



# Integrated Library API Reference

---

## Summary

Technical Reference  
TR0136 (v1.2) Jun 29, 2006

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 interfaces for the Integrated Library objects in the Integrated Library Object Model.

### What are Interfaces?

Each method in the interface is implemented in the corresponding class. Interfaces 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 its 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, 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.

There are two main interfaces from the Integrated Library Object Model. To obtain the Integrated Library Manager interface that points to the Integrated Library manager object, invoke the **IntegratedLibraryManager** function in your script which returns you the **IIntegratedLibraryManager** interface. To obtain the model type manager, invoke the **ModelTypeManager** function in your script which returns you the **IModelTypeManger** interface..

### Example

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

Main Nexar Interfaces

IModelTypeManager interface

IIntegratedLibraryManager interface

### Script Examples

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

### See Also

Integrated Library **Overview**

Client Server Interfaces

Integrated Library API Reference

## Integrated Library API Reference

Nexar API Reference

PCB API Reference

Schematic API Reference

Work Space Manager API Reference

## Integrated Library 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.

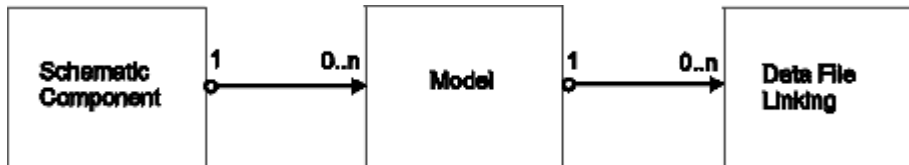
Each domain needs some information about each component, and also some way to map that information to the pins of the schematic component. Some of the domain information resides in model files, the format of which is typically pre-defined, examples of these includes IBIS, MDL and CKT files. Some of the 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 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.



### PCB Footprints

For the PCB footprints, the model and the data file are both the same.

### Simulation Models

With the simulation models, you can have a simulation model which is a 4ohm resistor for example, there is a simulation model here, but there is no information is coming from an external file, therefore, a no external file is needed for this as the resistor model is built from spice. 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.

### 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.

Note that a model can also be called an implementation. For each implementation there are parameters and data file links.

**See also**

IModelTypeManager interface

IIntegratedLibraryManager interface

## Integrated Library Interfaces

### IHighlightedModelEditor Interface

---

#### Overview

#### IHighlightedModelEditor methods

HighlightComponentPins

ShowSpecifiedPinsOnly

ShowPinsAsSelected

DrawModel\_PinsSelected

RegisterListener

See also

#### IHighlightedModelEditor properties

### IntegratedLibraryManager 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.

#### IIntegratedLibraryManager methods

#### IIntegratedLibraryManager properties

## ***Integrated Library API Reference***

AddRemoveLibraries  
AvailableLibraryCount  
AvailableLibraryPath  
AvailableLibraryType  
BrowseForComponent  
BrowseForComponentAndPart  
BrowseForComponentAndPartCheckDBLibs  
BrowseForComponentCheckDBLibs  
BrowseForDatafile  
ComponentHasModelOfType  
CreateIntegratedLibrary  
ExtractSources  
ExtractSourcesToPath  
FindDatafileInStandardLibs  
GetAvailableDBLibDocAtPath  
GetComponentCount  
GetComponentDatafileLocation  
GetComponentLocation  
GetComponentLocationFromDatabase  
GetComponentName  
GetDatabaseDatafileLocation  
GetDatafileEntityCount  
GetDatafilePath  
GetModelCount  
GetModelName  
GetModelType  
GetParametersForDBComponent  
GetSchLibPathForDBComponent  
GetSchLibRefForDBComponent  
InstalledLibraryCount  
InstalledLibraryPath  
InstallLibrary  
IsParameterDatabaseKey

MakeCurrentProject  
ModelCount  
ModelName  
ParseDatabaseKeys  
PlaceLibraryComponent  
UninstallLibrary

**See also**

## UninstallLibrary method

(IIntegratedLibraryManager interface)

**Syntax**

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

**Description**

This procedure removes the installed library from the Installed Libraries in Altium Designer.

