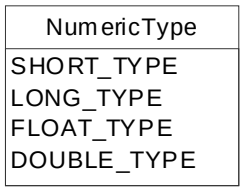
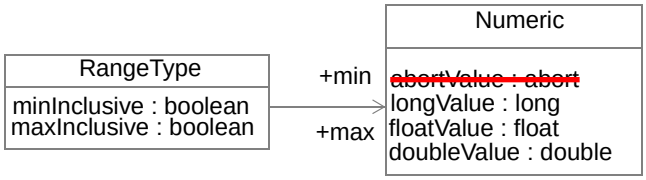
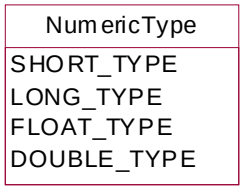
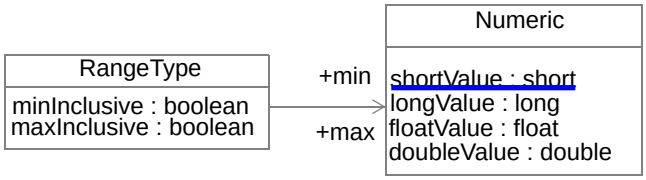
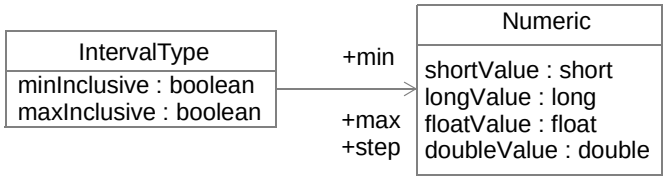
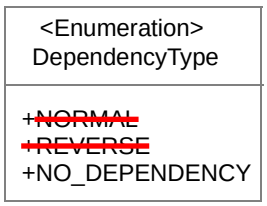
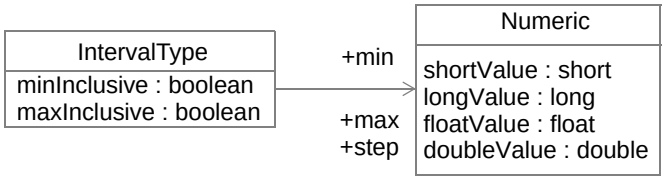
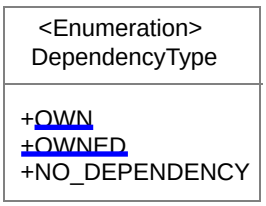
NumericType	 This enumeration is used by structures **RangeType** and **IntervalType**. It defines numeric types used to specify the bounds of intervals or ranges, and the step between interval values.
RangeType	 Data structure representing a range of values that can be assigned to the attribute AllowedValues of **Parameter**. • **min** – The lower bound of the range. • **max** – The upper bound of the range. • **minInclusive** – A boolean value showing if the lower bound value is included in the range. • **maxInclusive** – A boolean value showing if the upper bound value is included in the range. Example: The range ((int) 0, (int) 20, false, true) defines values {1,2,…20}.

NumericType	 This enumeration is used by structures **RangeType** and **IntervalType**. It defines numeric types used to specify the bounds of intervals or ranges, and the step between interval values.
RangeType	 Data structure representing a range of values that can be assigned to the attribute AllowedValues of **Parameter**. • **min** – The lower bound of the range. • **max** – The upper bound of the range. • **minInclusive** – A boolean value showing if the lower bound value is included in the range. • **maxInclusive** – A boolean value showing if the upper bound value is included in the range. Example: The range ((int) 0, (int) 20, false, true) defines values {1,2,…20}.

IntervalType	 Data structure representing an interval of values that can be assigned to the attribute AllowedValues of **Parameter** defined as interval. • **min** – The lower bound of the interval. • **max** – The upper bound of the interval. • **minInclusive** – A boolean value showing if the lower bound value is included in the interval. • **maxInclusive** – A boolean value showing if the upper bound value is included in the interval. • **step** – The step between the values in the interval. Example: The interval ((int) 0, (int) 20, false, true, 5) defines values {5,10,15,20}.
ParameterList	A list of **Parameter** structures.
DependencyType	Data type used to specify relation between elements in an organization. The value indicates if one side of an **Organization** depends on the other side. If the **Organization** represents dependency relationship, it also indicates which side depends on which side. Enumeration **DependencyType** includes three possible forms of dependency. 

1. It is beyond the scope of this specification to define the format of identifiers and the algorithm to generate them, because they are implementation dependent. For example, some applications may use standardized schemes such as the UUID [2], others may use proprietary ones. Different SDO systems need to follow an agreed scheme for identifiers to maintain the interoperability between SDOs.

IntervalType	 Data structure representing an interval of values that can be assigned to the attribute AllowedValues of **Parameter** defined as interval. • **min** – The lower bound of the interval. • **max** – The upper bound of the interval. • **minInclusive** – A boolean value showing if the lower bound value is included in the interval. • **maxInclusive** – A boolean value showing if the upper bound value is included in the interval. • **step** – The step between the values in the interval. Example: The interval ((int) 0, (int) 20, false, true, 5) defines values {5,10,15,20}.
ParameterList	A list of **Parameter** structures.
DependencyType	Data type used to specify relation between elements in an organization. The value indicates if one side of an **Organization** depends on the other side. If the **Organization** represents dependency relationship, it also indicates which side depends on which side. Enumeration **DependencyType** includes three possible forms of dependency. 

1. It is beyond the scope of this specification to define the format of identifiers and the algorithm to generate them, because they are implementation dependent. For example, some applications may use standardized schemes such as the UUID [2], others may use proprietary ones. Different SDO systems need to follow an agreed scheme for identifiers to maintain the interoperability between SDOs.

2.2.3 SDOSystemElement

SDOSystemElement
- organizations : OrganizationList

SDOSystemElement is the base class of the classes that represent SDO system elements. It is used to indicate that its subclasses represent any system elements running on the SDO environments. A representative example of the SDO's system elements is SDOs. SDO is defined as a subclass of **SDOSystemElement** in Section 2.2."SDO" on page 2-84. And, there are system elements that are running on the SDO environment but are not SDOs as specified in this document. Examples of those elements that are not SDO include human users and locations. It is left to future specifications to define those elements as additional subclasses of **SDOSystemElement**.

*<**Attributes**>*

Attribute	Type	Description
ownedOrganizations	**OrganizationList**	A list of **Organization** objects that **SDOSystemElement** has.

2.2.4 SDO

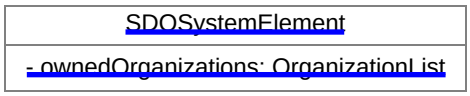
SDO
- id : UniqueIdentifier - sdoType : String - deviceProfile : DeviceProfile - serviceProfiles : ServiceProfileList - configurationProfile : ConfigurationProfile - organizations : OrganizationList - status : Status

The class **SDO** defines a set of common properties for hardware device and software component representations.

SDOs are characterized by properties. Due to the nature of the SDO different SDOs can have different properties. Which properties are monitorable and which are configurable is implementation dependent. Properties that can be monitored are provided through the monitoring interface. Properties that can be configured are provided through the configuration. A property can be both configurable and monitorable. Where and how these properties are stored is implementation dependent.

Services provided by the SDO can have their own properties that define the behavior of the service. These properties can also be provided to be configured. Some sort of mechanism must be provided to indicate the difference between service and the SDO properties.

2.2.3 SDOSystemElement

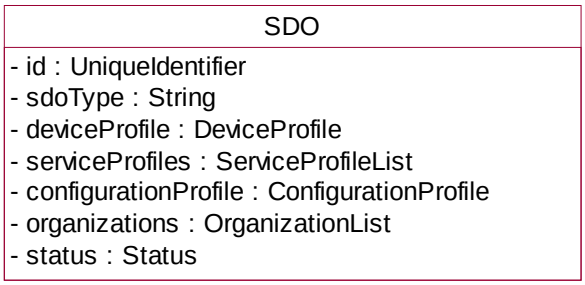


SDOSystemElement is the base class of the classes that represent SDO system elements. It is used to indicate that its subclasses represent any system elements running on the SDO environments. A representative example of the SDO's system elements is SDOs. SDO is defined as a subclass of **SDOSystemElement** in Section 2.2."SDO" on page 2-84. And, there are system elements that are running on the SDO environment but are not SDOs as specified in this document. Examples of those elements that are not SDO include human users and locations. It is left to future specifications to define those elements as additional subclasses of **SDOSystemElement**.

<Attributes>

Attribute	Type	Description
ownedOrganizations	**OrganizationList**	A list of **Organization** objects that **SDOSystemElement** has.

2.2.4 SDO



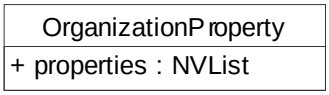
The class **SDO** defines a set of common properties for hardware device and software component representations.

SDOs are characterized by properties. Due to the nature of the SDO different SDOs can have different properties. Which properties are monitorable and which are configurable is implementation dependent. Properties that can be monitored are provided through the monitoring interface. Properties that can be configured are provided through the configuration. A property can be both configurable and monitorable. Where and how these properties are stored is implementation dependent.

Services provided by the SDO can have their own properties that define the behavior of the service. These properties can also be provided to be configured. Some sort of mechanism must be provided to indicate the difference between service and the SDO properties.

- Location (owner)-SDO (members): When one or more SDOs (members) are operating in a specific location (owner), the **organization** represents topology pattern (3). For example, multiple PDAs in the same place (e.g., a room) have equal relationships among them to communicate with each other.

~~2.2.6 OrganizationProperty~~

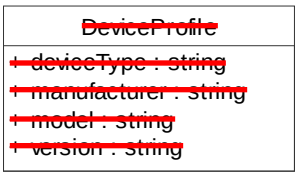


~~OrganizationProperty~~ contains the properties of an **Organization**. An Organization has zero or one (at most one) instance of **OrganizationProperty**.

<Attributes>

Attribute	Type	Description
properties	**NVList**	A set of properties of an **Organization**. The property values contained in this attribute are implementation dependent. Examples include the identifier of an **Organization** and the time when an **Organization** is established.

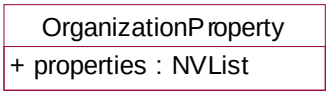
~~2.2.7 DeviceProfile~~



DeviceProfile defines the properties of a device that an SDO represents.

- Location (owner)-SDO (members): When one or more SDOs (members) are operating in a specific location (owner), the **organization** represents topology pattern (3). For example, multiple PDAs in the same place (e.g., a room) have equal relationships among them to communicate with each other.

2.2.6 OrganizationProperty

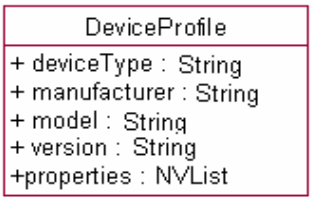


OrganizationProperty contains the properties of an **Organization**. An Organization has zero or one (at most one) instance of **OrganizationProperty**.

<Attributes>

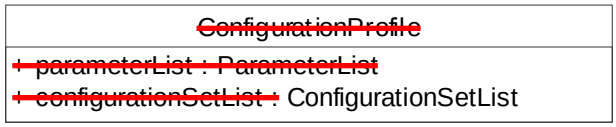
Attribute	Type	Description
properties	**NVList**	A set of properties of an **Organization**. The property values contained in this attribute are implementation dependent. Examples include the identifier of an **Organization** and the time when an **Organization** is established.

2.2.7 DeviceProfile



DeviceProfile defines the properties of a device that an SDO represents.

2.2.10 ConfigurationProfile

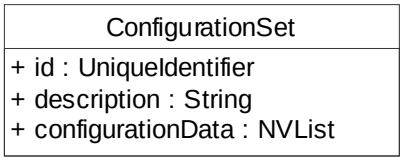


ConfigurationProfile contains a set of properties to configure an SDO.

*<**Attributes**>*

Attribute	Type	Description
parameters	**ParameterList**	A list of **Parameter** that represents the kinds of properties to configure an SDO.
configurationSets	**ConfigurationSetList**	A list of **configurationSet** (described below) objects that represents a set of properties with their values to configure an SDO.

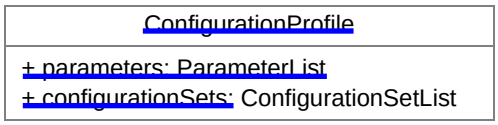
- Data type definition: **ConfigurationSet**



Parameter defines the data type of a variable.

Attribute	Type	Description
id	**UniqueIdentifier**	Identifier of the set of configuration data stored in **ConfigurationProfile**. This can be used to activate the stored configuration.
description	**String**	Descriptive information for configuration data.
configurationData	**NVList**	A set of configuration data. This is used to configure an SDO.

2.2.10 ConfigurationProfile

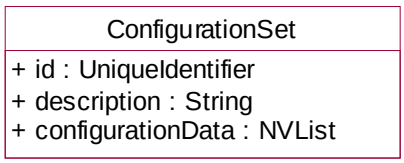


ConfigurationProfile contains a set of properties to configure an SDO.

*<**Attributes**>*

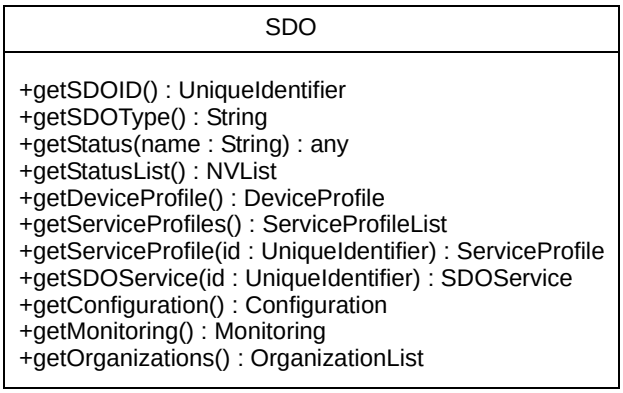
Attribute	Type	Description
parameters	**ParameterList**	A list of **Parameter** that represents the kinds of properties to configure an SDO.
configurationSets	**ConfigurationSetList**	A list of **configurationSet** (described below) objects that represents a set of properties with their values to configure an SDO.

- Data type definition: **ConfigurationSet**



Parameter defines the data type of a variable.

Attribute	Type	Description
id	**UniqueIdentifier**	Identifier of the set of configuration data stored in **ConfigurationProfile**. This can be used to activate the stored configuration.
description	**String**	Descriptive information for configuration data.
configurationData	**NVList**	A set of configuration data. This is used to configure an SDO.



(1) ~~*getSDOID ( ) : String*~~

This operation returns **id** of the SDO .

Parameter	Type	Description
<return>	**UniqueIdentifier**	**id** of the SDO defined in the resource data model.

Exceptions

This operation throws SDOException with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(2) *+ getSDOType ( ) : String*

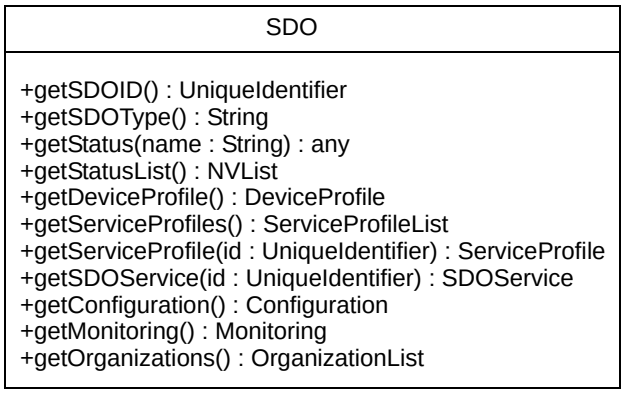
This operation returns **sdoType** of the SDO.

Parameter	Type	Description
<return>	**String**	**sdoType** of the SDO defined in the resource data model.

Exceptions

This operation throws SDOException with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.



(1) *+getSDOID ( ) : UniqueIdentifier*

This operation returns **id** of the SDO .

