

Simio API Note: DirectConnect DataProvider

Dec 2019 (D.Houck)

Contents

Simio API Note: DirectConnect DataProvider	1
Overview	2
Using the Direct Connect	3
Modifying the Simio Model	3
Running the Test Sample	5
Source Contents	5
Appendix – The IModelHelper Interface.....	6
The IModelHelperAddIn Interface	10
IModelHelperContext	11
Appendix – Troubleshooting.....	12
Keyword not Supported ‘provider’	12
File was opened with warnings.....	12

Overview

This API Note describes the architecture and use of the Simio DirectConnect API extensions, which provides the ability to export Simio Planning data with an external SQL database.

Like the existing Simio Data Binding, the Direct Connect will assist with the data-driven interfacing. However, it is unlike the normal data-binding in that is specifically targeted to exporting Planning tables and logs to a SQL Server database, and can automatically perform the export upon certain events (such as the Model being saved).

Unlike other API extensions, DirectConnect employs a combinations of techniques, conventions, along with the Simio APIs to allow your model to run and then – at the completion of the run – automatically export the results to an external database.

This document describes:

1. How DirectConnect functions
2. How to setup it up in your Simio model, and
3. The underlying programming to make it work.

The use of DirectConnect requires a specifically constructed Simo model as well as the DirectConnect user extension. If the current features of the DirectConnect suit your needs you will not have to alter the .NET code, but the code is discussed later to give you a better understanding of the mechanism so that you can – if necessary – modify it to your own purposes.

Using the Direct Connect

This section describes how to use the DirectConnect extension. As mentioned before, the use of DirectConnect requires a specifically constructed Simio model as well as the DirectConnect user extension.

Toward this end, you must:

1. Modify your model by adding the tables to export to Simio tables created by DirectConnect.
2. Place the DirectConnect DLL in a place where Simio can find it.
3. Modify model properties to include a SQL Server database connect string.

When “exporting data” is mentioned, it is rather nebulous concept; for example, do we want to add the data to the end of a database table, or simply replace the existing data? And what if the data has a key? Do we want to replace the import records with the same key and append all others?

DirectConnect handles the common situations by providing you with prebuilt export actions.

Modifying the Simio Model

When a Model is loaded, DirectConnect takes the opportunity to make sure the Model has the necessary infrastructure that is needed. This includes the ability to communicate with the database (such as `ConnectionString`) and Simio tables that are needed for configuration.

When DirectConnect first runs it uses `DefineSchema` to add properties to the Model. These are:

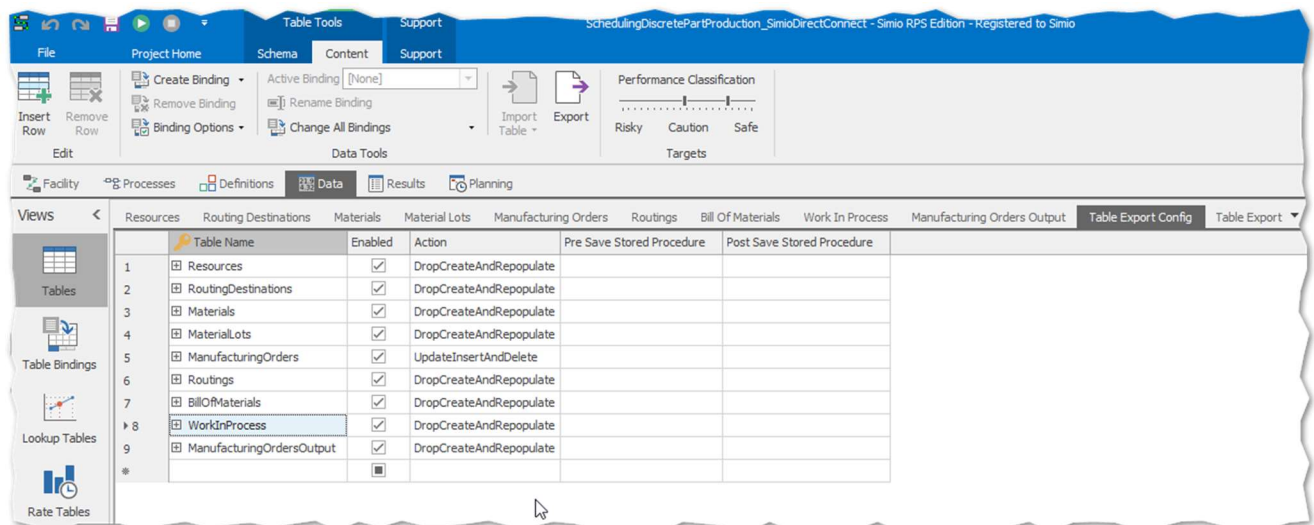
1. `ConnectionString` (for MS SQL Server)
2. `ConnectionTimeout` (in seconds)
3. `DateTimeFormat` (for saving dates to the database)

