Integrated Library API Reference

Summary

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

The Integrated Library Application Programming Interface reference covers the Integrated Library interface objects.

Integrated Library API Overview

A schematic design is a collection of components which have been connected logically. To test or implement the design it needs to be transferred to another modelling domain, such as simulation, PCB layout, Signal Integrity analysis and so on.

Each domain needs some information about each component, and also some way to map that information to the pins of the schematic component. Some of the domain information resides in model files, the format of which is typically pre-defined, examples of these includes IBIS, MDL and CKT files. Some of the information does not reside in the model files, for example the spice pin mapping and net list data.

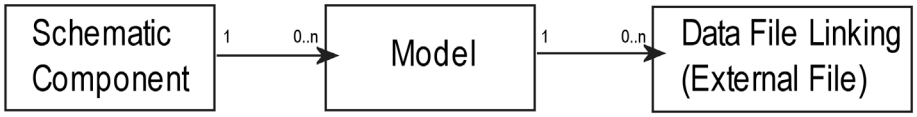
There are different types of libraries in Altium Designer– normal standalone libraries like PCB Libraries and Schematic Libraries and another type called an integrated library which contains different source libraries such as PCB libraries bundled together.

Models

Each schematic component can have models from one or more domains. A schematic component can also have multiple models per domain, one of which will be the current model for that domain.

A model represents all the information needed for a component in a given domain, while a datafile entity (or link) is the only information which is in an external file. See the diagram below for a relationship between a Schematic component and its models. 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 the main editor servers supported in Altium Designer.

A Model has Ports that are mapped to the pins of a schematic component. Note that a model can also be called an implementation. A model/implementation can have its own parameters and data file links.



PCB Footprints

For the PCB footprints, the model and the data file are both the same (no external file needed).

PCB 3D Models

For the PCB 3D models, the model and the data file are both the same (no external file needed).

Simulation Models

With the simulation models, you can have a simulation model which is a 4ohm resistor for example. But there is no information coming from an external file and thus no external file is needed for this resistor as the resistor model is built from Spice modeling language in Altium Designer. This is the case where you have a model with no data file entity.

Thus the parameters are used for these types of simulation models that don't have data file links (external files). Simulation Models can have up to 3 different model data files – Model File(`*.MDL`), Subcircuit file (`*.CKT`) and SIMetrix Model Library file (`*.LB`).

Signal Integrity Models

With signal integrity models, it can have information required for each pin. If we used IBIS datafiles, not the Altium Designer's central database, then each signal integrity model would then have multiple data files, each one for each type of pin.

Integrated Library Interfaces

What are Object Interfaces?

Each method in the object interface is implemented in the corresponding class. Object Interfaces (interfaces for short) are declared like classes but cannot be directly instantiated and do not have their own method definitions.

Each interface, a class supports is actually a list of pointers to methods. Therefore, each time a method call is made to an interface, the interface actually diverts that call to one of it's pointers to a method, thus giving the object that really implements it, the chance to act.

The Integrated Library 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.

To obtain the Integrated Library Manager object interface which represents to the Integrated Library manager object, invoke the `IntegratedLibraryManager` function in your script or code which returns you the `IIntegratedLibraryManager` object interface.

To obtain the model type manager, invoke the `ModelTypeManager` function in your script which returns you the `IModelTypeManger` interface.

Main Integrated Library Interfaces

There are three main interfaces from the Integrated Library Object Model.

- *IIntegratedLibraryManager* Interface
- *IModelTypeManager* Interface
- *IDeviceSheetManager* Interface

IntegratedLibraryManager Interface Example

```
Procedure CheckDataFilesInIntLibrary;
Var
    IntMan          : IIntegratedLibraryManager;
    FoundLocation   : String;
    AFootprintName : String;
    InIntLib        : Boolean;
    AModelType       : String;
Begin
    IntMan := IntegratedLibraryManager;
    If IntMan = Nil Then Exit;
    IntMan.InstallLibrary('C:\Program Files\Altium Designer\Library\Xilinx\Xilinx Spartan-
3E.IntLib');
    //Look for a footprint in a Xilinx Spartan-3E.IntLib
    AModelType      := 'PCBLIB';
    AFootprintName := 'TQ144';
    InIntLib        := True;
    IntMan.FindDatafileInStandardLibs (AFootprintName, AModelType, '', InIntLib,
FoundLocation);
```

```
    ShowMessage(FoundLocation);
    IntMan.UnInstallLibrary('C:\Program Files\Altium Designer 6\Library\Xilinx\Xilinx Spartan-3E.IntLib');
End;
```

Script Examples

There are script examples in the `\Examples\Scripts\` folder of the Altium Designer installation.

IIntegratedLibraryManager Interface

Overview

The `IIntegratedLibraryManager` interface represents the integrated library manager that manages schematic components and its models from installed libraries in Altium Designer.

Invoke the `IntegratedLibraryManager` function to fetch the `IIntegratedLibraryManager` interface.

Integrated Library Manager Methods and Properties Table

IIntegratedLibraryManager methods	IIntegratedLibraryManager properties
AddRemoveLibraries	
AvailableLibraryCount	
AvailableLibraryPath	
AvailableLibraryType	
BrowseComponent	
BrowseDatafileEntity	
BrowseDatafileEntityInDatafile	
BrowseForComponent	
BrowseForComponentAndPart	
BrowseForComponentAndPartCheckDBLibs	
BrowseForComponentCheckDBLibs	
BrowseForDatafile	
BrowseModel	
BrowseSymbol	
ComponentHasModelOfType	
CreateIntegratedLibrary	
ExtractSources	
ExtractSourcesToDatabaseLib	
ExtractSourcesToPath	
FindDatafileInStandardLibs	
FindComponentLibraryPath	
FindComponentDisplayPath	
FindComponentSymbol	
FindDatafileEntityDatafilePath	
FindDatafileEntitySourceDatafilePath	
FindDatafileEntitySourceLibraryPath	
FindDatafileEntityLibraryPath	
FindLibraryInformation	
FindModelLibraryPath	

GetAllParametersFromSourceLib
GetAvailableDBLibDocAtPath
GetComponentCount
GetComponentDatafileLocation
GetComponentLocation
GetComponentLocationFromDatabase
GetComponentName
GetDatabaseDatafileLocation
GetDatafileEntityCount
GetDatafilePath
GetModelCount
GetModelName
GetModelType
GetParametersForDBComponent
GetSchLibPathForDBComponent
GetSchLibRefForDBComponent
GetParameterCount
GetParameterName
GetParameterValue
GetComponentPlacementParameters
InstalledLibraryCount
InstalledLibraryPath
InstallLibrary
IsParameterDatabaseKey
MakeCurrentProject
ModelCount
ModelName
ParseDatabaseKeys
PlaceLibraryComponent
UninstallLibrary

See also

`Examples\Scripts\DXP_Scripts\` folder of Altium Designer installation

Integrated Library Manager Methods

AddRemoveLibraries method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure AddRemoveLibraries;
```

Description

This method invokes the *Available Libraries* dialog with a list of installed libraries if any and their activated, path and type values.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;
IntMan.AddRemoveLibraries;
```

See also

IIntegratedLibraryManager interface

AvailableLibraryType method

(IIntegratedLibraryManager interface)

Syntax

```
Function AvailableLibraryType (LibraryIndex : Integer) : TLibraryType;
```

Description

The AvailableLibraryType function determines what type the indexed library is. Note, the first installed library in the *Available Libraries* dialog is indexed zero (0).

Notes

An available library is one of the libraries on the Installed, Project and Search path tabs within the *Available Libraries* dialog.

An installed library appears in the **Installed** tab of the *Available Libraries* dialog.

```
TLibraryType = (eLibIntegrated, eLibSource, eLibDatafile, eLibDatabase, eLibNone, eLibQuery,
eLibDesignItems);
```

Example

```
    IntMan := IntegratedLibraryManager;
    If IntMan = Nil Then Exit;

    LibType := IntMan.AvailableLibraryType(0);
    Case LibType Of
        eLibIntegrated  : ShowMessage('Integrated');
        eLibSource      : ShowMessage('Lib Source');
        eLibDatafile    : ShowMessage('Lib data File');
        eLibDatabase    : ShowMessage('Database');
        eLibNone        : ShowMessage('None');
        eLibQuery       : ShowMessage('Query');
        eLibDesignItems : ShowMessage('Design Items');
    End;
```

See also

IIntegratedLibraryManager interface

TLibraryType type

AvailableLibraryPath method

(IIntegratedLibraryManager interface)

Syntax

```
Function AvailableLibraryPath (LibraryIndex : Integer) : WideString;
```

Description

The `AvailableLibraryPath` function retrieves the file path of the indexed library in the *Available Libraries* dialog. Note, the first installed library in the *Available Libraries* dialog is indexed zero (0).

Notes

An available library is one of the libraries on the Installed, Project and Search path tabs within the *Available Libraries* dialog.

An installed library appears in the **Installed** tab of the *Available Libraries* dialog.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;
ShowMessage(IntMan.AvailableLibraryPath(0));
```

See also

IIntegratedLibraryManager interface

AvailableLibraryCount method

(IIntegratedLibraryManager interface)

Syntax

```
Function AvailableLibraryCount : Integer;
```

Description

The `AvailableLibraryCount` function determines the number of available libraries. Note, the first installed library in the *Available Libraries* dialog is indexed zero (0).

Notes

An available library is one of the libraries on the Installed, Project and Search path tabs within the *Available Libraries* dialog.

An installed library appears in the **Installed** tab of the *Available Libraries* dialog.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;
ShowMessage(IntToStr(IntMan.AvailableLibraryCount));
```

See also

IIntegratedLibraryManager interface

AvailableLibraryPath method

AvailableLibraryType method

BrowseForDatafile method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure BrowseForDatafile (AModelName : PChar;AModelPath : PChar; LibPath : PChar; ModelType
: PChar; ForComponentInstance : LongBool);
```

Description

This `BrowseForDataFile` procedure invokes the *Browse Libraries* dialog.

Example

```
LibraryPath := 'C:\Program Files\Altium Designer Summer 08\Library\Xilinx\Xilinx Spartan-
3E.IntLib';
ComponentName  := 'XC3S100E-4TQ144I';
ModelType      := 'PCBLIB';
AFootprintName := 'TQ144_N';
IntMan.BrowseForDatafile(AFootprintName, LibraryPath, Librarypath,ModelType, True);
```

See also

IIntegratedLibraryManager interface

BrowseForComponentAndPart method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure BrowseForComponentAndPart (ALibReference : PChar;ASCHLibraryPath :
PChar;SelModelName : PChar;SelModelLib : PChar;LibPath : PChar;ModelType : PChar;Var PartID :
Integer);
```

Description

This `BrowseForComponentAndPart` procedure invokes the Browse for Parts dialog.