**Example**

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

**See also**

IIntegratedLibraryManager interface  
InstallLibrary method

## 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**

**See also**

IIntegratedLibraryManager interface

## ModelCount method

(IIntegratedLibraryManager interface)

## ***Integrated Library API Reference***

### **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.

### **Example**

### **See also**

IIntegratedLibraryManager interface

## **InstallLibrary method**

(IIntegratedLibraryManager interface)

### **Syntax**

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

### **Description**

### **Example**

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

### **See also**

IIntegratedLibraryManager interface

UnInstallLibrary method

## **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**

### **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

### Example

### See also

IIntegratedLibraryManager interface

## FindDatafileInStandardLibs method

(IIntegratedLibraryManager interface)

### Syntax

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

### Description

### Example

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

## ***Integrated Library API Reference***

```
IntMan.InstallLibrary('C:\Program Files\Altium Designer 6
\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

## **ExtractSourcesToPath method**

(IIntegratedLibraryManager interface)

### **Syntax**

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

### **Description**

### **Example**

### **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;
                End;
            End;
        End.

```

**See also**

IIntegratedLibraryManager interface

**CreateIntegratedLibrary method**

(IIntegratedLibraryManager interface)

**Syntax**

```

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

```

**Description**

**Example**

**See also**

IIntegratedLibraryManager interface

**BrowseForDatafile method**

(IIntegratedLibraryManager interface)

**Syntax**

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

**Description**

This BrowseForDataFile procedure invokes the Browse for Models dialog.

**Example**

**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 BrowseForDataFile 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

**AddRemoveLibraries method**

(IIntegratedLibraryManager interface)

**Syntax**

```
Procedure AddRemoveLibraries;
```

**Description**

**Example**

**See also**

IIntegratedLibraryManager interface

**PlaceLibraryComponent method**

(IIntegratedLibraryManager interface)

**Syntax**

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

**Description**

**Example**

**See also**

IIntegratedLibraryManager interface

**MakeCurrentProject method**

(IIntegratedLibraryManager interface)

**Syntax**

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

**Description**

This procedure includes the current library into the current project.

**Example**

**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;
    End;
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**

IIIntegratedLibraryManager interface

InstalledLibraryPath method

AvailableLibraryPath method

AvailableLibraryCount method

**GetModelType method**

(IIIntegratedLibraryManager 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.

**Example****See also**

IIIntegratedLibraryManager interface

IModelType interface

**GetModelName method**

(IIIntegratedLibraryManager 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.

**Example**

**See also**

IIntegratedLibraryManager interface

**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.

**Example**

**See also**

IIntegratedLibraryManager interface

**GetDatafilePath method**

(IIntegratedLibraryManager interface)

**Syntax**

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

**Description**

**Example**

**See also**

IIntegratedLibraryManager interface

**GetDatafileEntityCount method**

(IIntegratedLibraryManager interface)

**Syntax**

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

**Description**

**Example**

**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 library.

### Example

### See also

IIntegratedLibraryManager interface

## GetComponentCount method

(IIntegratedLibraryManager interface)

### Syntax

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

### Description

This function retrieves the component count of components within the specified library.

### Example

### See also

IIntegratedLibraryManager interface

## ComponentHasModelOfType method

(IIntegratedLibraryManager interface)

### Syntax

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

### Description

### Example

### 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.

### 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

### 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 available library.

### 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

### See also

IIntegratedLibraryManager interface

## AvailableLibraryCount method

(IIntegratedLibraryManager interface)

### Syntax

```
Function AvailableLibraryCount : Integer;
```

### Description

The AvailableLibraryCount function determines the number of available libraries.

## 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

## See also

IIntegratedLibraryManager interface

AvailableLibraryPath method

AvailableLibraryType method

## IModelDataFile Interface

---

### Overview

The IModelDatafile interface represents the data 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

FullPath

EntityCount

EntityName

AddEntity

IModelDatafile properties

EntityNames

## See also

## EntityNames property

(IModelDatafile interface)

### Syntax

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

### Description

## Example

## See also

IModelDatafile interface

## EntityName method

(IModelDatafile interface)

### Syntax

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

### Description

### Example

### See also

IModelDatafile interface

## EntityCount method

(IModelDatafile interface)

### Syntax

```
Function EntityCount : Integer;
```

### Description

### Example

### See also

IModelDatafile interface

## AddEntity method

(IModelDatafile interface)

### Syntax

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

### Description

### Example

### See also

IModelDatafile interface

## FullPath method

(IModelDatafile interface)

### Syntax

```
Function FullPath : WideString;
```

**Description****Example****See also**

IModelDatafile interface

## IModelDatafileType interface

---

### IModelDatafileType Interface

**Overview**

The IModelDatafileType interface represents the data file as one of the models for that model type.