Parameter	Type	Description
<return>	**UniqueIdentifier**	**id** of the SDO defined in the resource data model.

Exceptions

This operation throws SDOException with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(2) *+ getSDOType ( ) : String*

This operation returns **sdoType** of the SDO.

Parameter	Type	Description
<return>	**String**	**sdoType** of the SDO defined in the resource data model.

Exceptions

This operation throws SDOException with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.

- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(3) + *getStatus (name : String) : any*

This operation returns the value of the specified status parameter.

Parameter	Type	Description
Name	**String**	One of the parameters defining the 'status' of an SDO.
<return>	**Any**	The value of the specified status parameter.

Exceptions

This operation throws **SDOException** with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InvalidParameter - if the parameter defined by 'name' is null or does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(4) + *getStatusList( ) : NVList*

This operation returns an NVlist describing the status of an SDO.

Parameter	Type	Description
<return>	**NVList**	The actual status of an SDO.

Exceptions

This operation throws **SDOException** with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(5) + *getDeviceProfile ( ) : DeviceProfile*

This operation returns the **DeviceProfile** of the SDO. If the SDO does not represent any hardware device, then a **DeviceProfile** with empty values are returned.

- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(3) + *getStatus (name : String) : any*

This operation returns the value of the specified status parameter.

Parameter	Type	Description
Name	**String**	One of the parameters defining the 'status' of an SDO.
<return>	**any**	The value of the specified status parameter.

Exceptions

This operation throws **SDOException** with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InvalidParameter - if the parameter defined by 'name' is null or does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(4) + *getStatusList( ) : NVList*

This operation returns an NVlist describing the status of an SDO.

Parameter	Type	Description
<return>	**NVList**	The actual status of an SDO.

Exceptions

This operation throws **SDOException** with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(5) + *getDeviceProfile ( ) : DeviceProfile*

This operation returns the **DeviceProfile** of the SDO. If the SDO does not represent any hardware device, then a **DeviceProfile** with empty values are returned.

.

Parameter	Type	Description
<return>	**DeviceProfile**	The **DeviceProfile** of the SDO.

Exceptions

This operation throws **SDOException** with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(6) + *getServiceProfiles ( ) : ServiceProfileList*

This operation returns a list of **ServiceProfile**s that the SDO has. If the SDO does not provide any service, then an empty list is returned.

Parameter	Type	Description
<return>	~~**ServiceProfilesList**~~	List of **ServiceProfile**s of all the services the SDO is providing.

Exceptions

This operation throws SDOException with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(7) + *getServiceProfile (id : UniqueIdentifier) : ServiceProfile*

This operation returns the **ServiceProfile** that is specified by the argument "**id**."

Parameter	Type	Description
id	**UniqueIdentifier**	The identifier referring to one of the **ServiceProfile**s.
<return>	**ServiceProfile**	The profile of the specified service.

Exceptions

This operation throws SDOException with one of the following types.

.

Parameter	Type	Description
<return>	**DeviceProfile**	The **DeviceProfile** of the SDO.

Exceptions

This operation throws **SDOException** with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(6) + *getServiceProfiles ( ) : ServiceProfileList*

This operation returns a list of **ServiceProfile**s that the SDO has. If the SDO does not provide any service, then an empty list is returned.

Parameter	Type	Description
<return>	**ServiceProfileList**	List of **ServiceProfile**s of all the services the SDO is providing.

Exceptions

This operation throws SDOException with one of the following types.

- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(7) + *getServiceProfile (id : UniqueIdentifier) : ServiceProfile*

This operation returns the **ServiceProfile** that is specified by the argument "**id**."

Parameter	Type	Description
id	**UniqueIdentifier**	The identifier referring to one of the **ServiceProfile**s.
<return>	**ServiceProfile**	The profile of the specified service.

Exceptions

This operation throws SDOException with one of the following types.

2.3.4.1 Usage: SDO interface

The section describes examples about how to use the operations of the SDO interface to get the SDO identifier. As an example, the operation to get SDO parameter **id** is shown in Figure 2-4.

In Figure 2-4, sdo1 makes requests to sdo2 to acquire the parameter. The example of invocation of the operations that enable to get other interfaces by using SDO interface is described in Section 2.3.7, “Monitoring Interface,” on page 2-35.

Figure 2-4 Sequence Chart: SDO operation concern with data resource

Message 1: sdo1 requests identifier of the sdo2. This identifier is described using String type.

Message 2: sdo2 returns the value of identifier.

2.3.5 Configuration Interface

Configuration interface provides operations to add or remove data specified in resource data model. These operations provide functions to change **DeviceProfile**, **ServiceProfile**, **ConfigurationProfile**, and **Organization**. This specification does not address access control or security aspects. Access to operations that modifies or removes profiles should be controlled depending upon the application.

Different configurations can be stored for simple and quick activation. Different predefined configurations are stored as different **ConfigurationSet**s or configuration profile. A **ConfigurationSet** stores the value of all properties assigned for the particular configuration along with its unique id and description to identify and describe the configuration respectively.

Operations in the configuration interface help manage these **ConfigurationSets**.

getConfigurationSets () - This operation returns a list of all **ConfigurationSets** of the SDO. If no predefined **ConfigurationSets** exist, then empty list is returned.

2.3.4.1 Usage: SDO interface

This section describes examples about how to use the operations of the SDO interface to get the SDO identifier. As an example, the operation to get SDO parameter **id** is shown in Figure 2-4.

In Figure 2-4, sdo1 makes requests to sdo2 to acquire the parameter. The example of invocation of the operations that enable to get other interfaces by using SDO interface is described in Section 2.3.7, “Monitoring Interface,” on page 2-35.

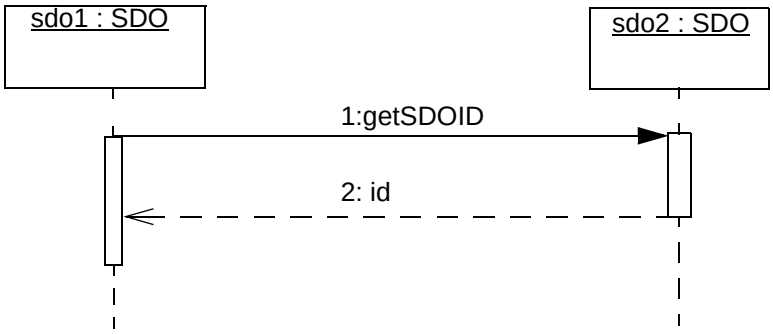


Figure 2-4 Sequence Chart: SDO operation concern with data resource

Message 1: sdo1 requests identifier of the sdo2. This identifier is described using String type.

Message 2: sdo2 returns the value of identifier.

2.3.5 Configuration Interface

Configuration interface provides operations to add or remove data specified in resource data model. These operations provide functions to change **DeviceProfile**, **ServiceProfile**, **ConfigurationProfile**, and **Organization**. This specification does not address access control or security aspects. Access to operations that modifies or removes profiles should be controlled depending upon the application.

Different configurations can be stored for simple and quick activation. Different predefined configurations are stored as different **ConfigurationSet**s or configuration profile. A **ConfigurationSet** stores the value of all properties assigned for the particular configuration along with its unique id and description to identify and describe the configuration respectively.

Operations in the configuration interface help manage these **ConfigurationSets**.

getConfigurationSets () - This operation returns a list of all **ConfigurationSets** of the SDO. If no predefined **ConfigurationSets** exist, then empty list is returned.

addConfigurationSet(..) - This operation adds a new **ConfigurationSet**. The given **ConfigurationSet** contains the property names and their desired configuration values. This **ConfigurationSet** may not provide desired value for all configurable properties. If so, then the values of the configuration properties that are not given are the current values of these properties.

removeConfigurationSet(..) - This operation removes the given **ConfigurationSet**. Once the configuration set is removed, this configuration set cannot be activated.

activateConfigurationSet(..) - This operation activates the specified stored **ConfigurationSet**s. This means that the configuration properties of the SDO are changed as the values of these properties specified in the stored **ConfigurationSet**. In other words, values of the specified **ConfigurationSet** are now copied to the active configuration.

getConfigurationSet(..) - This operation gets the **ConfigurationSet** specified by the parameter.

getActiveConfigurationSet() - This operation gets the current active **ConfigurationSet** if any.

Configuration
+setDeviceProfile(dProfile : DeviceProfile) : Boolean +setServiceProfile(sProfile : ServiceProfile) : Boolean +addOrganization(organization : Organization) : Boolean +removeServiceProfile(id : UniqueIdentifier) : Boolean +removeOrganization(organizationID : UniqueIdentifier) : Boolean +getConfigurationParameters() : ParameterList +getConfiguration ParameterValues() : NVList +getConfigurationParameterValue(name : String) : any +setConfigurationParameter(name : String, value : any) : Boolean +getConfigurationSets() : ConfigurationSetList +getConfigurationSet(configurationSetID : UniqueIdentifier) : ConfigurationSet +getActiveConfigurationSet() : ConfigurationSet +addConfigurationSet(configurationSet : ConfigurationSet) : Boolean +setConfigurationSetValues(configurationSetID : UniqueIdentifier) : Boolean +removeConfigurationSet(configurationSetID : UniqueIdentifier) : Boolean +activateConfigurationSet(configurationSetID : UniqueIdentifier) : Boolean

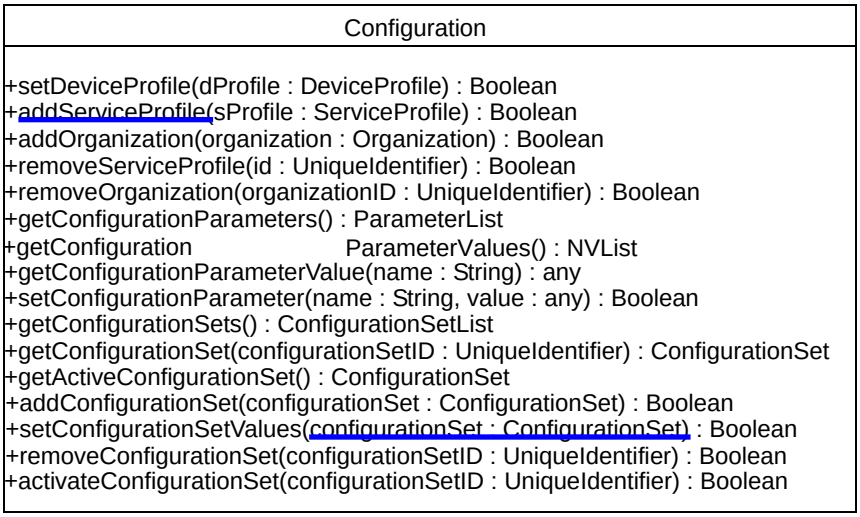
addConfigurationSet(..) - This operation adds a new **ConfigurationSet**. The given **ConfigurationSet** contains the property names and their desired configuration values. This **ConfigurationSet** may not provide desired value for all configurable properties. If so, then the values of the configuration properties that are not given are the current values of these properties.

removeConfigurationSet(..) - This operation removes the given **ConfigurationSet**. Once the configuration set is removed, this configuration set cannot be activated.

activateConfigurationSet(..) - This operation activates the specified stored **ConfigurationSet**s. This means that the configuration properties of the SDO are changed as the values of these properties specified in the stored **ConfigurationSet**. In other words, values of the specified **ConfigurationSet** are now copied to the active configuration.

getConfigurationSet(..) - This operation gets the **ConfigurationSet** specified by the parameter.

getActiveConfigurationSet() - This operation gets the current active **ConfigurationSet** if any.



Exceptions

This operation throws SDOException with one of the following exception types.

- SDONotExists - if the target SDO does not exist.
- InvalidParameter - if the argument "**configurationSet**" is null, or if one of the attributes defining "**configurationSet**" is invalid, or if the specified identifier of the configuration set already exists.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(14) + setConfigurationSetValues(configurationSet: ConfigurationSet): Boolean

This operation modifies the specified **ConfigurationSet** of an SDO.

Parameter	Type	Description
<return>	**Boolean**	A flag indicating if the **ConfigurationSet** was modified successfully. 'true' - The **ConfigurationSet** was modified successfully. 'false' - The **ConfigurationSet** could not be modified successfully.

Exceptions

This operation throws **SDOException** with one of the following exception types.

- InvalidParameter - if the parameter '**configurationSetID**' is null or if there is no **ConfigurationSet** stored with such id. This exception is also raised if one of the attributes defining **ConfigurationSet** is not valid.
- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(15) + removeConfigurationSet (configurationSetID : UniqueIdentifier) :Boolean

This operation removes a **ConfigurationSet** from the **ConfigurationProfile**.

Parameter	Type	Description
configurationSetID	**UniqueIdentifier**	The **ConfigurationSet** which is removed.

Exceptions

This operation throws SDOException with one of the following exception types.

- SDONotExists - if the target SDO does not exist.
- InvalidParameter - if the argument "**configurationSet**" is null, or if one of the attributes defining "**configurationSet**" is invalid, or if the specified identifier of the configuration set already exists.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(14) + setConfigurationSetValues(configurationSet: ConfigurationSet): Boolean

This operation modifies the specified **ConfigurationSet** of an SDO.

Parameter	Type	Description
configurationSet	**ConfigurationSet**	The **ConfigurationSet** that is modified.
<return>	**Boolean**	A flag indicating if the **ConfigurationSet** was modified successfully. 'true' - The **ConfigurationSet** was modified successfully. 'false' - The **ConfigurationSet** could not be modified successfully.

Exceptions

This operation throws **SDOException** with one of the following exception types.

- InvalidParameter - if the parameter '**configurationSetID**' is null or if there is no **ConfigurationSet** stored with such id. This exception is also raised if one of the attributes defining **ConfigurationSet** is not valid.
- SDONotExists - if the target SDO does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(15) + removeConfigurationSet (configurationSetID : UniqueIdentifier) :Boolean

This operation removes a **ConfigurationSet** from the **ConfigurationProfile**.

Parameter	Type	Description
configurationSetID	**UniqueIdentifier**	The **ConfigurationSet** which is removed.

<return>	**Boolean**	If the operation was successfully completed.

Exceptions

This operation throws SDOException with one of the following exception types.

- SDONotExists - if the target SDO does not exist.
- InvalidParameter - if the arguments "**configurationSetID**" is null, or if the object specified by the argument "**configurationSetID**" does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(16) + activateConfigurationSet (~~configID~~ : UniqueIdentifier) : Boolean

This operation activates one of the stored **ConfigurationSets** in the **ConfigurationProfile**.

Parameter	Type	Description
~~configID~~	**UniqueIdentifier**	Identifier of **ConfigurationSet** to be activated.
<return>	**Boolean**	If the operation was successfully completed.

Exceptions

This operation throws **SDOException** with one of the following exception types.

- SDONotExists - if the target SDO does not exist.
- InvalidParameter - if the argument ("**configID**") is null or there is no configuration set with identifier specified by the argument.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

2.3.5.2 Usage: Configuration

As an example, the ~~message sequence chart~~ for the case of profile acquisition is shown in Figure 2-5. And, as an example of parameter acquisition, the sequence of "**getConfigurationParameters()**" is shown in Figure 2-6. First, the configuring SDO (sdo1) gets the **Configuration** object by invocation of operations of the SDO that is configured (sdo2). And the sequences of the other operations that belong to the **Configuration** interface as shown in Figure 2-5 and Figure 2-6.

<return>	**Boolean**	If the operation was successfully completed.

Exceptions

This operation throws SDOException with one of the following exception types.

- SDONotExists - if the target SDO does not exist.
- InvalidParameter - if the arguments "**configurationSetID**" is null, or if the object specified by the argument "**configurationSetID**" does not exist.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(16) + activateConfigurationSet (configurationSetID : UniqueIdentifier) :Boolean

This operation activates one of the stored **ConfigurationSets** in the **ConfigurationProfile**.

Parameter	Type	Description
configurationSetID	**UniqueIdentifier**	Identifier of **ConfigurationSet** to be activated.
<return>	**Boolean**	If the operation was successfully completed.