It also ties itself to these events:

1. `ModelSaved`
2. `ModelTablesImporting`, and
3. `ModelTablesExporting`

It is within the handling of this ModelSaved event that the Simio tables and logs are exported to SQL Server tables.

DirectConnect employs a small number of Simio Tables. It will use these tables to help make decisions when importing and/or exporting data. If these tables are not present, DirectConnect will create them for you.



For example, consider the table “Table Export Config”. This is a table of table names that will be exported. The Action column of this table indicates how the export will be accomplished.

These Actions are stored in a Simio NameList called “ExportTableActions” and “ExportLogActions”.

The Simio Tables created are:

1. TableExportConfig
2. TableExportExcludeForUpdate
3. LogExportConfig

Running the Test Sample

This sample uses a local SQL Server instance “.\SQLExpress” and the SQL database “SimioDirectConnect”, so the connection string will be something like:

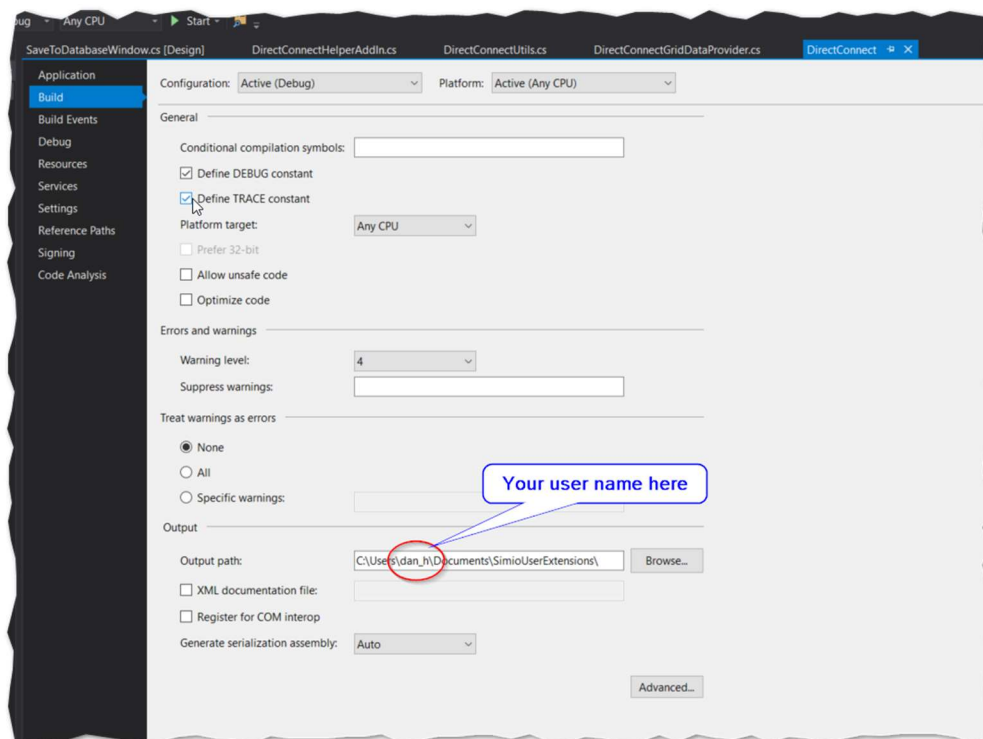
```
.\SQLExpress;Database=SimioDirectConnect;Trusted_Connection=True
```

The database can be completely empty, as DirectConnect will construct the tables if they are not present.

Source Contents

The DirectConnect C# project under the GitHub Source folder provides a runnable project that illustrates the use of DirectConnect with the supplied model in the Models folder.