Example

See also

IIntegratedLibraryManager interface

BrowseForComponent method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure BrowseForComponent (ALibReference : PChar; ASCHLibraryPath : PChar; SelModelName :
PChar; SelModelLib : PChar; LibPath : PChar; ModelType : PChar);
```

Description

This BrowseForDataFile procedure invokes the Browse for Components dialog.

Example

See also

IIntegratedLibraryManager interface

BrowseForComponentAndpartCheckDBLibs method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure BrowseForComponentAndPartCheckDBLibs (ALibReference : PChar; ASCHLibraryPath :
PChar; SelModelName : PChar;  SelModelLib : PChar; LibPath : PChar; ModelType : PChar;
ADatabaseTableName : PChar; ADatabaseKeys : PChar; Var PartID : Integer);
```

Description

This `BrowseForComponentAndPartCheckDBLibs` procedure invokes the Browse for Components dialog.

Example

See also

IIntegratedLibraryManager interface

ComponentHasModelOfType method

(IIntegratedLibraryManager interface)

Syntax

```
Function ComponentHasModelOfType (LibraryPath : WideString; ComponentIndex : Integer;
AModelType : WideString) : Boolean;
```

Description

This function checks if this indexed component from the specified library has this model type. Model Types include:

- PCBLIB
- PCB3DLIB
- SIM
- SI

Example

```
ComponentIndex := 0;
Status := IntMan.ComponentHasModelOfType(LibraryPath, ComponentIndex, 'PCBLIB');
If Status Then ShowMessage('True') Else ShowMessage('False');
```

See also

IIntegratedLibraryManager interface

CreateIntegratedLibrary method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure CreateIntegratedLibrary (AProject : IProject; AnOutputPath : WideString; Install :
Boolean);
```

Description

This `CreateIntegratedLibrary` procedure creates an integrated library from a project into the specified `AnOutputPath` path and depending on the `Install` parameter is installed in the *Available Libraries* dialog.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;
WSM := GetWorkSpace;
If WSM = Nil Then Exit;

Project := WSM.DM_FocusedProject;
If Project = Nil Then Exit;
LibPath := ChangeFileExt(Project.DM_ProjectFullPath,'.INTLIB');
IntMan.CreateIntegratedLibrary(Project,LibPath,True);
IntMan.MakeCurrentProject(Project);
```

See also

IIntegratedLibraryManager interface

ExtractSourcesToPath method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure ExtractSourcesToPath (ALibraryPath : WideString ;ADestinationPath : WideString);
```

Description

This `ExtractSources` procedure extracts the source files such as PCBLIB and PCB3DLIb files to the destination path (`ADestinationPath` parameter) from the Integrated Library specified by its `ALibraryPath` parameter.

Example

See example for ExtractSources method.

See also

IIntegratedLibraryManager interface

ExtractSources method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure ExtractSources (ALibraryPath : WideString);
```

Description

This `ExtractSources` procedure extracts the source files such as PCBLIB and PCB3DLIb files from the Integrated Library specified by its `ALibraryPath` parameter.

Example

```
Program ExtractSourceLibsFromIntLibs;
Var
   SourceFolder : String;
   FilesList    : TStringList;
   i            : Integer;
Begin
    If IntegratedLibraryManager = Nil then Exit;
    If (InputQuery('Extract IntLib Files','Enter folder containing IntLib
files',SourceFolder)) Then
    Begin
        If (SourceFolder <> '') Then
            If (SourceFolder[Length(SourceFolder)] <> '\') Then
```

```
            SourceFolder := SourceFolder + '\';
        If (DirectoryExists(SourceFolder)) Then
        Begin
           Try
                  FilesList            := TStringList.Create;
                  FilesList.Sorted     := True;
                  FilesList.Duplicates := dupIgnore;
                  // FindFiles function is a built in function from Scripting...
                  FindFiles(SourceFolder,'*.IntLib',faAnyFile,False,FilesList);
                  For i := 0 To FilesList.Count - 1 Do
                       IntegratedLibraryManager.ExtractSources(FilesList.Strings[i]);
           Finally
                     FilesList.Free;
           End;
        End;
    End;
End.
```

See also

IIntegratedLibraryManager interface

FindComponentDisplayPath method

(IIntegratedLibraryManager interface)

Syntax

```
Function FindComponentDisplayPath(ALibIdentifierKind  : TLibIdentifierKind;
                            Const ALibraryIdentifier  : WideString;
                            Const ADesignItemID       : WideString) : WideString;
```

Description

The function returns the full path of the library that the supplied component is part of.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;
// Component is a ISch_Component interface from Schematic API
If Component.LibraryIdentifier <> '' Then
    ShowMessage(IntMan. FindComponentDisplayPath(Component.LibIdentifierKind,
Component.LibraryIdentifier, Component.DesignItemID));
```

See also

IIntegratedLibraryManager interface

FindComponentLibraryPath method

(IIntegratedLibraryManager interface)

Syntax

```
Function  FindComponentLibraryPath(ALibIdentifierKind  : TLibIdentifierKind;
                             Const ALibraryIdentifier  : WideString;
                             Const ADesignItemID       : WideString)  : WideString;
```

Description

The function returns the path of the library the component is part of.

The `ALibIdentifierKind` parameter denotes which type of library the component is from.

```
TLibIdentifierKind = (eLibIdentifierKind_Any,
                      eLibIdentifierKind_NameNoType,
                      eLibIdentifierKind_NameWithType,
                      eLibIdentifierKind_FullPath);
```

The `ALIbraryIdentifier` parameter is the library identifier string that the component is associated with.

The `ADesignItemID` parameter is the symbol reference (library reference) of the component from a Schematic or Integrated Library or an unique part number from a record within a table of a Database.

Example

```
// For each component found on the schematic, you can iterate for its implementations.
//SchImplementation  : ISch_Implementation;
If SchImplementation.UseComponentLibrary Then
Begin
    ComponentPath  :=
IntegratedLibraryManager.FindComponentLibraryPath(Component.LibIdentifierKind,
Component.LibraryIdentifier, Component.DesignItemID);
    ModelPath      := IntegratedLibraryManager.FindModelLibraryPath
(Component.LibIdentifierKind, Component.LibraryIdentifier, Component.DesignItemID,
SchImplementation.ModelName, SchImplementation.ModelType);

    ModelsList.Add(' The Component is in Integrated Library');
    ModelsList.Add('  Component path: ' + ComponentPath);
    ModelsList.Add('  Model path: '     + ModelPath);
End
```

See also

IIntegratedLibraryManager interface

TLibIdentifierKind type

`SimModelsOfComponents` script from `\Examples\Scripts\DelphiScript Scripts\Sch` folder of the Altium Designer installation

FindComponentSymbol method

(IIntegratedLibraryManager interface)

Syntax

```
Function FindComponentSymbol(ALibIdentifierKind  : TLibIdentifierKind;
                      Const ALibraryIdentifier  : WideString;
                      Const ADesignItemID       : WideString;
                      Out   ASymbolLibraryPath  : WideString;
                      Out   ASymbolReference    : WideString)  : Boolean;
```

Description

The function validates whether if the component symbol is available or not dependent on the supplied parameters.

The `ALibIdentifierKind` parameter denotes which type of library the component is from.

```
TLibIdentifierKind = (eLibIdentifierKind_Any,
                      eLibIdentifierKind_NameNoType,
                      eLibIdentifierKind_NameWithType,
                      eLibIdentifierKind_FullPath);
```

The `ALibraryIdentifier` parameter is the library identifier string that the component is associated with.

The `ADesignItemID` parameter is the symbol reference (library reference) of the component from a Schematic or Integrated Library or an unique part number from a record within a table of a Database.

The `ASymbolLibraryPath` is the library.

The ASymbolReference is the name of the component symbol.

Example

```
If IntegratedLibraryManager.FindComponentSymbol(APart.LibIdentifierKind,
APart.LibraryIdentifier, APart.DesignItemID, SymbolLibraryPath, SymbolReference) Then
Begin
    NewSymbolLibrarypath := SymbolLibrarypath;
    NewSymbolReference   := SymbolReference;
End
Else
Begin
    NewSymbolLibrarypath := '';
    NewSymbolReference   := '';
End;
```

See also

IIntegratedLibraryManager interface

FindDatafileEntityDatafilePath method

(IIntegratedLibraryManager interface)

Syntax

```
Function  FindDatafileEntityDatafilePath(ALibIdentifierKind  : TLibIdentifierKind;
                                   Const ALibraryIdentifier  : WideString;
                                   Const ADatafileEntityName : WideString;
                                   Const ADatafileType       : WideString;
                                         AUseIntAndDBLibrary : Boolean)     : WideString;
```

Description

The function returns the path of the library the component is part of.

The ALibIdentifierKind parameter denotes which type of library the component is from.