Exceptions

This operation throws **SDOException** with one of the following exception types.

- SDONotExists - if the target SDO does not exist.
- InvalidParameter - if the argument ("**configID**") is null or there is no configuration set with identifier specified by the argument.
- NotAvailable - if the target SDO is reachable but cannot respond.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

2.3.5.2 Usage: Configuration

As an example, the sequence diagram for the case of profile acquisition is shown in Figure 2-5. And, as an example of parameter acquisition, the sequence of "**getConfigurationParameters()**" is shown in Figure 2-6. First, the configuring SDO (sdo1) gets the **Configuration** object by invocation of operations of the SDO that is configured (sdo2). And the sequences of the other operations that belong to the **Configuration** interface as shown in Figure 2-5 and Figure 2-6.

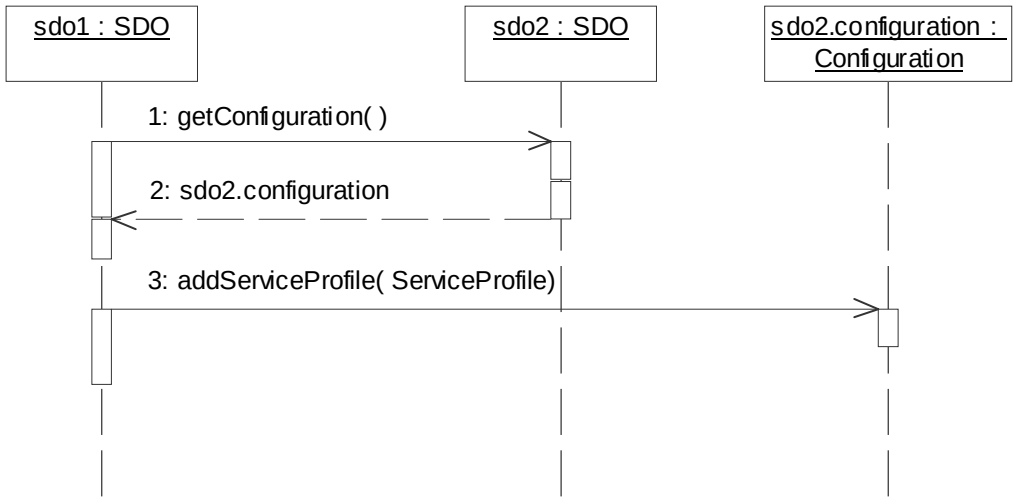


Figure 2-5 Sequence Chart: Configuration: Add Service Profile

Message 1: the configuring SDO (sdo1) gets the object implementing the **Configuration** of the SDO that is being configured (sdo2).

Message 2: sdo2 returns the object sdo2.configuration that implements the **Configuration**.

Message 3: sdo1 adds the **ServiceProfile** object to sdo2.

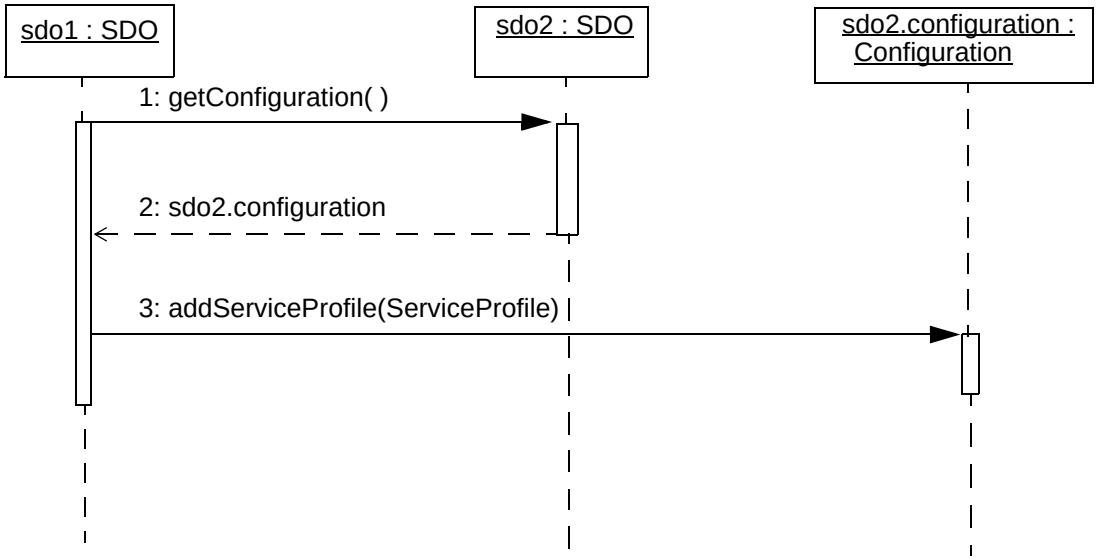


Figure 2-5 Sequence Diagram: Configuration: Add Service Profile

Message 1: the configuring SDO (sdo1) gets the object implementing the **Configuration** of the SDO that is being configured (sdo2).

Message 2: sdo2 returns the object sdo2.configuration that implements the **Configuration**.

Message 3: sdo1 adds the **ServiceProfile** object to sdo2.

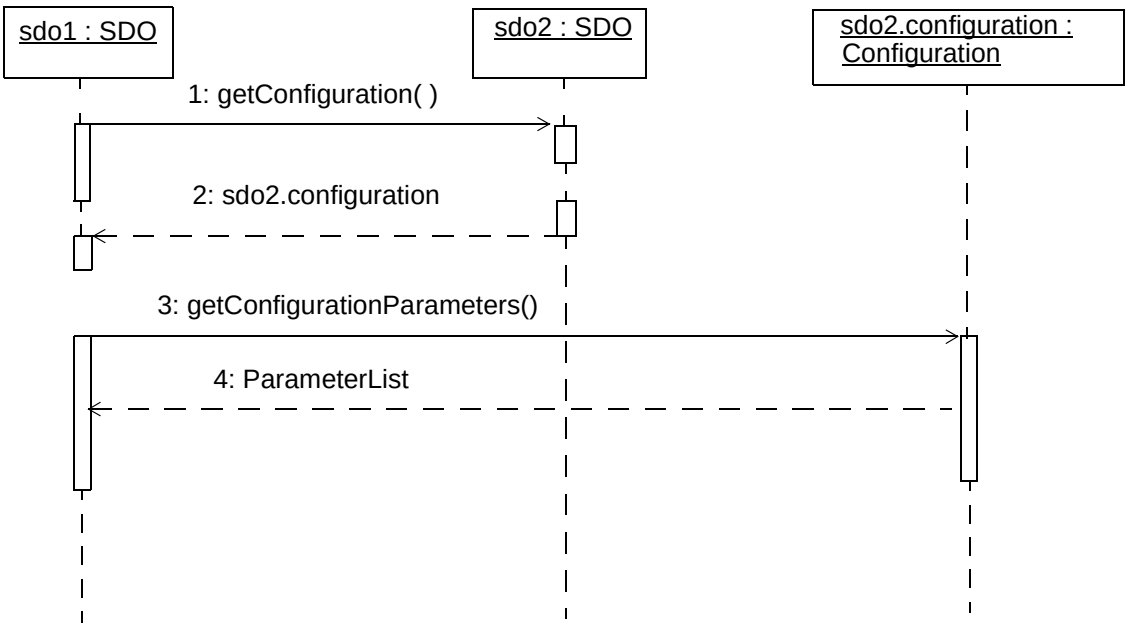


Figure 2-6 Sequence ~~Chart:~~ Configuration

Message 1: the configuring SDO (sdo1) gets the object implementing the **Configuration** of the SDO that is being configured (sdo2).

Message 2: sdo2 returns the object sdo2.configuration that implements the **Configuration**.

Message 3: sdo1 requests parameter list of sdo2.

Message 4: sdo2 returns the parameters as list of names and values of properties represented as **NVList** object.

~~2.3.6 SDOService~~ Interface

SDOService is the interface used to define an operation(s) through which an SDO provides its service(s). An **SDOService** object may define an operation for a single service or may define multiple operations for different services, each of which is represented by a **ServiceProfile** object. It is implementation dependent how to design **SDOService** objects.

~~2.3.7 Monitoring~~ Interface

Each SDO is characterized by properties. These properties can depict the current state of an SDO (subject to monitoring) and can be used to control the SDO behavior (subject to configuration). Each SDO can have different numbers and different kinds of attributes. It is dependent upon the actual implementation which properties are provided by an SDO.

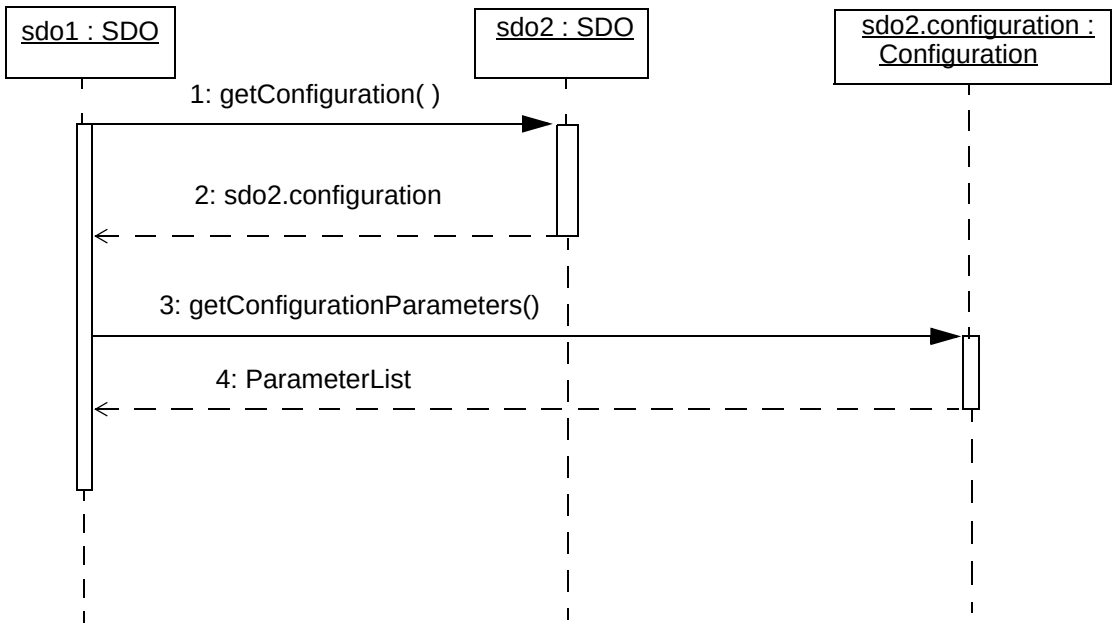


Figure 2-6 Sequence Diagram: Configuration

Message 1: the configuring SDO (sdo1) gets the object implementing the **Configuration** of the SDO that is being configured (sdo2).

Message 2: sdo2 returns the object sdo2.configuration that implements the **Configuration**.

Message 3: sdo1 requests parameter list of sdo2.

Message 4: sdo2 returns the parameters as list of names and values of properties represented as **NVList** object.

2.3.6 *SDOService Interface*

SDOService is the interface used to define an operation(s) through which an SDO provides its service(s). An **SDOService** object may define an operation for a single service or may define multiple operations for different services, each of which is represented by a **ServiceProfile** object. It is implementation dependent how to design **SDOService** objects.

2.3.7 *Monitoring Interface*

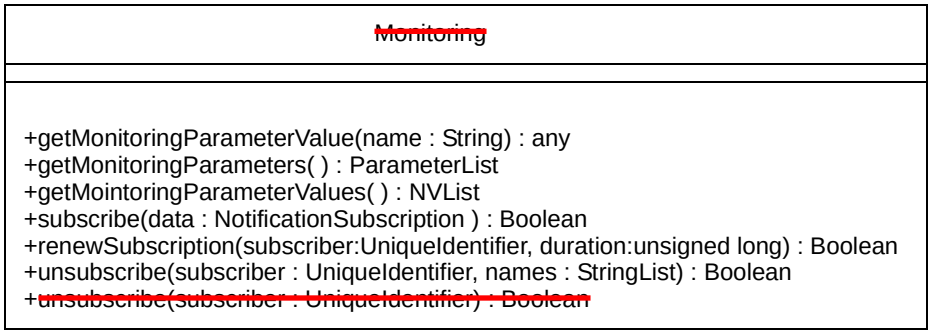
Each SDO is characterized by properties. These properties can depict the current state of an SDO (subject to monitoring) and can be used to control the SDO behavior (subject to configuration). Each SDO can have different numbers and different kinds of attributes. It is dependent upon the actual implementation which properties are provided by an SDO.

The **Monitoring** interface provides mechanisms to monitor the properties of an SDO. Each SDO implementing the **Monitoring** interface must specify the properties that can be monitored.

The properties of an SDO can be monitored mainly in two different ways: by *polling* and by *subscription*.

Polling is the simpler way of monitoring. The observer requests the current values of the properties it is interested in. The SDO that wants to monitor one or more properties must send a request message to the particular SDO. The **Monitoring** interface provides the functions (**getMonitoringParameterValues()**, and **getMonitoringParameterValue()**) that support the monitoring by polling. The monitoring by polling is described in detail in Section 2.3.7.4.1, “Monitoring by Polling,” on page 2-43.

Using *subscription* an observing SDO is notified about changes of monitored properties. The observing SDO has to be an SDO that is to be monitored. According to the subscription the monitored SDO notifies the subscriber using its **Callback** interface (see Section 2.3.7.5, “NotificationCallback Interface,” on page 2-49). The Monitoring interface supports the subscription through appropriate functions (**subscribe()**, **renewSubscription()**, **unsubscribe(), unsubscribeAll()**). The monitoring by subscription is described in detail in Section 2.3.7.4.2, “Monitoring by Subscription,” on page 2-45.



The **Monitoring** interface is optional. As mentioned earlier each SDO specifies the properties that can be monitored. If an SDO does not want to provide any of its properties to be monitored, it can do so and in this case it does not need to implement the **Monitoring** interface.

2.3.7.1 Data Structures Defined for Monitoring Interface

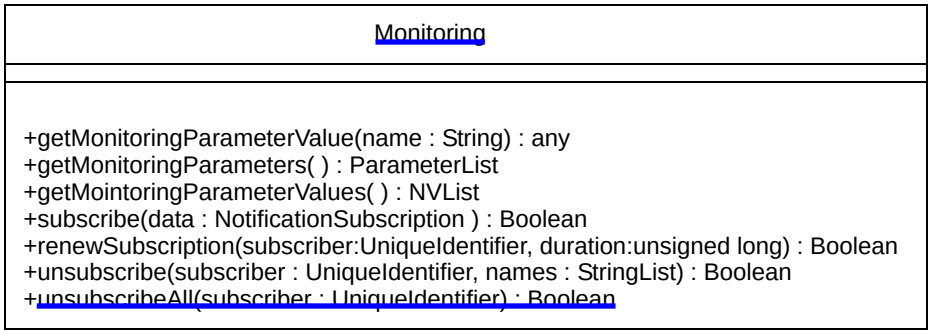
Various data structures are defined to implement the **Monitoring** interface. This section defines all such data structures including the general data structures and data structures required only for the **Monitoring** interface. All such data structures are listed in Table 2-1 and described in detail individually later on.

The **Monitoring** interface provides mechanisms to monitor the properties of an SDO. Each SDO implementing the **Monitoring** interface must specify the properties that can be monitored.

The properties of an SDO can be monitored mainly in two different ways: by *polling* and by *subscription*.

Polling is the simpler way of monitoring. The observer requests the current values of the properties it is interested in. The SDO that wants to monitor one or more properties must send a request message to the particular SDO. The **Monitoring** interface provides the functions (**getMonitoringParameterValues()**, and **getMonitoringParameterValue()**) that support the monitoring by polling. The monitoring by polling is described in detail in Section 2.3.7.4.1, "Monitoring by Polling," on page 2-43.

