Workspace Manager API Reference

Summary

This reference provides a concise reference of the Workspace API as part of the Altium Designer Run Time Library.

The WorkSpace Manager Application Programming Interface reference covers the Workspace manager object interfaces from the Workspace Manager Object Model.

The workspace manager interfaces exist as long there are associated existing objects in memory, thus when writing a script or server code, you have the responsibility of checking whether the interface you wish to query exists or not before you proceed to invoke the interface's methods.

You will have to ensure that the design project is compiled first otherwise the workspace manager interfaces are in an invalid state and will be returning nil values.

The workspace manager provides a bridge between source documents (such as Schematic documents) and its corresponding primary implementation documents (such as PCB documents). This workspace manager provides you information on how a project is structured, and information on nets and its associated net aware objects of source and implementation documents.

The **IWorkSpace** interface deals with projects, documents and objects on the open documents. To use workspace interfaces, the project needs to be compiled first refreshing all the linkages and nets up to date.

Main Workspace Manager interfaces

- The **IDMObject** interface is a generic interface used for all other WorkSpace interfaces.
- The **IWorkSpace** interface is the top level interface and contains many interfaces within. For example the **IWorkSpace** interface has a **DM_OpenProject** function which returns a currently open or currently focussed **IProject** interface.
- The **IProject** interface represents the current project in Altium Designer.
- The **IDocument** interface represents a document in Altium Designer.

Logical and Physical Documents

An important note, there are logical and physical documents; these terms are used to differentiate the documents in multi-channel projects. A multi channel design means that a single sheet is referenced repeatedly for a channel design. This sheet is called a logical document. A physical document (usually a PCB document) has components with unique names within a room which is mapped to a channel on a Schematic sheet. So a multi channel design translates to multiple rooms with components with unique physical designators on a PCB.

A physical designator of a PCB component is calculated to have the hierarchy path of a schematic project as well as the logical designator of the associated Schematic component to ensure that this designator for the PCB component is unique.

Example

Obtaining the project path from the current **IProject** interface.

```
// Get WSM interface (the shell of the WorkSpace Manager interface).
WSM := GetWorkSpace;
If WSM = Nil Then Exit;
Document := WSM.DM_Document;
If Document = Nil Then Exit;
Project := Document.DM_Project;
```

Script Examples

There are script examples in the **\Examples\Scripts\WSM** folder

See also

WorkSpace Manager Interface Overview

WorkSpace Manager Interfaces

WorkSpace Manager Enumerated Types

WorkSpace Manager Functions

Workspace Manager Object Model

To have access to the workspace interface object which represents the workspace manager in Altium Designer, you need to invoke the **GetWorkspace** function first. This function returns you the **IWorkspace** interface object. An object interface is just a means of access to an object in memory.

The workspace manager provides a bridge between source documents (such as Schematic documents) and its corresponding primary implementation documents (such as PCB documents). This workspace manager provides you the ability to manipulate the contents of a design project in Altium Designer.

A few of Workspace manager Object Interfaces

The **IDMObject** interface is a generic interface used for all other WorkSpace interfaces.

The **IWorkSpace** interface is the top level interface and contains many interfaces within. For example the IWorkSpace interface has a **DM_OpenProject** function which returns a currently open or currently focussed IProject interface.

The **IProject** interface represents a design project in Altium Designer.

The **IPart** interface represents a part of a multi-part component. This component is represented by the IComponent interface.

The **IDocument** interface represents a document in Altium Designer.

The **IECO** interface is used for the Engineering Change Order system in PCB and Schematic servers.

The **INet** interface is a container storing Net aware objects (which are INetItem interfaces) that have the same net property. So there are INet interfaces representing nets on a document.

The **INetItem** interface is the ancestor interface for the Cross, Pin, Port, Netlabel, Sheet entry and Power Object interfaces. These interface objects have a net property and thus these objects can be part of a net.

Document interface

Overview

The `IDocument` interface represents the existing document in Altium Designer. A document can be a Schematic, PCB, VHDL, PCB Library document etc.

The `DM_DocumentKind` method of the `IDoc`ument interface when invoked returns you the document type. A document can be part of a project or free documents project.

An existing document can be queried to return the project interface this document is associated with.

Notes

The **IDocument** interface is a standalone interface.

IDocument methods

DM_BusCount

DM_Buses

DM_ChannelClassCount

DM_ChannelClasses

DM_ChannelIndex

DM_ChannelPrefix

DM_ChannelRoomNamingStyle

DM_ChildDocumentCount

DM_ChildDocuments

DM_Compile

DM_ComponentClassCount

DM_ComponentClasses

DM_ComponentCount

DM_Components

DM_ConstraintGroupCount

IDocument properties

DM_ConstraintGroups
DM_CreateViolation
DM_CrossSheetConnectorCount
DM_CrossSheetConnectors
DM_CurrentInstanceNumber
DM_DifferentialPairs
DM_DifferentialPairCount
DM_DocumentIsLoaded
DM_DocumentIsTextual
DM_DocumentKind
DM_FileName
DM_FullPath
DM_IndentLevel
DM_IsPhysicalDocument
DM_IsPrimaryImplementationDocument
DM_LoadDocument
DM_LogicalDocument
DM_ModelKind
DM_NetClassCount
DM_NetClasses
DM_NetCount
DM_Nets
DM_ParentDocumentCount
DM_ParentDocuments
DM_PartCount
DM_Parts
DM_PhysicalDocumentCount
DM_PhysicalDocumentParent
DM_PhysicalInstanceName
DM_PhysicalInstancePath
DM_PhysicalRoomName
DM_PortCount
DM_Ports
DM_Project
DM_RoomCount
DM_Rooms
DM_RuleCount
DM_Rules
DM_ScrapCompile
DM_SheetSymbolCount
DM_SheetSymbols
DM_SignalManager
DM_TextFrameCount
DM_TextFrames
DM_UniqueComponentCount
DM_UniqueComponents

DM_UniquePartCount

DM_UniqueParts

DM_UpdateDateModified

DM_VHDLEntities

DM_VHDLEntityCount

See also

Methods

DM_BusCount method

(IDocument interface)

Syntax

```
Function DM_BusCount : Integer;
```

Description

The function returns the number of bus objects from this document. Use this in conjunction with the DM_Buses(Index) to go through each bus object.

See also

IDocument interface

DM_Buses method

(IDocument interface)

Syntax

```
Function DM_Buses (Index : Integer) : IBus;
```

Description

The function returns the indexed Bus instance from this document.

See also

IDocument interface

DM_ChannelClassCount method

(IDocument interface)

Syntax

```
Function DM_ChannelClassCount : Integer;
```

Description

The function denotes the number of Channel Classes from this document. Use this Channel Class count in conjunction with the DM_ChannelClasses(index) to go through each channel class.

See also

IDocument interface

DM_ChannelClasses method

(IDocument interface)

Syntax

```
Function DM_ChannelClasses (Index : Integer) : IChannelClass;
```

Description

The function returns the indexed ChannelClass instance from this document. Use this in conjunction with the DM_ChannelClassCount function

See also

IDocument interface

DM_ChannelIndex method

(IDocument interface)

Syntax

```
Function DM_ChannelIndex : Integer;
```

Description

The function returns the channel index of this document. This is especially for multi-channel designs where a single source document can be referenced multiple times.

See also

IDocument interface

DM_ChannelPrefix method

(IDocument interface)

Syntax

```
Function DM_ChannelPrefix : WideString;
```

Description

The function returns the channel prefix of this document. This is especially for multi-channel designs where a single source document can be referenced multiple times.

See also

IDocument interface

DM_ChannelRoomNamingStyle method

(IDocument interface)

Syntax

```
Function DM_ChannelRoomNamingStyle : TChannelRoomNamingStyle;
```

Description

The function returns the channel room naming style value.

See also

IDocument interface

DM_ChildDocumentCount method

(IDocument interface)

Syntax

```
Function DM_ChildDocumentCount : Integer;
```

Description

The function returns the number of child documents relative to this document.

See also

IDocument interface

DM_ChildDocuments method

(IDocument interface)

Syntax

```
Function DM_ChildDocuments (Index : Integer) : IDocument;
```

Description

The function returns the indexed child document. A hierarchical design consists of multi layered parent-child documents.

See also

IDocument interface

DM_Compile method

(IDocument interface)

Syntax

```
Function DM_Compile : LongBool;
```

Description

The function invokes the compiler to compile this document. If the compile was successful, a true value is returned.

See also

IDocument interface

DM_ComponentClassCount method

(IDocument interface)

Syntax

```
Function DM_ComponentClassCount : Integer;
```

Description

The function denotes the number of component classes from this document. Use this Component class count in conjunction with the DM_ComponentClasses(index) to go through each component class.

See also

IDocument interface

DM_ComponentClasses method

(IDocument interface)

Syntax

```
Function DM_ComponentClasses (Index : Integer) : IComponentClass;
```

Description

The function returns the indexed ComponentClass instance from this document. Use this in conjuniton with the DM_ComponentClassCount function.

See also

IDocument interface

DM_ComponentCount method

(IDocument interface)

Syntax

```
Function DM_ComponentCount : Integer;
```

Description

The function returns the number of component instances on this document. Use this in conjunction with the DM_Components(Index) method to go through each component object.

See also

IDocument interface

DM_Components method

(IDocument interface)

Syntax

```
Function DM_Components (Index : Integer) : IComponent;
```

Description

The function returns the indexed component instance from this document. This is to be used in conjunction with the DM_ComponentCount method.

See also

IDocument interface

DM_ConstraintGroupCount method

(IDocument interface)

Syntax

```
Function DM_ConstraintGroupCount : Integer;
```

Description

The function denotes the number of constraint groups.

See also

IDocument interface

DM_ConstraintGroups method

(IDocument interface)

Syntax

```
Function DM_ConstraintGroups (Index : Integer) : IConstraintGroup;
```

Description

The function returns the indexed constraint group. Use the DM_ConstraintGroupCount function to get the number of constraint groups.

See also

IDocument interface

DM_CreateViolation method

(IDocument interface)

Syntax

```
Function DM_CreateViolation (AErrorKind : TErrorKind;AErrorString : WideString) : IViolation;
```

Description

The function creates a violation based on the error kind and error string upon an incorrect design.

See also

IDocument interface

DM_CrossSheetConnectorCount method

(IDocument interface)

Syntax

```
Function DM_CrossSheetConnectorCount : Integer;
```

Description

The function returns the number of cross sheet connectors on this document. Use this in conjunction with the DM_CrossConnectors(index) to go through each cross connector object.

See also

IDocument interface

DM_CrossSheetConnectors method

(IDocument interface)

Syntax

```
Function DM_CrossSheetConnectors (Index : Integer) : ICrossSheet;
```

Description

The function returns the indexed cross sheet connector instance from this document. This is to be used in conjunction with the DM_CrossSheetConnectorCount method.

See also

IDocument interface

DM_CurrentInstanceNumber method

(IDocument interface)

Syntax

```
Function DM_CurrentInstanceNumber : Integer;
```

Description

The function returns the current instance number for this document (especially for multi–channel designs where a design document can be referenced multiple times).

Example

See also

IDocument

DM_DifferentialPairs method

(IDocument interface)

Syntax

```
Function DM_DifferentialPairs(Index : Integer) : IDifferentialPair;
```

Description

This function returns an indexed differential pair from a document in the project.

See also

IDocument interface

IDifferentialPair interface

DM_DifferentialPairCount method

(IDocument interface)

Syntax

```
Function    DM_DifferentialPairCount          : Integer;
```

Description

This function returns the number of differential pairs used in a document in the project.

See also

IDocument interface

IDifferentialPair interface

DM_DocumentIsLoaded method

(IDocument interface)

Syntax

```
Function DM_DocumentIsLoaded : Boolean;
```

Description

This function returns a boolean value whether this document has been loaded in Altium Designer or not.

See also

IDocument

DM_DocumentIsTextual method

(IDocument interface)

Syntax

```
Function DM_DocumentIsTextual : Boolean;
```

Description

The function denotes whether the document is a text document.

See also

IDocument interface

DM_DocumentKind method

(IDocument interface)

Syntax

```
Function DM_DocumentKind : WideString;
```

Description

This function returns the document kind for the current document. A document could be a Schematic document and thus the string returned is ‘SCH’. Check the installation file of each server for the Server Name.

Example

See also

IDocument

DM_FileName method

(IDocument interface)

Syntax

```
Function DM_FileName : WideString;
```

Description

This function returns the file name string of this document.

See also

IDocument

DM_FullPath method

(IDocument interface)

Syntax

```
Function DM_FullPath : WideString;
```

Description

This function returns the full path of where this document lives.

See also

IDocument

DM_IndentLevel method

(IDocument interface)

Syntax

```
Function DM_IndentLevel : Integer;
```

Description

The function returns the indent level for this current document with respect to the current project.

See also

IDocument interface

DM_IsPhysicalDocument method

(IDocument interface)

Syntax

```
Function DM_IsPhysicalDocument : Boolean;
```

Description

This function returns a Boolean value whether this document is a physical document or not. There are logical and physical documents; these terms are used to differentiate the documents in multi-channel projects. A multi channel design means that a single sheet is referenced repeatedly for a channel design. This sheet is called a logical document. A physical document (usually a PCB document) has components with unique names within a room which is mapped to a channel on a Schematic sheet. So a multi channel design translates to multiple rooms with components with unique physical designators on a PCB.

A physical designator of a PCB component is calculated to have the hierarchy path of a schematic project as well as the logical designator of the associated Schematic component to ensure that this designator for the PCB component is unique.

See also

IDocument

DM_IsPrimaryImplementationDocument method

(IDocument interface)

Syntax

```
Function DM_IsPrimaryImplementationDocument : Boolean;
```

Description

This function returns a Boolean value whether this document is a primary implementation document (namely a PCB document for instance). A schematic document is a source document and is centric to a design project.

Example

See also

IDocument

DM_LoadDocument method

(IDocument interface)

Syntax

```
Function DM_LoadDocument : Boolean;
```

Description

This function returns a Boolean value whether this document has been loaded or not.

Example

See also

IDocument

DM_LogicalDocument method

(IDocument interface)

Syntax

```
Function DM_LogicalDocument : IDocument;
```

Description

This function returns the logical document if valid. Otherwise a nil value is returned. There are logical and physical documents; these terms are used to differentiate the documents in multi-channel projects. A multi channel design means that a single sheet is referenced repeatedly for a channel design. This sheet is called a logical document. A physical document (usually a PCB document) has components with unique names within a room which is mapped to a channel on a Schematic sheet. So a multi channel design translates to multiple rooms with components with unique physical designators on a PCB.

A physical designator of a PCB component is calculated to have the hierarchy path of a schematic project as well as the logical designator of the associated Schematic component to ensure that this designator for the PCB component is unique.

See also

IDocument

DM_ModelKind method

(IDocument interface)

Syntax

```
Function DM_ModelKind : WideString;
```

Description

The function returns the model kind string related to this document.

See also

IDocument interface

DM_NetClassCount method

(IDocument interface)

Syntax

```
Function DM_NetClassCount : Integer;
```

Description

The function denotes the number of net classes on this document. Use this NetClass count in conjunction with the DM_NetClasses(Index) method to go through each net class.

See also

IDocument interface

DM_NetClasses method

(IDocument interface)

Syntax

```
Function DM_NetClasses (Index : Integer) : INetClass;
```

Description

The function returns the indexed NetClass instance from this document. Use this in conjunction with the DM_NetClassCount function.

See also

IDocument interface

DM_NetCount method

(IDocument interface)

Syntax

```
Function DM_NetCount : Integer;
```

Description

The function returns the number of nets from this document. Use this Net count in conjunction with the DM_Nets(Index) to go through each sheet symbol object

See also

IDocument interface

DM_Nets method

(IDocument interface)

Syntax

```
Function DM_Nets (Index : Integer) : INet;
```

Description

The function returns an indexed net associated with this document.

See also

IDocument interface

DM_ParentDocumentCount method

(IDocument interface)

Syntax

```
Function DM_ParentDocumentCount : Integer;
```

Description

The function returns the number of parent documents relative to this document.

See also

IDocument interface

DM_ParentDocuments method

(IDocument interface)

Syntax

```
Function DM_ParentDocuments (Index : Integer) : IDocument;
```

Description

The function returns the indexed parent document. A hierarchical design consists of multi layered parent-child documents.

See also

IDocument

DM_PartCount method

(IDocument interface)

Syntax

```
Function DM_PartCount : Integer;
```

Description

The function returns the number of part objects from this document. Use this PartCount in conjunction with the DM_Parts(Index) to go through each part object.

See also

IDocument interface

DM_Parts method

(IDocument interface)

Syntax

```
Function DM_Parts (Index : Integer) : IPart;
```

Description

The function returns an indexed part associated with this document.

See also

IDocument interface

DM_PhysicalDocumentCount method

(IDocument interface)

Syntax

```
Function DM_PhysicalDocumentCount : Integer;
```

Description

The function returns the number of physical documents associated with this document.

See also

IDocument interface

DM_PhysicalDocumentParent method

(IDocument interface)

Syntax

```
Function DM_PhysicalDocumentParent : IDocument;
```

Description

The function returns the IDocument interface for a parent physical document. Could be a VHDL or a PCB document for example.

See also

IDocument interface

DM_PhysicalInstanceName method

(IDocument interface)

Syntax

```
Function DM_PhysicalInstanceName : WideString;
```

Description

The function returns the name of this physical document if valid. Otherwise an empty string is returned.

Example

See also

IDocument

DM_PhysicalInstancePath method

(IDocument interface)

Syntax

```
Function DM_PhysicalInstancePath : WideString;
```

Description

The function returns the path to the physical document instance if valid. Otherwise an empty string is returned.

See also

IDocument interface

DM_PhysicalRoomName method

(IDocument interface)

Syntax

```
Function DM_PhysicalRoomName : WideString;
```

Description

The function returns the name of the room on this physical document if valid. Otherwise a nil value is returned.

See also

IDocument interface

DM_PortCount method

(IDocument interface)

Syntax

```
Function DM_PortCount : Integer;
```

Description

The function returns the number of port objects on this document. Use this in conjunction with the DM_Ports(index) to go through each port object.

See also

IDocument interface

DM_Ports method

(IDocument interface)

Syntax

```
Function DM_Ports (Index : Integer) : INetItem;
```

Description

The function returns the indexed port instance from this document. This is to be used in conjunction with the DM_PortCount method

See also

IDocument

DM_Project method

(IDocument interface)

Syntax

```
Function DM_Project : IProject;
```

Description

This function returns the IProject object interface that this document is associated with.

See also

IDocument

DM_RoomCount method

(IDocument interface)

Syntax

```
Function DM_RoomCount : Integer;
```

Description

The function denotes the number of rooms on this document. Use this RoomCount in conjunction with the DM_Rooms(Index) to go through each room object.

See also

IDocument interface

DM_Rooms method

(IDocument interface)

Syntax

```
Function DM_Rooms (Index : Integer) : IRoom;
```

Description

The function returns the indexed room instance from this document. Use this in conjunction with the DM_RoomCount function.

See also

IDocument interface

DM_RuleCount method

(IDocument interface)

Syntax

```
Function DM_RuleCount : Integer;
```

Description

The function returns the number of rules from this document. Use this Rule count in conjunction with the DM_Rules(Index) to go through each sheet symbol object

See also

IDocument interface

DM_Rules method

(IDocument interface)

Syntax

```
Function DM_Rules (Index : Integer) : IRule;
```

Description

The function denotes the indexed rule from this document. Use this DM_RuleCount in conjunction with the DM_Rules to go through each rule found from this document..

See also

IDocument interface

DM_ScrapCompile method

(IDocument interface)

Syntax

```
Function DM_ScrapCompile(ForceCompile : Boolean) : LongBool;
```

Description

The function invokes a scrap compile (by force or not). A scrap compile is the background compile in Altium Designer on a design document and does all the auto - junctions for bus and wire objects. Also the scrap compile does the online rule checks in schematics. It is totally separate from the main compile which compile projects.

See also

IDocument interface

DM_SheetSymbolCount method

(IDocument interface)

Syntax

```
Function DM_SheetSymbolCount : Integer;
```

Description

The function returns the number of sheet symbols from this document. Use this SheetSymbol count in conjunction with the DM_SheetSymbols(Index) to go through each sheet symbol object.

See also

IDocument interface

DM_SheetSymbols method

(IDocument interface)

Syntax

```
Function DM_SheetSymbols (Index : Integer) : ISheetSymbol;
```

Description

The function returns an indexed sheet symbol associated with this document.

See also

IDocument interface

DM_SignalManager method

(IDocument interface)

Syntax

```
Function DM_SignalManager : ISignalManager;
```

Description

The function returns the signal manager interface.

See also

IDocument interface

ISignalManager interface

DM_TextFrameCount method

(IDocument interface)

Syntax

```
Function DM_TextFrameCount : Integer;
```

Description

The function returns the number of text frame objects from this document. Use this TextFrame count in conjunction with the DM_TextFrames(Index) to go through each sheet symbol object

See also

IDocument interface

DM_TextFrames method

(IDocument interface)

Syntax

```
Function DM_TextFrames (Index : Integer) : ITextFrame;
```

Description

The function returns an indexed textframe object associated with this document.

See also

IDocument interface

DM_UniqueComponentCount method

(IDocument interface)

Syntax

```
Function DM_UniqueComponentCount : Integer;
```

Description

The function returns the number of unique components according to the library (ies) they are placed from. A duplicate of components of the same component kind is counted as one (1). Use this in conjunction with the DM_UniqueComponents(Index) method to go through each unique component object.

See also

IDocument

DM_UniqueComponents method

(IDocument interface)

Syntax

```
Function DM_UniqueComponents (Index : Integer) : IComponent;
```

Description

The function returns the indexed unique component instance from this document. This function is to be used in conjunction with the DM_UniqueComponentCount method.

See also

IDocument interface

DM_UniquePartCount method

(IDocument interface)

Syntax

```
Function DM_UniquePartCount : Integer;
```

Description

The function denotes the number of unique parts from this document. Duplicates of the same part kind are only returned as a count of one (1).

See also

IDocument interface

DM_UniqueParts method

(IDocument interface)

Syntax

```
Function DM_UniqueParts (Index : Integer) : IPart;
```

Description

The function returns an indexed unique part associated with this document. Note, if multiple instances of the same part exist, then only one of these parts will be recognized.

See also

IDocument

DM_UpdateDateModified method

(IDocument interface)

Syntax

```
Procedure DM_UpdateDateModified;
```

Description

The procedure sets the modified date for this document.

See also

IDocument interface

DM_VHDLEntities method

(IDocument interface)

Syntax

```
Function DM_VHDLEntities (Index : Integer) : IVHDLEntity;
```

Description

The function returns the indexed VHDL entity instance from this document. Use this in conjunction with the DM_VHDLEntityCount function.

See also

IDocument interface

DM_VHDLEntityCount function

DM_VHDLEntityCount method

(IDocument interface)

Syntax

```
Function DM_VHDLEntityCount : Integer;
```

Description

The function denotes the number of VHDL entities from this document. Use this VHDL Entity count in conjunction with the DM_VHDLEntities(Index) to go through each VHDL entity.

See also

IDocument interface

DM_VHDLEntities method

System Interfaces

IChangeManager interface

Overview

The IChangeManager interface represents the change manager where you can execute an ECO of pins to be swapped for the target component of the target document.

Interface Methods

```
Procedure DM_SetProject1(AProject : IProject);
Procedure DM_SetProject2(AProject : IProject);
Function  DM_ExecuteChanges(IsSilent : LongBool) : LongBool;
Procedure DM_CreateECO_SwapPin        (TargetDocument : IDocument;
                                       TargetComponent: IComponent;
                                       TargetPin      : IPin;
                                       NewPinNumber   : WideString;
                                       OldPinNet      : WideString;
                                       NewPinNet      : WideString);
```

See also

Workspace Manager Interfaces

IDocument interface

IComponent interface

IPin interface

IComponentMappings interface

Overview

The IComponentMappings interface represents the mapping of source components and target components in schematic and PCB documents.

Interface Methods

Method	Description
Function DM_UnmatchedSourceComponent(Index : Integer) : IComponent;	Returns the indexed unmatched source component, that is, a target component could not be found to map to this source component. Use the DM_UnmatchedSourceComponentCount function.
Function DM_UnmatchedTargetComponent(Index : Integer) : IComponent;	Returns the indexed unmatched target component, that is, a source component could not be found to map to the target component. Use the DM_UnmatchedTargetComponentCount function.
Function DM_MatchedSourceComponent (Index : Integer) : IComponent;	Returns the indexed matched source component (that has been matched with a target component). Use the DM_MatchedSourceComponentCount function.
Function DM_MatchedTargetComponent (Index : Integer) : IComponent;	Returns the indexed matched source component (that has been matched with a target component). Use the DM_MatchedTargetComponentCount function.
Function DM_UnmatchedSourceComponentCount : Integer;	Returns the number of unmatched source components.
Function DM_UnmatchedTargetComponentCount : Integer;	Returns the number of unmatched target components.
Function DM_MatchedComponentCount: Integer;	Returns the number of matched components.

ICustomClipboardFormat interface

Overview

Interface Methods

```
Function RegisterCustomClipboardFormat(Const AFormatName : WideString) : Longword;
```

See also

Workspace Manager Interfaces

IDoToManager

Overview

The **IDoToManager** interface represents the To Do panel in Altium Designer. This To Do list manager allows you to manage a list of what to do and assign a priority to each what to do item.

Interface Methods

```
Function  AddItem    (Const AnItem  : WideString) : LongBool;
Function  RemoveItem (Const AnItem  : WideString) : LongBool;
Function  GetItem    (      Index   : Integer   ) : WideString;
```

```
Function  GetCount                : Integer;
Procedure Clear;
```

Interface Properties

```
Property  Item[Index : Integer] : WideString Read GetItem;
Property  Count                 : Integer   Read GetCount;
```

See also

Workspace Manager Interfaces

IDocumentBackups interface

Overview

Interface Properties

```
Property Count : Integer
Property Backups[AIndex : Integer] : WideString
```

See also

IClient interface

IECO interface

Overview

The **IECO** interface represents an Engineering Change Order interface in the Work Space Manager. Basically an Engineering Change Order attempts to keep a project containing source documents and its corresponding primary implementation documents synchronized. For example a schematic project and its PCB document, every time something changes in a schematic project, it is necessary to bring the changes forward to the PCB document via the Engineering Change Order feature.