```
TLibIdentifierKind = (eLibIdentifierKind_Any,
                      eLibIdentifierKind_NameNoType,
                      eLibIdentifierKind_NameWithType,
                      eLibIdentifierKind_FullPath);
```

The ALibraryIdentifier parameter is the library identifier string that the component is associated with.

The ADatafileEntityName parameter

The ADatafileType parameter

The AUseIntAndDBLibrary parameter

Example

```
// For each component found on the schematic, you can iterate for its implementations.
If SchImplementation.DatafileLinkCount > 0 Then
Begin
    // Assumption: use the first data file link for the simulation model since we
    // normally only use one sim model per component.
    ModelDataFile := SchImplementation.DatafileLink[0];

    SourceLibraryPath :=
IntegratedLibraryManager.FindDatafileEntitySourceLibraryPath(ModelDataFile.LibIdentifierKind,
ModelDataFile.LibraryIdentifier, ModelDataFile.EntityName, ModelDataFile.FileKind);
```

```
    LibraryPath       := IntegratedLibraryManager.FindDatafileEntityLibraryPath
(ModelDataFile.LibIdentifierKind, ModelDataFile.LibraryIdentifier, ModelDataFile.EntityName,
ModelDataFile.FileKind);

    SourceDatafilePath :=
IntegratedLibrarymanager.FindDatafileEntitySourceDatafilePath(ModelDataFile.LibIdentifierKind,
ModelDataFile.LibraryIdentifier, ModelDataFile.EntityName, ModelDataFile.FileKind, True);
    Datafilepath      := IntegratedLibrarymanager.FindDatafileEntityDatafilePath
(ModelDataFile.LibIdentifierKind, ModelDataFile.LibraryIdentifier, ModeldataFile.EntityName,
ModelDataFile.FileKind, True);

    ModelsList.Add(' Model : DatafilelinkCount > 0');
    ModelsList.Add('  Source Library path: '  + SourceLibraryPath);
    ModelsList.Add('  Library path: '         + LibraryPath);
    ModelsList.Add('  Source datafile path: ' + SourceDatafilePath);
    ModelsList.Add('  Datafile path: '        + DataFilePath);
End;
```

See also

IIntegratedLibraryManager interface

TLibIdentifierKind type

`SimModelsOfComponents` script from `\Examples\Scripts\DelphiScript Scripts\Sch` folder of the Altium Designer installation

FindDatafileEntitySourceDatafilePath

(IIntegratedLibraryManager interface)

Syntax

```
Function FindDatafileEntitySourceDatafilePath(ALibIdentifierKind  : TLibIdentifierKind;
                                      Const ALibraryIdentifier  : WideString;
                                      Const ADatafileEntityName : WideString;
                                      Const ADatafileType       : WideString;
                                            AUseIntAndDBLibrary : Boolean) : WideString;
```

Description

This function returns the path of the data file in library .

Example

```
// For each component found on the schematic, you can iterate for its implementations.
If SchImplementation.DatafileLinkCount > 0 Then
Begin
    // Assumption: use the first data file link for the simulation model since we
    // normally only use one sim model per component.
    ModelDataFile := SchImplementation.DatafileLink[0];

    SourceLibraryPath :=
IntegratedLibraryManager.FindDatafileEntitySourceLibraryPath(ModelDataFile.LibIdentifierKind,
ModelDataFile.LibraryIdentifier, ModelDataFile.EntityName, ModelDataFile.FileKind);
    LibraryPath       := IntegratedLibraryManager.FindDatafileEntityLibraryPath
(ModelDataFile.LibIdentifierKind, ModelDataFile.LibraryIdentifier, ModelDataFile.EntityName,
ModelDataFile.FileKind);
```

```
    SourceDatafilePath :=
IntegratedLibrarymanager.FindDatafileEntitySourceDatafilePath(ModelDataFile.LibIdentifierKind,
ModelDataFile.LibraryIdentifier, ModelDataFile.EntityName, ModelDataFile.FileKind, True);
    Datafilepath       := IntegratedLibrarymanager.FindDatafileEntityDatafilePath
(ModelDataFile.LibIdentifierKind, ModelDataFile.LibraryIdentifier, ModeldataFile.EntityName,
ModelDataFile.FileKind, True);

    ModelsList.Add(' Model : DatafilelinkCount > 0');
    ModelsList.Add('  Source Library path: '  + SourceLibraryPath);
    ModelsList.Add('  Library path: '         + LibraryPath);
    ModelsList.Add('  Source datafile path: ' + SourceDatafilePath);
    ModelsList.Add('  Datafile path: '        + DataFilePath);
End;
```

See also

IIntegratedLibraryManager interface

FindDatafileEntitySourceLibraryPath

(IIntegratedLibraryManager interface)

Syntax

Description

See also

IIntegratedLibraryManager interface

FindDatafileEntityLibraryPath

(IIntegratedLibraryManager interface)

Syntax

Description

See also

IIntegratedLibraryManager interface

FindDatafileInStandardLibs method

(IIntegratedLibraryManager interface)

Syntax

```
Function FindDatafileInStandardLibs (ADatafileEntityName : WideString; ADatafileType :
WideString; ADatafileLocation : WideString; ForComponentInstance : Boolean; Var
FoundInLibraryPath : WideString) : WideString;
```

Description

This function returns the path of the data file for the specified model in an integrated library. You need to specify the `ADataFileEntityName` parameter which is the footprint name, 3D model name, Sim name or SI name.

The `ADataFileType` parameter denotes the model type represented by the datafiletype (PCB, PCB3DLIB, SIM, SI).

The `ADatafileLocation` parameter is optional.

The `ForComponentInstance` is a Boolean and is true if it is in an integrated library, false otherwise.

The `FoundInLibraryPath` parameter is a returnable value and returns the location of the data file if all the supplied parameters are valid.

Example

```
Var
    IntMan        : IntegratedLibraryManager;
InIntLib      : Boolean;
FoundLocation :
Begin
    IntMan := IntegratedLibraryManager;
    If IntMan = Nil Then Exit;

    IntMan.InstallLibrary('C:\Program Files\Altium Designer\Examples\Reference Designs\4 Port
Serial Interface\Libraries\4 Port Serial Interface.PcbLib');

    InIntLib := False;
    IntMan.FindDatafileInStandardLibs ('DIP14', 'PCBLIB', '', InIntLib, FoundLocation);
    ShowMessage(FoundLocation);
End;
```

See also

IIntegratedLibraryManager interface

FindDatafileEntitySourceLibraryPath

(IIntegratedLibraryManager interface)

Syntax

Description

See also

IIntegratedLibraryManager interface

FindModelLibraryPath

(IIntegratedLibraryManager interface)

Syntax

```
Function  FindModelLibraryPath(ALibIdentifierKind : TLibIdentifierKind;
                        Const ALibraryIdentifier : WideString;
                        Const ADesignItemID      : WideString;
                        Const AModelName         : WideString;
                        Const AModelType         : WideString)  : WideString;
```

Description

The function returns the path of the library the model is part of.

The `ALibIdentifierKind` parameter denotes which type of library the component is from.

```
TLibIdentifierKind = (eLibIdentifierKind_Any,
                      eLibIdentifierKind_NameNoType,
                      eLibIdentifierKind_NameWithType,
                      eLibIdentifierKind_FullPath);
```

The `ALIbraryIdentifier` parameter is the library identifier string that the component is associated with.

The `ADesignItemID` parameter is the symbol reference (library reference) of the component from a Schematic or Integrated Library or an unique part number from a record within a table of a Database.

The `AModelName` parameter is the name of the implementation (model) linked to this component.

The `AModelType` parameter is the model type of the implementation (model) linked to this component.

Example

```
// For each component found on the schematic, you can iterate for its implementations.
//SchImplementation  : ISch_Implementation;
If SchImplementation.UseComponentLibrary Then
Begin
    ComponentPath  :=
IntegratedLibraryManager.FindComponentLibraryPath(Component.LibIdentifierKind,
Component.LibraryIdentifier, Component.DesignItemID);
    ModelPath      := IntegratedLibraryManager.FindModelLibraryPath
(Component.LibIdentifierKind, Component.LibraryIdentifier, Component.DesignItemID,
SchImplementation.ModelName, SchImplementation.ModelType);

    ModelsList.Add(' The Component is in Integrated Library');
    ModelsList.Add('  Component path: ' + ComponentPath);
    ModelsList.Add('  Model path: '     + ModelPath);
End
```

See also

IIntegratedLibraryManager interface

TLibIdentifierKind type

`SimModelsOfComponents` script from `\Examples\Scripts\DelphiScript Scripts\Sch` folder of the Altium Designer installation

FindLibraryInformation method

(IIntegratedLibraryManager interface)

Syntax

```
Function  FindLibraryInformation(ALibIdentifierKind  : TLibIdentifierKind;
                          Const ALibraryIdentifier  : WideString;
                          Const ADesignItemID       : WideString;
                          Out   ALibraryPath        : WideString;
                          Out   ADBTableName        : WideString) : Boolean;
```

Description

The function validates the existence of the library.

The `ALibIdentifierKind` parameter denotes which type of library the component is from.

```
TLibIdentifierKind = (eLibIdentifierKind_Any,
                      eLibIdentifierKind_NameNoType,
                      eLibIdentifierKind_NameWithType,
                      eLibIdentifierKind_FullPath);
```

The `ALIbraryIdentifier` parameter is the library identifier string that the component is associated with. Normally a path to a library.

The `ADesignItemID` parameter is the symbol reference (library reference) of the component from a Schematic or Integrated Library or an unique part number from a record within a table of a Database.

The `ALibraryPath` parameter is returned for the valid design item of a component.

The `ADBTableName` is returned if a component is from a database.

Example

```
If Not IntegratedLibraryManager.FindLibraryInformation(eLibIdentifierKind_NameWithType,
ALibraryIdentifier, ADesignItemID, ALibraryPath, Path, DBTableName) Then Path := '';
```

See also

IIntegratedLibraryManager interface

GetComponentLocation method

(IIntegratedLibraryManager interface)

Syntax

```
Function GetComponentLocation (ALibraryName : WideString;
AComponentName : WideString;
Var FoundInLibraryPath : WideString) : WideString;
```

Description

This `GetComponentLocation` returns the path of the specified component name within the specified library.

Example

```
IntMan.GetComponentLocation('Xilinx Spartan-3E.IntLib',ComponentName, FoundLocation);
ShowMessage(FoundLocation + #13 + 'for ' + ComponentName);
//C:\Program Files\Altium Designer Summer 08\Library\Xilinx\Xilinx Spartan-3E.IntLib
```

See also

IIntegratedLibraryManager interface

GetComponentDatafileLocation method

(IIntegratedLibraryManager interface)

Syntax

```
Function GetComponentDatafileLocation(DatafileIndex : Integer; AModelName : WideString;
AModelType : WideString; AComponentName : WideString; AComponentLibraryName : WideString; Var
FoundInLibraryPath : WideString) : WideString;
```

Description

This `GetComponentDatafileLocation` function obtains the location of the datafile for the component with the specified data file index, model name and its model type, component name and the full library. The result is returned in the `FoundInLibraryPath` parameter or by the function itself.

Example

