SIEMENS

COMOS

Platform
Class documentation Device_dll

Programming Manual

09/2014
A5E32035361-AA

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E32035361-AA
Ⓟ 09/2014 Subject to change

Table of contents

1 Public Enum

1.1 Enum emLabelRunType

Enum emLabelRunType	
emcRunAsFirst = 0	' Calculating the label for the first time
emcRunAsNext = 1	' Calculating the label, use the old values
emcNotRun = 3	' Do not calculate label

1.2 Enum uceSaveWireType

This constant controls the following shortcut menu:

Interactive report, cable selection: "Cable > Disconnect / Disconnect (retain wires) x / Disconnect (retain wires) y"

Enum uceSaveWireType	If a cable is placed on a connection line that is deleted, the user can decide via the shortcut menu whether or not the cable information should remain at the affected objects.
uccSaveWireNone = 0	The cable information is lost.
uccSaveWireOwn = 1	The wires and therefore the cable information is retained at its own connector.
uccSaveWirePartner = 2	The wires and therefore the cable information is retained at the partner connector.

If the wires are retained, the wire information is also visible in the terminal strip editor and in the cable list.

See also

DisconnectConnectors (Page 11)

Class: Device 2

2.1 Public Functions

2.1.1 CDeviceChange

Switches the base object at an engineering object

```
CDeviceChange(ByVal Device As Object, ByVal CDevice As Object,
Optional ByVal RunType As emLabelRunType, Optional ByVal CheckMasks
As Boolean = True) As Boolean
```

Parameter

- Device
  Is the engineering object for which the base object is switched.
- CDevice
  Is the new base object.
- CheckMasks
  Is optional and defines if the mask entries for name and label should be checked.

Return value

Boolean. Is "True" if the switch of the base object was successful.

2.1.2 ConnectConnectors

Connects two connectors with one another

```
ConnectConnectors(ByVal Connector1 As Object, ByVal Connector2 As
Object, Optional ByVal ScreenMessages As Boolean = True) As Boolean
```

Parameter

- Connector1
  Is the first connector.
- Connector2
  Is the second connector.
- ScreenMessages
  Defines if error messages are supposed to be displayed instantly or not.

Return value

Is "True" if both connectors were successfully connected with each other.

2.1.3 ConnectionPossible

Checks if two connectors can be connected with one another

```
ConnectionPossible(ByVal Connector1 As Object, ByVal Connector2 As
Object, Optional ByVal ScreenMessages As Boolean = True) As Integer
```

Parameter

- Connector1
  Is the first connector.
- Connector2
  Is the second connector.
- ScreenMessages
  Is optional and defines if error messages are to be displayed instantly or not.

Return value

0	If the connection is possible.
-1	If it is an undefined error.
1	If they are the same connections.
2	If the connections have different layers.
3	If the connections have different categories.
4	If the connections own differing wirings.
5	If one or both connections are already occupied.
6	If one or both connections are not part of the engineering object.

2.1.4 CreateDeviceObject

Creates an engineering object

```
CreateDeviceObject(ByVal Owner As Object, ByVal CDevice As
IComosDCDevice, Optional ByVal Name As String, Optional ByVal Class
As String, Optional ByVal Unit As IComosBaseObject, Optional ByVal
Location As IComosBaseObject) As Object
```

Parameter

- Owner
  The owner of the object.
- CDevice
  Is the base object.
- Name
  Is optional and returns the name of the object.
- Unit
  Is optional and sets the possible unit pointer.
- Location
  Is optional and sets the possible location pointer.

Return value

The new engineering object.

2.1.5 CreateDeviceObjectEx

Creates an engineering object

```
CreateDeviceObjectEx(ByVal Owner As Object, ByVal CDevice As
IComosDCDevice, Optional ByVal Name As String, Optional ByVal Class
As String, Optional ByVal Unit As IComosBaseObject, Optional ByVal
Location As IComosBaseObject, Optional ByVal WithMsg As Boolean =
True) As Object
```

Parameter

- Owner
  The owner of the object.
- CDevice
  Is the base object.
- Name
  Is optional and returns the name of the object.
- Unit
  Is optional and sets the possible unit pointer.
- Location
  Is optional and sets the possible location pointer.
- WithMsg
  Is optional and defines if an error message is to be displayed.