Interface Methods

Method	Description
Procedure DM_Begin;	Denotes that the ECO manager has started.
Procedure DM_End;	Denotes that the ECO manager has ended.
Function DM_AddObject (Mode : TECO_Mode; ReferenceObject : IDMObject)	Adds a reference object for the ECO to compare the target document against this reference document.
Function DM_RemoveObject (Mode : TECO_Mode; ObjectToRemove : IDMObject)	Removes a reference object depending on what ECO mode is.
Function DM_AddMemberToObject (Mode : TECO_Mode; ReferenceMember : IDMObject; ReferenceParent : IDMObject; TargetParent : IDMObject)	Adds a specific action in the ECO manager.
Function DM_RemoveMemberFromObject (Mode : TECO_Mode; MemberObject : IDMObject; ParentObject : IDMObject)	Removes a specific action in the ECO manager.
Function DM_ChangeObject (Mode : TECO_Mode; Kind : TModificationKind; ObjectToChange : IDMObject;	Changes a specific action in the ECO manager.

ReferenceObject : IDMObject)	

IMessagesManager

Overview

The IMessagesManager interface represents the Messages panel in Altium Designer.

IMessagesManager interface table

IMessagesManager methods	IMessagesManager properties
AddMessage	
AddMessageParametric	
ClearMessages	
ClearMessagesOfClass	
ClearMessagesForDocument	
ClearMessageByIndex	
BeginUpdate	
EndUpdate	
MessagesCount	
Messages	

Example

//Populating the Message Panel using the Workspace manager's functionality

```
Procedure InsertMessagesIntoMessagePanel;
Var
    WSM         : IWorkSpace;
    MM          : IMessagesManager;
    ImageIndex  : Integer;
    F           : Boolean;
Begin
    WSM := GetWorkSpace;
    If WSM = Nil Then Exit;

    // Tick icon for the lines in the Message panel
    // Refer to the Image Index table in the
    // Workspace Manager API reference online help.
    ImageIndex := 3;

    MM := WSM.DM_MessagesManager;
    If MM = Nil Then Exit;

    // Clear out messages from the Message panel...
    MM.DM_ClearMessages;
    MM.DM_ShowMessageView;
    WSM.DM_MessageViewBeginUpdate;

    F := False;
    MM.DM_AddMessage({MessageClass             } 'MessageClass 1',
```

```
                        {MessageText              } 'MessageText 1',
                        {MessageSource            } 'Altium Designer Message',
                        {MessageDocument          } 'Pseudo Doc 1',
                        {MessageCallBackProcess   } '',
                        {MessageCallBackParameters} '',
                        ImageIndex,
                        F);

    MM.DM_AddMessage({MessageClass             } 'MessageClass 2',
                        {MessageText              } 'MessageText 2',
                        {MessageSource            } 'Altium Designer Message 2',
                        {MessageDocument          } 'Pseudo Doc 2',
                        {MessageCallBackProcess   } '',
                        {MessageCallBackParameters} '',
                        ImageIndex,
                        F);

    MM.DM_MessageEndUpdate;
End;
```

See also

Image Index Table

Methods

AddMessage method

(IMessagesManager interface)

Syntax

```
Procedure AddMessage
(Const MessageClass,
       MessageText,
       MessageSource,
       MessageDocument,
       MessageCallBackProcess,
       MessageCallBackParameters     : WideString;
       ImageIndex                    : Integer;
       ReplaceLastMessageIfSameClass : Boolean = False;
       MessageCallBackProcess2       : WideString = '';
       MessageCallBackParameters2    : WideString = '');
```

Description

This method gives you the ability to a Altium Designer Message on the Message panel.

MessageClass :- which sort of message it belongs to. (User defined)

MessageText :- the message text to appears in the Message panel.

MessageSource :- could be one of the following pre-defined strings such as : Comparator, Back-Annotate, Output Generator, Compiler or you can define your own MessageSource string.

MessageDocument :- Owner Document name – normally a full path name of the document that the Message is associated with.

MessageCallBackProcess :- process name to call back.

MessageCallbackParameters :- parameters for the CallBackProcess.

ImageIndex :- the index to the image depending on which Message Class. Refer to the Image Index Table topic to check out the appropriate image for each message.

ReplaceLastMessageIfSameClass :- (defaults to false).

MessageCallBackProcess2

MessageCallBackParameters2

Example

See also

IMessagesManager Interfaces

AddMessageParametric method

(IMessagesManager interface)

Syntax

```
Procedure AddMessageParametric(MessageParams : PChar;MessageCallBackParameters : PChar);
```

Description

Inserts a Altium Designer message in the Message panel. Similar to the DM_AddMessage only that you define the Name / Value blocks in the MessageParams nullterminated string.

Class:– Back-Annotate class, Error level, Differences.

Text:-text displayed in the Message panel.

Source:- could be one of the following: Comparator, Back-Annotate, Output Generator, Compiler,.

Document:- Owner Document name

CallBackProcess:- process name to call back.

UserId:- Unique ID

HelpFileName:- Name of the Help file

HelpTopic:- specific help topic string

ImageIndex::- the index to the image depending on which Message Class.

'ReplaceLastMessageIfSameClass':- Boolean. If Same MessageClass, specify whether this class is to be overridden or not by the current message class information.

The **MessageCallBackParameters** parameter :– parameters for the CallBackProcess.

Example

See also

IMessagesManager

BeginUpdate method

(IMessagesManager interface)

Syntax

```
Procedure BeginUpdate;
```

Description

Invoke this method before you wish to add Messages (DM_AddMessage or DM_AddMessageParameteric methods) to the Message panel.

Example

See also

IMessagesManager

ClearMessageByIndex method

(IMessagesManager interface)

Syntax

```
Procedure ClearMessageByIndex ( AIndex : Integer );
```

Description

Example

See also

IMessagesManager

ClearMessages method

(IMessagesManager interface)

Syntax

```
Procedure ClearMessages;
```

Description

Clears out the Messages panel.

Example

See also

IMessagesManager

ClearMessagesForDocument method

(IMessagesManager interface)

Syntax

```
Procedure ClearMessagesForDocument(Const DocumentPath : WideString);
```

Description

Example

See also

IMessagesManager

ClearMessagesOfClass method

(IMessagesManager interface)

Syntax

```
Procedure ClearMessagesOfClass (Const AMsgClass : WideString);
```

Description

This method gives you the ability to clear messages of the same class type. Various class types include Back-Annotate class, Error level, Differences

Example

See also

IMessagesManager

EndUpdate method

(IMessagesManager interface)

Syntax

```
Procedure EndUpdate;
```

Description

Invoke this method after you have added Messages to the Message panel.

Example

See also

IMessagesManager

Messages method

(IMessagesManager interface)

Syntax

```
Function Messages(Index : Integer) : IMessageItem;
```

Description

Example

See also

IMessagesManager

MessagesCount method

(IMessagesManager interface)

Syntax

```
Function MessagesCount : Integer;
```

Description

Example

See also

IMessagesManager

IMessageItem interface

Overview

The Message Manager interface has messages if any on the messages panel. Each message item is represented by the IMessageItem interface.

IMessageItem Properties

```
Property MsgClass            : WideString
Property Text                : WideString
Property Source              : WideString
Property Document            : WideString
Property MsgDateTime         : TDateTime
Property ImageIndex          : Integer
Property UserId              : WideString
Property CallBackProcess     : WideString
Property CallBackParameters  : WideString
Property CallBackProcess2    : WideString
Property CallBackParameters2 : WideString
Property HelpFileName        : WideString
Property HelpFileID          : WideString
Property MsgIndex            : Integer
```

See also

IMessagesManager Interfaces

ISearchPath interface

Overview

The ISearchPath interface represents the paths of a project. This ISearchPath interface has a link to the associated open project in Altium Designer.

Interface Methods

Method	Description
Function DM_Path : WideString;	Returns the path of the focussed project in Altium Designer.
Function DM_AbsolutePath : WideString;	Returns the absolute path of the focussed project in Altium Designer.
Function DM_IncludeSubFolders : Boolean;	Returns whether sub folders are included in the focussed project in Altium Designer.
Function DM_Project : IProject;	Returns the project in which this ISearchPath interface is associated with.

ISymbolGenerator

Overview

The ISymbolGenerator interface represents the symbol with parameters added if necessary generated by the ICoreProject interface.

Important Notes

- ICoreProject interface's DM_CreateSymbolGenerator method returns a ISymbolGenerator interface.

Interface Methods

```
Procedure DM_ClearParameters;
Procedure DM_AddParameter(Name, Value : WideString);
Procedure DM_GenerateComponent;
```

See also

Workspace Manager Interfaces

ICoreProject interface

IVCSProjectAccessor interface

Overview

Interface methods

```
Function ObjectAddress : IInterface;
```

See also

IClient interface

IExternalForm interface

IVersionControlServer interface

Overview

Interface methods

```
Function GetStatusString (Const AObject : IDMObject) : WideString;
```

See also

Workspace Manager Interfaces

IVhdlEntity interface

Overview

The IVhdlEntity interface represents the existing VHDL entity object on a VHDL document. Basically a VHDL document can contain many VHDL entities and each entity corresponds to a schematic document.

Since every object interface (inherited from the IDMObject interface) has a DM_VHDLEntity method. This method can be useful in cases such as determining which ports correspond to VHDL entities.

Interface Methods

Method	Description
Function DM_Name : WideString;	Returns the name of the VHDL entity.

IWorkspacePreferences

Overview

The **IWorkspacePreferences** interface represents the Preferences object in Altium Designer This interface details with file ownership - that is the rights of access to a design document.

Interface Methods

```
Function  GetDefaultTemplateFile(Const ADocKind : Widestring) : Widestring;
Function  GetFileOwnership_Enabled : Boolean;
Function  GetFileOwnership_EnabledOutputDirectory : Boolean;
Function  GetFileOwnership_WarningLevelOpen : TFileOwnershipWarningLevel;
Function  GetFileOwnership_WarningLevelSave : TFileOwnershipWarningLevel;
Function  GetDefaultLibraryPath : WideString;
Function  GetHighlightMethodSet : THighlightMethodSet;
Function  GetObjectsToDisplay   : TWorkspaceObjectIdSet;
Function  GetHighlightConnectedPowerParts : Boolean;

Procedure SetDefaultTemplateFile(Const ADocKind, AFileName : Widestring);
Procedure SetFileOwnership_Enabled(AValue : Boolean);
Procedure SetFileOwnership_EnabledOutputDirectory(AValue : Boolean);
Procedure SetFileOwnership_WarningLevelOpen(AValue : TFileOwnershipWarningLevel);
Procedure SetFileOwnership_WarningLevelSave(AValue : TFileOwnershipWarningLevel);
```

Interface Properties

```
Property  DefaultTemplateFile[Const ADocKind   : Widestring] : Widestring
Property  FileOwnership_Enabled                : Boolean
Property  FileOwnership_EnabledOutputDirectory : Boolean
Property  FileOwnership_WarningLevelOpen       : TFileOwnershipWarningLevel
Property  FileOwnership_WarningLevelSave       : TFileOwnershipWarningLevel
Property  DefaultLibraryPath                   : WideString
```

See also

Workspace Manager Interfaces

Project Interfaces

IProject Interface

Overview

The IProject interface deals with an open project in Altium Designer. There are project and document variants, that is actually a project or document can be specified to have project or document variants (actual project / document variants do not exist) and on these document variants have component variants.

To have access to the data of a project, you need to do a compile first. Projects deal with logical and physical documents. Logical documents are the connected documents which are part of a design which include a PCB document associated with this design. Physical documents are source documents expanded by the Altium Designer compiler as in a flattened design project.

Thus, a project contains source documents and implementation documents. To have access to the most current data of a project, you need to compile the project first. The compiler maps (or expands) all the logical source documents into physical documents.

Normally there is a one logical document to a one physical document for a simple flat design project, but for hierarchical design projects (for example multi channel projects), the documents that have sheet symbols with a Repeat statement, then logical documents are expanded into multiple physical documents.

There are Output jobs consisting of available output generators installed in Altium Designer.

The **IProject** interface hierarchy is as follows;

IProject methods

DM_AddConfiguration
DM_AddConfigurationParameters
DM_AddConfigurationParameters_Physical
DM_AddControlPanel
DM_AddGeneratedDocument
DM_AddSearchPath
DM_AddSourceDocument
DM_ChannelDesignatorFormat
DM_ChannelRoomLevelSeperator
DM_ChannelRoomNamingStyle
DM_ClearViolations
DM_Compile
DM_CompileEx
DM_ComponentMappings
DM_ConfigurationCount
DM_Configurations
DM_CurrentProjectVariant
DM_DoCrossSelection SafeCall
DM_DocumentFlattened
DM_EditOptions
DM_ErrorLevels
DM_GeneratedDocumentCount
DM_GeneratedDocuments
DM_GetAllowPortNetNames
DM_GetAllowSheetEntryNetNames
DM_GetAppendSheetNumberToLocalNets
DM_GetConfigurationByName
DM_GetDefaultConfiguration
DM_GetDefaultConfigurationName
DM_GetDefaultPcbType

IProject properties

DM_GetDocumentFromPath
DM_GetOutputPath
DM_GetPinSwapBy_Pin
DM_GetPinSwapByNetlabel
DM_GetScrapDocument
DM_HierarchyMode
DM_HierarchyModeForCompile
DM_IndexOfSourceDocument
DM_InitializeOutputPath
DM_LogicalDocumentCount
DM_LogicalDocuments
DM_MoveSourceDocument
DM_NavigationZoomPrecision
DM_OptionsStorage
DM_Outputers
DM_OwnedProjectCount
DM_OwnedProjects
DM_PhysicalDocumentCount
DM_PhysicalDocuments
DM_PrimaryImplementationDocument
DM_ProjectFileName
DM_ProjectFullPath
DM_ProjectVariantCount
DM_ProjectVariants
DM_RemoveAllConfigurations
DM_RemoveConfigurationByName
DM_RemoveSourceDocument
DM_SearchPathCount
DM_SearchPaths
DM_SetAllowPortNetNames
DM_SetAllowSheetEntryNetNames
DM_SetAppendSheetNumberToLocalNets
DM_SetAsCurrentProject
DM_SetDefaultConfigurationName
DM_SetDefaultPcbType
DM_SetErrorLevels
DM_SetHierarchyMode
DM_SetOutputPath
DM_SetPinSwapBy_Netlabel
DM_SetPinSwapBy_Pin
DM_StartCrossProbing
DM_StartNavigation
DM_ToDoManager
DM_TopLevelLogicalDocument
DM_TopLevelPhysicalDocument
DM_UpdateConstraints

DM_UserID

DM_ViolationCount

DM_Violations

GetNavigationHistory

See also

Methods

DM_AddConfigurationParameters method

(IProject interface)

Syntax

```
Procedure DM_AddConfigurationParameters(Configuration : WideString);
```

Description

A configuration is a list of constraints file which manages the mapping of pins to ports of a FPGA project. Invoke this method to add parameters of a specified configuration file for a FPGA project.

See also

IProject interface

DM_AddConfigurationParameters_Physical method

(IProject interface)

Syntax

```
Procedure DM_AddConfigurationParameters_Physical(Configuration : WideString);
```

Description

A configuration is a list of constraints file which manages the mapping of pins to ports of a FPGA project. Invoke this method to add parameters of a specified configuration file for a FPGA project.

See also

IProject interface

DM_AddControlPanel method

(IProject interface)

Syntax

```
Procedure DM_AddControlPanel (Filename : WideString);
```

Description

The procedure adds a document to the main section of the the panel which could be part of a project or free documents.

See also

IProject interface

DM_AddGeneratedDocument method

(IProject interface)

Syntax

```
Procedure DM_AddGeneratedDocument (Filename : WideString);
```

Description

This procedure adds a new generated document referenced by its filename parameter in this current project, and this document appears in the **Generated** folder of this project on Altium Designer Projects panel.

See also

IProject interface

DM_AddSearchPath method

(IProject interface)

Syntax

```
Procedure DM_AddSearchPath (SearchPath : WideString; IncludeSubFolders : Boolean);
```

Description

This procedure adds a new serach path for the current project.

See also

IProject interface

DM_AddSourceDocument method

(IProject interface)

Syntax

```
Procedure DM_AddSourceDocument (Filename : WideString);
```

Description

The procedure adds a source document referenced by its filename parameter in the current project.

See also

IProject interface

DM_ChannelDesignatorFormat method

(IProject interface)

Syntax

```
Function DM_ChannelDesignatorFormat : WideString;
```

Description

This function returns the formatted channel designator string. This string is basedon the settings defined in the Multi-Channel page of the Options for Project dialog from the Project » Project Options menu item.

See also

IProject interface

DM_ChannelRoomLevelSeperator method

(IProject interface)

Syntax

```
Function DM_ChannelRoomLevelSeperator : WideString;
```

Description

The function returns the separator character for the Channel Room Level string. The default is an underline character used for room naming styles when there are paths (based on hierarchical designs).

See also

IProject interface

DM_ChannelRoomNamingStyle method

(IProject interface)

Syntax

```
Function DM_ChannelRoomNamingStyle : TChannelRoomNamingStyle;
```

Description

The function returns the TChannelRoomNamingStyle type. There are alternative styles for naming rooms on a PCB document.

See also

IProject interface

DM_ClearViolations method

(IProject interface)

Syntax

```
Procedure DM_ClearViolations;
```

Description

The procedure clears all existing violations within the project.

See also

IProject interface

DM_Compile method

(IProject interface)

Syntax

```
Function DM_Compile : LongBool;
```

Description

Invoke this function to compile the current project. Once the project is compiled, navigation of nets and comparing the differences of documents and other tasks can be performed.

See also

IProject interface

DM_CompileEx method

(IProject interface)

Syntax

```
Function DM_CompileEx(All : LongBool; Var Cancelled : LongBool) : LongBool;
```

Description

Invoke this function to compile all documents of all opened projects in Altium Designer. Pass a Boolean parmeter in to cancel the compiling process.

See also

IProject interface

DM_ComponentMappings method

(IProject interface)

Syntax

```
Function DM_ComponentMappings (AnImplementationDocument : WideString) : IComponentMappings;
```

Description

The function returns the IComponentMapping interface which details which PCB components are linked to Schematic components. Check the IComponentMappings interface.

See also

IProject interface

DM_ConfigurationCount method

(IProject interface)

Syntax

```
Function DM_ConfigurationCount : Integer;
```

Description

The function returns the number of configurations for the current project. To be used in conjunction with DM_Configurations function.

Example

See also

IProject interface

DM_Configurations method

(IProject interface)

Syntax

```
Function DM_Configurations (Index : Integer ) : IConfiguration;
```

Description

The function returns the indexed configuration of a FPGA project. A configuration can have a list of different constraint files.

See also

IProject interface

DM_CurrentProjectVariant method

(IProject interface)

Syntax

```
Function DM_CurrentProjectVariant : IProjectVariant;
```

Description

The function returns the current project variant from this current project. Check out the IProjectVariant interface.

See also

IProject interface

DM_DoCrossSelection method

(IProject interface)

Syntax

```
Procedure DM_DoCrossSelection
```

Description

Activates the cross probing function where you can jump from a Schematic object to its corresponding PCB object (both source and primary implementation documents need to be open in Altium Designer).

See also

IProject interface

DM_DocumentFlattened method

(IProject interface)

Syntax

```
Function DM_DocumentFlattened : IDocument;
```

Description

The function returns the flattened document. A flattened document is part of a flattened hierarchy of a project and all objects of this project appear in the Instance list of the Navigator panel.

See also

IProject interface

DM_EditOptions method

(IProject interface)

Syntax

```
Function DM_EditOptions(DefaultPage : WideString) : LongBool;
```

Description

Example

See also

IProject interface

DM_ErrorLevels method

(IProject interface)

Syntax

```
Function DM_ErrorLevels (AErrorKind : TErrorKind) : TErrorLevel;
```

Description

The function returns the error level for the specified error type. For each violation type, you can have up to four different error levels, No Report, Warning, Error and Fatal Error with four different colored folders.

See also

IProject interface

DM_GeneratedDocumentCount method

(IProject interface)

Syntax

```
Function DM_GeneratedDocumentCount : Integer;
```

Description

The function returns the number of generated documents such as those documents generated by the OutPut generator (from a OutJob document). Use this function in conjunction with the DM_GeneratedDocuments function.

Example

See also

IProject interface

DM_GeneratedDocuments method

(IProject interface)

Syntax

```
Function DM_GeneratedDocuments (Index : Integer ) : IDocument;
```

Description

The function returns the indexed generated document which is generated by the Output Generator.

See also

IProject interface

DM_GetAllowPortNetNames method

(IProject interface)

Syntax

```
Function DM_GetAllowPortNetNames : Boolean;
```

Description

Invoke this function to check whether port net names are used for navigation in Altium Designer or not.

See also

IProject interface

DM_GetAllowSheetEntryNetNames method

(IProject interface)

Syntax

```
Function DM_GetAllowSheetEntryNetNames : Boolean;
```

Description

Invoke this function to check whether sheet entry net anmes are used for navigation in Altium Designer or not.

See also

IProject interface

DM_GetAppendSheetNumberToLocalNets method

(IProject interface)

Syntax

```
Function DM_GetAppendSheetNumberToLocalNets : Boolean;
```

Description

Invoke this function to check whether sheet numbers are appended to local nets or not.

See also

IProject interface

DM_GetConfigurationByName method

(IProject interface)

Syntax

```
Function DM_GetConfigurationByName(Configuration : WideString) : IConfiguration;
```

Description

The function returns you the configuration object for the project (normally for FPGA projects) if configuration parameter is valid. A configuration file contains mapping information to link from a FPGA project to a linked PCB project.

See also

IProject interface

DM_GetDefaultConfiguration method

(IProject interface)

Syntax

```
Function DM_GetDefaultConfiguration : IConfiguration;
```

Description

The function returns the default configuration for a FPGA project.

See also

IProject interface

DM_GetDefaultConfigurationName method

(IProject interface)

Syntax

```
Function DM_GetDefaultConfigurationName : WideString;
```

Description

Returns the name of the default configuration for a FPGA project

See also

IProject interface

DM_GetDefaultPcbType method

(IProject interface)

Syntax

```
Function DM_GetDefaultPcbType : WideString;
```

Description

Example

See also

IProject interface

DM_GetDocumentFromPath method

(IProject interface)

Syntax

```
Function DM_GetDocumentFromPath(DocumentPath : WideString) : IDocument;
```

Description

This function returns the IDocument interface associated with the document path parameter. Otherwise a Nil value is returned.

See also

IProject interface

DM_GetOutputPath method

(IProject interface)

Syntax

```
Function DM_GetOutputPath : WideString;
```

Description

The function returns the output path for generated documents for the current project.

See also

IProject interface

DM_GetScrapDocument method

(IProject interface)

Syntax

```
Function DM_GetScrapDocument(DocumentPath : WideString) : IDocument;
```

Description

Returns the scrap document for the project. A scrap document is a temporary document used when creating a new document and once a document is saved, the contents of the scrap document is copied and freed.

See also

IProject interface

DM_HierarchyMode method

(IProject interface)

Syntax

```
Function DM_HierarchyMode : TFlattenMode;
```

Description

This function returns the hierarchy mode as a TFlattenMode parameter.

See also

IProject interface

DM_HierarchyModeForCompile method

(IProject interface)

Syntax

```
Function DM_HierarchyModeForCompile : TFlattenMode;
```

Description

Example

See also

IProject interface

DM_IndexOfSourceDocument method

(IProject interface)

Syntax

```
Function DM_IndexOfSourceDocument(Filename : WideString) : Integer;
```

Description

The function returns the index of the source document based on the filename of this document. This is for hierarchical or connected schematic documents.

See also

IProject interface

DM_InitializeOutputPath method

(IProject interface)

Syntax

```
Function DM_InitializeOutputPath(AnOutputType : WideString) : WideString;
```

Description

The function returns the output path for the Output Generator based on the AnOutputType parameter.

See also

IProject interface

DM_LogicalDocumentCount method

(IProject interface)

Syntax

```
Function DM_LogicalDocumentCount : Integer;
```

Description

The function returns the number of logical documents which represent the actual documents of a design project (documents that exist in the design project but are not part of the design are not logical documents). Use this function in conjunction with the DM_LogicalDocuments function.

See also

IProject interface

DM_LogicalDocuments method

(IProject interface)

Syntax

```
Function DM_LogicalDocuments (Index : Integer ) : IDocument;
```

Description

The function returns the indexed logical document of a project.

See also

IProject interface

DM_MoveSourceDocument method

(IProject interface)

Syntax

```
Procedure DM_MoveSourceDocument (Filename : WideString; NewIndex : Integer);
```

Description

The procedure re-assigns the source document referenced by the filename a new index number.

See also

IProject interface

DM_NavigationZoomPrecision method

(IProject interface)

Syntax

```
Function DM_NavigationZoomPrecision : Integer;
```

Description

Sets how precise the document zoom is when the interactive navigator is being used to trace the connection in a project.

See also

IProject interface

DM_OptionsStorage method

(IProject interface)

Syntax

```
Function DM_OptionsStorage : IOptionsStorage;
```

Description

Example

See also

IProject interface

DM_Outputers method

(IProject interface)

Syntax

```
Function DM_Outputers (Name : WideString) : IOutputer;
```

Description

The function returns the indexed Output Generator. An output generator could be a Simple BOM.

See also

IProject interface

DM_PhysicalDocumentCount method

(IProject interface)

Syntax

```
Function DM_PhysicalDocumentCount : Integer;
```

Description

The function returns the number of physical source documents (which are expanded logical documents of the design project). Source documents are usually schematic documents. Use this function in conjunction with the DM_PhysicalDocuments function.

See also

IProject interface

DM_PhysicalDocuments method

(IProject interface)

Syntax

```
Function DM_PhysicalDocuments (Index : Integer ) : IDocument;
```

Description

The function returns the indexed physical document of a project.

See also

IProject interface

DM_PrimaryImplementationDocument method

(IProject interface)

Syntax

```
Function DM_PrimaryImplementationDocument : IDocument;
```

Description

The function returns the primary implementation document for example PCB documents. Source documents are Schematic documents for example.

See also

IProject interface

DM_ProjectFileName method

(IProject interface)

Syntax

```
Function DM_ProjectFileName : WideString;
```

Description

This function returns the file name of this current project in Altium Designer.

See also

IProject interface

DM_ProjectFullPath method

(IProject interface)

Syntax

```
Function DM_ProjectFullPath : WideString;
```

Description

This function returns the full path of this current project in Altium Designer.