```
LibraryPath := 'C:\Program Files\Altium Designer Summer 08\Library\Xilinx\Xilinx Spartan-
3E.IntLib';
ComponentName  := 'XC3S100E-4TQ144I';
IntMan.GetComponentDatafileLocation(0, 'TQ144_L', 'PCBLIB', ComponentName, LibraryPath,
FoundLocation);
ShowMessage(FoundLocation + ' for Component Datafile location');
// 'C:\Program Files\Altium Designer Summer 08\Library\Xilinx\Xilinx Spartan-3E.IntLib';
```

See also

IIntegratedLibraryManager interface

GetModelType method

(IIntegratedLibraryManager interface)

Syntax

```
Function GetModelType (LibraryPath : WideString; ComponentIndex : Integer; ModelIndex :
Integer) : IModelType;
```

Description

This function retrieves the model type for the indexed component within the specified library. The first indexed component is 0.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;


LibraryPath := 'C:\Program Files\Altium Designer Summer 08\Library\Xilinx\Xilinx Spartan-
3E.IntLib';
```

```
// when ComponentIndex = 0, Component = 'XC3S100E-4TQ144I';

ModelType := IntMan.GetModelType(LibraryPath, 0, 2); //0 = PCBLIB, 1 = PCB3DLIB 2 = SI
Showmessage(ModelType.Name);
```

See also

IIntegratedLibraryManager interface

IModelType interface

GetModelName method

GetModelCount method

GetModelName method

(IIntegratedLibraryManager interface)

Syntax

```
Function GetModelName (LibraryPath : WideString; ComponentIndex : Integer; ModelIndex :
Integer) : WideString;
```

Description

This function retrieves the model name for the indexed component within the specified library. The first indexed component is 0.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;

LibraryPath := 'C:\Program Files\Altium Designer Summer 08\Library\Xilinx\Xilinx Spartan-
3E.IntLib';
// when ComponentIndex = 0, Component = 'XC3S100E-4TQ144I';
Showmessage(IntMan.GetModelName(LibraryPath, 0, 0)); //0 = CP132, 1 = XC3S100E-CP132 2 =
XC3S100E_CP132
```

See also

IIntegratedLibraryManager interface

GetModelCount method

GetModelType method

GetModelCount method

(IIntegratedLibraryManager interface)

Syntax

```
Function GetModelCount (LibraryPath : WideString; ComponentIndex : Integer) : Integer;
```

Description

This function retrieves the model count for the indexed component within the specified library. The first indexed component is 0.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;
LibraryPath := 'C:\Program Files\Altium Designer Summer 08\Library\Xilinx\Xilinx Spartan-
3E.IntLib';
// when ComponentIndex = 0, Component = 'XC3S100E-4TQ144I';
Showmessage(IntMan.GetModelName(LibraryPath, 0)); //3 models for this component
```

See also

IIntegratedLibraryManager interface

GetModelName method

GetModelType method

GetDatafilePath method

(IIntegratedLibraryManager interface)

Syntax

```
Function GetDatafilePath (LibraryPath : WideString; ComponentIndex : Integer; ModelIndex :
Integer; DatafileIndex : Integer) : WideString;
```

Description

This function gets datafile path for the specified component, its indexed model and its indexed datafile in the specified library path. Remember first index is 0.

Example

See also

IIntegratedLibraryManager interface

GetDatafileEntityCount method

(IIntegratedLibraryManager interface)

Syntax

```
Function GetDatafileEntityCount (LibraryPath : WideString; ComponentIndex : Integer;
ModelIndex : Integer) : Integer;
```

Description

This function gets datafile entity count for the specified component and its indexed model in the specified library path. Remember first index is 0.

Example

```
DataEntityCount := IntLib.GetDatafileEntityCount(Librarypath,I,0);
ShowMessage(IntToStr(DataEntityCount);
// indexed component is I and 0 is the first model for the component.
```

See also

IIntegratedLibraryManager interface

GetComponentName method

(IIntegratedLibraryManager interface)

Syntax

```
Function GetComponentName (LibraryPath : WideString; ComponentIndex : Integer) : WideString;
```

Description

This function retrieves the name for the indexed component within the specified integrated library. Remember first index is 0.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;

S :=  '';
AvailLibPath := IntMan.AvailableLibraryPath(1);
AComponentIndex := IntMan.GetComponentCount(IntMan.AvailableLibraryPath(1));

For I := 0 To AComponentIndex Do
    S := S + ' ' + Intman.GetComponentName (AvailLibpath,I);

ShowMessage(s);
```

See also

IIntegratedLibraryManager interface

GetComponentCount method

(IIntegratedLibraryManager interface)

Syntax

```
Function GetComponentCount (LibraryPath : WideString) : Integer;
```

Description

This function retrieves the count of components within the integrated library specified by the `LibraryPath` parameter.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;
ShowMessage(IntMan.GetComponentCount(IntMan.AvailableLibraryPath(1)));
```

See also

IIntegratedLibraryManager interface

InstalledLibraryPath method

(IIntegratedLibraryManager interface)

Syntax

```
Function InstalledLibraryPath (anIndex : Integer) : WideString;
```

Description

This `InstalledLibraryPath` function retrieves the path of the indexed installed library in Altium Designer. An installed library appears in the installed libraries list box in the **Installed** tab of the *Available Libraries* dialog.

Example

```
Procedure RemoveOriginalInstalledFiles;
Var
    I : Integer;
Begin
    IntMan := IntegratedLibraryManager;
    If IntMan = Nil then Exit;

    OriginalInstalledList := TStringList.Create;
    For I := 0 to IntMan.InstalledLibraryCount - 1 Do
    Begin
        OriginalInstalledList.Add(IntMan.InstalledLibraryPath(I));
        IntMan.UnInstallLibrary(IntMan.InstalledLibraryPath(I));
    End;
    // do what you want with the OriginalInstalledList
    OriginalInstalledList.Free;
End;
```

See also

IIntegratedLibraryManager interface

InstalledLibraryCount method

(IIntegratedLibraryManager interface)

Syntax

```
Function InstalledLibraryCount : Integer;
```

Description

This `InstalledLibraryCount` function reports the number of installed libraries as in the **Installed** tab of the *Available Libraries* dialog in Altium Designer.

Example

```
Procedure RemoveOriginalInstalledFiles;
Var
    I : Integer;
Begin
    IntMan := IntegratedLibraryManager;
    If IntMan = Nil then Exit;

    OriginalInstalledList := TStringList.Create;
    For I := 0 to IntMan.InstalledLibraryCount - 1 Do
    Begin
        OriginalInstalledList.Add(IntMan.InstalledLibraryPath(I));
        IntMan.UnInstallLibrary(IntMan.InstalledLibraryPath(I));
    End;
End;
```

See also

IIntegratedLibraryManager interface

InstalledLibraryPath method

AvailableLibraryPath method

AvailableLibraryCount method

InstallLibrary method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure InstallLibrary (ALibraryPath : WideString);
```

Description

This procedure installs the library (full path) in the *Available Libraries* dialog (in the **Installed** page) in Altium Designer.

Example

```
IntegratedLibraryManager.InstallLibrary('C:\Program Files\Altium
Designer\Library\Xilinx\Xilinx Spartan-3E.IntLib');
```

See also

IIntegratedLibraryManager interface

UnInstallLibrary method

MakeCurrentProject method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure MakeCurrentProject (AProject : IProject);
```

Description

This procedure makes the current library in the Libraries panel based on the project.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;
WSM := GetWorkSpace;
If WSM = Nil Then Exit;

Project := WSM.DM_FocusedProject;
If Project = Nil Then Exit;
```

```
LibPath := ChangeFileExt(Project.DM_ProjectFullPath,'.INTLIB');
IntMan.CreateIntegratedLibrary(Project,LibPath,True);
IntMan.MakeCurrentProject(Project);
```

See also

IIntegratedLibraryManager interface

ModelName method

(IIntegratedLibraryManager interface)

Syntax

```
Function ModelName (AComponentName : WideString; AComponentLibraryName : WideString;
AModelType : WideString; AnIndex : Integer) : WideString;
```

Description

This `ModelName` function returns the name of the model type associated with the component within a specified library.

Example

```
LibraryPath := 'C:\Program Files\Altium Designer Summer 08\Library\Xilinx\Xilinx Spartan-
3E.IntLib';
ModelCount := IntMan.ModelCount(ComponentName,LibraryPath, 'PCBLIB');
ShowMessage(ComponentName + '''s ModelCount : ' + IntToStr(ModelCount));

S := '';
For I := 0 to ModelCount - 1 Do
    S := S + #13 + IntMan.ModelName (ComponentName, LibraryPath, 'PCBLIB', I);

ShowMessage(S);
// TQ144_L, TQ144_M, TQ144_N
```

See also

IIntegratedLibraryManager interface

ModelCount method

(IIntegratedLibraryManager interface)

Syntax

```
Function ModelCount (AComponentName : WideString; AComponentLibraryName : WideString;
AModelType : WideString) : Integer;
```

Description

This `ModelCount` function returns the number of models of the same type associated with the component within the specified library. The `AComponentName` parameter is the name of the component. The `AComponentLibraryName` parameter is the full path of the library the component is from, and the `AModelType` parameter is the model type you wish to find how many.

Example

```
ModelCount := IntMan.ModelCount(ComponentName,'C:\Program Files\Altium Designer Summer
08\Library\Xilinx\Xilinx Spartan-3E.IntLib', 'PCBLIB');
ShowMessage(ComponentName + '''s ModelCount : ' + IntToStr(ModelCount));
// XC3S100E-4TQ144I's Model count: 3
```

See also

IIntegratedLibraryManager interface

PlaceLibraryComponent method

(IIntegratedLibraryManager interface)

Syntax