Using *subscription* an observing SDO is notified about changes of monitored properties. The observing SDO has to be an SDO that is to be monitored. According to the subscription the monitored SDO notifies the subscriber using its **Callback** interface (see Section 2.3.7.5, "NotificationCallback Interface," on page 2-49). The Monitoring interface supports the subscription through appropriate functions (**subscribe()**, **renewSubscription()**, **unsubscribe(), unsubscribeAll()**). The monitoring by subscription is described in detail in Section 2.3.7.4.2, "Monitoring by Subscription," on page 2-45.



The **Monitoring** interface is optional. As mentioned earlier each SDO specifies the properties that can be monitored. If an SDO does not want to provide any of its properties to be monitored, it can do so and in this case it does not need to implement the **Monitoring** interface.

2.3.7.1 Data Structures Defined for Monitoring Interface

Various data structures are defined to implement the **Monitoring** interface. This section defines all such data structures including the general data structures and data structures required only for the **Monitoring** interface. All such data structures are listed in Table 2-1 and described in detail individually later on.

Table 2-1 Detailed Description of Data Structures Required for **Monitoring** Interface

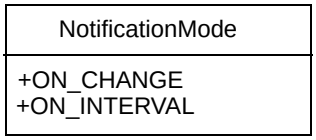
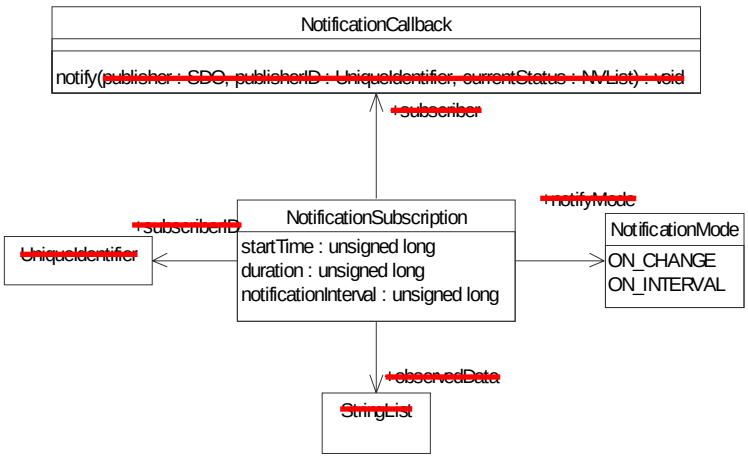
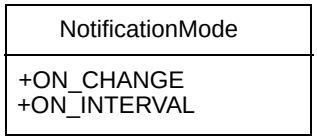
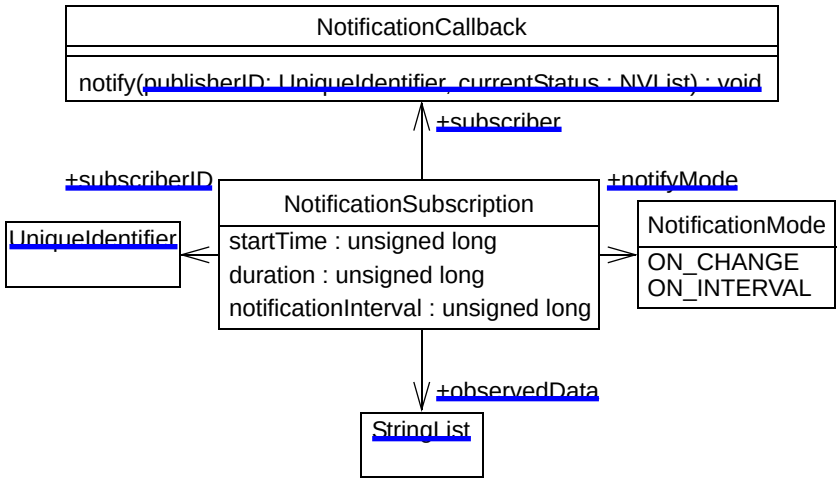
NotificationMode	Possible notification modes while subscribing to monitoring properties.  **ON_CHANGE** - To be notified of the subscribed monitoring property every time the value of the parameter changes. **ON_INTERVAL** - To be notified of the subscribed monitoring property on the specified interval of time. That is, if a subscription to some property is made in this mode, the notification is sent to the subscribing SDO only at the specified time with the current value.
NotificationSubscription	
	
	This structure outlines the details of the subscription message. This message is received and evaluated by the SDO providing properties for monitoring. Depending on the notification mode, the subscriber (the message sender) will receive appropriate messages containing the property value periodically or if the property's value changes.

Table 2-1 Detailed Description of Data Structures Required for **Monitoring** Interface

NotificationMode	Possible notification modes while subscribing to monitoring properties.  **ON_CHANGE** - To be notified of the subscribed monitoring property every time the value of the parameter changes. **ON_INTERVAL** - To be notified of the subscribed monitoring property on the specified interval of time. That is, if a subscription to some property is made in this mode, the notification is sent to the subscribing SDO only at the specified time with the current value.
NotificationSubscription	
	
	This structure outlines the details of the subscription message. This message is received and evaluated by the SDO providing properties for monitoring. Depending on the notification mode, the subscriber (the message sender) will receive appropriate messages containing the property value periodically or if the property's value changes.

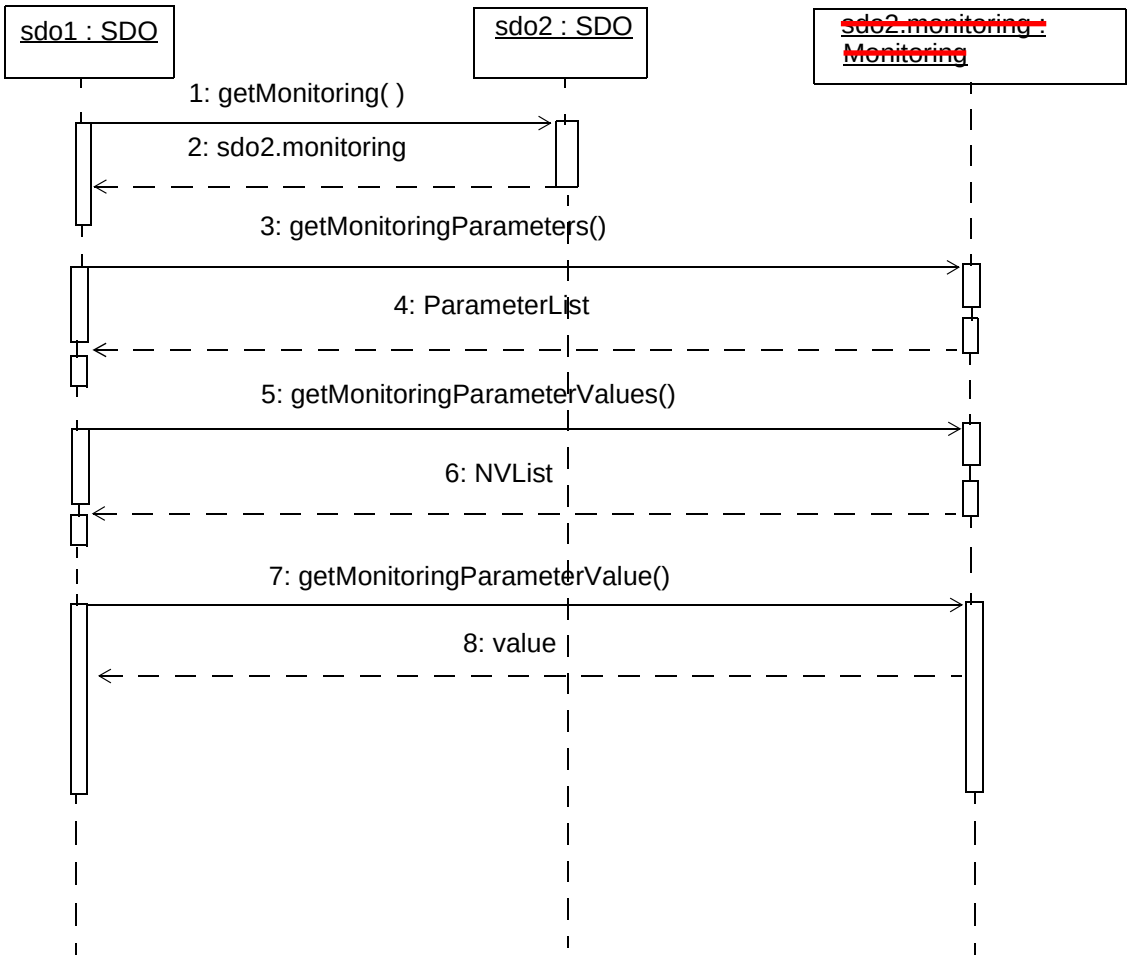


Figure 2-7 Sequence Chart: Monitoring

Message 1: the monitoring SDO (sdo1) gets the object implementing the **Monitoring** interface of the monitored SDO (sdo2).

Message 2: sdo2 returns the object sdo2.monitoring that implements the **Monitoring** interface.

Message 3: sdo1 requests the list of all monitoring properties of sdo2.

Message 4: sdo2 sends response, containing the list of monitoring properties specifying the sdo2.

Message 5: sdo1 requests the current values of monitoring parameters of the sdo2.

Message 6: sdo2 returns the data requested in message 5 as list of names and current values of properties represented as **NVList** object. Knowing the respective types of status properties, their values can be interpreted properly.

Message 7: sdo1 requests the current value of a particular monitoring property.

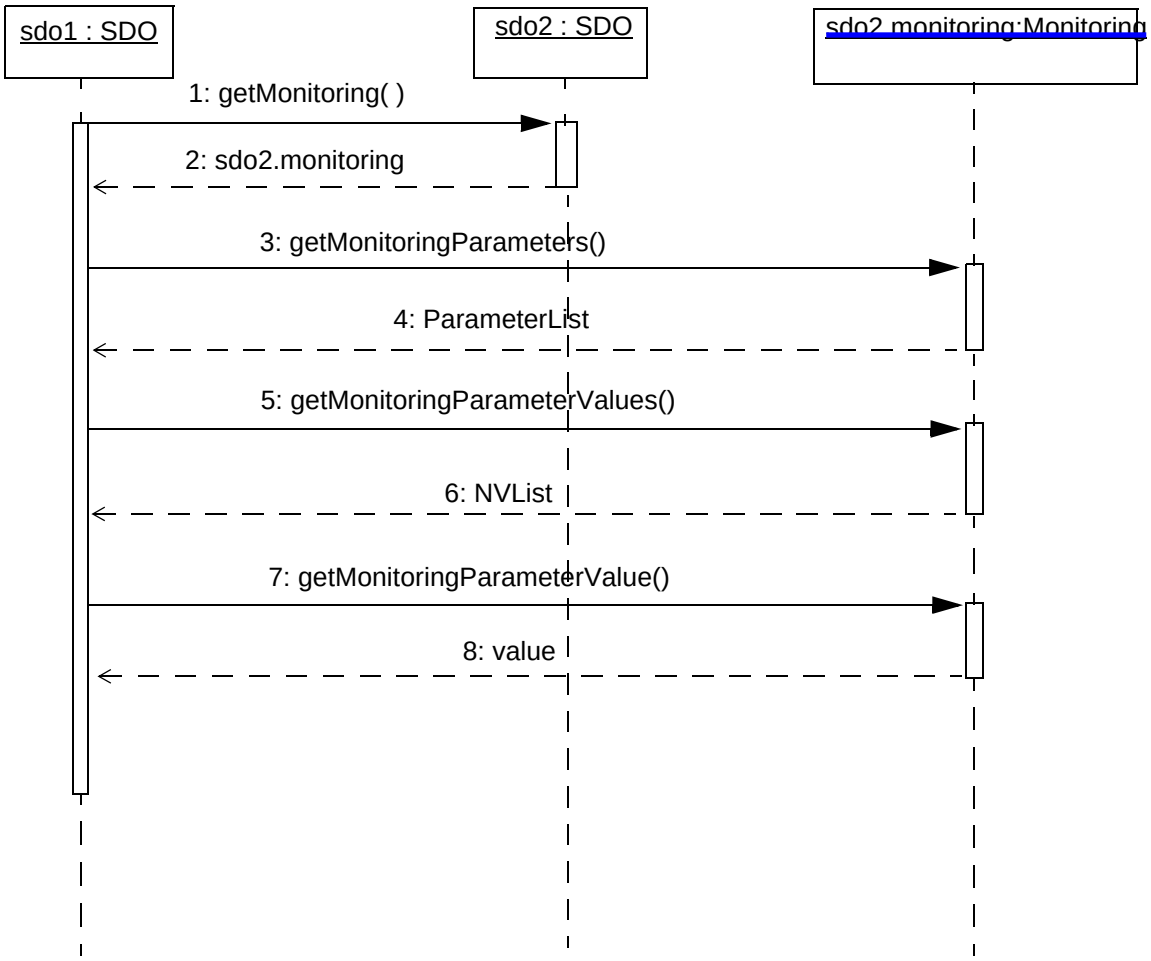


Figure 2-7 Sequence Diagram: Monitoring

Message 1: the monitoring SDO (sdo1) gets the object implementing the **Monitoring** interface of the monitored SDO (sdo2).

Message 2: sdo2 returns the object sdo2.monitoring that implements the **Monitoring** interface.

Message 3: sdo1 requests the list of all monitoring properties of sdo2.

Message 4: sdo2 sends response, containing the list of monitoring properties specifying the sdo2.

Message 5: sdo1 requests the current values of monitoring parameters of the sdo2.

Message 6: sdo2 returns the data requested in message 5 as list of names and current values of properties represented as **NVList** object. Knowing the respective types of status properties, their values can be interpreted properly.

Message 7: sdo1 requests the current value of a particular monitoring property.

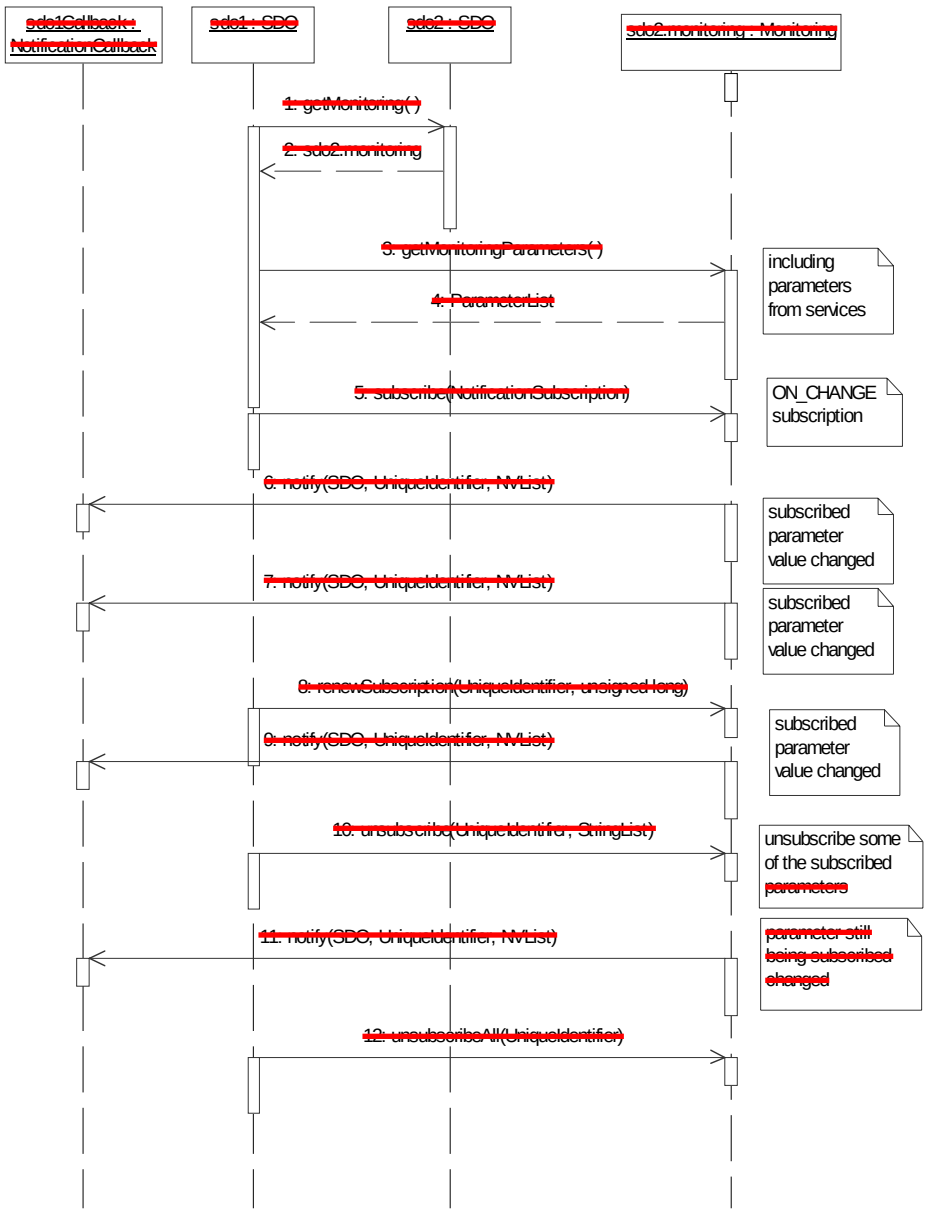


Figure 2-8 Subscription and Notification in ON_CHANGE mode

The subscription of properties is shown in the sequence diagram (Figure 2-8). The interaction depicted in Figure 2-8 occurs between two SDOs, sdo1 and sdo2. In the figure above, sdo1 intends to monitor the status of sdo2. sdo2 possesses an object that represents the **Monitoring** interface of sdo2. sdo1 implements the interface **NotificationCallback** to be able to receive notifications about status changes.