See also

IProject interface

DM_ProjectVariantCount method

(IProject interface)

Syntax

```
Function DM_ProjectVariantCount : Integer;
```

Description

The function returns the number of project variants for this current project.

See also

IProject interface

DM_ProjectVariants method

(IProject interface)

Syntax

```
Function DM_ProjectVariants (Index : Integer ) : IProjectVariant;
```

Description

The function returns the indexed IProjectVariant interface. A project variant interface is only a conceptual representation of a project that can have project variants. That is there is only one physical board but this same board can have certain components disabled or enabled leading to document variants. The variations of a PCB board are referred to as the IDocumentVariant and to check which components are enabled or not for this particular document variant, check out the IComponentVariant interface.

This is to be used in conjunction with the DM_ProjectVariantCount method.

See also

IProject interface

DM_RemoveSourceDocument method

(IProject interface)

Syntax

```
Procedure DM_RemoveSourceDocument (Filename : WideString);
```

Description

This procedure removes a source document referenced by its filename from this current project.

See also

IProject interface

DM_SearchPathCount method

(IProject interface)

Syntax

```
Function DM_SearchPathCount : Integer;
```

Description

The function returns the number of search paths for this current project. Use this function in conjunction with the DM_SearchPaths function.

See also

IProject interface

DM_SearchPaths method

(IProject interface)

Syntax

```
Function DM_SearchPaths (Index : Integer ) : ISearchPath;
```

Description

The function returns the indexed search path object defined for this project.

See also

IProject interface

DM_SetAllowPortNetNames method

(IProject interface)

Syntax

```
Procedure DM_SetAllowPortNetNames (AAllow : Boolean);
```

Description

Invoke this procedure to allow port net names be used for navigation.

See also

IProject interface

DM_SetAllowSheetEntryNetNames method

(IProject interface)

Syntax

```
Procedure DM_SetAllowSheetEntryNetNames (AAllow : Boolean);
```

Description

Invoke this procedure to allow sheet entry net names be used for navigation in Altium Designer.

See also

IProject interface

DM_SetAppendSheetNumberToLocalNets method

(IProject interface)

Syntax

```
Procedure DM_SetAppendSheetNumberToLocalNets (AAppend : Boolean);
```

Description

Invoke this procedure to have the ability to append sheet numbers to local nets on a document / project.

See also

IProject interface

DM_SetAsCurrentProject method

(IProject interface)

Syntax

```
Procedure DM_SetAsCurrentProject;
```

Description

Invoke this function to set the project as the current project in Altium Designer.

See also

IProject interface

DM_SetDefaultConfigurationName method

(IProject interface)

Syntax

```
Procedure DM_SetDefaultConfigurationName(Configuration : WideString);
```

Description

The procedure sets the name for the default configuration of a FPGA project.

See also

IProject interface

DM_SetDefaultPcbType method

(IProject interface)

Syntax

```
Procedure DM_SetDefaultPcbType(PcbType : WideString);
```

Description

Example

See also

IProject interface

DM_SetErrorLevels method

(IProject interface)

Syntax

```
Procedure DM_SetErrorLevels(AErrorKind : TErrorKind;AErrorLevel : TErrorLevel);
```

Description

Example

See also

IProject interface

DM_SetHierarchyMode method

(IProject interface)

Syntax

```
Procedure DM_SetHierarchyMode (AFlatten : TFlattenMode);
```

Description

Invoke this function to set which hierarchy mode for this project. It can be one of the following modes: eFlatten_Smart,eFlatten_Flat,eFlatten_Hierarchical,eFlatten_Global

See also

IProject interface

DM_SetOutputPath method

(IProject interface)

Syntax

```
Procedure DM_SetOutputPath (AnOutputPath : WideString);
```

Description

Sets the output path for generated documents to go in by the Altium Designer output generator.

See also

IProject interface

DM_StartCrossProbing method

(IProject interface)

Syntax

```
Procedure DM_StartCrossProbing(CtrlDoesSwitch : Boolean);
```

Description

This procedure invokes the cross probing function. Both source and primary implementation documents need to be open in Altium Designer in order for the cross probing to work.

See also

IProject interface

DM_StartNavigation method

(IProject interface)

Syntax

```
Procedure DM_StartNavigation;
```

Description

This procedure invokes the navigation panel for the current project. The project needs to be compiled first.

See also

IProject interface

DM_ToDoManager method

(IProject interface)

Syntax

```
Function DM_ToDoManager : IToDoManager;
```

Description

Invoke this function to have access to the IToDoManager object. This ToDo manager allows you to define to dos for your current project.

See also

IProject interface

DM_TopLevelLogicalDocument method

(IProject interface)

Syntax

```
Function DM_TopLevelLogicalDocument : IDocument;
```

Description

This function returns the top level logical document of this current project. A logical document is usually a Schematic document and can represent a document of a multi channel project for example.

See also

IProject interface

DM_TopLevelPhysicalDocument method

(IProject interface)

Syntax

```
Function DM_TopLevelPhysicalDocument : IDocument;
```

Description

This function returns the top level physical document of this current project. A physical document usually is a PCB document.

See also

IProject interface

DM_UpdateConstraints method

(IProject interface)

Syntax

```
Function DM_UpdateConstraints : LongBool;
```

Description

Invoke this function to update the constraint files used for a FPGA project and for corresponding PCB projects with FPGA components.

See also

IProject interface

DM_UserID method

(IProject interface)

Syntax

```
Function DM_UserID : WideString;
```

Description

The function returns a value that represents the UserID of the project.

See also

IProject interface

DM_ViolationCount method

(IProject interface)

Syntax

```
Function DM_ViolationCount : Integer;
```

Description

This function returns the number of violations reported by Altium Designer for this current project.

See also

IProject interface

DM_Violations method

(IProject interface)

Syntax

```
Function DM_Violations(Index : Integer) : IViolation;
```

Description

Returns the indexed violation for a current project. This is to be used in conjunction with the DM_ViolationCount method.

See also

IProject interface

GetNavigationHistory method

(IProject interface)

Syntax

```
Function GetNavigationHistory : INavigationHistory;
```

Description

This function returns the status of the navigation buttons on the Navigator panel for the current project in Altium Designer. Check out INavigationHistory interface for details.

See also

IProject interface

IAbstractVHDLProject

Overview

The IAbstractVHDLProject interface represents a project that hosts VHDL documents.

Important notes

- Inherited from IProject interface

Interface Methods

```
Function  DM_GetTargetDeviceName(ConfigurationName : WideString) : WideString;
```

See also

```
Workspace Manager Interfaces
IProject interface
```

IBoardProject

Overview

The IBoardProject interface represents a project compromising of Schematic and corresponding PCB documents along with other document kinds.

Important notes

- Inherited from IProject interface

Interface Methods

IProject methods

Interface Properties

IProject Properties

See also

```
Workspace Manager Interfaces
IProject interface
```

ICoreProject

Overview

The ICoreProject interface represents the project that hosts core designs. A core project is typically created to develop pre-synthesized user models whose EDIF output becomes the model for these user defined components.

Important notes

- Inherited from IAbstractVHDLProject interface

Interface Methods

```
Function DM_CreateSymbolGenerator     : ISymbolGenerator;
Function DM_GetIncludeModelsInArchive : LongBool;
```

See also

```
Workspace Manager Interfaces
IProject interface
```

IAbstractVHDLProject interface

ISymbolGenerator interface

IEmbeddedProject

Overview

The IEmbeddedProject interface represents the project that hosts embedded designs that can be targetted to the hard device on the Nanoboard.

Important notes

- The IEmbeddedProject interface is inherited from IProject interface.

Interface Methods

DM_SetToolchain method.

See also

```
Workspace Manager Interfaces
IProject interface
```

IFPGAProject

Overview

The IFPGAProject interface represents the project that hosts FPGA designs.

Important notes

- Inherited from IAbstractVHDLProject interface

Interface Methods

```
Function  DM_GetTargetBoardName (ConfigurationName : WideString) : WideString;
```

See also

```
Workspace Manager Interfaces
IProject interface
```

IAbstractVHDLProject Interface

IIntegratedLibraryProject interface

Overview

The IIntegratedLibraryProject interface represents the project that deals with integrated libraries.

IImportant notes

- Inherited from IProject interface

Interface Methods

- IProject methods

Interface Properties

- IProject Properties

See also

```
Workspace Manager Interfaces
IProject interface
```

Project Variations

IComponentVariation interface

Overview

The IComponentVariation interface represents the component variant on a PCB document. There is only one physical document, but each component on this document can be specified to be a variant and when the output is generated, a specific variant document is generated. This variant output is controlled by the Output Job files.

Interface Methods

Method	Description
Function DM_ProjectVariant : IDocumentVariant;	This function returns the IProjectVariant interface which represents a container that stores the component variants for the project.
Function DM_VariationKind : TVariationKind;	This function returns the variation kind for this component.
Function DM_PhysicalDesignator : WideString;	Returns the full physical designator string for this component variant.
Function DM_UniqueId : WideString;	Returns the unique ID for this component variant.
Function DM_AlternatePart : WideString;	Returns the alternate part string for this component variant.
Function DM_VariationCount : Integer;	Returns the number of variations.
Function DM_Variations (Index : Integer) : IParameterVariation;	Returns the indexed parameter variation for this component variation.

See also

IProjectVariant interface

IParameterVariation interface

IProjectVariant interface

Overview

The IProjectVariation interface represents the project that contains component variations. Physically, there is only one PCB document with components that are specified. So for each output requirement, each document variant is generated, although there is only one PCB design document.

Interface Methods

Method	Description
Function DM_Project : IProject;	Returns the IProject interface this variant is associated with.
Function DM_Name : WideString;	Returns the name of this variant.
Function DM_Description : WideString;	Returns the description of this variant.
Function DM_VariationCount : Integer;	Returns the count of variants. To be used in conjunction with the DM_Variations(index) method.
Function DM_Variations (Index : Integer) : IComponentVariation;	Returns the indexed component variation for this project. To be used in conjunction with the DM_VariationCount method.

See also

Workspace Manager Interfaces

IProject interface

IParameterVariation interface

Overview

The IParameterVariation interface represents the component that contains parameter variations. Physically, there is only one PCB document with components that are specified. So for each output requirement, each document variant is generated, although there is only one PCB design document.

Interface Methods

Method	Description
Function DM_ParameterName : WideString;	Denotes the name of the parameter that the component is associated with.
Function DM_VariedValue : WideString;	Denotes the value of the parameter that the component is associated with. A component variant can have parameter variants.

See also

Workspace Manager Interfaces

IProject interface

IProjectVariant interface

IComponentVariation interface

Configuration Constraints Interfaces

IConfiguration interface

Overview

The IConfiguration interface represents the configuration container that contains a group of constraints which targets a specific FPGA device.

Interface Methods

```
Function     DM_Name                              : WideString;
```

```
Function    DM_ConstraintGroupCount              : Integer;
Function    DM_ConstraintGroups(Index : Integer) : IConstraintGroup;
Function    DM_ConstraintsFileCount                 : Integer;
Function    DM_ConstraintsFilePath(Index : Integer) : WideString;
Procedure   DM_AddConstraintFile(AConstraintFilePath : WideString);
Function    DM_GetTargetDeviceName                  : WideString;
```

See also

Workspace Manager Interfaces

IConstraintGroup interface

Overview

The IConstraintGroup interface represents a constraint file made up of constraints (as IConstraint interface).

Important notes

- Inherited from IDMObject interface

Interface Methods

```
Function   DM_TargetKindString             : WideString;
Function   DM_TargetId                     : WideString;
Function   DM_ConstraintCount              : Integer;
Function   DM_Constraints(Index : Integer) : IConstraint;
```

See also

Workspace Manager Interfaces

IConstraint interface

IConstraint interface

Overview

The IConstraint interface represents the data entry in a constraint file represented by the IConstraintGroup interface.

Important notes

- Inherited from IDMObject interface

Interface Methods

```
Function   DM_Kind : WideString;
Function   DM_Data : WideString;
```

See also

Workspace Manager Interfaces

IConstraintGroup interface

IInstalledConstraintFiles interface

Overview

The IInstalledConstraintFiles interface represents the constraint files that are installed in Altium Designer, ie available to a FPGA project.

Interface Methods

```
Function  InstalledConstraintFileCount                     : Integer;
Function  InstalledConstraintFile   (aIndex   : Integer)   : WideString;
Function  ConstraintFileIsInstalled (aPath    : WideString) : LongBool;
Function  DefaultConstraintFile                            : WideString;
Function  EditInstalledConstraintFiles                     : LongBool;
```

See also

Workspace Manager Interfaces

IOutputer interface

Overview

The IOutputer interface represents the one of the outputs of an output job within a design project.

Interface Methods

```
Function    DM_ViewName                   : WideString
Function    DM_EditProperties             : Boolean;
Function    DM_Generate_OutputFilesTo (OutputDirectory : WideString; ParameterOverrides :
PChar) : Boolean;
Function    DM_Generate_OutputFiles   (AGeneratedFilename : PChar) : Boolean;
Procedure   DM_SetPrintScale          (APrintScale        : Double);
Procedure   DM_SetPrintMode           (AFitPrintToPage    : Boolean);
Procedure   DM_SetDocumentPath        (ADocPath           : WideString);
```

See also

Workspace Manager Interfaces

IProject interface

IOutputJob interface

IWSM_OutputJobDocument interface

IStrings interface

Overview

The IStrings interface represents the strings container – more like a list of strings.

Interface Methods

```
Function GetCount : Integer;
Function GetItem(Index : Integer) : WideString;
Function IndexOf(Const Value : WideString) : Integer;
```

Interface Properties

```
Property Count : Integer read GetCount;
Property Items[Index : Integer] : WideString read GetItem; default;
```

See also

Workspace Manager Interfaces

IWSM_OutputJobDocument interface

Overview

The IWSM_OutputJobDocument interface represents the output jobs document in Altium Designer.

Interface Methods

```
Function  GetState_Outputer      (AIndex : Integer) : IOutputer;
Function  GetState_OutputerCount                    : Integer;

Function  CreateOutputer      (Const AOutputCategoryName   : WideString;
                               Const APredefinedOutputName : WideString;
                               Const AOutputerName         : WideString) : IOutputer;
Function  BeginModifyOutputer (Const AOutputer : IOutputer ) : Boolean;
Procedure EndModifyOutputer;
```

Interface Properties

```
Property  Outputer [AIndex : Integer] : IOutputer Read GetState_Outputer;
Property  OutputerCount               : Integer   Read GetState_OutputerCount;
```

See also

Workspace Manager Interfaces
IProject interface
IOutputJob interface
IWSM_OutputJobDocument interface

IWSM_ServerInterface interface

Overview

The IWSM_ServerInterface interface represents the high level mechanism that manages different output job documents.

Interface Methods

```
Function  GetOutputJobDocumentByPath (APath : WideString) : IWSM_OutputJobDocument;
```

Interface Properties

See also

Workspace Manager Interfaces
IProject interface
IOutputJob interface
IWSM_OutputJobDocument interface

IDifferentialPair interface

Overview

The IDifferentialPair interface represents a differential pair object.

Interface Methods

```
IObjectClass methods
```

Interface Properties

```
IObjectClass properties
```

See also

Workspace Manager Interfaces

IDatabaseLibDocument interface

Overview

The IDatabaseLibDocument interface represents a database library document.

Interface Methods

```
Procedure GetModelFieldNamesAt(AnIndex            : Integer;
                               ATableIndex        : Integer;
                               AModelType         : WideString;
                           Var AModelPathName : WideString;
                           Var AModelRefName  : WideString;
                           Var AOrcadModel    : Boolean);
Procedure InitialiseExportToDatabase (ADatabaseFileName  : WideString);
Procedure FinaliseExportFromDatabase;
Procedure DisposeIfNotShowing;
Procedure GetOrcadLibraryDetails     (AParseString       : WideString;
                                      Var LibRef         : WideString;
                                      Var LibPath        : WideString);

Function  ExportNewRecordFromIntLib  (ATableName       : WideString;
                                      AFieldParameters : WideString) : WideString;
```

```
Function InitialiseExportFromIntLib  (ATableName         : WideString) : WideString;
Function GetConnectionString                 : WideString;
Function GetCommandString(ATableIndex        : Integer;
                          AFilterText        : WideString;
                          ASQLWhereClause    : WideString) : WideString;
Function GetFilterText   (ATableIndex        : Integer;
                          AFilterColumnNames : WideString;
                          AFilterValue       : WideString) : WideString;
Function GetParametersForComponent   (ATableIndex        : Integer;
                                      AComponentKeys     : WideString) : WideString;

Function GetTableCount : Integer;
Function GetFileName   : Widestring;

Function GetKeyFieldCount(ATableIndex       : Integer)     : Integer;
Function GetKeyField     (AParameterName    : Boolean;
                          ATableIndex       : Integer;
                          AKeyIndex         : Integer)     : WideString;
Function GetLibraryRefFieldName(ATableIndex       : Integer;
                            Var AOrcadLibrary : Boolean)     : WideString;
Function GetLibraryPathFieldName(ATableIndex   : Integer)         : WideString;
Function GetFieldCount   (ATableIndex   : Integer)         : Integer;
Function GetTableNameAt  (AnIndex       : Integer)         : WideString;
Function GetFieldNameAt  (ATableIndex   : Integer;
                          AFieldIndex   : Integer)         : WideString;
Function GetTableIndex   (ATableName    : WideString)      : Integer;
Function TableEnabled    (AnIndex       : Integer)         : Boolean;
Function DocumentObject                               : Pointer;
Function GetItemCount(    ACommand  : WideString;
                      Var AnError   : WideString)      : Integer;
Function TableContainsColumn(ATableIndex   : Integer;
                             AColumnName   : WideString)      : Boolean;
Function IsValidSQLStatementForTable (ATableName    : WideString;
                                      AQuery        : WideString)      : Boolean;
Function LoadAllRecordsLimit                                          : Integer;

Function ValidateSQLQuery            (ASqlQuery         : WideString)  : WideString;
Function GetDatafilePath             (AName             : WideString;
                                      AType             : WideString;
                                      ATableName        : WideString;
                                      AComponentKeys    : WideString)  : WideString;
    Function GetSchLibPathForComponent   (ATableIndex       : Integer;
                                      AComponentKeys    : WideString)  : WideString;
Function GetSchLibRefForComponent(ATableIndex       : Integer;
                                  AComponentKeys    : WideString)  : WideString;
```

```
Function IsParameterDatabaseKey(ATableIndex       : Integer;
                                AParameterName    : WideString)  : Boolean;

Function ObjectAddress           : Pointer;
Function GetLibrarySearchPath    : WideString;
Function GetSearchSubDirectories : Boolean;
Function OrcadDelimiter           : Char;
```

Interface Properties

`IObjectClass properties`

See also

Workspace Manager Interfaces

Design Objects

IDMObject interface

Overview

The IDMObject interface is the base object interface for all object interfaces used in the Work Space Manager system extension server for Altium Designer.

IDMObject methods

DM_CurrentSheetInstanceNumber
DM_FullCrossProbeString
DM_GeneralField
DM_ImageIndex
DM_IsInferredObject
DM_LocationString
DM_LocationX
DM_LocationY
DM_LongDescriptorString
DM_NetIndex_Flat
DM_NetIndex_Sheet
DM_NetIndex_SubNet
DM_ObjectAdress
DM_ObjectKindString
DM_ObjectKindStringForCrossProbe
DM_OwnerDocument
DM_OwnerDocumentFullPath
DM_OwnerDocumentName
DM_ParameterCount
DM_Parameters
DM_PCBObjectHandle
DM_PrimaryCrossProbeString
DM_SCHObjectHandle
DM_SecondaryCrossProbeString
DM_SheetIndex_Logical
DM_SheetIndex_Physical
DM_ShortDescriptorString

IDMObject properties

DM_ValidForNavigation

DM_VHDLEntity

See also

Methods

DM_CurrentSheetInstanceNumber method

(IDMObject interface)

Syntax

```
Function DM_CurrentSheetInstanceNumber : Integer;
```

Description

The function returns the current sheet instance number of the schematic document.

See also

IDMObject interface

DM_FullCrossProbeString method

(IDMObject interface)

Syntax

```
Function DM_FullCrossProbeString : WideString;
```

Description

The function returns the full cross probe string.

See also

IDMObject interface

DM_GeneralField method

(IDMObject interface)

Syntax

```
Function DM_GeneralField : Integer;
```

Description

The function can returns an integral value for this general field. This General Field can be used for any purpose - as a tag property, as an index property or as a flag to denote something.

See also

IDMObject interface

DM_ImageIndex method

(IDMObject interface)

Syntax

```
Function DM_ImageIndex : Integer;
```

Description

The function returns the image index depending on what type of object the image represents.

See also

IDMObject interface

DM_IsInferredObject method

(IDMObject interface)

Syntax

```
Function DM_IsInferredObject : Boolean;
```

Description

The function denotes whether the object is an inferred object with respect to connective objects. Bus and Sheet Symbols can be defined in ranges using the NetLabel [] and Repeat statements respectively and once the project has been compiled, inferred

objects are created in memory for navigation/connective purposes. For example, a Bus with a range of A[0..4] ends up with five wires with A0...A5 net labels (only in memory). This property is useful for multi – channel projects and for sheets that have Bus objects.

See also

IDMObject interface

DM_LocationString method

(IDMObject interface)

Syntax

```
Function DM_LocationString : WideString;
```

Description

The function returns the Location string formatted as a X,Y format or if the object kind is a Text Documnt set, then the string returned is a formatted Line: LocationY Offset: XLocation string.

See also

IDMObject interface

DM_LocationX method

(IDMObject interface)

Syntax

```
Function DM_LocationX : Integer;
```

Description

The function returns the location of this interface object on the X axis.

See also

IDMObject interface

DM_LocationY method

(IDMObject interface)

Syntax

```
Function DM_LocationY : Integer;
```

Description

The function returns the location of this interface object on the Y axis.

See also

IDMObject interface

DM_LongDescriptorString method

(IDMObject interface)

Syntax

```
Function DM_LongDescriptorString : WideString;
```

Description

The function returns the long description version string.

See also

IDMObject interface

DM_NetIndex_Flat method

(IDMObject interface)

Syntax

```
Function DM_NetIndex_Flat : Integer;
```

Description

The function returns the net index for a flattened design.

See also

IDMObject interface

DM_NetIndex_Sheet method

(IDMObject interface)

Syntax

```
Function DM_NetIndex_Sheet : Integer;
```

Description

The function returns the netindex for a schematic sheet.

See also

IDMObject interface

DM_NetIndex_SubNet method

(IDMObject interface)

Syntax

```
Function DM_NetIndex_SubNet : Integer;
```

Description

The function returns the net index within a sub net.

See also

IDMObject interface

DM_ObjectAdress method

(IDMObject interface)

Syntax

```
Function DM_ObjectAdress : Pointer;
```

Description

The function returns the pointer of the interface object itself. Also called a handle.

See also

IDMObject interface

DM_ObjectKindString method

(IDMObject interface)

Syntax

```
Function DM_ObjectKindString : WideString;
```

Description

The function returns the object kind string which denotes the design document type.

See also

IDMObject interface

DM_ObjectKindStringForCrossProbe method

(IDMObject interface)

Syntax

```
Function DM_ObjectKindStringForCrossProbe : WideString;
```

Description

The function returns the specially formatted object kind string for the cross probing mechanism.

See also

IDMObject interface

DM_OwnerDocument method

(IDMObject interface)

Syntax

```
Function DM_OwnerDocument : IDocument;
```

Description

The function returns the document interface object. Refer to IDocument interface for details.

See also

IDMObject interface

DM_OwnerDocumentFullPath method

(IDMObject interface)

Syntax

```
Function DM_OwnerDocumentFullPath : WideString;
```

Description

The function returns the full path of the document.

See also

IDMObject interface

DM_OwnerDocumentName method

(IDMObject interface)

Syntax

```
Function DM_OwnerDocumentName : WideString;
```

Description

The function returns the name of the document that this object interface is part of.

See also

IDMObject interface

DM_ParameterCount method

(IDMObject interface)

Syntax

```
Function DM_ParameterCount : Integer;
```

Description

The function returns the number of parameters this object has.

See also

IDMObject interface

DM_Parameters method

(IDMObject interface)

Syntax

```
Function DM_Parameters (Index : Integer) : IParameter;
```

Description

The function returns the indexed parameter object with the index parameter. Use the IParameter interface to wrap the returned result.

See also

IDMObject interface

DM_PCBObjectHandle method

(IDMObject interface)

Syntax

```
Function DM_PCBObjectHandle : Integer;
```

Description

The function returns the object handle of a PCB object. If void, a Nil value is returned.

See also

IDMObject interface

DM_PrimaryCrossProbeString method

(IDMObject interface)

Syntax

```
Function DM_PrimaryCrossProbeString : WideString;
```

Description

The function returns the primary cross probe string.

See also

IDMObject interface

DM_SCHObjectHandle method

(IDMObject interface)

Syntax

```
Function DM_SCHObjectHandle : Pointer;
```

Description

The function returns the object handle of a Schematic object. If void, a zero value is returned.

See also

IDMObject interface

DM_SecondaryCrossProbeString method

(IDMObject interface)

Syntax

```
Function DM_SecondaryCrossProbeString : WideString;
```

Description

The function returns the secondary cross probe string.

See also

IDMObject interface

DM_SheetIndex_Logical method

(IDMObject interface)

Syntax

```
Function DM_SheetIndex_Logical : Integer;
```

Description

The function returns the sheet index for a logical design (multi – channel designs for example).

See also

IDMObject interface

DM_SheetIndex_Physical method

(IDMObject interface)

Syntax

```
Function DM_SheetIndex_Physical : Integer;
```

Description

The function returns the sheet index for a physical design. (that have unique designators)

See also

IDMObject interface

DM_ShortDescriptorString method

(IDMObject interface)

Syntax

```
Function DM_ShortDescriptorString : WideString;
```

Description

The function returns the short description version string.

See also

IDMObject interface

DM_ValidForNavigation method

(IDMObject interface)

Syntax

```
Function DM_ValidForNavigation : Boolean;
```

Description

The function toggles whether navigation is valid for this object. Navigation is performed on net aware objects such as components, nets and busses.

See also

IDMObject interface

DM_VHDLEntity method

(IDMObject interface)

Syntax

```
Function DM_VHDLEntity : IVHDLEntity;
```

Description

The function returns the VHDL entity interface object if it exists on a VHDL document. Basically every object interface has an access to this VHDL entity interface, so to check whether VHDL entity exists for this particular object, you can check out the Name field within the IVHDLEntity interface.