IModelDatafileType methods    IModelDatafileType properties

FileKind

ExtensionFilter

**Description**

EntityType

ModelType

SupportsParameters

**See also**

### Description method

(IModelDatafileType interface)

**Syntax**

```
Function Description : PChar;
```

**Description****Example****See also**

IModelDatafileType interface

### EntityType method

(IModelDatafileType interface)

**Syntax**

## ***Integrated Library API Reference***

```
Function EntityType : PChar;
```

### **Description**

### **Example**

### **See also**

IModelDatafileType interface

## **ExtensionFilter method**

(IModelDatafileType interface)

### **Syntax**

```
Function ExtensionFilter : PChar;
```

### **Description**

### **Example**

### **See also**

IModelDatafileType interface

## **FileKind method**

(IModelDatafileType interface)

### **Syntax**

```
Function FileKind : PChar;
```

### **Description**

### **Example**

### **See also**

IModelDatafileType interface

## **ModelType method**

(IModelDatafileType interface)

### **Syntax**

```
Function ModelType : IModelType;
```

### **Description**

### **Example**

**See also**

IModelDatafileType interface

**SupportsParameters method**

(IModelDatafileType interface)

**Syntax**

```
Function SupportsParameters : Boolean;
```

**Description****Example****See also**

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

EditModel

CreateDatafile

StartingLibraryCompile

FinishedLibraryCompile

PrepareModel

CreateServerModel

GetExternalForm

DrawModel

GetEntityParameters

SetDefaultModelState

CrossProbeEntity

DrawModelToMetaFile

IModelEditor properties

**See also**

## 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

## IModelEditorSelectionListener interface

---

**Overview**

**IModelEditorSelectionListener methods**

PinSelectionChanged

**IModelEditorSelectionListener properties**

**See also**

## **IModelType interface**

---

### **Overview**

The IModelType interface represents the model domain. Each model domain has at least one data file type or entity type.

### **IModelType methods**

Name

Description

ServerName

PortDescriptor

Editor

Previewable

Highlightable

### **IModelType properties**

### **See also**

## **Description method**

(IModelType interface)

### **Syntax**

```
Function Description : PChar;
```

### **Description**

### **Example**

### **See also**

IModelType interface

## **Editor method**

(IModelType interface)

### **Syntax**

```
Function Editor : IModelEditor;
```

### **Description**

### **Example**

### **See also**

IModelType interface

## Name method

(IModelType interface)

### Syntax

```
Function Name : PChar;
```

### Description

### Example

### See also

IModelType interface

## PortDescriptor method

(IModelType interface)

### Syntax

```
Function PortDescriptor : PChar;
```

### Description

### Example

### See also

IModelType interface

## Previewable method

(IModelType interface)

### Syntax

```
Function Previewable : Boolean;
```

### Description

### Example

### See also

IModelType interface

## ServerName method

(IModelType interface)

### Syntax

```
Function ServerName : PChar;
```

### Description

### Example

### See also

IModelType interface

## IModelTypeManager interface

---

### Overview

The IModelTypeManager interface is like a repository of available model types in Altium Designer. The IMP files are collected and processed by this manager. This manager uses IModelType and IModelDataType interfaces.

Invoke the ModelTypeManager function to fetch the IModelTypeManager interface.

#### IModelTypeManager methods

ModelTypeCount

ModelTypeAt

ModelTypeFromName

ModelTypeFromServerName

ModelDatafileTypeCount

ModelDatafileTypeAt

ModelDatafileTypeFromKind

#### IModelTypeManager properties

ModelTypes [AnIndex

ModelDatafileTypes[AnIndex

### See also

## ModelTypeFromServerName method

(IModelTypeManager interface)

### Syntax

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

### Description

### Example

### See also

IModelTypeManager interface

## ModelTypeFromName method

(IModelTypeManager interface)

### Syntax

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

### Description

### Example

### See also

IModelTypeManager interface

## ModelTypeCount method

(IModelTypeManager interface)

### Syntax

```
Function ModelTypeCount : Integer;
```

### Description

### Example

### See also

IModelTypeManager interface

## ModelTypeAt method

(IModelTypeManager interface)

### Syntax

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

### Description

### Example

### See also

IModelTypeManager interface

## ModelDatafileTypes[AnIndex property

(IModelTypeManager interface)