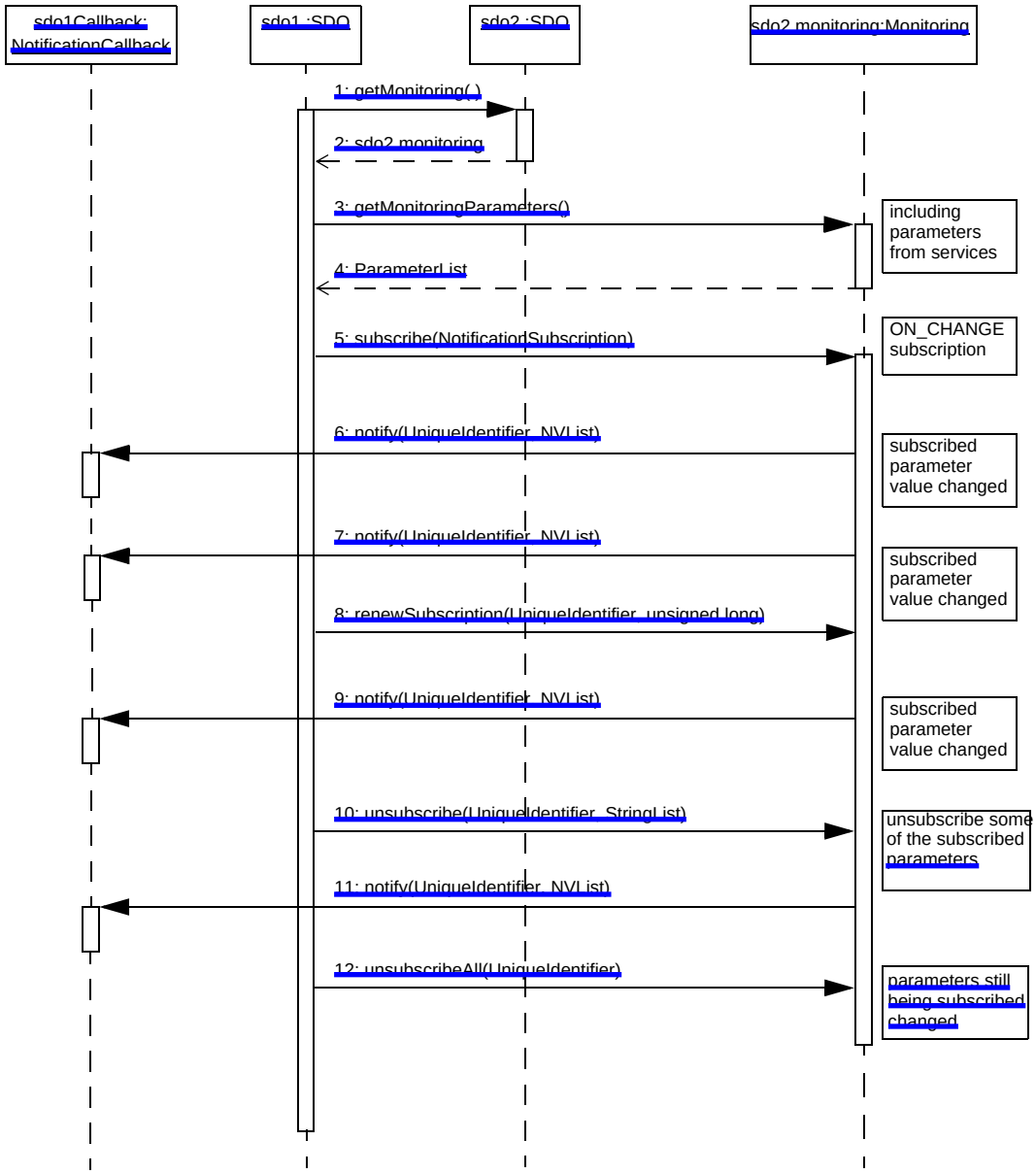


Figure 2-8 Subscription and Notification in ON_CHANGE mode

The subscription of properties is shown in the sequence diagram (Figure 2-8). The interaction depicted in Figure 2-8 occurs between two SDOs, sdo1 and sdo2. In the figure above, sdo1 intends to monitor the status of sdo2. sdo2 possesses an object that represents the **Monitoring** interface of sdo2. sdo1 implements the interface **NotificationCallback** to be able to receive notifications about status changes.

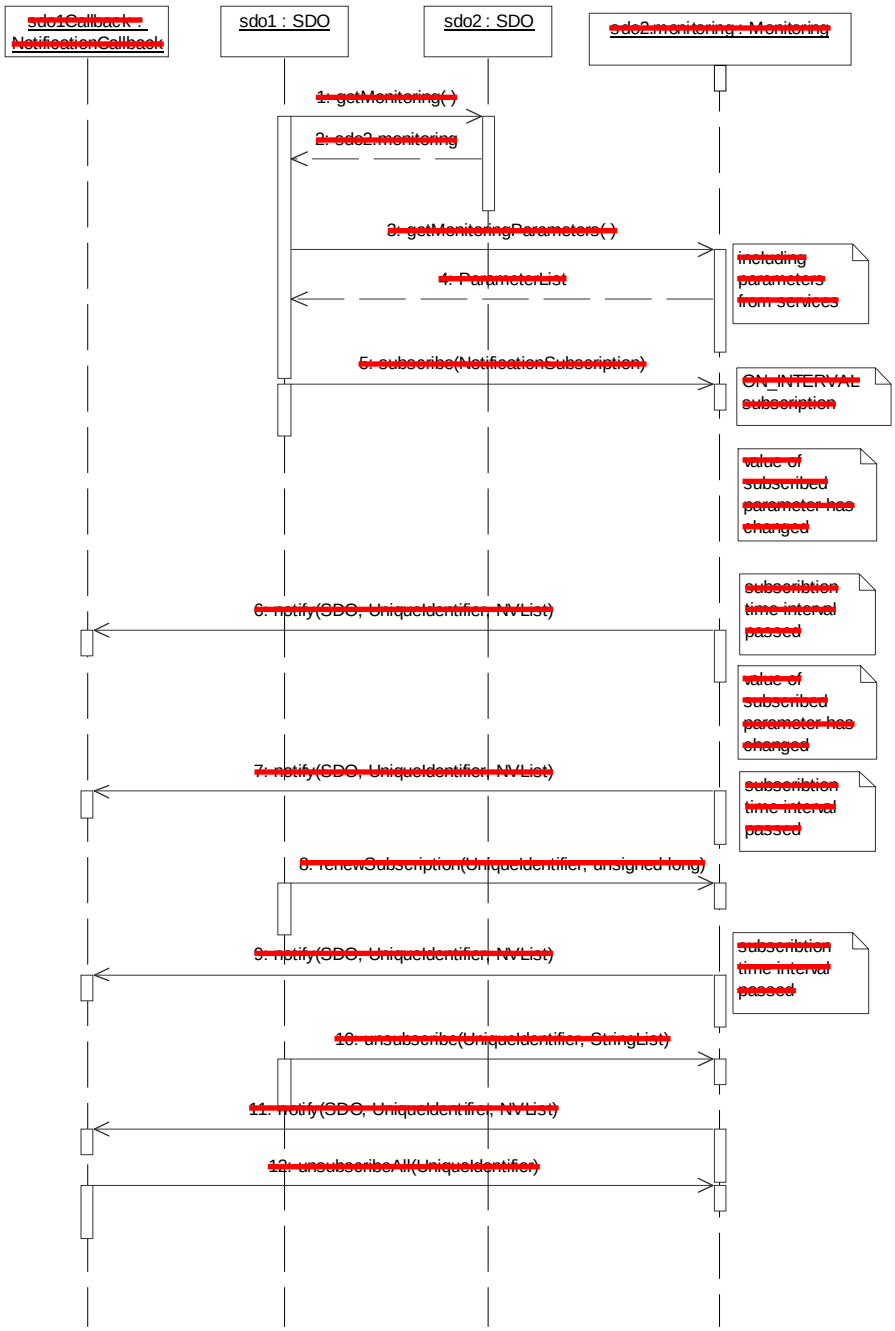


Figure 2-9 Subscription and Notification ON_INTERVAL

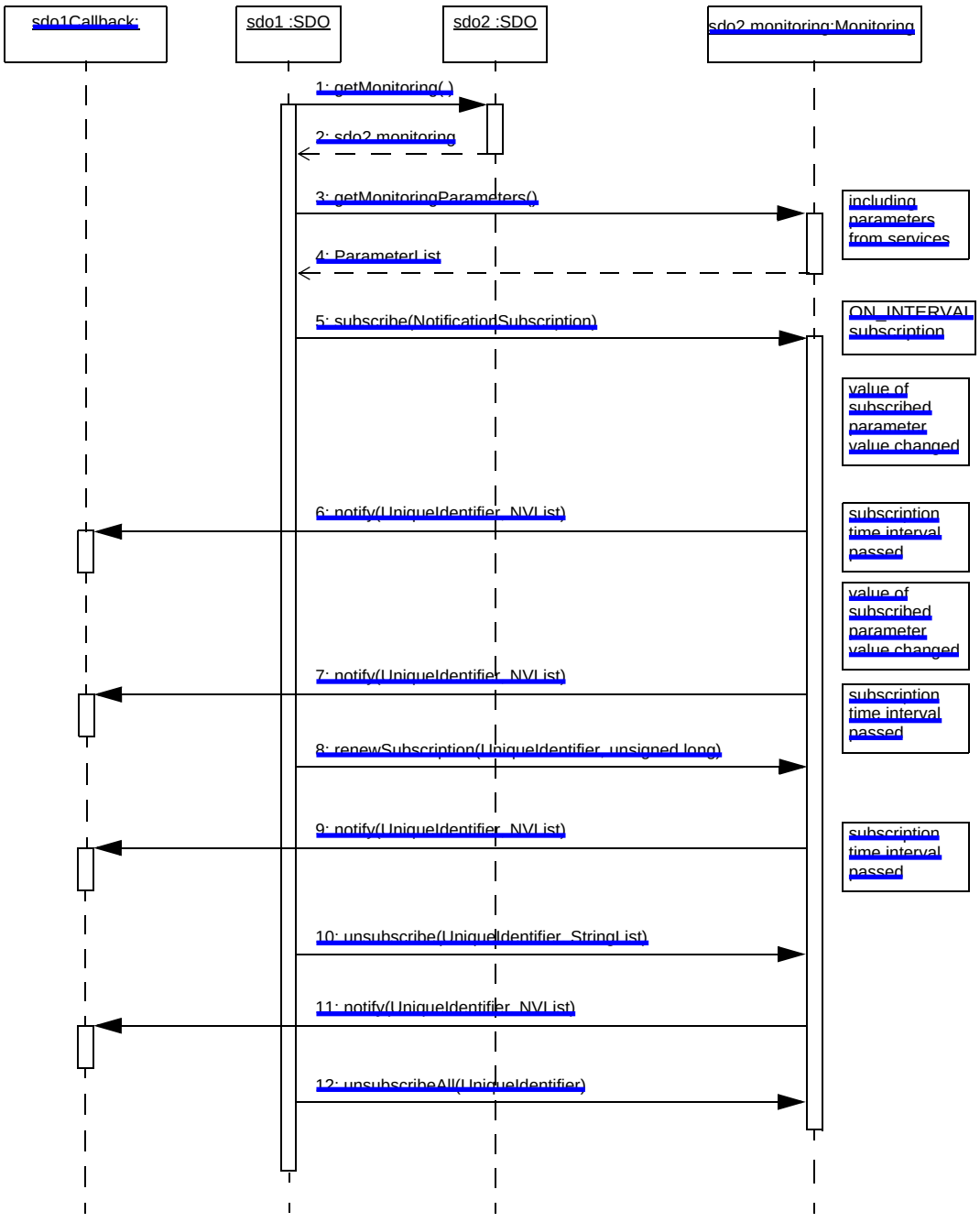


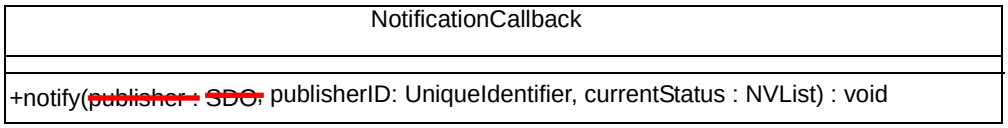
Figure 2-9 Subscription and Notification ON_INTERVAL

Figure 2-9 shows the notifications made in interval mode. In the diagram shown on the picture, sdo1 intends to monitor the status of sdo2.

In general, the same sequence of operations shown in Figure 2-8 is used to subscribe, renew, and cancel property notification. The difference is that notifications are sent not in the event when one of subscribed properties changes its value, but periodically in a specified time interval. Notifications contain the property names and their values at the moment when the notification was sent. Notifications are sent irrespectively of the fact whether the property values have changed or not since the last notification. So it can occur that between two notifications some properties have changed their values more than once or never at all. For example, in Figure 2-9 the property values change twice between notification messages 5 and 8. Furthermore, it can also occur that property values remain unchanged during several notification intervals, like in the sequence diagram in Figure 2-9 between notification messages 9 and 11.

2.3.7.5 NotificationCallback Interface

This interface provides call back mechanism for an SDO for the subscription notification. Monitoring properties of an SDO can be monitored by other SDOs by subscribing such properties. The changes in the monitored properties are subsequently notified to the subscribing SDOs. This call back interface will provide interface to notify subscribing SDOs.



2.3.7.6 Data Structures Defined for NotificationCallback Interface

Two general data structures (**UniqueIdentifier** and **NVList**) are used in the **NotificationCallback** interface. Please see Section 2.3.7.1, “Data Structures Defined for Monitoring Interface,” on page 2-36 for their detail description.

2.3.7.7 Operations provided by NotificationCallback Interface

(1) +notify(publisherID : UniqueIdentifier, currentStatus : NVList) : void

This operation notifies the subscriber that either the value of the property has changed or the notification interval has elapsed.