See also

IDMObject interface

DM_GetVCSProject

(IDMObject interface)

Syntax

Description

Example

See also

IClient interface

IDMObject interface

Properties

VCSProject property

(IDMObject interface)

Syntax

Description

Example

See also

IClient interface

IVCSProjectAccessor interface

IWorkspace interface

Overview

The **IWorkspace** interface represents the Work-Space manager in the Altium Designer which deals with project and documents and their related attributes and options. This interface object is the starting point and upon querying this object, it can return currently open projects, number of projects, installed libraries, and create a new document for example.

Remember the projects need to be compiled first, before you can invoke the **GetWorkSpace** function to obtain the **IWorkSpace** interface and its descendant interfaces which represent actual objects in Altium Designer.

It is highly recommended not to hold an interface of the Workspace manager, but re-query the work-space manager every-time the access to the information within is required.

IWorkspace methods

DM_AddDocumentToActiveProject
DM_AddOutputLine
DM_ChangeManager
DM_ClearOutputLines
DM_CreateNewDocument
DM_FileOwnership_BulkWarnEnd
DM_FileOwnership_BulkWarnRegister
DM_FileOwnership_BulkWarnStart
DM_FocusedDocument
DM_FocusedProject
DM_FreeDocumentsProject
DM_GenerateUniqueID
DM_GetDefaultPcbType
DM_GetDocumentFromPath
DM_GetOutputLine
DM_GetOutputLineCount
DM_GetProjectFromPath
DM_GetRecoveryInterval
DM_GetRecoveryIsEnabled
DM_ImageIndexForDocumentKind
DM_InstalledLibraries
DM_InstalledLibraryCount
DM_LoadProjectHidden
DM_LocalHistoryManager
DM_MessagesManager
DM_NavigationZoomPrecision
DM_OpenProject
DM_OptionsStorage
DM_Preferences
DM_ProjectCount
DM_Projects
DM_PromptForDefaultPcbType
DM_RunOpenDocumentDialog
DM_SetRecoveryParameters
DM_SetRightClickCompStructure

IWorkspace properties

DM_ShowMessageView
DM_ShowToDoList
DM_ViolationTypeDescription
DM_ViolationTypeGroup
DM_WorkspaceFileName
DM_WorkspaceFullPath

See also

Methods

DM_AddDocumentToActiveProject method

(IWorkspace interface)

Syntax

```
Procedure DM_AddDocumentToActiveProject(DocumentPath : WideString);
```

Description

This method adds an existing document with its valid full path into an active project within Altium Designer.

Example

See also

IWorkspace

DM_AddOutputLine method

(IWorkspace interface)

Syntax

```
Procedure DM_AddOutputLine(MessageLine : PChar;ReplaceLastLine : Boolean = False);
```

Description

Outputs the line to the output's dialog. An Internal operation.

Example

See also

IWorkspace

DM_ChangeManager method

(IWorkspace interface)

Syntax

```
Function DM_ChangeManager : IChangeManager;
```

Description

Returns the Engineering Change Order Manager interface object which compares with two projects and creates an ECO to perform a pin swapping process to synchronize the specified two projects.

Example

See also

IWorkspace

DM_ClearOutputLines method

(IWorkspace interface)

Syntax

```
Procedure DM_ClearOutputLines;
```

Description

Clears out the Output Memo. An Internal operation.

Example

See also

IWorkspace

Description

See also

IClient interface

IWorkSpace interface

DM_CreateNewDocument method

(IWorkspace interface)

Syntax

```
Function DM_CreateNewDocument (ADocKind : WideString) : WideString;
```

Description

This method creates a new document based on the Document Kind. The Kinds include – 'PCBLIB','PCB','SCH','SCHLIB' and so on depending on which document servers are installed in Altium Designer.

DelphiScript Example

//Creating a new PCB document in Altium Designer

```
Var
    WSM         : IWorkSpace;
Begin
    WSM := GetWorkSpace;
    If WSM = Nil Then Exit;
    WSM.DM_CreateNewDocument('PCB');
End;
```

See also

IWorkspace

DM_FileOwnerShip_BulkWarnStart method

(IWorkspace interface)

Syntax

```
Procedure   DM_FileOwnership_BulkWarnStart;
```

Description

Example

See also

IWorkspace interface

DM_FileOwnerShip_BulkWarnEnd method

(IWorkspace interface)

Syntax

```
Procedure   DM_FileOwnership_BulkWarnEnd(AWarningLevel : TFileOwnershipWarningLevel);
```

Description

Example

See also

IWorkspace interface

DM_FileOwnerShip_BulkWarnRegister method

(IWorkspace interface)

Syntax

```
Function    DM_FileOwnership_BulkWarnRegister(AFileName : TDynamicString) : Boolean;
```

Description

Example

See also

IWorkspace interface

DM_FocusedDocument method

(IWorkspace interface)

Syntax

```
Function DM_FocusedDocument : IDocument;
```

Description

Returns the focussed document interface object (if any) in Altium Designer. A focussed document is a document that is currently open and in focus (this document is active).

Example

See also

IWorkspace

DM_FocusedProject method

(IWorkspace interface)

Syntax

```
Function DM_FocusedProject : IProject;
```

Description

Returns the focussed project (if any) in Altium Designer.

Example

See also

IWorkspace

DM_FreeDocumentsProject method

(IWorkspace interface)

Syntax

```
Function DM_FreeDocumentsProject : IProject;
```

Description

Returns the **IProject** interface that contains free documents. A free document is a standalone document that lives in the Free Documents project.

Example

See also

IWorkspace

DM_GenerateUniqueID method

(IWorkspace interface)

Syntax

```
Function DM_GenerateUniqueID : WideString;
```

Description

Invoke this method, and a generated Unique ID will be returned which can be used for any newly created or existing object in Altium Designer. Objects in Schematic have their own Unique IDs which are tracked by the synchronizator so that the objects on the PCB document are sychronized to their equivalents in a corresponding schematic project.

Example - an incomplete example that assigns new UIDs to Schematic components

```
// get the workspace manager interface so you can
// generate unique ids...
WSM := GetWorkspace;
If WSM = Nil Then Exit;
// get the schematic server interface
If SchServer = Nil Then Exit;
// get the current sch sheet
CurrentSheet := SchServer.GetCurrentSchDocument;
If CurrentSheet = Nil Then Exit;

// Set up an iterator to look for components
// on a schematic document.
Iterator := CurrentSheet.SchIterator_Create;
Iterator.AddFilter_ObjectSet(MkSet(eSchComponent));

Try
    Comp := Iterator.FirstSchObject;
    While Compt <> Nil Do
    Begin
        Comp.UniqueID := WSM. DM_GenerateUniqueID;
        Comp := Iterator.NextSchObject;
    End;
Finally
    CurrentSheet.SchIterator_Destroy(Iterator);
End;
```

See also

IWorkspace

DM_GetDefaultPcbType method

(IWorkspace interface)

Syntax

```
Function DM_GetDefaultPcbType : WideString;
```

Description

Example

See also

IWorkspace

DM_GetDocumentFromPath method

(IWorkspace interface)

Syntax

```
Function DM_GetDocumentFromPath(DocumentPath : WideString) : IDocument;
```

Description

Retrieves the IDocument interface object by passing the full document path to this document. With this IDocument interface, you have access to its functionality, such as compiling the document itself.

Example

See also

IWorkspace

DM_GetOutputLine method

(IWorkspace interface)

Syntax

```
Function DM_GetOutputLine(Index : Integer) : WideString;
```

Description

Example

See also

IWorkspace

DM_GetOutputLineCount method

(IWorkspace interface)

Syntax

```
Function DM_GetOutputLineCount : Integer;
```

Description

Returns the number of output lines in the Output dialog. An Internal operation.

Example

See also

IWorkspace

DM_GetProjectFromPath method

(IWorkspace interface)

Syntax

```
Function DM_GetProjectFromPath (ProjectPath : WideString) : IProject;
```

Description

Retrieves the IProject interface object by passing the full project path to this project. With this IProject interface, you have access to its interface methods. A project is a container that has links to associated design documents in an organized manner.

Example

See also

IWorkspace

DM_GetRecoveryInterval method

(IWorkspace interface)

Syntax

```
Function DM_GetRecoveryInterval : Integer;
```

Description

Returns the number of minutes as the interval when the recovery mechanism kicks in.

Example

See also

IWorkspace

DM_GetRecoveryIsEnabled method

(IWorkspace interface)

Syntax

```
Function DM_GetRecoveryIsEnabled : Boolean;
```

Description

Returns a Boolean value whether the recovery mechanism is active or not.

Example

See also

IWorkspace

DM_ImageIndexForDocumentKind method

(IWorkspace interface)

Syntax

```
Function DM_ImageIndexForDocumentKind(ADocumentKind : WideString) : Integer;
```

Description

Returns the image index depending on the document kind for example PCB, CAM etc.

Example

See also

IWorkspace

Image Index Table

DM_InstalledLibraries method

(IWorkspace interface)

Syntax

```
Function DM_InstalledLibraries (Index : Integer) : IDocument;
```

Description

Returns an indexed library (currently installed in Altium Designer only), to be used in conjunction with the DM_InstalledLibraryCount.

Example

See also

IWorkspace interface

DM_InstalledLibraryCount method

DM_InstalledLibraryCount method

(IWorkspace interface)

Syntax

```
Function DM_InstalledLibraryCount : Integer;
```

Description

Returns the number of installed libraries in Altium Designer.

Example

See also

IWorkspace

DM_MessagesManager method

(IWorkspace interface)

Syntax

```
Function DM_MessagesManager : IMessagesManager;
```

Description

This function returns you the interface to the Messages panel in Altium Designer.

Example

See also

IWorkspace interface

IMessagesManager interface

DM_OpenProject method

(IWorkspace interface)

Syntax

```
Function DM_OpenProject ( ProjectPath : WideString;Const Show : Boolean) : IProject;
```

Description

Opens a project with the full project path and set this project in focus depending on its Show parameter.

Example

See also

IWorkspace

DM_OptionsStorage method

(IWorkspace interface)

Syntax

```
Function DM_OptionsStorage : IOptionsStorage;
```

Description

Represents a options storage container where Altium Designer can use to retrieve and store options for storing parameters of EDE options such as Toolchain name, folder and default options and project options.

Example

See also

IWorkspace

DM_ProjectCount method

(IWorkspace interface)

Syntax

```
Function DM_ProjectCount : Integer;
```

Description

Returns the number of projects open in Altium Designer.

Example

See also

IWorkspace

DM_Preferences method

(IWorkspace interface)

Syntax

```
Function DM_Preferences : IWorkspacePreferences;
```

Description

Example

See also

IWorkspace interface

IWorkspacePreferences interface

DM_Projects method

(IWorkspace interface)

Syntax

```
Function DM_Projects (Index : Integer) : IProject;
```

Description

Returns the indexed project (currently loaded in Altium Designer only), to be used in conjunction with the DM_ProjectCount interface.

Example

See also

IWorkspace

DM_PromptForDefaultPcbType method

(IWorkspace interface)

Syntax

```
Function DM_PromptForDefaultPcbType(Var PcbType : WideString) : Boolean;
```

Description

Example

See also

IWorkspace

DM_SetRecoveryParameters method

(IWorkspace interface)

Syntax

```
Procedure DM_SetRecoveryParameters(IsEnabled : Boolean; Interval : Integer);
```

Description

Set the interval when the autosave / recovery mechanism in Altium Designer kicks in. The interval is in minutes, and whether to enable the recovery mechanism.

Example

See also

IWorkspace

DM_ShowMessageView method

(IWorkspace interface)

Syntax

```
Procedure DM_ShowMessageView;
```

Description

Invoke this method to refresh the Message panel.

Example

See also

IWorkspace

DM_ShowToDoList method

(IWorkspace interface)

Syntax

```
Procedure DM_ShowToDoList;
```

Description

This method displays the To Do List manager panel. This To Do List panel can be used to define your To Dos.

Example

See also

IWorkspace

DM_ViolationTypeDescription method

(IWorkspace interface)

Syntax

```
Function DM_ViolationTypeDescription(ErrorKind : TErrorKind) : WideString;
```

Description

Returns the violation type description string with the error kind value passed in. Check the TErrorKind for its range of values.

Example

See also

IWorkspace

DM_ViolationTypeGroup method

(IWorkspace interface)

Syntax

```
Function DM_ViolationTypeGroup (ErrorKind : TErrorKind) : TErrorGroup;
```

Description

Returns the error group for which this error kind parameter belongs to. Check the TErrorGroup type for its range of values.

Example

See also

IWorkspace

DM_WorkspaceFileName method

(IWorkspace interface)

Syntax

```
Function DM_WorkspaceFileName : WideString;
```

Description

Returns the filename only of the workspace.

Example

See also

IWorkspace

DM_WorkspaceFullPath method

(IWorkspace interface)

Syntax

```
Function DM_WorkspaceFullPath : WideString;
```

Description

Returns the full path of the workspace.

Example

See also

IWorkspace

IBus interface

Overview

The IBus interface represents a bus object on the schematic sheet. Buses are special graphical elements that represent a common pathway for multiple signals on a schematic document. Buses have no electrical properties, and they must be correctly identified by net labels and ports.

When a schematic document is compiled, bus objects have inferred objects (wires with netlabels on them) in memory that aids the connectivity and navigation features within Altium Designer.

IBus methods	IBus properties
DM_Wires	
DM_Sections	
DM_WireCount	
DM_SectionCount	
DM_Scope	
DM_Electric	
DM_SignalType	
DM_FullBusName	
DM_BusName	
DM_BusRange1	
DM_BusRange2	
DM_BusRangeValue1	
DM_BusRangeValue2	
DM_BusKind	
DM_BusWidth	
DM_RangeDefinedByValue	

DM_IsLocal

See also

Methods

DM_BusKind method

(IBus interface)

Syntax

```
Function DM_BusKind : TBusKind;
```

Description

The function returns the bus kind which is one of the following enumerated values; eBusKindUndefined, eBusKindLowValueFirst, eBusKindHighValueFirst, eBusKindGeneric.

Example

See also

IBus interface

TBusKind type (from RT_Unit in Altium Designer RTL)

DM_BusName method

(IBus interface)

Syntax

```
Function DM_BusName : WideString;
```

Description

This function returns the full bus name of this bus interface.

Example

See also

IBus interface

DM_BusRange1 method

(IBus interface)

Syntax

```
Function DM_BusRange1 : WideString;
```

Description

This function returns the Bus range 1 value. Eg. A[1..6], the bus range1 is 1.

Example

See also

IBus interface

DM_BusRange2 method

(IBus interface)

Syntax

```
Function DM_BusRange2 : WideString;
```

Description

The function returns the Bus range 2 value. Eg. A[1..6], the bus range2 is 6.

Example

See also

IBus interface

DM_BusRangeValue1 method

(IBus interface)

Syntax

```
Function DM_BusRangeValue1 : Integer;
```

Description

The function returns the Bus range 1 value.

Example

See also

IBus interface

DM_BusRangeValue2 method

(IBus interface)

Syntax

```
Function DM_BusRangeValue2 : Integer;
```

Description

The function returns the Bus range 2 value.

Example

See also

IBus interface

DM_BusWidth method

(IBus interface)

Syntax

```
Function DM_BusWidth : Integer;
```

Description

The function returns the bus width.

Example

See also

IBus interface

DM_Electric method

(IBus interface)

Syntax

```
Function DM_Electric : TPinElectrical;
```

Description

The function returns the electrical property for this bus. Various values include :eElectricInput, eElectricIO, eElectricOutput, eElectricOpenCollector, eElectricPassive, eElectricHiZ, eElectricOpenEmitter, eElectricPower

Example

See also

IBus interface

TPinElectrical type

DM_FullBusName method

(IBus interface)

Syntax

```
Function DM_FullBusName : WideString;
```

Description

The function returns the full bus name of this bus interface.

Example

See also

IBus interface

DM_IsLocal method

(IBus interface)

Syntax

```
Function DM_IsLocal : Boolean;
```

Description

The function returns a Boolean value whether this bus is a local object or not.

Example

See also

IBus interface

DM_RangeDefinedByValue method

(IBus interface)

Syntax

```
Function DM_RangeDefinedByValue : Boolean;
```

Description

The function returns a Boolean value whether this range is defined by a two specific range values or not.

Example

See also

IBus interface

DM_Scope method

(IBus interface)

Syntax

```
Function DM_Scope : TNetScope;
```

Description

The function denotes the net scope of this IBus interface.

Example

See also

IBus interface

TNetScope type

DM_SectionCount method

(IBus interface)

Syntax

```
Function DM_SectionCount : Integer;
```

Description

The function returns the number of sections for this IBus interface. This is used for the DM_Sections function.

Example

See also

IBus interface

DM_Sections method

(IBus interface)

Syntax

```
Function DM_Sections (Index : Integer) : INet;
```

Description

The function returns the indexed section of the bus. Used in conjunction with the DM_SectionCount function.

Example

See also

IBus interface

DM_SignalType method

(IBus interface)

Syntax

```
Function DM_SignalType : WideString;
```

Description

The function returns the signal type string for this bus.

Example

See also

IBus interface

DM_WireCount method

(IBus interface)

Syntax

```
Function DM_WireCount : Integer;
```

Description

The function returns the number of wires for this IBus interface. This is used for the DM_Wires function.

Example

See also

IBus interface

DM_Wires method

(IBus interface)

Syntax

```
Function DM_Wires (Index : Integer) : INet;
```

Description

The function returns the indexed wire. Used in conjunction with the DM_WireCount function.

Example

See also

IBus interface

IChannelClass interface

Overview

The IChannelClass interface is a PCB Channel class object interface for an existing Channel Class on a PCB document. An existing Channel (room) class contains members of specific components. Each component within a Channel Class object can either be a member or not. The ‘All Components’ Channel Class exists in every PCB document by default, it includes all Components in the document. It is not possible to change which components are members of that Channel class, but the user has full control over which components are members of any other Channel classes (which are created and named by the User)

Notes

- Inherited from IObjectClass interface.

See also

IObjectClass interface

IComponent Interface

Overview

The IChannelClass interface is a PCB Channel class object interface for an existing Channel Class on a PCB document. An existing Channel (room) class contains members of specific components.

Each component within a Channel Class object can either be a member or not. The ‘All Components’ Channel Class exists in every PCB document by default, it includes all Components in the document.

It is not possible to change which components are members of that Channel class, but the user has full control over which components are members of any other Channel classes (which are created and named by the User)

IComponent methods

DM_SubParts

DM_PhysicalComponents

DM_SubPartCount

DM_PhysicalComponentCount

DM_PhysicalPath

DM_UniqueId

DM_UniqueIdName

DM_UniqueIdPath

IComponent properties

See also

Methods

DM_SubParts method

(IComponent interface)

Syntax

```
Function DM_SubParts (Index : Integer) : IPart;
```

Description

The function returns the indexed sub-part of a multi-part component. Use the DM_SubPartCount function.

Example

See also

IComponent interface

DM_SubPartCount method

(IComponent interface)

Syntax

```
Function DM_SubPartCount : Integer;
```

Description

The function returns the number of parts for this multi-part component. A standalone component returns 1 (only one part for a standalone component).

Example

See also

IComponent interface

DM_PhysicalComponents method

(IComponent interface)

Syntax

```
Function DM_PhysicalComponents (Index : Integer) : IComponent;
```

Description

The function returns the indexed physical component. Use this in conjunction with the DM_PhysicalComponentCount function.

Example

See also

IComponent interface

DM_PhysicalComponentCount method

(IComponent interface)

Syntax

```
Function DM_PhysicalComponentCount : Integer;
```

Description

The function returns the number of physical components.

Example

See also

IComponent interface

DM_UniqueIdPath method

(IComponent interface)

Syntax

```
Function DM_UniqueIdPath : WideString;
```

Description

The function returns the unique path portion of the Unique ID for this component. Includes the back slash.

Example

See also

IComponent interface

DM_UniqueIdName method

(IComponent interface)

Syntax

```
Function DM_UniqueIdName : WideString;
```

Description

The function returns the unique name portion of the Unique ID for this component.

Example

See also

IComponent interface

DM_UniqueId method

(IComponent interface)

Syntax

```
Function DM_UniqueId : WideString;
```

Description

The function returns the Unique ID string for this component so this component can be synchronized on the source document and the primary implementation document (PCB).

Example

See also

IComponent interface

DM_PhysicalPath method

(IComponent interface)

Syntax

```
Function DM_PhysicalPath : WideString;
```

Description

The function returns the full physical path for this component. For example the string can consist of the schematic filename \ channel name and instance.

Example

See also

IComponent interface

IComponentClass interface

Overview

The IComponentClass interface is a PCB Component class object interface for an existing Component Class on a PCB document. An existing Component class contains members of specific Components. Each Component within a ComponentClass object can either be a member or not. The ‘All Components’ Component Class exists in every PCB document by default, it includes all Components in the document. It is not possible to change which components are members of that Component class, but the user has full control over which components are members of any other Component classes (which are created and named by the User).

Notes

- Inherited from IObjectClass interface.

See also

IObjectClass interface

IComponentImplementation interface

Overview

The IComponentImplementation interface is associated with an IPart/IComponent interface in terms of model linking. Note that the IComponent interface is inherited from the IPart interface.

A model represents all the information needed for a component in a given domain (a model can be a PCB footprint, Simulation file or a Signal Integrity model). A model is also called an implementation.

Each schematic component can contain links to different model implementations such as PCB, Signal Integrity and Simulation models. Only one model of a particular model type (PCB footprint, SIM, SI, EDIF Macro and VHDL) can be enabled as the currently linked model, at any one time.

A model can be represented by external data sources called data file links. For example, pins of a component can have links to different data files, as for signal integrity models. We will consider each model type in respect to the data file links.

- For PCB footprints, the data file link and the model is the same since the external file is the PCB footprint library.
- For simulation models, there can be no data file links because these models are defined using the Spice format.
- However for signal integrity models, each pin can have different pieces of information represented by ibis data files. These signal integrity models can have multiple data files, that is, each pin of a component can have a separate IBIS file. A signal integrity model can however use the Altium Designer's central Signal Integrity database.

Thus depending on which model type, you can have a number of data file links. Each data file link describes the model name, the path to where the library is stored in and what sort of model it is.

IComponentImplementation methods

DM_Description
DM_ModelName
DM_ModelType
DM_DatafileCount
DM_DatafileLocation
DM_DatafileEntity
DM_DatafileKind
DM_SetDatafileLocation
DM_SetDatafileEntity
DM_SetDatafileKind
DM_SetDatafileCount
DM_DatafileFullPath
DM_IntegratedModel
DM_DatalinksLocked
DM_IsCurrent
DM_Part
DM_PortMap
DM_PortMapList

IComponentImplementation properties

See also

Methods

DM_SetDatafileKind method

(IComponentImplementation interface)

Syntax

```
Procedure DM_SetDatafileKind (Index : Integer; AKind : WideString);
```

Description

The procedure sets the data file kind which denotes the type of implementation model. Example, a PCB Footprint is a PCBLIB data file kind.

Example

See also

IComponentImplementation interface

DM_SetDatafileEntity method

(IComponentImplementation interface)

Syntax

```
Procedure DM_SetDatafileEntity (Index : Integer; AEntity : WideString);
```

Description

The procedure sets the data file entity which denotes the name of the implementation model linked to a schematic component/part.

Example

See also

IComponentImplementation interface

DM_SetDatafileCount method

(IComponentImplementation interface)

Syntax

```
Procedure DM_SetDatafileCount (ACount : Integer);
```

Description

The procedure sets the number of data files associated with the IPart/IComponent interface.

Example

See also

IComponentImplementation interface

DM_ModelType method

(IComponentImplementation interface)

Syntax

```
Function DM_ModelType : WideString;
```

Description

The function returns the model type as a string;

Example

See also

IComponentImplementation interface

DM_ModelName method

(IComponentImplementation interface)

Syntax

```
Function DM_ModelName : WideString;
```

Description

The function returns the model name of the implementation model.

Example

See also

IComponentImplementation interface

DM_Description method

(IComponentImplementation interface)

Syntax

```
Function DM_Description : WideString;
```

Description

The function returns the description string of the implementation model.

Example

See also

IComponentImplementation interface

DM_DatafileLocation method

(IComponentImplementation interface)

Syntax

```
Function DM_DatafileLocation (Index : Integer) : WideString;
```

Description

The function returns the indexed data file location. Used in conjunction with the DM_DataFileCount function.

Example

See also

IComponentImplementation interface

DM_DatafileKind method

(IComponentImplementation interface)

Syntax

```
Function DM_DatafileKind (Index : Integer) : WideString;
```

Description

The function returns the indexed data file kind (the model kind eg PCB etc)Used in conjunction with the DM_DataFileCount function.

Example

See also

IComponentImplementation interface

DM_DatafileFullPath method

(IComponentImplementation interface)

Syntax

```
Function DM_DatafileFullPath (Index : Integer;EntityName, FileKind : WideString;Var FoundIn :
WideString) : WideString;
```

Description

The function returns you the full path to the data file via the FoundIn parameter, if the Entity name, the file Kind are valid and Found In strings Used in conjunction with the DM_DataFileCount function.

Example

See also

IComponentImplementation interface

DM_DatafileEntity method

(IComponentImplementation interface)

Syntax

```
Function DM_DatafileEntity (Index : Integer) : WideString;
```

Description

The function returns the indexed data file entity (the name of the implementation model). Used in conjunction with the DM_DataFileCount function.

Example

See also

IComponentImplementation interface

DM_DatafileCount method

(IComponentImplementation interface)

Syntax

```
Function DM_DatafileCount : Integer;
```

Description

The function returns the number of data files for the model. A data file is an internal aggregrate object and each data file describes the model name, the path to where the library is stored in and what implementation model type.

Example

See also

IComponentImplementation interface

DM_SetDatafileLocation method

(IComponentImplementation interface)

Syntax

```
Procedure DM_SetDatafileLocation (Index : Integer; ALocation : WideString);
```

Description

The procedure sets the data file location which denotes the full path of the implementation model associated with the IPart/IComponent interface.

Example

See also

IComponentImplementation interface

DM_PortMapList method

(IComponentImplementation interface)

Syntax

```
Function DM_PortMapList : WideString;
```

Description

The function returns the mapping of pins of a component and its corresponding model.