There are strings in the build section of the project that must be changed to your username before running:



When the model is save, the DirectConnect will respond the “On Saved” event and write the tables and logs to the SQL Server.

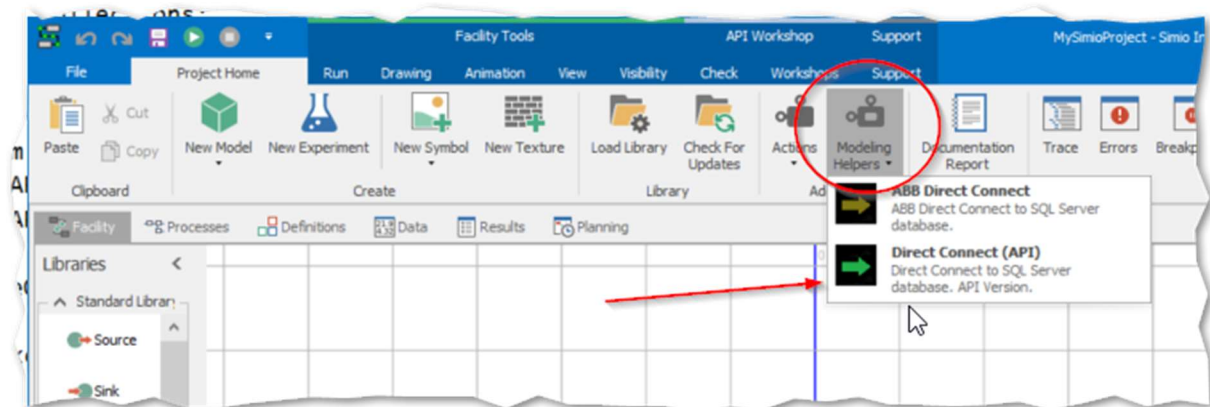
Appendix – The IModelHelper Interface

The IModelHelper interface was added to assist with the unique problems encountered when running unattended (without the UI) in a real-time production environment.

There are several features of this add-in:

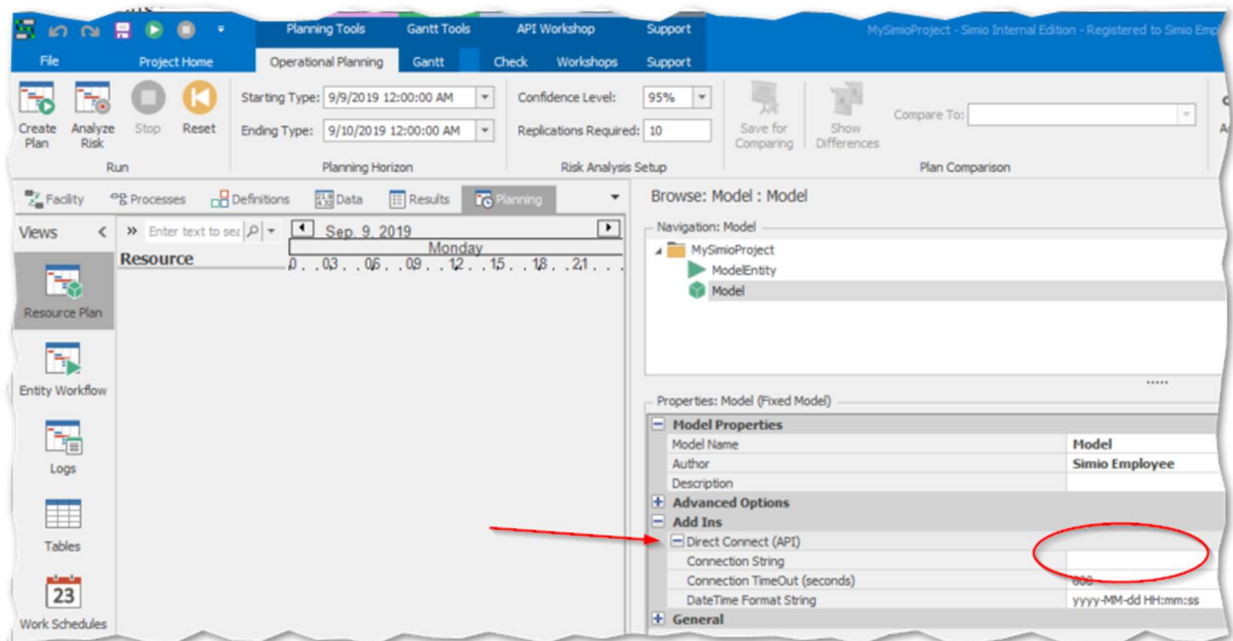
1. Modeling Helpers in the Project Home Ribbon / Facility tab.
2. The addition of Properties to the Model for items such as Connection String