```
Function PlaceLibraryComponent (ALibReference : PChar; ALibraryPath : PChar; Parameters :
PChar) : Boolean;
```

Description

This method places a component from a specified library with Library Reference and Parameters that describe/define this component.

The `ALIbReference` parameter defines the component. For example ‘Res2’

The `ALibraryPath` parameter defines the path to the library that the component is from. For example ‘Miscellaneous Devices.IntLib’

The `Parameters` parameter defines the parameters needed for the component to be able to be placed on the schematic sheet. For example `'ModelType=SIM|ModelParameterName0=Value|ModelParameterValue0=1K|Orientation=1|Location.X=10000000|Location.Y=20000000'`. Normally you will need Location.X and Location.Y parameters at the least to be able to place this component on the schematic sheet.

Example

```
Procedure PlaceAPartProgrammatically;
Begin
    If SchServer = Nil Then Exit;
    If SchServer.GetCurrentSchDocument = Nil Then Exit;
    If IntegratedLibraryManager = Nil Then Exit;

    // Integrated Library object model is used to place a
    // component from the library onto the schematic sheet.
    IntegratedLibraryManager.PlaceLibraryComponent(
        'Res2',
        'Miscellaneous Devices.IntLib',
        'ModelType=SIM|ModelParameterName0=Value|' +
        'ModelParameterValue0=1K|Orientation=1|Location.X=10000000|Location.Y=20000000');

     // Refresh screen
    SchServer.GetCurrentSchDocument.GraphicallyInvalidate;
End;
```

See also

IIntegratedLibraryManager interface

UninstallLibrary method

(IIntegratedLibraryManager interface)

Syntax

```
Procedure UninstallLibrary (ALibraryPath : WideString);
```

Description

This procedure removes the specified library (full path) in the *Available Libraries* dialog (in the **Installed** page) in Altium Designer

Example

```
IntegratedLibraryManager.UnInstallLibrary('C:\Program Files\Altium
Designer\Library\Xilinx\Xilinx Spartan-3E.IntLib');
```

See also

IIntegratedLibraryManager interface

InstallLibrary method

IModelTypeManager Interface

Overview

The `IModelTypeManager` interface represents a repository of available model types in Altium Designer. The Implementation files (`*.IMP`) from the System folder of Altium Designer Installation are collected and processed by this manager.

Each model that can be linked to a schematic component has a model type and model data file(s).

- PCB Model has one model data file – footprints (`*.PCBLIB`)
- PCB 3D Model has one model data file –3D models (`*.PCB3DLib`)
- Signal Integrity Model has one model data file – pin model library.
- Simulation has 3 model data files – Model File(`*.MDL`), Subcircuit file (`*.CKT`) and SIMetrix Model Library file (`*.LB`).

This `IModelTypeManager` interface uses `IModelType` and `IModelDataType` interfaces to store different model types and their model data types.

Invoke the `ModelTypeManager` function to fetch the `IModelTypeManager` interface.

IModelTypeManager Methods and Properties Table

IModelTypeManager methods	IModelTypeManager properties
ModelTypeCount	ModelTypes
ModelTypeAt	ModelDatafileTypes
ModelTypeFromName	
ModelTypeFromServerName	
ModelDatafileTypeCount	
ModelDatafileTypeAt	
ModelDatafileTypeFromKind	

See also

IModelType interface

IModelDataType interface

`Examples\Scripts\DXP_Scripts\` folder of Altium Designer installation

IModelTypeManager Methods

ModelTypeFromServerName method

(IModelTypeManager interface)

Syntax

```
Function ModelTypeFromServerName (AName : PChar) : IModelType;
```

Description

This function returns the model type interface based on the server name. The Server names can be:

- PCB3D
- PCB
- Sim
- SignalIntegrity

Example

```
Procedure ShowAModelFromModelTypeManager;
Var
    ModelTypeMan : IModelTypeManager;
    I            : Integer;
    ModelType    : IModelType;
```

```
Begin
    ModelTypeMan := ModelTypeManager;
    If ModelTypeMan = Nil Then Exit;
    ModelType := ModelTypeMan.ModelTypeFromServerName('SIM');
    ShowMessage(ModelType.Description); //Simulation
End;
```

See also

IModelTypeManager interface

ModelTypeFromName method

(IModelTypeManager interface)

Syntax

```
Function ModelTypeFromName (AName : PChar) : IModelType;
```

Description

This function returns the model type interface based on the model type name. The names can be:

- PCB3DLIB
- PCBLIB
- SI
- SIM

Example

```
Procedure DisplayModelTypeFromName;
Var
    AModelTypeManager : IModelTypeManager;
    AModelType        : IModelType;
Begin
    AModelTypeManager := ModelTypeManager;
    If AModelTypeManager = Nil Then Exit;

    //AModelType := AModelTypeManager.ModelTypeFromName('PCBLIB');
    //AModelType := AModelTypeManager.ModelTypeFromName('PCB3DLib');
    //AModelType := AModelTypeManager.ModelTypeFromName('SI');
    AModelType := AModelTypeManager.ModelTypeFromName('SIM');
    If AModelType <> Nil Then
        ShowMessage(AModelType.Description);
End;
```

See also

IModelTypeManager interface

ModelTypeCount method

(IModelTypeManager interface)

Syntax

```
Function ModelTypeCount : Integer;
```

Description

This function returns the number of models supported by Altium Designer. The available models are PCBLIB, SI, SIM and PCB3DLIB types.

Example

```
Procedure ShowModelTypesFromModelTypeManager;
```

```
Var
    ModelTypeMan : IModelTypeManager;
    I            : Integer;
Begin
    ModelTypeMan := ModelTypeManager;
    If ModelTypeMan = Nil Then Exit;

    For I := 0 To ModelTypeMan.ModelTypeCount -1 do
       ShowMessage(ModelTypeMan.ModelTypes[i].Name); // 4 model types supported
End;
```

See also

IModelTypeManager interface

ModelTypeAt method

(IModelTypeManager interface)

Syntax

```
Function ModelTypeAt (AnIndex : Integer) : IModelType;
```

Description

This function returns the indexed model type. First model type starts at 0. This method is used by the ModelTypes property.

Example

```
Procedure ShowFirstModelTypeFromModelTypeManager;
Var
    ModelTypeMan : IModelTypeManager;
    I            : Integer;
Begin
    ModelTypeMan := ModelTypeManager;
    If ModelTypeMan = Nil Then Exit;
    ShowMessage(ModelTypeMan.ModelTypeAt(0).Name);
End;
```

See also

IModelTypeManager interface

ModelTypeCount method

ModelDatafileTypeFromKind method

(IModelTypeManager interface)

Syntax

```
Function ModelDatafileTypeFromKind (AKind : PChar) : IModelDatafileType;
```

Description

This function returns the IModelDatafileType based on the datafile kind. The datafile kinds for:

Model Type (Kind)	DatafileType Description
MDL	Sim Model File
CKT	Sim Subcircuit File
LB	SIMetrix Model Library File
SIPinModelLibrary	SI Pin Model Library
PCBLIB	Protel Footprint Library

PCB3DLIB	PCB3D Model Library

Example

```
Procedure ShowDataFileTypeFromModelTypeManager;
Var
    ModelTypeMan : IModelTypeManager;
Begin
    ModelTypeMan := ModelTypeManager;
    If ModelTypeMan = Nil Then Exit;
    Showmessage(ModelTypeMan.ModelDatafileTypeFromKind('SIPinModelLibrary').Description);
    //SI Pin Model Library
End;
```

See also

IModelTypeManager interface

ModelDatafileTypeCount method

(IModelTypeManager interface)

Syntax

```
Function ModelDatafileTypeCount : Integer;
```

Description

This function reports the number of model data file types used by Altium Designer. Since there are four models supported and Simulation model has 3 types while the other 3 models has one type each making 6 in total.

Example

```
Procedure ShowModelDatafileTypeCount;
Var
    ModelTypeMan : IModelTypeManager;
    I            : Integer;
Begin
    ModelTypeMan := ModelTypeManager;
    If ModelTypeMan = Nil Then Exit;
    Showmessage(IntToStr(ModelTypeMan.ModelDatafileTypeCount)); //6 data file types
End;
```

See also

IModelTypeManager interface

ModelDatafileTypeAt method

(IModelTypeManager interface)

Syntax

```
Function ModelDatafileTypeAt (AnIndex : Integer) : IModelDatafileType;
```

Description

This method returns the data file types supported by Altium Designer. First data file type starts at 0. This method is used by the ModelDataFileTypes property.

Example

```
Procedure ShowModelDatafileTypes;
Var
    ModelTypeMan      : IModelTypeManager;
    ModelDatafileType : ModelDatafileType;
    I                 : Integer;
```

```
Begin
    ModelTypeMan := ModelTypeManager;
    If ModelTypeMan = Nil Then Exit;
    //6 data file types
    For I := 0 To ModelTypeMan.ModelDatafileTypeCount - 1 do
    Begin
        ModelDatafileType := ModelTypeMan.ModelDatafileTypeAt(I);
        ShowMessage(ModelDatafileType.FileKind + #13 + ModelDatafileType.Description);
    End;
End;
```

See also

IModelTypeManager interface

ModelDatafileTypes property

ModelDatafileTypeCount method

IModelTypeManager Properties

ModelDatafileTypes property

(IModelTypeManager interface)

Syntax

```
Property ModelDatafileTypes[AnIndex : Integer] : IModelDatafileType Read ModelDatafileTypeAt;
```

Description

This property returns the data file types supported by Altium Designer. First data file type starts at 0. This property is supported by the ModelDataFileTypeAt method.

Example

```
Procedure ShowModelDatafileTypes;
Var
    ModelTypeMan      : IModelTypeManager;
    ModelDatafileType : ModelDatafileType;
    I                 : Integer;
Begin
    ModelTypeMan := ModelTypeManager;
    If ModelTypeMan = Nil Then Exit;
    //6 data file types
    For I := 0 To ModelTypeMan.ModelDatafileTypeCount - 1 do
    Begin
        ModelDatafileType := ModelTypeMan.ModelDatafileTypes[I];
        ShowMessage(ModelDatafileType.FileKind + #13 + ModelDatafileType.Description);
    End;
End;
```

See also

IModelTypeManager interface

IModelDatafileType interface

ModelDatafileTypeCount method

ModelTypes property

(IModelTypeManager interface)

Syntax

```
Property ModelTypes [AnIndex : Integer] : IModelType Read ModelTypeAt;
```

Description

This function returns the indexed model type. First model type starts at 0. This property is supported by the ModelTypeAt method.

Example