Example

See also

IComponentImplementation interface

DM_PortMap method

(IComponentImplementation interface)

Syntax

```
Function DM_PortMap : WideString;
```

Description

The function denotes the mapping of pins of a component and its corresponding model.

Example

See also

IComponentImplementation interface

DM_Part method

(IComponentImplementation interface)

Syntax

```
Function DM_Part : IPart;
```

Description

The function denotes the mapping of pins of a component and its corresponding model.

Example

See also

IComponentImplementation interface

DM_IsCurrent method

(IComponentImplementation interface)

Syntax

```
Function DM_IsCurrent : Boolean;
```

Description

The function denotes a boolean value whether this model implementation is current or not.

Example

See also

IComponentImplementation interface

DM_IntegratedModel method

(IComponentImplementation interface)

Syntax

```
Function DM_IntegratedModel : Boolean;
```

Description

This function denotes a boolean value whether this is a model from an integrated library or not.

Example

See also

IComponentImplementation interface

DM_DatalinksLocked method

(IComponentImplementation interface)

Syntax

```
Function DM_DatalinksLocked : Boolean;
```

Description

The function denotes a boolean value whether datalinks are locked or not. Note, a data file kind denotes the type of implementation model. Example, a PCB Footprint is a PCBLIB data file kind.

Example

See also

IComponentImplementation interface

ICrossSheet interface

Overview

The ICrossSheet interface is a cross sheet connector object interface. Cross sheet connector objects can be used to link a net from a sheet to other sheets within a project. This method defines global connections between sheets within a project. An active cross sheet object is associated with a net.

An equivalent Cross Sheet Connector object representation is the ISch_CrossSheetConnector interface in Schematic API Reference.

Important notes

- ICrossSheet interface is inherited from INetItem interface.

See also

INetItem interface.

ILine Interface

Overview

The ILine interface is a line object interface for an existing line object on a Schematic document. A line is a graphical drawing object with any number of joined segments.

An equivalent Line object representation is the ISch_Line interface in the Schematic API reference.

The **ILine** interface hierarchy is as follows;

IDMObject

ILine

ILine methods

DM_LX

DM_LY

DM_HX

DM_HY

ILine properties

See also

IDMObject interface

Methods

DM_LX method

(ILine interface)

Syntax

```
Function DM_LX : Integer;
```

Description

This function returns the lower left coordinate of the line.

Example

See also

ILine interface

DM_LY method

(ILine interface)

Syntax

```
Function DM_LY : Integer;
```

Description

This function returns the lower left coordinate of the line.

Example

See also

ILine interface

DM_HY method

(ILine interface)

Syntax

```
Function DM_HY : Integer;
```

Description

This function returns the upper right coordinate of the line.

Example

See also

ILine interface

DM_HX method

(ILine interface)

Syntax

```
Function DM_HX : Integer;
```

Description

This function returns the upper right coordinate of the line.

Example

See also

ILine interface

INet interface

Overview

The INet interface is associated with an existing net object of a design document. A net is a series of connections of net identifiers (electrically aware objects such as sheet entries, pins, wires and ports) with the same net name.

That is, all connections sharing the same net name is a net and can be connected on a sheet or between sheets in a project.

INet methods

DM_AllNetItems
DM_RemovedNetItems
DM_Directives
DM_Pins
DM_PowerObjects
DM_Ports
DM_CrossSheetConnectors
DM_NetLabels
DM_SheetEntrys
DM_Lines
DM_SubWires
DM_AllNetItemCount
DM_RemovedNetItemCount
DM_DirectiveCount

INet properties

DM_PinCount
DM_PowerObjectCount
DM_PortCount
DM_CrossSheetConnectorCount
DM_NetLabelCount
DM_SheetEntryCount
DM_LineCount
DM_SubWireCount
DM_Electric
DM_ElectricalString
DM_SignalType
DM_AutoNumber
DM_Scope
DM_CalculatedNetName
DM_HiddenNetName
DM_IsAutoGenerated
DM_IsLocal
DM_NetNumber
DM_NetName
DM_FullNetName
DM_BusRange1
DM_BusRange2
DM_BusRangeValue1
DM_BusRangeValue2
DM_BusIndex
DM_BusWidth
DM_BusKind
DM_IsBusElement
DM_IsBusSection
DM_IsBusMember
DM_RangeDefinedByValue
DM_BusPrefix
DM_CountOfNonPinItems
DM_CountOfElectricalType
DM_SuppressERC
DM_BusSectionParent

See also

Methods

DM_AllNetItemCount method

(INet interface)

Syntax

```
Function DM_AllNetItemCount : Integer;
```

Description

The function returns the number of net aware objects (that is inherited from the INetItem interface).

Example

See also

INet interface

DM_AllNetItems method

(INet interface)

Syntax

```
Function DM_AllNetItems (Index : Integer) : INetItem;
```

Description

The function returns an indexed net aware object. Use the DM_AllNetItemCount function for this function.

Example

See also

INet interface

DM_AllNetItemCount function

DM_AutoNumber method

(INet interface)

Syntax

```
Function DM_AutoNumber : Integer;
```

Description

The function returns the auto number value used for auto-numbering nets.

Example

See also

INet interface

DM_BusIndex method

(INet interface)

Syntax

```
Function DM_BusIndex : Integer;
```

Description

The function returns the bus index. An IBus interface is inherited from a INetItem interface.

Example

See also

INet interface

DM_BusKind method

(INet interface)

Syntax

```
Function DM_BusKind : TBusKind;
```

Description

The function returns the bus kind. which is one of the following enumerated values; eBusKindUndefined, eBusKindLowValueFirst, eBusKindHighValueFirst, eBusKindGeneric.

Example

See also

INet interface

TBusKind type (from RT_Unit in Altium Designer RTL)

DM_BusPrefix method

(INet interface)

Syntax

```
Function DM_BusPrefix : WideString;
```

Description

The function returns the bus prefix as used in this net.

Example

See also

INet interface

DM_BusRange1 method

(INet interface)

Syntax

```
Function DM_BusRange1 : WideString;
```

Description

The function returns the first index of the Bus range. Eg. A[1..6], the bus range1 is 1.

Example

See also

INet interface

DM_BusRange2 method

(INet interface)

Syntax

```
Function DM_BusRange2 : WideString;
```

Description

The function returns the last index of the Bus Range. Eg A[0..4], the bus range 2 is 4.

Example

See also

INet interface

DM_BusRangeValue1 method

(INet interface)

Syntax

```
Function DM_BusRangeValue1 : Integer;
```

Description

The function returns the value of the first index of the Bus range

Example

See also

INet interface

DM_BusRangeValue2 method

(INet interface)

Syntax

```
Function DM_BusRangeValue2 : Integer;
```

Description

The function returns the value of the second index of the Bus range.

Example

See also

INet interface

DM_BusSectionParent method

(INet interface)

Syntax

```
Function DM_BusSectionParent : INet;
```

Description

This function returns an INet interface for the parent bus section.

Example

See also

INet interface

DM_BusWidth method

(INet interface)

Syntax

```
Function DM_BusWidth : Integer;
```

Description

The function returns the bus width. An IBus interface is inherited from a INetItem interface.

Example

See also

INet interface

DM_CalculatedNetName method

(INet interface)

Syntax

```
Function DM_CalculatedNetName : WideString;
```

Description

The function returns the bus width. An IBus interface is inherited from a INetItem interface.

Example

See also

INet interface

DM_CountOfElectricalType method

(INet interface)

Syntax

```
Function DM_CountOfElectricalType (AElectric : TPinElectrical) : Integer;
```

Description

The function returns the number of electrical types used by the current sheet or the project.

Example

See also

INet interface

DM_CountOfNonPinItems method

(INet interface)

Syntax

```
Function DM_CountOfNonPinItems : Integer;
```

Description

The function returns the number of non-pin objects used on the current sheet or the project.

Example

See also

INet interface

DM_CrossSheetConnectorCount method

(INet interface)

Syntax

```
Function DM_CrossSheetConnectorCount : Integer;
```

Description

The function returns the number of cross sheet connectors associated with this net.

Example

See also

INet interface

DM_CrossSheetConnectors method

(INet interface)

Syntax

```
Function DM_CrossSheetConnectors (Index : Integer) : ICrossSheet;
```

Description

The function returns an indexed cross sheet connector that is part of the current net. Use the DM_CrossSheetConnectorCount function.

Example

See also

INet interface

DM_DirectiveCount method

(INet interface)

Syntax

```
Function DM_DirectiveCount : Integer;
```

Description

The function returns the number of directives associated with this net.

Example

See also

INet interface

DM_Directives method

(INet interface)

Syntax

```
Function DM_Directives (Index : Integer) : INetItem;
```

Description

The function returns an indexed directive (which could be a PCB layout directive that contains PCB fules). Use the DM_DirectiveCount function.

Example

See also

INet interface

DM_Electric method

(INet interface)

Syntax

```
Function DM_Electric : TPinElectrical;
```

Description

The function returns the type of electrical property the pin is associated with. Various values include :eElectricInput, eElectricIO, eElectricOutput, eElectricOpenCollector, eElectricPassive, eElectricHiZ, eElectricOpenEmitter, eElectricPower

Example

See also

INet interface

TPinElectrical type

DM_ElectricalString method

(INet interface)

Syntax

```
Function DM_ElectricalString : WideString;
```

Description

The function returns the electrical property associated with this net.

Example

See also

INet interface

DM_FullNetName method

(INet interface)

Syntax

```
Function DM_FullNetName : WideString; {Including Bus Index etc}
```

Description

The function denotes the full net name (includes the bus index and so on).

Example

See also

INet interface

DM_HiddenNetName method

(INet interface)

Syntax

```
Function DM_HiddenNetName : WideString;
```

Description

The function denotes the hidden net name (like power nets).

Example

See also

INet interface

DM_IsAutoGenerated method

(INet interface)

Syntax

```
Function DM_IsAutoGenerated : Boolean;
```

Description

The function denotes a boolean value whether this net has been system generated or not.

Example

See also

INet interface

DM_IsBusElement method

(INet interface)

Syntax

```
Function DM_IsBusElement : Boolean;
```

Description

The function returns a Boolean value whether this bus element exists or not for this INetItem interface. An IBus interface is inherited from a INetItem interface.

Example

See also

INet interface

DM_IsBusMember method

(INet interface)

Syntax

```
Function DM_IsBusMember : Boolean;
```

Description

The function returns a Boolean value whether this bus member exists or not for this INetItem interface. An IBus interface is inherited from a INetItem interface.

Example

See also

INet interface

DM_IsBusSection method

(INet interface)

Syntax

```
Function DM_IsBusSection : Boolean;
```

Description

The function returns a Boolean value whether the bus section exists or not for this INetItem interface. An IBus interface is inherited from a INetItem interface.

Example

See also

INet interface

DM_IsLocal method

(INet interface)

Syntax

```
Function DM_IsLocal : Boolean;
```

Description

The function denotes whether this net is a local net restricted to the document or not.

Example

See also

INet interface

DM_LineCount method

(INet interface)

Syntax

```
Function DM_LineCount : Integer;
```

Description

The function returns the number of lines associated with this net.

Example

See also

INet interface

DM_Lines method

(INet interface)

Syntax

```
Function DM_Lines (Index : Integer) : ILine;
```

Description

The function returns an indexed line that is part of the current net. use the DM_LineCount function.

Example

See also

INet interface

DM_NetLabelCount method

(INet interface)

Syntax

```
Function DM_NetLabelCount : Integer;
```

Description

The function returns the number of net labels associated with this net.

Example

See also

INet interface

INetLabel interface

DM_NetLabels method

(INet interface)

Syntax

```
Function DM_NetLabels (Index : Integer) : INetItem;
```

Description

The function returns an indexed net label that is part of the current net. Use DM_NetLabelCount function.

Example

See also

INet interface

INetItem interface

INetLabel interface

DM_NetName method

(INet interface)

Syntax

```
Function DM_NetName : WideString;
```

Description

The function denotes the net name of this net.

Example

See also

INet interface

DM_NetNumber method

(INet interface)

Syntax

```
Function DM_NetNumber : WideString;
```

Description

The function denotes the net number of this net.

Example

See also

INet interface

DM_PinCount method

(INet interface)

Syntax

```
Function DM_PinCount : Integer;
```

Description

The function returns the number of pins associated with this net.

Example

See also

INet interface

DM_Pins method

(INet interface)

Syntax

```
Function DM_Pins (Index : Integer) : INetItem;
```

Description

The function reurns an indexed pin that is part of the current net. Use the DM_PinCount function.

Example

See also

INet interface

DM_PortCount method

(INet interface)

Syntax

```
Function DM_PortCount : Integer;
```

Description

The function returns the number of ports associated with this net.

Example

See also

INet interface

DM_Ports method

(INet interface)

Syntax

```
Function DM_Ports (Index : Integer) : INetItem;
```

Description

The function returns an indexed port that is part of the current net. Use the DM_PortCount function.

Example

See also

INet interface

DM_PowerObjectCount method

(INet interface)

Syntax

```
Function DM_PowerObjectCount : Integer;
```

Description

The function returns the number of power objects associated with this net.

Example

See also

INet interface

DM_PowerObjects method

(INet interface)

Syntax

```
Function DM_PowerObjects (Index : Integer) : INetItem;
```

Description

The function returns an indexed power object that is part of the current net. Use the DM_PowerObjectCount function.

Example

See also

INet interface

DM_RangeDefinedByValue method

(INet interface)

Syntax

```
Function DM_RangeDefinedByValue : Boolean;
```

Description

The function returns a boolean value whether the range has been defined by a two specific range values or not.

Example

See also

INet interface

DM_RemovedNetItemCount method

(INet interface)

Syntax

```
Function DM_RemovedNetItemCount : Integer;
```

Description

The function returns the number of net items that have been removed from the nets.

Example

See also

INet interface

DM_RemovedNetItems method

(INet interface)

Syntax

```
Function DM_RemovedNetItems (Index : Integer) : INetItem;
```

Description

The function returns an indexed net item that has been removed from net of the schematic document.

Example

See also

INet interface

DM_Scope method

(INet interface)

Syntax

```
Function DM_Scope : TNetScope;
```

Description

The funciton denotes the scope of this net as defined by the TNetScope type.

Example

See also

INet interface

TNetScope type

DM_SheetEntryCount method

(INet interface)

Syntax

```
Function DM_SheetEntryCount : Integer;
```

Description

The function returns the number of sheet entries associated with this net.

Example

See also

INet interface

DM_SheetEntrys method

(INet interface)

Syntax

```
Function DM_SheetEntrys (Index : Integer) : INetItem;
```

Description

The function returns an indexed sheet entry that is part of the current net. Use DM_SheetEntryCount function.

Example

See also

INet interface

INetItem interface

DM_SheetEntryCount function

DM_SignalType method

(INet interface)

Syntax

```
Function DM_SignalType : WideString;
```

Description

The function returns the signal type property associated with this net.

Example

See also

INet interface

DM_SubWireCount method

(INet interface)

Syntax

```
Function DM_SubWireCount : Integer;
```

Description

The function returns the number of sub wires associated with this net.

Example

See also

INet interface

DM_SubWires method

(INet interface)

Syntax

```
Function DM_SubWires (Index : Integer) : INet; {For BusSections only}
```

Description

The function Returns an indexed sub wire (part of a bus object). Use the DM_SubWireCount. A bus object conceptually carries multiple wires.

Example

See also

INet interface

DM_SuppressERC method

(INet interface)

Syntax

```
Function DM_SuppressERC : Boolean;
```

Description

The function returns a boolean value whether the ERC has been suppressed for this net or not.

Example

See also

INet interface

INetClass interface

Overview

The INetClass interface is a PCB Net Class object interface for an existing NetClass on a PCB document. An existing Net class contains members of specific Net objects. Each Net within a NetClass object can either be a member, or not. The 'All Nets' Net Class exists in every PCB file by default; it includes all Nets in the document. It is not possible to change which Nets are members of that Net Class, but the user has full control over which Nets are members of any other Net Classes (which are created and named by the user).

Notes

- An INetClass interface is inherited from the IObjectClass interface.

See also

IObjectClass

INetItem Interface

Overview

The INetItem interface represents the ancestor or parent interface for the following interfaces – IBus, ICrossSheetConnector, IPin, IPort, INetlabel, ISheetEntry and IPowerObject interfaces that have a Net property.

These interface objects have a net property and thus these objects can be part of a net.

INetItem methods

DM_OwnerNetLogical

DM_OwnerNetPhysical

DM_ParentID

DM_Electric

DM_Id

DM_NetName

DM_FlattenedNetName

DM_NetNumber

DM_Electrical

DM_ElectricalString

DM_SignalType

DM_BusRange1

DM_BusRange2

DM_BusRangeValue1

INetItem properties

DM_BusRangeValue2
DM_BusKind
DM_BusIndex
DM_BusWidth
DM_BusPrefix
DM_IsAutoGenerated
DM_IsBusMember
DM_IsBusElement
DM_IsBusSection
DM_RangeDefinedByValue
DM_Part
DM_PartId
DM_DisplayMode
DM_PinName
DM_PinNumber
DM_FullPinName
DM_IsHidden
DM_LogicalPartDesignator
DM_FullLogicalPartDesignator
DM_PhysicalPartDesignator
DM_FullPhysicalPartDesignator
DM_PartUniqueId
DM_PartType
DM_FootPrint
DM_PinNameNoPartId
DM_FullUniqueId
DM_PartSwapId
DM_PinSwapId
DM_SheetSymbol
DM_ParentSheetSymbolSheetName
DM_ParentSheetSymbolName
DM_LinkObject

See also

Methods

DM_BusIndex method

(INetItem interface)

Syntax

```
Function DM_BusIndex : Integer;
```

Description

The function returns the bus index. An IBus interface is inherited from a INetItem interface.

Example

See also

INetItem interface

DM_BusKind method

(INetItem interface)

Syntax

```
Function DM_BusKind : TBusKind;
```

Description

The function returns the type of bus. An IBus interface is inherited from a INetItem interface.

Example

See also

INetItem interface

DM_BusPrefix method

(INetItem interface)

Syntax

```
Function DM_BusPrefix : WideString;
```

Description

The function returns the bus prefix. An example, a bus object could have this A[0..7] net label, and the prefix is A. An IBus interface is inherited from a INetItem interface.

Example

See also

INetItem interface

DM_BusRange1 method

(INetItem interface)

Syntax

```
Function DM_BusRange1 : WideString;
```

Description

This function returns the bus range 1 string.

Example

See also

INetItem interface

DM_BusRange2 method

(INetItem interface)

Syntax

```
Function DM_BusRange2 : WideString;
```

Description

This function returns the bus range 1 string.

Example

See also

INetItem interface

DM_BusRangeValue1 method

(INetItem interface)

Syntax

```
Function DM_BusRangeValue1 : Integer;
```

Description

This function returns value of the bus range 1.

Example

See also

INetItem interface

DM_BusRangeValue2 method

(INetItem interface)

Syntax

```
Function DM_BusRangeValue2 : Integer;
```

Description

This function returns value of the bus range 1.

Example

See also

INetItem interface

DM_BusWidth method

(INetItem interface)

Syntax

```
Function DM_BusWidth : Integer;
```

Description

The function returns the bus width. An IBus interface is inherited from a INetItem interface.

Example

See also

INetItem interface

DM_DisplayMode method

(INetItem interface)

Syntax

```
Function DM_DisplayMode : TDisplayMode;
```

Description

The function returns the display mode for this part object. A part object can have up to 254 alternative graphical displays along with the normal graphical display.

Example

See also

INetItem interface

TDisplayMode type

DM_Electric method

(INetItem interface)

Syntax

```
Function DM_Electric : TPinElectrical;
```

Description

This function denotes the electrical pin property for a net aware object.

Example

See also

INetItem interface

TPinElectrical type

DM_Electrical method

(INetItem interface)

Syntax

```
Function DM_Electrical : TPinElectrical;
```

Description

The function returns the electrical pin property.

Example

See also

INetItem interface

TPinElectrical type

DM_ElectricalString method

(INetItem interface)

Syntax

```
Function DM_ElectricalString : WideString;
```

Description

The function returns the electrical property string.

Example

See also

INetItem interface

DM_FlattenedNetName method

(INetItem interface)

Syntax

```
Function DM_FlattenedNetName : WideString;
```

Description

The function returns the net name of the flattened net where the net aware object is associated with.

Example

See also

INetItem interface

DM_FootPrint method

(INetItem interface)

Syntax

```
Function DM_FootPrint : WideString;
```

Description

The function returns the Footprint string for this INetItem associated with an IPart object.

Example

See also

INetItem interface

DM_FullLogicalPartDesignator method

(INetItem interface)

Syntax

```
Function DM_FullLogicalPartDesignator : WideString;
```

Description

The function returns the logical part designator and the channel instance for this INetItem Interface.

Example

See also

INetItem interface

DM_FullPhysicalPartDesignator method

(INetItem interface)

Syntax

```
Function DM_FullPhysicalPartDesignator : WideString;
```

Description

The function returns the full logical part designator and the channel instance for this INetItem Interface.

Example

See also

INetItem interface

DM_FullPinName method

(INetItem interface)

Syntax

```
Function DM_FullPinName : WideString;
```

Description

The function returns the full Pin name and number that this INetItem interface is associated with. An IPin interface is inherited from an INetItem interface.

Example

See also

INetItem interface

DM_FullUniqueId method

(INetItem interface)

Syntax

```
Function DM_FullUniqueId : WideString;
```

Description

The function returns the full Unique ID string for this INetItem interface.

Example

See also

INetItem interface

DM_Id method

(INetItem interface)

Syntax

```
Function DM_Id : WideString;
```

Description

The function returns the Id value for this net aware object.

Example

See also

INetItem interface

DM_IsAutoGenerated method

(INetItem interface)

Syntax

```
Function DM_IsAutoGenerated : Boolean;
```

Description

The function returns a Boolean value whether this INetItem has been automatically generated by Atlium Designer or not.

Example

See also

INetItem interface

DM_IsBusElement method

(INetItem interface)

Syntax

```
Function DM_IsBusElement : Boolean;
```

Description

The function returns a Boolean value whether this bus element exists or not for this INetItem interface. An IBus interface is inherited from a INetItem interface.

Example

See also

INetItem interface

DM_IsBusMember method

(INetItem interface)

Syntax

```
Function DM_IsBusMember : Boolean;
```

Description

The function returns a Boolean value whether this bus member exists or not for this INetItem interface. An IBus interface is inherited from a INetItem interface.

Example

See also

INetItem interface

DM_IsBusSection method

(INetItem interface)

Syntax

```
Function DM_IsBusSection : Boolean;
```

Description

The function returns a Boolean value whether the bus section exists or not for this INetItem interface. An IBus interface is inherited from a INetItem interface.

Example

See also

INetItem interface

DM_IsHidden method

(INetItem interface)

Syntax

```
Function DM_IsHidden : Boolean;
```

Description

The function returns whether this pin object is hidden or not. An IPin interface is inherited from an INetItem interface.

Example

See also

INetItem interface

DM_LinkObject method

(INetItem interface)

Syntax

```
Function DM_LinkObject : INetItem;
```

Description

The function denotes the linked object to a sheet entry or port from a port or a sheet entry respectively. This method is for port objects that are connected from child schematic sheets to sheet entries of sheet symbols on a parent sheet.

Example

See also

INetItem interface

DM_LogicalPartDesignator method

(INetItem interface)

Syntax

```
Function DM_LogicalPartDesignator : WideString;
```

Description

The function returns the logical part designator for this INetItem interface.

Example

See also

INetItem interface

DM_NetName method

(INetItem interface)

Syntax

```
Function DM_NetName : WideString;
```

Description

The function returns the net name of the net where the net aware object is associated with.

Example

See also

INetItem interface

DM_NetNumber method

(INetItem interface)

Syntax

```
Function DM_NetNumber : WideString;
```

Description

Example

See also

INetItem interface

DM_OwnerNetLogical method

(INetItem interface)

Syntax

```
Function DM_OwnerNetLogical : INet;
```

Description

The function denotes whether this net aware object is associated with the net of a logical document.

Example

See also

INetItem interface

DM_OwnerNetPhysical method

(INetItem interface)

Syntax

```
Function DM_OwnerNetPhysical : INet;
```

Description

The function denotes whether this net aware object is associated with the net of a physical document.

Example

See also

INetItem interface

DM_ParentID method

(INetItem interface)

Syntax

```
Function DM_ParentID : WideString;
```

Description

The function denotes the parent ID or the Sheet document name / Net Name property where this interface is associated with. For example a sheet entry on a sheet symbol object's parent ID is the name of the schematic sheet where the port is.

Example

See also

INetItem interface

DM_ParentSheetSymbolName method

(INetItem interface)

Syntax

```
Function DM_ParentSheetSymbolName : WideString;
```

Description

The function Returns the parent sheet symbol name associated with this INetItem interface (which is a SheetEntry object).

Example

See also

INetItem interface

DM_ParentSheetSymbolSheetName method

(INetItem interface)

Syntax

```
Function DM_ParentSheetSymbolSheetName : WideString;
```

Description

The function returns the parent sheet symbol sheet name string associated with this INetItem interface (which is a sheet entry object).

Example

See also

INetItem interface

DM_Part method

(INetItem interface)

Syntax

```
Function DM_Part : IPart;
```

Description

The function returns the IPart interface.

Example

See also

INetItem interface

DM_PartId method

(INetItem interface)

Syntax

```
Function DM_PartId : Integer;
```

Description

The function returns the Part ID value. A part object is a composite of a multi-part component, and thus each part object is referenced by its Part Id.

Example

See also

INetItem interface

DM_PartSwapId method

(INetItem interface)

Syntax

```
Function DM_PartSwapId : WideString;
```

Description

The function returns the wide string for the part swap Id.

Example

See also

INetItem interface

DM_PartType method

(INetItem interface)

Syntax

```
Function DM_PartType : WideString;
```

Description

The function returns the part type for this INetItem associated with an IPart object.

Example

See also

INetItem interface

DM_PartUniqueId method

(INetItem interface)

Syntax

```
Function DM_PartUniqueId : WideString;
```

Description

The function returns the Unique ID for this part the NetItem is associated with.

Example

See also

INetItem interface

DM_PhysicalPartDesignator method

(INetItem interface)

Syntax

```
Function DM_PhysicalPartDesignator : WideString;
```

Description

The function returns the logical part designator and the channel instance for this INetItem Interface.

Example

See also

INetItem interface

DM_PinName method

(INetItem interface)

Syntax

```
Function DM_PinName : WideString;
```

Description

The function returns the Pin name that this INetItem interface is associated with. Since an IPin interface is inherited from an INetItem interface.

