator and driver.

### Return Value

Describes the executed communication by the GDI driver:

0 = synchronous return;

1 = asynchronous return.

### **B.2.7 «E,F» GdiCoFloatRT**

This interface contains services for the handling of the type Float

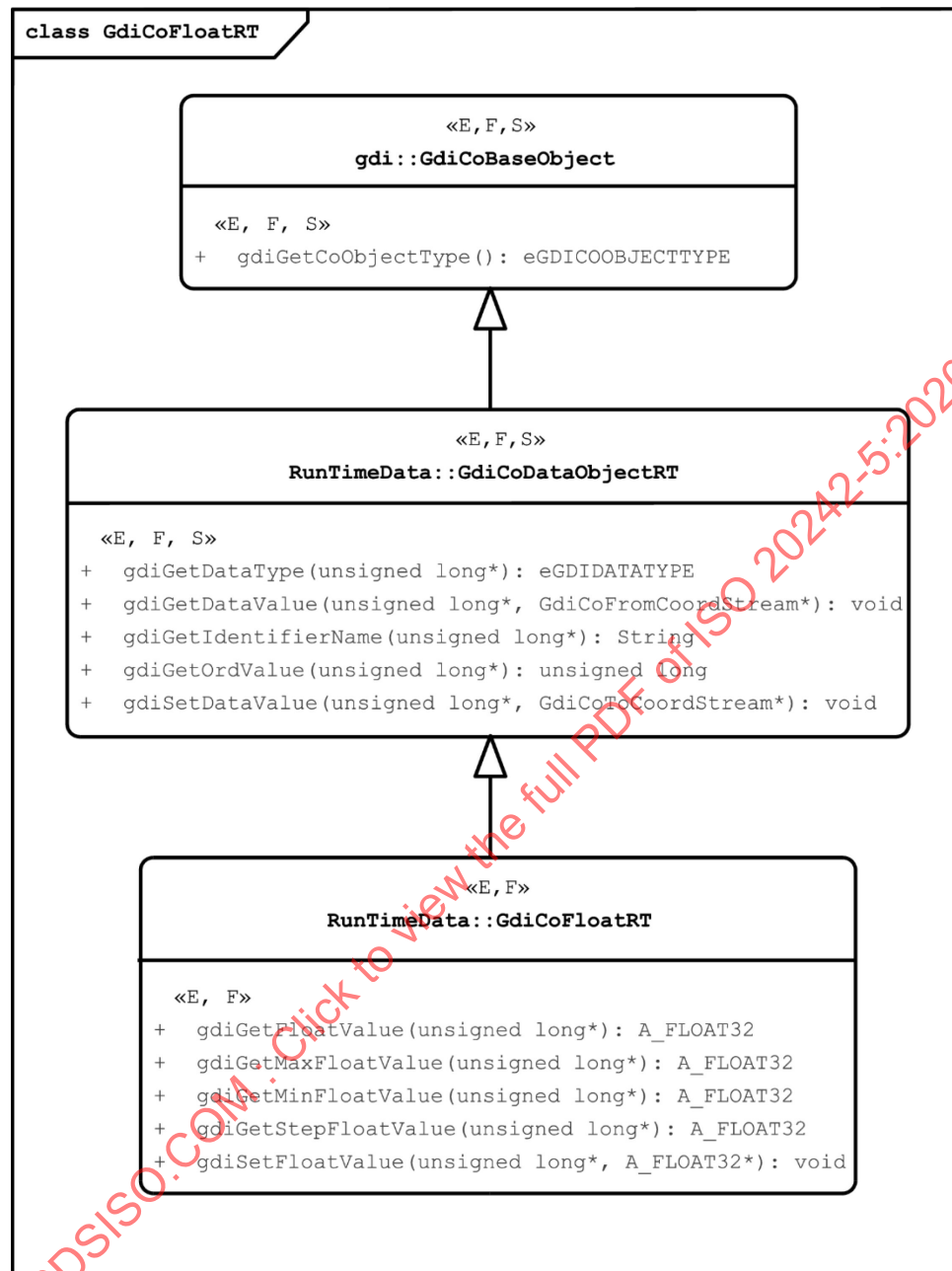


Figure B.7 — Hierarchical diagram of GdiCoFloatRT

**B.2.7.1 GdiCoFloatRT :: <<E,F>> gdiGetFloatValue()**Function Call

```

gdiGetFloatValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT32
  
```

### Function Description

Returns the value of a Float datum. This might also be a sub-element of the type Char of a datum (Array, Sequence, Structure, Union).

### Return Value

Returns the Float value of the (Sub)datum.

## **B.2.7.2 GdiCoFloatRT :: «E,F» gdiGetMaxFloatValue()**

### Function Call

```
gdiGetMaxFloatValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT32
```

### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 3,402823466 E + 38.

### Return Value

Returns the maximum value of a datum, based upon this type.

## **B.2.7.3 GdiCoFloatRT :: «E,F» gdiGetMinFloatValue()**

### Function Call

```
gdiGetMinFloatValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT32
```

### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 1,175494351 E - 38.

### Return Value

Returns the minimum value of a datum, based upon this type.

## **B.2.7.4 GdiCoFloatRT :: «E,F» gdiGetStepFloatValue()**

### Function Call

```
gdiGetStepFloatValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
) : A_FLOAT32

```

### Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

### Return Value

Returns the step width of a datum, based upon this type.

## **B.2.7.5 GdiCoFloatRT :: «E,F» gdiSetFloatValue()**

### Function Call

```

gdiSetFloatValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_FLOAT32 fValue
    /* in parameter
    reference to the value to be set. */
) : void

```

### Function Description

Copies a Float value to the data area of the Coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

### Return Value

None

## **B.2.8 «E,F» GdiCoFuncObjectRefRT**

This interface shall contain services for the handling of the type GdiCoFuncObjRefRT. The reference type shall be the same as the result type of the gdiGetFuncObjectRTByName method of gdiWorkspace. The referenced Function Object shall be instantiated by the coordinator. If an attribute of this type is changed by a write operation, the referenced Function Object shall not be deleted. The attribute shall refer to another Function Object.

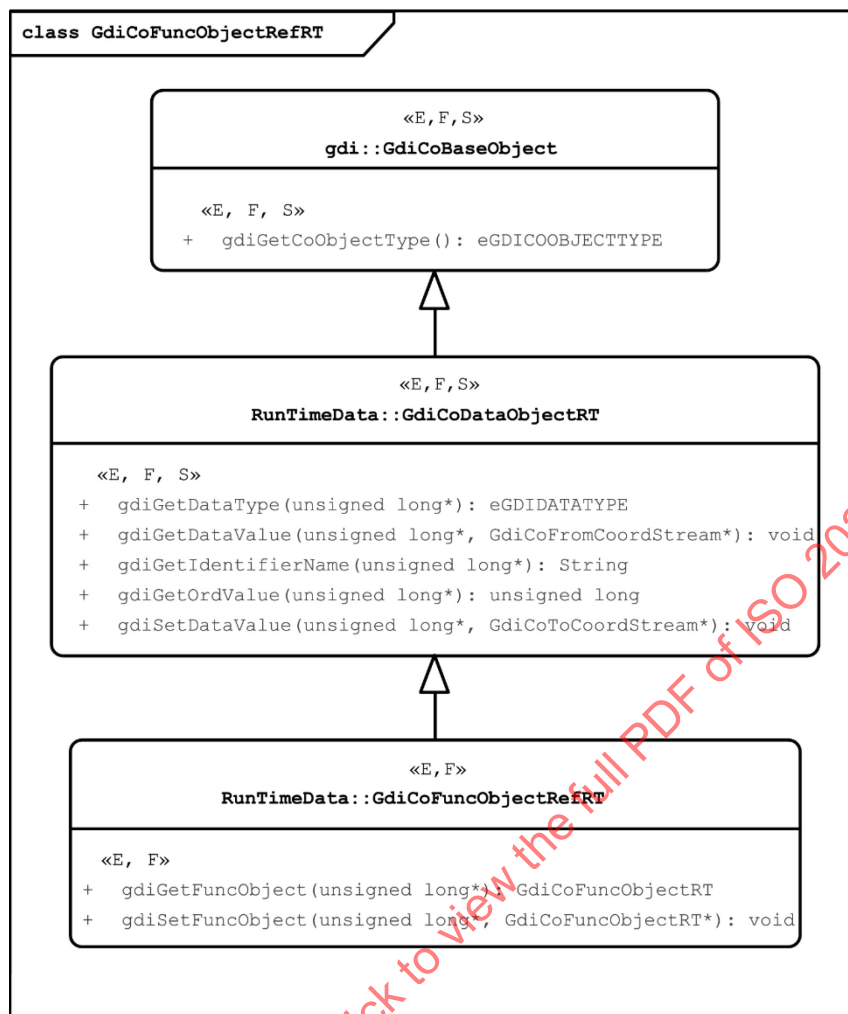


Figure B.8 — Hierarchical diagram of GdiCoFuncObjectRefRT

### B.2.8.1 GdiCoFuncObjectRefRT :: «E,F» gdiGetFuncObject()

#### Function Call

```

gdiGetFuncObject
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoFuncObjectRT
  
```

#### Function Description

Returns the value (FO instance reference) of the datum. This might also be a sub-element of the type Char of a datum (Array, Sequence, Structure, Union).

#### Return Value

Returns reference of a func object. This might also be a sub-element of the type FoReference of a datum (Array, Sequence, Structure, Union).



**B.2.8.2 GdiCoFuncObjectRefRT :: «E,F» gdiSetFuncObject()****Function Call**

```

gdiSetFuncObject
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoFuncObjectRT refFuncObject
    /* in parameter
    reference of a func object */
) : void

```

**Function Description**

Copies the handle and the id of the func object to the data area of the coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

**Return Value**

None

**B.2.9 «E,F» GdiCoLongRT**

This interface contains services for the handling of the type Long.

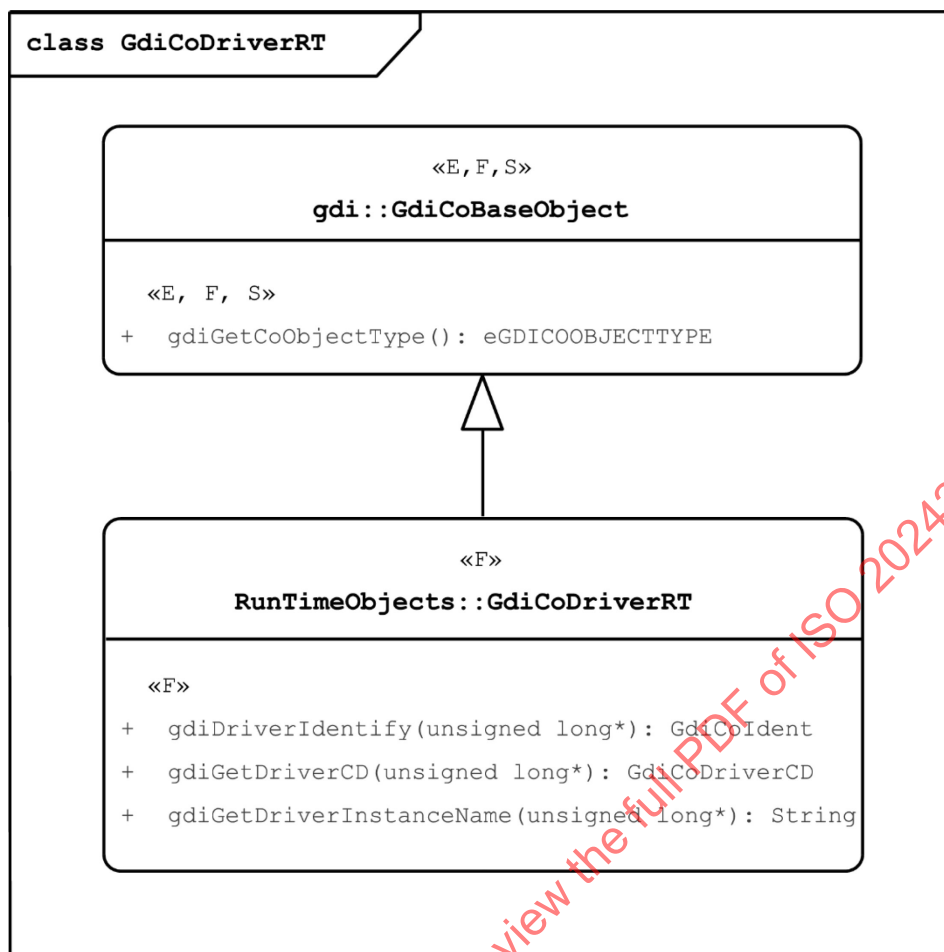


Figure B.9 — Hierarchical diagram of GdiCoLongRT

### B.2.9.1 GdiCoLongRT :: «E,F» gdiGetLongValue()

#### Function Call

```

gdiGetLongValue
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT32
  
```

#### Function Description

Returns the value of a Long datum. This might also be a sub-element of the type Char of a datum (Array, Sequence, Structure, Union).

#### Return Value

Returns Long value of the (Sub)datum.

**B.2.9.2 GdiCoLongRT :: «E,F» gdiGetMaxLongValue()****Function Call**

```

gdiGetMaxLongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT32

```

**Function Description**

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 2147483647.

**Return Value**

Returns the maximum value of a datum, based upon this type.

**B.2.9.3 GdiCoLongRT :: «E,F» gdiGetMinLongValue()****Function Call**

```

gdiGetMinLongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT32

```

**Function Description**

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value -2147483648.

**Return Value**

Returns the minimum value of a datum, based upon this type.

**B.2.9.4 GdiCoLongRT :: «E,F» gdiGetStepLongValue()****Function Call**

```

gdiGetStepLongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT32

```

**Function Description**

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

### Return Value

Returns the step width of a datum, based upon this type.

### **B.2.9.5 GdiCoLongRT :: «E,F» gdiSetLongValue()**

#### Function Call

```
gdiSetLongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT32 lValue
    /* in parameter
    reference to the value to be set. */
) : void
```

#### Function Description

Copies a Long value to the data area of the Coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

### Return Value

None

### **B.2.10 «E,F» GdiCoOctetRT**

This interface contains services for the handling of the type octet (unsigned char).

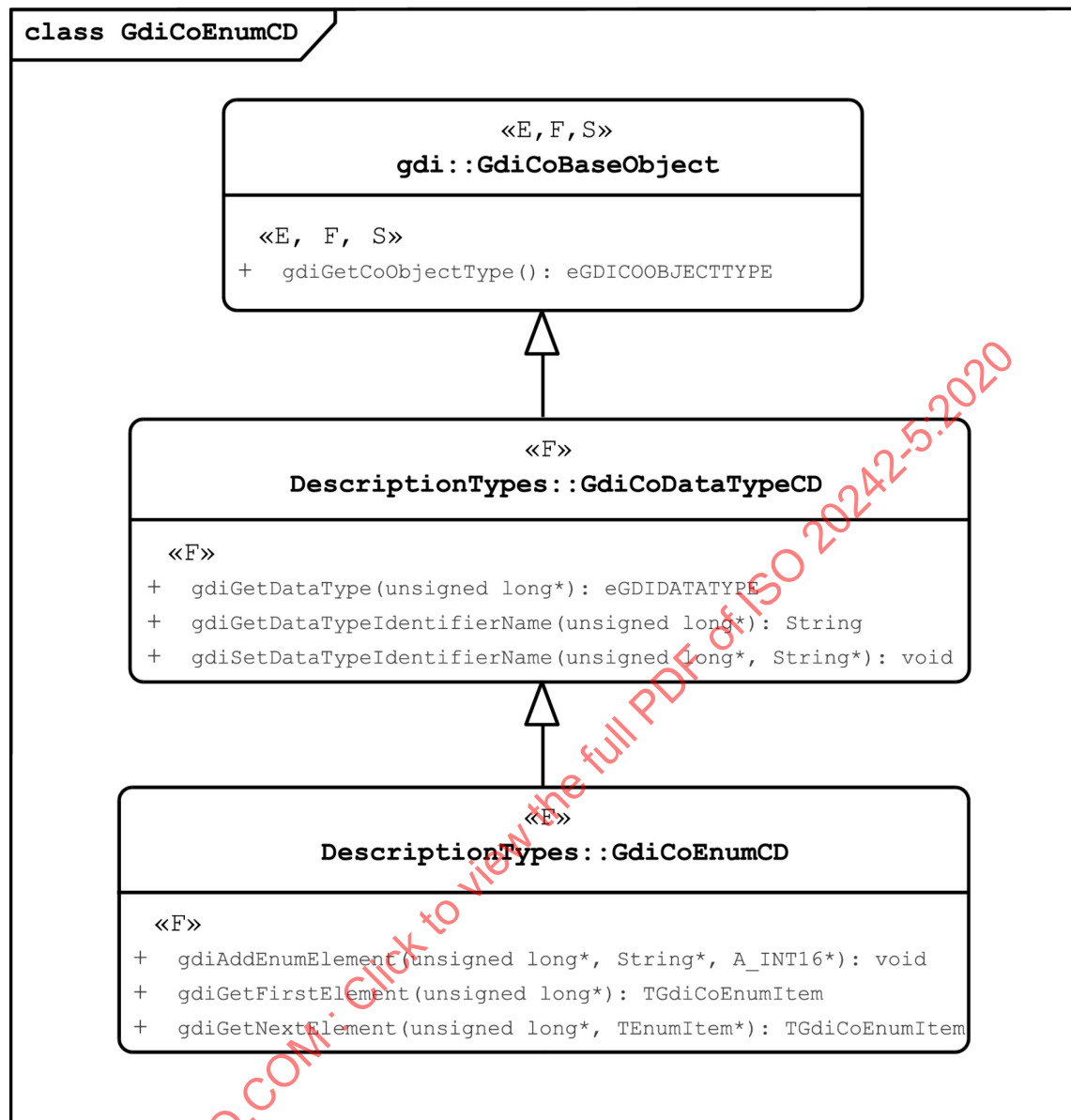


Figure B.10 — Hierarchical diagram of GdiCoOctetRT

### B.2.10.1 GdiCoOctetRT :: «E,F» gdiGetMaxOctetValue()

#### Function Call

```

gdiGetMaxOctetValue
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT8
  
```

#### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 255.

### Return Value

Returns the maximum value of a datum, based upon this type.

### **B.2.10.2 GdiCoOctetRT :: «E,F» gdiGetMinOctetValue()**

#### Function Call

```
gdiGetMinOctetValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT8
```

#### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 0.

### Return Value

Returns the minimum value of a datum, based upon this type.

### **B.2.10.3 GdiCoOctetRT :: «E,F» gdiGetOctetValue()**

#### Function Call

```
gdiGetOctetValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT8
```

#### Function Description

Returns the value of an Unsigned Char datum. This might also be a sub-element of the type Char of a datum (Array, Sequence, Structure, Union).

### Return Value

Returns the Unsigned Char value of the (Sub)datum.

### **B.2.10.4 GdiCoOctetRT :: «E,F» gdiGetStepOctetValue()**

#### Function Call

```
gdiGetStepOctetValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT8
```

**Function Description**

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

**Return Value**

Returns the step width of a datum, based upon this type.

**B.2.10.5 GdiCoOctetRT :: «E,F» gdiSetOctetValue()****Function Call**

```
gdiSetOctetValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_UINT8 ucValue
    /* in parameter
    reference to the value to be set. */
) : void
```

**Function Description**

Copies an Unsigned Char value to the data area of the coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

**Return Value**

None

**B.2.11 «E,F» GdiCoSequenceRT**

This interface contains services for the handling of the type Sequence.

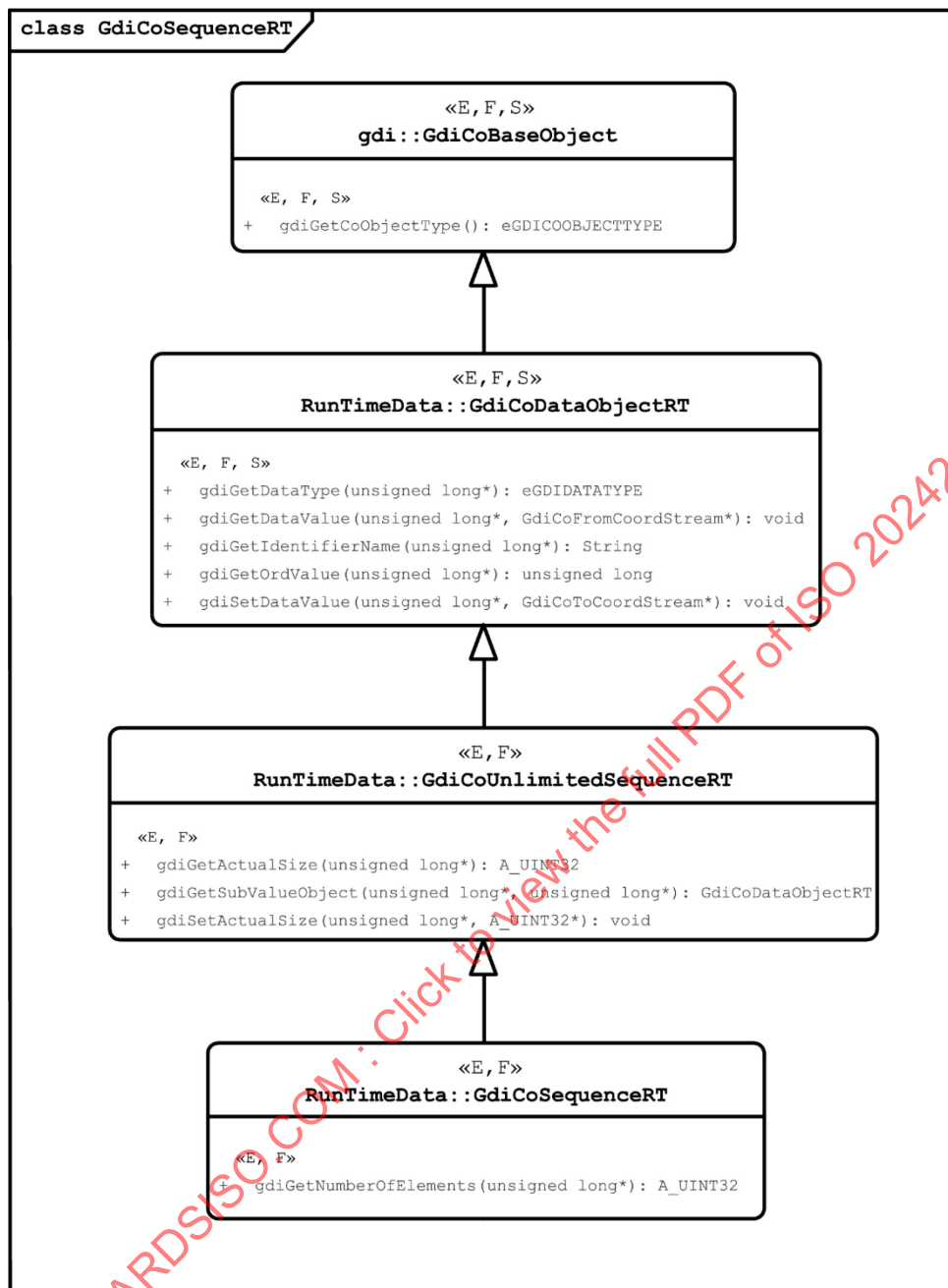


Figure B.11 — Hierarchical diagram of GdiCoSequenceRT

## B.2.11.1 GdiCoSequenceRT :: «E,F» gdiGetNumberOfElements()

## Function Call

```

gdiGetNumberOfElements
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32
  
```



Function Description

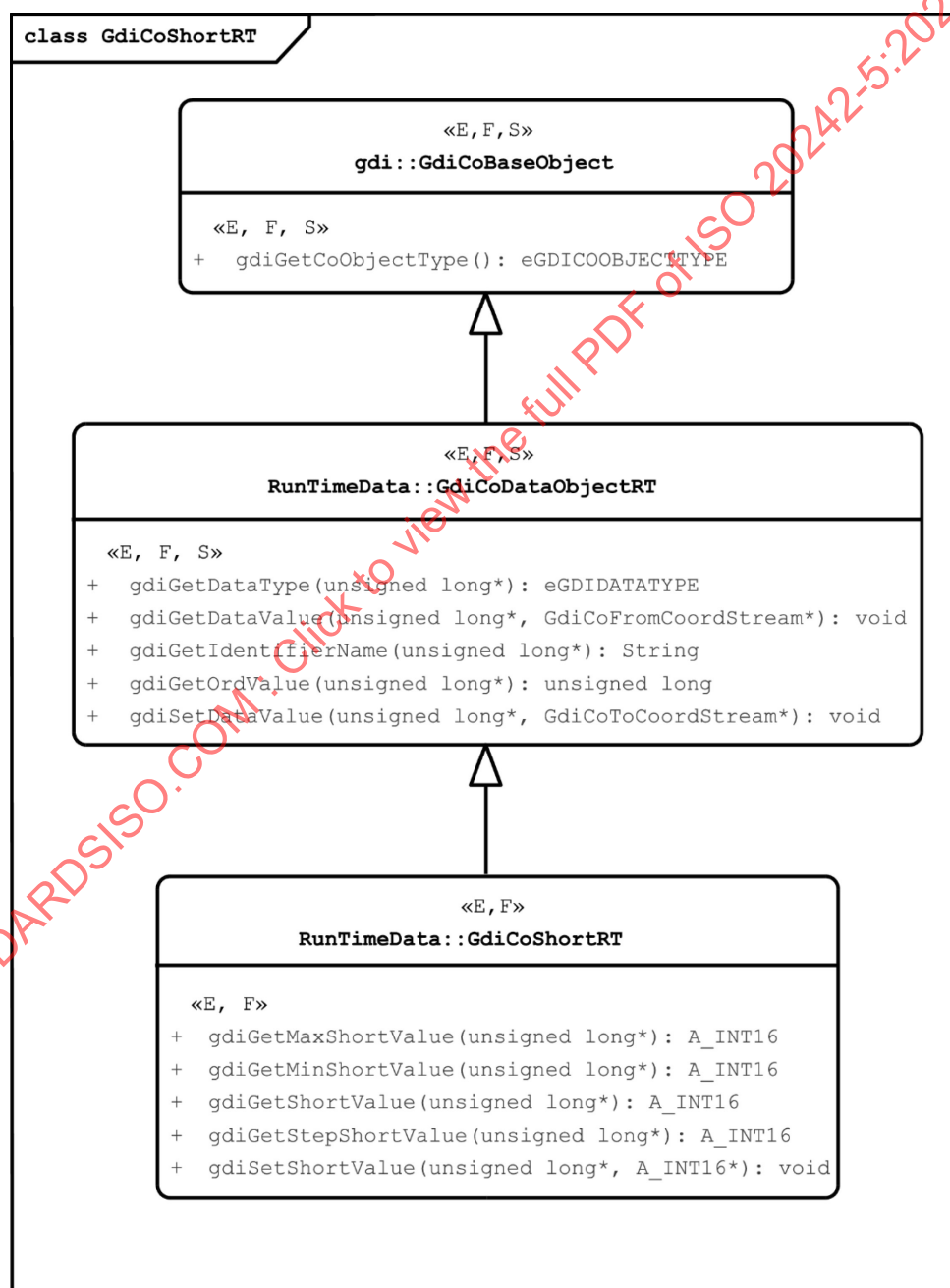
It is used for the detection of the maximum number of elements of a Sequence.

Return Value

Returns the maximum number of elements of a Sequence.

**B.2.12 «E,F» GdiCoShortRT**

This interface contains services for the handling of the type Short.



**Figure B.12 — Hierarchical diagram of GdiCoShortRT**

**B.2.12.1 GdiCoShortRT :: «E,F» gdiGetMaxShortValue()****Function Call**

```

gdiGetMaxShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT16

```

**Function Description**

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 32767.

**Return Value**

Returns the maximum value of a datum, based upon this type.

**B.2.12.2 GdiCoShortRT :: «E,F» gdiGetMinShortValue()****Function Call**

```

gdiGetMinShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT16

```

**Function Description**

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value -32768.

**Return Value**

Returns the minimum value of a datum, based upon this type.

**B.2.12.3 GdiCoShortRT :: «E,F» gdiGetShortValue()****Function Call**

```

gdiGetShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT16

```

**Function Description**

Returns the value of a Short datum. This might also be a sub-element of the type short of a datum (Array, Sequence, Structure, Union).

Return Value

Returns Short value of the (Sub)datum.

**B.2.12.4 GdiCoShortRT :: «E,F» gdiGetStepShortValue()**Function Call

```
gdiGetStepShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT16
```

Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

Return Value

Returns the step width of a datum, based upon this type.

**B.2.12.5 GdiCoShortRT :: «E,F» gdiSetShortValue()**Function Call

```
gdiSetShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT16 nValue
    /* in parameter
    reference to the value to be set. */
) : void
```

Function Description

Copies a Short value to the data area of the coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

Return Value

None

**B.2.13 «E,F» GdiCoStructRT**

This interface contains services for the handling of the type Structure.