As an example, when you create a new project, and select DirectConnect from the Modeling Helpers tab, the following occurs:



This creates Properties for the IModelHelperAddInSchema:

- ConnectionString (default is string.Empty)
- ConnectionTimeout (default is "600")
- DateTimeFormat (default is "yyyy-MM-dd HH:mm:ss")



The key to getting the data is the *Connection String*, which is an encoded set of instructions on how to connect to a provider of data (usually a relational database product like MS SQL Server, IBM DB, or Oracle). The Connection String is a weirdly formatted string that has a long and complicated history, but here is an abbreviated explanation:

Its components are separated by semicolons and are deciphered by a *Provider*, which is a piece of software that must be installed on your computer. How do you know what it looks like? Well, unless you are an uber-geek in the field, you don't. But you can go to the web and look up "Connection String" and easily find examples. Or, you can (if you are on Windows) create a blank file with the extension ".UDL" and double-click on it and you will likely be presented with a builder that has a list of Providers installed on your machine in the Provider tab. Each Provider requires different information, which will be shown on the Connection tab.

Below is shown the list of Providers on my computer, and the information needed for the Microsoft OLE DB Provider for SQL Server.

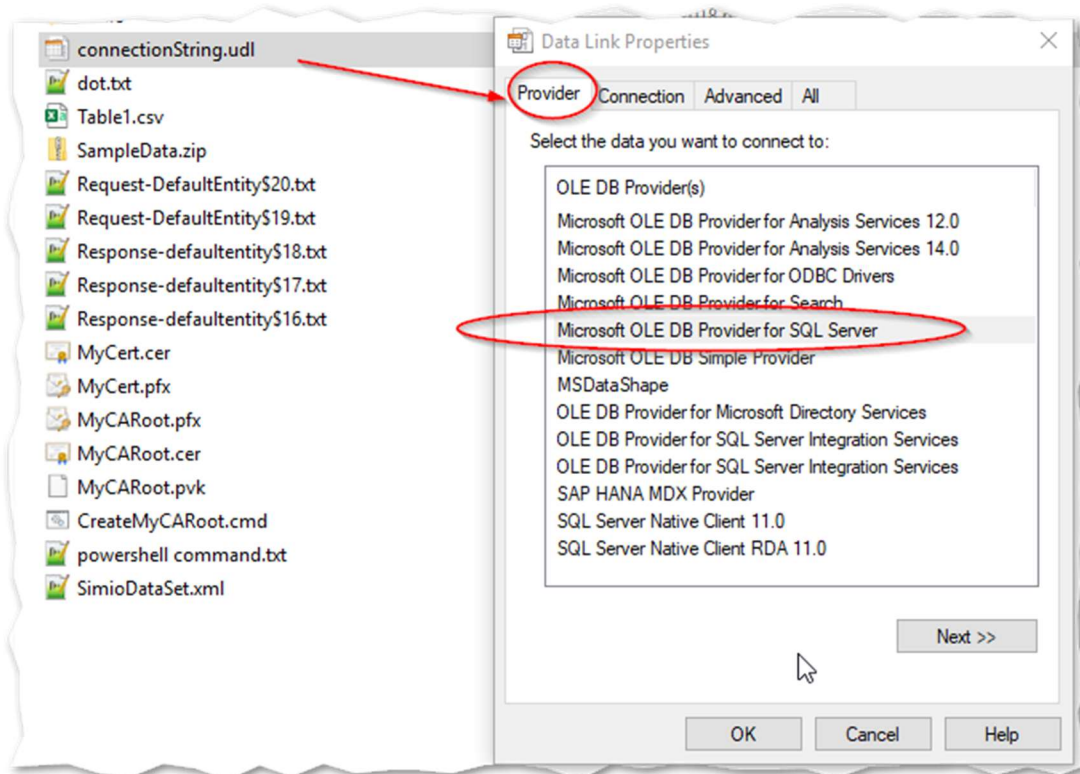


Figure 1 - Provider Tab

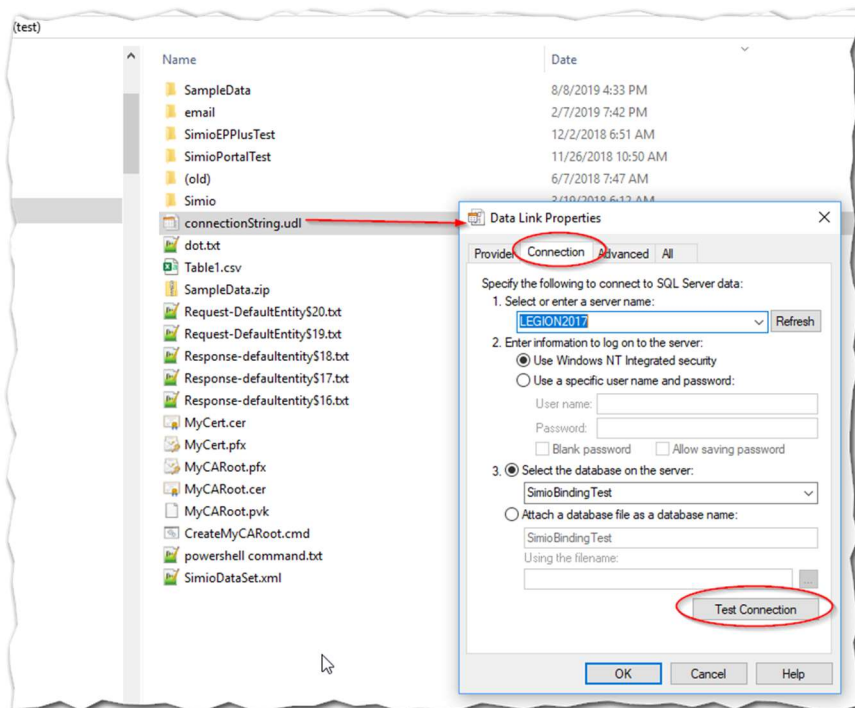


Figure 2 - Connection Tab for the Microsoft OLE DB Provider for SQL Server

After you enter and test this information, you can open the UDL file and see that a Connection String has been built:

```
[oledb]
```

```
; Everything after this line is an OLE DB initstring
```

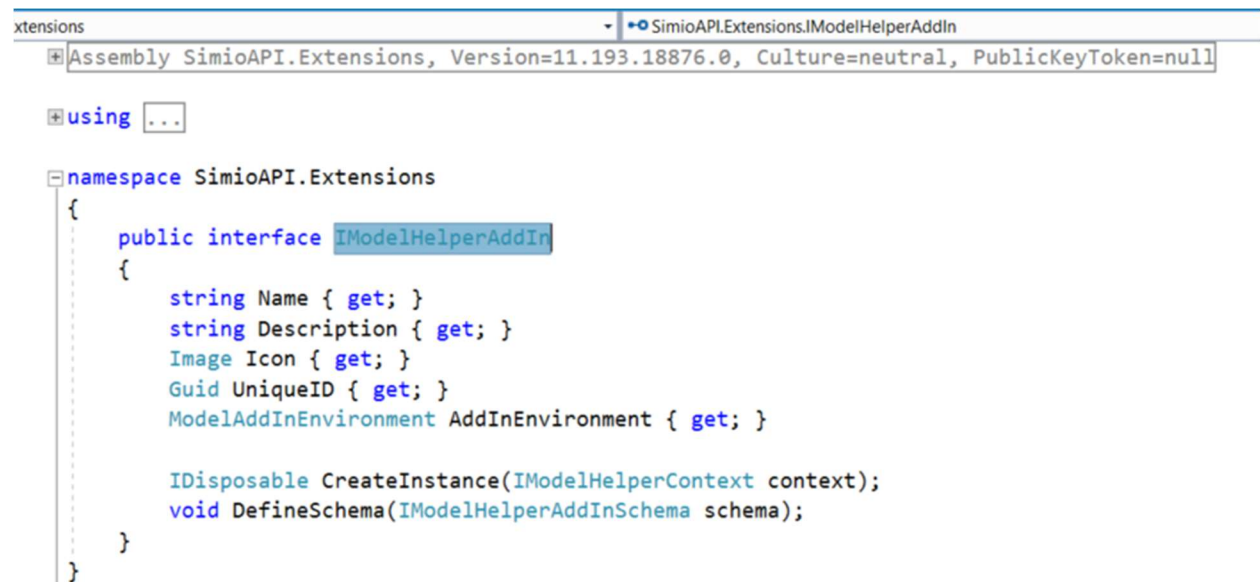
```
Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial  
Catalog=SimioBindingTest;Data Source=LEGION2017
```