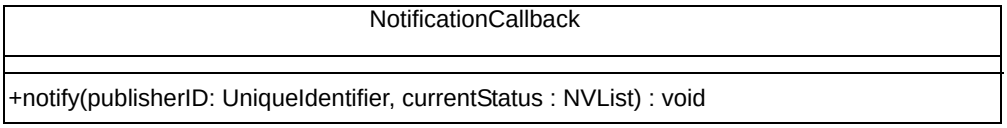
Parameter	Type	Description
publisherID	**UniqueIdentifier**	Unique identifier of the SDO that is sending the notification.

Figure 2-9 shows the notifications made in interval mode. In the diagram shown on the picture, sdo1 intends to monitor the status of sdo2.

In general, the same sequence of operations shown in Figure 2-8 is used to subscribe, renew, and cancel property notification. The difference is that notifications are sent not in the event when one of subscribed properties changes its value, but periodically in a specified time interval. Notifications contain the property names and their values at the moment when the notification was sent. Notifications are sent irrespectively of the fact whether the property values have changed or not since the last notification. So it can occur that between two notifications some properties have changed their values more than once or never at all. For example, in Figure 2-9 the property values change twice between notification messages 5 and 8. Furthermore, it can also occur that property values remain unchanged during several notification intervals, like in the sequence diagram in Figure 2-9 between notification messages 9 and 11.

2.3.7.5 NotificationCallback Interface

This interface provides call back mechanism for an SDO for the subscription notification. Monitoring properties of an SDO can be monitored by other SDOs by subscribing such properties. The changes in the monitored properties are subsequently notified to the subscribing SDOs. This call back interface will provide interface to notify subscribing SDOs.



2.3.7.6 Data Structures Defined for NotificationCallback Interface

Two general data structures (**UniqueIdentifier** and **NVList**) are used in the **NotificationCallback** interface. Please see Section 2.3.7.1, “Data Structures Defined for Monitoring Interface,” on page 2-36 for their detail description.

2.3.7.7 Operations provided by NotificationCallback Interface

(1) +notify(publisherID : UniqueIdentifier, currentStatus : NVList) : void

This operation notifies the subscriber that either the value of the property has changed or the notification interval has elapsed.

Parameter	Type	Description
publisherID	**UniqueIdentifier**	Unique identifier of the SDO that is sending the notification.

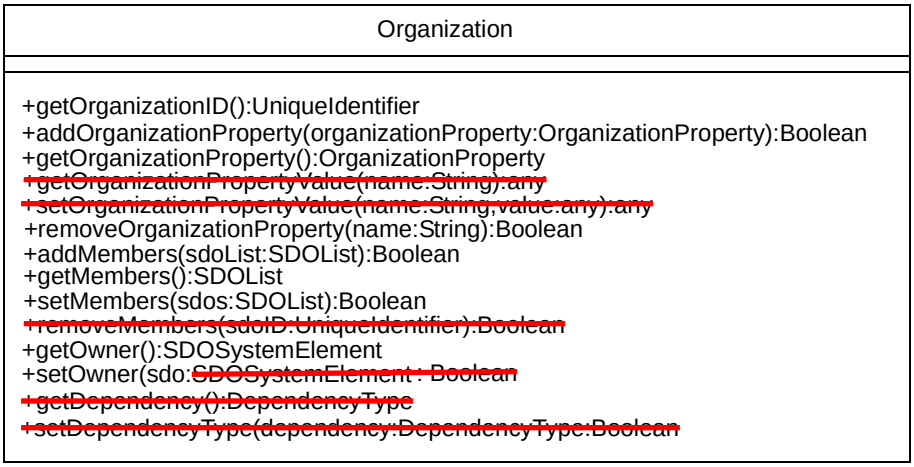
currentStatus	**NVList**	A list containing the properties and their current values. Please note that this list may not contain all the properties that an SDO has been subscribed to, probably because not all property has changed or because the notification interval of some properties has not elapsed yet.

2.3.7.8 *Usage: NotificationCallback Interface*

The examples of usage of operations of **NotificationCallback** interface are shown in Figure 2-8 and Figure 2-9. The operations are used to convey the changes in values of monitoring properties to notification subscriber.

~~2.3.8 Organization~~ *Interface*

The **Organization** interface is used to manage the **Organization** attribute.



(1) + *getOrganizationID() : UniqueIdentifier*

This operation returns the 'id' of the **Organization**.

Parameter	Type	Description
<return>	**UniqueIdentifier**	The identifier of the Organization defined in the resource data model.

Exception

This operation throws SDOException with one of the following exception types.

- SDONotExists - if the target SDO does not exist.

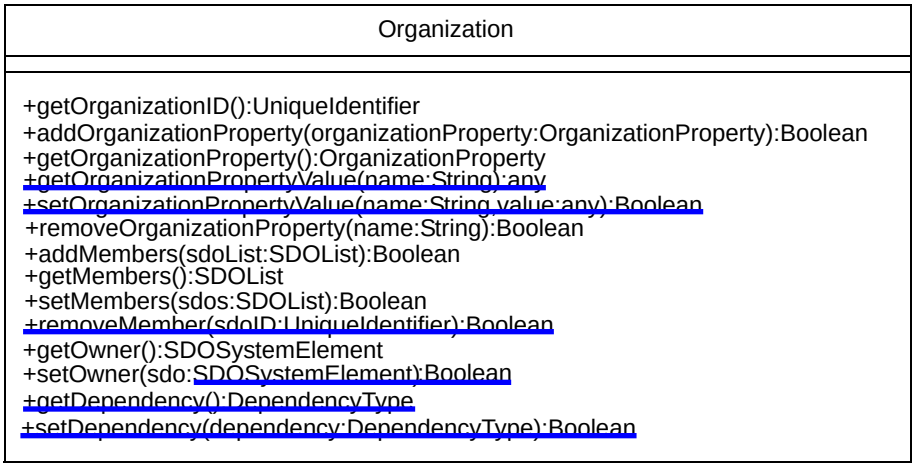
currentStatus	**NVList**	A list containing the properties and their current values. Please note that this list may not contain all the properties that an SDO has been subscribed to, probably because not all property has changed or because the notification interval of some properties has not elapsed yet.

2.3.7.8 *Usage: NotificationCallback Interface*

The examples of usage of operations of **NotificationCallback** interface are shown in Figure 2-8 and Figure 2-9. The operations are used to convey the changes in values of monitoring properties to notification subscriber.

2.3.8 *Organization Interface*

The **Organization** interface is used to manage the **Organization** attribute.



(1) *+ getOrganizationID() : UniqueIdentifier*

This operation returns the 'id' of the **Organization**.

Parameter	Type	Description
<return>	**UniqueIdentifier**	The identifier of the Organization defined in the resource data model.

Exception

This operation throws SDOException with one of the following exception types.

- SDONotExists - if the target SDO does not exist.

(4) + getOrganizationPropertyValue(name : String): any

This operation returns a value in the **OrganizationProperty**. The value to be returned is specified by argument '**name**.'

Parameter	Type	Description
Name	**String**	The name of the value to be returned.
<return>	~~**Any**~~	The value of property which is specified by argument 'name.'

Exception

This operation throws SDOException with one of the following exception types.

- InvalidParameter - if the property which is specified by argument "name" does not exist.
- NotAvailable - if there is no response from the target object.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.
- SDONotExists - if the target SDO does not exist.

(5) + setOrganizationPropertyValue (name : String, value : any): Boolean

This operation adds or updates a pair of name and value as a property of **Organization** to/in **NVList** of the **OrganizationProperty**. The name and the value to be added/updated are specified by argument '**name**' and '**value**.'

Parameter	Type	Description
Name	**String**	The name of the property to be added/updated.
Value	~~**Any**~~	The value of the property to be added/updated.

Exception

This operation throws SDOException with one of the following exception types.

- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.
- InvalidParameter - if the property that is specified by argument "name" does not exist.

(4) + getOrganizationPropertyValue(name : String): any

This operation returns a value in the **OrganizationProperty**. The value to be returned is specified by argument '**name**.'

Parameter	Type	Description
Name	**String**	The name of the value to be returned.
<return>	**any**	The value of property which is specified by argument 'name.'

Exception

This operation throws SDOException with one of the following exception types.

- InvalidParameter - if the property which is specified by argument "name" does not exist.
- NotAvailable - if there is no response from the target object.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.
- SDONotExists - if the target SDO does not exist.

(5) + setOrganizationPropertyValue (name : String, value : any): Boolean

This operation adds or updates a pair of name and value as a property of **Organization** to/in **NVList** of the **OrganizationProperty**. The name and the value to be added/updated are specified by argument '**name**' and '**value**.'

Parameter	Type	Description
Name	**String**	The name of the property to be added/updated.
Value	**any**	The value of the property to be added/updated.
<return>	**Boolean**	If the operation was successfully completed

Exception

This operation throws SDOException with one of the following exception types.

- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.
- InvalidParameter - if the property that is specified by argument "name" does not exist.

(6) + removeOrganizationProperty (name : String): Boolean

This operation removes a property of **Organization** from **NVList** of the **OrganizationProperty**. The property to be removed is specified by argument '**name**.'

Parameter	Value	Description
Name	**String**	The name of the property to be removed.

Exception

This operation throws SDOException with one of the following exception types.

- InvalidParameter - if the property that is specified by argument "name" does not exist.
- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(7) + addMembers(sdoList : SDOList) :Boolean

This operation adds a member that is an SDO to the organization. The member to be added is specified by argument '~~**sdo**.~~'

~~Parameter~~	~~Value~~	~~Description~~
~~**sdo**~~	~~**SDO**~~	The member to be added to the organization.

Exception

This operation throws SDOException with one of the following exception types.

- ~~InvalidParameter - if argument "sdo" is null.~~
- NotAvailable - if there is no response from the target object.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.
- SDONotExists - if the target SDO does not exist.

(8) + getMembers ( ) : SDOList

This operation returns a list of **SDO**s that are members of an Organization. An empty list is returned if the **Organization** does not have any members.

(6) + removeOrganizationProperty (name : String): Boolean

This operation removes a property of **Organization** from **NVList** of the **OrganizationProperty**. The property to be removed is specified by argument '**name**.'

Parameter	Value	Description
Name	**String**	The name of the property to be removed.
<return>	**Boolean**	If the operation was successfully completed.

Exception

This operation throws SDOException with one of the following exception types.

- InvalidParameter - if the property that is specified by argument "name" does not exist.
- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(7) + addMembers(sdoList : SDOList) :Boolean

This operation adds a member that is an SDO to the organization. The member to be added is specified by argument '**sdoList**.'

Parameter	Value	Description
sdoList	**SDOList**	The member to be added to the organization.
<return>	**Boolean**	If the operation was successfully completed.

Exception

This operation throws SDOException with one of the following exception types.

- InvalidParameter - if argument "sdoList" is null.
- NotAvailable - if there is no response from the target object.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.
- SDONotExists - if the target SDO does not exist.

(8) + getMembers ( ) : SDOList

This operation returns a list of **SDO**s that are members of an Organization. An empty list is returned if the **Organization** does not have any members.

Parameter	Type	Description
<return>	**SDOList**	Member SDOs that are contained in the **Organization** object.

Exception

This operation throws SDOException with one of the following exception types.

- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(9) + setMembers (sdos : SDOList) : Boolean

This operation assigns a list of **SDO**s to an **Organization** as its members. If the **Organization** has already maintained a member **SDO**(s) when it is called, the operation replaces the member(s) with specified list of **SDO**s.

Parameter	Type	Description
sdos	**SDOList**	Member **SDO**s to be assigned.
<return>	**Boolean**	If the operation was successfully completed.

Exception

This operation throws SDOException with one of the following exception types.

- InvalidParameter - if argument “SDOList” is null, or if the object that is specified by the argument “sdos” does not exist.
- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(10) + removeMember(sdoID : UniqueIdentifier) :Boolean

This operation removes a member from the organization. The member to be removed is specified by argument ‘**id.**’

Parameter	Value	Description
sdoID	**UniqueIdentifier**	Id of the SDO to be removed from the organization.

Parameter	Type	Description
<return>	**SDOList**	Member SDOs that are contained in the **Organization** object.

Exception

This operation throws SDOException with one of the following exception types.

- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(9) + setMembers (sdos : SDOList) : Boolean

This operation assigns a list of **SDO**s to an **Organization** as its members. If the **Organization** has already maintained a member **SDO**(s) when it is called, the operation replaces the member(s) with specified list of **SDO**s.

Parameter	Type	Description
sdos	**SDOList**	Member **SDO**s to be assigned.
<return>	**Boolean**	If the operation was successfully completed.

Exception

This operation throws SDOException with one of the following exception types.

- InvalidParameter - if argument “SDOList” is null, or if the object that is specified by the argument “sdos” does not exist.
- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(10) + removeMember(sdoID : UniqueIdentifier) :Boolean

This operation removes a member from the organization. The member to be removed is specified by argument ‘**sdoID**.’

Parameter	Value	Description
sdoID	**UniqueIdentifier**	Id of the SDO to be removed from the organization.

Exception

This operation throws SDOException with one of the following exception types.

- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.
- InvalidParameter - if argument “id” is null or does not exist.
- NotAvailable - if there is no response from the target object.

(11) + getOwner ( ) : SDOSystemElement

This operation returns the SDOSystemElement that is owner of the Organization.

Parameter	Type	Description
<return>	SDOSystemElement	Reference of owner object.

Exception

This operation throws SDOException with one of the following exception types.

- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(12) + setOwner (sdo : SDOSystemElement) : Boolean

This operation sets an **SDOSystemElement** to the owner of the Organization. The **SDOSystemElement** to be set is specified by argument “**sdo**.”

Parameter	Type	Description
sdo	SDOSystemElement	Reference of owner object.
<return>	Boolean	If the operation was successfully completed.

Exception

This operation throws SDOException with one of the following exception types.

- InvalidParameter - if argument “**sdo**” is null, or if the object that is specified by “**sdo**” in argument “**sdo**” does not exist.
- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.

<return>	**Boolean**	If the operation was successfully completed.

Exception

This operation throws SDOException with one of the following exception types.

- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.
- InvalidParameter - if argument “id” is null or does not exist.
- NotAvailable - if there is no response from the target object.

(11) + getOwner ( ) : SDOSystemElement

This operation returns the SDOSystemElement that is owner of the Organization.

Parameter	Type	Description
<return>	**SDOSystemElement**	Reference of owner object.

Exception

This operation throws SDOException with one of the following exception types.

- NotAvailable - if there is no response from the target object.
- SDONotExists - if the target SDO does not exist.
- InternalError - if the target SDO cannot execute the operation completely due to some internal error.

(12) + setOwner (sdo : SDOSystemElement) : Boolean

This operation sets an **SDOSystemElement** to the owner of the Organization. The **SDOSystemElement** to be set is specified by argument “**sdo**.”

Parameter	Type	Description
sdo	**SDOSystemElement**	Reference of owner object.
<return>	**Boolean**	If the operation was successfully completed.

Exception

This operation throws SDOException with one of the following exception types.

- InvalidParameter - if argument “**sdo**” is null, or if the object that is specified by “**sdo**” in argument “**sdo**” does not exist.
- NotAvailable - if there is no response from the target object.

2.3.8.1 *Usage: Organization*

Figure 2-10 shows a sequence diagram to explain how to obtain a set of properties from an Organization. In this example, an SDO (sdo1) calls **getOrganizations()** on another SDO (sdo2) to obtain the Organization objects that sdo2 is associated with, chooses one of the obtained **Organization** objects (organization1), and then calls **getOrganizationProperty()** on organization1 in order to see its properties.