```
Procedure ShowModelsFromModelTypeManager;
Var
    ModelTypeMan : IModelTypeManager;
    I            : Integer;
Begin
    ModelTypeMan := ModelTypeManager;
    If ModelTypeMan = Nil Then Exit;

    For i := 0 To ModelTypeMan.ModelTypeCount -1 do
       ShowMessage(ModelTypeMan.ModelTypes[i].Name);
End;
```

See also

IModelTypeManager interface

IModelType interface

ModelTypeAt method

IDeviceSheetManager Interface

Overview

The `IDeviceSheetManager` interface represents the *Device Sheets Folder* dialog in Altium Designer. Invoke the `DeviceSheetManager` function to fetch the `IDeviceSheetManager` object interface.

IDeviceSheetManager Methods and Properties Table

IDeviceSheetManager methods	IDeviceSheetManager properties
EditDeviceFolderList	
FindDeviceSheetPath	
BrowseDeviceSheet	
GetFoldersCount	
GetFolders_FolderPath	
GetFolders_SearchSubFolders	
WillSearchDeviceFolder	
AddDeviceFolder	
ChooseDeviceFolder	
ConvertDeviceSheetPathToName	

See also

DeviceSheetManager function.

IDeviceSheetManager Methods

AddDeviceFolder Method

(IDeviceSheetManagerManager interface)

Syntax

```
Function  AddDeviceFolder(Const AFolderPath : WideString; ASearchSubfolder : Boolean) : Boolean;
```

Description

This function adds a new device folder into the existing top level Device Folder and whether sub folders can be searched from that folder.

Example

```
If Not DeviceSheetManager.WillSearchDeviceFolder(ExtractFilePath(ASheetPath)) Then
    DeviceSheetManager.AddDeviceFolder(ExtractFilePath(ASheetPath), False);
```

See also

IDeviceSheetManagerManager interface

BrowseDeviceSheet Method

(IDeviceSheetManagerManager interface)

Syntax

```
Function BrowseDeviceSheet (Var   AFileName : WideString; Out AFilePath : WideString) : Boolean;
```

Description

The function `BrowseDeviceSheet` invokes the *Select Device Sheet* dialog and when you select a device sheet, the filename (without the file extension) is returned for the device sheet you chose from this dialog. This filename is returned in the `AFilename` parameter.

Example

```
DeviceSheetMan := DeviceSheetManager;
If DeviceSheetMan = Nil Then Exit;
AFilepath := ''; Afilepath is returned blank.
DeviceSheetMan.BrowseDeviceSheet(AFileName,AFilepath);
ShowMessage('Filename ' + AFileName);
```

See also

IDeviceSheetManagerManager interface

ChooseDeviceFolder Method

(IDeviceSheetManagerManager interface)

Syntax

```
Function  ChooseDeviceFolder(Var AFolderPath : WideString) : Boolean;
```

Description

This function invokes the *Choose Device Sheet Folder* dialog and returns you the valid device folder via the `AFolderPath` parameter. The function returns a false value if the dialog is cancelled.

Example

```
FolderPath := ExtractFilePath(DeviceSheetPathText);
If DeviceSheetManager.ChooseDeviceFolder(FolderPath) Then
     DeviceSheetPathText := AddSlash(FolderPath) + ExtractFileName(DeviceSheetPathText);
```

See also

IDeviceSheetManagerManager interface

ConvertDeviceSheetPathToName Method

(IDeviceSheetManagerManager interface)

Syntax

```
Function ConvertDeviceSheetPathToName (Const AFilePath : WideString) : WideString;
```

Description

The function converts the full file path (the `AFilePath` parameter) of the device sheet to the valid device sheet filename (without the file extension). If the `AFilePath` parameter is invalid, an empty string is returned.

Example

```
ShowMessage(DeviceSheetMan.ConvertDeviceSheetPathToName('C:\Program Files\Altium Designer
Summer 08\Library\Device Sheets\Audio\AUDIO_AMP_LM4849.Harness'));
//Returns the filename of the valid device sheet (without the file extension).
```

See also

IDeviceSheetManagerManager interface

EditDeviceFolderList Method

(IDeviceSheetManagerManager interface)

Syntax

```
Procedure EditDeviceFolderList;
```

Description

This procedure invokes the *Device Sheet Folders* dialog with all Device Sheet Folders if any.

Example

```
DeviceSheetMan := DeviceSheetManager;
If DeviceSheetMan = Nil Then Exit;
DeviceSheetMan.EditDeviceFolderList;
```

See also

IDeviceSheetManagerManager interface

FindDeviceSheetPath Method

(IDeviceSheetManagerManager interface)

Syntax

```
Function FindDeviceSheetPath (Const AFileName : WideString) : WideString;
```

Description

This function finds the Device Sheet path for the valid device sheet (without the file extension). The valid device sheet is defined by the `AFilename` parameter. If the `AFileName` is invalid, a blank string is returned.

Example

```
ShowMessage(DeviceSheetMan.FindDeviceSheetPath('AUDIO_AMP_LM4849'));
```

See also

IDeviceSheetManagerManager interface

GetFoldersCount Method

(IDeviceSheetManagerManager interface)

Syntax

```
Function  GetFoldersCount : Integer;
```

Description

The GetFoldersCount function returns the number of Device Sheet Folders in Altium Designer.

Example

```
DeviceSheetMan := DeviceSheetManager;
If DeviceSheetMan = Nil Then Exit;
Count := DeviceSheetMan.GetFoldersCount;
ShowMessage(IntToStr(Count));
```

See also

IDeviceSheetManagerManager interface

GetFolders_FolderPath Method

(IDeviceSheetManagerManager interface)

Syntax

```
Function  GetFolders_FolderPath(AIndex : Integer) : WideString;
```

Description

This function returns the indexed path of device sheets (as in the *Device Sheet Folders* dialog). The first entry starts at zero (0).

Example

```
DeviceSheetMan := DeviceSheetManager;
If DeviceSheetMan = Nil Then Exit;
Count := DeviceSheetMan.GetFoldersCount;
ShowMessage(DeviceSheetMan.GetFolders_FolderPath(Count-1));
```

See also

IDeviceSheetManagerManager interface

GetFoldersCount method

GetFolders_SearchSubFolders Method

(IDeviceSheetManagerManager interface)

Syntax

```
Function GetFolders_SearchSubFolders(AIndex : Integer): Boolean;
```

Description

This function returns a boolean result for sub folders of the indexed path of device sheets (as in the *Device Sheet Folders* dialog). The first entry starts at zero (0).

Example

```
DeviceSheetMan := DeviceSheetManager;
If DeviceSheetMan = Nil Then Exit;
Result := DeviceSheetMan.GetFolders_SearchSubFolders(0);
If Result Then
    ShowMessage(DeviceSheetMan.GetFolders_FolderPath(0) + 'has its sub folders');
```

See also

IDeviceSheetManagerManager interface

GetFoldersCount method

WillSearchDeviceFolder Method

(IDeviceSheetManagerManager interface)

Syntax

```
Function WillSearchDeviceFolder(Const AFolderPath : WideString) : Boolean;
```

Description

This function determines whether the Device Sheet Folder represented by the `AFolderPath` parameter exists or not.

Example

```
If Not DeviceSheetManager.WillSearchDeviceFolder(ExtractFilePath(ASheetPath)) Then
    DeviceSheetManager.AddDeviceFolder(ExtractFilePath(ASheetPath), False);
```

See also

IDeviceSheetManagerManager interface

IModelDataFile Interface

Overview

The `IModelDatafile` interface represents the data file (external file) that is associated with a model. Each model can have multiple data files (different representations of the same model type).

This interface is used within the `IServerModel` interface.

IModelDataFile Methods and Properties Table

IModelDatafile methods	IModelDatafile properties
FullPath	EntityNames
EntityCount	
EntityName	
AddEntity	

See also

`IModelDatafileType interface`

`Examples\Scripts\DelphiScript Scripts\DXP_Scripts\` folder of Altium Designer installation

IModelDataFile Methods

EntityName method

(IModelDatafile interface)

Syntax

```
Function EntityName (AnIndex : Integer) : WideString;
```

Description

The function returns the indexed entityname for the datafile related to the model.

See also

IModelDatafile interface

EntityCount method

EntityCount method

(IModelDatafile interface)

Syntax

```
Function EntityCount : Integer;
```

Description

This function returns the number of entities for the data file related to the model.

See also

IModelDatafile interface

EntityName method

AddEntity method

(IModelDatafile interface)

Syntax

```
Procedure AddEntity (AName : WideString);
```

Description

This procedure adds a new entity for the datafile.

See also

IModelDatafile interface

FullPath method

(IModelDatafile interface)

Syntax

```
Function FullPath : WideString;
```

Description

This procedure fetches the full path of the data file part of the model.

See also

IModelDatafile interface

IModelDataFile Properties

EntityNames Property

(IModelDatafile interface)

Syntax

```
Property EntityNames[AnIndex : Integer] : WideString Read EntityName;
```

Description

This Entitynames property returns the indexed entity name for the datafile related to the model. This property is supported by the Entitynames method.

See also

IModelDatafile interface

EntityNames method.

IModelDatafileType Interface

Overview

The `IModelDatafileType` interface represents the data file type for the specified model. Simulation Model has three model types and thus three data files, PCB LIB has one model type and one data file, PCB3DLib has one model type and one data file and SI has one model type and one data file.

The IModelDatafileType interface is used by the IModelTypeManager

IModelDatafileType Methods and Properties Table

IModelDatafileType methods	IModelDatafileType properties
FileKind	
ExtensionFilter	
Description	
EntityType	
ModelType	
SupportsParameters	

See also

ReportIntLibData script from the `Examples\Scripts\Delphiscript Scripts\DXP_Scripts\` folder of Altium Designer installation

IModelDatafileType Methods

Description method

(IModelDatafileType interface)

Syntax