### Syntax

## ***Integrated Library API Reference***

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

### **Description**

### **Example**

### **See also**

IModelTypeManager interface

## **ModelDatafileTypeFromKind method**

(IModelTypeManager interface)

### **Syntax**

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

### **Description**

### **Example**

### **See also**

IModelTypeManager interface

## **ModelDatafileTypeCount method**

(IModelTypeManager interface)

### **Syntax**

```
Function ModelDatafileTypeCount : Integer;
```

### **Description**

### **Example**

### **See also**

IModelTypeManager interface

## **ModelDatafileTypeAt method**

(IModelTypeManager interface)

### **Syntax**

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

### **Description**

### **Example**

**See also**

IModelTypeManager interface

**ModelTypes [AnIndex property]**

(IModelTypeManager interface)

**Syntax**

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

**Description****Example****See also**

IModelTypeManager interface

**IServerModel interface**

---

**Overview**

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

## IServerModel methods

Name

PortCount

PortName

AddPort

CheckSchPins

CheckModelPins

## IServerModel properties

PortNames[AnIndex

**See also****AddPort method**

(IServerModel interface)

**Syntax**

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

**Description****Example**

**See also**

IServerModel interface

**Name method**

(IServerModel interface)

**Syntax**

```
Function Name : PChar;
```

**Description**

**Example**

**See also**

IServerModel interface

**PortNames property**

(IServerModel interface)

**Syntax**

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

**Description**

**Example**

**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

### Example

### 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

## 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;
```

# Index

---

## I

### Integrated Library API

AddEntity.....	18
AddPort.....	31
AddRemoveLibraries.....	10
AvailableLibraryCount.....	16
AvailableLibraryPath.....	16
AvailableLibraryType.....	15
BrowseForComponent.....	10
BrowseForComponentAndPart.....	9
BrowseForDatafile.....	9
CheckModelPins.....	33
CheckSchPins.....	32
ComponentHasModelOfType.....	15
CreateDatafile.....	21
CreateIntegratedLibrary.....	9
CreateServerModel.....	22
CrossProbeEntity.....	22
Description.....	19
DrawModel.....	22
DrawModelToMetaFile.....	23
EditModel.....	23
Editor.....	26
EntityCount.....	18
EntityName.....	17
EntityNames.....	17
EntityType.....	19
ExtensionFilter.....	20
ExtractSources.....	8
ExtractSourcesToPath.....	7
FileKind.....	20
FindDatafileInStandardLibs.....	6
FinishedLibraryCompile.....	23
FullPath.....	18

GetComponentCount.....	15
GetComponentDatafileLocation.....	6
GetComponentLocation.....	6
GetComponentName.....	14
GetDatafileEntityCount.....	14
GetDatafilePath.....	14
GetEntityParameters.....	24
GetExternalForm.....	24
GetModelCount.....	13
GetModelName.....	13
GetModelType.....	13
IModelDatafileType.....	19
IModelEditor.....	21
InstalledLibraryCount.....	12
InstalledLibraryPath.....	11
InstallLibrary.....	5
Integrated Library Constants.....	33
Integrated Library Enumerated Types.....	33
Integrated Library Functions.....	33
Integrated Library Overview.....	2
MakeCurrentProject.....	11
ModelCount.....	5
ModelDatafileTypeAt.....	30
ModelDatafileTypeCount.....	30
ModelDatafileTypeFromKind.....	29
ModelDatafileTypes[AnIndex.....	29
ModelName.....	4
ModelType.....	20
ModelTypeAt.....	29
ModelTypeCount.....	28
ModelTypeFromName.....	28
ModelTypeFromServerName.....	28
ModelTypes.....	30
Name.....	26

## ***Integrated Library API Reference***

PlaceLibraryComponent .....	11	Previewable.....	27
PortCount.....	32	ServerName .....	27
PortDescriptor .....	26	SetDefaultModelState .....	25
PortName.....	32	StartingLibraryCompile.....	24
PortNames .....	32	SupportsParameters .....	20
PrepareModel .....	24	UninstallLibrary .....	4

## 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

Software, hardware, documentation and related materials:

Copyright © 2007 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment. Altium, Altium Designer, Board Insight, Design Explorer, DXP, LiveDesign, NanoBoard, NanoTalk, P-CAD, SimCode, Situs, TASKING, and Topological Autorouting and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.