Return value

The new engineering object.

2.1.6 CreateTemplate

Copies an assembly group

```
CreateTemplate(ByVal Owner As IComosBaseObject, ByVal TemplateOwner
As IComosDDevice, Optional ByVal CDevice As IComosDCDevice =
Nothing) As Collection
```

Parameter

- Owner
  The object underneath which the assembly group is inserted.
- TemplateOwner
  Imports the by the TemplateOwner stated assembly group.
- CDevice
  Is used in order to decide if the create option "Normal" or "Block" is used. This parameter is only used if it is set.

Return value

A new collection with newly created root objects.

2.1.7 DeleteConnectionForDevice

Dissolves the connection for the engineering object

```
DeleteConnectionForDevice(ByVal DeviceObject As Object, ByVal
ConnectorsLayer As String, Optional ByVal Side As String)
```

Parameter

- DeviceObject
  Is the engineering object.
- ConnectorsLayer
  The connector type that needs to be dissolved.
- Side
  Optional. Is the connection side: Input, Output or all ("I", "O", "").

Return value

No

2.1.8 DeviceNewByPrototype

Creates a new engineering object according to the structure of an already existing engineering object

The new engineering object which is created has the same base object, the same unit pointer, the same location pointer and the same specification values as the prototype.

```
DeviceNewByPrototype(ByVal Prototype As IComosDDevice) As
IComosDDevice
```

Parameter

- Prototype
  An existing engineering object.

Return value

A new engineering object.

2.1.9 DisconnectConnectors

Dissolves the connection between two connectors

```
DisconnectConnectors(ByVal CurrentConnector As IComosDConnector,
Optional ByVal PartnerCon As IComosDConnector, Optional ByVal
SaveWireType As uceSaveWireType = uccSaveWireNone) As Boolean
```

Parameter

- CurrentConnector
  Is the connection.
- PartnerCon
  Optional. Is the second connector.
- SaveWireType
  Optional. Defines if also the wires that are part of the connection are supposed to be dissolved.
  See also section Enum uceSaveWireType (Page 5).

Return value

Is "True" if the operation was successful.

2.1.10 DocumentsSearchAndAssume

Creates all predefined documents for a given engineering object

```
DocumentsSearchAndAssume(ByVal Device As Object)
```

Parameter

- Device
  Is the engineering object.

Return value

No

2.1.11 GetChildsForNew

Offers possible base objects for the creation of new objects under a given owner

```
GetChildsForNew(ByVal Owner As Object, ByVal CDevice As
IComosDCDevice, ByVal WithSubsFromBlock As Boolean) As Collection
```

Parameter

- Owner
  Is the owner.
- CDevice
  Is the base object.
- WithSubsFromBlock
  Defines if the elements of the create option "Block" are considered.

Return value

A list of base objects.

2.1.12 GetCObject

Returns the base object of a given object

```
GetCObject(ByVal ComosObject As IComosBaseObject) As IComosDCDevice
```

Parameter

ComosObject

Is a COMOS object. Following object types are considered:

SystemTypeDevice	
SystemTypeUnit	
SystemTypeLocation	
SystemTypeCDevice	
SystemTypeProject	
SystemTypeDocument	

Return value

Is the base object.

2.1.13 GetComosObject

Returns a COMOS object according to the stated parameters

```
GetComosObject(ByVal WorkSet As IComosDWorkset, ByVal SystemUID As
String, ByVal SystemType As Long, ByVal PathFullName As String) As
IComosBaseObject
```

Parameter

- WorkSet
  Is the COMOS workset.
- SystemUID
  Is the SystemUID of the object.
- SystemType
  Is the system type of the object.
- PathFullName
  Is the PathFullName of the object, relative to the project.

Return value

The found COMOS object.

2.1.14 GetDeviceByDocObj

Returns the referenced engineering object of DocObj objects

```
GetDeviceByDocObj(ByVal DocObjObject As Object) As Object
```

Parameter

- DocObjObject
  Is the DocObj object.

2.1.15 GetDeviceLevel

States the relative level of an element in accordance with a main object (MainDevice)

```
GetDeviceLevel(ByVal CurrentDevice As Object) As Integer
```

Parameter

- CurrentDevice
  Is the engineering object.