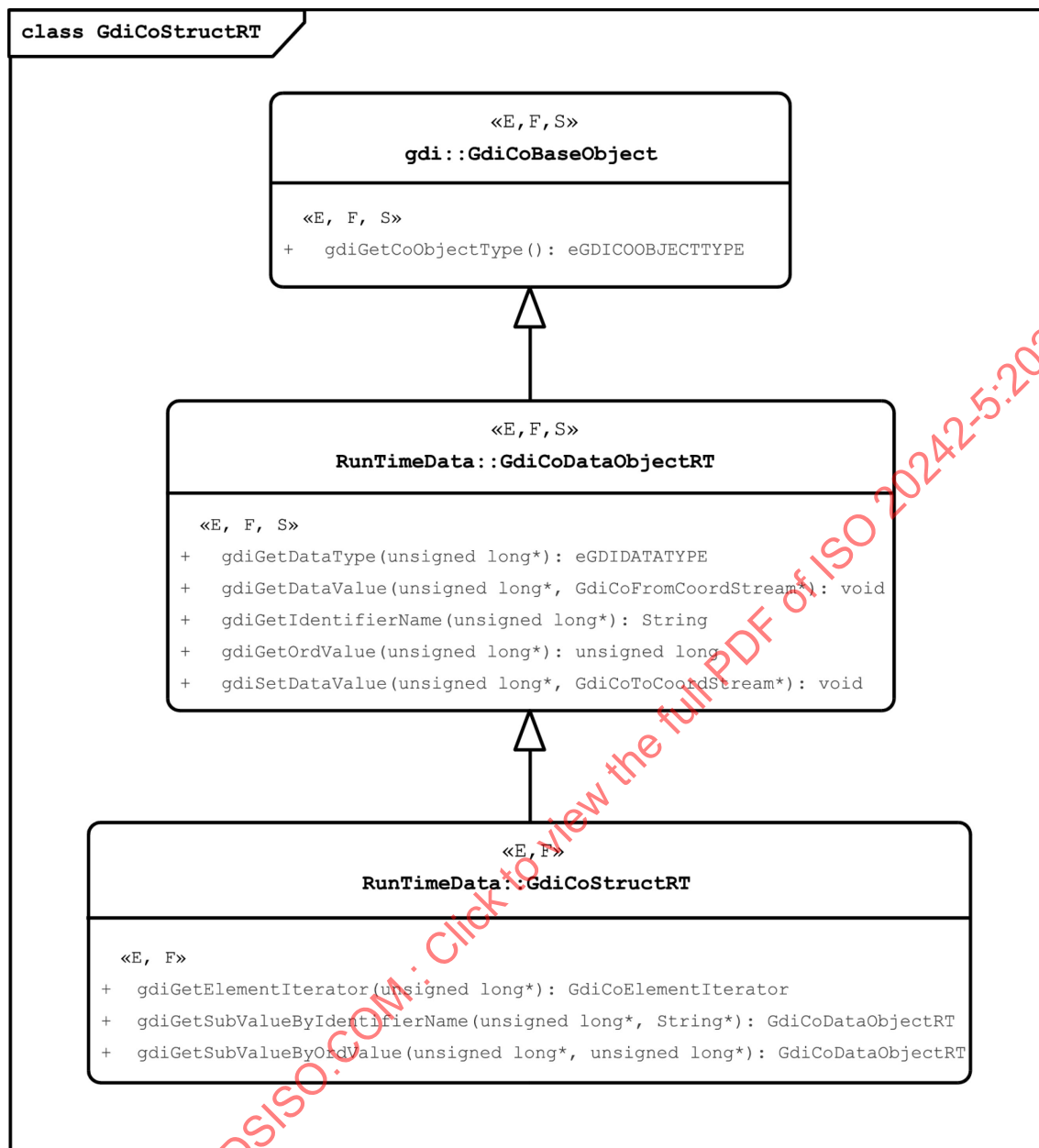


Figure B.13 — Hierarchical diagram of GdiCoStructRT

## B.2.13.1 GdiCoStructRT :: «E,F» gdiGetElementIterator()

Function Call

```

gdiGetElementIterator
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoElementIterator

```

**Function Description**

Generates an element iterator, by means of which all sub elements of the structure can be detected. The iterator is reset.

**Return Value**

Returns the Reference (Pointer, Handle, Class) to a element iterator.

**B.2.13.2 GdiCoStructRT :: «E,F» gdiGetSubValueByIdentifierName()****Function Call**

```
gdiGetSubValueByIdentifierName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sElementName
    /* in parameter
    Element Identifier of the referenced structure element. */
) : GdiCoDataObjectRT
```

**Function Description**

Returns the reference to a structure element. The Data object instance is returned as reference of basic type. The name is used as selector. The actual type of the structure element (if unknown) may be detected by means of gdiGetType(). This reference may then be used as a reference to the detected data type.

**Return Value**

Returns the reference to a datum, NIL if it does not exist.

**B.2.13.3 GdiCoStructRT :: «E,F» gdiGetSubValueByOrdValue()****Function Call**

```
gdiGetSubValueByOrdValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                unsigned long ulSection
    /* in parameter
    Ordinal value of the structure element. */
) : GdiCoDataObjectRT
```

**Function Description**

Returns the reference to a structure element. The basic type is returned as reference. A numeric value, which describes the position of the structure element within the structure, is used as selector. The actual type of the structure element (if unknown) may be detected by means of gdiGetType(). This reference may then be used as a reference to the detected data type.

**Return Value**

Returns the reference to a datum.

## B.2.14 «E,F» GdiCoULongRT

This interface contains services for the handling of the type unsigned Long.

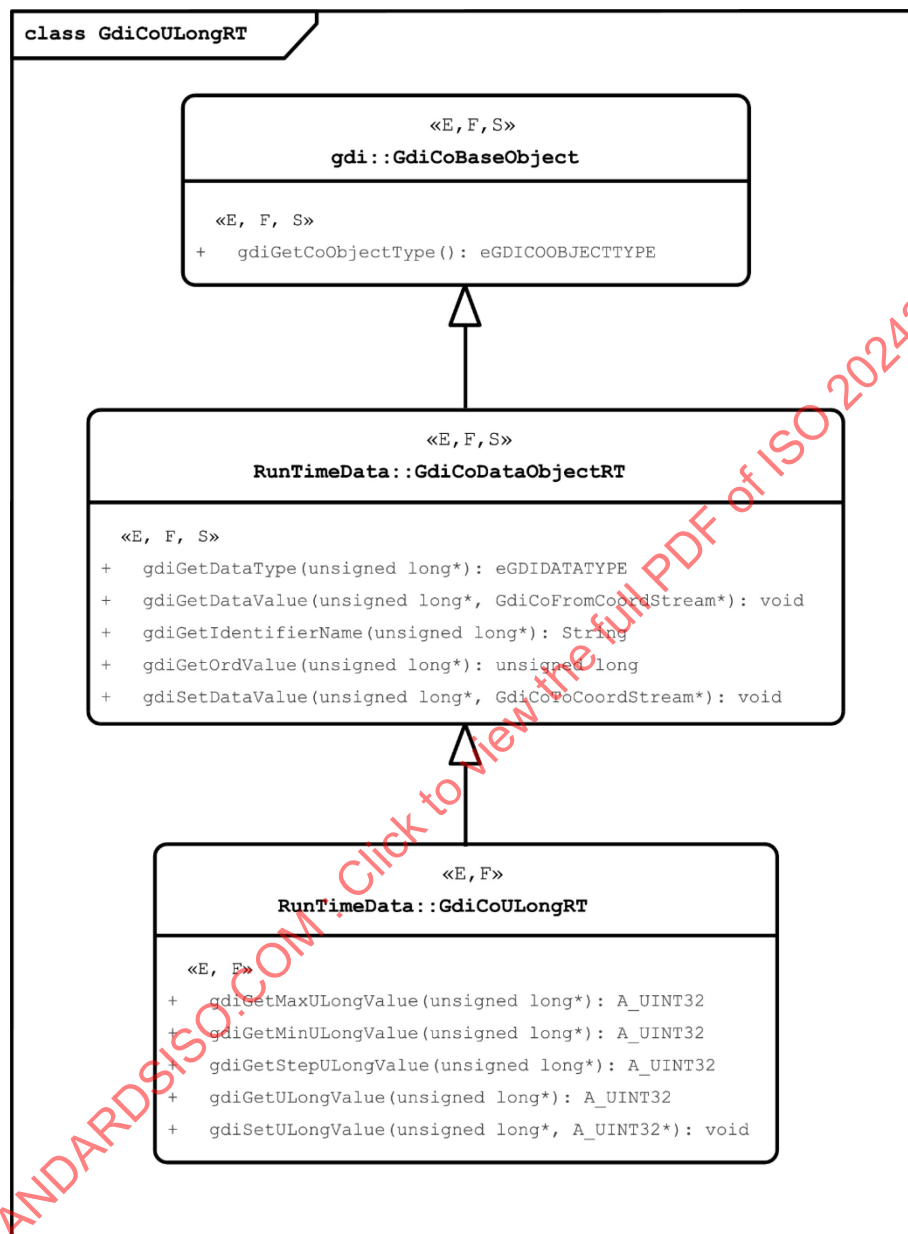


Figure B.14 — Hierarchical diagram of GdiCoULongRT

## B.2.14.1 GdiCoULongRT :: «E,F» gdiGetMaxULongValue()

Function Call

```

gdiGetMaxULongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
  
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
) : A_UINT32

```

#### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 4294967295.

#### Return Value

Returns the maximum value of a datum, based upon this type.

### **B.2.14.2 GdiCoULongRT :: «E,F» gdiGetMinULongValue()**

#### Function Call

```

gdiGetMinULongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32

```

#### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 0.

#### Return Value

Returns the minimum value of a datum, based upon this type.

### **B.2.14.3 GdiCoULongRT :: «E,F» gdiGetStepULongValue()**

#### Function Call

```

gdiGetStepULongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32

```

#### Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

#### Return Value

Returns the step width of a datum, based upon this type.

**B.2.14.4 GdiCoULongRT :: «E,F» gdiGetULongValue()****Function Call**

```

gdiGetULongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32

```

**Function Description**

Returns the value of an Unsigned Long datum. This might also be a sub-element of the type Char of a datum (Array, Sequence, Structure, Union).

**Return Value**

Returns the Unsigned Long value of the (Sub)datum.

**B.2.14.5 GdiCoULongRT :: «E,F» gdiSetULongValue()****Function Call**

```

gdiSetULongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_UINT32 ulValue
    /* in parameter
    reference to the value to be set. */
) : void

```

**Function Description**

Copies an Unsigned Long value to the data area of the coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Unions).

**Return Value**

None

**B.2.15 «E,F» GdiCoUnionRT**

This interface contains services for the handling of the type Union.



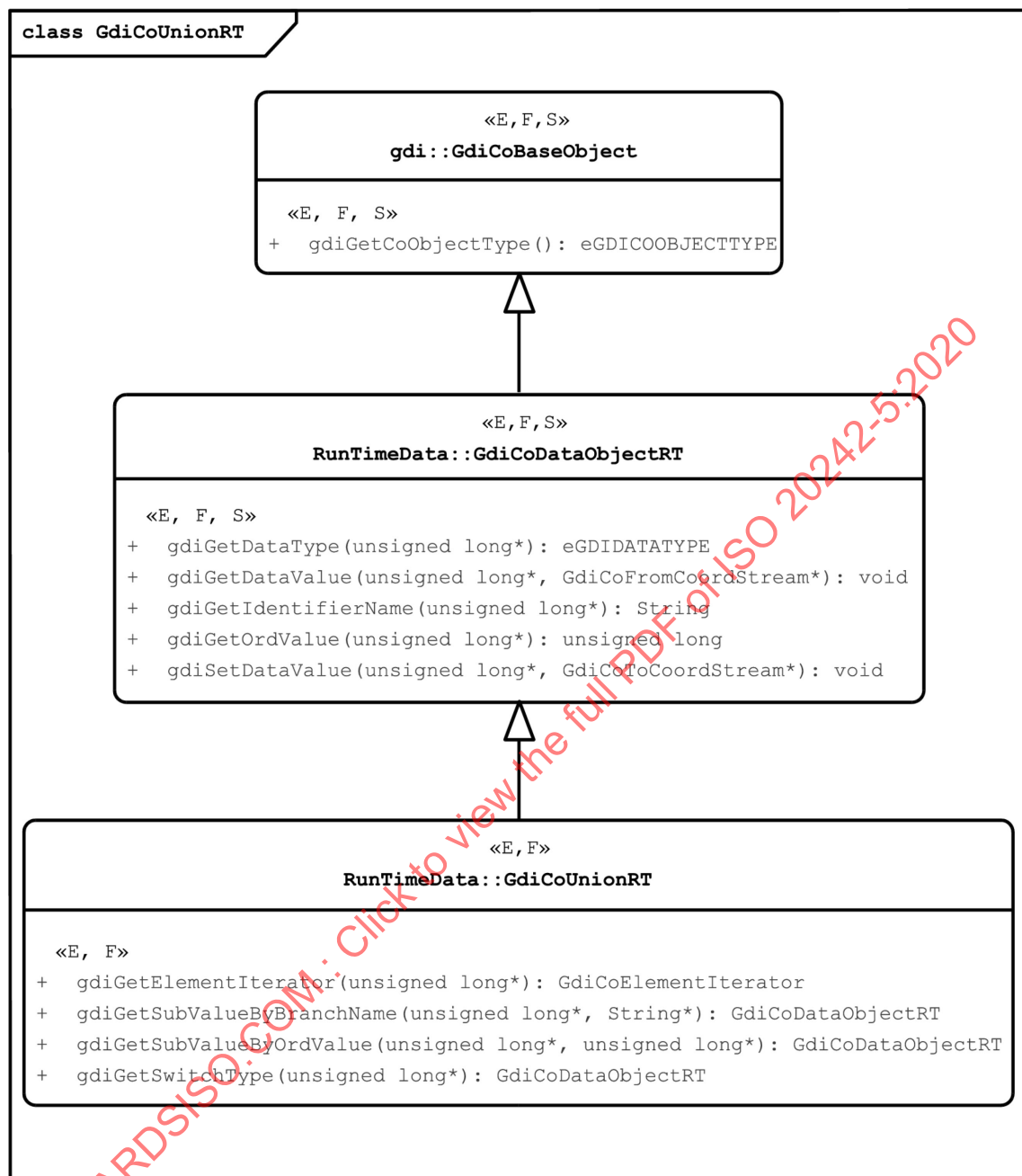


Figure B.15 — Hierarchical diagram of GdiCoUnionRT

### B.2.15.1 GdiCoUnionRT :: «E,F» gdiGetElementIterator()

#### Function Call

```

gdiGetElementIterator
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoElementIterator
  
```

Function Description

Generates a element iterator, by means of which all sub elements of the union can be detected. The iterator is set to the first element of the union area.

Return Value

Returns the reference (Pointer, Handle, Class) to a type iterator.

**B.2.15.2 GdiCoUnionRT :: «E,F» gdiGetSubValueByBranchName()**Function Call

```
gdiGetSubValueByBranchName
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout          String sBranch
    /* in parameter
    Section name of the referenced structure element. */
) : GdiCoDataObjectRT
```

Function Description

Returns the reference to an overlay element of a Union. The basic type is returned as reference. The element name is used as selector. The actual type of the structure element (if unknown) may be detected by means of gdiGetType(). This reference may then be used as a reference to the detected data type.

If the name of the default branch is given, the type of the default branch will be returned.

Return Value

Returns the reference to a datum.

**B.2.15.3 GdiCoUnionRT :: «E,F» gdiGetSubValueByOrdValue()**Function Call

```
gdiGetSubValueByOrdValue
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout          unsigned long ulBranch
    /* in parameter
    Switch value of the structure element. */
) : GdiCoDataObjectRT
```

Function Description

Returns the reference to an overlay element of a Union. The basic type is returned as reference. The switch value is used as selector (branch). The actual type of the structure element (if unknown) may be detected by means of gdiGetType(). This reference may then be used as a reference to the detected data type.

If no branch matches to the set value, the default branch is used if it exists.

#### Return Value

Returns the reference to a datum.

### **B.2.15.4 GdiCoUnionRT :: «E,F» gdiGetSwitchType()**

#### Function Call

```
gdiGetSwitchType
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataObjectRT
```

#### Function Description

Generates a reference to RefGdiCoBaseType. The actual type of the Switch value is detected by the service gdiGetType() in relation to the returned reference.

The branch selector can be set at the returned sub object. If no branch matches to the set value, the default branch is used if it exists.

#### Return Value

Returns the type of the Switch value.

### **B.2.16 «E,F» GdiCoUnlimitedSequenceRT**

This interface contains services for the handling of the type Unlimited Sequence

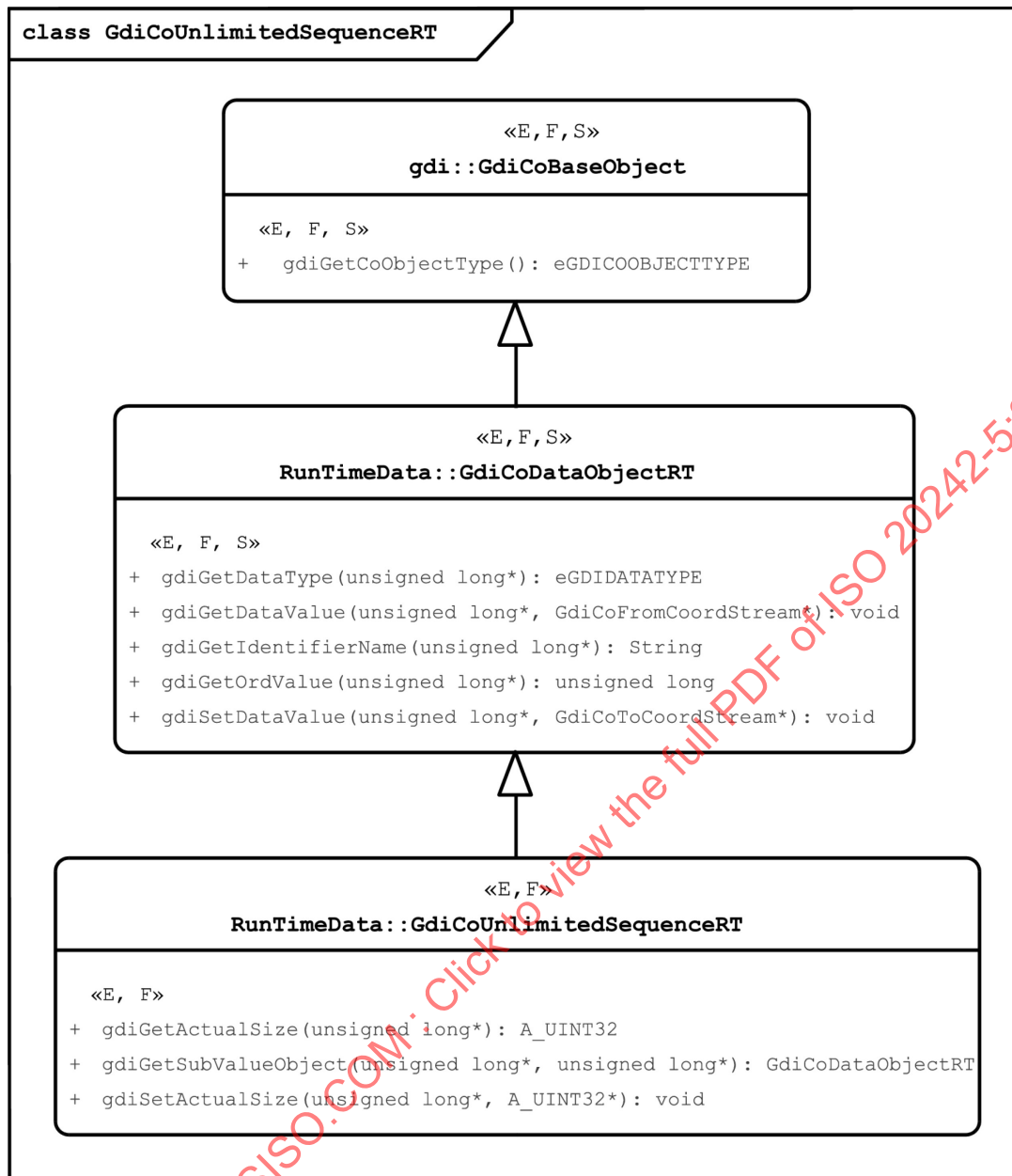


Figure B.16 — Hierarchical diagram of GdiCoUnlimitedSequenceRT

## B.2.16.1 GdiCoUnlimitedSequenceRT :: «E,F» gdiGetActualSize()

Function Call

```

gdiGetActualSize
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32
  
```

**Function Description**

Returns the number of valid data elements within a Sequence array. Starting from index 0, the data are located contiguously within the array.

**Return Value**

Returns the number of available sequence array elements.

**B.2.16.2 GdiCoUnlimitedSequenceRT :: «E,F» gdiGetSubValueObject()****Function Call**

```
gdiGetSubValueObject
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                unsigned long ulIndex
    /* in parameter
    Index Value of the data element */
) : GdiCoDataObjectRT
```

**Function Description**

Returns the reference to a data element within a Sequence or an Array. After this, the type of this element (if unknown) may be detected by means of gdiGetType() and the data area of the array element may be accessed in relation to the type.

**Return Value**

Returns a reference to a sub value object of the sequence.

**B.2.16.3 GdiCoUnlimitedSequenceRT :: «E,F» gdiSetActualSize()****Function Call**

```
gdiSetActualSize
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_UINT32 ulActualSize
    /* in parameter
    New number of valid data of a Sequence */
) : void
```

**Function Description**

Sets the number of valid data elements within a Sequence array Starting from index 0, the data are located contiguously within the array. If the number of data is increased, the contents of the added data are undefined.

**Return Value**

None

## B.2.17 «E,F» GdiCoUShortRT

This interface contains services for the handling of the type unsigned Short.

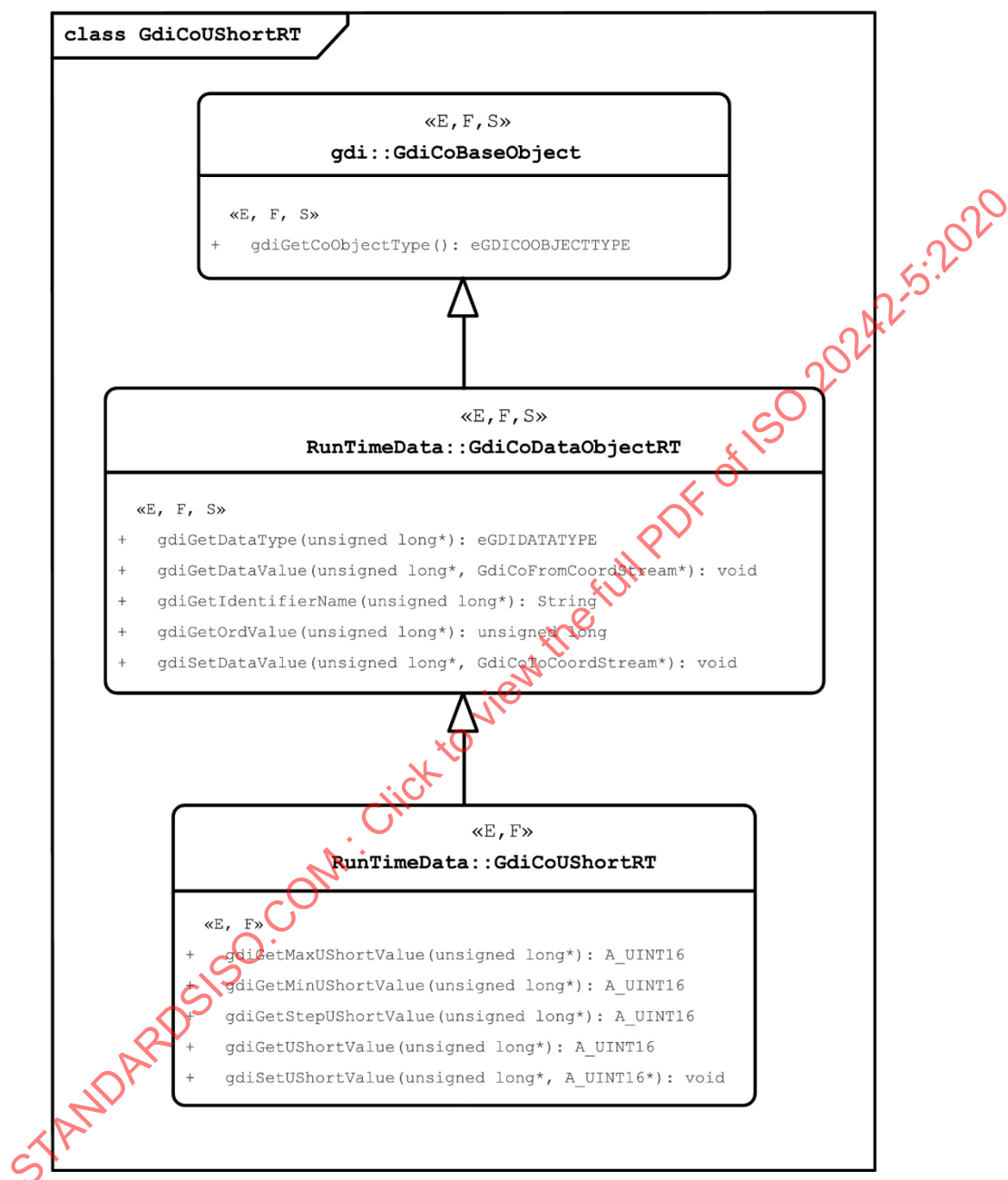


Figure B.17 — Hierarchical diagram of GdiCoUShortRT

## B.2.17.1 GdiCoUShortRT :: «E,F» gdiGetMaxUShortValue()

## Function Call

```

gdiGetMaxUShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
  
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
) : A_UINT16

```

### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 65535.

### Return Value

Returns the maximum value of a datum, based upon this type.

## **B.2.17.2 GdiCoUShortRT :: «E,F» gdiGetMinUShortValue()**

### Function Call

```

gdiGetMinUShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT16

```

### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 0.

### Return Value

Returns the minimum value of a datum, based upon this type.

## **B.2.17.3 GdiCoUShortRT :: «E,F» gdiGetStepUShortValue()**

### Function Call

```

gdiGetStepUShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT16

```

### Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

### Return Value

Returns the step width of a datum, based upon this type.

**B.2.17.4 GdiCoUShortRT :: «E,F» gdiGetUShortValue()****Function Call**

```

gdiGetUShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT16

```

**Function Description**

Returns the value of an Unsigned Short datum. This might also be a sub-element of the type unsigned short of a datum (Array, Sequence, Structure, Union).

**Return Value**

Returns the Unsigned Short value of the (Sub)datum.

**B.2.17.5 GdiCoUShortRT :: «E,F» gdiSetUShortValue()****Function Call**

```

gdiSetUShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_UINT16 unValue
    /* in parameter
    reference to the value to be set. */
) : void

```

**Function Description**

Copies an Unsigned Short value to the data area of the coordinator. This service is also used to fill sub areas of a datum with values (elements of Arrays, Sequences, Structures, Union).

**Return Value**

None



## **Annex C**

(normative)

### **Programmer's reference guide — Full Access Interface**

#### **C.1 General Information**

Single interfaces (respectively classes and their methods and attributes) of the Full Access Interface, marked with «F», do not belong to the Smart or Extended Access Interface. Therefore all single interfaces are marked with «F». All single interfaces of the Smart Access Interface described in Annex A and of the Extended Access Interface described in Annex B shall be contained in Full Access Interface and are not repeated in this annex. The following detailed description is completed by Figures C.1 to C.29 for an overview on classes and their methods

#### **C.2 Detailed description of Full Access Interface**

##### **C.2.1 «F» GdiCoArrayCD**

This interface contains services for the handling of the type description of an Array. This object reference is valid, till the creation or referencing the next sub description.

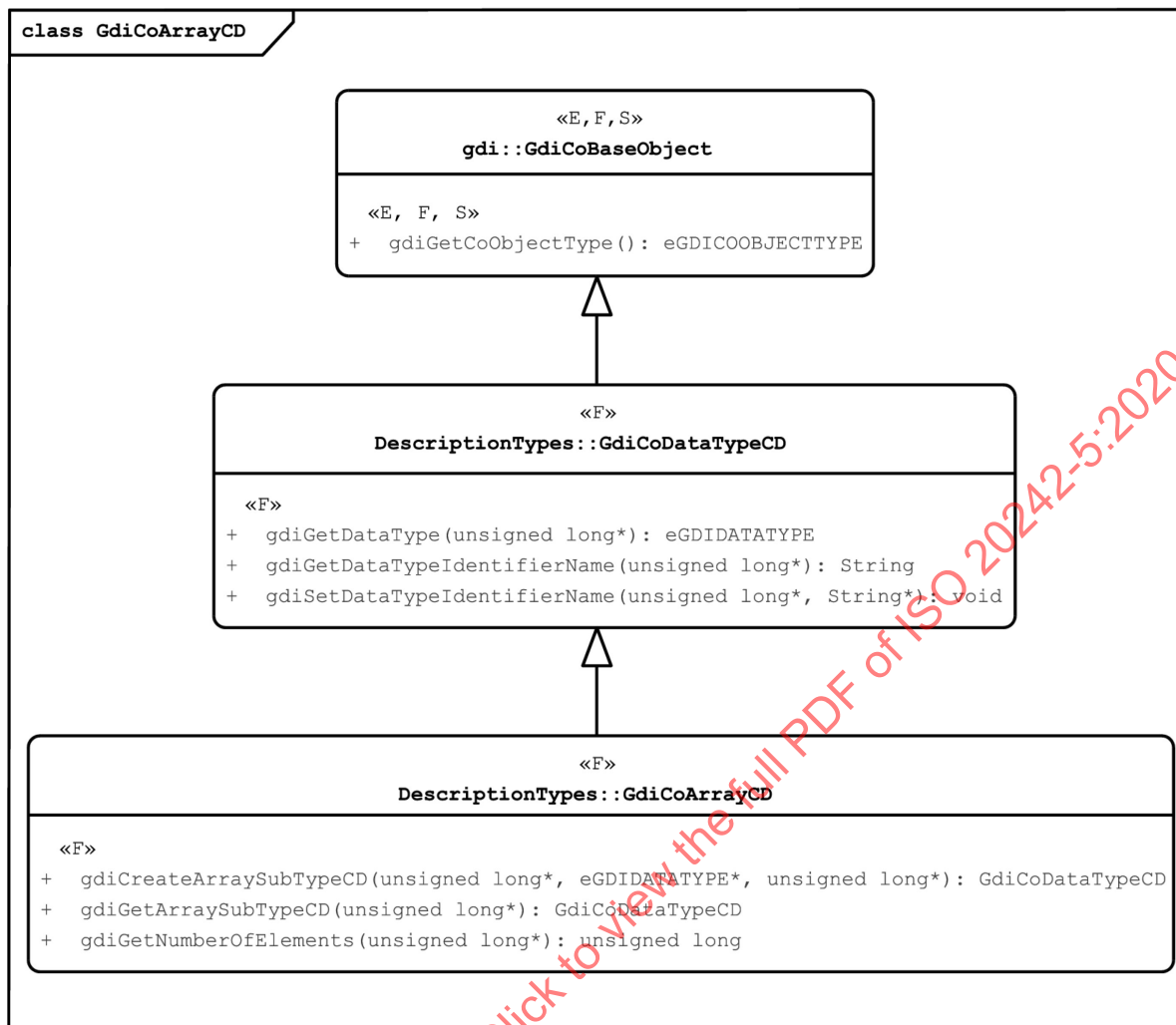


Figure C.1 — Hierarchical diagram of GdiCoArrayCD

### C.2.1.1 GdiCoArrayCD :: «F» gdiCreateArraySubTypeCD()

#### Function Call

```

gdiCreateArraySubTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                eGDIDATATYPE eNewType
    /* in parameter
    GDI type of the Array elements
    The value range includes both ASAM Datatype enumeration and GDI Type
    enumeration. */
    inout                unsigned long ulNumberOfElements
    /* Size of the Array in elements */
) : GdiCoDataTypeCD
  
```

Function Description

Creates the subtype description of the array.