Example

See also

INetItem interface

DM_PinNameNoPartId method

(INetItem interface)

Syntax

```
Function DM_PinNameNoPartId : WideString;
```

Description

The function returns the Pin Name Number and Part ID string for this INetItem associated with an Part object. A pin is part of a part / component.

Example

See also

INetItem interface

DM_PinNumber method

(INetItem interface)

Syntax

```
Function DM_PinNumber : WideString;
```

Description

The function returns the Pin Number that this INetItem interface is associated with. An IPin interface is inherited from an INetItem interface.

Example

See also

INetItem interface

DM_PinSwapId method

(INetItem interface)

Syntax

```
Function DM_PinSwapId : WideString;
```

Description

The function returns the wide string for the pin swap Id.

Example

See also

INetItem interface

DM_RangeDefinedByValue method

(INetItem interface)

Syntax

```
Function DM_RangeDefinedByValue : Boolean;
```

Description

The function returns a Boolean value whether the range is defined by a two specific range values or not.

Example

See also

INetItem interface

DM_SheetSymbol method

(INetItem interface)

Syntax

```
Function DM_SheetSymbol : ISheetSymbol;
```

Description

The function returns the ISheetSymbol interface where this INetItem (representing a ISheetEntry interface if it exists) is associated with. If not, a nil value is returned.

Example

See also

INetItem interface

DM_SignalType method

(INetItem interface)

Syntax

```
Function DM_SignalType : WideString;
```

Description

The function returns the signal type string.

Example

See also

INetItem interface

INetLabel interface

Overview

The INetLabel interface is a net label interface to an existing net label object on the schematic sheet document. A net describes a connection from one component pin, to a second pin, and then to a third pin and so on.

Notes

- The INetLabel interface is inherited from the INetItem interface.
- An equivalent NetLabel object representation is the ISch_NetLabel class in Schematic API Reference.

See also

INetItem interface.

IObjectClass interface

Overview

The IObjectClass interface is the ancestor object class interface for Channel Class, Component Class and Net Class interfaces.

IObjectClass methods

DM_Name

DM_MemberCount

DM_Members

IObjectClass properties

See also

Methods

DM_MemberCount method

(IObjectClass interface)

Syntax

```
Function DM_MemberCount : Integer;
```

Description

The function returns the number of members associated with the object class (one of its descendants ie Channel Class, Component class or Net class).

This method is to be used in conjunction with the DM_Members(index) method.

Example

See also

IObjectClass interface

DM_Members method

(IObjectClass interface)

Syntax

```
Function DM_Members (Index : Integer) : WideString;
```

Description

The function returns the indexed member of the object class (one of its descendants that is, a channel class, component class or a net class).

Example

See also

IObjectClass interface

DM_Name method

(IObjectClass interface)

Syntax

```
Function DM_Name : WideString;
```

Description

The function returns the name of the Object class (one of its descendants ie Channel Class, Component class or Net class)

Example

See also

IObjectClass interface

IParameter interface

Overview

The IParameter interface is a parameter object interface to an existing parameter object on a schematic sheet. There are two types of parameters – system parameters which are owned by a schematic document and parameters owned by certain schematic design objects.

A parameter is a child object of a Parameter Set, Part, Pin, Port, or Sheet Symbol object. A Parameter object has a Name property and Value property which can be used to store information, thus the parameters are a way of defining and associating information and could include strings that identify component manufacturer, date added to the document and also a string for the component's value (e.g. 100K for a resistor or 10PF for a capacitor).

Each parameter has a Unique Id assigned to it. This is used for those parameters that have been added as design rule directives. When transferring the design to the PCB document, any defined rule parameters will be used to generate the relevant design rules in the PCB. These generated rules will be given the same Unique Ids, allowing you to change rule constraints in either schematic or PCB and push the change across when performing a synchronization.

An equivalent object representation is the ISch_Parameter class in the Sch API reference.

Interface Methods

Method	Description

Function DM_Name : WideString;	Denotes the name of the parameter object.
Function DM_ConfigurationName : WideString;	Returns the configuration name, that the parameter object is associated with.
Function DM_Kind : TParameterKind;	Denotes the specific kind that can be assigned to this parameter object. String, Boolean, Integer or float..
Function DM_Value : WideString;	Denotes the value placeholder for this parameter object.
Function DM_RawText : WideString	Returns the raw text for this parameter object.
Function DM_UniqueId : WideString;	Any parameter that is configured as a container for design rule directives need to have a unique ID that will be ported onto the corresponding PCB implementation document.
Function DM_Description : WideString;	Denotes the description of this parameter object.
Function DM_NewName : WideString;	Denotes the New Name for the parameter object, especially when there is an ECO change. You can then compare the original and new names.
Function DM_NewValue : WideString;	Denoes the New Value for the parameter object, especially when there is an ECO change. You can then compare the original and new values.
Function DM_OriginalOwner : IDMObject;	This function returns the interface of the owner object this parameter object is associated with.
Function DM_Visible : Boolean;	Denotes whether this parameter object is visible or not.

IParameter interface

Overview

The IParameter interface is a parameter object interface to an existing parameter object on a schematic sheet. There are two types of parameters – system parameters which are owned by a schematic document and parameters owned by certain schematic design objects.

A parameter is a child object of a Parameter Set, Part, Pin, Port, or Sheet Symbol object. A Parameter object has a Name property and Value property which can be used to store information, thus the parameters are a way of defining and associating information and could include strings that identify component manufacturer, date added to the document and also a string for the component's value (e.g. 100K for a resistor or 10PF for a capacitor).

Each parameter has a Unique Id assigned to it. This is used for those parameters that have been added as design rule directives. When transferring the design to the PCB document, any defined rule parameters will be used to generate the relevant design rules in the PCB. These generated rules will be given the same Unique Ids, allowing you to change rule constraints in either schematic or PCB and push the change across when performing a synchronization.

An equivalent object representation is the ISch_Parameter class in the Sch API reference.

IParameter methods

DM_Name
DM_ConfigurationName
DM_Kind
DM_Value
DM_RawText
DM_UniqueId
DM_Description
DM_NewName
DM_NewValue
DM_OriginalOwner
DM_Visible

IParameter properties

See also

Methods

DM_ConfigurationName method

(IParameter interface)

Syntax

```
Function DM_ConfigurationName : WideString;
```

Description

The function returns the configuration name, that the parameter object is associated with.

Example

See also

IParameter interface

DM_Description method

(IParameter interface)

Syntax

```
Function DM_Description : WideString;
```

Description

The function denotes the description of this parameter object.

Example

See also

IParameter interface

DM_Kind method

(IParameter interface)

Syntax

```
Function DM_Kind : TParameterKind;
```

Description

The function denotes the specific kind that can be assigned to this parameter object. String, Boolean, Integer or float.

Example

See also

IParameter interface

TParameterKind type

DM_Name method

(IParameter interface)

Syntax

```
Function DM_Name : WideString;
```

Description

The function denotes the name of the parameter object.

Example

See also

IParameter interface

DM_NewName method

(IParameter interface)

Syntax

```
Function DM_NewName : WideString;
```

Description

The function denotes the New Name for the parameter object, especially when there is an ECO change. You can then compare the original and new names.

Example

See also

IParameter interface

DM_NewValue method

(IParameter interface)

Syntax

```
Function DM_NewValue : WideString;
```

Description

The function denotes the New Value for the parameter object, especially when there is an ECO change. You can then compare the original and new values.

Example

See also

IParameter interface

DM_OriginalOwner method

(IParameter interface)

Syntax

```
Function DM_OriginalOwner : IDMObject;
```

Description

This function returns the interface of the owner object this parameter object is associated with.

Example

See also

IParameter interface

DM_RawText method

(IParameter interface)

Syntax

```
Function DM_RawText : WideString;
```

Description

The function returns the raw text for this parameter object.

Example

See also

IParameter interface

DM_UniqueId method

(IParameter interface)

Syntax

```
Function DM_UniqueId : WideString;
```

Description

Any parameter that is configured as a container for design rule directives need to have a unique ID that will be ported onto the corresponding PCB implementation document.

The function returns the Unique ID value for the parameter object.

Example

See also

IParameter interface

DM_Value method

(IParameter interface)

Syntax

```
Function DM_Value : WideString;
```

Description

The function denotes the value placeholder for this parameter object.

Example

See also

IParameter interface

DM_Visible method

(IParameter interface)

Syntax

```
Function DM_Visible : Boolean;
```

Description

The function denotes whether this parameter object is visible or not.

Example

See also

IParameter interface

IPart interface

Overview

The **IPart** interface is the interface of an existing schematic part on a Schematic sheet. A part object is “part” of a component, that is, a multi-part component consists of part objects. For example a multiple gate integrated circuit has duplicate gates, and that a component represents the multi-part gate and a part represents the gate itself.

An equivalent component object representation is the **ISch_Component** class in Schematic API. The **ISch_Component** interface represents a component that can contain links to different model implementations such as PCB, Signal Integrity and Simulation models. Only one model of a particular model type (PCB footprint, SIM, SI, EDIF Macro and VHDL) can be enabled as the currently linked model, at any one time.

IPart methods

DM_AddConfigurationParameters

DM_AssignedDesignator

DM_CalculatedDesignator

DM_CenterLocationX

DM_CenterLocationY

IPart properties

DM_ChannelOffset
DM_ChildProjectSheet
DM_ChildVHDLEntity
DM_Comment
DM_ComponentKind
DM_ConfiguratorName
DM_CurrentImplementation
DM_Description
DM_DesignatorLocationX
DM_DesignatorLocationY
DM_DesignatorLocked
DM_DisplayMode
DM_FirstPinLocationX
DM_FirstPinLocationY
DM_Footprint
DM_FullLogicalDesignator
DM_FullPhysicalDesignator
DM_Height
DM_ImplementationCount
DM_Implementations
DM_InstanceCount
DM_Layer
DM_LibraryReference
DM_LogicalDesignator
DM_LogicalOwnerDocument
DM_MaxPartCount
DM_NewDesignator
DM_NewPartId
DM_PartID
DM_PartIdLocked
DM_PartType
DM_PhysicalDesignator
DM_PinCount
DM_Pins
DM_ReferenceLocationX
DM_ReferenceLocationY
DM_Rotation
DM_SourceDesignator
DM_SourceHierarchicalPath
DM_SourceLibraryName
DM_SourceUniqueId
DM_SubProject
DM_UniqueId
DM_UniqueIdName
DM_UniqueIdPath

See also

Methods

DM_AddConfigurationParameters method

(IPart interface)

Syntax

```
Procedure DM_AddConfigurationParameters;
```

Description

The procedure adds configuration parameters to this part object.

Example

See also

IPart interface

DM_AssignedDesignator method

(IPart interface)

Syntax

```
Function DM_AssignedDesignator : WideString;
```

Description

The function denotes the assigned designator for this part which is equivalent to the DM_CalculatedDesignator method. The DM_AssignedDesignator method returns a string that contains the designator and multi channel information but does not include multi part id information.

This function returns the calculated designator string which contains the hierarchical path and the logical designator strings. Only when a project is compiled and up to date, designators of parts are calculated based on the compiled documents they are on.

Example

See also

IPart interface

DM_CalculatedDesignator method

DM_CalculatedDesignator method

(IPart interface)

Syntax

```
Function DM_CalculatedDesignator : WideString;
```

Description

The function denotes the system compiled designator for this part. The assigned designator for this part is equivalent to the DM_CalculatedDesignator method. A DM_CalculatedDesignator method returns a string that contains the designator and multi channel information but does not include multi part information.

This function returns the calculated designator string which contains the hierarchical path and the logical designator strings. Only when a project is compiled and up to date, designators of parts are calculated based on the compiled documents they are on.

Example

See also

IPart interface

DM_CenterLocationX method

(IPart interface)

Syntax

```
Function DM_CenterLocationX : Integer;
```

Description

The function returns the central location X of the designator associated with this component.

Example

See also

IPart interface

DM_CenterLocationY method

(IPart interface)

Syntax

```
Function DM_CenterLocationY : Integer;
```

Description

The function returns the central location Y of the designator associated with this component.

Example

See also

IPart interface

DM_ChannelOffset method

(IPart interface)

Syntax

```
Function DM_ChannelOffset : Integer;
```

Description

The offset represents which part is offset in relation to the reference channel and the associated channels are also affected. The function returns the ChannelOffset value.

Example

See also

IPart interface

DM_ChildProjectSheet method

(IPart interface)

Syntax

```
Function DM_ChildProjectSheet : IDocument;
```

Description

The function denotes the IDocument interface representing the child project sheet associated with this part.

Example

See also

IPart interface

DM_ChildVHDLEntity method

(IPart interface)

Syntax

```
Function DM_ChildVHDLEntity : WideString;
```

Description

The function returns the Child VHDL entity string representing the VHDL document that the part (component) is linked to.

Example

See also

IPart interface

DM_Comment method

(IPart interface)

Syntax

```
Function DM_Comment : WideString;
```

Description

The function denotes the comment string for this part.

Example

See also

IPart interface

DM_ComponentKind method

(IPart interface)

Syntax

```
Function DM_ComponentKind : TComponentKind;
```

Description

The function denotes the component kind that this part is represented as. in the BOM and are maintained during synchronization.

A component kind can be one of the following:

- eComponentKind_Standard : These components possess standard electrical properties, are always synchronized and are the type most commonly used on a board.
- eComponentKind_Mechanical: These components do not have electrical properties and will appear in the BOM. They are synchronized if the same components exist on both the Schematic and PCB documents. An example is a heatsink.
- eComponentKind_Graphical: These components are not used during synchronization or checked for electrical errors. These components are used, for example, when adding company logos to documents.
- eComponentKind_NetTie_BOM: These components short two or more different nets for routing and these components will appear.
- eComponentKind_NetTie_NoBOM: These components short two or more different nets for routing and these components will NOT appear in the BOM and are maintained during synchronization.
- unit.

Example

See also

IPart interface

TComponentKind type

DM_CurrentImplementation method

(IPart interface)

Syntax

```
Function DM_CurrentImplementation (AType : WideSTring) : IComponentImplementation;
```

Description

The function returns the current implementation which is usually a PCB footprint.

Example

See also

IPart interface

IComponentImplementation interface

DM_Description method

(IPart interface)

Syntax

```
Function DM_Description : WideString;
```

Description

The function denotes the description of the reference link to a source component or as a device name.

Example

See also

IPart interface

DM_DesignatorLocationX method

(IPart interface)

Syntax

```
Function DM_DesignatorLocationX : Integer;
```

Description

The function returns the location X of the designator associated with this component.

Example

See also

IPart interface

DM_DesignatorLocationY method

(IPart interface)

Syntax

```
Function DM_DesignatorLocationY : Integer;
```

Description

The function returns the location Y of the designator associated with this component.

Example

See also

IPart interface

DM_DesignatorLocked method

(IPart interface)

Syntax

```
Function DM_DesignatorLocked : Boolean;
```

Description

The function denotes whether or not the designator string is locked (unmoveable).

Example

See also

IPart interface

DM_DisplayMode method

(IPart interface)

Syntax

```
Function DM_DisplayMode : TDisplayMode;
```

Description

The function Denotes one of the 255 display modes. The mode 0 is the normal graphical display for this part object. The other 254 modes are alternative graphical displays of this same part object.

Example

See also

IPart interface

DM_FirstPinLocationX method

(IPart interface)

Syntax

```
Function DM_FirstPinLocationX : Integer;
```

Description

The function denotes the reference X location of the first pin of a part object.

Example

See also

IPart interface

DM_FirstPinLocationY method

(IPart interface)

Syntax

```
Function DM_FirstPinLocationY : Integer;
```

Description

The function denotes the reference Y location of the first pin of a part object.

Example

See also

IPart interface

DM_Footprint method

(IPart interface)

Syntax

```
Function DM_Footprint : WideString;
```

Description

The function denotes the footprint string that this part is associated with.

Example

See also

IPart interface

DM_FullLogicalDesignator method

(IPart interface)

Syntax

```
Function DM_FullLogicalDesignator : WideString;
```

Description

The function denotes the full logical designator which includes the part designator and part id information.

Example

See also

IPart interface

DM_FullPhysicalDesignator method

(IPart interface)

Syntax

```
Function DM_FullPhysicalDesignator : WideString;
```

Description

The function denotes the full physical designator of a part which includes the calculated designator and the part id information on compiled schematic sheets.

Example

See also

IPart interface

DM_Height method

(IPart interface)

Syntax

```
Function DM_Height : Integer;
```

Description

The function denotes the height property of the part object. A part object is “part” of a multi-part component.

Example

See also

IPart interface

DM_ImplementationCount method

(IPart interface)

Syntax

```
Function DM_ImplementationCount : Integer;
```

Description

The function returns the number of implementations of this schematic component.

Example

See also

IPart interface

DM_Implementations method

(IPart interface)

Syntax

```
Function DM_Implementations (Index : Integer) : IComponentImplementation;
```

Description

The function returns the particular IComponentImplementation for the specified indexed implementations of a Schematic component.

Example

See also

IPart interface

DM_InstanceCount method

(IPart interface)

Syntax

```
Function DM_InstanceCount : Integer;
```

Description

The function returns the number of instances of this part.

Example

See also

IPart interface

DM_Layer method

(IPart interface)

Syntax

```
Function DM_Layer : WideString;
```

Description

The function denotes which layer this part is on.

Example

See also

IPart interface

DM_LibraryReference method

(IPart interface)

Syntax

```
Function DM_LibraryReference : WideString;
```

Description

The function denotes the name of the component from the library

Example

See also

IPart interface

DM_LogicalDesignator method

(IPart interface)

Syntax

```
Function DM_LogicalDesignator : WideString;
```

Description

The function denotes the logical designator of this part on a schematic sheet.

Example

See also

IPart interface

DM_LogicalOwnerDocument method

(IPart interface)

Syntax

```
Function DM_LogicalOwnerDocument : IDocument;
```

Description

The function denotes the IDocument representing the logical owner document that this part is associated to a schematic component.

Example

See also

IPart interface

DM_MaxPartCount method

(IPart interface)

Syntax

```
Function DM_MaxPartCount : Integer;
```

Description

The function returns the maximum part count for this part object.

Example

See also

IPart interface

DM_NewDesignator method

(IPart interface)

Syntax

```
Function DM_NewDesignator : WideString;
```

Description

The function denotes the new designator for this part.

Example

See also

IPart interface

DM_NewPartId method

(IPart interface)

Syntax

```
Function DM_NewPartId : Integer;
```

Description

The function denotes the new part id for this part.

Example

See also

IPart interface

DM_PartID method

(IPart interface)

Syntax

```
Function DM_PartID : Integer;
```

Description

The function denotes the PartID for this part. A multi-part component references each part by its PartID, for example a four part component has four unique PartIDs.

Example

See also

IPart interface

DM_PartIdLocked method

(IPart interface)

Syntax

```
Function DM_PartIdLocked : Boolean;
```

Description

The function denotes whether or not the part id string is locked (unmoveable).

Example

See also

IPart interface

DM_PartType method

(IPart interface)

Syntax

```
Function DM_PartType : WideString;
```

Description

The function denotes the part type for this part. (Footprint type).

Example

See also

IPart interface

DM_PhysicalDesignator method

(IPart interface)

Syntax

```
Function DM_PhysicalDesignator : WideString;
```

Description

The function denotes the physical designator string of a part. Note, a logical designator doesn't include the channel instance string.

Example

See also

IPart interface

DM_PinCount method

(IPart interface)

Syntax

```
Function DM_PinCount : Integer;
```

Description

The function returns the number of pins for this schematic component.

Example

See also

IPart interface

DM_Pins method

(IPart interface)

Syntax

```
Function DM_Pins (Index : Integer) : INetItem;
```

Description

The function returns the INetItem interface for the specified indexed Pin of a Schematic Component.

Example

See also

IPart interface

DM_ReferenceLocationX method

(IPart interface)

Syntax

```
Function DM_ReferenceLocationX : Integer;
```

Description

The function returns the reference location X of the designator associated with this component.

Example

See also

IPart interface

DM_ReferenceLocationY method

(IPart interface)

Syntax

```
Function DM_ReferenceLocationY : Integer;
```

Description

The function returns the reference location Y of the designator associated with this component.

Example

See also

IPart interface

DM_Rotation method

(IPart interface)

Syntax

```
Function DM_Rotation : Double;
```

Description

The function denotes the rotation property of a part (orientation) in degrees.

Example

See also

IPart interface

DM_SourceDesignator method

(IPart interface)

Syntax

```
Function DM_SourceDesignator : WideString;
```

Description

The function denotes the current designator of the source component from the corresponding schematic.

Example

See also

IPart interface

DM_SourceHierarchicalPath method

(IPart interface)

Syntax

```
Function DM_SourceHierarchicalPath : WideString;
```

Description

The function denotes the source reference path to the PCB component. The path can be multi level depending on whether it is a multi channel or a bormal design. When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library path names.

Example

See also

IPart interface

DM_SourceLibraryName method

(IPart interface)

Syntax

```
Function DM_SourceLibraryName : WideString;
```

Description

The function denotes the name of the source library where the schematic component and its associated part come from.

Example

See also

IPart interface

DM_SourceUniqueId method

(IPart interface)

Syntax

```
Function DM_SourceUniqueId : WideString;
```

Description

Unique IDs (UIDs) are used to match each schematic component to the corresponding PCB component. When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library pathnames. The UID is a system generated value that uniquely identifies the source component.

The function returns the UniqueID for this part.

Example

See also

IPart interface

DM_SubProject method

(IPart interface)

Syntax

```
Function DM_SubProject : WideString;
```

Description

The function returns the sub project string of this part. A part can represent a schematic sheet, like a sheet symbol.

Example

See also

IPart interface

DM_UniqueId method

(IPart interface)

Syntax

```
Function DM_UniqueId : WideString;
```

Description

The function denotes the Unique ID for this part. Unique IDs are used in Schematic – PCB documents synchronization so that Sch components and its corresponding PCB components are in sync.

Example

See also

IPart interface

DM_UniqueIdName method

(IPart interface)

Syntax

```
Function DM_UniqueIdName : WideString;
```

Description

The function denotes the Unique ID name of this part.

Example

See also

IPart interface

DM_UniqueIdPath method

(IPart interface)

Syntax

```
Function DM_UniqueIdPath : WideString;
```

Description

The function denotes the Unique ID path of this part (includes the back slash).

Example

See also

IPart interface

IPin interface

Overview

The IPin interface is a pin object interface to an existing pin object on the schematic. Pins are special objects that have electrical characteristics and are used to direct signals in and out of components. Pins connect directly to other pins, wires, net labels, sheet entries or ports.

Notes

- The IPin interface is inherited from the INetItem interface.
- The pins are part of a schematic component, thus if you wish to have access to the pins, invoke the DM_Pins and DM_PinCount method call from the part object interface.
- An equivalent Pin object representation is the ISch_Pin interface in Schematic API Reference

Example

```
For J := 0 to Doc.DM_ComponentCount - 1 Do
Begin
    Comp := Doc.DM_Components(J);
    //Comp.DM_Footprint;
    //Comp.DM_Comment;
    For K := 0 to Comp.DM_PinCount - 1 Do
    Begin
        Pin := Comp.DM_Pins(K);
        PinName := Pin.DM_PinNumber;
        // Check for parts of a multi-part component that are not used in the project
        // then add 'No Net' for unused pins...
        If Pin.DM_FlattenedNetName = '?' Then
           // these pins of the part is not used on the schematic.
    End;
End;
```

See also

INetItem interface

IPort interface

Overview

The IPort interface is a port object interface to an existing port object on the schematic. A port is used to connect a net on one sheet to Ports with the same name on other sheets. Ports can also connect from child sheets to Sheet entries, in the appropriate sheet symbol on the parent sheet.

Notes

- The IPort interface is inherited from the INetItem interface.
- An equivalent Port object representation is the ISch_Port class in Schematic API Reference.

Example

```
Var
    DM_Port        : IPort;
    I              : Integer;
    S              : TDynamicString;
    ServerDocument : IServerDocument;
Begin
    If ADM_Document = Nil Then Exit;
    If Not ADM_Document.DM_ValidForNavigation Then Exit;

    S := ADM_Document.DM_FullPath;
    ServerDocument := Client.GetDocumentByPath(PChar(S));
    If ServerDocument = Nil Then Exit;

    If Not StringsEqual(TDynamicString(ServerDocument.Kind), 'Sch') Then Exit;

    For i := 0 To ADM_Document.DM_PortCount - 1 Do
    Begin
        DM_Port := ADM_Document.DM_Ports(i);
        If DM_Port <> Nil Then
```

```
          If DM_Port.DM_ValidForNavigation Then
        Begin
           // port is available for manipulation here.
        End;
    End;
End;
```

See also

INetItem interface

IPowerObject interface

Overview

The IPowerObject interface is a power object interface to an existing power object on the schematic. Power ports are special symbols that represent a power supply and are always identified by their net names.

Notes

- The IPowerObject interface is inherited from the INetItem interface.
- An equivalent PowerObject object representation is the ISch_PowerObject class in Sch API Reference.

See also

INetItem interface.

IRoom interface

Overview

The IRoom interface is a PCB room object. A room is controlled by the room design rule. This room serves as a boundary constraint for a group of specified components as a component or channel class.

Interface Methods

Method	Description
Function DM_LX : Integer;	Returns the lower X coordinate of the room object.
Function DM_LY : Integer;	Returns the lower Y coordinate of the room object.
Function DM_HX : Integer;	Returns the higher X coordinate of the room object.
Function DM_HY : Integer;	Returns the higher Y coordinate of the room object.
Function DM_RoomName : WideString;	Returns the name of this room object.
Function DM_Scope1Expression : WideString;	Returns the scope 1 expression which describes the scope of this room object.
Function DM_Layer : Integer;	Returns the PCB layer where the room resides on.

IRoom Interface

Overview

The IRoom interface is a PCB room object. A room is controlled by the room design rule. This room serves as a boundary constraint for a group of specified components as a component or channel class.

IRoom methods

DM_LX

DM_LY

DM_HX

IRoom properties

DM_HY
DM_RoomName
DM_Scope1Expression
DM_Layer

See also

Methods

DM_HX method

(IRoom interface)

Syntax

```
Function DM_HX : Integer;
```

Description

Returns the higher X coordinate of the room object.

Example

See also

IRoom interface

DM_HY method

(IRoom interface)

Syntax

```
Function DM_HY : Integer;
```

Description

Returns the higher Y coordinate of the room object.