Return value

Is the level on which the element is located.

2.1.16 GetIdentTextForComosObject

Generates the standard display text for a COMOS object

```
GetIdentTextForComosObject(ByVal ComosObject As IComosBaseObject) As
String
```

Parameter

- ComosObject
  Is the COMOS object for which the text is to be generated.

Return value

Is the display text.

2.1.17 GetMainDevice

Identical to the MainDevice function of the kernel. Exists due to compatibility reasons. See also the kernel documentation "comos.dll".

```
GetMainDevice(ByVal CurrentDevice As Object) As Object
```

2.1.18 GetTmpCollection

Converts the first parameter regardless of type into IComosDCollection

```
GetTmpCollection(ByVal ComosObjects As Object, Optional ByVal STypes
As VBA.Collection) As IComosDCollection
```

Parameter

- ComosObjects
  Can be one or more COMOS objects. Following types are supported: IComosBaseObject, VBACollection, IComosDCollection.
- STypes
  Is optional and filters the object according to the stated types.

Return value

Listing of objects of the IComosDCollection type.

2.1.19 IsInheritCheckIn

Imports the object, if possible

```
IsInheritCheckIn(ByVal ComosObject As Object) As Boolean
```

Parameter

- ComosObject
  Is the object on the engineering side which is to be imported.

Return value

Is "True" if the operation was successful.

2.1.20 InheritCheckIn

Imports the object, if possible

```
InheritCheckIn(ByVal ComosObject As Object, ByVal ShowMessage As
Boolean, ByVal CreateMessage As Boolean) As Boolean
```

Parameter

- ComosObject
  Is the object on the engineering side which is to be imported.
- ShowMessage
  Displays error messages.
- CreateMessage
  Defines if error messages are written to an own message pack.

Return value

Is "True" if the operation was successful.

2.1.21 IsMainDevice

Checks if an entered object is a main object (MainDevice)

```
IsMainDevice(ByVal CurrentDevice As Object) As Boolean
```

Parameter

- CurrentDevice
  Is the engineering object.

Return value

Is "True" if the engineering object is a main object (MainDevice).

2.1.22 IsImplementation

Checks if a COMOS object is valid as an implementation pointer for the entered request

```
IsImplementation(ByVal ComosObject As IComosBaseObject, ByVal
Request As IComosDDevice) As Boolean
```

Parameter

- ComosObject
  The checked engineering object.
- Request
  The entered request.

Return value

Is "True" if the checked object can be used as an implementation.

2.1.23 IsImplemented

Checks if the object has already been used as an implementation

```
IsImplemented(ByVal curObj As IComosDDevice) As Boolean
```

Parameter

- curObj
  Is the to be checked engineering object.

Return value

Is "True" if the object was used as an implementation.

2.1.24 IsObjectSystemRO

Checks if the object is write-protected

```
IsObjectSystemRO(ByVal ComosObject As Object) As Boolean
```

Parameter

- ComosObject
  Is the COMOS object for which the evaluation is to be conducted.

Return value

Is "True" if the object is write-protected.

2.1.25 IsSetAllowed

Checks if a pointer can be set for an entered COMOS object

```
IsSetAllowed(ByVal ComosObject As IComosBaseObject, ByVal PointerObj
As Object, ByVal PropName As String, ByVal WithMsg As Boolean,
Optional ByVal CreateMessage As Boolean = True) As Boolean
```

Parameter

- ComosObject
  Is the COMOS object for which the evaluation is to be conducted.
- PointerObj
  The pointer which is to be set.
- PropName
  The name of the property. Following properties are supported:
  - CDevice
  - Cobject
  - Location
  - Unit
  - Alias
  - Implementation
  - Link.Object
  - StandardTable
  - Signal
  - Potential
  - Connected with
  - Wire
- WithMsg
  Defines if an error message is to be displayed.
- CreateMessage
  Is optional and defines that the error message are written in an own message pack.

Return value

Is "True" if you are allowed to set the pointer.