Describes the type and the number of elements of an Array. The element type may be complex. By means of the return value the element type may be specified in more detail. If the basic type of the element is not allowed, a NIL reference is handed over (e.g. if the Array itself is part of a Union, the element type of the Array must not be a Sequence).

Return Value

Returns the reference to GdiCoDataTypeCD if successful. Otherwise NIL reference.

**C.2.1.2 GdiCoArrayCD :: «F» gdiGetArraySubTypeCD()**Function Call

```
gdiGetArraySubTypeCD
(
  inout          unsigned long ulAppHnd
  /*            in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD
```

Function Description

Returns the reference to the description of the sub data type.

Return Value

Returns a reference to the Description of the sub type if present, otherwise a NIL reference.

**C.2.1.3 GdiCoArrayCD :: «F» gdiGetNumberOfElements()**Function Call

```
gdiGetNumberOfElements
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
) : unsigned long
```

Function Description

It is used for the detection of the number of elements of an Array.

Return Value

Returns the number of elements of the Array .

**C.2.2 «F» GdiCoBooleanCD**

Contains services concerning the type Boolean for the definition of restrictions.

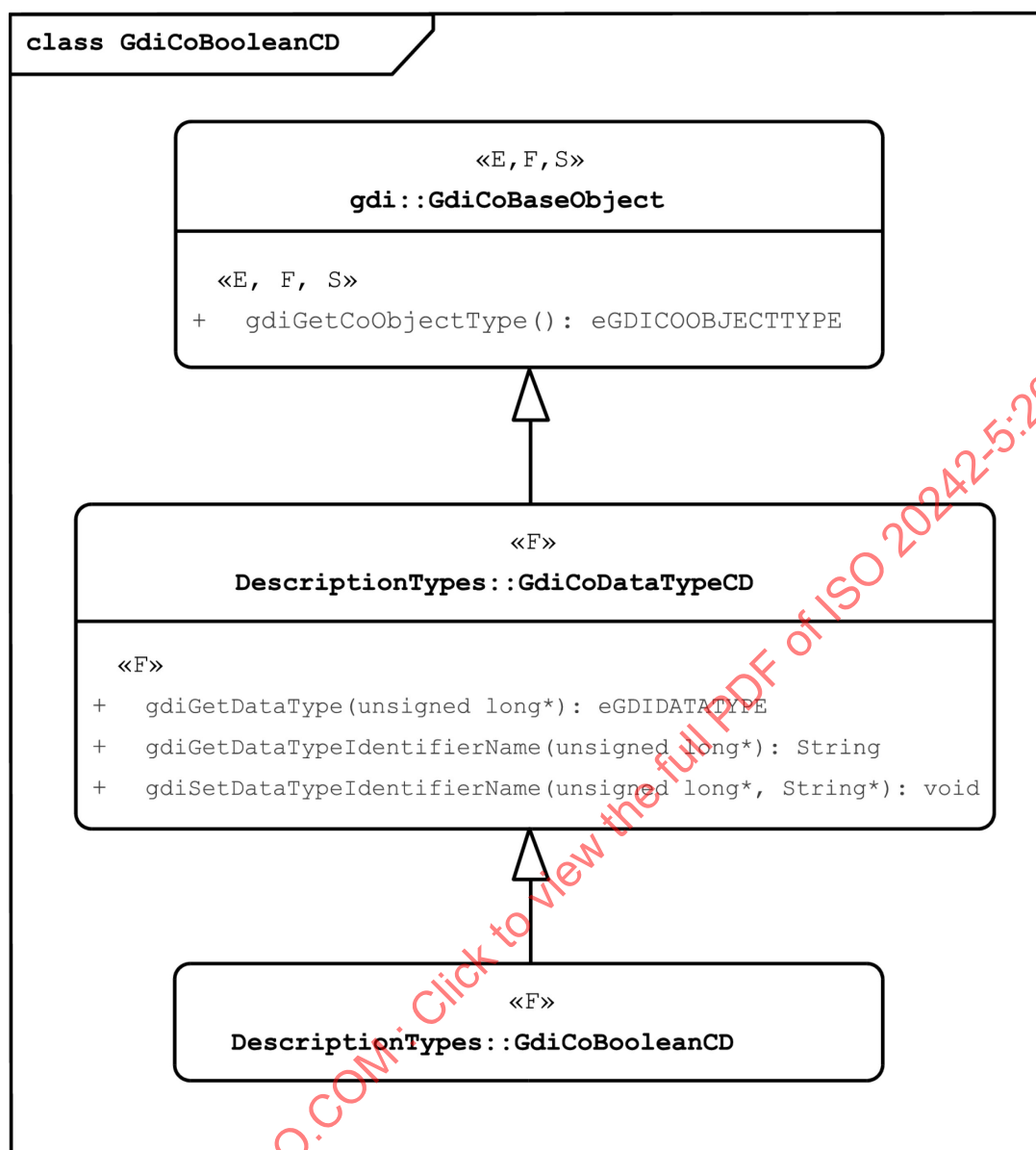


Figure C.2 — Hierarchical diagram of GdiCoBooleanCD

### C.2.3 «F» GdiCoCharCD

Contains services concerning the type Char for the definition of restrictions.

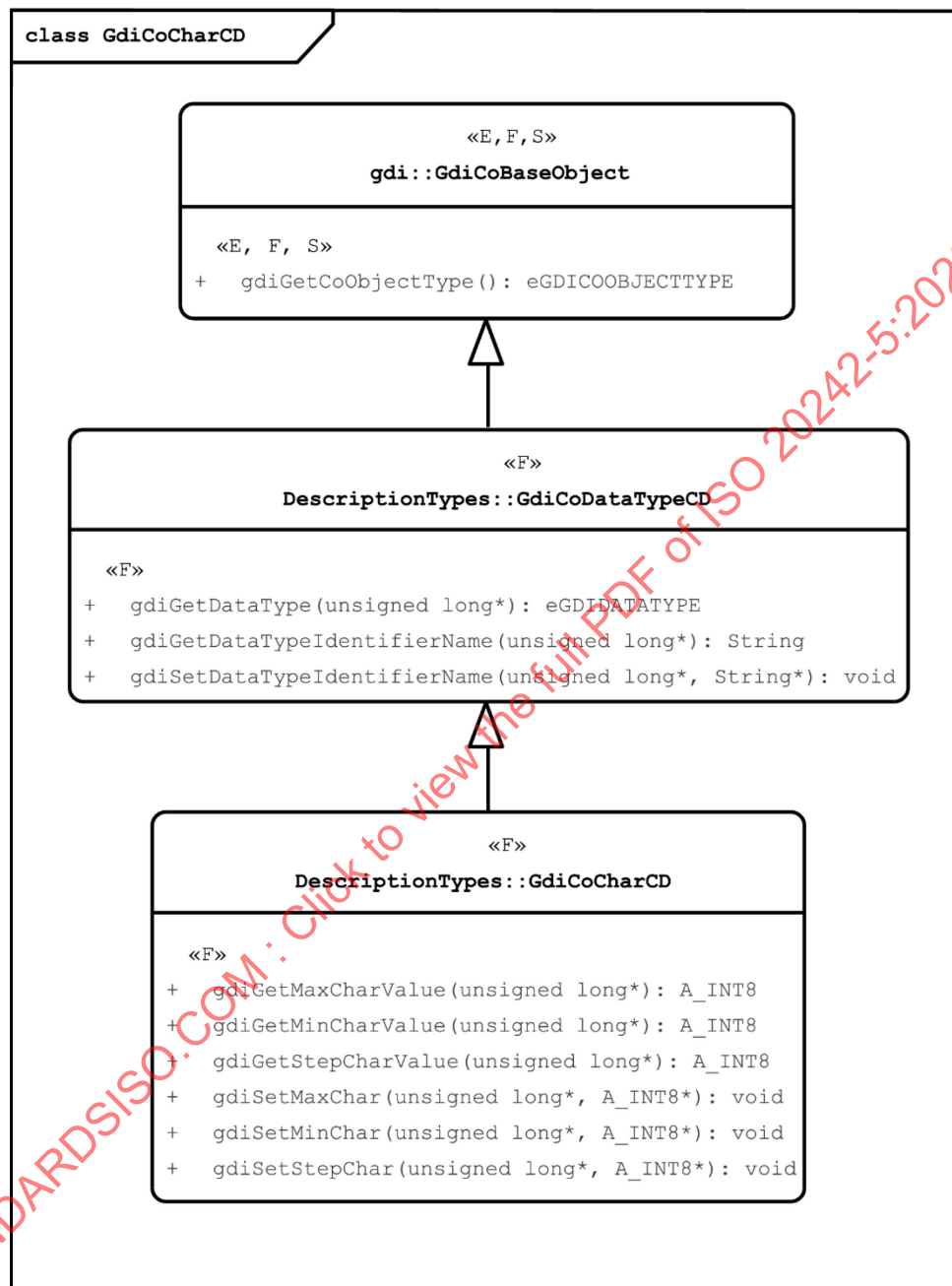


Figure C.3 — Hierarchical diagram of GdiCoCharCD

### C.2.3.1 GdiCoCharCD :: «F» gdiGetMaxCharValue()

#### Function Call

```

gdiGetMaxCharValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */

```

) : A\_INT8

#### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 127.

#### Return Value

Returns the maximum value of a datum, based upon this type.

### **C.2.3.2 GdiCoCharCD :: «F» gdiGetMinCharValue()**

#### Function Call

```
gdiGetMinCharValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT8
```

#### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value -128.

#### Return Value

Returns the minimum value of a datum, based upon this type.

### **C.2.3.3 GdiCoCharCD :: «F» gdiGetStepCharValue()**

#### Function Call

```
gdiGetStepCharValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT8
```

#### Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

#### Return Value

Returns the step width of a datum, based upon this type.

### **C.2.3.4 GdiCoCharCD :: «F» gdiSetMaxChar()**

#### Function Call

```
gdiSetMaxChar
(
```

```

    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT8 cLimitValue
    /* in parameter
    new maximum value (< 128) */
    ) : void

```

### Function Description

Defines the maximum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.3.5 GdiCoCharCD :: «F» gdiSetMinChar()**

### Function Call

```

gdiSetMinChar
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT8 cLimitValue
    /* new minimum value (> -129) */
    ) : void

```

### Function Description

Defines the minimum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.3.6 GdiCoCharCD :: «F» gdiSetStepChar()**

### Function Call

```

gdiSetStepChar
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT8 cLimitValue
    /* in parameter
    new step width (0 < LimitValue < 127) */
    ) : void

```

Function Description

Defines the step width for valid data based upon this type between minimum and maximum. The minimum is the starting value. In case of wrong inputs, empty sets may occur. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

Return Value

None

C.2.4 «F» GdiCoCommObjectCD

This interface describes a communication object. This object reference is valid, till the creation or referencing the next sub description.

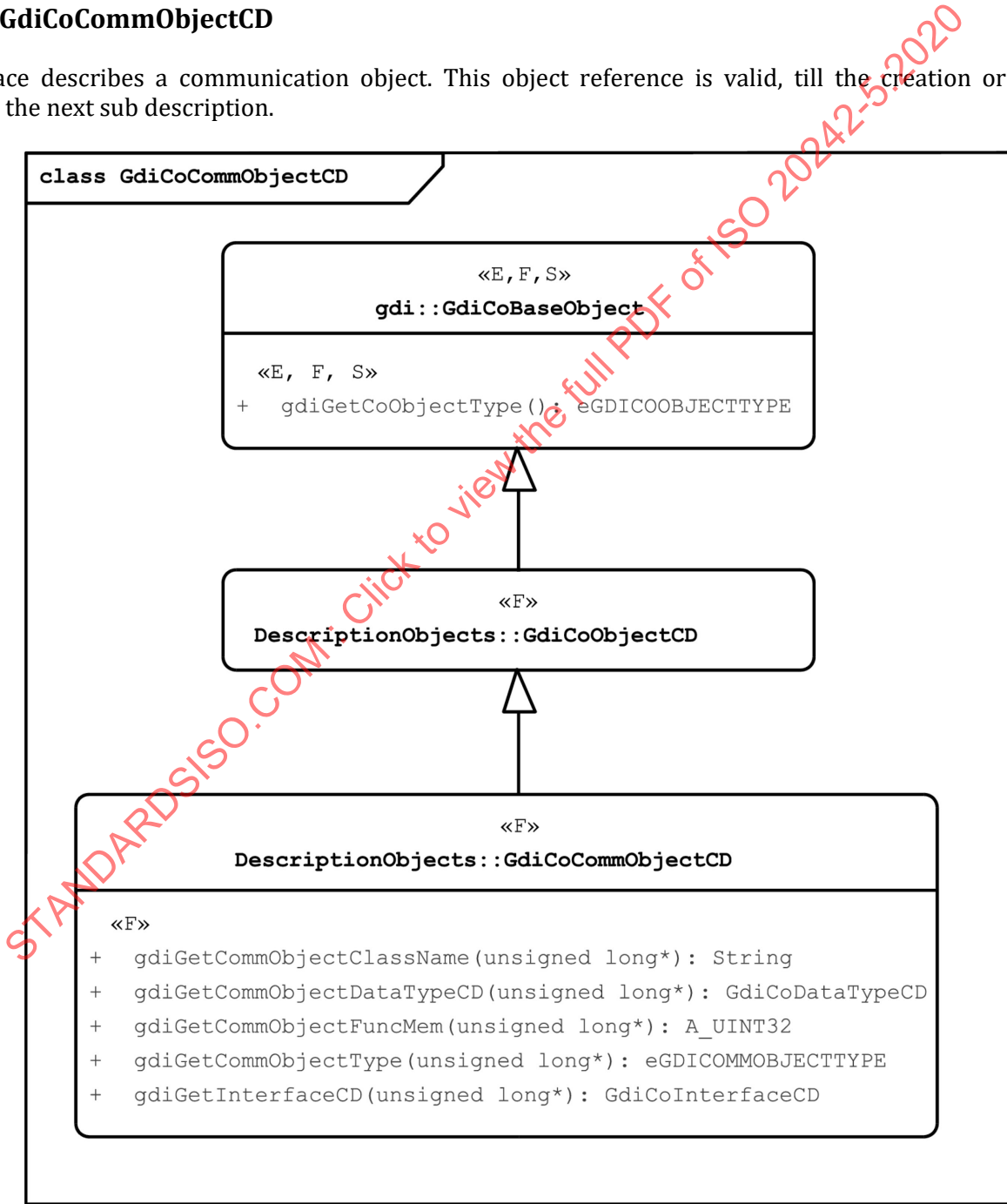


Figure C.4 — Hierarchical diagram of GdiCoCommObjectCD



**C.2.4.1 GdiCoCommObjectCD :: «F» gdiGetCommObjectClassName()****Function Call**

```

gdiGetCommObjectClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String

```

**Function Description**

Returns the class name of the communication object described in this CO description.

In case the CO name is already set because the coordinator handles a PID file, the CO name can be get by calling this method.

**Return Value**

Returns the class name of the communication object, described in the module description.

**C.2.4.2 GdiCoCommObjectCD :: «F» gdiGetCommObjectDataTypeCD()****Function Call**

```

gdiGetCommObjectDataTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD

```

**Function Description**

Returns the type description of the communication object.

In case the type description is already set because the coordinator handles a PID file, the type description can be get by calling this method.

**Return Value**

Returns the type description of the communication object.

**C.2.4.3 GdiCoCommObjectCD :: «F» gdiGetCommObjectFuncMem()****Function Call**

```

gdiGetCommObjectFuncMem
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32

```

### Function Description

Returns the absolute index of the communication object described in the communication object description.

In case the absolute index is already set because the coordinator handles a PID file, the absolute index can be get by calling this method.

### Return Value

Returns the absolute index (FuncMem) described in this communication object description.

## **C.2.4.4 GdiCoCommObjectCD :: «F» gdiGetCommObjectType()**

### Function Call

```
gdiGetCommObjectType
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : eGDICOMMOBJECTTYPE
```

### Function Description

Returns the type of the communication object for this description object.

### Return Value

Returns the type of the communication object.

## **C.2.4.5 GdiCoCommObjectCD :: «F» gdiGetInterfaceCD()**

### Function Call

```
gdiGetInterfaceCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoInterfaceCD
```

### Function Description

Creates a reference to the upper GdiCoFuncObjectCD.

### Return Value

Returns GdiCoCommObjectCD.

## **C.2.5 «F» GdiCoDataTypeCD**

This interface is the base type of all datatype descriptions.

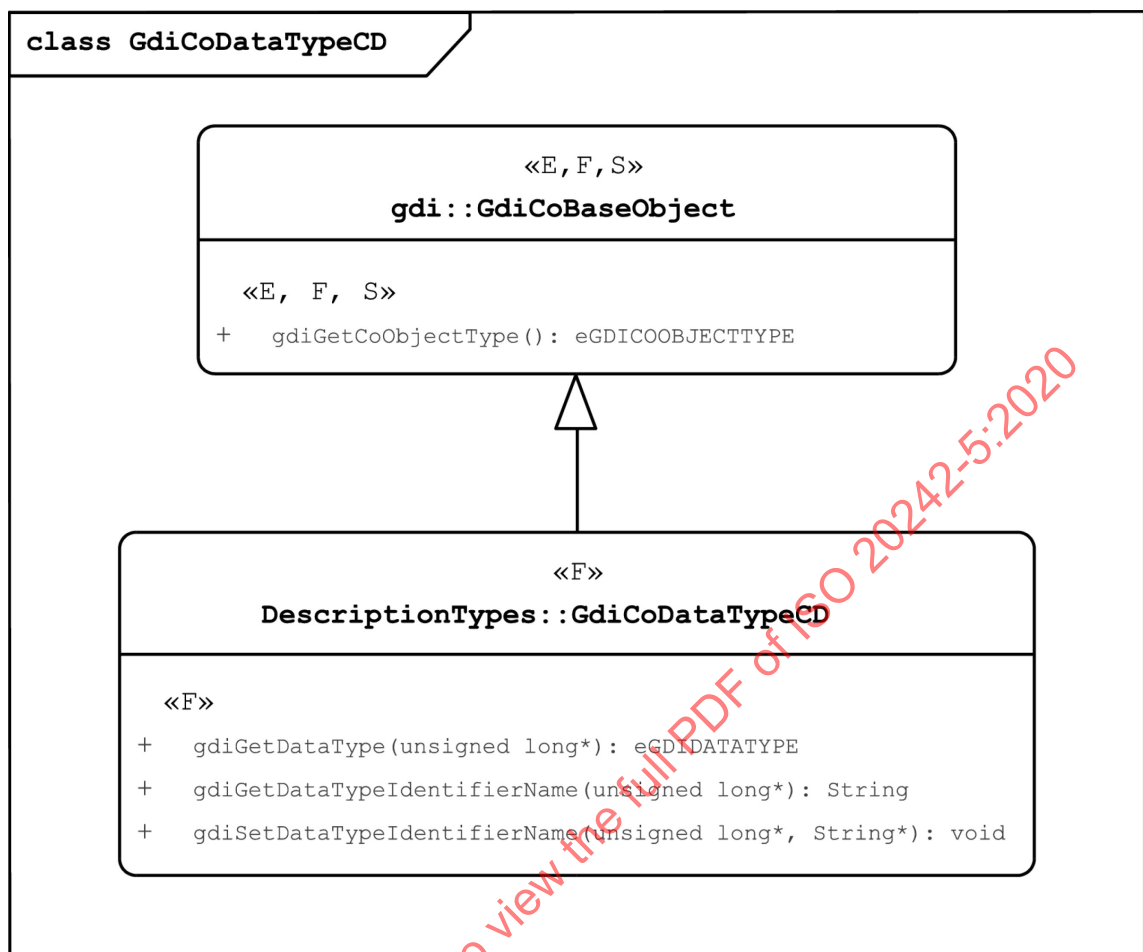


Figure C.5 — Hierarchical diagram of GdiCoDataTypeCD

### C.2.5.1 GdiCoDataTypeCD::«F» gdiGetDataType()

#### Function Call

```

gdiGetDataType
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : eGDIDATATYPE
  
```

#### Function Description

Returns the type as numeric value (enum).

#### Return Value

Returns the type as numeric value (described in Semantics).

### C.2.5.2 GdiCoDataTypeCD :: «F» gdiGetDataTypeIdentifierName()

#### Function Call

```
gdiGetDataTypeIdentifierName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String
```

#### Function Description

Returns the name of the datatype described in this datatype description.

In case the name is already set because the coordinator handles a PID file, the datatype name can be get by calling this method.

#### Return Value

Returns the datatype name, described in the datatype description.

In case no name is set an empty string will be returned.

### C.2.5.3 GdiCoDataTypeCD :: «F» gdiSetDataTypeIdentifierName()

#### Function Call

```
gdiSetDataTypeIdentifierName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sName
    /* contains the name of the described data type. */
) : void
```

#### Function Description

Set the name of the datatype for this description object.

#### Return Value

None

### C.2.6 «F» GdiCoDoubleCD

Contains services concerning the type Double for the definition of restrictions.

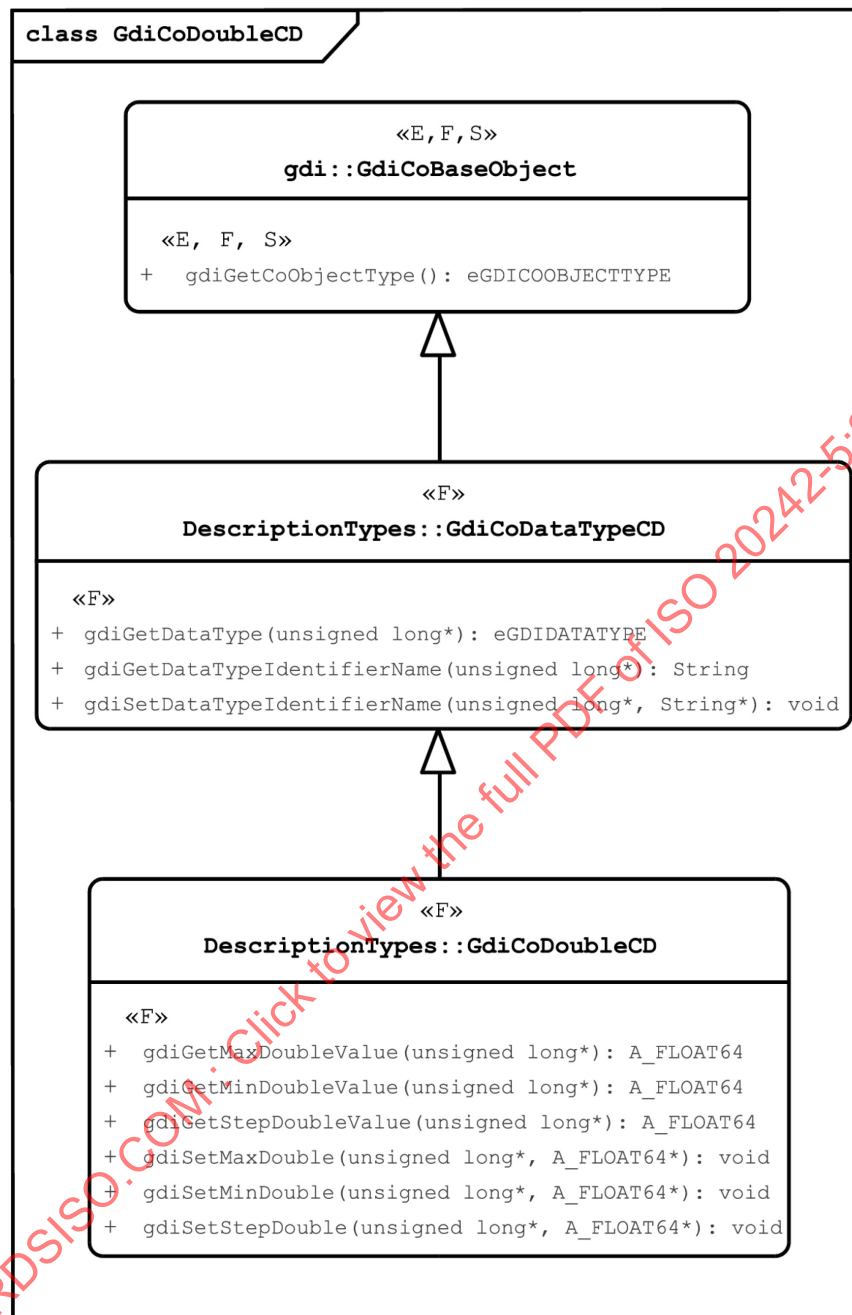


Figure C.6 — Hierarchical diagram of GdiCoDoubleCD

### C.2.6.1 GdiCoDoubleCD :: «F» gdiGetMaxDoubleValue()

#### Function Call

```

gdiGetMaxDoubleValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT64
  
```

### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 1,7976931348623158 E + 308.

### Return Value

Returns the maximum value of a datum, based upon this type.

## **C.2.6.2 GdiCoDoubleCD :: «F» gdiGetMinDoubleValue()**

### Function Call

```
gdiGetMinDoubleValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT64
```

### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 2,2250738585072014 E – 308.

### Return Value

Returns the minimum value of a datum, based upon this type.

## **C.2.6.3 GdiCoDoubleCD :: «F» gdiGetStepDoubleValue()**

### Function Call

```
gdiGetStepDoubleValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT64
```

### Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

### Return Value

Returns the step width of a datum, based upon this type.

## **C.2.6.4 GdiCoDoubleCD :: «F» gdiSetMaxDouble()**

### Function Call

```
gdiSetMaxDouble
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          A_FLOAT64 dLimitValue
/* in parameter
new maximum value (< 1,7976931348623158 E + 309) */
) : void

```

### Function Description

Defines the maximum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.6.5 GdiCoDoubleCD :: «F» gdiSetMinDouble()**

### Function Call

```

gdiSetMinDouble
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_FLOAT64 dLimitValue
  /* in parameter
  new minimum value (> 2,2250738585072014 E -309) */
) : void

```

### Function Description

Defines the minimum value for a datum based upon this type. The value has no effects on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.6.6 GdiCoDoubleCD :: «F» gdiSetStepDouble()**

### Function Call

```

gdiSetStepDouble
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_FLOAT64 dLimitValue
  /* in parameter
  new step width (0 < LimitValue < 1,7976931348623158 E + 308) */
) : void

```

Function Description

Defines the step width for valid data based upon this type between minimum and maximum. The minimum is the starting value. In case of wrong inputs, empty sets may occur. The value has no effects on the communication to the device driver. It is read by certain applications to obtain detailed type information.

Return Value

None

C.2.7 «F» GdiCoDriverCD

Describes the GDI Driver Object. This object reference is valid, till the creation or referencing the next sub description.