There is also a named list called “ExportTableActions” created with defaults for how the data is to be exported:

- DropCreateAndRepopulate
- TruncateAndRepopulate
- UpdateAndInsert
- UpdateInsertAndDelete

The IModelHelperAddIn Interface

The interface provided by this AddIn contains the ubiquitous Name, Description, Icon, and UniqueID of other Add-Ins, but also contains a CreateInstance method that provides an IModelHelperContext to allow for interactions with the model (including setting up model events), as well as a DefineSchema method for setting up model properties.



```
xtensions | SimioAPI.Extensions.IModelHelperAddIn
+ Assembly SimioAPI.Extensions, Version=11.193.18876.0, Culture=neutral, PublicKeyToken=null

+ using ...

namespace SimioAPI.Extensions
{
    public interface IModelHelperAddIn
    {
        string Name { get; }
        string Description { get; }
        Image Icon { get; }
        Guid UniqueID { get; }
        ModelAddInEnvironment AddInEnvironment { get; }

        IDisposable CreateInstance(IModelHelperContext context);
        void DefineSchema(IModelHelperAddInSchema schema);
    }
}
```

Figure 3 - The Interface Definition for Simio's IModelHelperAddIn

IModelHelperContext

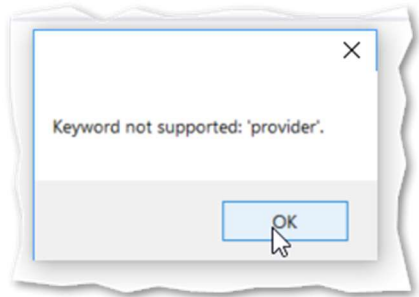
The Context that is used during the model run is defined by the Interface IModelHelperContext. As seen below, it has references to both the Project and the Model and defines a number of events that can be used within the API.

```
1  [Assembly: SimioAPI.Extensions, Version=11.198.19348.0, Culture=neutral, PublicKeyToken=null]
4
5  using System.Collections.Generic;
6
7  namespace SimioAPI.Extensions
8  {
9      public interface IModelHelperContext
10     {
11         ISimioProject Project { get; }
12         IModel Model { get; }
13         string ProjectFileName { get; }
14         ModelAddInEnvironment AddInEnvironment { get; }
15         IEnumerable<IAddInPropertyValue> PropertyValues { get; }
16
17         event ModelSavedDelegate ModelSaved;
18         event ModelPublishedDelegate ModelPublished;
19         event ModelPlanRunStartingDelegate ModelPlanRunStarting;
20         event ModelPlanRunEndedDelegate ModelPlanRunEnded;
21         event ModelTablesImportingDelegate ModelTablesImporting;
22         event ModelTableImportingDelegate ModelTableImporting;
23         event ModelTableImportedDelegate ModelTableImported;
24         event ModelTablesImportedDelegate ModelTablesImported;
25     }
26 }
```

Figure 4 - IModelHelperContext Interface

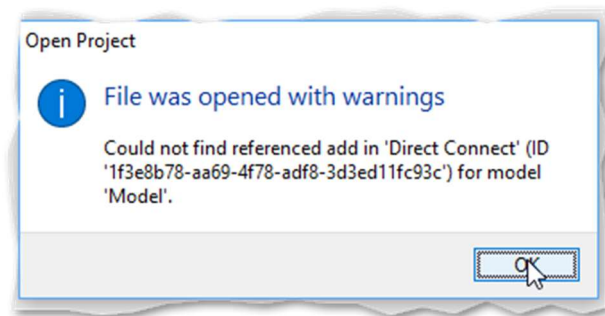
Appendix – Troubleshooting

Keyword not Supported 'provider'



The connect string is probably incorrect, or you haven't installed the correct database provider.