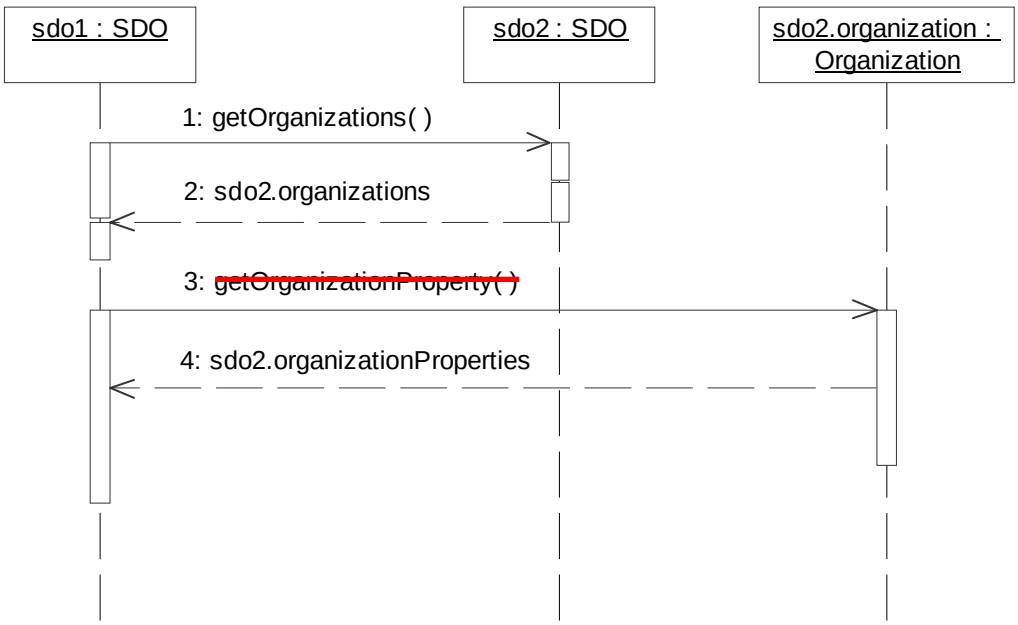


Figure 2-10 An example to obtain an OrganizationProperty

Message 1: An SDO (sdo1) asks another SDO (sdo2) to return the **Organization** objects that sdo2 is associated with.

Message 2: sdo2 returns sdo2.organizations, which is a list of **Organization** objects.

Message 3: sdo1 chooses one of the obtained **Organization** object (organization1) and requests its properties (i.e., organization1.property).

Message 4: organization1 returns organization1.property.

2.3.8.1 Usage: Organization

Figure 2-10 shows a sequence diagram to explain how to obtain a set of properties from an Organization. In this example, an SDO (sdo1) calls **getOrganizations()** on another SDO (sdo2) to obtain the Organization objects that sdo2 is associated with, chooses one of the obtained **Organization** objects (organization1), and then calls **getOrganizationProperty()** on organization1 in order to see its properties.

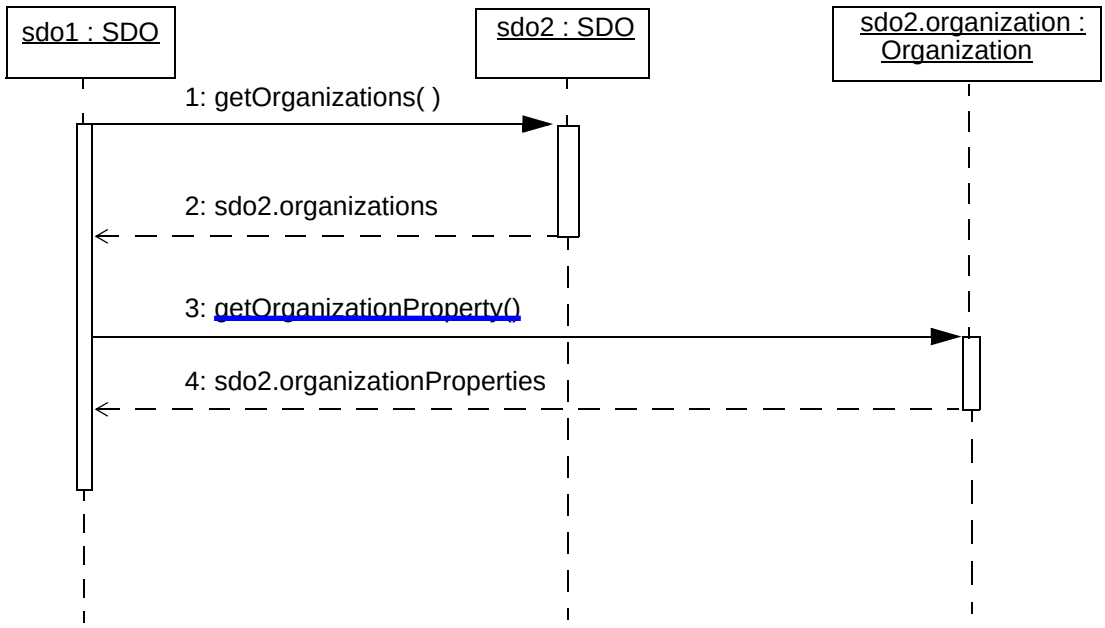


Figure 2-10 An example to obtain an OrganizationProperty

Message 1: An SDO (sdo1) asks another SDO (sdo2) to return the **Organization** objects that sdo2 is associated with.

Message 2: sdo2 returns sdo2.organizations, which is a list of **Organization** objects.

Message 3: sdo1 chooses one of the obtained **Organization** object (organization1) and requests its properties (i.e., organization1.property).

Message 4: organization1 returns organization1.property.

```
      case RANGE: RangeType   allowed_range;
    };
    struct Parameter {
      string        name;
      CORBA::TCKind  type;
      AllowedValues allowed_values;
    };
    typedef sequence<Parameter> ParameterList;
    struct OrganizationProperty {
      NVList properties;
    };
enum DependencyType {
      OWN,
      OWNED,
      NO_DEPENDENCY
};
    struct DeviceProfile {
      string device_type;
      string manufacturer;
      string model;
      string version;
      NVList properties;
    };
    struct ServiceProfile {
      string id;
      string interface_type;
      NVList properties;
      SDOService service;
    };
typedef sequence<ServiceProfile> ServiceProfileList;

struct ConfigurationSet {
      string id;
      string description;
      NVList configuration_data;
};
typedef sequence<ConfigurationSet> ConfigurationSetList;
```

3.3 Exceptions

The methods of SDO interfaces can raise SDOException. The kind of exception is defined in the attribute *type* (see Section 2.3.2.1, “SDOException,” on page 2-17). This exception is mapped to several CORBA exceptions. All defined exceptions have structure specified by a macro **exception_body**. Five exceptions are defined in this specification: NotAvailable, InterfaceNotImplemented, InvalidParameter, NotFound, and InternalError.

```
#define exception_body { string description; }
…
exception NotAvailable                    exception_body;
```

```
    case RANGE: RangeType   allowed_range;
  };
  struct Parameter {
    string        name;
    CORBA::TCKind  type;
    AllowedValues allowed_values;
  };
  typedef sequence<Parameter> ParameterList;
  struct OrganizationProperty {
    NVList properties;
  };
enum DependencyType {
    OWN,
    OWNED,
    NO_DEPENDENCY
};
  struct DeviceProfile {
    string device_type;
    string manufacturer;
    string model;
    string version;
    NVList properties;
  };
  struct ServiceProfile {
    string id;
    string interface_type;
    NVList properties;
    SDOService service;
  };
typedef sequence<ServiceProfile> ServiceProfileList;

struct ConfigurationSet {
    string id;
    string description;
    NVList configuration_data;
};
typedef sequence<ConfigurationSet> ConfigurationSetList;
```

3.3 Exceptions

The methods of SDO interfaces can raise SDOException. The kind of exception is defined in the attribute *type* (see Section 2.3.2.1, “SDOException,” on page 2-17). This exception is mapped to several CORBA exceptions. All defined exceptions have structure specified by a macro **exception_body**. Five exceptions are defined in this specification: NotAvailable, InterfaceNotImplemented, InvalidParameter, SDONotExists, and InternalError.

```
#define exception_body { string description; }
...
exception NotAvailable                  exception_body;
```

3.4.2 SDO Interface

The **SDO** interface in the PIM is mapped directly to a CORBA interface. It inherits the **SDOSystemElement** interface.

```
interface SDO : SDOSystemElement {
    UniqueIdentifier get_sdo_id()
        raises (NotAvailable, InternalError);
    string get_sdo_type()
        raises (NotAvailable, InternalError);
    DeviceProfile get_device_profile ()
        raises (NotAvailable, InternalError);
    ServiceProfileList get_service_profiles ()
        raises (NotAvailable, InternalError);
    ServiceProfile get_service_profile (in UniqueIdentifier id)
        raises (InvalidParameter, NotAvailable, InternalError);
    SDOService get_sdo_service (in UniqueIdentifier id)
        raises (InvalidParameter, NotAvailable, InternalError);
    Configuration get_configuration ()
        raises (InterfaceNotImplemented, NotAvailable, InternalError);
    Monitoring get_monitoring ()
        raises (InterfaceNotImplemented, NotAvailable, InternalError);
    OrganizationList get_organizations ()
        raises (NotAvailable, InternalError);
    NVList get_status_list ()
        raises (NotAvailable, InternalError);
            any get_status ()
        raises (InvalidParameter, NotAvailable, InternalError);
};
```

3.4.3 Configuration Interface

The **Configuration** interface in the PIM is mapped directly to a CORBA interface:

```
interface Configuration {
    boolean set_device_profile (in DeviceProfile dProfile)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean set_service_profile (in ServiceProfile sProfile)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean add_organization (in Organization organization)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean remove_service_profile (in UniqueIdentifier id)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean remove_organization (in UniqueIdentifier organization_id)
        raises (InvalidParameter, NotAvailable, InternalError);
    ParameterList get_configuration_parameters ()
        raises (NotAvailable, InternalError);
    NVList get_configuration_parameter_values ()
        raises (NotAvailable, InternalError);
    any get_configuration_parameter_value (in string name)
```

3.4.2 SDO Interface

The **SDO** interface in the PIM is mapped directly to a CORBA interface. It inherits the **SDOSystemElement** interface.

```
interface SDO : SDOSystemElement {
    UniqueIdentifier get_sdo_id()
        raises (NotAvailable, InternalError);
    string get_sdo_type()
        raises (NotAvailable, InternalError);
    DeviceProfile get_device_profile ()
        raises (NotAvailable, InternalError);
    ServiceProfileList get_service_profiles ()
        raises (NotAvailable, InternalError);
    ServiceProfile get_service_profile (in UniqueIdentifier id)
        raises (InvalidParameter, NotAvailable, InternalError);
    SDOService get_sdo_service (in UniqueIdentifier id)
        raises (InvalidParameter, NotAvailable, InternalError);
    Configuration get_configuration ()
        raises (InterfaceNotImplemented, NotAvailable, InternalError);
    Monitoring get_monitoring ()
        raises (InterfaceNotImplemented, NotAvailable, InternalError);
    OrganizationList get_organizations ()
        raises (NotAvailable, InternalError);
    NVList get_status_list ()
        raises (NotAvailable, InternalError);
            any get_status (in string name)
        raises (InvalidParameter, NotAvailable, InternalError);
};
```

3.4.3 Configuration Interface

The **Configuration** interface in the PIM is mapped directly to a CORBA interface:

```
interface Configuration {
    boolean set_device_profile (in DeviceProfile dProfile)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean add_service_profile (in ServiceProfile sProfile)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean add_organization (in Organization organization_object)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean remove_service_profile (in UniqueIdentifier id)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean remove_organization (in UniqueIdentifier organization_id)
        raises (InvalidParameter, NotAvailable, InternalError);
    ParameterList get_configuration_parameters ()
        raises (NotAvailable, InternalError);
    NVList get_configuration_parameter_values ()
        raises (NotAvailable, InternalError);
    any get_configuration_parameter_value (in string name)
```

```
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean set_configuration_parameter (
            in string name,
            in any value)
        raises (InvalidParameter, NotAvailable, InternalError);
    ConfigurationSetList get_configuration_sets ()
        raises (NotAvailable, InternalError);
    ConfigurationSet get_configuration_set (in UniqueIdentifier config_id)
        raises (NotAvailable, InternalError);
    boolean set_configuration_set_values (
            in UniqueIdentifier config_id,
            in ConfigurationSet configuration_set)
        raises (InvalidParameter, NotAvailable, InternalError);
    ConfigurationSet get_active_configuration_set ()
        raises (NotAvailable, InternalError);
    boolean add_configuration_set (in ConfigurationSet configuration_set)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean remove_configuration_set (in UniqueIdentifier config_id)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean activate_configuration_set (in UniqueIdentifier config_id)
        raises (InvalidParameter, NotAvailable, InternalError);
};
```

3.4.4 SDOService

In the PSM, SDO services are represented by CORBA objects. The class **SDOService** is mapped to an empty IDL interface. When implementing real services, their interfaces should be derived from the **SDOService** interface.

3.4.5 Monitoring Interface

The interface **Monitoring** in the PIM is mapped to a CORBA interface. The operations that enable to obtain the list of monitoring parameters supported by the SDO and their current values are mapped straight forward to operations of **Monitoring** Interface. The subscription and notification mechanisms described in Section 2.3.7, “Monitoring Interface,” on page 2-35 are implemented using the OMG Notification Service [4].

```
interface Monitoring : CosNotifyComm::StructuredPushConsumer,
            CosNotifyComm::StructuredPushSupplier {
    any get_monitoring_parameter_value (
            in string name
        ) raises (InvalidParameter, NotAvailable, InternalError);
    ParameterList get_monitoring_parameters ()
        raises (NotAvailable, InternalError);
    NVList get_monitoring_parameter_values ()
        raises (NotAvailable, InternalError);
};
```

```
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean set_configuration_parameter (
            in string name,
            in any value)
        raises (InvalidParameter, NotAvailable, InternalError);
    ConfigurationSetList get_configuration_sets ()
        raises (NotAvailable, InternalError);
    ConfigurationSet get_configuration_set (in UniqueIdentifier config_id)
        raises (NotAvailable, InternalError, InvalidParameter);
    boolean set_configuration_set_values (
            in ConfigurationSet configuration_set)
        raises (InvalidParameter, NotAvailable, InternalError);
    ConfigurationSet get_active_configuration_set ()
        raises (NotAvailable, InternalError);
    boolean add_configuration_set (in ConfigurationSet configuration_set)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean remove_configuration_set (in UniqueIdentifier config_id)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean activate_configuration_set (in UniqueIdentifier config_id)
        raises (InvalidParameter, NotAvailable, InternalError);
};
```

3.4.4 SDOService

In the PSM, SDO services are represented by CORBA objects. The class **SDOService** is mapped to an empty IDL interface. When implementing real services, their interfaces should be derived from the **SDOService** interface.

3.4.5 *Monitoring Interface*

The interface **Monitoring** in the PIM is mapped to a CORBA interface. The operations that enable to obtain the list of monitoring parameters supported by the SDO and their current values are mapped straight forward to operations of **Monitoring** Interface. The subscription and notification mechanisms described in Section 2.3.7, “Monitoring Interface,” on page 2-35 are implemented using the OMG Notification Service [4].

```
interface Monitoring : CosNotifyComm::StructuredPushConsumer,
            CosNotifyComm::StructuredPushSupplier {
    any get_monitoring_parameter_value (
            in string name
        ) raises (InvalidParameter, NotAvailable, InternalError);
    ParameterList get_monitoring_parameters ()
        raises (NotAvailable, InternalError);
    NVList get_monitoring_parameter_values ()
        raises (NotAvailable, InternalError);
};
```

To use the mechanisms defined in Notification Service, the Monitoring interface inherits the interfaces **StructuredPushSupplier** and **StructuredPushConsumer**, defined in **CosNotifyComm** module.

To use the mechanisms defined in Notification Service, the Monitoring interface inherits the interfaces **StructuredPushSupplier** and **StructuredPushConsumer**, defined in **CosNotifyComm** module.