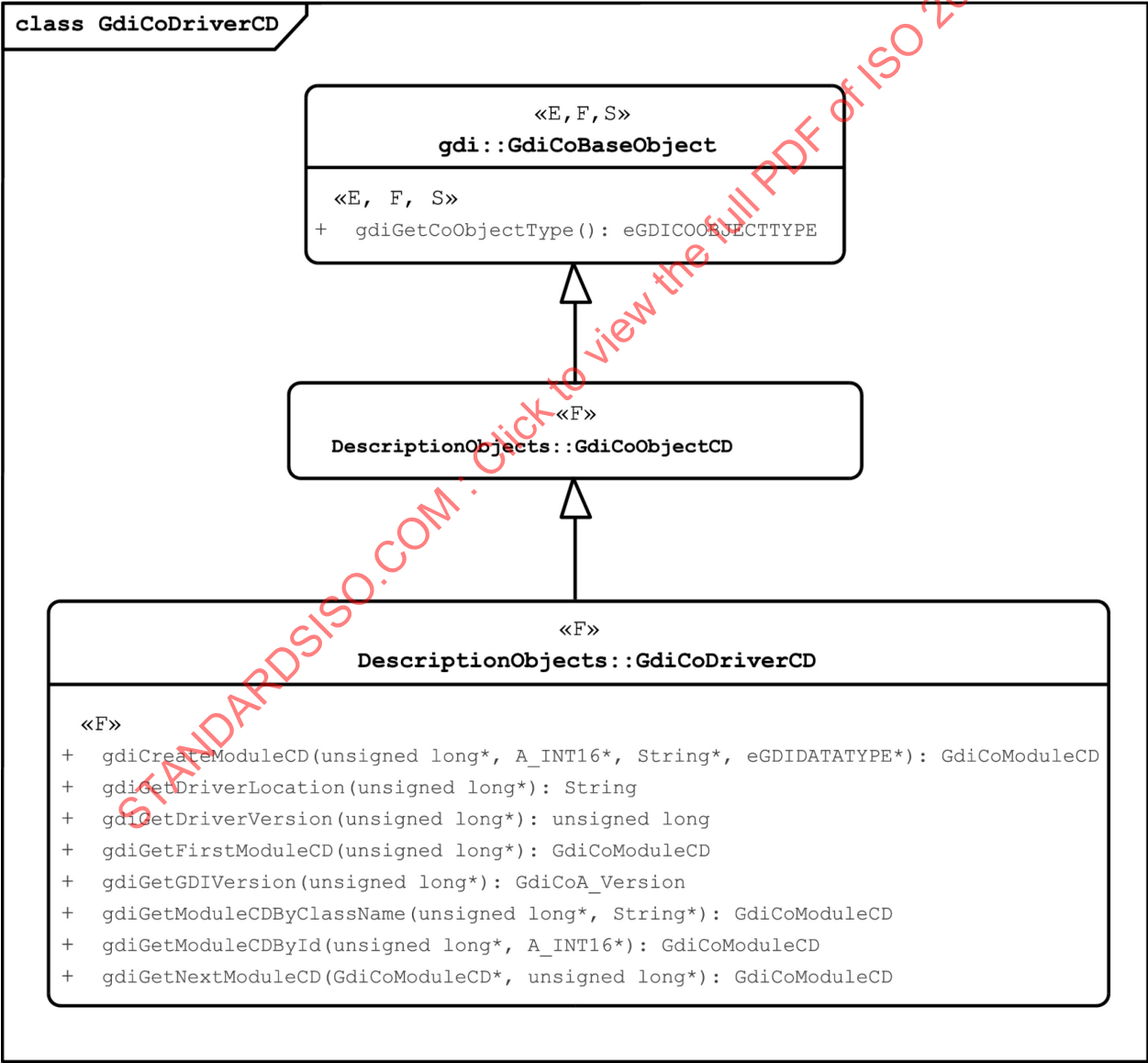


Figure C.7 — Hierarchical diagram of GdiCoDriverCD



**C.2.7.1 GdiCoDriverCD :: «F» gdiCreateModuleCD()****Function Call**

```

gdiCreateModuleCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT16 nModuleId
    /* in parameter
    contains the module ID */
    inout                String sModuleClassName
    /* in parameter
    contains the class name of the module. */
    inout                eGDIDATATYPE eCreateParamType
    /* in parameter
    contains the type of the create parameter.
    eDT_NO_DEFINITION if not necessary. */
) : GdiCoModuleCD

```

**Function Description**

Returns a new GdiCoModuleCD object.

**Return Value**

Returns a new GdiCoModuleCD object.

**C.2.7.2 GdiCoDriverCD :: «F» gdiGetDriverLocation()****Function Call**

```

gdiGetDriverLocation
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String

```

**Function Description**

Returns the location specification of the driver described in the driver description.

In case the location specification of the driver is already set because the coordinator handles a PID file, the location specification can be get by calling this method.

**Return Value**

Returns the location specification of the driver described in the driver description.

**C.2.7.3 GdiCoDriverCD :: «F» gdiGetDriverVersion()****Function Call**

```

gdiGetDriverVersion
(
    inout                unsigned long ulAppHnd

```

```
/* in parameter  
Identifier of the application returned by gdiCreateWorkspace(...) or by  
gdiGetWorkspaceByName() */  
) : unsigned long
```

#### Function Description

Returns the driver version described in the driver description.

In case the driver version is already set because the coordinator handles a PID file, the driver version can be get by calling this method.

#### Return Value

Returns the driver version described in the Driver Description.

### **C.2.7.4 GdiCoDriverCD :: «F» gdiGetFirstModuleCD()**

#### Function Call

```
gdiGetFirstModuleCD  
(  
    inout                unsigned long ulAppHnd  
/* in parameter  
Identifier of the application returned by gdiCreateWorkspace(...) or by  
gdiGetWorkspaceByName() */  
) : GdiCoModuleCD
```

#### Function Description

Creates reference to the first Module description within the driver description if it exists.

#### Return Value

Returns GdiCoModuleCD if it exists.

### **C.2.7.5 GdiCoDriverCD :: «F» gdiGetGDIVersion()**

#### Function Call

```
gdiGetGDIVersion  
(  
    inout                unsigned long ulAppHnd  
/* in parameter  
Identifier of the application returned by gdiCreateWorkspace(...) or by  
gdiGetWorkspaceByName() */  
) : GdiCoA_Version
```

#### Function Description

Returns the GDI version described in the Driver Description.

In case the GDI version is already set because the coordinator handles a PID file, the GDI version can be get by calling this method.

#### Return Value

Returns the GDI version described in the Driver Description.

**C.2.7.6 GdiCoDriverCD :: «F» gdiGetModuleCDByClassName()****Function Call**

```

gdiGetModuleCDByClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sModuleName
    /* in parameter
    contains the module name. */
) : GdiCoModuleCD

```

**Function Description**

Returns the module description corresponding to the given model name.

**Return Value**

Returns the module description corresponding to the given module name.

**C.2.7.7 GdiCoDriverCD :: «F» gdiGetModuleCDBById()****Function Call**

```

gdiGetModuleCDBById
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT16 nModuleId
    /* in parameter
    contains the module ID. */
) : GdiCoModuleCD

```

**Function Description**

Returns the module description corresponding to the given modul ID.

**Return Value**

Returns the module description corresponding to the given model ID.

**C.2.7.8 GdiCoDriverCD :: «D,F» gdiGetNextModuleCD()****Function Call**

```

gdiGetNextModuleCD
(
    inout                GdiCoModuleCD refLastElement
    /* in parameter
    reference to the last Module description. */
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoModuleCD

```

Function Description

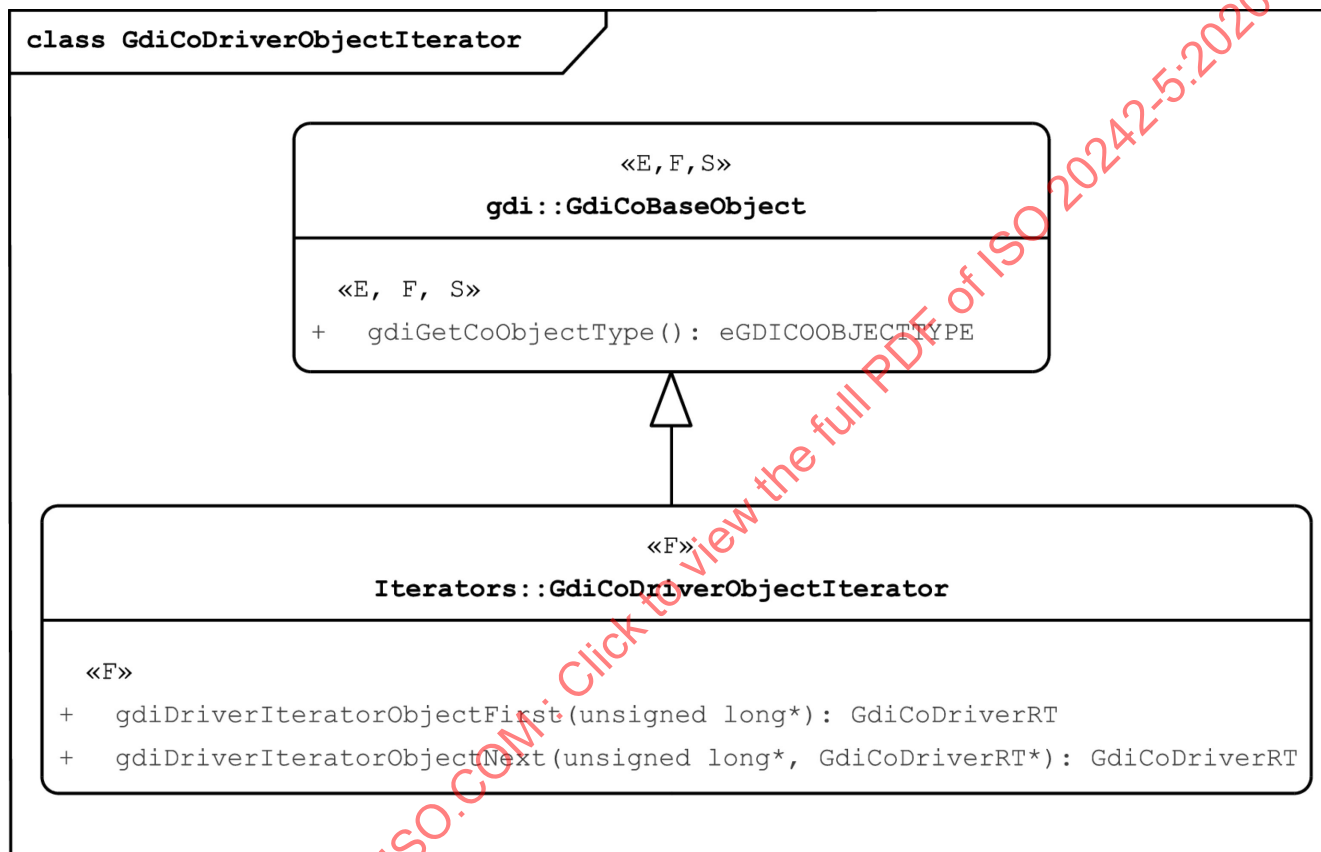
Returns the reference to the next Module description within the Driver description.

Return Value

Returns GdiCoModuleCD if it exists.

**C.2.8 «F» GdiCoDriverObjectIterator**

This interface contains the service for the application of a Driver iterator, sequential output of all Driver objects which have been set up within a Workspace.



**Figure C.8 — Hierarchical diagram of GdiCoDriverObjectIterator**

**C.2.8.1 GdiCoDriverObjectIterator :: «F» gdiDriverIteratorObjectFirst()**Function Call

```

gdiDriverIteratorObjectFirst
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDriverRT
  
```

Function Description

Sets the internal pointer to the first Driver Object within the Workspace.

Return Value

Returns the reference to a coordinator Driver Object (Administration unit within the coordinator) if it exists, otherwise NIL.

**C.2.8.2 GdiCoDriverObjectIterator :: «F» gdiDriverIteratorObjectNext()**Function Call

```
gdiDriverIteratorObjectNext
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoDriverRT refLastElement
    /* in parameter
    A reference to a element returned by gdiDriverObjectFirst or
    gdiDriverObjectNext. */
) : GdiCoDriverRT
```

Function Description

Detects the reference to the current DriverRT Object (Parameter) within the workspace, sets the internal pointer to the next DriverRT Object (if existing) and returns the reference.

Return Value

Returns the reference to a coordinator Driver Object (Administration unit within the coordinator) if it exists, NIL otherwise.

**C.2.9 «F» GdiCoDriverRT**

This interface contains services for the setting up and deleting of Control VDs

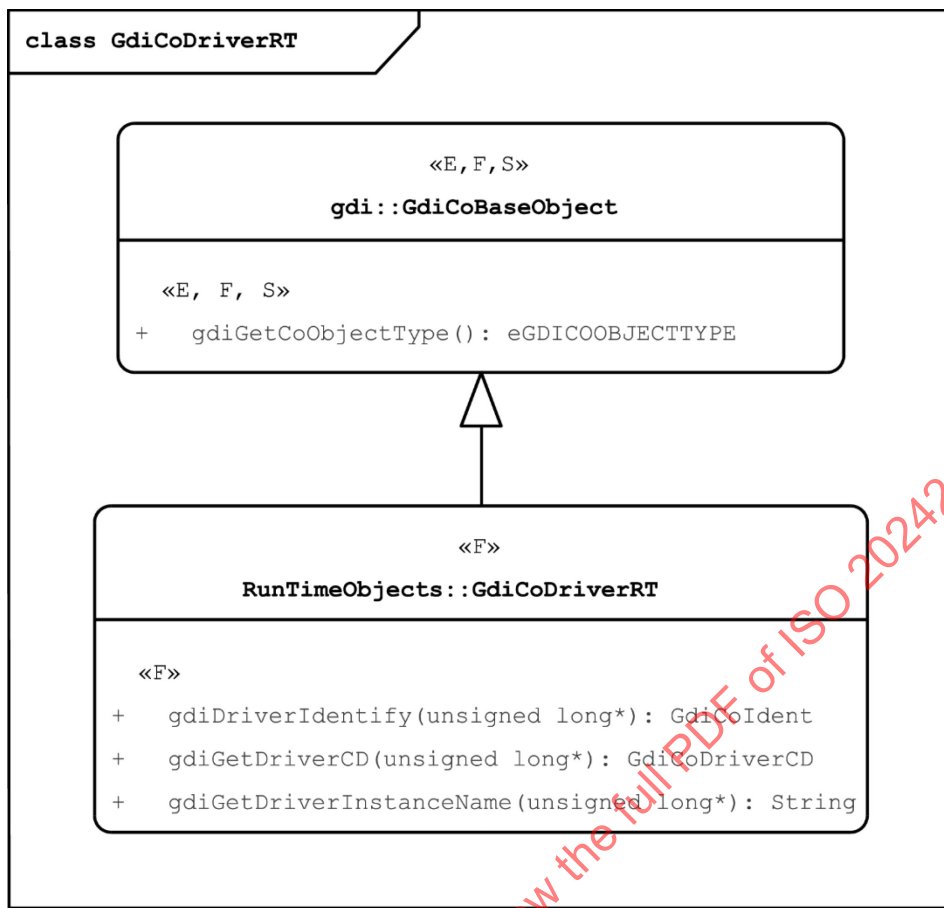


Figure C.9 — Hierarchical diagram of GdiCoDriverRT

### C.2.9.1 GdiCoDriverRT :: «F» gdiDriverIdentify()

#### Function Call

```

gdiDriverIdentify
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoIdent
  
```

#### Function Description

Effects the execution of Identify calls to the device driver. The detected data corresponds with the identification data of the device driver.

#### Return Value

Returns GDIIDENT data according to GDI specification.

**C.2.9.2 GdiCoDriverRT :: «F» gdiGetDriverCD()****Function Call**

```

gdiGetDriverCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDriverCD

```

**Function Description**

Generates a Reference to the Description of the Driver as GdiCoDriverCD object.

**Return Value**

Returns the reference to GdiCoDriverCD.

**C.2.9.3 GdiCoDriverRT :: «F» gdiGetDriverInstanceName()****Function Call**

```

gdiGetDriverInstanceName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String

```

**Function Description**

Returns the instance name of the Driver object defined during parameterization.

Shall be unique inside a workspace.

**Return Value**

Returns the Driver instance name as String.

**C.2.10 «F» GdiCoEnumCD**

Contains services for the definition of enumeration elements.

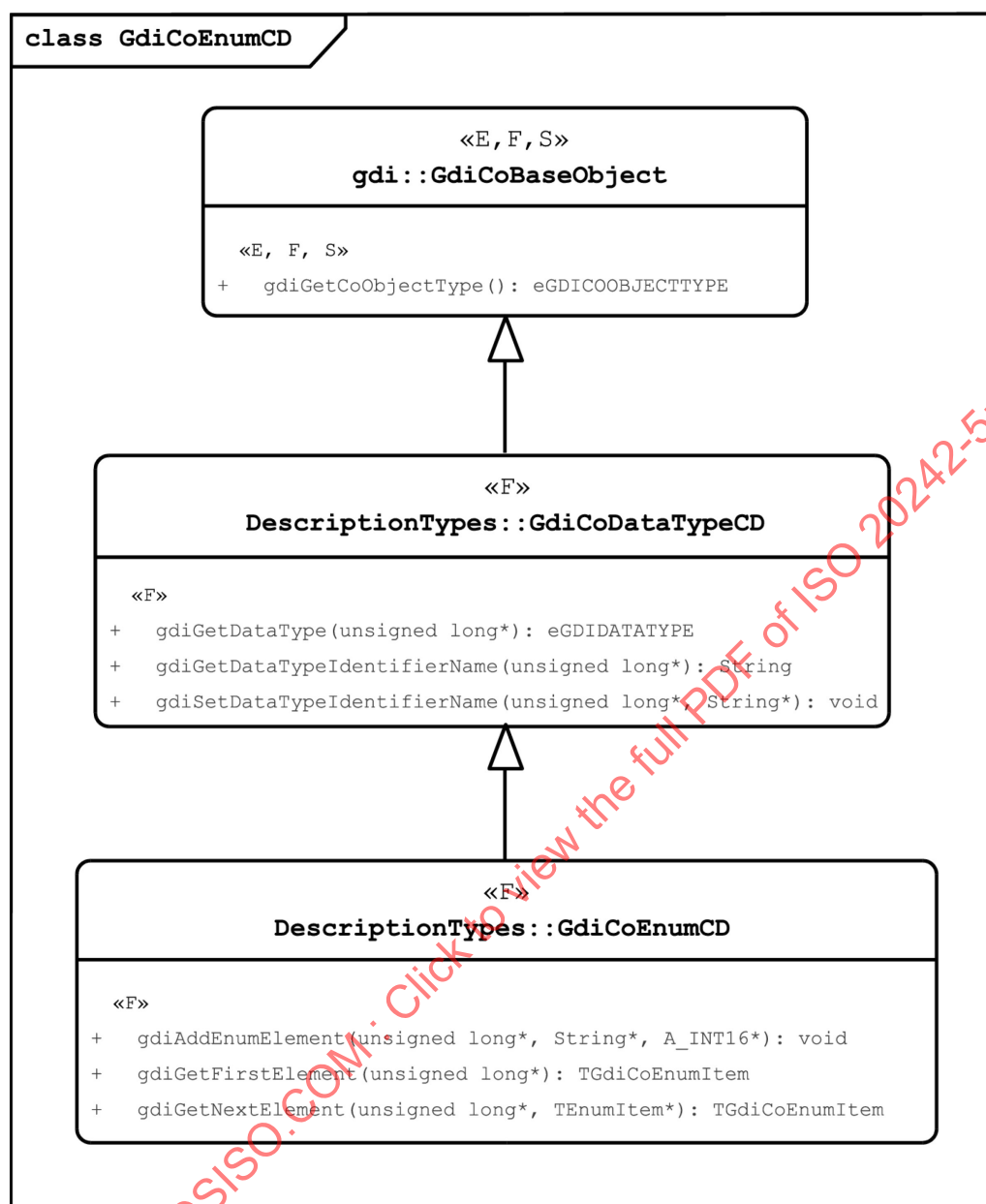


Figure C.10 — Hierarchical diagram of GdiCoEnumCD

### C.2.10.1 GdiCoEnumCD :: «F» gdiAddEnumElement()

#### Function Call

```

gdiAddEnumElement
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout          String sName
    /* in parameter
    Identifier of the enumeration element. */

```



```

    inout                A_INT16 nValue
    /* in parameter
    numeric value of the enumeration element */
    ) : void

```

### Function Description

Adds new elements to the enumeration. Overwriting is permitted. The key value is the name.

### Return Value

None

## **C.2.10.2 GdiCoEnumCD :: «F» gdiGetFirstElement()**

### Function Call

```

gdiGetFirstElement
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    ) : TGdiCoEnumItem

```

### Function Description

Returns the first element of an enumeration of single elements. Usually this is the element with the lowest value. The return element is a structure which consists of the sub elements String and short.

### Return Value

Returns the structure consisting of the sub elements String and short. If no element is available, the value "noGDIElement" as name and the numeric value 32768 is returned.

## **C.2.10.3 GdiCoEnumCD :: «F» gdiGetNextElement()**

### Function Call

```

gdiGetNextElement
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                TEnumItem refLastElement
    /* in parameter
    contains the reference to the last Enumeration Description. */
    ) : TGdiCoEnumItem

```

### Function Description

Returns the next element of an enumeration of single elements. The return element is a structure which consists of the sub elements String and short.

### Return Value

If no element is available, the value "noGDIElement" as name and the numeric value 32768 is returned.

## C.2.11 «F» GdiCoFactory

This interface is a static class for creating and deleting GDI Objects.

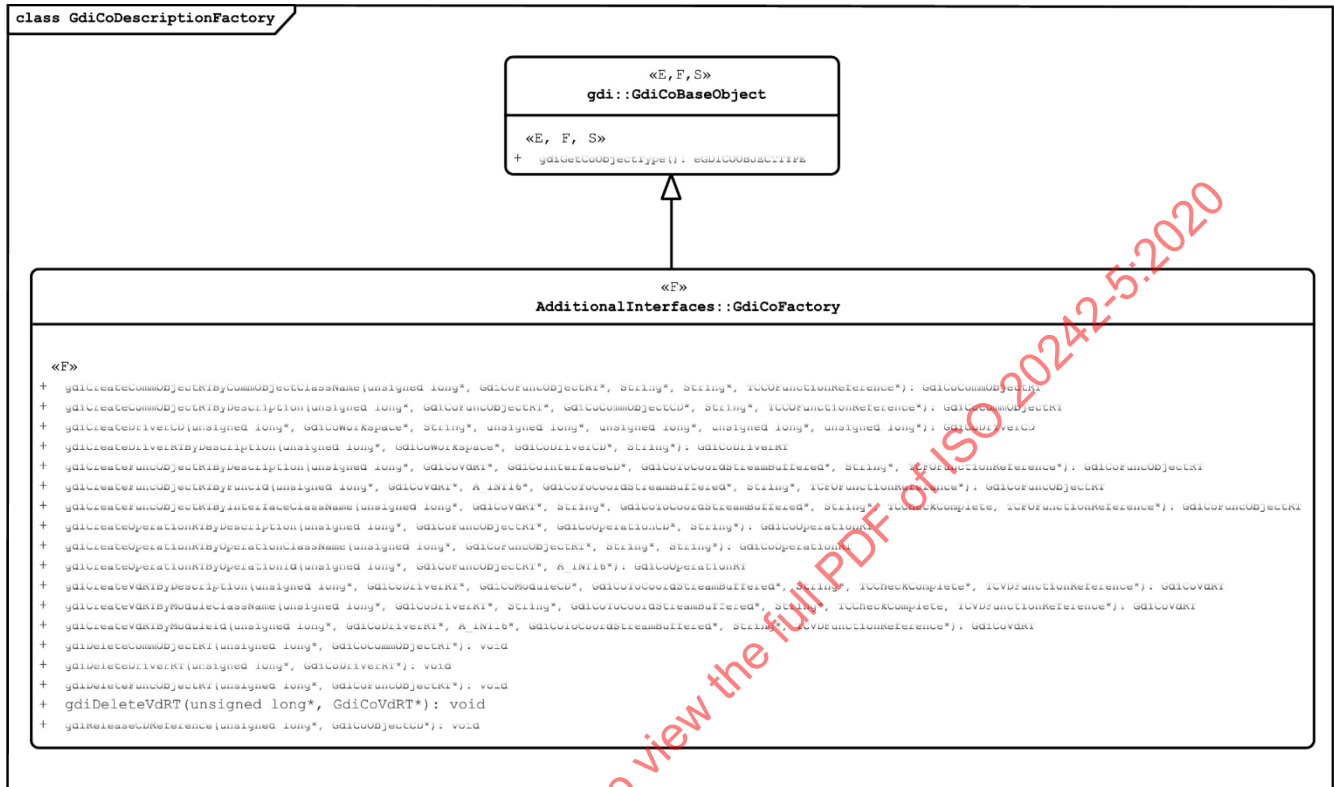


Figure C.11 — Hierarchical diagram of GdiCoFactory

### C.2.11.1 GdiCoFactory :: «F» gdiCreateCommObjectRTByCommObjectClassName()

#### Function Call

```

gdiCreateCommObjectRTByCommObjectClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoFuncObjectRT refFuncObjectRT
    /* in parameter
    Contains the reference to the function object. The Communication Object
    will be connected to this FO. */
    inout                String sCommObjectClassName
    /* in parameter
    Communication Object Class name (DCD) of the Communication Object. */
    inout                String sCommObjectInstanceName
    /* in parameter
    instance name of the new created Communication Object. */
    inout                TCCOFFunctionReference
                        refCBCompleteCreateCommObject
    /* in parameter

```

```

Reference to a call back method which handles the asynchronous
initialization mechanism. A NIL reference signals a synchronous call.
*/
) : GdiCoCommObjectRT

```

### Function Description

Effects the creation of a Communication object within the Coordinator and as instance within the device driver based on the communication object description defined by the given class (DCD) name.

In case of an asynchronous operation, the reference to the Communication Object will be delivered by the complete call back. If no call back reference is given, the operation will be executed synchronously.

### Return Value

Returns the reference to the new created GdiCoCommObjectRT Instance as GdiCoAttributeRT or GdiCoParameterRT.

NIL in case of an asynchronous operation.

## **C.2.11.2 GdiCoFactory :: «F» gdiCreateCommObjectRTByDescription()**

### Function Call

```

gdiCreateCommObjectRTByDescription
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoFuncObjectRT refFuncObjectRT
    /* in parameter
    Contains the reference to the function object. The Communication Object
    will be connected to this FO. */
    inout                GdiCoCommObjectCD refCommObjectCD
    /* in parameter
    reference to the GdiCoCommObjectCD which describes the Communication
    Object. */
    inout                String sCommObjectInstanceName
    /* in parameter
    InstanceName of the communication object. */
    inout                TCCOFunctionReference
                        refCBCompleteCreateCommObject
    /* in parameter
    Reference to a call back method which handles the asynchronous
    initialization mechanism. A NIL reference signals a synchronous call.
    */
) : GdiCoCommObjectRT

```

### Function Description

Effects the creation of a Communication object within the Coordinator and as instance within the device driver based on the given communication object description.

In case of an asynchronous operation, the reference to the Communication Object will be delivered by the complete call back. If no call back reference is given, the operation will be executed synchronously.

Return Value

Returns the reference to the new created GdiCoCommObjectRT as GdiCoAttributeRT or GdiCoParameterRT.

NIL in case of an asynchronus operation.

**C.2.11.3 GdiCoFactory :: «F» gdiCreateDriverCD()**Function Call

```
gdiCreateDriverCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoWorkspace refWorkspace
    /* in parameter
    contains the reference to the workspace. The driver will be connected
    to this workspace. */
    inout                String sLocation
    /* in parameter
    contains the location specification of the driver. */
    inout                unsigned long ulDriverVersion
    /* in parameter
    Contains the manufacturer driver version. */
    inout                unsigned long ulGdiVersionMajor
    /* in parameter
    Contains the GDI major version. */
    inout                unsigned long ulGdiVersionMinor
    /* in parameter
    Contains the GDI minor version. */
    inout                unsigned long ulGdiVersionRevision
    /* in parameter
    Contains the GDI revision number. */
) : GdiCoDriverCD
```

Function Description

Returns a new GdiCoDriverCD object.

Return Value

Returns a new GdiDriverCD object.

**C.2.11.4 GdiCoFactory :: «F» gdiCreateDriverRTByDescription()**Function Call

```
gdiCreateDriverRTByDescription
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...)
    or by gdiGetWorkspaceByName() */
    inout                GdiCoWorkspace refWorkspace
    /* in parameter
```

```

contains the reference to the workspace. The driver will be
connected to this workspace. */
inout          GdiCoDriverCD refDriverCD
/* in parameter
Reference to the Description of the driver. */
inout          String sDriverInstanceName
/* in parameter
contains the Driver instance name. */
) : GdiCoDriverRT

```

### Function Description

Loads a GDI driver and connect the driver with the given Workspace, if not yet loaded to the Coordinator. Otherwise, the administration counter for the driver is incremented.

### Return Value

Returns the reference to a GdiCoDriverRT if successful.

## **C.2.11.5 GdiCoFactory :: «F» gdiCreateFuncObjectRTByDescription()**

### Function Call

```

gdiCreateFuncObjectRTByDescription
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          GdiCoVdRT refVdRT
  /* in parameter
  Contains the reference to the application VD. The function object will
  be connected to this VD. */
  inout          GdiCoInterfaceCD refInterfaceCD
  /* in parameter
  reference to the GdiCoInterfaceCD which describes the function object.
  */
  inout          GdiCoToCoordStreamBuffered refCreateParameter
  /* in parameter
  the given Stream object contains the create parameter of the VD.
  If no create Parameter exists, NIL is handed over. */
  inout          String sFuncObjectInstanceName
  /* in parameter
  Name of the FO instance. */
  inout          TCFOFunctionReference
                  refCBCompleteCreateFuncObject
  /* in parameter
  Reference to a call back method which handles the initialization
  mechanism. A NIL reference signals a synchronous call. */
) : GdiCoFuncObjectRT

```

### Function Description

Effects the setting up of the Control structure of a Function object within the Coordinator. An instance within the driver is not created.

In case of an asynchronous operation, the reference to the Function Object will be delivered by the complete call back. If no call back reference is given, the operation will be executed synchronously.

Return Value

Returns the reference to the new created GdiCoFuncObjectRT.  
NIL in case of an asynchronus operation.

**C.2.11.6 GdiCoFactory :: «F» gdiCreateFuncObjectRTByFuncId()**Function Call

```
gdiCreateFuncObjectRTByFuncId
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoVdRT refVdRT
    /* in parameter
    contains the reference to the VD. The Function Object will be connected
    to this VD. */
    inout                A_INT16 nFuncId
    /* in parameter
    Function Object value, to which the FO shall be created. The FO value
    has to be specified within the DCD of the driver. */
    inout                GdiCoToCoordStreamBuffered refCreateParameter
    /* in parameter
    the given Stream object contains the create parameter of the VD.
    If no create Parameter exists, NIL is handed over. */
    inout                String sFuncObjectInstanceName
    /* in parameter
    Name of the FO instance. */
    inout                TCFOFunctionReference
                        refCBCCompleteCreateFuncObject
    /* in parameter
    Reference to a call back method which handles the initialization
    mechanism. A NIL reference signals a synchronous call. */
) : GdiCoFuncObjectRT
```

Function Description

Effects the setting up of the Control structure of a Function object within the Coordinator. An instance within the driver is not created.

In case of an asynchronus operation, the reference to the Function Object will be delivered by the complete call back. If no call back reference is given, the operation will be executed synchronously.

Return Value

Returns the reference to the new created GdiCoFuncObjectRT.  
NIL in case of an asynchronus operation.