```
Function Description : PChar;
```

Description

This function returns the description string for this `IModelDatafiletype` interface depending on the model's data file type. Since Altium Designer supports four models and six model types:

Model Type (FileKind)	ExtensionFilter	DatafileType Description	Entity Type	Supports Parameters
MDL	*.MDL	Sim Model File	Sim Model	False
CKT	*.CKT	Sim Subcircuit File	Sim Subcircuit	False
LB	*.LB	SIMetrix Model Library File	SIMetrix Model	False
SIPinModelLibrary		SI Pin Model Library	SI Pin Model	False
PCBLIB	*.PCBLIB	Protel Footprint Library	Footprint	True
PCB3DLIB	*.PCB3DLib	PCB3D Model Library	PCB3D Model	False

Example

```
For I := 0 To ModelTypeMan.ModelDatafileTypeCount - 1 do
Begin
    ModelDatafileType := ModelTypeMan.ModelDatafileTypeAt(I);
    ShowMessage(
        ModelDatafileType.FileKind + #13 +
```

```
        ModelDatafileType.ExtensionFilter  + #13 +
        ModelDatafileType.Description + #13 +
        ModelDatafileType.EntityType  + #13 +
        ModelDatafileType.ModelType.Name  + #13 +
        BooleanToStr(ModelDatafileType.SupportsParameters));
End;
```

See also

IModelTypeManager interface

IModelDatafileType interface

EntityType method

(IModelDatafileType interface)

Syntax

```
Function EntityType : PChar;
```

Description

This function returns the Entity type string for this `IModelDatafiletype` interface depending on the model's data file type. Since Altium Designer supports four models and six model types:

Model Type (FileKind)	ExtensionFilter	DatafileType Description	Entity Type	Supports Parameters
MDL	*.MDL	Sim Model File	Sim Model	False
CKT	*.CKT	Sim Subcircuit File	Sim Subcircuit	False
LB	*.LB	SIMetrix Model Library File	SIMetrix Model	False
SIPinModelLibrary		SI Pin Model Library	SI Pin Model	False
PCBLIB	*.PCBLIB	Protel Footprint Library	Footprint	True
PCB3DLIB	*.PCB3DLib	PCB3D Model Library	PCB3D Model	False

Example

```
For I := 0 To ModelTypeMan.ModelDatafileTypeCount - 1 do
Begin
    ModelDatafileType := ModelTypeMan.ModelDatafileTypeAt(I);
    ShowMessage(
        ModelDatafileType.FileKind + #13 +
        ModelDatafileType.ExtensionFilter  + #13 +
        ModelDatafileType.Description + #13 +
        ModelDatafileType.EntityType  + #13 +
        ModelDatafileType.ModelType.Name  + #13 +
        BooleanToStr(ModelDatafileType.SupportsParameters));
End;
```

See also

IModelTypeManager interface

IModelDatafileType interface

ExtensionFilter method

(IModelDatafileType interface)

Syntax

```
Function ExtensionFilter : PChar;
```

Description

This function returns the extension filter string for this `IModelDatafiletype` interface depending on the model's data file type. Since Altium Designer supports four models and six model types:

Model Type (FileKind)	ExtensionFilter	DatafileType Description	Entity Type	Supports Parameters
MDL	*.MDL	Sim Model File	Sim Model	False
CKT	*.CKT	Sim Subcircuit File	Sim Subcircuit	False
LB	*.LB	SIMetrix Model Library File	SIMetrix Model	False
SIPinModelLibrary		SI Pin Model Library	SI Pin Model	False
PCBLIB	*.PCBLIB	Protel Footprint Library	Footprint	True
PCB3DLIB	*.PCB3DLib	PCB3D Model Library	PCB3D Model	False

Example

```
For I := 0 To ModelTypeMan.ModelDatafileTypeCount - 1 do
Begin
    ModelDatafileType := ModelTypeMan.ModelDatafileTypeAt(I);
    ShowMessage(
        ModelDatafileType.FileKind + #13 +
        ModelDatafileType.ExtensionFilter  + #13 +
        ModelDatafileType.Description + #13 +
        ModelDatafileType.EntityType  + #13 +
        ModelDatafileType.ModelType.Name  + #13 +
        BooleanToStr(ModelDatafileType.SupportsParameters));
End;
```

See also

IModelTypeManager interface

IModelDatafileType interface

FileKind method

(IModelDatafileType interface)

Syntax

```
Function FileKind : PChar;
```

Description

This function returns the FileKind string for this `IModelDatafiletype` interface depending on the model's data file type. Since Altium Designer supports four models and six model types:

Model Type (FileKind)	ExtensionFilter	DatafileType Description	Entity Type	Supports Parameters
MDL	*.MDL	Sim Model File	Sim Model	False
CKT	*.CKT	Sim Subcircuit File	Sim Subcircuit	False
LB	*.LB	SIMetrix Model Library File	SIMetrix Model	False
SIPinModelLibrary		SI Pin Model Library	SI Pin Model	False

PCBLIB	*.PCBLIB	Protel Footprint Library	Footprint	True
PCB3DLIB	*.PCB3DLib	PCB3D Model Library	PCB3D Model	False

Example

```
For I := 0 To ModelTypeMan.ModelDatafileTypeCount - 1 do
Begin
    ModelDatafileType := ModelTypeMan.ModelDatafileTypeAt(I);
    ShowMessage(
        ModelDatafileType.FileKind + #13 +
        ModelDatafileType.ExtensionFilter  + #13 +
        ModelDatafileType.Description + #13 +
        ModelDatafileType.EntityType  + #13 +
        ModelDatafileType.ModelType.Name  + #13 +
        BooleanToStr(ModelDatafileType.SupportsParameters));
End;
```

See also

IModelTypeManager interface

IModelDatafileType interface

ModelType method

(IModelDatafileType interface)

Syntax

```
Function ModelType : IModelType;
```

Description

This function returns the ModelType string for this `IModelDatafiletype` interface depending on the model's data file type. Since Altium Designer supports four models and six model types:

Model Type (FileKind)	ExtensionFilter	DatafileType Description	Entity Type	Supports Parameters
MDL	*.MDL	Sim Model File	Sim Model	False
CKT	*.CKT	Sim Subcircuit File	Sim Subcircuit	False
LB	*.LB	SIMetrix Model Library File	SIMetrix Model	False
SIPinModelLibrary		SI Pin Model Library	SI Pin Model	False
PCBLIB	*.PCBLIB	Protel Footprint Library	Footprint	True
PCB3DLIB	*.PCB3DLib	PCB3D Model Library	PCB3D Model	False

Example

```
For I := 0 To ModelTypeMan.ModelDatafileTypeCount - 1 do
Begin
    ModelDatafileType := ModelTypeMan.ModelDatafileTypeAt(I);
    ShowMessage(
        ModelDatafileType.FileKind + #13 +
        ModelDatafileType.ExtensionFilter  + #13 +
        ModelDatafileType.Description + #13 +
        ModelDatafileType.EntityType  + #13 +
        ModelDatafileType.ModelType.Name  + #13 +
```

```
        BooleanToStr(ModelDatafileType.SupportsParameters));
End;
```

See also

IModelTypeManager interface

IModelDatafileType interface

SupportsParameters method

(IModelDatafileType interface)

Syntax

```
Function SupportsParameters : Boolean;
```

Description

This function returns the SupportsParameters Boolean value for this `IModelDatafiletype` interface depending on the model's data file type. Since Altium Designer supports four models and six model types:

Model Type (FileKind)	ExtensionFilter	DatafileType Description	Entity Type	Supports Parameters
MDL	*.MDL	Sim Model File	Sim Model	False
CKT	*.CKT	Sim Subcircuit File	Sim Subcircuit	False
LB	*.LB	SIMetrix Model Library File	SIMetrix Model	False
SIPinModelLibrary		SI Pin Model Library	SI Pin Model	False
PCBLIB	*.PCBLIB	Protel Footprint Library	Footprint	True
PCB3DLIB	*.PCB3DLib	PCB3D Model Library	PCB3D Model	False

Example

```
For I := 0 To ModelTypeMan.ModelDatafileTypeCount - 1 do
Begin
    ModelDatafileType := ModelTypeMan.ModelDatafileTypeAt(I);
    ShowMessage(
        ModelDatafileType.FileKind + #13 +
        ModelDatafileType.ExtensionFilter  + #13 +
        ModelDatafileType.Description + #13 +
        ModelDatafileType.EntityType  + #13 +
        ModelDatafileType.ModelType.Name  + #13 +
        BooleanToStr(ModelDatafileType.SupportsParameters));
End;
```

See also

IModelTypeManager interface

IModelDatafileType interface

IModelEditor Interface

Overview

The `IModelEditor` interface represents the Model Editor hosted by a server which normally has a dialog that displays data about the model properties in Altium Designer. This `IModelEditor` interface is the front end for the actual implementation of a Model Editor for a specific model domain (PCB, Signal Integrity and other model types).

IModelEditor Methods and Properties Table

IModelEditor methods	IModelEditor properties
EditModel	
CreateDatafile	
StartingLibraryCompile	
FinishedLibraryCompile	
PrepareModel	
CreateServerModel	
GetExternalForm	
DrawModel	
GetEntityParameters	
SetDefaultModelState	
CrossProbeEntity	
DrawModelToMetaFile	

IModelEditor Methods

CreateDatafile method

(IModelEditor interface)

Syntax

```
Function CreateDatafile (ADatafilePath : PChar) : IModelDatafile;
```

Description

Example

See also

IModelEditor interface

CreateServerModel method

(IModelEditor interface)

Syntax

```
Function CreateServerModel (AModel : IComponentImplementation) : IServerModel;
```

Description

Example

See also

IModelEditor interface

CrossProbeEntity method

(IModelEditor interface)

Syntax

```
Procedure CrossProbeEntity (AEntityName : WideString;ADataFilePath : WideString);
```

Description

Example

See also

IModelEditor interface

DrawModel method

(IModelEditor interface)

Syntax

```
Procedure DrawModel (AExternalForm : IExternalForm;AModelName : PChar;ADataFilePath : PChar);
```

Description

Example

See also

IModelEditor interface

DrawModelToMetaFile method

(IModelEditor interface)

Syntax

```
Procedure DrawModelToMetaFile (Const AFileName : WideString;Const AModelName :
WideString;Const ADataFilePath : WideString;APaintColorMode : TPaintColorMode;APaintScaleMode
: TPaintScaleMode);
```

Description

Example

See also

IModelEditor interface

EditModel method

(IModelEditor interface)

Syntax