Example

See also

IRoom interface

DM_Layer method

(IRoom interface)

Syntax

```
Function DM_Layer : Integer;
```

Description

Returns the PCB Layer value of the room object that it is on.

Example

See also

IRoom interface

DM_LX method

(IRoom interface)

Syntax

```
Function DM_LX : Integer;
```

Description

Returns the lower X coordinate of the room object.

Example

See also

IRoom interface

DM_LY method

(IRoom interface)

Syntax

```
Function DM_LY : Integer;
```

Description

Returns the lower Y coordinate of the room object.

Example

See also

IRoom interface

DM_RoomName method

(IRoom interface)

Syntax

```
Function DM_RoomName : WideString;
```

Description

The function returns the room name.

Example

See also

IRoom interface

DM_Scope1Expression method

(IRoom interface)

Syntax

```
Function DM_Scope1Expression : WideString;
```

Description

The function returns the scope 1 expression which describes the scope of this room object.

Example

See also

IRoom interface

IRule Interface

Overview

The IRule interface represents the one of the rules attached to a parameter within the PCB Layout directive (as a Parameter Set object with a small flag symbol) on a net aware object on a schematic object. A parameter set object can be placed on the schematic sheet by the Place » Directives » PCB Layout menu item.

This PCB Layout directive allows you to assign PCB layout information to a net in the schematic. When a PCB is created from the schematic, the information in the PCB layout directive is used to create relevant PCB design rules.

IRule methods

DM_RuleKind

DM_Scope1Expression

IRule properties

DM_Scope2Expression
DM_MaxWidth
DM_MinWidth
DM_PreferedWidth
DM_ViaHole
DM_ViaWidth
DM_MinViaHole
DM_MaxViaHole
DM_MinViaWidth
DM_MaxViaWidth
DM_ViaStyle
DM_Topology
DM_Priority
DM_RoutingLayers
DM_Attributes
DM_Description
DM_RuleName
DM_UniqueId

See also

Methods

DM_Attributes method

(IRule interface)

Syntax

```
Function DM_Attributes : WideString;
```

Description

The function denotes the attributes of the IRule interface.

Example

See also

IRule interface

DM_Description method

(IRule interface)

Syntax

```
Function DM_Description : WideString;
```

Description

The function denotes the description of this IRule interface.

Example

See also

IRule interface

DM_MaxViaHole method

(IRule interface)

Syntax

```
Function DM_MaxViaHole : Integer;
```

Description

The function denotes the max Via Hole rule property of a Routing Via style PCB Rule.

Example

See also

IRule interface

DM_MaxViaWidth method

(IRule interface)

Syntax

```
Function DM_MaxViaWidth : Integer;
```

Description

The function denotes the max Via width rule property of a Routing Via style PCB Rule.

Example

See also

IRule interface

DM_MaxWidth method

(IRule interface)

Syntax

```
Function DM_MaxWidth : Integer;
```

Description

The function denotes the Maximum Width rule property of a PCB rule.

Example

See also

IRule interface

DM_MinViaHole method

(IRule interface)

Syntax

```
Function DM_MinViaHole : Integer;
```

Description

The function denotes the min Via Hole rule property of a Routing Via style PCB Rule.

Example

See also

IRule interface

DM_MinViaWidth method

(IRule interface)

Syntax

```
Function DM_MinViaWidth : Integer;
```

Description

The function denotes the min Via width rule property of a Routing Via style PCB Rule.

Example

See also

IRule interface

DM_MinWidth method

(IRule interface)

Syntax

```
Function DM_MinWidth : Integer;
```

Description

The function denotes the Minimum Width rule property of a PCB Rule.

Example

See also

IRule interface

DM_PreferedWidth method

(IRule interface)

Syntax

```
Function DM_PreferedWidth : Integer;
```

Description

The function denotes the preferred Width rule property of a PCB Rule.

Example

See also

IRule interface

DM_Priority method

(IRule interface)

Syntax

```
Function DM_Priority : Integer;
```

Description

The function denotes the priority of the PCB Design Rule. The priority value of 1 denotes the highest priority.

Example

See also

IRule interface

DM_RoutingLayers method

(IRule interface)

Syntax

```
Function DM_RoutingLayers (IndexLayer : Integer) : Integer;
```

Description

The function denotes the indexed routing layer rule property (Top layer, Mid1-Mid30, Bottom Layer) of a Routing Layers PCB rule.

Example

See also

IRule interface

DM_RuleKind method

(IRule interface)

Syntax

```
Function DM_RuleKind : Integer;
```

Description

The function denotes the type of PCB Rule.

Example

See also

IRule interface

DM_RuleName method

(IRule interface)

Syntax

```
Function DM_RuleName : WideString;
```

Description

The function denotes the name of this IRule interface representing a PCB rule.

Example

See also

IRule interface

DM_Scope1Expression method

(IRule interface)

Syntax

```
Function DM_Scope1Expression : WideString;
```

Description

The function denotes the first scope expression string. The scope of Design rules are determined by the defined boundary or objects.

Example

See also

IRule interface

DM_Scope2Expression method

(IRule interface)

Syntax

```
Function DM_Scope2Expression : WideString;
```

Description

The function denotes the second scope expression string. The scope of Design rules are determined by the defined boundary or objects.

Example

See also

IRule interface

DM_Topology method

(IRule interface)

Syntax

```
Function DM_Topology : Integer;
```

Description

The function Denotes the topology (Shortest, Horizontal, Vertical, Daisy-Simple, Daisy-MidDriven, Daisy-Balanced and Daisy-StarBurst) rule property of a Routing Topology PCB Rule.

Example

See also

IRule interface

DM_UniqueId method

(IRule interface)

Syntax

```
Function DM_UniqueId : WideString;
```

Description

Each rule has a Unique ID assigned so that when Schematic and PCB documents are synchronized, the ECO knows which rules to update or apply to/from.

Example

See also

IRule interface

DM_ViaHole method

(IRule interface)

Syntax

```
Function DM_ViaHole : Integer;
```

Description

Denotes the Via Hole rule property of a Routing Via style PCB Rule.

Example

See also

IRule interface

DM_ViaStyle method

(IRule interface)

Syntax

```
Function DM_ViaStyle : Integer;
```

Description

This function denotes the via style rule property of a Routing Via style PCB Rule.

Example

See also

IRule interface

DM_ViaWidth method

(IRule interface)

Syntax

```
Function DM_ViaWidth : Integer;
```

Description

The function denotes the Via width rule property of a Routing Via style PCB Rule.

Example

See also

IRule interface

ISheetSymbol Interface

Overview

The ISheetSymbol interface is a sheet symbol interface to an existing sheet symbol object on the schematic. Sheet symbols represent other schematic sheets (often referred to as a child sheet). The link between a sheet symbol and other schematic sheets is the FileName attribute, which must be the same as the name of the child sheet.

An equivalent Sheet Symbol object representation is the ISch_SheetSymbol class in Sch API Reference.

ISheetSymbol methods

DM_SheetEntries

DM_SheetEntryCount

DM_ChildSheet

DM_ChildSheetCount

DM_SheetSymbolFileName

DM_LogicalDesignator

DM_CalculatedDesignator

DM_PhysicalDesignator

DM_UniqueId

ISheetSymbol properties

See also

Methods

DM_CalculatedDesignator method

(ISheetSymbol interface)

Syntax

```
Function DM_CalculatedDesignator : WideString;
```

Description

This function returns the calculated designator string which contains the hierarchical path and the logical designator strings. Only when a project is compiled and up to date, designators of sheet symbols are calculated based on the physical documents they are on.

Example

See also

ISheetSymbol interface

DM_ChildSheet method

(ISheetSymbol interface)

Syntax

```
Function DM_ChildSheet (Index : Integer) : IDocument;
```

Description

Returns the indexed child sheet associated with this sheet symbol object. Use in conjunction with the DM_ChildSheetCount method.

Example

See also

ISheetSymbol interface

DM_ChildSheetCount method

(ISheetSymbol interface)

Syntax

```
Function DM_ChildSheetCount : Integer;
```

Description

Returns the number of child sheets associated with this sheet symbol object.

Example

See also

ISheetSymbol interface

DM_LogicalDesignator method

(ISheetSymbol interface)

Syntax

```
Function DM_LogicalDesignator : WideString;
```

Description

Returns the logical designator of this sheet symbol. A logical designator is not unique, since logical designators are used in multi channel designs.

Example

See also

ISheetSymbol interface

DM_PhysicalDesignator method

(ISheetSymbol interface)

Syntax

```
Function DM_PhysicalDesignator : WideString;
```

Description

Returns the designator of this sheet symbol. Every physical designator is unique.

Example

See also

ISheetSymbol interface

DM_SheetEntries method

(ISheetSymbol interface)

Syntax

```
Function DM_SheetEntries (Index : Integer) : INetItem;
```

Description

Returns the number of sheet entries that are associated with this sheet symbol. Since a sheet entry is of a INetItem type, thus a INetItem interface is returned.

Example

See also

ISheetSymbol interface

DM_SheetEntryCount method

(ISheetSymbol interface)

Syntax

```
Function DM_SheetEntryCount : Integer;
```

Description

Returns the number of sheet entries associated with this sheet symbol object.

Example

See also

ISheetSymbol interface

DM_SheetSymbolFileName method

(ISheetSymbol interface)

Syntax

```
Function DM_SheetSymbolFileName : WideString;
```

Description

Returns the filename which is a link between this sheet symbol object and the other schematic sheet.

Example

See also

ISheetSymbol interface

DM_UniqueId method

(ISheetSymbol interface)

Syntax

```
Function DM_UniqueId : WideString;
```

Description

Returns the unique ID of this sheet symbol object.

Example

See also

ISheetSymbol interface

ISheetEntry interface

Overview

The **ISheetEntry** interface is a sheet entry object interface to an existing sheet entry object on the schematic. A sheet entry creates a connection between the net touching on the parent sheet, to a Port with the same name on the child sheet.

Notes

- The **ISheetEntry** interface is inherited from the **INetItem** interface.
- An equivalent SheetEntry object representation is the **ISch_SheetEntry** class in Sch API Reference.

See also

INetItem interface.

ITextFrame Interface

Overview

The ITextFrame interface is a text frame object for an existing text frame on a schematic document. It is a container holding lines of text like a memo.

An equivalent TextFrame object representation is the ISch_TextFrame interface in the Schematic API reference.

ITextFrame methods **ITextFrame properties**

DM_Text

See also

Methods

DM_Text method

(ITextFrame interface)

Syntax

```
Function DM_Text : WideString;
```

Description

This function returns the text string from this current TextFrame object.

Example

See also

ITextFrame interface

IViolation Interface

Overview

The IViolation interface represents a violation object on a design document in the Workspace Manager of Altium Designer.

IViolation methods

DM_ErrorKind
DM_ErrorLevel
DM_CompilationStage
DM_AddRelatedObject
DM_RelatedObjectCount
DM_RelatedObjects
DM_DescriptorString
DM_DetailString

IViolation properties

See also

Methods

DM_AddRelatedObject method

(IViolation interface)

Syntax

```
Procedure DM_AddRelatedObject (AnObject : IDMObject);
```

Description

This procedure adds the object that is part of the violation.

Example

See also

IViolation interface

DM_CompilationStage method

(IViolation interface)

Syntax

```
Function DM_CompilationStage : TCompilationStage;
```

Description

This function returns the status of the complation stage: during compilation or during flattening process.

Example

See also

IViolation interface

DM_DescriptorString method

(IViolation interface)

Syntax

```
Function DM_DescriptorString : WideString;
```

Description

This function returns the description string for this violation interface.

Example

See also

IViolation interface

DM_DetailString method

(IViolation interface)

Syntax

```
Function DM_DetailString : WideString;
```

Description

This function returns the detailed description stirng of this violation interface.

Example

See also

IViolation interface

DM_ErrorKind method

(IViolation interface)

Syntax

```
Function DM_ErrorKind : TErrorKind;
```

Description

Returns the kind of error this violation has been assigned to.

Example

See also

IViolation interface

DM_ErrorLevel method

(IViolation interface)

Syntax

```
Function DM_ErrorLevel : TErrorLevel;
```

Description

Returns the level of error this violation has been assigned to. Various error levels include : eErrorLevelNoReport,eErrorLevelWarning,eErrorLevelError,eErrorLevelFatal

Example

See also

IViolation interface

TErrorLevel type

DM_RelatedObjectCount method

(IViolation interface)

Syntax

```
Function DM_RelatedObjectCount : Integer;
```

Description

This function returns the number of related objects of the violation.

Example

See also

IViolation interface

DM_RelatedObjects method

(IViolation interface)

Syntax

```
Function DM_RelatedObjects (Index : Integer) : IDMObject;
```

Description

This function returns the indexed related object of the violation.

Example

See also

IViolation interface

IWrapper Interface

Overview

The **IWrapper** interface hierarchy is as follows;

IWrapper methods	IWrapper properties
DM_GroupedSchObjects_Count	
DM_GroupedSchObject	

See also

Methods

DM_GroupedSchObject method

(IWrapper interface)

Syntax

```
Function DM_GroupedSchObject(i : Integer) : IWrapper;End;
```

Description

Example

See also

IWrapper interface

DM_GroupedSchObjects_Count method

(IWrapper interface)

Syntax

```
Function DM_GroupedSchObjects_Count : Integer;
```

Description

Example

See also

IWrapper interface

Signals Manager interfaces

IEntityPort interface

Overview

Important notes

- Inherited from ISignalNode interface

Interface Methods

- All methods from ISignalNode interface.

See also

Workspace Manager Interfaces

ISignalManager interface

ISignalNode interface

IExternalParameter interface

Overview

The IExternalParameter interface defines the external parameter object

Interface Methods

Method	Description
Function DM_GetSection : WideString;	Returns the Section string of the external parameter interface.
Function DM_GetName : WideString;	Returns the Name string of the external parameter interface.
Function DM_GetValue : WideString;	Returns the Value string of the external parameter interface.
Procedure DM_SetValue(AValue : WideString);	Sets the new value string for this external parameter.

IInstance interface

Overview

Interface Methods

```
Function    DM_Part         : IPart;
```

```
Function    DM_SheetSymbol  : ISheetSymbol;
Function    DM_Ports (Index : Integer) :  IInstancePort;
Function    DM_PortCount    : Integer;
Function    DM_Designator   : WideString;
Function    DM_InstanceType : WideString;
```

See also

Workspace Manager Interfaces

ISignalManager interface

IPart interface

ISheetSymbol interface

IInstancePort interface

IInstancePort interface

Overview

Important notes

- Inherited from ISignalNode interface

Interface Methods

- All methods from ISignalNode interface.

See also

Workspace Manager Interfaces

ISignalManager interface

ISignalNode interface

ISignal interface

Overview

Interface Methods

```
Function    DM_Namers     (Index : Integer) : ISignalNode;
Function    DM_SubNets    (Index : Integer) : ISubNet;
Function    DM_DriverLinks(Index : Integer) : ISignalLink;
Function    DM_TargetLinks(Index : Integer) : ISignalLink;
Function    DM_NamerCount      : Integer;
Function    DM_SubNetCount     : Integer;
Function    DM_DriverLinkCount : Integer;
Function    DM_TargetLinkCount : Integer;
Function    DM_DriverBits (BitNo,Index : Integer) : ISignalNode;
Function    DM_TargetBits (BitNo,Index : Integer) : ISignalNode;
Function    DM_DriverBitCount(BitNo : Integer) : Integer;
Function    DM_TargetBitCount(BitNo : Integer) : Integer;
Function    DM_Prefix          : WideString;
Function    DM_Range1          : WideString;
Function    DM_Range2          : WideString;
Function    DM_RangeValue1     : Integer;
Function    DM_RangeValue2     : Integer;
Function    DM_BusKind         : TBusKind;
Function    DM_Width           : Integer;
```

```
Function    DM_RangeMax        : Integer;
Function    DM_RangeMin        : Integer;
Function    DM_PrimaryNode     : ISignalNode;
Function    DM_PowerNode       : ISignalNode;
Function    DM_PowerName       : WideString;
```

See also

Workspace Manager Interfaces

ISignalManager interface

ISignalNode interface

ISubNet interface

ISignalLink interface

TBusKind interface

ISignalLink

Overview

Interface Methods

```
Function    DM_DriverNode              : ISignalNode;
Function    DM_TargetNode              : ISignalNode;
Function    DM_DriverSignal            : ISignal;
Function    DM_DriverNodeRange1        : WideString;
Function    DM_DriverNodeRange2        : WideString;
Function    DM_DriverNodeRangeValue1   : Integer;
Function    DM_DriverNodeRangeValue2   : Integer;
Function    DM_TargetSignal            : ISignal;
Function    DM_TargetNodeRange1        : WideString;
Function    DM_TargetNodeRange2        : WideString;
Function    DM_TargetNodeRangeValue1   : Integer;
Function    DM_TargetNodeRangeValue2   : Integer;
Function    DM_DriverRangeMax          : Integer;
Function    DM_DriverRangeMin          : Integer;
Function    DM_TargetRangeMax          : Integer;
Function    DM_TargetRangeMin          : Integer;
```

See also

Workspace Manager Interfaces

ISignalManager interface

ISignal interface

ISignalNode interface

ISignalManager interface

Overview

Interface Methods

```
Function    DM_SubNets       (Index : Integer) : ISubNet;
Function    DM_Instances     (Index : Integer) : IInstance;
Function    DM_InstanceKinds (Index : Integer) : IInstance;
Function    DM_Signals       (Index : Integer) : ISignal;
```

```
Function    DM_EntityPorts   (Index : Integer) : IEntityPort;

Function    DM_SubNetCount       : Integer;
Function    DM_InstanceCount     : Integer;
Function    DM_InstanceKindCount : Integer;
Function    DM_SignalCount       : Integer;
Function    DM_EntityPortCount   : Integer;
```

See also

Workspace Manager Interfaces

ISubNet interface

IInstance interface

ISignal interface

IEntityPort interface

ISignalNode

Overview

Interface Methods

```
Function    DM_NetItem             : INetItem;
Function    DM_SubNet              : ISubNet;
Function    DM_GetDescription      : WideString;
Function    DM_GetName             : WideString;
Function    DM_Direction           : TSignalDirection;
Function    DM_IsDriver            : LongBool;
Function    DM_Range1              : WideString;
Function    DM_Range2              : WideString;
Function    DM_RangeValue1         : Integer;
Function    DM_RangeValue2         : Integer;
Function    DM_RangeMax            : Integer;
Function    DM_RangeMin            : Integer;
Function    DM_BusIndex            : Integer;
Function    DM_Width               : Integer;

Function    DM_TargetLinks   (Index : Integer) : ISignalLink;
Function    DM_DriverLinks   (Index : Integer) : ISignalLink;
Function    DM_TargetLinkCount     : Integer;
Function    DM_DriverLinkCount     : Integer;
Function    DM_Signal              : ISignal;
Function    DM_EntityPort          : IEntityPort;
Function    DM_ConstantExpression  : WideString;
```

See also

Workspace Manager Interfaces

ISignalManager interface

ISignal interface

ISignalLink interface

IEntityPort interface

TSignalDirection interface

ISubNet interface

Overview

Interface Methods

```
Function    DM_Lines            (Index : Integer) : ILine;
Function    DM_SignalLinks      (Index : Integer) : ISignalLink;
Function    DM_Signals          (Index : Integer) : ISignal;
Function    DM_Nodes            (Index : Integer) : ISignalNode;
Function    DM_PinNodes         (Index : Integer) : ISignalNode;
Function    DM_PowerObjectNodes (Index : Integer) : ISignalNode;
Function    DM_PortNodes        (Index : Integer) : ISignalNode;
Function    DM_NetLabelNodes    (Index : Integer) : ISignalNode;
Function    DM_SheetEntryNodes  (Index : Integer) : ISignalNode;
Function    DM_CrossSheetNodes  (Index : Integer) : ISignalNode;
Function    DM_LineCount            : Integer;
Function    DM_SignalLinkCount      : Integer;
Function    DM_SignalCount          : Integer;
Function    DM_NodeCount            : Integer;
Function    DM_PinNodeCount         : Integer;
Function    DM_PowerObjectNodeCount : Integer;
Function    DM_PortNodeCount        : Integer;
Function    DM_NetLabelNodeCount    : Integer;
Function    DM_SheetEntryNodeCount  : Integer;
Function    DM_CrossSheetNodeCount  : Integer;
Function    DM_Net                  : INet;
```

See also

Workspace Manager Interfaces

ISignalManager interface

ISignal interface

ISignalNode interface

ISignalLink interface

ILine interface

INet interface

WorkSpace Enumerated Types

WorkSpace Enumerated Types

The enumerated types are used for many of the WorkSpace Manager interfaces methods which are covered in this section. For example the IPart interface has a Function DM_ComponentKind : TComponentKind; method. You can use this Enumerated Types section to check what the range is for the TComponentKind type.

See also

Work Space Manager API Reference

TCompilationStage type

TCompileMode type

TECO_Mode type

TErrorGroup type
TErrorKind type
TErrorLevel type
TFlattenMode type
TFlowState type
TModificationKind type
TChannelRoomNamingStyle type
TNetScope type
TParameterKind type
TPinElectrical type
TSystemParmeterKind type
TVariationKind type
TSignalDirection type

TChannelRoomNamingStyle

```
TChannelRoomNamingStyle = (eChannelRoomNamingStyle_FlatNumericWithNames,
                           eChannelRoomNamingStyle_FlatAlphaWithNames,
                           eChannelRoomNamingStyle_NumericNamePath,
                           eChannelRoomNamingStyle_AlphaNamePath,
                           eChannelRoomNamingStyle_MixedNamePath);
```

TCompilationStage

```
TCompilationStage = (eCompilationStage_Compiling,eCompilationStage_Flattening);
```

TCompilationStageSet

```
TCompilationStageSet = Set of TCompilationStage;
```

TCompileMode

```
TCompileMode = (eCompile_None,eCompile_Document,eCompile_All,eCompile_Smart);
```

TComponentKind

```
TComponentKind = (eComponentKind_Standard,
                  eComponentKind_Mechanical,
                  eComponentKind_Graphical,
                  eComponentKind_NetTie_BOM,
                  eComponentKind_NetTie_NoBOM,
                  eComponentKind_Standard_NoBOM);
```

TDisplayMode

```
TDisplayMode = Byte; // one of 255 display modes
```

TECO_Mode

```
TECO_Mode = (eECO_PerformAction,
             eECO_ValidateAction,
             eECO_CheckSupportForAction);
```

TErrorGroup

```
TErrorGroup = (eErrorGroupDocument,
               eErrorGroupComponent,
               eErrorGroupParameters,
```

```
                eErrorGroupBus,
                eErrorGroupNet,
                eErrorGroupMisc);
```

TErrorKind

```
TErrorKind = (eError_OffGridObject,
              eError_OffDocumentObject,
              eError_MissingChildDocument,
              eError_MissingChildProject,
              eError_PortNotLinkedToSheetSymbol,
              eError_SheetEntryNotLinkedToPort,
              eError_DuplicateDocumentNumbers,
              eError_UnconnectedWire,
              eError_UnconnectedNetItem,
              eError_NetWithNoDrivingSource,
              eError_FloatingInputPinsOnNet,
              eError_DifferentConnectionCodesOnNet,
              eError_MultipleSameConnectionCodeOnNet,
              eError_MultipleNamesForNet,
              eError_AddingItemsFromHiddenNetToNet,
              eError_AddingHiddenNet,
              eError_PowerObjectScopeChange,
              eError_NetParameterInvalidName,
              eError_NetParameterInvalidValue,
              eError_MismatchedBusSectionOrdering,
              eError_MismatchedFirstGenericIndex,
              eError_MismatchedSecondGenericIndex,
              eError_MismatchedIOTypeOnBus,
              eError_BusIndexOutOfRange,
              eError_RangeSyntaxError,
              eError_IllegalBusDefinition,
              eError_IllegalBusRangeValue,
              eError_MismatchedBusWidths,
              eError_MismatchedBusLabelOrdering,
              eError_MixedGenericAndNumericBusLabels,
              eError_UnDesignatedPart,
              eError_DuplicateComponentDesignator,
              eError_DuplicateSheetSymbolDesignator,
              eError_DuplicateNets,
              eError_DuplicatePinsInComponent,
              eError_DuplicateSheetEntrysInSheetSymbol,
              eError_DuplicatePortsInDocument,
              eError_DuplicateSubParts,
              eError_MismatchedHiddenPinConnections,
              eError_MismatchedPinVisibility,
              eError_SameParameterWithDifferentValues,
```

```
eError_SameParameterWithDifferentTypes,
eError_MissingModel,
eError_ModelInDifferentLocation,
eError_MissingModelInFile,
eError_DuplicateModelsFound,
eError_MissingModelParameter,
eError_ErrorInModelParameter,
eError_DuplicatePinsInPortMap,
eError_MissingPinInPortMap,
eError_MissingPinsPortMapSequence,
eError_DuplicateImplementation,
eError_UnusedPartInComponent,
eError_ExtraPinInComponentDisplayMode,
eError_MissingPinInComponentDisplayMode,
eError_MismatchedBusAndWire,
eError_FloatingNetLabel,
eError_FloatingPowerObject,
eError_SinglePinNet,
eError_SignalWithNoLoad,
eError_SignalWithNoDriver,
eError_SignalWithMultipleDrivers,
eError_AutoAssignedPin,
eError_NoError,
eError_MultipleTopLevelDocuments,
eError_MultipleConfigurationTargets,
eError_ConflictingConstraints,
eError_MissingConfigurationTarget,
eError_DuplicateUniqueIDs,
eError_DifferentialPairMissingPositiveNet,
eError_DifferentialPairMissingNegativeNet,
eError_DifferentialPairSameNetInMultiplePairs,
eError_DifferentialPairMisConnected,
eError_DifferentialPairInversedPolarity,
eError_DifferentialPairUnconnectedNet);
```

TErrorKindSet

```
TErrorKindSet = Set of TErrorKind;
```

TErrorLevel

```
TErrorLevel = (eErrorLevelNoReport,eErrorLevelWarning,eErrorLevelError,eErrorLevelFatal);
```

TErrorLevelSet

```
TErrorLevelSet = set of TErrorLevel;
```

TFlattenMode

```
TFlattenMode = (eFlatten_Smart,eFlatten_Flat,eFlatten_Hierarchical,eFlatten_Global);
```