**C.2.11.7 GdiCoFactory :: «F» gdiCreateFuncObjectRTByInterfaceClassName()**Function Call

```
gdiCreateFuncObjectRTByInterfaceClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          GdiCoVdRT refVdRT
/* in parameter
Contains the reference to the application VD. The function object will
be connected to this VD. */
inout          String sInterfaceClassName
/* in parameter
Contains the class name of the FO to create. */
inout          GdiCoToCoordStreamBuffered refCreateParameter
/* in parameter
the given Stream object contains the create parameter of the VD.
If no create Parameter exists, NIL is handed over. */
inout          String sFuncObjectInstanceName
/* in parameter
Name of the FO instance. */
in             TCCheckComplete refCBCheckComplete
/* in parameter
Reference to a call back method which handles the unsolicited report of
the VD state check complete.
If a NIL reference is given, no event will be raised by the coordinator.
*/
inout          TCFOFunctionReference
               refCBCompleteCreateFuncObject
/* in parameter
Reference to a call back method which handles the initialization
mechanism. A NIL reference signals a synchronous call. */
) : GdiCoFuncObjectRT

```

### Function Description

Effects the setting up of the Control structure of a Function object within the Coordinator. An instance within the driver is not created.

In case of an asynchronous operation, the reference to the Function Object will be delivered by the complete call back. If no call back reference is given, the operation will be executed synchronously.

### Return Value

Returns the reference to the new created GdiCoFuncObjectRT.  
NIL in case of an asynchronous operation.

## **C.2.11.8 GdiCoFactory :: «F» gdiCreateOperationRTByDescription()**

### Function Call

```

gdiCreateOperationRTByDescription
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout          GdiCoFuncObjectRT refFuncObjectRT
    /* in parameter
    Contains the reference to the function object. The operation Object will
    be connected to this FO. */
    inout          GdiCoOperationCD refOperationCD
    /* in parameter
    reference to the GdiCoOperationCD which describes the Operation. */
    inout          String sOperationObjectInstanceName

```

```

/* in parameter
instance name (alias) of the new operation instance. */
) : GdiCoOperationRT

```

### Function Description

Effects the creation of an operation within the Coordinator. No interaction with the driver takes place. This service only sets up administration structures. This administration structure does not have to be deleted explicitly, it will be deleted automatically together with the Function object instance.

### Return Value

Returns the reference to the new created GdiCoOperationRT.

## **C.2.11.9 GdiCoFactory :: «F» gdiCreateOperationRTByOperationClassName()**

### Function Call

```

gdiCreateOperationRTByOperationClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
    inout                GdiCoFuncObjectRT refFuncObjectRT
    /* in parameter
Contains the reference to the function object. The operation Object will
be connected to this FO. */
    inout                String sOperationClassName
    /* in parameter
Class name (DCD) of the Operation, which should be created. */
    inout                String sOperationObjectInstanceName
    /* in parameter
Instance name, which is set to the operation instance. */
) : GdiCoOperationRT

```

### Function Description

Effects the creation of an operation within the Coordinator. No interaction with the driver takes place. This service only sets up administration structures. This administration structure does not have to be deleted explicitly, it will be deleted automatically together with the Function object instance.

### Return Value

Returns the reference to the new created GdiCoOperationRT.

## **C.2.11.10 GdiCoFactory :: «F» gdiCreateOperationRTByOperationId()**

### Function Call

```

gdiCreateOperationRTByOperationId
(
    inout                unsigned long ulAppHnd
    /* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
    inout                GdiCoFuncObjectRT refFuncObjectRT
    /* in parameter

```



```

Contains the reference to the function object. The operation Object will
be connected to this FO. */
inout          A_INT16 nOperationId
/* in parameter
contains the operation Id according to the Description (DCD) */
) : GdiCoOperationRT

```

### Function Description

Effects the creation of an operation within the Coordinator. No interaction with the driver takes place. This service only sets up administration structures. This administration structure does not have to be deleted explicitly, it will be deleted automatically together with the Function object instance.

### Return Value

Returns the reference to the new created GdiCoOperationRT.

## **C.2.11.11 GdiCoFactory :: «F» gdiCreateVdRTByDescription()**

### Function Call

```

gdiCreateVdRTByDescription
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          GdiCoDriverRT refDriverRT
  /* in parameter
  contains the reference to the driver. The new Application VD
  (administration structure) will be connected to this driver. */
  inout          GdiCoModuleCD refModuleCD
  /* in parameter
  Reference to a given module description. */
  inout          GdiCoToCoordStreamBuffered refCreateParameter
  /* in parameter
  the given Stream object contains the create parameter of the VD.
  If no create Parameter exists, NIL is handed over. */
  inout          String sVdInstanceName
  /* in parameter
  Instance name of the new VD. */
  inout          TCCheckComplete refCBCheckComplete
  /* in parameter
  Reference to a call back method which handles the unsolicited report of
  the VD state check complete.
  If a NIL reference is given, no event will be raised by the coordinator.
  */
  inout          TCVDFunctionReference refCBCompleteCreateVd
  /* in parameter
  Reference to a call back method which handles the initialization
  mechanism. A NIL reference signals a synchronous call. */
) : GdiCoVdRT

```

### Function Description

Effects the setting up of the VD within the Coordinator and the device driver.

In case of an asynchronous operation, the reference to the VD will be delivered by the complete call back. If no call back reference is given, the operation will be executed synchronously.

Return Value

Returns the reference to the new created GdiCoVdRT.  
NIL in case of an asynchronus operation.

**C.2.11.12 GdiCoFactory :: «F» gdiCreateVdRTByModuleClassName()**Function Call

```
gdiCreateVdRTByModuleClassName
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout          GdiCoDriverRT refDriverRT
    /* in parameter
    contains the reference to the driver. The new Application VD
    (administration structure) will be connected to this driver. */
    inout          String sModuleClassName
    /* in parameter
    contains the Class Name of the module, according to the DCD. */
    inout          GdiCoToCoordStreamBuffered refCreateParameter
    /* in parameter
    the given Stream object contains the create parameter of the VD.
    If no create Parameter exists, NIL is handed over. */
    inout          String sVdInstanceName
    /* in parameter
    Instancename of the new VD. */
    in             TCCheckComplete refCBCheckComplete
    /* in parameter
    Reference to a call back method which handles the unsolicited report of
    the VD state check complete.
    If a NIL reference is given, no event will be raised by the coordinator.
    */
    inout          TCVDFunctionReference refCBCompleteCreateVd
    /* in parameter
    Reference to a call back method which handles the initialization
    mechanism. A NIL reference signals a synchronous call. */
) : GdiCoVdRT
```

Function Description

Effects the setting up of the VD within the Coordinator and the device driver.

In case of an asynchronus operation, the reference to the VD will be delivered by the complete call back. If no call back reference is given, the operation will be executed synchronously.

Return Value

Returns the reference to the new created GdiCoVdRT.  
NIL in case of an asynchronus operation.

**C.2.11.13 GdiCoFactory :: «F» gdiCreateVdRTByModuleId()**Function Call

```
gdiCreateVdRTByModuleId
(
```

```

inout                unsigned long ulAppHnd
/* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout                GdiCoDriverRT refDriver
/* in parameter
contains the reference to the driver. The new Application VD
(administration structure) will be connected to this driver. */
inout                A_INT16 nModuleId
/* in parameter
Module ID, to which the VD shall be created. The module ID has to be
specified within the DCD of the driver or module description. */
inout                GdiCoToCoordStreamBuffered refCreateParameter
/* in parameter
the given Stream object contains the create parameter of the VD.
If no create Parameter exists, NIL is handed over. */
inout                String sVdInstanceName
/* in parameter
Instancename of the new VD. */
inout                TCVDFunctionReference refCBCompleteCreateVd
/* in parameter
Reference to a call back method which handles the initialization
mechanism. A NIL reference signals a synchronous call. */
) : GdiCoVdRT

```

### Function Description

Effects the setting up of the VD within the Coordinator and the device driver.

In case of an asynchronous operation, the reference to the VD will be delivered by the complete call back. If no call back reference is given, the operation will be executed synchronously.

### Return Value

Returns the reference to a GdiCoVdRT object if successful.

## **C.2.11.14 GdiCoFactory :: «F» gdiDeleteCommObjectRT()**

### Function Call

```

gdiDeleteCommObjectRT
(
    inout                unsigned long ulAppHnd
/* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
    inout                GdiCoCommObjectRT refCommObjectRT
/* in parameter
Reference to the Communication Object. */
) : void

```

### Function Description

Effects the deleting of the given Communication object instance within the driver and within the Coordinator.

### Return Value

None

**C.2.11.15 GdiCoFactory :: «F» gdiDeleteDriverRT()****Function Call**

```

gdiDeleteDriverRT
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoDriverRT refDriverRT
    /* in parameter
    reference to the DriverRT object. */
) : void

```

**Function Description**

Dereferences a GDI driver. If there are still applications VD of this Workspace within the driver, these are aborted. If further applications VD (other Workspace) are within the driver, the administration counter is decremented. If the administration counter is 0, the Control VD of the driver can be deleted and the driver can be unloaded. The time of the unloading is not defined. Possibly this may take place at a delayed time, to avoid the continued loading and unloading of a periodically-used driver.

**Return Value**

None

**C.2.11.16 GdiCoFactory :: «F» gdiDeleteFuncObjectRT()**

```

Function Call
gdiDeleteFuncObjectRT
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...)
    or by gdiGetWorkspaceByName() */
    inout                GdiCoFuncObjectRT refFuncObjectRT
    /* Reference to a Function object within the VD. */
) : void

```

**Function Description**

If the Function object is set up within the device driver, the Function object and all its Communication Objects are deleted within the driver. The administration structure of the FO and all its COs within the Coordinator are deleted.

**Return Value**

None

**C.2.11.17 GdiCoFactory :: «F» gdiDeleteVdRT()****Function Call**

```

gdiDeleteVdRT
(
    inout                unsigned long ulAppHnd
    /* in parameter

```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          GdiCoVdRT refVdRT
/* in parameter
Reference to a VD. */
) : void

```

### Function Description

Deletes the VD within the device driver. In case of success, the administration structure is deleted as well. The VD must not contain any Function objects.

### Return Value

None

## **C.2.11.18 GdiCoFactory :: «F» gdiReleaseCDReference()**

### Function Call

```

gdiReleaseCDReference
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout          GdiCoObjectCD refCD
    /* in parameter
    Reference to the Gdi Object Description object */
) : void

```

### Function Description

Deletes the references to the GdiCoDriverCD object and all its sub references to the description objects. The existing GDI object instances will not be affected.

### Return Value

None

## **C.2.12 «F» GdiCoFloatCD**

Contains services concerning the type Float for the definition of restrictions.

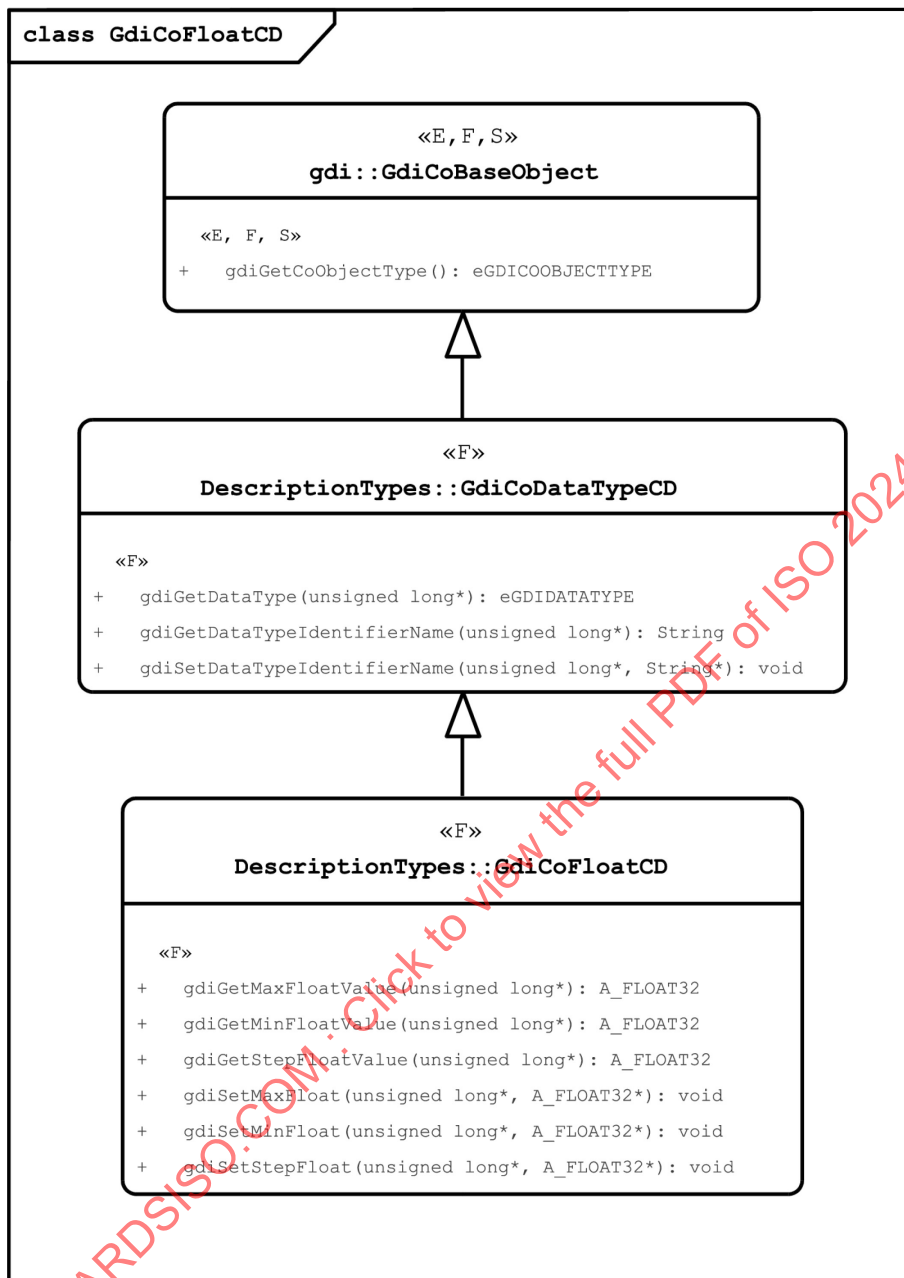


Figure C.12 — Hierarchical diagram of GdiCoFloatCD

## C.2.12.1 GdiCoFloatCD :: «F» gdiGetMaxFloatValue()

## Function Call

```

gdiGetMaxFloatValue
(
    inout          unsigned long ulAppHnd
    /*            in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT32
  
```

Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 3,402823466 E + 38.

Return Value

Returns the maximum value of a datum, based upon this type.

**C.2.12.2 GdiCoFloatCD :: «F» gdiGetMinFloatValue()**Function Call

```
gdiGetMinFloatValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT32
```

Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 1,175494351 E – 38.

Return Value

Returns the minimum value of a datum, based upon this type.

**C.2.12.3 GdiCoFloatCD :: «F» gdiGetStepFloatValue()**Function Call

```
gdiGetStepFloatValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_FLOAT32
```

Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

Return Value

Returns the step width of a datum, based upon this type.

**C.2.12.4 GdiCoFloatCD :: «F» gdiSetMaxFloat()**Function Call

```
gdiSetMaxFloat
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          A_FLOAT32 fLimitValue
/* in parameter
new maximum value (< 3,402823466 E + 39) */
) : void

```

### Function Description

Defines the maximum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.12.5 GdiCoFloatCD :: «F» gdiSetMinFloat()**

### Function Call

```

gdiSetMinFloat
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_FLOAT32 fLimitValue
  /* in parameter
  new minimum value (> 1,175494351 E - 39) */
) : void

```

### Function Description

Defines the minimum value for a datum based upon this type. The value has no effects on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.12.6 GdiCoFloatCD :: «F» gdiSetStepFloat()**

### Function Call

```

gdiSetStepFloat
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_FLOAT32 fLimitValue
  /* in parameter
  new step width (0 < LimitValue < 3,402823466 E + 38) */
) : void

```



Function Description

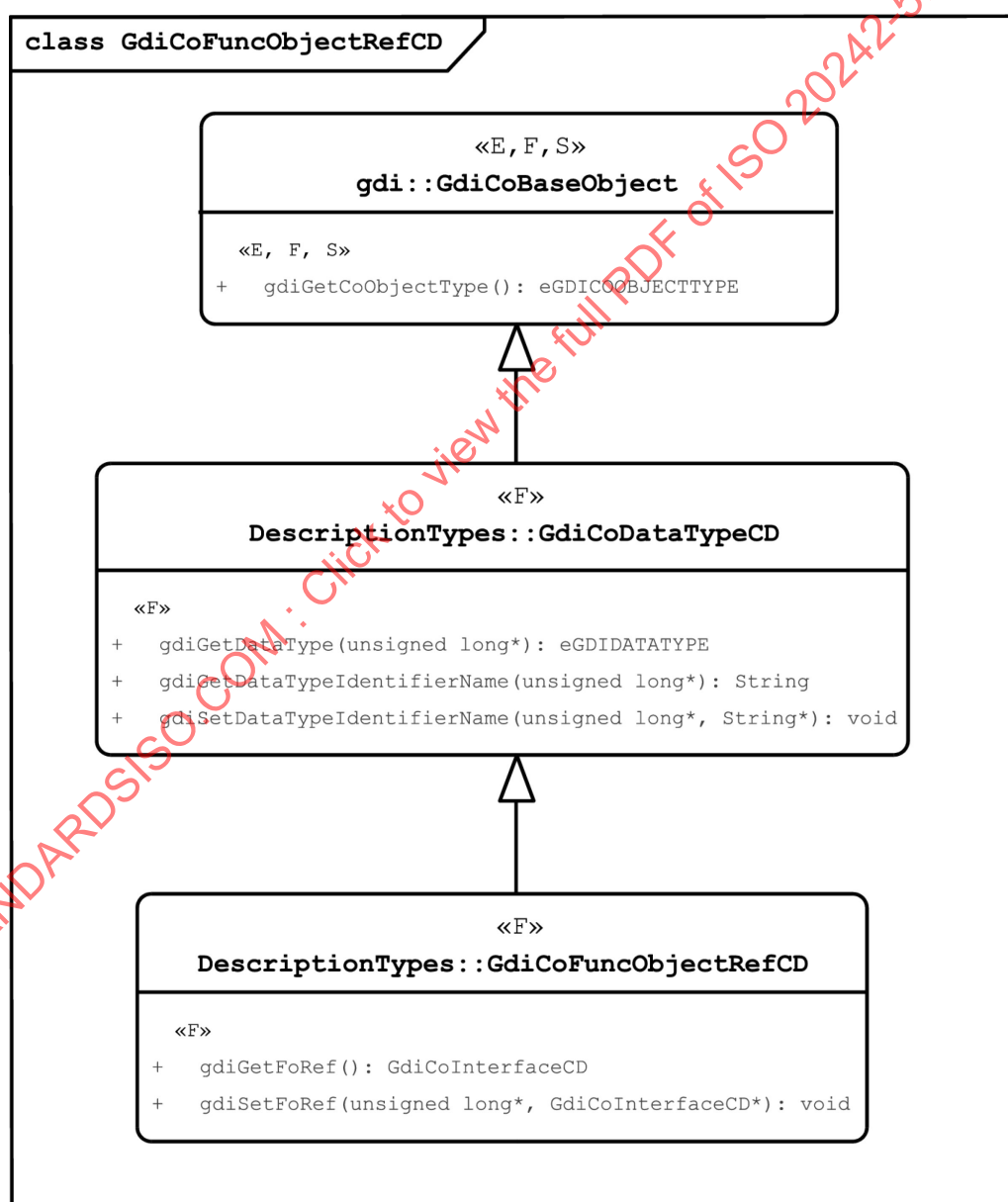
Defines the step width for valid data based upon this type between minimum and maximum. The minimum is the starting value. In case of wrong inputs, empty sets may occur. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

Return Value

None

**C.2.13 «F» GdiCoFuncObjectRefCD**

This interface contains services for the handling of the type function object reference.



**Figure C.13 — Hierarchical diagram of GdiCoFuncObjectRefCD**

**C.2.13.1 GdiCoFuncObjectRefCD :: «F» gdiGetFoRef()****Function Call**

```
gdiGetFoRef
(
    ) : GdiCoInterfaceCD
```

**Function Description**

Returns the description of the referenced function object. The value has no effects on the communication to the device driver. It is read by certain applications to obtain detailed type information.

**Return Value**

Returns the reference to the description of the dynamic Interface.

**C.2.13.2 GdiCoFuncObjectRefCD :: «F» gdiSetFoRef()****Function Call**

```
gdiSetFoRef
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout          GdiCoInterfaceCD refInterfaceCD
    /* in parameter
    contain the reference to the description of the dynamic Interface. */
) : void
```

**Function Description**

Defines the Function ID of the function object reference. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

**Return Value**

None

**C.2.14 «F» GdiCoInterfaceCD**

This interface describes a Function Object. This object reference is valid, till the creation or referencing the next sub description.

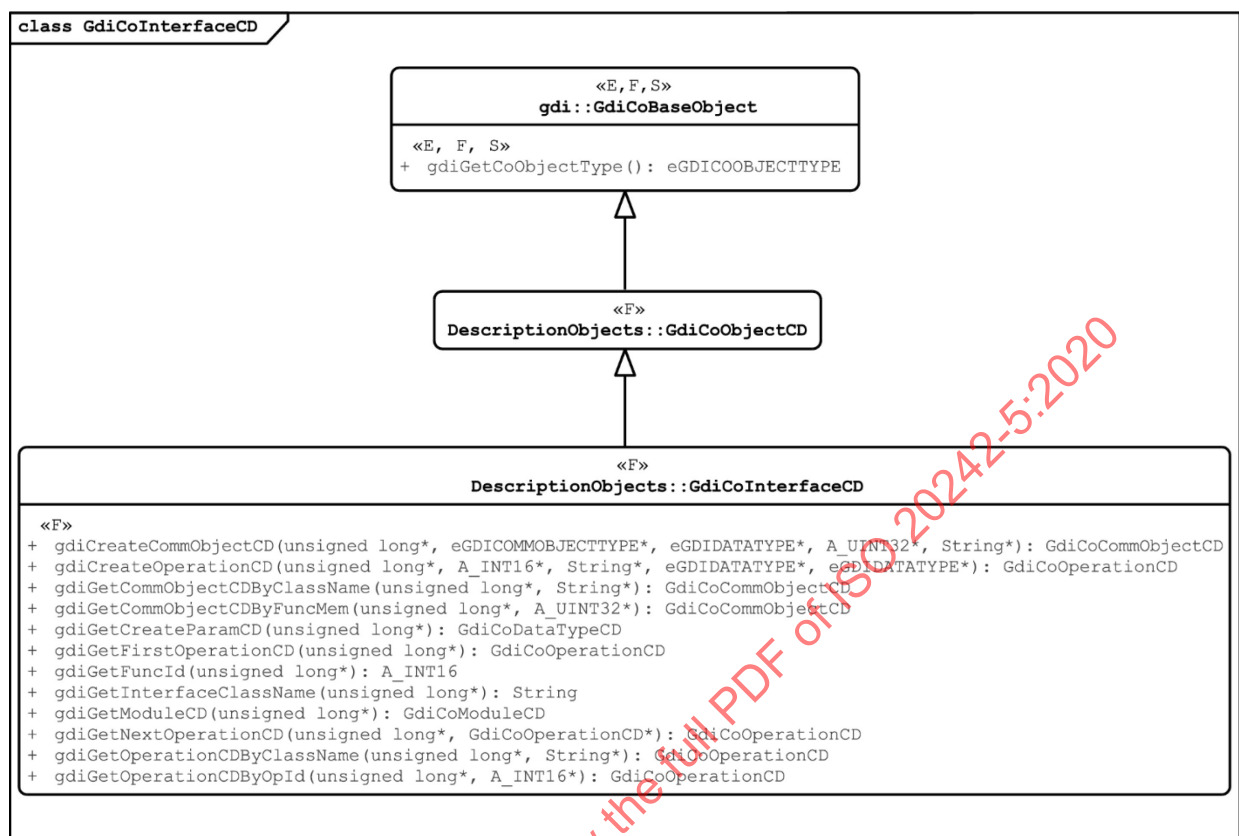


Figure C.14 — Hierarchical diagram of GdiCoInterfaceCD

### C.2.14.1 GdiCoInterfaceCD :: «F» gdiCreateCommObjectCD()

#### Function Call

```

gdiCreateCommObjectCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                eGDICOMMOBJECTTYPE eCommObjectType
    /* in parameter
    Controls the instancing of the administration structure as attributes
    or as parameters. */
    inout                eGDIDATATYPE eDataType
    /* in parameter
    contains the GDI type of the element. */
    inout                A_UINT32 ulCommObjectFuncMem
    /* in parameter
    contains the absolute index of the communication object.
    The index must be unique inside the scope of this FO.
    Inherited COs are resolved. */
    inout                String sCommObjectClassName
    /* in parameter
    contains the class name of the communication object. */
) : GdiCoCommObjectCD
  
```

Function Description

Returns a new GdiCoCommObjectCD reference.

Return Value

Returns a new GdiCommObjectCD reference.

**C.2.14.2 GdiCoInterfaceCD :: «F» gdiCreateOperationCD()**Function Call

```
gdiCreateOperationCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT16 nOperationId
    /* in parameter
    contains the operation ID */
    inout                String sOperationClassName
    /* in parameter
    contains the class name of the Operation. */
    inout                eGDIDATATYPE eInType
    /* in parameter
    contains the type of the in parameter of the operation.
    EDT_NO_DEFINITION if void. */
    inout                eGDIDATATYPE eOutType
    /* in parameter
    contains the type of the out parameter of the operation.
    EDT_NO_DEFINITION if void. */
) : GdiCoOperationCD
```

Function Description

Returns a new GdiCoOperationCD reference.

Return Value

Returns a new reference to GdiOperationCD.

**C.2.14.3 GdiCoInterfaceCD :: «F» gdiGetCommObjectCDByClassName()**Function Call

```
gdiGetCommObjectCDByClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sCommObjectName
    /* in parameter
    contains the communication object name. */
) : GdiCoCommObjectCD
```

**Function Description**

Returns the reference to the communication object description corresponding to the given operation name.

**Return Value**

Returns the communication object description corresponding to the given communication object name.

**C.2.14.4 GdiCoInterfaceCD :: «F» gdiGetCommObjectCDByFuncMem()****Function Call**

```
gdiGetCommObjectCDByFuncMem
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_UINT32 ulFuncMem
    /* in parameter
    contains the absolute index of the Communication Object */
) : GdiCoCommObjectCD
```

**Function Description**

Returns the reference to the communication object description corresponding to the given FuncMem index. The absolute index is unique in the FO scope.

**Return Value**

Returns the communication object description corresponding to the given communication object absolute index.

**C.2.14.5 GdiCoInterfaceCD :: «F» gdiGetCreateParamCD()****Function Call**

```
gdiGetCreateParamCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD
```

**Function Description**

Returns the create parameter type described in the FuncObject description.

In case the create parameter type is already set because the coordinator handles a PID file, the create parameter type can be get by calling this method.

**Return Value**

Returns the create parameter type, described in the FuncObject description.  
In case no create parameter is used, a NIL Reference will be returned.

**C.2.14.6 GdiCoInterfaceCD :: «F» gdiGetFirstOperationCD()****Function Call**

```

gdiGetFirstOperationCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoOperationCD

```

**Function Description**

Creates reference to the first Operation description within the FO description if exists.

**Return Value**

Returns reference to the first gdiCoOperationCD if it exists, NIL otherwise.

**C.2.14.7 GdiCoInterfaceCD :: «F» gdiGetFuncId()****Function Call**

```

gdiGetFuncId
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT16

```

**Function Description**

Returns the ID of the FO described in the FO description.

In case the FO ID is already set because the coordinator handles a PID file, the module ID can be get by calling this method.

**Return Value**

Returns the FO ID, described in this function object description.

**C.2.14.8 GdiCoInterfaceCD :: «F» gdiGetInterfaceClassName()****Function Call**

```

gdiGetInterfaceClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String

```

**Function Description**

Returns the class name of the FO described in this FO description.

In case the FO class name is already set because the coordinator handles a PID file, the FO class name can be get by calling this method.

#### Return Value

Returns the Interface name, described in the function object description.

### **C.2.14.9 GdiCoInterfaceCD :: «F» gdiGetModuleCD()**

#### Function Call

```
gdiGetModuleCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoModuleCD
```

#### Function Description

Create a reference to the upper GdiCoModuleCD.

#### Return Value

Returns the reference to GdiCoModuleCD.

### **C.2.14.10 GdiCoInterfaceCD :: «F» gdiGetNextOperationCD()**

#### Function Call

```
gdiGetNextOperationCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoOperationCD refLastElement
    /* in parameter
    contains the reference to the last operation description */
) : GdiCoOperationCD
```

#### Function Description

Creates reference to the next Operation description within the module description if exists.

#### Return Value

Returns GdiCoOperationCD if it exists, NIL otherwise.

### **C.2.14.11 GdiCoInterfaceCD :: «F» gdiGetOperationCDByClassName()**

#### Function Call

```
gdiGetOperationCDByClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```
Identifier of the application returned by gdiCreateWorkspace(...) or by  
gdiGetWorkspaceByName() */  
inout      String sOperationName  
/* in parameter  
contains the operation name. */  
) : GdiCoOperationCD
```

#### Function Description

Returns the operation description corresponding to the given operation name.

#### Return Value

Returns the operation description corresponding to the given operation name.

### **C.2.14.12 GdiCoInterfaceCD :: «F» gdiGetOperationCDByOpId()**

#### Function Call