2.1.26 IsWriteAllowed

Checks if you can change value type properties for a COMOS object

```
IsWriteAllowed(ByVal ComosObject As IComosBaseObject, Optional ByVal
WithMsg As Boolean = False, Optional ByVal PropName As String,
Optional ByVal PropParameter As Variant) As Boolean
```

Parameter

- ComosObject
  Is the COMOS object for which the evaluation is to be conducted.
- WithMsg
  Defines if an error message is to be displayed.
- PropName
  Optional. The name of the property.
- PropParameter
  Optional. Is the parameter of the property.

Return value

Is "True" if the property can be changed.

2.1.27 IsWriteAllowedByValue

Checks if you can change value type properties for a COMOS object

```
IsWriteAllowedByValue(ByVal ComosObject As IComosBaseObject,
Optional ByVal WithMsg As Boolean = False, Optional ByVal PropName
As String, Optional ByVal PropParameter As Variant, Optional ByVal
vNewValue As Variant) As Boolean
```

Parameter

- ComosObject
  Is the COMOS object for which the evaluation is to be conducted.
- WithMsg
  Defines if an error message is to be displayed.
- PropName
  Optional. The name of the property.
- PropParameter
  Optional. Is the parameter of the property.
- vNewValue
  Optional. Is the value which is to be set.

Return value

Is "True" if the property can be changed.

2.1.28 LanguageDescription

Provides the description of the language

```
LanguageDescription(ByVal Language As IComosDLanguage, Optional
Index As Integer = -1) As String
```

Parameter

- Language
  Is the COMOS language object.
- Index
  Optional. Is the index of the language in the COMOS collection which is to be translated.

Return value

Is the text in the selected language.

2.1.29 LockedStatusText

Returns a text which states why the property of an object is protected

```
LockedStatusText(ByVal ComosObject As IComosBaseObject, ByVal
PropName As String, ByVal PropParameter As Variant) As String
```

Parameter

- ComosObject
  Is the COMOS object for which the property is write protected.
- PropName
  The name of the property which is write protected.
- PropParameter
  Parameter of the property which is write protected.

Return value

The status text which is displayed in the status bar of the application.

2.1.30 SelectPointer

Opens a window for the reference selection

```
SelectPointer(ByVal ComosObject As IComosBaseObject, ByVal PropName
As String, Optional ByVal WithMsg As Boolean = False, Optional ByRef
Cancel As Boolean = False) As IComosBaseObject
```

Parameter

- ComosObject
  Is the document or the specification.
- PropName
  Only two properties are supported: CDocument and LinkObject.
- WithMsg
  Is optional and defines if an error message is to be displayed.
- Cancel
  Is optional. For reference Cancel = True, when the action is aborted.

Return value

The selected COMOS object.

2.1.31 SetImplementation

Implements the request

```
SetImplementation(ByVal Request As IComosDDevice, ByVal
Implementation As IComosDDevice, Optional ByVal WithMsgByReplace As
Boolean = False) As Integer
```

Parameter

- Request
  Is the request object which is to be implemented.
- Implementation
  Is the implementation object.
- WithMsgByReplace
  Is optional and defines if an error message is to be displayed.

Return value

Is 0 if the implementation was successful.

2.1.32 SetLocation

Sets the location pointer for a COMOS object

```
SetLocation(ByVal ComosObject As IComosBaseObject, ByVal PointerObj
As Object, Optional ByVal WithMsg As Boolean = False) As Boolean
```

Parameter

- ComosObject
  The COMOS object for which the location pointer is to be set.
- PointerObj
  The location pointer which is to be set.
- WithMsg
  Is optional and defines if an error message is to be displayed.

Return value

Is "True" if the operation was successful.

2.1.33 SetLabel

Sets a label for a COMOS object

```
SetLabel(ByVal ComosObject As IComosBaseObject, ByVal NewLabel As
String, Optional ByVal WithMsg As Boolean = False) As Boolean
```

Parameter

- ComosObject
  The COMOS object for the label is to be set.
- NewLabel
  The new name.
- WithMsg
  Is optional and defines if an error message is to be displayed.

Return value

Is "True" if the operation was successful.