The interface **StructuredPushSupplier** enables SDOs to publish event notifications to the Notification Service event channel (referred henceforth as the *notification channel*). This interface supports the behavior of objects that send Structured Events into the notification channel using push-style communication. (Models of event propagation are described in [5].) The operation **subscription_change** enables a notification consumer to inform an instance supporting this interface whenever there is a change to the types of events it is interested in receiving. The operation **disconnect_structured_push_supplier** is invoked to terminate a connection between the target **StructuredPushSupplier**, and its associated consumer. The operations of the interface **StructuredPushSupplier** cover the group of subscription operations defined for Monitoring interface in Section 2.3.7.2, “Operations provided by Monitoring Interface,” on page 2-39.

The interface **StructuredPushConsumer** enables the notification channel to send SDOs status notifications supplied by event suppliers as Structured Events by the push model, using the operation **push_structured_event**. The operation **offer_change** enables a notification supplier to inform an instance supporting this interface whenever there is a change to the types of events it intends to produce. The interface **StructuredPushConsumer** provides functionality that covers the functionality of **NotificationCallback** interface defined in Section 2.3.7.7, “Operations provided by NotificationCallback Interface,” on page 2-49.

3.4.6 Organization Interface

The **Organization** interface is mapped in the PSM to a CORBA interface. The class attributes are mapped to interface operations. For example, the attribute members is mapped to the operation pair **getMembers()** and **setMembers()**. It should also be noticed that both these operations work with lists of references of SDOs that belong to the organization. The operations **getOwner()** and **setOwner()** manipulate the reference of an object that owns the organization.

```
interface Organization {
UniqueIdentifier get_organization_id ()
        raises (InvalidParameter, NotAvailable, InternalError);
    OrganizationProperty get_organization_property ()
        raises (NotAvailable, InternalError);
    any get_organization_property_value (in string name)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean set_organization_property (
            in OrganizationProperty organization_property
        ) raises (InvalidParameter, NotAvailable, InternalError);
    boolean set_organization_property_value (
            in string name,
            in any value
        ) raises (InvalidParameter, NotAvailable, InternalError);
    boolean remove_organization_property ( in string name )
```

The interface **StructuredPushSupplier** enables SDOs to publish event notifications to the Notification Service event channel (referred henceforth as the *notification channel*). This interface supports the behavior of objects that send Structured Events into the notification channel using push-style communication. (Models of event propagation are described in [5].) The operation **subscription_change** enables a notification consumer to inform an instance supporting this interface whenever there is a change to the types of events it is interested in receiving. The operation **disconnect_structured_push_supplier** is invoked to terminate a connection between the target **StructuredPushSupplier**, and its associated consumer. The operations of the interface **StructuredPushSupplier** cover the group of subscription operations defined for Monitoring interface in Section 2.3.7.2, “Operations provided by Monitoring Interface,” on page 2-39.

The interface **StructuredPushConsumer** enables the notification channel to send SDOs status notifications supplied by event suppliers as Structured Events by the push model, using the operation **push_structured_event**. The operation **offer_change** enables a notification supplier to inform an instance supporting this interface whenever there is a change to the types of events it intends to produce. The interface **StructuredPushConsumer** provides functionality that covers the functionality of **NotificationCallback** interface defined in Section 2.3.7.7, “Operations provided by NotificationCallback Interface,” on page 2-49.

3.4.6 *Organization Interface*

The **Organization** interface is mapped in the PSM to a CORBA interface. The class attributes are mapped to interface operations. For example, the attribute members is mapped to the operation pair **getMembers()** and **setMembers()**. It should also be noticed that both these operations work with lists of references of SDOs that belong to the organization. The operations **getOwner()** and **setOwner()** manipulate the reference of an object that owns the organization.

```
interface Organization {
UniqueIdentifier get_organization_id ()
        raises (InvalidParameter, NotAvailable, InternalError);
    OrganizationProperty get_organization_property ()
        raises (NotAvailable, InternalError);
    any get_organization_property_value (in string name)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean add_organization_property (
            in OrganizationProperty organization_property
        ) raises (InvalidParameter, NotAvailable, InternalError);
    boolean set_organization_property_value (
            in string name,
            in any value
        ) raises (InvalidParameter, NotAvailable, InternalError);
    boolean remove_organization_property ( in string name )
        raises (InvalidParameter, NotAvailable, InternalError);
    SDOSystemElement get_owner ()
        raises (NotAvailable, InternalError);
    boolean set_owner (in SDOSystemElement sdo)
```

```
        raises (InvalidParameter, NotAvailable, InternalError);
    SDOSystemElement get_owner ()
        raises (NotAvailable, InternalError);
    boolean set_owner (in SDOSystemElement sdo)
        raises (InvalidParameter, NotAvailable, InternalError);
    SDOList get_members ()
        raises (NotAvailable, InternalError);
    boolean set_members (in SDOList sdos)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean add_members ( in SDOList sdo_list)
        raises (InvalidParameter, NotAvailable, InternalError);
    boolean remove_member (in UniqueIdentifier id)
        raises (InvalidParameter, NotAvailable, InternalError);
    DependencyType get_dependency()
        raises (NotAvailable, InternalError);
    boolean set_dependency (in DependencyType dependency)
        raises (NotAvailable, InternalError);
};
```

```
            raises (InvalidParameter, NotAvailable, InternalError);
        SDOList get_members ()
            raises (NotAvailable, InternalError);
        boolean set_members (in SDOList sdos)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean add_members ( in SDOList sdo_list)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean remove_member (in UniqueIdentifier id)
            raises (InvalidParameter, NotAvailable, InternalError);
        DependencyType get_dependency()
            raises (NotAvailable, InternalError);
        boolean set_dependency (in DependencyType dependency)
            raises (NotAvailable, InternalError);
    };
```

```
        string manufacturer;
        string model;
        string version;
        NVList properties;
    };
    struct ServiceProfile {
        string id;
        string interface_type;
        NVList properties;
        SDOService service;
    };
    typedef sequence <ServiceProfile> ServiceProfileList;
    struct ConfigurationSet {
        string id;
        string description;
        NVList configuration_data;
    };
    typedef sequence<ConfigurationSet> ConfigurationSetList;

     /** ------- Exceptions -------*/
    exception NotAvailable  exception_body;
    exception InterfaceNotImplemented  exception_body;
    exception InvalidParameter  exception_body;
    exception InternalError  exception_body;

     /** ------- Interfaces -------*/
    interface SDOSystemElement {
        OrganizationList get_owned, organizations()
            raises (NotAvailable);
    };

    interface SDO : SDOSystemElement {
        UniqueIdentifier get_sdo_id()
            raises (NotAvailable, InternalError);
        string get_sdo_type()
            raises (NotAvailable, InternalError);
        DeviceProfile get_device_profile ()
            raises (NotAvailable, InternalError);
        ServiceProfileList get_service_profiles ()
            raises (NotAvailable, InternalError);
        ServiceProfile get_service_profile (in UniqueIdentifier id)
            raises (InvalidParameter, NotAvailable, InternalError);
        SDOService get_sdo_service (in UniqueIdentifier id)
            raises (InvalidParameter, NotAvailable, InternalError);
        Configuration get_configuration ()
            raises (InterfaceNotImplemented, NotAvailable, InternalError);
        Monitoring get_monitoring ()
            raises (InterfaceNotImplemented, NotAvailable, InternalError);
        OrganizationList get_organizations ()
            raises (NotAvailable, InternalError);
        NVList get_status_list ()
```

```
        string manufacturer;
        string model;
        string version;
        NVList properties;
    };
    struct ServiceProfile {
        string id;
        string interface_type;
        NVList properties;
        SDOService service;
};
typedef sequence <ServiceProfile> ServiceProfileList;
struct ConfigurationSet {
        string id;
        string description;
        NVList configuration_data;
};
typedef sequence<ConfigurationSet> ConfigurationSetList;

    /** ------- Exceptions -------*/
exception NotAvailable  exception_body;
exception InterfaceNotImplemented  exception_body;
exception InvalidParameter  exception_body;
exception InternalError  exception_body;

    /** ------- Interfaces -------*/
interface SDOSystemElement {
        OrganizationList get_owned_organizations()
            raises (NotAvailable);
};

interface SDO : SDOSystemElement {
        UniqueIdentifier get_sdo_id()
            raises (NotAvailable, InternalError);
        string get_sdo_type()
            raises (NotAvailable, InternalError);
        DeviceProfile get_device_profile ()
            raises (NotAvailable, InternalError);
        ServiceProfileList get_service_profiles ()
            raises (NotAvailable, InternalError);
        ServiceProfile get_service_profile (in UniqueIdentifier id)
            raises (InvalidParameter, NotAvailable, InternalError);
        SDOService get_sdo_service (in UniqueIdentifier id)
            raises (InvalidParameter, NotAvailable, InternalError);
        Configuration get_configuration ()
            raises (InterfaceNotImplemented, NotAvailable, InternalError);
        Monitoring get_monitoring ()
            raises (InterfaceNotImplemented, NotAvailable, InternalError);
        OrganizationList get_organizations ()
            raises (NotAvailable, InternalError);
        NVList get_status_list ()
```

```
            raises (NotAvailable, InternalError);
        any get_status ()
            raises (InvalidParameter, NotAvailable, InternalError);
        };


    interface Configuration {
        boolean set_device_profile (in DeviceProfile dProfile)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean set_service_profile (in ServiceProfile sProfile)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean add_organization (in Organization organization)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean remove_service_profile (in UniqueIdentifier id)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean remove_organization (in UniqueIdentifier organization_id)
            raises (InvalidParameter, NotAvailable, InternalError);
        ParameterList get_configuration_parameters ()
            raises (NotAvailable, InternalError);
        NVList get_configuration_parameter_values ()
            raises (NotAvailable, InternalError);
        any get_configuration_parameter_value (in string name)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean set_configuration_parameter (
                in string name,
                in any value)
            raises (InvalidParameter, NotAvailable, InternalError);
        ConfigurationSetList get_configuration_sets ()
            raises (NotAvailable, InternalError);
        ConfigurationSet get_configuration_set
                (in UniqueIdentifier config_id)
            raises (NotAvailable, InternalError);
        boolean set_configuration_set_values (
                in UniqueIdentifier config_id,
                in ConfigurationSet configuration_set)
            raises (InvalidParameter, NotAvailable, InternalError);
        ConfigurationSet get_active_configuration_set ()
            raises (NotAvailable, InternalError);
        boolean add_configuration_set
                (in ConfigurationSet configuration_set)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean remove_configuration_set (in UniqueIdentifier config_id)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean activate_configuration_set (in UniqueIdentifier config_id)
            raises (InvalidParameter, NotAvailable, InternalError);
    };


    interface Monitoring : CosNotifyComm::StructuredPushConsumer,
                CosNotifyComm::StructuredPushSupplier {
            any get_monitoring_parameter_value (
```

```
            raises (NotAvailable, InternalError);
        any get_status (in string name)
            raises (InvalidParameter, NotAvailable, InternalError);
        };


    interface Configuration {
        boolean set_device_profile (in DeviceProfile dProfile)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean add_service_profile (in ServiceProfile sProfile)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean add_organization (in Organization organization_object)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean remove_service_profile (in UniqueIdentifier id)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean remove_organization (in UniqueIdentifier organization_id)
            raises (InvalidParameter, NotAvailable, InternalError);
        ParameterList get_configuration_parameters ()
            raises (NotAvailable, InternalError);
        NVList get_configuration_parameter_values ()
            raises (NotAvailable, InternalError);
        any get_configuration_parameter_value (in string name)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean set_configuration_parameter (
                in string name,
                in any value)
            raises (InvalidParameter, NotAvailable, InternalError);
        ConfigurationSetList get_configuration_sets ()
            raises (NotAvailable, InternalError);
        ConfigurationSet get_configuration_set
                (in UniqueIdentifier config_id)
            raises (NotAvailable, InternalError, InvalidParameter);
        boolean set_configuration_set_values (
                in ConfigurationSet configuration_set)
            raises (InvalidParameter, NotAvailable, InternalError);
        ConfigurationSet get_active_configuration_set ()
            raises (NotAvailable, InternalError);
        boolean add_configuration_set
                (in ConfigurationSet configuration_set)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean remove_configuration_set (in UniqueIdentifier config_id)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean activate_configuration_set (in UniqueIdentifier config_id)
            raises (InvalidParameter, NotAvailable, InternalError);
    };


    interface Monitoring : CosNotifyComm::StructuredPushConsumer,
                CosNotifyComm::StructuredPushSupplier {
            any get_monitoring_parameter_value (
                in string name
```

```
                    in string name
                ) raises (InvalidParameter, NotAvailable, InternalError);
            ParameterList get_monitoring_parameters ()
                raises (NotAvailable, InternalError);
            NVList get_monitoring_parameter_values ()
                raises (NotAvailable, InternalError);
        };


        interface SDOService {};


        interface Organization {
            UniqueIdentifier get_organization_id ()
                raises (InvalidParameter, NotAvailable, InternalError);
            OrganizationProperty get_organization_property ()
                raises (NotAvailable, InternalError);
            any get_organization_property_value (in string name)
                raises (InvalidParameter, NotAvailable, InternalError);
            boolean set_organization_property (
                    in OrganizationProperty organization_property
                ) raises (InvalidParameter, NotAvailable, InternalError);
            boolean set_organization_property_value (
                    in string name,
                    in any value
                ) raises (InvalidParameter, NotAvailable, InternalError);
            boolean remove_organization_property ( in string name )
                raises (InvalidParameter, NotAvailable, InternalError);
            SDOSystemElement get_owner ()
                raises (NotAvailable, InternalError);
            boolean set_owner (in SDOSystemElement sdo)
                raises (InvalidParameter, NotAvailable, InternalError);
            SDOList get_members ()
                raises (NotAvailable, InternalError);
            boolean set_members (in SDOList sdos)
                raises (InvalidParameter, NotAvailable, InternalError);
            boolean add_members ( in SDOList sdo_list)
                raises (InvalidParameter, NotAvailable, InternalError);
            boolean remove_member (in UniqueIdentifier id)
                raises (InvalidParameter, NotAvailable, InternalError);
            DependencyType get_dependency()
                raises (NotAvailable, InternalError);
            boolean set_dependency (in DependencyType dependency)
                raises (NotAvailable, InternalError);
        };
    };
    #endif //_SDO_PACKAGE_IDL_
```

```
            ) raises (InvalidParameter, NotAvailable, InternalError);
        ParameterList get_monitoring_parameters ()
            raises (NotAvailable, InternalError);
        NVList get_monitoring_parameter_values ()
            raises (NotAvailable, InternalError);
    };


    interface SDOService {};


    interface Organization {
        UniqueIdentifier get_organization_id ()
            raises (InvalidParameter, NotAvailable, InternalError);
        OrganizationProperty get_organization_property ()
            raises (NotAvailable, InternalError);
        any get_organization_property_value (in string name)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean add_organization_property (
                in OrganizationProperty organization_property
            ) raises (InvalidParameter, NotAvailable, InternalError);
        boolean set_organization_property_value (
                in string name,
                in any value
            ) raises (InvalidParameter, NotAvailable, InternalError);
        boolean remove_organization_property ( in string name )
            raises (InvalidParameter, NotAvailable, InternalError);
        SDOSystemElement get_owner ()
            raises (NotAvailable, InternalError);
        boolean set_owner (in SDOSystemElement sdo)
            raises (InvalidParameter, NotAvailable, InternalError);
        SDOList get_members ()
            raises (NotAvailable, InternalError);
        boolean set_members (in SDOList sdos)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean add_members ( in SDOList sdo_list)
            raises (InvalidParameter, NotAvailable, InternalError);
        boolean remove_member (in UniqueIdentifier id)
            raises (InvalidParameter, NotAvailable, InternalError);
        DependencyType get_dependency()
            raises (NotAvailable, InternalError);
        boolean set_dependency (in DependencyType dependency)
            raises (NotAvailable, InternalError);
    };
};
#endif //_SDO_PACKAGE_IDL_
```