```
gdiGetOperationCDByOpId  
(  
    inout      unsigned long ulAppHnd  
    /* in parameter  
    Identifier of the application returned by gdiCreateWorkspace(...) or by  
    gdiGetWorkspaceByName() */  
    inout      A_INT16 nOpId  
    /* in parameter  
    contains the FO ID. */  
) : GdiCoOperationCD
```

#### Function Description

Returns the operation description corresponding to the given operation ID.

#### Return Value

Returns the operation description corresponding to the given operation ID.

### **C.2.15 «F» GdiCoLongCD**

Contains services concerning the type long for the definition of restrictions.



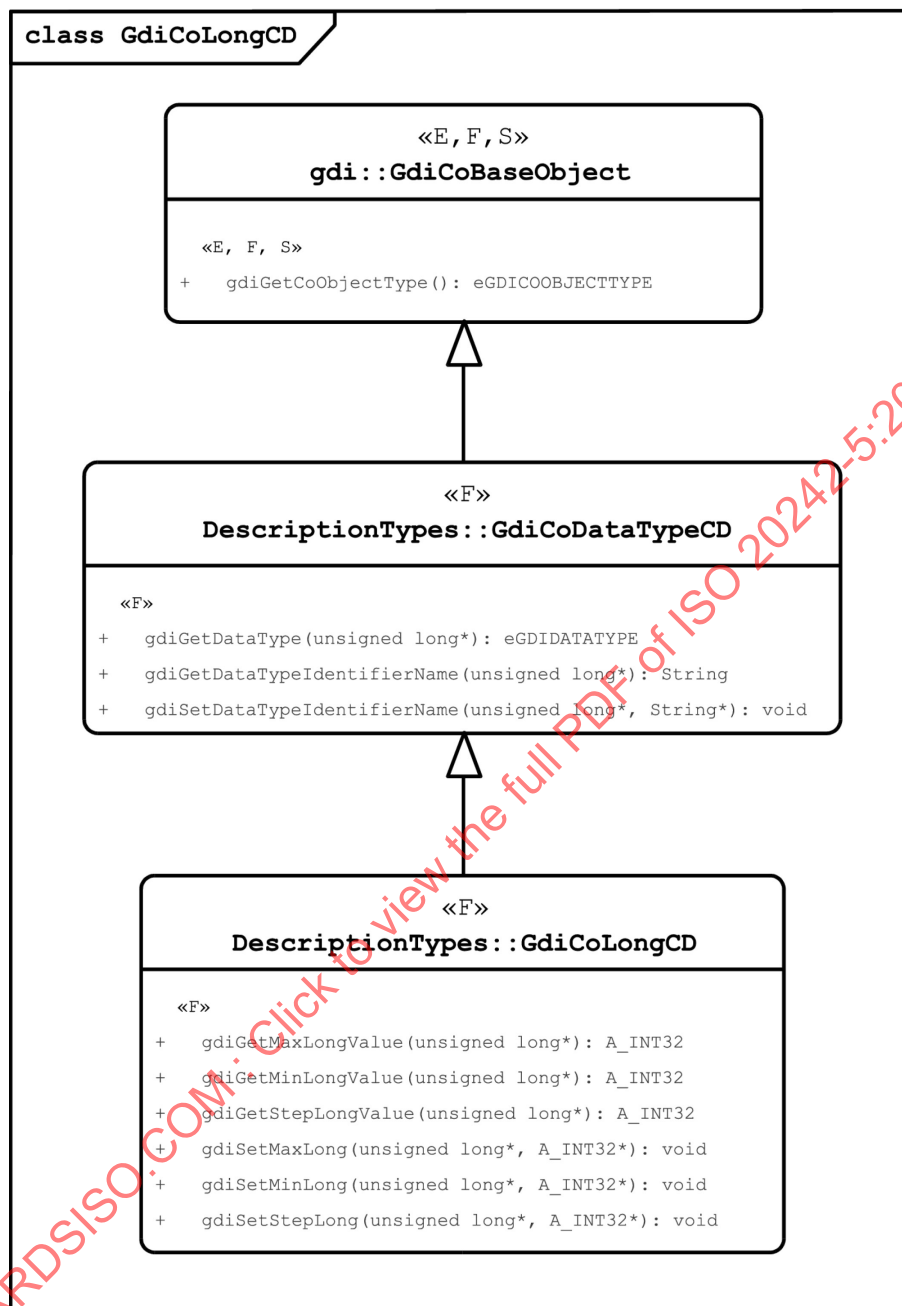


Figure C.15 — Hierarchical diagram of GdiCoLongCD

## C.2.15.1 GdiCoLongCD :: «F» gdiGetMaxLongValue()

Function Call

```

gdiGetMaxLongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT32
  
```

### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 2147483647.

### Return Value

Returns the maximum value of a datum, based upon this type.

## **C.2.15.2 GdiCoLongCD :: «F» gdiGetMinLongValue()**

### Function Call

```
gdiGetMinLongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT32
```

### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value -2147483648.

### Return Value

Returns the minimum value of a datum, based upon this type.

## **C.2.15.3 GdiCoLongCD :: «F» gdiGetStepLongValue()**

### Function Call

```
gdiGetStepLongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT32
```

### Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

### Return Value

Returns the step width of a datum, based upon this type.

## **C.2.15.4 GdiCoLongCD :: «F» gdiSetMaxLong()**

### Function Call

```
gdiSetMaxLong
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          A_INT32 lLimitValue
/* in parameter
new maximum value (< 2147483648) */
) : void

```

### Function Description

Defines the maximum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.15.5 GdiCoLongCD :: «F» gdiSetMinLong()**

### Function Call

```

gdiSetMinLong
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_INT32 lLimitValue
  /* in parameter
  new minimum value (> -2147483649) */
) : void

```

### Function Description

Defines the minimum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.15.6 GdiCoLongCD :: «F» gdiSetStepLong()**

### Function Call

```

gdiSetStepLong
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_INT32 lLimitValue
  /* in parameter
  new step width (0 < LimitValue < 2147483647) */
) : void

```

Function Description

Defines the step width for valid data based upon this type between minimum and maximum. The minimum is the starting value. In case of wrong inputs, empty sets may occur. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

Return Value

None

C.2.16 «F» GdiCoModuleCD

This interface describes a Modul. This object reference is valid, till the creation or referencing the next sub description.

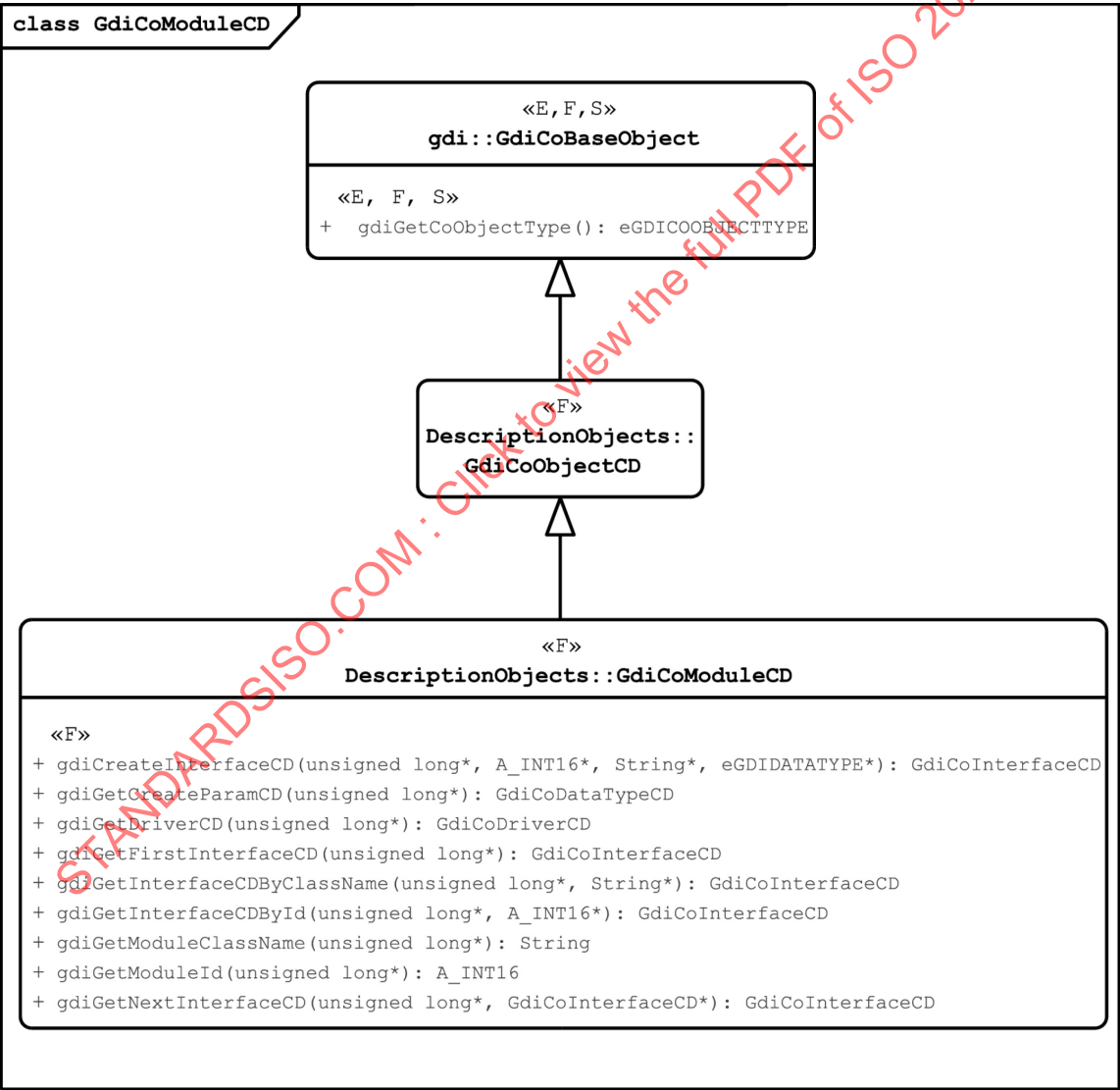


Figure C.16 — Hierarchical diagram of GdiCoModuleCD

**C.2.16.1 GdiCoModuleCD :: «F» gdiCreateInterfaceCD()****Function Call**

```

gdiCreateInterfaceCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT16 nFuncId
    /* in parameter
    contains the Function object ID */
    inout                String sInterfaceClassName
    /* in parameter
    contains the class name of the Func Object. */
    inout                eGDIDATATYPE eCreateParamType
    /* in parameter
    contains the type of the create parameter.
    eDT_NO_DEFINITION if not necessary. */
) : GdiCoInterfaceCD

```

**Function Description**

Returns a new GdiCoFuncObjectCD.

**Return Value**

Returns a new GdiFuncObjectCD object.

**C.2.16.2 GdiCoModuleCD :: «F» gdiGetCreateParamCD()****Function Call**

```

gdiGetCreateParamCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD

```

**Function Description**

Returns the create parameter type described in the module description.

In case the create parameter type is already set because the coordinator handles a PID file, the create parameter type can be get by calling this method.

**Return Value**

Returns the create parameter type, described in the module description.

In case no create parameter is used, a NIL Reference will be returned.

**C.2.16.3 GdiCoModuleCD :: «F» gdiGetDriverCD()****Function Call**

```

gdiGetDriverCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDriverCD

```

**Function Description**

Create a reference to the upper GdiCoDriverCD.

**Return Value**

Returns the reference to GdiCoDriverCD.

**C.2.16.4 GdiCoModuleCD :: «F» gdiGetFirstInterfaceCD()****Function Call**

```

gdiGetFirstInterfaceCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoInterfaceCD

```

**Function Description**

Creates reference to the first FO description within the module description if it exists.

**Return Value**

Returns the reference to the first GdiCoInterfaceCD if it exists, NIL otherwise.

**C.2.16.5 GdiCoModuleCD :: «F» gdiGetInterfaceCDByClassName()****Function Call**

```

gdiGetInterfaceCDByClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sInterfaceName
    /* in parameter
    contains the FO lass name. */
) : GdiCoInterfaceCD

```

**Function Description**

Returns the FO description corresponding to the given FO class name.

Return Value

Returns the FO description corresponding to the given FO name.

**C.2.16.6 GdiCoModuleCD :: «F» gdiGetInterfaceCDBById()**Function Call

```
gdiGetInterfaceCDBById
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_INT16 nFuncId
    /* in parameter
    contains the FO ID. */
) : GdiCoInterfaceCD
```

Function Description

Returns the FO description corresponding to the given FO ID.

Return Value

Returns the FO description corresponding to the given function object ID, NIL if no FO description with the given ID exists in this module description.

**C.2.16.7 GdiCoModuleCD :: «F» gdiGetModuleClassName()**Function Call

```
gdiGetModuleClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String
```

Function Description

Returns the class name of the module described in the module description.

In case the module name is already set because the coordinator handles a PID file, the module name can be get by calling this method.

Return Value

Returns the module name, described in the module description.

**C.2.16.8 GdiCoModuleCD :: «F» gdiGetModuleId()**Function Call

```
gdiGetModuleId
(
    inout                unsigned long ulAppHnd
```

```
/* in parameter  
Identifier of the application returned by gdiCreateWorkspace(...) or by  
gdiGetWorkspaceByName() */  
) : A_INT16
```

#### Function Description

Returns the ID of the module described in the module description.

In case the module ID is already set because the coordinator handles a PID file, the module ID can be get by calling this method.

#### Return Value

Returns the module ID, described in the module description.

### **C.2.16.9 GdiCoModuleCD :: «F» gdiGetNextInterfaceCD()**

#### Function Call

```
gdiGetNextInterfaceCD  
(  
    inout                unsigned long ulAppHnd  
    /* in parameter  
    Identifier of the application returned by gdiCreateWorkspace(...) or by  
    gdiGetWorkspaceByName() */  
    inout                GdiCoInterfaceCD refLastElement  
    /* in parameter  
    contains the reference to the last Function Object description */  
) : GdiCoInterfaceCD
```

#### Function Description

Creates reference to the next FO description within the module description if exists.

#### Return Value

Returns GdiFuncObjectCD if exists, NIL otherwise.

### **C.2.17 «F» GdiCoObjectCD**

This is the abstract Base Interface of all GDI Object Descriptions.



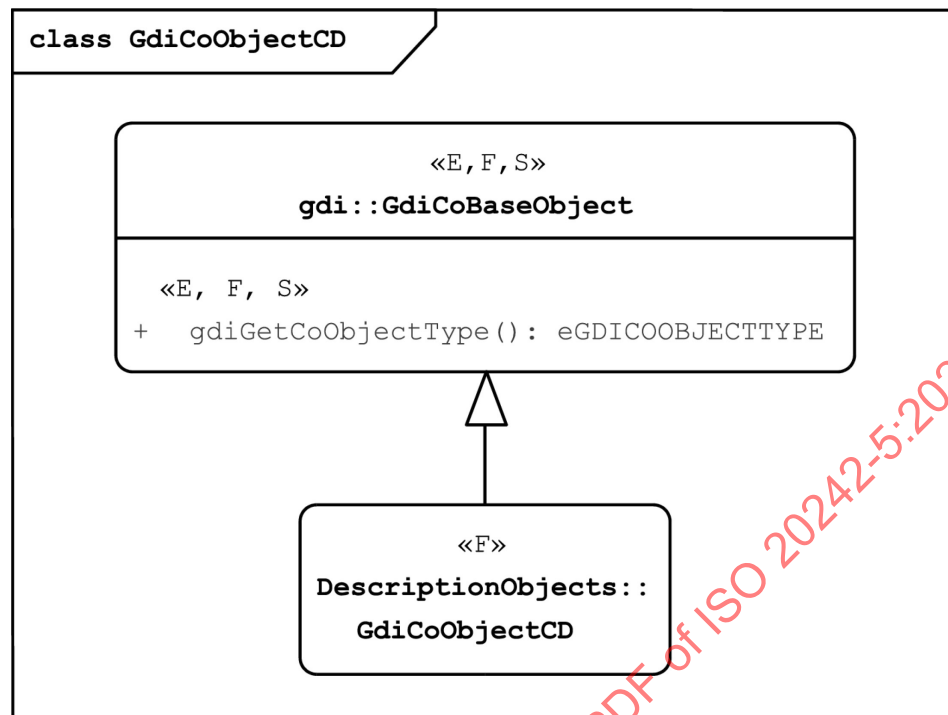


Figure C.17 — Hierarchical diagram of GdiCoObjectCD

### C.2.18 «F» GdiCoObjectNavigator

This static interface contains services for iterating over GDI Description Objects, Instances and GDI types.



Figure C.18 — Hierarchical diagram of GdiCoObjectNavigator

### C.2.18.1 GdiCoObjectNavigator :: «F» gdiGetCommObjectCD()

#### Function Call

```

gdiGetCommObjectCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoCommObjectRT refCommObjectRT
    /* in parameter
    Contains the Reference to the Communication Object. */
) : GdiCoCommObjectCD
  
```

#### Function Description

Generates a Reference to the Description of the given Communication Object as GdiCoCommObjectCD object.

#### Return Value

Returns the reference to GdiCoCommObjectCD.

**C.2.18.2 GdiCoObjectNavigator :: «F» gdiGetDriverRT()****Function Call**

```

gdiGetDriverRT
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoVdRT refVdRT
    /* in parameter
    contains the reference to the Application VdRT. */
) : GdiCoDriverRT

```

**Function Description**

Creates reference to an existing Driver Object (RT).

The referenced DriverRT contains the given Application VdRT (passed as parameter).

**Return Value**

Returns GdiCoDriverRT.

**C.2.18.3 GdiCoObjectNavigator :: «F» gdiGetFirstDriverCD()****Function Call**

```

gdiGetFirstDriverCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoWorkspace refWorkspace
    /* in parameter
    contains the reference to the workspace. The driver will be connected
    to this workspace. */
) : GdiCoDriverCD

```

**Function Description**

Returns the first GdiCoDriverCD if it exists.

**Return Value**

Returns GdiCoDriverCD if it exists, NIL otherwise.

**C.2.18.4 GdiCoObjectNavigator :: «F» gdiGetInterfaceCD()****Function Call**

```

gdiGetInterfaceCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoFuncObjectRT refFuncObjectRT

```

```
/* in parameter  
Contains the Reference to the Function Object. */  
) : GdiCoInterfaceCD
```

#### Function Description

Generates a Reference to the Description of the given Function Object as GdiCoInterfaceCD object.

#### Return Value

Returns the reference to GdiCoInterfaceCD.

### **C.2.18.5 GdiCoObjectNavigator :: «F» gdiGetNextDriverCD()**

#### Function Call

```
gdiGetNextDriverCD  
(  
    inout                unsigned long ulAppHnd  
    /* in parameter  
    Identifier of the application returned by gdiCreateWorkspace(...) or by  
    gdiGetWorkspaceByName() */  
    inout                GdiCoWorkspace refWorkspace  
    /* in parameter  
    contains the reference to the workspace. The driver will be connected  
    to this workspace. */  
    inout                GdiCoDriverCD refLastDriverCD  
    /* in parameter  
    contains the reference to the last Driver Description. */  
) : GdiCoDriverCD
```

#### Function Description

Returns the next Driver description, connected to the Workspace.

#### Return Value

Returns GdiCoDriverCD if it exists, NIL otherwise.

### **C.2.18.6 GdiCoObjectNavigator :: «F» gdiGetOperationCD()**

#### Function Call

```
gdiGetOperationCD  
(  
    inout                unsigned long ulAppHnd  
    /* in parameter  
    Identifier of the application returned by gdiCreateWorkspace(...) or by  
    gdiGetWorkspaceByName() */  
    inout                GdiCoOperationRT refOperationRT  
    /* in parameter  
    Contains the Reference to the OperationRT object. */  
) : GdiCoOperationCD
```

#### Function Description

Generate a Reference to the Description of the given OperationRT as GdiCoOperationCD object.

Return Value

Returns the references to GdiCoOperationCD.

**C.2.18.7 GdiCoObjectNavigator :: «F» gdiGetVdRT()**Function Call

```

gdiGetVdRT
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoFuncObjectRT refFuncObjectRT
    /* in parameter
    Contains the reference to the function object. */
) : GdiCoVdRT

```

Function Description

Creates reference to the existing Application VD.

The referenced Application VD contains the given Function Object (passed as parameter).

Return Value

Returns a reference to GdiCoVdRT corresponding to the given FuncObjectRT.

**C.2.19 «F» GdiCoOctetCD**

Contains services concerning the type unsigned Char for the definition of restrictions.

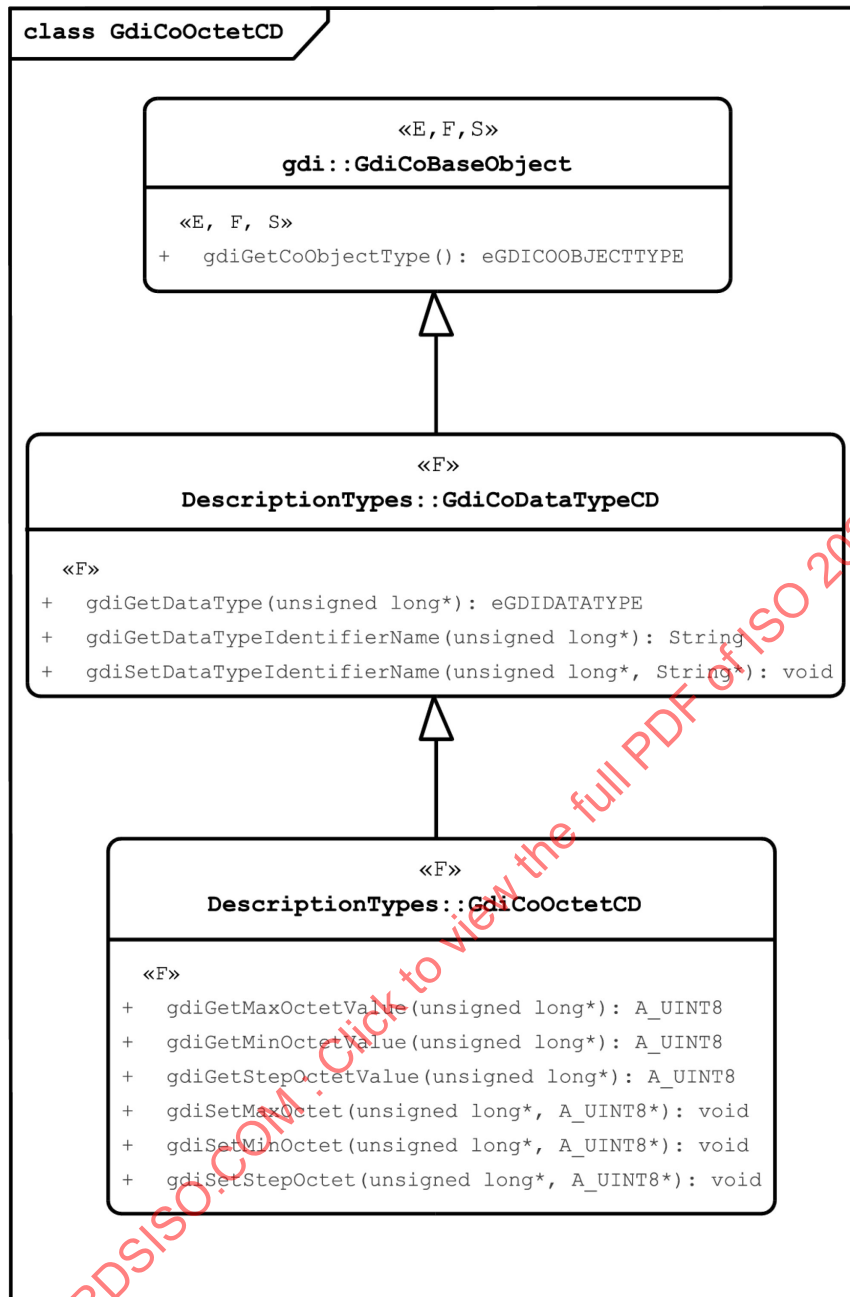


Figure C.19 — Hierarchical diagram of GdiCoOctetCD

### C.2.19.1 GdiCoOctetCD :: «F» gdiGetMaxOctetValue()

#### Function Call

```

gdiGetMaxOctetValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT8
  
```

**Function Description**

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 255.

**Return Value**

Returns the maximum value of a datum, based upon this type.

**C.2.19.2 GdiCoOctetCD :: «F» gdiGetMinOctetValue()****Function Call**

```
gdiGetMinOctetValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT8
```

**Function Description**

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 0.

**Return Value**

Returns the minimum value of a datum, based upon this type.

**C.2.19.3 GdiCoOctetCD :: «F» gdiGetStepOctetValue()****Function Call**

```
gdiGetStepOctetValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT8
```

**Function Description**

Defines the step width for valid data based upon this type between minimum and maximum. The minimum is the starting value. In case of wrong inputs, empty sets may occur. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

**Return Value**

Returns the step width of a datum, based upon this type.

**C.2.19.4 GdiCoOctetCD :: «F» gdiSetMaxOctet()****Function Call**

```
gdiSetMaxOctet
(
```

```

    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_UINT8 ucLimitValue
    /* in parameter
    new maximum value (< 128) */
    ) : void

```

### Function Description

Defines the maximum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.19.5 GdiCoOctetCD :: «F» gdiSetMinOctet()**

### Function Call

```

gdiSetMinOctet
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_UINT8 ucLimitValue
    /*
    new minimum value (0) */
    ) : void

```

### Function Description

Defines the minimum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.19.6 GdiCoOctetCD :: «F» gdiSetStepOctet()**

### Function Call

```

gdiSetStepOctet
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_UINT8 ucLimitValue
    /* in parameter
    new step width (0 < LimitValue < 127) */
    ) : void

```



Function Description

Defines the step width for valid data based upon this type between minimum and maximum. The minimum is the starting value. In case of wrong inputs, empty sets may occur. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

Return Value

None

**C.2.20 «F» GdiCoOperationCD**

This interface describes an Operation Object. This object reference is valid, till the creation or referencing the next sub description.



**Figure C.20 — Hierarchical diagram of GdiCoOperationCD**

**C.2.20.1 GdiCoOperationCD :: «F» gdiGetInParamDataTypeCD()****Function Call**

```

gdiGetInParamDataTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD

```

**Function Description**

Returns the type description of the input parmeter.

In case the type description is already set because the coordinator handles a PID file, the type description can be get by calling this method.

**Return Value**

Returns the type description of the input parmeter.

In case no type description is set, a NIL Reference will be returned.

**C.2.20.2 GdiCoOperationCD :: «F» gdiGetInParamName()****Function Call**

```

gdiGetInParamName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String

```

**Function Description**

Returns the Identifier name of the In parameter.

**Return Value**

Returns Identifier of the in Parameter.

Returns Empty String if no Parameter Name is set.

**C.2.20.3 GdiCoOperationCD :: «F» gdiGetInterfaceCD()****Function Call**

```

gdiGetInterfaceCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoInterfaceCD

```

Function Description

Creates a reference to the upper GdiCoFuncObjectCD.

Return Value

Returns GdiCoFuncObjectCD.

**C.2.20.4 GdiCoOperationCD :: «F» gdiGetOperationClassName()**Function Call

```
gdiGetOperationClassName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String
```

Function Description

Returns the class name of the operation described in this operation description.

In case the operation class name is already set because the coordinator handles a PID file, the operation class name can be get by calling this method.

Return Value

Returns the class name of the operation, described in the operation description.

In case no class name is set an empty string will be returned.

**C.2.20.5 GdiCoOperationCD :: «F» gdiGetOperationId()**Function Call

```
gdiGetOperationId
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT16
```

Function Description

Returns the ID of the operation described in the operation description.

In case the module ID is already set because the coordinator handles a PID file, the module ID can be get by calling this method.

Return Value

Returns the operation ID, described in the operation description.

**C.2.20.6 GdiCoOperationCD :: «F» gdiGetOutParamDataTypeCD()****Function Call**

```

gdiGetOutParamDataTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD

```

**Function Description**

Returns the type description of the output parameter.

In case the type description is already set because the coordinator handles a PID file, the type description can be get by calling this method.

**Return Value**

Returns the type description of the output parameter.

In case no type description is set, a NIL Reference will be returned.

**C.2.20.7 GdiCoOperationCD :: «F» gdiGetOutParamName()****Function Call**

```

gdiGetOutParamName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String

```

**Function Description**

Returns the Identifier name of the Out parameter.

**Return Value**

Returns Identifier of the out Parameter.

Returns Empty String if no Parameter Name is set.

**C.2.20.8 GdiCoOperationCD :: «F» gdiSetInParamName()****Function Call**

```

gdiSetInParamName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sInParamName
    /* in parameter
    contains the name of the In Parameter. */
)

```

```
) : void
```

#### Function Description

Sets the name of the in parameter.

#### Return Value

None

### **C.2.20.9 GdiCoOperationCD :: «F» gdiSetOutParamName()**

#### Function Call