```
Function EditModel (SchModel : ISch_Implementation; SchComp : ISch_Component;IsLibrary :
Boolean) : Boolean;
```

Description

Example

See also

IModelEditor interface

FinishedLibraryCompile method

(IModelEditor interface)

Syntax

```
Procedure FinishedLibraryCompile;
```

Description

Example

See also

IModelEditor interface

GetEntityParameters method

(IModelEditor interface)

Syntax

```
Function GetEntityParameters (AEntityName : WideString; ADataFilePath : WideString) :
WideString;
```

Description

Example

See also

IModelEditor interface

GetExternalForm method

(IModelEditor interface)

Syntax

```
Function GetExternalForm : IExternalForm;
```

Description

Example

See also

IModelEditor interface

PrepareModel method

(IModelEditor interface)

Syntax

```
Function PrepareModel (AModel : IComponentImplementation) : Boolean;
```

Description

Example

See also

IModelEditor interface

StartingLibraryCompile method

(IModelEditor interface)

Syntax

```
Procedure StartingLibraryCompile;
```

Description

Example

See also

IModelEditor interface

SetDefaultModelState method

(IModelEditor interface)

Syntax

```
Function SetDefaultModelState (SchModel : ISch_Implementation;SchComp :
ISch_Component;IsLibrary : Boolean) : Boolean;
```

Description

Example

See also

IModelEditor interface

IModelType Interface

Overview

The `IModelType` interface represents the type used by a model linked in the Component. Each model has at least one data file type or entity type.

The `IModelDataFiletype` interface uses the `IModelType` interface

The `IModelTypeManager` interface uses the `IModelType` interface

IModelType Methods and Properties Table

IModelType methods	IModelType properties
Name	
Description	
ServerName	
PortDescriptor	
Editor	
Previewable	
Highlightable	

See Also

IModelDataFileType interface

IModelTypeManager interface

`Examples\Scripts\DXP_Scripts\` folder of Altium Designer installation.

IModelType Methods

Description method

(IModelType interface)

Syntax

```
Function Description : PChar;
```

Description

The function returns the description of the model type.

Model Type Description	Model Type Name	ServerName
Simulation	SIM	Sim
Signal Integrity	SI	SignalIntegrity
Footprint	PCBLIB	PCB
PCB3D	PCB3DLIB	PCB3D

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;

AModelType := IntMan.GetModelType(Libpath, AComponentIndex, AModelIndex)
ShowMessage(AModelType.Description);
```

See also

IModelType interface

Editor method

(IModelType interface)

Syntax

```
Function Editor : IModelEditor;
```

Description

This method returns the IModelEditor for this model type.

See also

IModelType interface

IModelEditor interface

IModelType.Description method

Name method

(IModelType interface)

Syntax

```
Function Name : PChar;
```

Description

The function returns the name of the model type supported by Altium Designer. The following model names supported by Altium Designer are:

Model Type Name	Model Type Description	ServerName
SIM	Simulation	Sim
SI	Signal Integrity	SignalIntegrity
PCBLIB	Footprint	PCB
PCB3DLIB	PCB3D	PCB3D

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;

AModelType := IntMan.GetModelType(Libpath, AComponentIndex, AModelIndex);
ShowMessage(AModelType.Name);
```

See also

IModelType interface

PortDescriptor method

(IModelType interface)

Syntax

```
Function PortDescriptor : PChar;
```

Description

The PortDescriptor

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;

AModelType := IntMan.GetModelType(Libpath, AComponentIndex, AModelIndex);
ShowMessage(AModelType.Descriptor);
```

See also

IModelType interface

Previewable method

(IModelType interface)

Syntax

```
Function Previewable : Boolean;
```

Description

This function returns a boolean value for the model that can be previewable. Simulation and Signal Integrity models are not highlightable or previewable and thus they don't have viewable document kinds.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;

AModelType := IntMan.GetModelType(Libpath, AComponentIndex, AModelIndex);
ShowMessage(BooleanToStr(AModelType.Previewable));
```

See also

IModelType interface

Highlightable method

ViewableDocKind method

ServerName method

(IModelType interface)

Syntax

```
Function ServerName : PChar;
```

Description

This function returns the Server Name associated with the model type.

ServerName	Model Type Name	Model Type Description
Sim	SIM	Simulation
SignalIntegrity	SI	Signal Integrity
PCB	PCBLIB	Footprint
PCB3D	PCB3DLIB	PCB3D

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;

AModelType := IntMan.GetModelType(Libpath, AComponentIndex, AModelIndex);
ShowMessage(AModelType.ServerName);
```

See also

IModelType interface

Highlightable method

(IModelType interface)

Syntax

```
Function Highlightable : Boolean;
```

Description

This function returns a boolean value for the model that can be highlightable (viewable on a document kind). Simulation and Signal Integrity models are not highlightable or previewable and thus they don't have viewable document kinds.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;
```

```
AModelType := IntMan.GetModelType(Libpath, AComponentIndex, AModelIndex);
ShowMessage(BooleanToStr(AModelType.Highlightable));
```

See also

IModelType interface

ViewableDocKind method

Previewable method

ViewableDocKind method

(IModelType interface)

Syntax

```
Function ViewableDocKind : PChar
```

Description

This function returns the name of the Document Kind that's viewable (related to the Highlightable method). Simulation and Signal Integrity models are not highlightable and thus they don't have document kinds.

Example

```
IntMan := IntegratedLibraryManager;
If IntMan = Nil Then Exit;

AModelType := IntMan.GetModelType(Libpath, AComponentIndex, AModelIndex);
ShowMessage(AModelType.ViewableDocKind);
```

See also

IModelType interface

Highlightable method

Previewable method

IServerModel Interface

Overview

The `IServerModel` interface represents the model set up by the server to be used by the integrated library server.

IServerModel Methods and Properties Table

IServerModel methods	IServerModel properties
Name	PortNames
PortCount	
PortName	
AddPort	
CheckSchPins	
CheckModelPins	

See also

IModelEditor interface

IServerModel Methods

AddPort method

(IServerModel interface)

Syntax

```
Procedure AddPort (AName : PChar);
```

Description

See also

IServerModel interface

Name method

(IServerModel interface)

Syntax

```
Function Name : PChar;
```

Description

The function gives the name for the Server Model.

See also

IServerModel interface

PortName method

(IServerModel interface)

Syntax

```
Function PortName (AnIndex : Integer) : PChar;
```

Description

Example

See also

IServerModel interface

PortCount method

(IServerModel interface)

Syntax

```
Function PortCount : Integer;
```

Description

This function returns the number of ports for this Server Model.

See also

IServerModel interface

CheckSchPins method

(IServerModel interface)

Syntax

```
Function CheckSchPins : Boolean
```

Description

Example

See also

IServerModel interface

CheckModelPins method

(IServerModel interface)

Syntax

```
Function CheckModelPins : Boolean;
```

Description

Example

See also

IServerModel interface

IServerModel Properties

PortNames property

(IServerModel interface)

Syntax

```
Property PortNames[AnIndex : Integer] : PChar Read PortName;
```

Description

Example

See also

IServerModel interface

IModelEditorSelectionListener Interface

Overview

IModelEditorSelectionListener methods

PinSelectionChanged

IModelEditorSelectionListener properties

See also

IHighlightedModelEditor Interface

Overview

IHighlightedModelEditor Methods and Properties Table

IHighlightedModelEditor methods

HighlightComponentPins

ShowSpecifiedPinsOnly

ShowPinsAsSelected

DrawModel_PinsSelected

RegisterListener

IHighlightedModelEditor properties

See also

IModelType interface

IHighlightedModelEditor Methods

HighlightComponentPins Method

(IHighlightedModelEditor interface)

Syntax

```
Procedure HighlightComponentPins(AExternalForm      : IExternalForm;
                                 APinNameList       : WideString;
                                 AHighlightColor    : TColor;
                                 ANonHighlightColor : TColor);
```

Description

Example

See also

IHighlightedModelEditor interface

ShowSpecifiedPinsOnly Method

(IHighlightedModelEditor interface)

Syntax

```
Procedure ShowSpecifiedPinsOnly (AExternalForm      : IExternalForm;
```

```
                                    APinNameList       : WideString);
```

Description

Example

See also

IHighlightedModelEditor interface

ShowPinsAsSelected Method

(IHighlightedModelEditor interface)

Syntax

```
Procedure ShowPinsAsSelected(AExternalForm      : IExternalForm;
                             APinNameList       : WideString);
```

Description

Example

See also

IHighlightedModelEditor interface

DrawModel_PinsSelected Method

(IHighlightedModelEditor interface)

Syntax

```
Procedure DrawModel_PinsSelected(AExternalForm      : IExternalForm;
                                 AModelName         : WideString;
                                 ADataFilePath      : WideString;
                                 APinNameList       : WideString);
```

Description

Example

See also

IHighlightedModelEditor interface

RegisterListener Method

(IHighlightedModelEditor interface)

Syntax

```
Procedure RegisterListener(AExternalForm : IExternalForm;
                           AListener     : IModelEditorSelectionListener);
```

Description

Example

See also

IHighlightedModelEditor interface

Integrated Library Enumerated Types

```
TLibraryType = (eLibIntegrated, eLibSource, eLibDatafile, eLibDatabase, eLibNone, eLibQuery);
```

Integrated Library Constants

```
cModelType_PCB   = 'PCBLIB';
cModelType_Sim   = 'SIM';
cModelType_PCB3D = 'PCB3DLib';
cModelType_PCAD  = 'PCADLib';
cModelType_SI    = 'SI';
```

Integrated Library Functions

```
Function ModelTypeManager         : IModelTypeManager;
Function IntegratedLibraryManager : IIntegratedLibraryManager;
Function DeviceSheetManager       : IDeviceSheetManager;
```

Revision History

Date	Version No.	Revision
22-Nov-2005	1.0	New product release
15-Dec-2005	1.1	Updated for Altium Designer 6
29-Jun-2006	1.2	Updated for Altium Designer 6.3
28-Feb-2008	1.3	Updated Page Size and updated API information.
20-Apr-2008	1.4	Updated path references.
4-Aug-2008	1.5	Updated API information.
01-Sep-2011	-	Updated template.