2.1.34 SetLock

Blocks or releases an engineering object

```
SetLock(ByVal ComosObj As Object, ByVal vNewValue As Boolean, ByVal
Recursive As Integer, ByVal WithMsg As Boolean) As Boolean
```

Parameter

- ComosObj
  Is a engineering object or a document.
- vNewValue
  Is either "True" or "False".
- Recursive
  Defines if the child objects should also be blocked or released.
  There are the following values:

0	Non-recursive
1	Recursive
2	With a query, for a recursive or non-recursive procedure.

- WithMsg
  Defines if an error message is to be displayed.

2.1.35 SetName

Sets a new name for a COMOS object

```
SetName(ByVal ComosObject As IComosBaseObject, ByVal NewName As
String, Optional ByVal WithMsg As Boolean = False) As Boolean
```

Parameter

- ComosObject
  The COMOS object for which a name is to be set.
- NewName
  The new name.
- WithMsg
  Is optional and defines if an error message is to be displayed.

Return value

Is "True" if the operation was successful.

2.1.36 SetPointer

Checks if a pointer can be set and sets the pointer if the check was successful

```
SetPointer(ByVal ComosObject As IComosBaseObject, ByVal PointerObj
As Object, ByVal PropName As String, Optional ByVal WithMsg As
Boolean = False) As Boolean
```

Parameter

- ComosObject
  The COMOS object for which the pointer can be set.
- PointerObj
  The to be set pointer object.
- PropName
  The name of the property. Following properties are supported:
  - CDevice
  - CObject
  - CDocument
  - Location
  - Unit
  - Alias
  - Implementation
  - Link.Object
  - StandardTable
  - Signal
  - Potential
  - ConnectedWith
  - Wire
  - Cable
- WithMsg
  Is optional and defines if an error message is to be displayed.

Return value

Is "True" if the operation was successful.

2.1.37 SetUnit

Sets the unit pointer for a COMOS object

```
SetUnit(ByVal ComosObject As IComosBaseObject, ByVal PointerObj As
Object, Optional ByVal WithMsg As Boolean = False) As Boolean
```

Parameter

- ComosObject
  The COMOS object for which the unit pointer is to be set.
- PointerObj
  The unit pointer which is to be set.
- WithMsg
  Is optional and defines if an error message is to be displayed.

Return value

Is "True" if the operation was successful.

3 Class: Implement

3.1 Public Functions

3.1.1 CheckConnectors

Checks connectors according to their names, types and subtypes

```
CheckConnectors(ByVal DeviceFrom As IComosDDevice, ByVal DeviceTo As
IComosDDevice, ByVal ConnectorType As String) As Integer
```

Parameter

- DeviceFrom
  Is a base engineering object whose connections are supposed to be checked with another engineering object.
- DeviceTo
  Is the engineering object which is to be checked.
- ConnectorType
  The connector type which is checked.

Return value

0	Check is OK.
4	The engineering objects have different connections.
8	The connections have different types.
9	The connections have different subtypes.

3.1.2 CheckSpecificationsValue

Checks the attributes of engineering objects with regard to their content

```
CheckSpecificationsValue(ByVal DeviceFrom As IComosDDevice, ByVal
DeviceTo As IComosDDevice, ByVal SpecNestedName As String) As Integer
```

Parameter

- DeviceFrom
  Is a base engineering object whose attributes are supposed to be checked with another engineering object.
- DeviceTo
  Is the engineering object which is to be checked.
- SpecNestedName
  Is the name of the attribute which is to be checked.

Return value

0	The check was successful.
11	Different attributes are existent.
12	The attributes have different values.

3.1.3 DisconnectOldRequest

Dissolves an implementation

```
DisconnectOldRequest(ByVal Implementation As IComosDDevice) As
IComosDDevice
```

Parameter

- Implementation
  Is an engineering object.

Return value

Is the engineering object which is the dissolved request.

3.1.4 EnableRestore

Informs if the implementation can be dissolved

```
EnableRestore(ByVal Implementation As IComosDDevice, ByVal Request
As IComosDDevice) As Boolean
```

Parameter

- Implementation
  Is the manufacturer device which was implemented.
- Request
  Is the request that was implemented.

Return value

Is "True" if the operation was successful.

3.1.5 InfoText

Returns information regarding the result number

```
InfoText(ByVal NumberCode As Integer) As String
```

Parameter

- NumberCode
  Is the result number.