```
gdiSetOutParamName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                String sOutParamName
    /* in parameter
    contains the name of the Out Parameter. */
) : void
```

#### Function Description

Sets the name of the in parameter.

#### Return Value

None

### **C.2.21 «F» GdiCoSequenceCD**

Description of the instance of a Sequence. This object reference is valid, till the creation or referencing the next sub description.

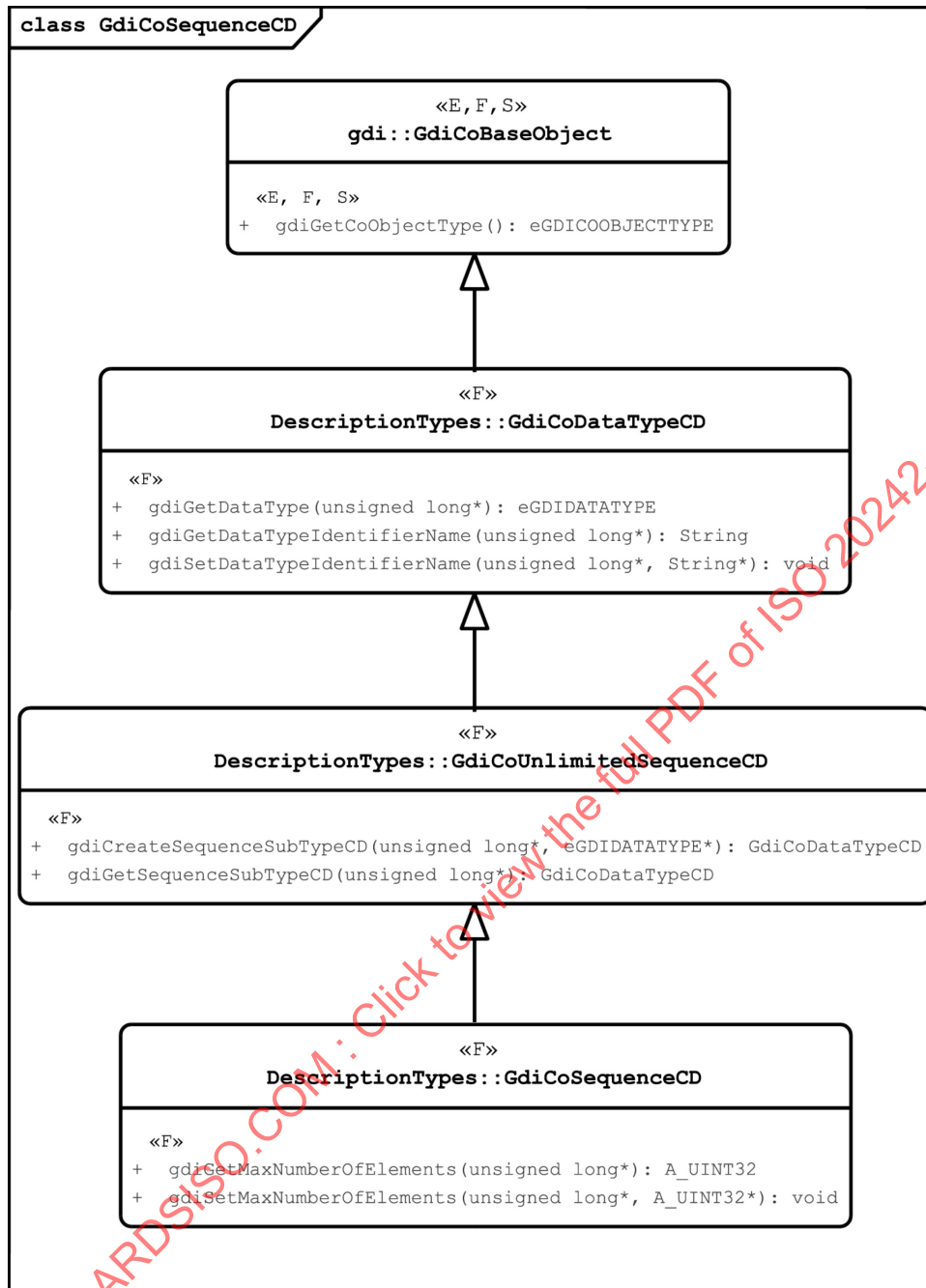


Figure C.21 — Hierarchical diagram of GdiCoSequenceCD

### C.2.21.1 GdiCoSequenceCD :: «F» gdiGetMaxNumberOfElements()

#### Function Call

```

gdiGetMaxNumberOfElements
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32
  
```

**Function Description**

It is used for the detection of the maximum number of elements of a Sequence.

**Return Value**

Returns the maximum number of elements of a Sequence.

**C.2.21.2 GdiCoSequenceCD :: «F» gdiSetMaxNumberOfElements()****Function Call**

```
gdiSetMaxNumberOfElements
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                A_UINT32 ulLimit
    /* in parameter
    contains the maximal number of elements of the limited sequence. */
) : void
```

**Function Description**

Defines the maximum number of elements of a Sequence.

**Return Value**

None

**C.2.22 «F» GdiCoShortCD**

Contains services concerning the type Short for the definition of restrictions.

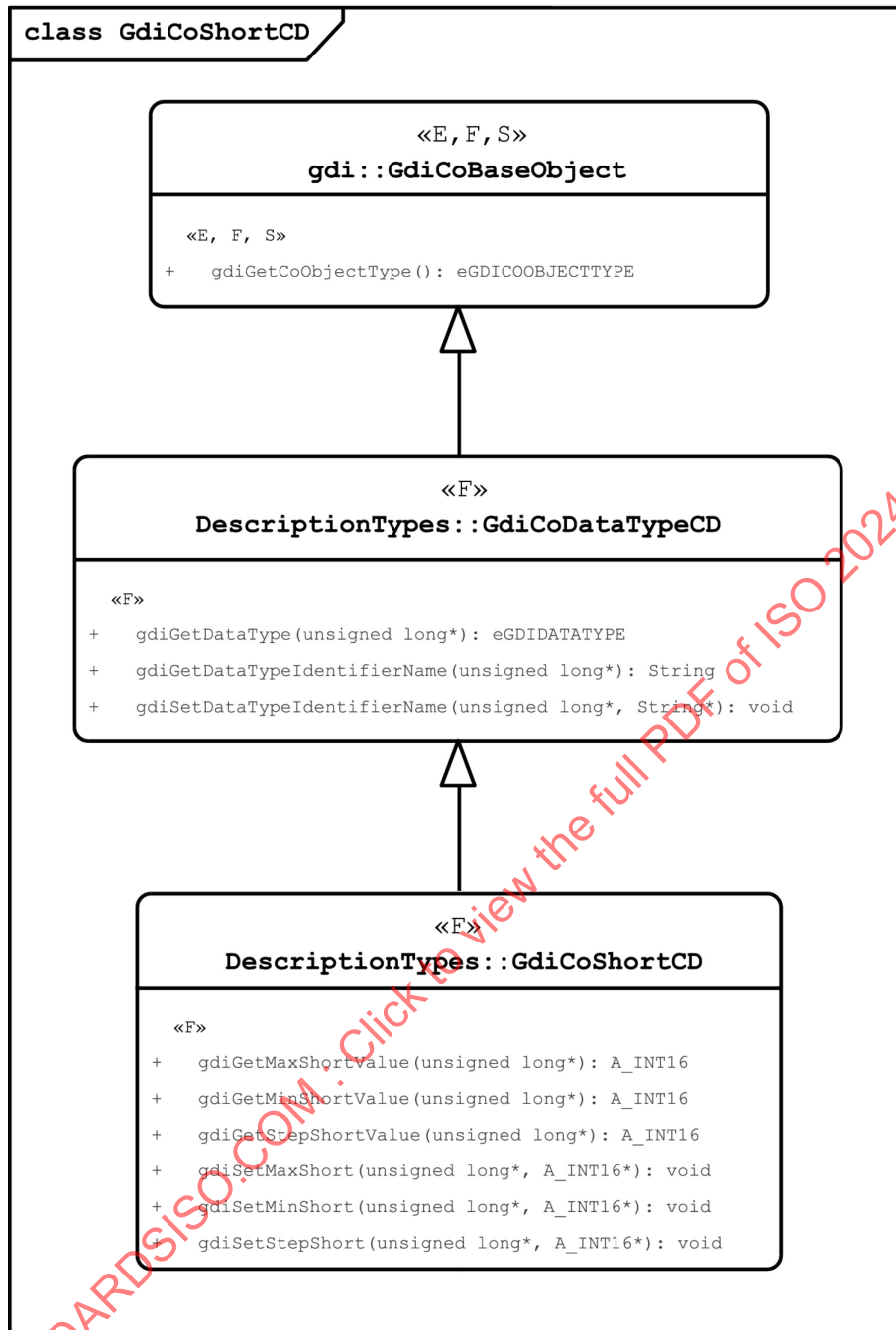


Figure C.22 — Hierarchical diagram of GdiCoShortCD

## C.2.22.1 GdiCoShortCD :: «F» gdiGetMaxShortValue()

Function Call

```

gdiGetMaxShortValue
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */

```



) : A\_INT16

#### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 32767.

#### Return Value

Returns the maximum value of a datum, based upon this type.

### **C.2.22.2 GdiCoShortCD :: «F» gdiGetMinShortValue()**

#### Function Call

```
gdiGetMinShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT16
```

#### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value -32768.

#### Return Value

Returns the minimum value of a datum, based upon this type.

### **C.2.22.3 GdiCoShortCD :: «F» gdiGetStepShortValue()**

#### Function Call

```
gdiGetStepShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_INT16
```

#### Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

#### Return Value

Returns the step width of a datum, based upon this type.

### **C.2.22.4 GdiCoShortCD :: «F» gdiSetMaxShort()**

#### Function Call

```
gdiSetMaxShort
(
```

```

inout                unsigned long ulAppHnd
/* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout                A_INT16 nLimitValue
/* in parameter
new maximum value (< 32768) */
) : void

```

### Function Description

Defines the maximum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.22.5 GdiCoShortCD :: «F» gdiSetMinShort()**

### Function Call

```

gdiSetMinShort
(
inout                unsigned long ulAppHnd
/* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout                A_INT16 nLimitValue
/* in parameter
new minimum value (> -32769) */
) : void

```

### Function Description

Defines the minimum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.22.6 GdiCoShortCD :: «F» gdiSetStepShort()**

### Function Call

```

gdiSetStepShort
(
inout                unsigned long ulAppHnd
/* in parameter
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout                A_INT16 nLimitValue
/* in parameter
new step width (0 < LimitValue < 32767) */
) : void

```

Function Description

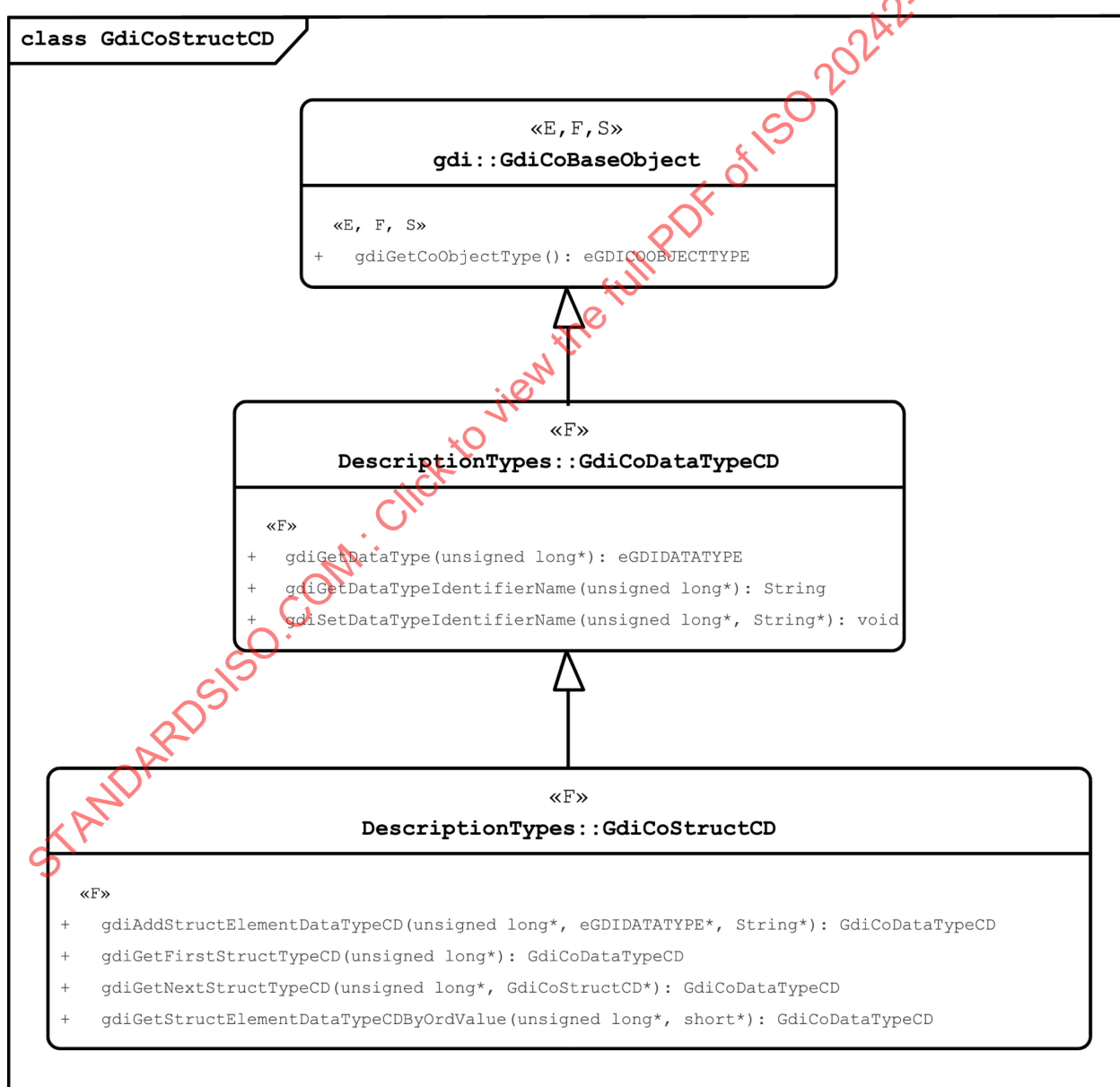
Defines the step width for valid data based upon this type between minimum and maximum. The minimum is the starting value. In case of wrong inputs, empty sets may occur. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

Return Value

None

**C.2.23 «F» GdiCoStructCD**

Contains services for the creation of a structure type.



**Figure C.23 — Hierarchical diagram of GdiCoStructCD**

**C.2.23.1 GdiCoStructCD :: «F» gdiAddStructElementTypeCD()****Function Call**

```

gdiAddStructElementTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                eGDIDATATYPE eNewType
    /* in parameter
    GDI type of the structure element
    The value range includes the GDI Type enumeration. */
    inout                String sElementIdentifier
    /* in parameter
    contains the Element Identifier of the new struct element as string. The
    element name can be empty, if no Identifier Name is used. */
) : GdiCoDataTypeCD

```

**Function Description**

Adding of an additional element to a structure. The element type may be complex. By means of the return value the element type may be specified in more detail. If the basic type of the element is not allowed, an exception will be thrown.

The ordinal value of the first struct element is 1. The last element gets the highest ordinal value.

**Return Value**

Returns the reference to the type description of the element, if it is successful.

**C.2.23.2 GdiCoStructCD :: «F» gdiGetFirstStructTypeCD()****Function Call**

```

gdiGetFirstStructTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD

```

**Function Description**

Generates the reference to the first Type description within the struct. With this methode all sub types of the struct can be detected.

**Return Value**

Returns the reference (Pointer, Handle, Class) to the first sub type of the structure.

**C.2.23.3 GdiCoStructCD :: «F» gdiGetNextStructTypeCD()****Function Call**

```

gdiGetNextStructTypeCD
(

```

```

    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoStructCD refLastElement
    /* in parameter
    contains the reference to the last structure description object. */
    ) : GdiCoDataTypeCD

```

### Function Description

Generates the reference to the next Type description. With this method all sub types of the structure can be detected.

With this method all sub types of the structure can be detected.

### Return Value

Returns the reference (Pointer, Handle, Class) to the next sub type of the structure.

### **C.2.23.4 GdiCoStructCD :: «F» gdiGetStructElementDataTypeCDByOrdValue()**

#### Function Call

```

gdiGetStructElementDataTypeCDByOrdValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                short nOrdValue
    /* in parameter
    contains the selector of the requested struct member. */
    ) : GdiCoDataTypeCD

```

### Function Description

Returns the reference to a structure element. The basic type is returned as reference. A numeric value, which describes the position of the structure element within the structure, is used as selector. The actual type of the structure element (if unknown) may be detected by means of gdiGetType(). This reference may then be used as a reference to the detected data type description.

### Return Value

Returns the reference to a GdiCoDataTypeCD.

### **C.2.24 «F» GdiCoULongCD**

Contains services concerning the type unsigned Long for the definition of restrictions.

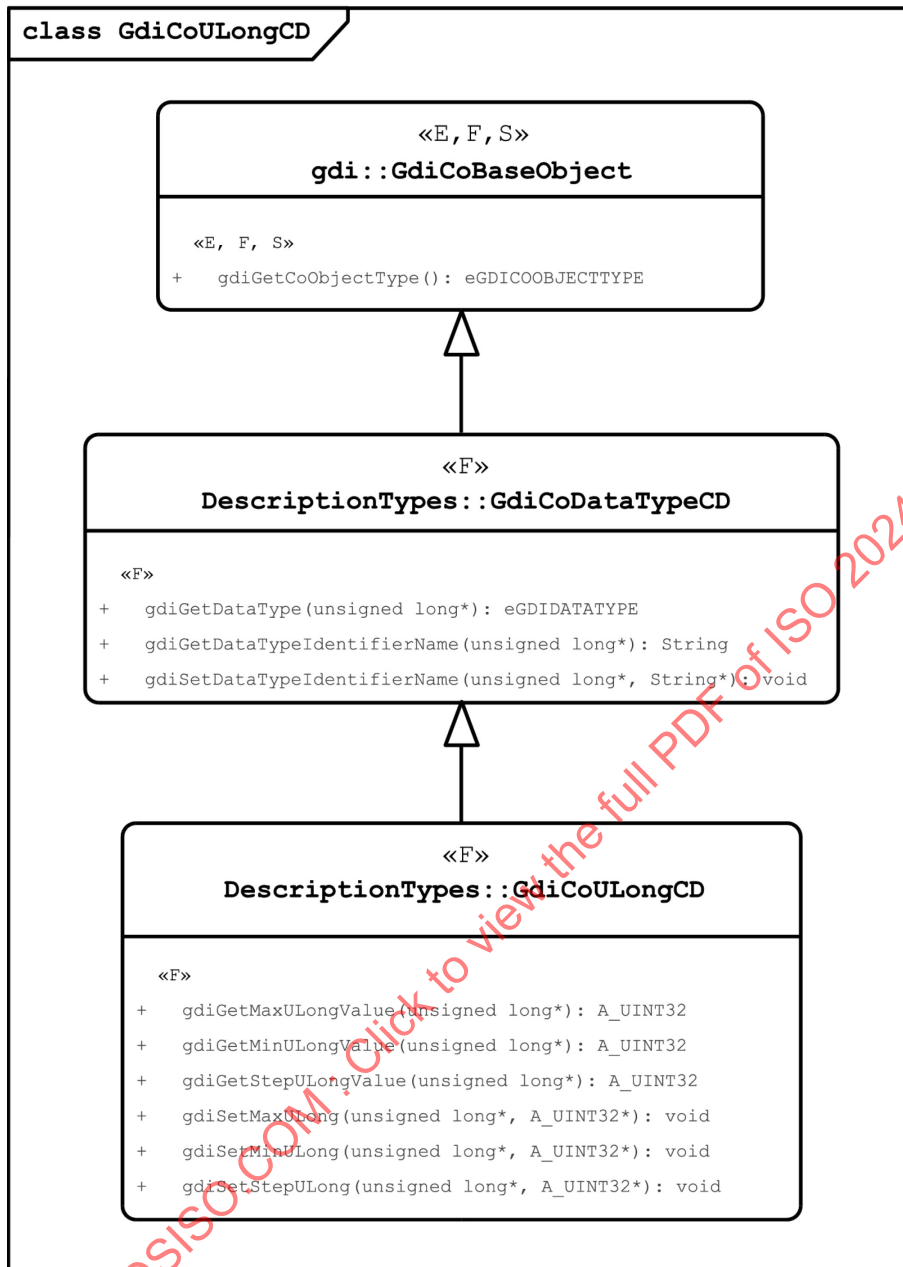


Figure C.24 — Hierarchical diagram of GdiCoULongCD

## C.2.24.1 GdiCoULongCD :: «F» gdiGetMaxULongValue()

Function Call

```

gdiGetMaxULongValue
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32
  
```

Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 4294967295.

Return Value

Returns the maximum value of a datum, based upon this type.

**C.2.24.2 GdiCoULongCD :: «F» gdiGetMinULongValue()**Function Call

```
gdiGetMinULongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32
```

Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 0.

Return Value

Returns the minimum value of a datum, based upon this type.

**C.2.24.3 GdiCoULongCD :: «F» gdiGetStepULongValue()**Function Call

```
gdiGetStepULongValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT32
```

Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

Return Value

Returns the step width of a datum, based upon this type.

**C.2.24.4 GdiCoULongCD :: «F» gdiSetMaxULong()**Function Call

```
gdiSetMaxULong
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```
Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          A_UINT32 ulLimitValue
/* new maximum value (< 4294967296) */
) : void
```

#### Function Description

Defines the maximum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

#### Return Value

None

### **C.2.24.5 GdiCoULongCD :: «F» gdiSetMinULong()**

#### Function Call

```
gdiSetMinULong
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_UINT32 ulLimitValue
  /* in parameter
  new minimum value (0) */
) : void
```

#### Function Description

Defines the minimum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

#### Return Value

None

### **C.2.24.6 GdiCoULongCD :: «F» gdiSetStepULong()**

#### Function Call

```
gdiSetStepULong
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_UINT32 ulLimitValue
  /* in parameter
  new step width (0 < LimitValue < 4294967295) */
) : void
```



Function Description

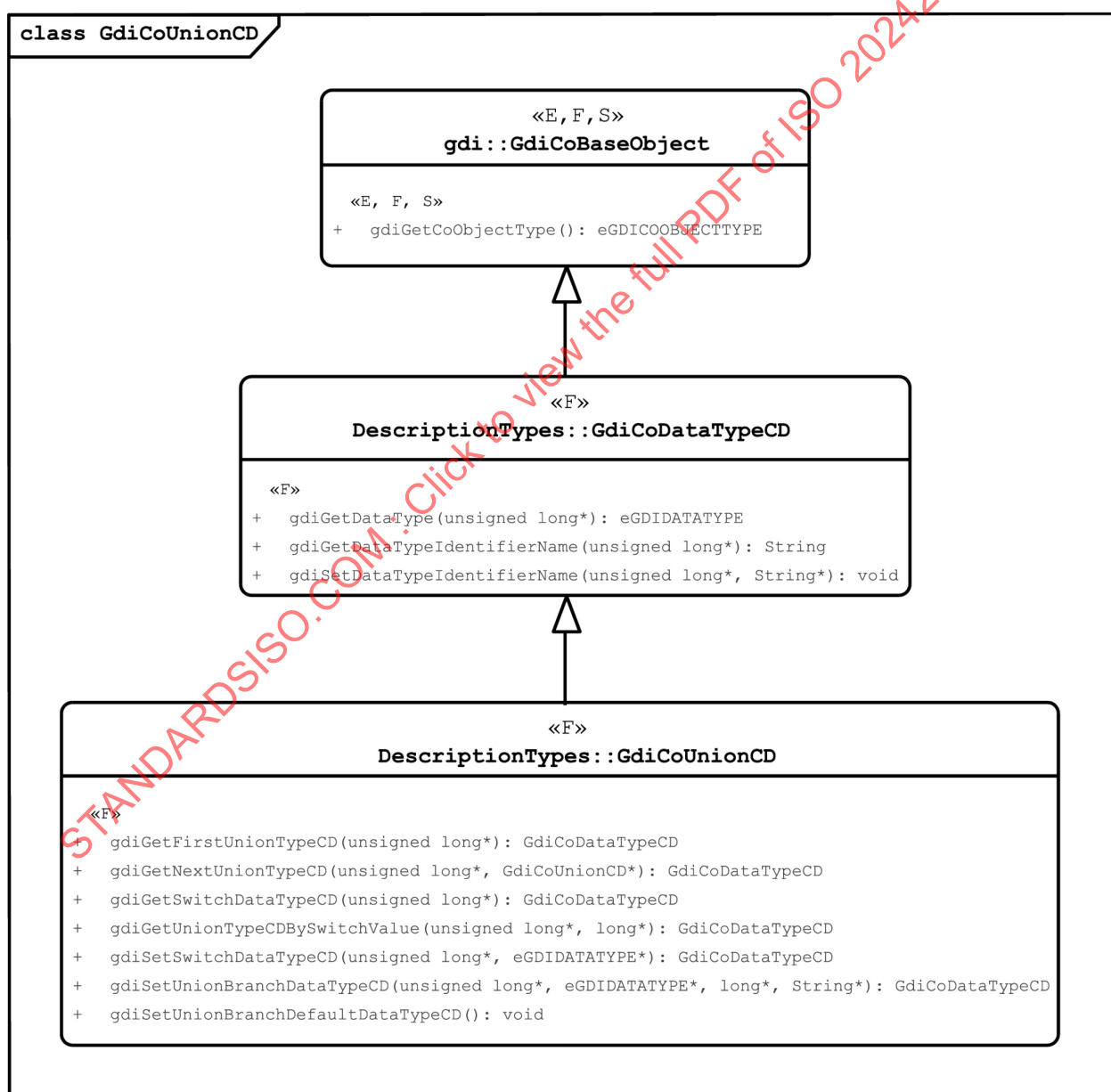
Defines the step width for valid data based upon this type between minimum and maximum. The minimum is the starting value. In case of wrong inputs, empty sets may occur. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

Return Value

None

**C.2.25 «F» GdiCoUnionCD**

Contains services for the creation and expansion of a Union type. This object reference is valid, till the creation or referencing the next sub description.



**Figure C.25 — Hierarchical diagram of GdiCoUnionCD**

**C.2.25.1 GdiCoUnionCD :: «F» gdiGetFirstUnionTypeCD()****Function Call**

```

gdiGetFirstUnionTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD

```

**Function Description**

Generates the reference to the first Type description within the Union. With this method all sub types of the union can be detected.

**Return Value**

Returns the reference (Pointer, Handle, Class) to the first sub type of the union.

**C.2.25.2 GdiCoUnionCD :: «F» gdiGetNextUnionTypeCD()****Function Call**

```

gdiGetNextUnionTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoUnionCD refLastElement
    /* in parameter
    contains the reference to the last Union description object. */
) : GdiCoDataTypeCD

```

**Function Description**

Generates the reference to the next Type description within the Union. With this method all sub types of the union can be detected.

**Return Value**

Returns the reference (Pointer, Handle, Class) to the next sub type of the union.

**C.2.25.3 GdiCoUnionCD :: «F» gdiGetSwitchDataTypeCD()****Function Call**

```

gdiGetSwitchDataTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD

```

Function Description

Generates a reference to GdiDescriptionDataType. The actual type of the Switch value is detected by the service gdiGetType() in relation to the returned reference.

Return Value

Returns the type of the Switch value if it is set.

**C.2.25.4 GdiCoUnionCD :: «F» gdiGetUnionTypeCDBySwitchValue()**Function Call

```
gdiGetUnionTypeCDBySwitchValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                long lSwitchValue
    /* in parameter
    Switch value of the union element, which is connected with this data
    type. */
) : GdiCoDataTypeCD
```

Function Description

Returns the reference to an overlay element of a Union. The basic type is returned as reference. The switch value is used as selector. The actual type of the union element (if unknown) may be detected by means of gdiGetType(). This reference may then be used as a reference to the detected data type.

Return Value

Returns the reference to a datum.

**C.2.25.5 GdiCoUnionCD :: «F» gdiSetSwitchDataTypeCD()**Function Call

```
gdiSetSwitchDataTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                eGDIDATATYPE eNewType
    /* in parameter
    GDI type of the element
    The value range includes both ASAM Datatype enumeration and GDI Type
    enumeration */
) : GdiCoDataTypeCD
```

Function Description

Defines the Switch type. This has to be a scalar type.

Return Value

Returns the reference to the type description of the elements if successful.

**C.2.25.6 GdiCoUnionCD :: «F» gdiSetUnionBranchDataTypeCD()****Function Call**

```

gdiSetUnionBranchDataTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                eGDIDATATYPE eNewType
    /* in parameter
    The value range includes both ASAM Datatype enumeration and GDI Type
    enumeration */
    inout                long lSwitchValue
    /* in parameter
    Switch value, which is connected with this data type. */
    inout                String sBranch
    /* in parameter
    contains the section name of the new Union Element as String. This String
    can be empty, is no Branch name is used. */
) : GdiCoDataTypeCD

```

**Function Description**

Adding of an additional element to a Union. The element type may be complex. By means of the return value the element type may be specified in more detail. If the basic type of the element is not allowed, a exception will be thrown.

**Return Value**

Returns the reference to the type description of the element if successful.

**C.2.25.7 GdiCoUnionCD :: «F» gdiSetUnionBranchDefaultDataTypeCD()****Function Call**