TFileOwnerShipWarningLevel

```
TFileOwnershipWarningLevel = (eFileOwnershipWarningLevel_None,
eFileOwnershipWarningLevel_Message, eFileOwnershipWarningLevel_Dialog);
```

TFilterKind

```
TFilterKind = (eFilter_All,
               eFilter_Documents,
               eFilter_Projects,
               eFilter_Workspaces,
               eFilter_ProjectSource_PCB,
               eFilter_ProjectSource_FPGA,
               eFilter_ProjectSource_Core,
               eFilter_ProjectSource_EMB,
               eFilter_ProjectSource_ILB,
               eFilter_ECOFiles,
               eFilter_ProjectSource_SCR);
```

TFlowState

```
TFlowState =
(eState_UpToDate,eState_OutOfDate,eState_Failed,eState_Missing,eState_Running,eState_None);
```

THighlightMethod

```
THighlightMethod =
(eHighlight_Filter,eHighlight_Zoom,eHighlight_Select,eHighlight_Graph,eHighlight_Dim,eHighligh
t_Thicken, eHighlight_ZoomCursor);
```

THighlightMethod

```
THighlightMethodSet = Set Of THighlightMethodSet;
```

TModificationKind

```
TModificationKind =
   ( eModification_Unknown,
     eModification_RemoveNode,
     eModification_RemoveComponentClassMember,
     eModification_RemoveNetClassMember,
     eModification_RemoveChannelClassMember,
     eModification_RemoveRule,
     eModification_RemoveNet,
     eModification_RemoveComponent,
     eModification_ChangeComponentFootPrint,
     eModification_ChangeComponentComment,
     eModification_ChangeComponentDesignator,
     eModification_ChangeComponentKind,
     eModification_AnnotateComponent,
     eModification_AddComponent,
     eModification_ChangeNetName,
     eModification_AddNet,
     eModification_AddNode,
     eModification_RemoveComponentClass,
```

```
eModification_RemoveNetClass,
eModification_RemoveDifferentialPair,
eModification_RemoveChannelClass,
eModification_ChangeComponentClassName,
eModification_ChangeNetClassName,
eModification_ChangeDifferentialPairPositiveNetName,
eModification_ChangeDifferentialPairNegativeNetName,
eModification_ChangeChannelClassName,
eModification_AddComponentClass,
eModification_AddNetClass,
eModification_AddDifferentialPair,
eModification_AddChannelClass,
eModification_AddComponentClassMember,
eModification_AddNetClassMember,
eModification_AddChannelClassMember,
eModification_RemoveRoom,
eModification_ChangeRoom,
eModification_AddRoom,
eModification_AddParameter,
eModification_RemoveParameter,
eModification_ChangeParameterName,
eModification_ChangeParameterValue,
eModification_ChangeParameterType,
eModification_AddRule,
eModification_ChangeRule,
eModification_FullPartUpdate,
eModification_UpdatePartSymbol,
eModification_UpdateImplementationValues,
eModification_AddImplementation,
eModification_RemoveImplementation,
eModification_UpdateCurrentImplementation,
eModification_ChangePinName,
eModification_ChangePinElectrical,
eModification_ChangePortElectrical,
eModification_SwapPin,
eModification_ChangePinSwapId_Pin,
eModification_AddConstraintGroup,
eModification_RemoveConstraintGroup,
eModification_AddPort,
eModification_RemovePort,
eModification_ChangePortName,
eModification_ChangeComponentLibRef,
eModification_ChangePin,
eModification_ChangePhysicalFootPrint,
eModification_ChangeSubPartID,
eModification_ChangeLibraryReference);
```

TNetScope (WSM)

```
TNetScope = (eScopeLocal,eScopeInterface,eScopeGlobal);
```

TParameterKind

```
TParameterKind = (eParameterKind_String,
                  eParameterKind_Boolean,
                  eParameterKind_Integer,
                  eParameterKind_Float);
```

TPathMode

```
TPathMode = (ePathAbsolute,ePathRelative);
```

TPinElectrical (WSM)

```
TPinElectrical    = (eElectricInput,
                     eElectricIO,
                     eElectricOutput,
                     eElectricOpenCollector,
                     eElectricPassive,
                     eElectricHiZ,
                     eElectricOpenEmitter,
                     eElectricPower);
```

TSearchMode

```
TSearchMode = (eSearchModeCurrentDatabase,
               eSearchModeSpecifiedDatabase,
               eSearchModeMultipleDatabases,
               eSearchmodeWitnodwsFileSystem);
```

TSignalDirection

```
TSignalDirection = (eSignalUndefined,eSignalInput,eSignalOutput,eSignalInOut);
```

TSystemParameterKind

```
TSystemParameterKind = (eSystemParameter_UserDefined,
                        eSystemParameter_CurrentTime,
                        eSystemParameter_CurrentDate,
                        eSystemParameter_Time       ,
                        eSystemParameter_Date       ,
                        eSystemParameter_DocFullPath,
                        eSystemParameter_DocName    ,
                        eSystemParameter_ModifiedDate,
                        eSystemParameter_ApprovedBy  ,
                        eSystemParameter_CheckedBy   ,
                        eSystemParameter_Author      ,
                        eSystemParameter_CompanyName ,
                        eSystemParameter_DrawnBy     ,
                        eSystemParameter_Engineer    ,
                        eSystemParameter_Organization,
                        eSystemParameter_Address1    ,
                        eSystemParameter_Address2    ,
```

```
                        eSystemParameter_Address3    ,
                        eSystemParameter_Address4    ,
                        eSystemParameter_Title       ,
                        eSystemParameter_DocNum      ,
                        eSystemParameter_Revision    ,
                        eSystemParameter_SheetNum    ,
                        eSystemParameter_SheetCount  ,
                        eSystemParameter_Rule        ,
                        eSystemParameter_ImagePath   ,
                        eSystemParameter_ConfigurableComponent,
                        eSystemParameter_Configuratorname);
```

TSystemParameterKindSet

```
TSystemParameterKindSet = Set of TSystemParameterKind;
```

TVariationKind

```
TVariationKind = (eVariation_None,eVariation_NotFitted,eVariation_Alternate);
```

TViolationTypeDescription

```
TViolationTypeDescription = Record
    DefaultLevel : TErrorevel;
    Group        : TErrorGroup;
    Description  : TDynamicString;
End;
```

TWorkspaceObjectId

```
TWorkspaceObjectId =
  (
{Abstract Objects}
    eIdAbstractObject,
    eIdConnective,
    eIdConnectiveList,

{Workspaces}
    eIdWorkspace,

{Projects}
    eIdProjectBoardLevel,
    eIdProjectFPGA,
    eIdProjectCore,
    eIdProjectEmbedded,
    eIdProjectLibrary,
    eIdProjectScript,
    eIdProjectFreeDocuments,

{Documents}
    eIdSourceDocument,
    eIdDocumentGeneric,
```

```
    eIdDocumentSCH,
    eIdDocumentSCHLIB,
    eIdDocumentPCBLIB,
    eIdDocumentFlattened,
    eIdDocumentEDIF,
    eIdDocumentVHDL,
    eIdDocumentVERILOG,
    eIdDocumentProtelNetlist,
    eIdDocumentPCADNetlist,
    eIdDocumentPCB,
    eIdDocumentCUPL,
    eIdDocumentAdvSimModel,
    eIdDocumentAdvSimSubCircuit,
    eIdDocumentIntegratedLib,
    eIdDocumentConstraints,
    eIdDocumentPCadPcb,
    eIdDocumentPCadLib,
    eIdDocumentWave,
    eIdDocumentLogicAnalyser,

{Electrical Objects}
    eIdNet,
    eIdBus,
    eIdConstraint,
    eIdConstraintGroup,
    eIdPart,
    eIdFullComponent,
    eIdFullComponentInstance,
    eIdImplementation,
    eIdParameter,
    eIdSheetSymbol,
    eIdTextFrame,
    eIdComponentClass,
    eIdNetClass,
    eIdDifferentialPair,
    eIdChannelClass,
    eIdRule,
    eIdRoom,
    eIdPin,
    eIdSheetEntry,
    eIdNetLabel,
    eIdPowerObject,
    eIdCrossSheet,
    eIdPort,

{InternalObjects}
```

```
    eIdDifference,
    eIdDifferenceManager,
    eIdModification,
    eIdChangeManager,
    eIdViolation,
    eIdViolationManager,
    eIdComponentMapManager,
    eIdVariantManager,
    eIdVHDLEntity,
    eIdConfiguration,

{Spatial Analysis Objects}
    eIdDirective,
    eIdSpatialLine,
    eIdSpatialAnalyser,
    eIdSpatialCompileMask,

{Document Readers}
    eIdReaderSCH,
    eIdReaderSCHLIB,
    eIdReaderPCBLIB,
    eIdReaderEDIF,
    eIdReaderCUPL,
    eIdReaderScript,
    eIdReaderVHDL,
    eIdReaderVerific,
    eIdReaderProtelNetlist,
    eIdReaderPCADNetlist,
    eIdReaderPCB,
    eIdReaderAdvSimLib,

{Cam Objects}
    eIdCamOutputDoc,
    eIdCamView,
    eIdCamJob,
{Printer Objects}
    eIdGeneratedPage,
    eIdPrintView,
    eIdPrinterJob,
    eIdPrinter,

    eIdSignalManager,
    eIdSignal,
    eIdSignalNode,
    eIdSignalLink,
    eIdSubNet,
```

```
    eIdEntityPort,
    eIdInstancePort,
    eIdInstance,
    eIdExternalObject);
```

TWorkspaceObjectIdSet

```
TWorkspaceObjectIdSet = Set Of TWorkspaceObjectId;
```

Workspace Manager Constants

General Constants

```
cStimulusParameterName = 'Stimulus';
cProbeParameterName    = 'Probe';
cAnyCompilationStage : TCompilationStageSet = [Low(TCompilationStage)..
High(TCompilationStage)];
cAnyErrorLevel       : TErrorLevelSet       = [Low(TErrorLevel)      .. High(TErrorLevel)];
cAnyErrorKind        : TErrorKindSet        = [Low(TErrorKind)       .. High(TErrorKind)];

cCompilationStageStrings : Array[TCompilationStage] Of TString = ('Compilation','Flattening');
cErrorLevelStrings       : Array[TErrorLevel]       Of TString = ('No
Report','Warning','Error','Fatal Error');

cImageIndex_ErrorLevels  : Array[TErrorLevel]       Of Integer = (cImageIndexMarker_NoError,
cImageIndexMarker_Warning,
 cImageIndexMarker_Error,   cImageIndexMarker_Fatal);

cNetScopeString          : Array[TNetScope] Of TString = ('Local To Document','Sheet
Interface','Global');
cVendorACTEL  : TDynamicString             = 'Actel';

cMiscellaneous           = 'Miscellaneous';
cCanFastCrossSelect_Emit = 'CanFastCrossSelect_Emit';

cDatabase_KeyFieldDelimiter  = '[[/\]]';

cSystemParameterNames : Array[TSystemParameterKind] Of TString  = (
    'User defined',
    'CurrentTime',
    'CurrentDate',
    'Time',
    'Date',
    'DocumentFullPathAndName',
    'DocumentName',
    'ModifiedDate',
    'ApprovedBy',
    'CheckedBy',
    'Author',
    'CompanyName',
```

```
    'DrawnBy',
    'Engineer',
    'Organization',
    'Address1',
    'Address2',
    'Address3',
    'Address4',
    'Title',
    'DocumentNumber',
    'Revision',
    'SheetNumber',
    'SheetTotal',
    'Rule',
    'ImagePath',
    'ConfigurationParameters',
    'ConfiguratorName');
cDocumentSystemParameters : Set Of TSystemParameterKind = [
    eSystemParameter_UserDefined ,
    eSystemParameter_CurrentTime ,
    eSystemParameter_CurrentDate ,
    eSystemParameter_Time        ,
    eSystemParameter_Date        ,
    eSystemParameter_DocFullPath ,
    eSystemParameter_DocName     ,
    eSystemParameter_ModifiedDate,
    eSystemParameter_ApprovedBy  ,
    eSystemParameter_CheckedBy   ,
    eSystemParameter_Author      ,
    eSystemParameter_CompanyName ,
    eSystemParameter_DrawnBy     ,
    eSystemParameter_Engineer    ,
    eSystemParameter_Organization,
    eSystemParameter_Address1    ,
    eSystemParameter_Address2    ,
    eSystemParameter_Address3    ,
    eSystemParameter_Address4    ,
    eSystemParameter_Title       ,
    eSystemParameter_DocNum      ,
    eSystemParameter_Revision    ,
    eSystemParameter_SheetNum    ,
    eSystemParameter_SheetCount  ,
    eSystemParameter_Rule        ,
    eSystemParameter_ImagePath
];
    cMiscellaneous           = 'Miscellaneous';
    cCanFastCrossSelect_Emit = 'CanFastCrossSelect_Emit';
```

Parameter Simulation Names

```
cParameterName_SimulationFile           = 'Simulation File';
cParameterName_SimulationKind           = 'Simulation Kind';
cParameterName_SimulationSubKind        = 'Simulation SubKind';
cParameterName_SimulationSpicePrefix    = 'Simulation SpicePrefix';
cParameterName_SimulationExcludedParts  = 'Simulation ExcludedParts';
cParameterName_SimulationNetList        = 'Simulation Netlist';
cParameterName_SimulationDescription    = 'Simulation Description';
cParameterName_SimulationModelName      = 'Simulation ModelName';
cParameterName_SimulationPortMap        = 'Simulation PortMap';
cParameterName_SimulationParameters     = 'Simulation Parameters';
```

Document Kind Constants

```
cDocKind_Asm                    =  'ASM';
cDocKind_C                      =  'C';
cDocKind_Camtastic              =  'CAMTASTIC';
cDocKind_Ckt                    =  'CKT';
cDocKind_Constraint             =  'CONSTRAINT';
cDocKind_CoreProject            =  'COREPROJECT';
cDocKind_Cupl                   =  'CUPL';
cDocKind_DatabaseLink           =  'DATABASELINK';
cDocKind_DatabaseLib            =  'DATABASELIB';
cDocKind_Disassembly            =  'DISASSEMBLY';
cDocKind_Edif                   =  'EDIF';
cDocKind_EditScript             =  'EDITSCRIPT';
cDocKind_EditScriptDSUnit       =  'EDITSCRIPTDSUNIT';
cDocKind_EditScriptDSForm       =  'EDITSCRIPTDSFORM';
cDocKind_EditScriptBasUnit      =  'EDITSCRIPTBAS';
cDocKind_EditScriptTclUnit      =  'EDITSCRIPTTCL';
cDocKind_EditScriptVBSUnit      =  'EDITSCRIPTVBSUnit';
cDocKind_EditScriptVBSForm      =  'EDITSCRIPTVBSForm';
cDocKind_EditScriptJSUnit       =  'EDITSCRIPTJSUNIT';
cDocKind_EditScriptJSForm       =  'EDITSCRIPTJSForm';
cDocKind_EmbeddedProject        =  'EMBEDDEDPROJECT';
cDocKind_FavLink                =  'FAVLINK';
cDocKind_Fpgaflow               =  'FPGAFLOW';
cDocKind_FpgaProject            =  'FPGAPROJECT';
cDocKind_FpgaWorkspace          =  'FPGAWORKSPACE';
cDocKind_FreeDocsProject        =  'FREEDOCSPROJECT';
cDocKind_Html                   =  'HTML';
cDocKind_Xml                    =  'XML';
cDocKind_HtmlHelp               =  'HTMLHELP';
cDocKind_IntegratedLibrary      =  'INTEGRATEDLIBRARY';
cDocKind_IntLibrary             =  'INTLIBRARY';
cDocKind_LogicAnalyser          =  'LogicAnalyser';
cDocKind_LogicAnalyserAnalog    =  'LogicAnalyserAnalog';
```

```
cDocKind_Mdl                    =  'MDL';
cDocKind_Nsx                    =  'NSX';
cDocKind_OutputJob              =  'OUTPUTJOB';
cDocKind_PCADPCB                =  'PCADPCB';
cDocKind_Pcb                    =  'PCB';
cDocKind_Situs                  =  'SITUS';
cDocKind_Pcb3DLib               =  'PCB3DLIB';
cDocKind_PcbLib                 =  'PCBLIB';
cDocKind_PCADLIB                =  'PCADLIB';
cDocKind_PcbProject             =  'PCBPROJECT';
cDocKind_PDF                    =  'PDF';
cDocKind_PickATask              =  'PICKATASK';
cDocKind_Profiler               =  'PROFILER';
cDocKind_ProjectGroup           =  'PROJECTGROUP';
cDocKind_ProtelNetlist          =  'PROTELNETLIST';
cDocKind_Sch                    =  'SCH';
cDocKind_Schlib                 =  'SCHLIB';
cDocKind_ScriptProject          =  'SCRIPTPROJECT';
cDocKind_Simdata                =  'SIMDATA';
cDocKind_SIPinModelLibrary      =  'SIPINMODELLIBRARY';
cDocKind_Targets                =  'TARGETS';
cDocKind_Text                   =  'TEXT';
cDocKind_Vhdl                   =  'VHDL';
cDocKind_Verilog                =  'VERILOG';
cDocKind_VhdLib                 =  'VHDLIB';
cDocKind_VhdlSim                =  'VHDLSIM';
cDocKind_VhdTst                 =  'VHDTST';
cDocKind_VerTst                 =  'VERTST';
cDocKind_VQM                    =  'VQM';
cDocKind_Wave                   =  'WAVE';
cDocKind_WaveSim                =  'WAVESIM';
cDocKind_DefaultPcb             =  'DefaultPcb';
cDocKind_DefaultPcbLib          =  'DefaultPcbLib';
cDocKind_SchTemplate            =  'SCHDOT';
cDocKind_DDB                    = 'DDB';

cDocKind_ORCAD7_DSN             = 'ORCAD7_DSN';
cDocKind_ORCAD7_OLB             = 'ORCAD7_OLB';
cDocKind_ORCAD7_MAX             = 'ORCAD7_MAX';
cDocKind_ORCAD7_OLB             = 'ORCAD7_OLB';
cDocKind_ORCAD7_LLB             = 'ORCAD7_LLB';
cDocKind_ORCAD7_CIS             = 'ORCAD7_CIS';

cDocKind_PCAD16_PCB             = 'PCAD16_PCB';
cDocKind_PCAD16_BIN_PCB         = 'PCAD16_BIN_PCB';
cDocKind_PCAD16_NETLIST         = 'PCAD16_NETLIST';
```

```
cDocKind_PCAD16_SCH              = 'PCAD16_SCH';
cDocKind_PCAD16_BIN_SCH          = 'PCAD16_BIN_SCH';
cDocKind_PCAD16_LIA              = 'PCAD16_LIA';
cDocKind_OLD_PCAD_LIB            = 'OLD_PCAD_LIB';
cDocKind_CIRCUITMAKER2000_CKT    = 'CM2000_CKT';
cDocKind_CIRCUITMAKER2000_LIB    = 'CM2000_LIB';
cDocKind_CIRCUITMAKER2000_DEVLIB = 'CM2000_DEVLIB';
cDocKind_PADS_PCB                = 'PADS_PCB';
cDocKind_NGC                     = 'NGC';
```

File Ownership Constants

```
cDefaultFileOwnership_Enabled                = False;
cDefaultFileOwnership_WarningLevelSave       = eFileOwnershipWarningLevel_Dialog;
cDefaultFileOwnership_WarningLevelOpen       = eFileOwnershipWarningLevel_Dialog;
cDefaultFileOwnership_EnabledOutputDirectory = False;

cFileOwnershipWarningLevelStrings : Array[TFileOwnershipWarningLevel] of TDynamicString =
(
   'No warning',
   'Warning in Message Panel',
   'Warning in Dialog box'
);
```

Image Index Table

The Message panel has icons which specify messages. The DM_AddMessage and DM_AddMessageParametric methods of the IWorkSpace interface require an icon.

Image Index Table

Index = -1;	IndexTick = 3;	IndexNoERC = 3;
IndexCross = 4;	IndexConnective = 4;	IndexConnectiveList = 6
Folder = 6;	IndexFreeDocumentsProject = 6	IndexSheetFileName = 15;
OpenDocument = 68;	CloseDocument = 69;	NewFromExistingDocument = 70;
IndexProjectGroup = 54;	IndexProjectGroup2 = 55;	IndexPcbLayer = 51;
IndexEmptySection = 9;	IndexCamJob = 67;	IndexBoardProject = 56;
IndexFpgaProject = 57;	IndexEmbeddedProject = 58;	IndexIntegratedLibrary = 59;
Search = 38;	SearchSelected = 39;	IndexPCB = 52;
IndexPCBVariant = 53;	IndexParameter = 24;	IndexDocumentList = 26;
IndexEdifDocument = 43;	IndexEdifDocumentSelected = 43;	IndexGenericDocument = 62;
IndexTextFile = 62;	IndexCUPLFile = 63;	IndexAdvSimModel = 64;
IndexAdvSimNSX = 48;	IndexAdvSimSubCircuit = 47;	IndexBasicScript = 65;
IndexDelphiScript = 66;	IndexCFile = 45;	IndexVHDLDocument = 44;
IndexVHDLDocumentSelected = 44;	IndexVHDLLibrary = 44;	IndexSheetSymbolList = 30;
HierarchyNets = 30;	IndexPartList = 32;	IndexPinList = 5;

IndexTextFrameList = 28;	IndexProtelNetlistFile = 46;	IndexSchematicSheetSelected = 10
IndexSchematicSheet = 15;	IndexSchematicLibrary = 32;	IndexFlattenedHierarchy = 15;
IndexPCBLibrary = 40;	IndexNet = 1;	IndexBus = 21;
IndexBusEntry = 74;	IndexPart = 2;	IndexComponent = 20;
IndexFootprint = 36;	IndexSubPart = 2;	IndexImplementation = 8;
IndexSheetSymbol = 13;	IndexTextFrame = 18;	IndexPin = 19;
IndexPad = 41;	IndexHiddenName = 19;	IndexNetLabel = 22;
IndexPowerObject = 16;	IndexPort = 17;	IndexSheetEntry = 14;
IndexViolation = 4;	IndexDesignatorMapping = 2	IndexDesignatorManager = 8;
IndexModification = 4;	IndexModificationList = 9;	IndexDifference = 4;
IndexDifferenceList = 8;	IndexNetParameter = 24;	IndexSchematicSheetProcessor = 15
IndexSchematicLibraryProcessor = 15;	IndexEdifDocumentProcessor = 15;	IndexVHDLDocumentProcessor = 15;
IndexVHDLLibraryProcessor = 15;	IndexNetlistFileProcessor = 15;	IndexBoardProcessor = 15;
IndexSpatialAnalyser = 15;	IndexBusSection = 21;	IndexBusElement = 34;
IndexErrorList = 6;	IndexSpatialLine = 1;	IndexComponentClass = 7;
IndexNetClass = 7;	IndexRule = 2;	IndexRoom = 3;
IndexGraphic = 75;	IndexJunction = 76;	IndexAnnotation = 77;
IndexBrowserNetIdentifiers = 78;	IndexLibRef = 79;	IndexComponentParameters = 80;
IndexSheetSymbolParameters = 81;	IndexPortParameters = 82;	IndexPinParameters = 83;
IndexErrorMarker = 84;	IndexParameterSet = 85;	IndexPinsAndParts = 86;
IndexRectangle = 87;	IndexArc = 88;	IndexEllipticalArc = 89;
IndexRoundRectangle = 90;	IndexDesignator = 91;	Indexellipse = 92;
IndexPie = 93;	IndexPolygon = 94;	IndexPolyline = 95;
IndexBezier = 96;	IndexSheetName = 97;	IndexSymbol = 98;
IndexTaskHolder = 99	IndexFolder_NoError = 6;	IndexFolder_Warning = 7;
IndexFolder_Error = 8;	IndexFolder_Fatal = 9;	IndexGeneratedPage = 33;
IndexPrintView = 61;	IndexPrinterJob = 67;	IndexPrinter = 49;
IndexOutput = 61;	IndexAlias = 71;	IndexAliases = 72;
IndexOffsheetPin = 73;	IndexOffSheetPart = 100;	IndexOffSheetNet = 101;
IndexOffSheetBus = 102;	IndexOffSheetPort = 103;	IndexOffSheetSheetEntry = 104;
IndexOffSheetNetLabel = 105;	IndexOffSheetPowerObject = 106;	IndexMarker_NoError = 107;
IndexMarker_Warning = 108;	IndexMarker_Error = 109;	IndexMarker_Fatal = 110;
Index_MainHotSpot1 = 0;	Index_MainHotSpot2 = 1;	Index_MainHotSpot3 = 2;
Index_MainHotSpot4 = 3;	Index_MainHotSpot5 = 4;	Index_MainHotSpot6 = 5;
Index_MainHotSpot7 = 6;	Index_MainHotSpot8 = 7;	Index_MainHotSpot9 = 8;
Index_MainHotSpot10 = 9;		

See also

Work Space Manager API Reference

WorkSpace Manager Functions

Main Interfaces

```
Function WSMServer    : IWSM_ServerInterface;
Function GetWorkspace : IWorkspace;
```

Project and Document Interfaces

```
Function GetProjectOfDocument(Const ADocPath : WideString) : IProject;
Function IsFreeDocument(Const FileName : WideString) : LongBool;
```

Object functions

```
Function IsBusConnector(ALibReference : TDynamicString) : Boolean;
```

Violation and Error Functions

```
Function GetViolationTypeInformation(ErrorKind : TErrorKind) : TViolationTypeDescription;
Function GetViolationTypeDescription(ErrorKind : TErrorKind) : TDynamicString;
Function GetViolationTypeDefaultLevel(ErrorKind : TErrorKind) : TErrorLevel;
Function GetViolationTypeGroup(ErrorKind : TErrorKind) : TErrorGroup;
Function GetErrorLevelColor(ErrorLevel : TErrorLevel) : TColor;
```

See also

Work Space Manager API Reference

IProject interface

TColor type

TDynamicString type

TErrorLevel type

TErrorGroup type

TViolationTypeDescription type

Revision History

Date	Version No.	Revision
22-Nov-2005	1.0	New product release
15-Dec-2005	1.1	Updated for Altium Designer 6
13-Feb-2006	1.2	Revised for Altium Designer 6
28-Jun-2006	1.3	Updated for Altium Designer 6.3
14-Nov-2006	1.4	Updated for Altium Designer 6.6
23-May-2008	1.5	Updated Page Size to A4 and interfaces update.
01-Sep-2011	-	Updated template.