Return value

Information text.

3.1.6 IsEnabledRestoreRequest

States if an engineering object owns the information to restore a request

```
IsEnabledRestoreRequest(ByVal Implementation As IComosDDevice) As
Boolean
```

Parameter

- Implementation
  Is the engineering object which functions as a manufacturer device.

Return value

Is "True" if the operation was successful.

3.1.7 IsUsedAsImplementation

Checks it the object is already set as an implementation on another object

```
IsUsedAsImplementation(ByVal Device As IComosDDevice) As Boolean
```

Parameter

- Device
  The engineering object to be evaluated.

Return value

Is "True" if the operation was successful.

3.1.8 IsUsedAsRequest

Checks if an object owns an implementation

```
IsUsedAsRequest(ByVal Device As IComosDDevice) As Boolean
```

Parameter

- Device
  The engineering object to be evaluated.

Return value

Is "True" if the operation was successful.

3.2 Public Subs

3.2.1 CopySpecificationsValue

Copies the attributes OwnValue, GetOwnXValue, OwnUnit and OwnLinkObject

```
CopySpecificationsValue(ByVal DeviceFrom As IComosDDevice, ByVal
DeviceTo As IComosDDevice, ByVal SpecNestedName As String, ByVal
Prio As Integer)
```

Parameter

- DeviceFrom
  Is the engineering object from which is copied.
- DeviceTo
  Is the engineering object in which is copied to.
- SpecNestedName
  Is the name of the attribute which is to be copied.
- Prio
  Is a number which states if the value of the to be replaced attribute is to be overwritten if it already exists

3.2.2 MoveConnectorsInfo

Moves the connection values of an engineering object to another engineering object

```
MoveConnectorsInfo(ByVal DeviceFrom As IComosDDevice, ByVal DeviceTo
As IComosDDevice, ByVal ConnectorType As String, ByVal Prio As
Integer, ByVal Recursive As Boolean)
```

Parameter

- DeviceFrom
  The engineering object from which the values are taken.
- DeviceTo
  The engineering object to which the values are passed to.
- ConnectorType
  Is the connection type which is to be moved.
- Prio
  Is a number which defines if the value of the connection to be replaced is to be overwritten if it already exists.
- Recursive
  Defines if the new values should also be applied to the connected connections.

3.2.3 MoveDocObjs

Moves DocObjs from one engineering object to another

```
MoveDocObjs(ByVal DeviceFrom As IComosDDevice, ByVal DeviceTo As
IComosDDevice, ByVal DocumentTypeName As String, ByVal SymbolType As
String)
```

Parameter

- DeviceFrom
  Is the engineering object from which the objects are moved.
- DeviceTo
  Is the engineering object to which the objects are moved to.
- DocumentTypeName
  The name of the document type.
- SymbolType
  The symbol type of the document.

3.2.4 MoveSpecificationLinks

Moves links of attributes from an engineering object to another

```
MoveSpecificationLinks(ByVal DeviceFrom As IComosDDevice, ByVal
DeviceTo As IComosDDevice)
```

Parameter

- DeviceFrom
  The engineering object from which the links are moved.
- DeviceTo
  The engineering object to which the links are moved to.

3.2.5 SetUnit

Sets the unit pointer from one object to another

```
SetUnit(ByVal DeviceFrom As IComosDDevice, ByVal DeviceTo As
IComosDDevice)
```

Parameter

- DeviceFrom
  Is the object from which the pointers derive.
- DeviceTo
  Is the object on which the pointer is set.

Class: InfoToCreateVE

4

Overview

Management class for virtual elements. This class controls whether the elements are also automatically created when a new device is created or not.

Class: Message 5

Overview

This class creates objects of the type "Message". These objects are used by the class MessagePack. See chapter Class: MessagePack (Page 37).

Class: MessagePack 6

Overview

If errors are found in COMOS, two procedures are available to report these errors to the user:

1. A MessageBox is generated when the error occurs
   or
2. The errors are collected and displayed as an error package after the procedure is finished. The errors in the error package can be listed in a table, for example. This procedure is recommended for mass operations.

The second procedure uses the class MessagePack. The class MessagePack collects objects of the type Message.

See also

Class: Message (Page 35)