```

gdiSetUnionBranchDefaultDataTypeCD
(
) : void

```

**Function Description**

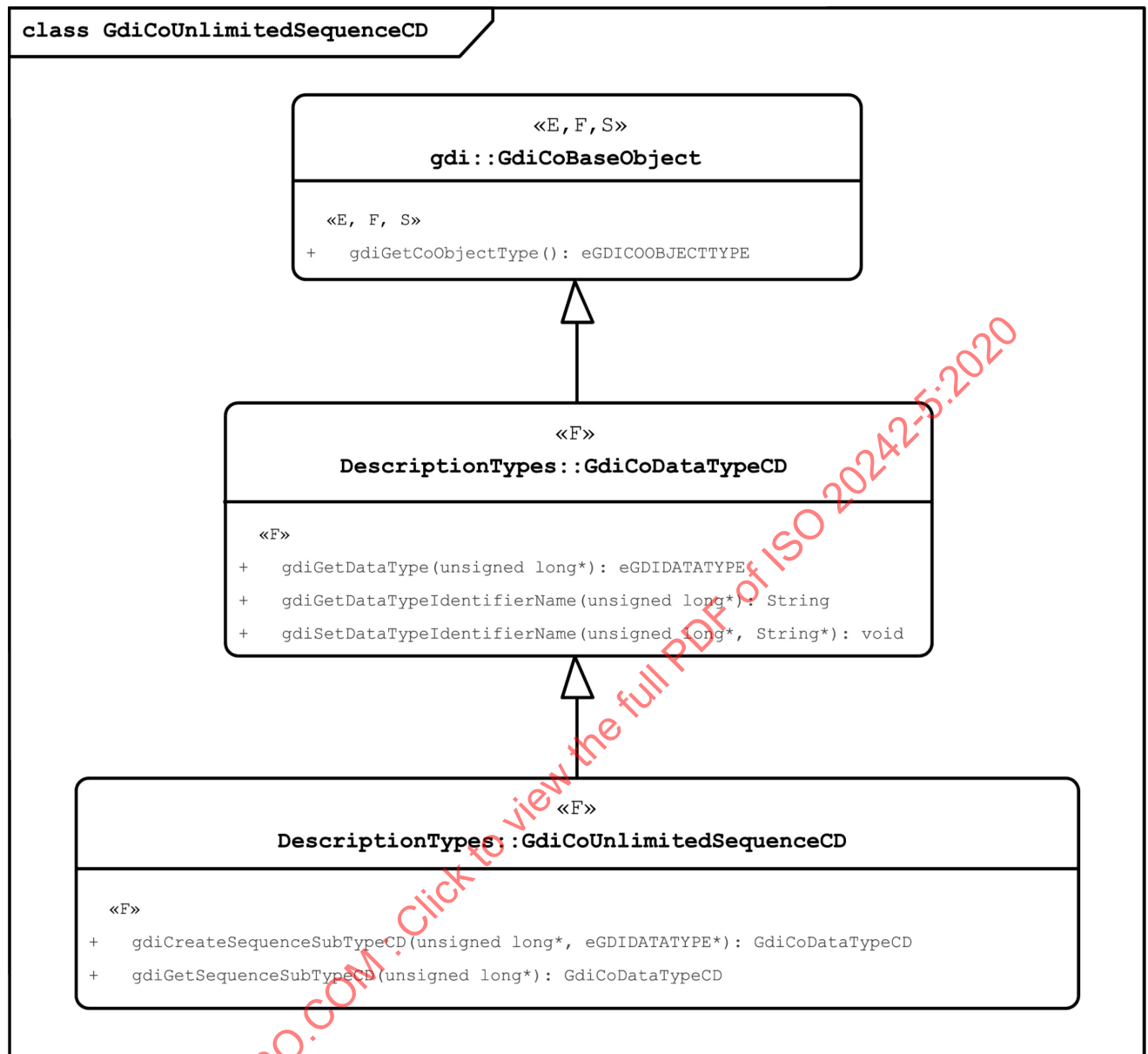
Set the default element of the Union. The element type may be complex. By means of the return value the element type may be specified in more detail. If the basic type of the element is not allowed, a exception will be thrown.

**Return Value**

None

**C.2.26 «F» GdiCoUnlimitedSequenceCD**

Description of the instance of an Unlimited Sequence type. This object reference is valid, till the creation or referencing the next sub description.



**Figure C.26 — Hierarchical diagram of GdiCoUnlimitedSequenceCD**

### C.2.26.1 GdiCoUnlimitedSequenceCD :: «F» gdiCreateSequenceSubTypeCD()

#### Function Call

```

gdiCreateSequenceSubTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                eGDIDATATYPE eNewType
    /* in parameter
    GDI type of the Sequence elements
    The value range includes both ASAM Datatype enumeration and GDI Type
    enumeration. */
) : GdiCoDataTypeCD
  
```

### Function Description

Describes the type and the number of elements of an Unlimited Sequence. The element type may be complex. By means of the return value the element type may be specified in more detail.

### Return Value

Returns the reference to the type description of the Sequence elements if successful.

## **C.2.26.2 GdiCoUnlimitedSequenceCD :: «F» gdiGetSequenceSubTypeCD()**

### Function Call

```
gdiGetSequenceSubTypeCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDataTypeCD
```

### Function Description

Returns the reference to the description of the sub data type.

### Return Value

Returns the reference to the Description of the sub type if present, a NIL reference otherwise.

## **C.2.27 «F» GdiCoUShortCD**

Contains services concerning the type unsigned Short for the definition of restrictions.

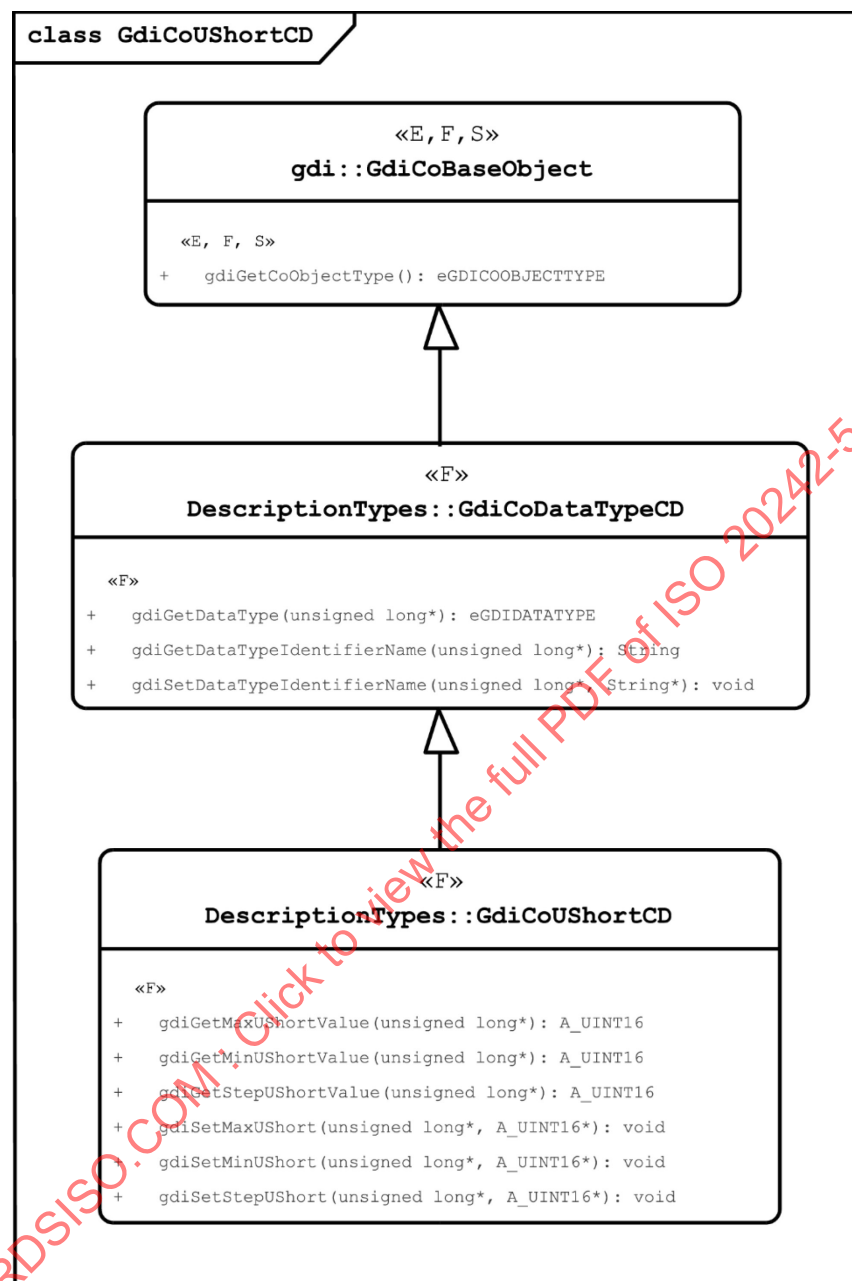


Figure C.27 — Hierarchical diagram of GdiCoUShortCD

**C.2.27.1 GdiCoUShortCD :: «F» gdiGetMaxUShortValue()****Function Call**

```

gdiGetMaxUShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT16
  
```

### Function Description

Returns the maximum value of a datum, based upon this type. If the maximum value has not been set explicitly, it takes the value 65535.

### Return Value

Returns the maximum value of a datum, based upon this type.

## **C.2.27.2 GdiCoUShortCD :: «F» gdiGetMinUShortValue()**

### Function Call

```
gdiGetMinUShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT16
```

### Function Description

Returns the minimum value of a datum, based upon this type. If the minimum value has not been set explicitly, it takes the value 0.

### Return Value

Returns the minimum value of a datum, based upon this type.

## **C.2.27.3 GdiCoUShortCD :: «F» gdiGetStepUShortValue()**

### Function Call

```
gdiGetStepUShortValue
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : A_UINT16
```

### Function Description

Returns the step width of a type, which defines valid elements of a datum starting from the minimum value. If the minimum value is not defined explicitly, it takes the value 1.

### Return Value

Returns the step width of a datum, based upon this type.

## **C.2.27.4 GdiCoUShortCD :: «F» gdiSetMaxUShort()**

### Function Call

```
gdiSetMaxUShort
(
    inout                unsigned long ulAppHnd
    /* in parameter
```



```

Identifier of the application returned by gdiCreateWorkspace(...) or by
gdiGetWorkspaceByName() */
inout          A_UINT16 unLimitValue
/* in parameter
new maximum value (< 32768) */
) : void

```

### Function Description

Defines the maximum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.27.5 GdiCoUShortCD :: «F» gdiSetMinUShort()**

### Function Call

```

gdiSetMinUShort
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_UINT16 unLimitValue
  /* in parameter
  new minimum value (0) */
) : void

```

### Function Description

Defines the minimum value for a datum based upon this type. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

### Return Value

None

## **C.2.27.6 GdiCoUShortCD :: «F» gdiSetStepUShort()**

### Function Call

```

gdiSetStepUShort
(
  inout          unsigned long ulAppHnd
  /* in parameter
  Identifier of the application returned by gdiCreateWorkspace(...) or by
  gdiGetWorkspaceByName() */
  inout          A_UINT16 unLimitValue
  /* in parameter
  new step width (0 < LimitValue < 65535) */
) : void

```

Function Description

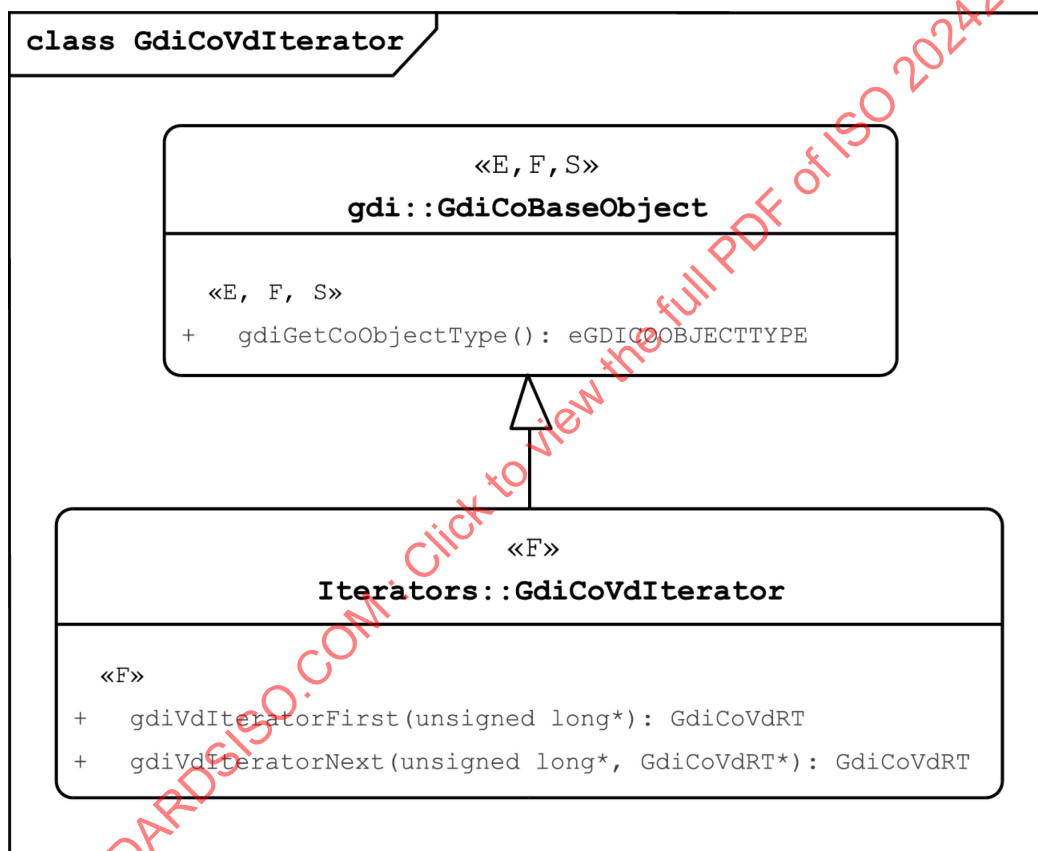
Defines the step width for valid data based upon this type between minimum and maximum. The minimum is the starting value. In case of wrong inputs, empty sets may occur. The value has no effect on the communication to the device driver. It is read by certain applications to obtain detailed type information.

Return Value

None

**C.2.28 «F» GdiCoVdIterator**

This interface contains the service for the application of a VD iterator, output of all application VD which have been set up within a Workspace.



**Figure C.28 — Hierarchical diagram of GdiCoVdIterator**

**C.2.28.1 GdiCoVdIterator :: «F» gdiVdIteratorFirst()**Function Call

```

gdiVdIteratorFirst
(
    inout          unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */

```

) : GdiCoVdRT

#### Function Description

Sets the internal pointer to the first VD within the workspace.

#### Return Value

Returns a reference to the first GdiCoVdRT if it exists, otherwise a NIL Reference returns.

### **C.2.28.2 GdiCoVdIterator :: «F» gdiVdIteratorNext()**

#### Function Call

```
gdiVdIteratorNext
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    inout                GdiCoVdRT refLastElement
    /* in parameter
    A reference to a element returned by gdiVDIteratorFirst or
    gdiVDIteratorNext. */
) : GdiCoVdRT
```

#### Function Description

Detects the reference to the current VD (Parameter) within the workspace. Sets the internal pointer to the next VD (if existing) and returns the reference.

#### Return Value

Returns a reference to the first GdiCoVdRT if it exists, otherwise a NIL Reference returns.

### **C.2.29 «F» GdiCoVdRT**

Contains all services for the creation and control of the states of an application VD.

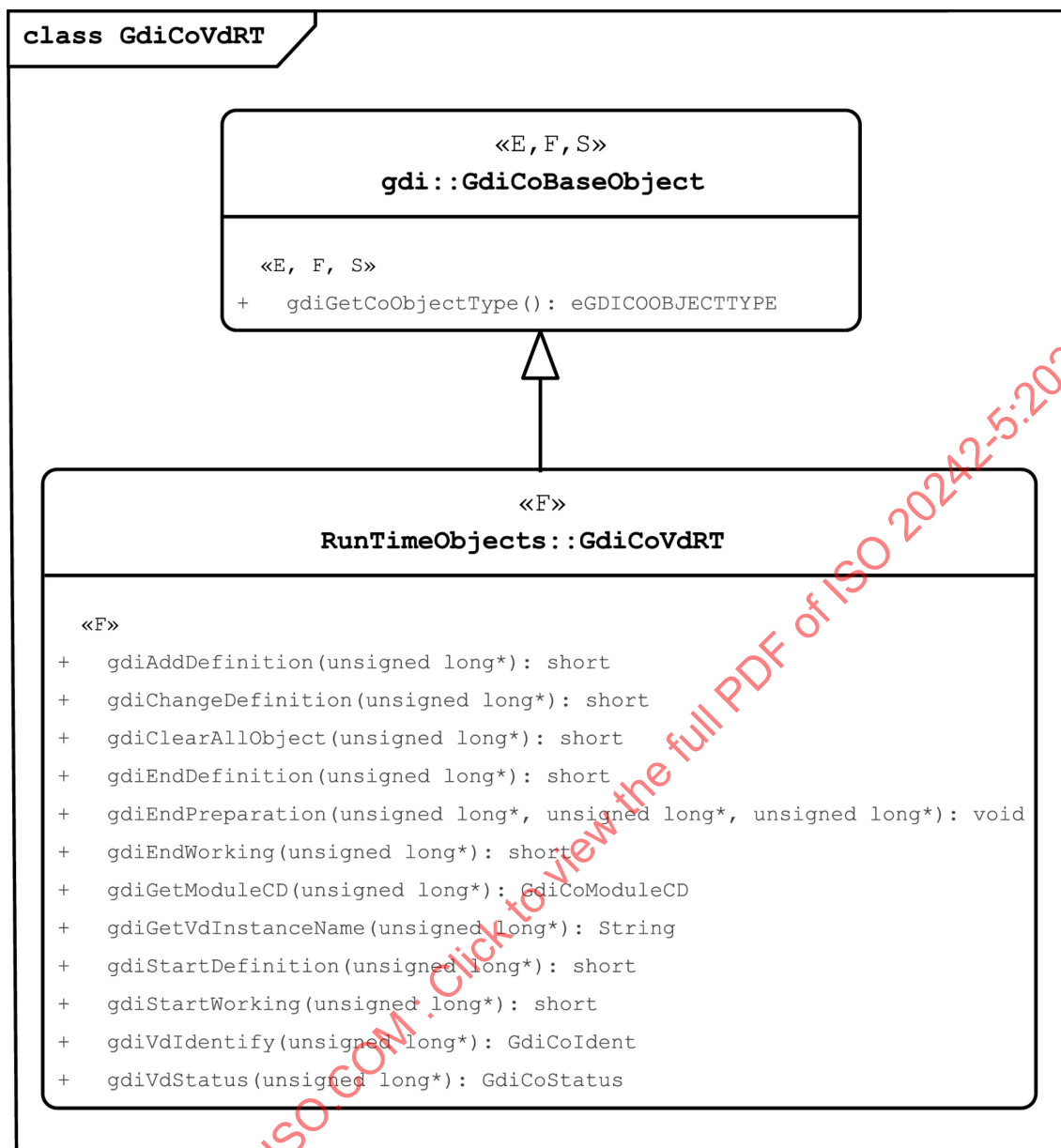


Figure C.29 — Hierarchical diagram of GdiCoVdRT

## C.2.29.1 GdiCoVdRT :: «F» gdiAddDefinition()

Function Call

```

gdiAddDefinition
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : short
  
```

Function Description

Transition of the application VD into the state Revise.

Return Value

== 0 if successful

!= 0 if otherwise

**C.2.29.2 GdiCoVdRT :: «F» gdiChangeDefinition()**Function Call

```
gdiChangeDefinition
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : short
```

Function Description

Transition of the application VD into the state Preparation.

Return Value

== 0 if successful

!= 0 if otherwise

**C.2.29.3 GdiCoVdRT :: «F» gdiClearAllObject()**Function Call

```
gdiClearAllObject
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : short
```

Function Description

Transition of the application VD into the state Initialized.

Return Value

== 0 if successful

!= 0 if otherwise

**C.2.29.4 GdiCoVdRT :: «F» gdiEndDefinition()**Function Call

```
gdiEndDefinition
(
    inout                unsigned long ulAppHnd
    /* in parameter
```

```
Identifier of the application returned by gdiCreateWorkspace(...) or by  
gdiGetWorkspaceByName() */  
) : short
```

#### Function Description

Transition of the application VD into the state Check.

#### Return Value

== 0 if successful  
!= 0 if otherwise

### C.2.29.5 GdiCoVdRT :: «F» gdiEndPreparation()

#### Function Call

```
gdiEndPreparation  
(  
    inout                unsigned long ulAppHnd  
    /* in parameter  
    Identifier of the application returned by gdiCreateWorkspace(...) or by  
    gdiGetWorkspaceByName() */  
    inout                unsigned long ulMsPollInterval  
    /* in parameter  
    contains the interval of polling the State of the Check Process. */  
    inout                unsigned long ulMsTimeOut  
    /* in parameter  
    contains the maximum waiting Time till the Check process be finished.  
    The global time out set by gdiCreateWorkspace will not be affected. */  
) : void
```

#### Function Description

This Method starts the Check process in the driver. The given configuration in the VD is validated by the driver.

The method succeeds if the Check Process is okay and Transition to working is performed.

The method fails if the given configuration within the VD is incorrect or a hardware fault is present.

#### Return Value

None

### C.2.29.6 GdiCoVdRT :: «F» gdiEndWorking()

#### Function Call

```
gdiEndWorking  
(  
    inout                unsigned long ulAppHnd  
    /* in parameter  
    Identifier of the application returned by gdiCreateWorkspace(...) or by  
    gdiGetWorkspaceByName() */  
) : short
```

Function Description

Transition of the application VD into the state Evaluation.

Return Value

== 0 if successful

!= 0 if otherwise

**C.2.29.7 GdiCoVdRT :: «F» gdiGetModuleCD()**Function Call

```
gdiGetModuleCD
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoModuleCD
```

Function Description

Generate a Reference to the Description of the VD as GdiCoModuleCD object.

Return Value

Returns the reference to GdiCoModuleCD.

**C.2.29.8 GdiCoVdRT :: «F» gdiGetVdInstanceName()**Function Call

```
gdiGetVdInstanceName
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : String
```

Function Description

Returns the VD instance name as String.

Return Value

Returns VD instance name as String.

**C.2.29.9 GdiCoVdRT :: «F» gdiStartDefinition()**Function Call

```
gdiStartDefinition
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
```

) : short

#### Function Description

Transition of the application VD into the state Preparation.

#### Return Value

== 0 if successful  
!= 0 if otherwise

### **C.2.29.10 GdiCoVdRT :: «F» gdiStartWorking()**

#### Function Call

```
gdiStartWorking
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : short
```

#### Function Description

Transition of the application VD into the state Working.

#### Return Value

== 0 if successful  
!= 0 if otherwise

### **C.2.29.11 GdiCoVdRT :: «F» gdiVdIdentify()**

#### Function Call

```
gdiVdIdentify
(
    inout                unsigned long ulAppHnd
    /* in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoIdent
```

#### Function Description

Effects the detection of the identification data of the device which is connected to the application VD.

#### Return Value

Returns GDIIDENT data corresponding with GDI specification.

### **C.2.29.12 GdiCoVdRT :: «F» gdiVdStatus()**

#### Function Call

```
gdiVdStatus
(
    inout                unsigned long ulAppHnd
```



```
/* in parameter  
Identifier of the application returned by gdiCreateWorkspace(...) or by  
gdiGetWorkspaceByName() */  
) : GdiCoStatus
```

#### Function Description

Effects the detection of the state data of the virtual and of the physical data of the physical device which is connected to the application VD.

#### Return Value

Returns GDISTATUS data corresponding with the GDI specification.

STANDARDSISO.COM : Click to view the full PDF of ISO 2042-5:2020

Annex D  
(normative)

Reference guide — Enums and status/ident information

D.1 General Information

The enumerations and constants defined in this annex shall be used by the coordinator to inform the user about details in several services. The following detailed description is completed by Figures D.1 and D.2 for an overview on classes and their methods

D.2 Detailed description of of enums and status/ident information

D.2.1 «S,E,F» GdiCoStatus

Returns the state of the VD.

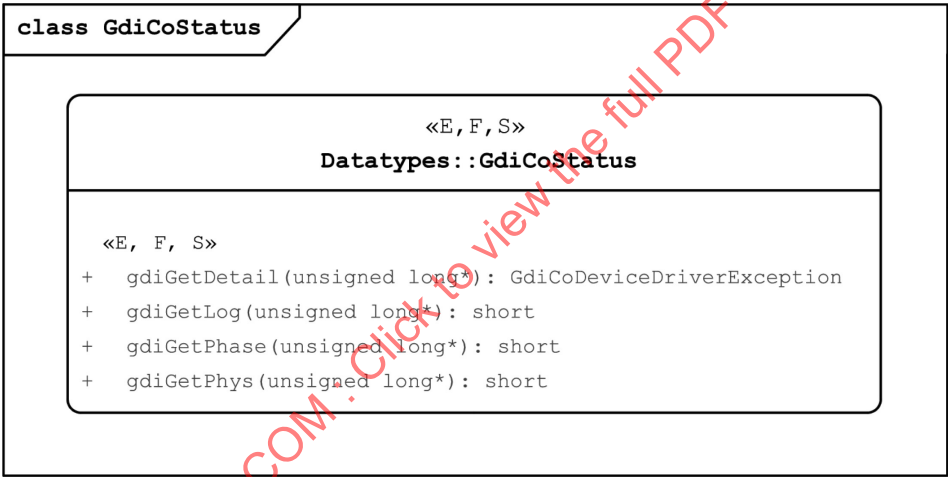


Figure D.1 — Hierarchical diagram of GdiCoStatus

D.2.1.1 GdiCoStatus :: «S,E,F» gdiGetDetail()

Function Call

```
gdiGetDetail
(
    inout unsigned long ulAppHnd
    /*in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : GdiCoDeviceDriverException
```

Function Description

Gets error class, which describes the current error of a VD.

Return Value

Returns the Error class.

**D.2.1.2 GdiCoStatus :: «S,E,F» gdiGetLog()**Function Call

```
gdiGetLog
(
    inout                unsigned long ulAppHnd
    /*in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : short
```

Function Description

Gets log value (It describes the reduced or unrestricted use of API functions.)

Return Value

Returns the log value.

**D.2.1.3 GdiCoStatus :: «S,E,F» gdiGetPhase()**Function Call

```
gdiGetPhase
(
    inout                unsigned long ulAppHnd
    /*in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : short
```

Function Description

Gets phase value (It describes the current transition state of the VD.)

Return Value

Returns the phase value.

**D.2.1.4 GdiCoStatus :: «S,E,F» gdiGetPhys()**Function Call

```
gdiGetPhys
(
    inout                unsigned long ulAppHnd
    /*in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : short
```

### Function Description

Gets phys value (It describes the reduced or unrestricted use of a VD.)

### Return Value

Returns the phys value.

## D.2.2 «S,E,F» GdiCoIdent

Returns specific details to a VD.

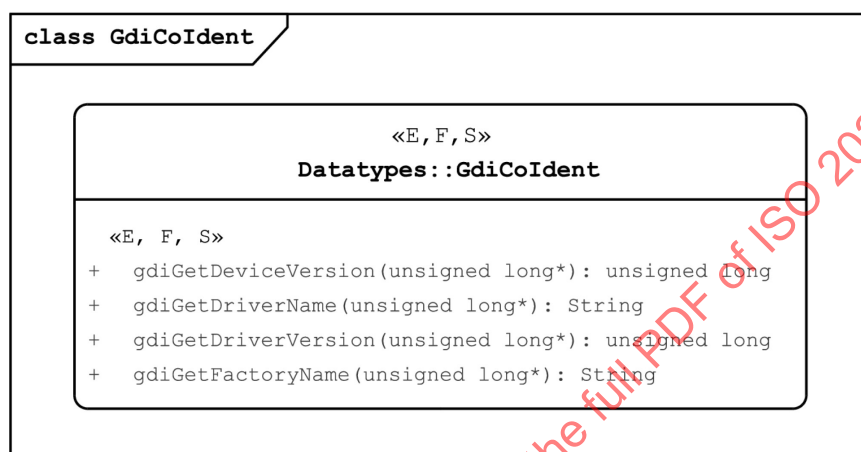


Figure D.2 — Hierarchical diagram of GdiCoIdent

### D.2.2.1 GdiCoIdent :: «S,E,F» gdiGetDeviceVersion()

#### Function Call

```

gdiGetDeviceVersion
(
    inout                unsigned long ulAppHnd
    /*in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
) : unsigned long
  
```

#### Function Description

Gets the device version number.

#### Return Value

Returns the device version number.

### D.2.2.2 GdiCoIdent :: «S,E,F» gdiGetDriverName()

#### Function Call

```

gdiGetDriverName
(
  
```

```

    inout                unsigned long ulAppHnd
    /*in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    ) : String

```

### Function Description

Gets the name of the driver.

### Return Value

Returns the name of the driver.

## **D.2.2.3 GdiColIdent :: «S,E,F» gdiGetDriverVersion()**

### Function Call

```

gdiGetDriverVersion
(
    inout                unsigned long ulAppHnd
    /*in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    ) : unsigned long

```

### Function Description

Gets the driver version number.

### Return Value

Returns the version number of the GDI driver.

## **D.2.2.4 GdiColIdent :: «S,E,F» gdiGetFactoryName()**

### Function Call

```

gdiGetFactoryName
(
    inout                unsigned long ulAppHnd
    /*in parameter
    Identifier of the application returned by gdiCreateWorkspace(...) or by
    gdiGetWorkspaceByName() */
    ) : String

```

### Function Description

Gets the name of the device manufacturer.

### Return Value

Returns the name of the factory.

## **D.2.3 «enum» eACCEPTOCCURRED**

eACCEPTOCCURRED returns the information, if an Accept has been carried out since the last test or gdiWriteValue (see Table D.1).

**Table D.1 — Members of eACCEPTOCCURRED**

| Name      | Notes              | Constraints and tags |
|-----------|--------------------|----------------------|
| eACCEPT   | Accept occurred    | Default: 1           |
| eNOACCEPT | no Accept occurred | Default: 0           |

**D.2.4 «enum» eGDIACCESSRIGHTS**

eGDIACCESSRIGHTS defines the rights of access on a communication object (see Table D.2).

**Table D.2 — Members of eGDIACCESSRIGHTS**

| Name       | Notes | Constraints and tags |
|------------|-------|----------------------|
| eREADONLY  |       | Default: 1           |
| eREADWRITE |       | Default: 0           |

**D.2.5 «enum» eGDIACCESSSTATE**

eGDIACCESSSTATE specifies the access state of a workspace instance (see Table D.3).

**Table D.3 — Members of eGDIACCESSSTATE**

| Name      | Notes                                                                                 | Constraints and tags |
|-----------|---------------------------------------------------------------------------------------|----------------------|
| eNOT_USED | The workspace is not used by any application. The workspace is ready for login.       | Default: 0           |
| eUSED     | The workspace is used by any other application. The workspace is not ready for login. | Default: 1           |

**D.2.6 «enum» eGDIBYTEORDER**

eGDIBYTEORDER defines Alignment and byteorder for Data Stream (see Table D.4).

**Table D.4 — Members of eGDIBYTEORDER**

| Name           | Notes              | Constraints and tags |
|----------------|--------------------|----------------------|
| eLITTLE_ENDIAN | Lowest Byte First  | Default: 0           |
| eBIG_ENDIAN    | Highest Byte First | Default: 1           |

**D.2.7 «enum» eGDICOIFKIND**

eGDICOIFKIND describes the expected Coordinator Interface type by the application (see Table D.5).

0 = eSMART

1 = eEXTENDED