File was opened with warnings



This likely occurs because the DirectConnect DLL is not being found by the Simio model. It should be in one of the standard locations (such as in your Documents\SimioUserExtensions folder).

Facility Processes Definitions **Data** Results Planning

Views < Resources Routing Destinations Materials Material Lots Manufacturing Orders Routings Bill Of Materials Manufacturing Orders Output Table Export Config Table Export Exclur

Tables

Lookup Tables

Rate Tables

23

Work Schedules

Table Name	Enabled	Action	Pre Save Stored Procedure	Post Save Stored Procedure
1 Resources	<input checked="" type="checkbox"/>	UpdateInsertAndDelete		
2 RoutingDestinations	<input checked="" type="checkbox"/>	TruncateAndRepopulate		
3 Materials	<input checked="" type="checkbox"/>	UpdateInsertAndDelete		
4 MaterialLots	<input checked="" type="checkbox"/>	TruncateAndRepopulate		
5 ManufacturingOrders	<input checked="" type="checkbox"/>	UpdateInsertAndDelete		
6 Routings	<input checked="" type="checkbox"/>	UpdateInsertAndDelete		
7 BillOfMaterials	<input checked="" type="checkbox"/>	TruncateAndRepopulate		
8 WorkInProgress	<input checked="" type="checkbox"/>	TruncateAndRepopulate		
9 ManufacturingOrdersOutput	<input checked="" type="checkbox"/>	TruncateAndRepopulate		
*	<input checked="" type="checkbox"/>			

Facility Processes Definitions **Data** Results Planning

Views < Materials Material Lots Manufacturing Orders Routings Bill Of Materials Manufacturing Orders Output Table Export Config Table Export Exclude For Update Work

Tables

Lookup Tables

Table Name	Column Name
1 ManufacturingOrders	OrderStatus
*	

File Project Home Schema Content Check Workshops Support

Insert Row Remove Row Edit

Create Binding Remove Binding Binding Options

Active Binding [Default]

Rename Binding Change All Bindings

Data Tools

Import Table Export

Performance Classification

Risky Caution Safe

Append Tables Add-1...

Facility Processes Definitions **Data** Results Planning

Views < Materials Material Lots Manufacturing Orders Routings Bill Of Materials Manufacturing Orders Output Table Export Config Table Export Exclude For Update Work In Process

Tables

Lookup Tables

Order Id	Current Route Number	Current Resource	Fraction Of Setup Completed	Completed Quantity	AccruedCost (Default Currency)
1 Order01	20	Weld1	1	5	970
2 Order03	20	Weld2	1	3	1020
3 Order04	10	Cut1	1	2	1050
4 Order05	10	Cut2	0.5	0	1130
*					

100 %

Results Messages

BOMKey	OperationKey	MaterialCode	RequiredQty	WithdrawnQty	UoM	LTOffset	OperationLTOffset	PeggedMaterial	LastUpdateDate	LastUpdateF
1 210001755232-10.0-0041	210001755232-10.0	WT-142040	1	0	NULL	0	0	1	2019-02-01 14:55:48.0000000	C:\n\T\LOIP
2 210001755232-10.0-0042	210001755232-10.0	WI-142040	1	0	NULL	0	0	1	2019-02-01 14:55:48.0000000	C:\n\T\LOIP
3 210001755237-10.0-0006	210001755237-10.0	OS-142100	1	0	NULL	0	0	1	2019-02-01 14:55:48.0000000	C:\n\T\LOIP
4 210001755237-10.0-0008	210001755237-10.0	MI-142040	1	0	NULL	0	0	1	2019-02-01 14:55:48.0000000	C:\n\T\LOIP
5 210001755237-10.0-0009	210001755237-10.0	MT-142040	1	0	NULL	0	0	1	2019-02-01 14:55:48.0000000	C:\n\T\LOIP
6 210001755237-10.0-0010	210001755237-10.0	KS-142040	1	0	NULL	0	0	1	2019-02-01 14:55:48.0000000	C:\n\T\LOIP
7 210001843742-10.0-0041	210001843742-10.0	WT-142040	1	0	NULL	0	0	1	2019-02-01 14:55:48.0000000	C:\n\T\LOIP