User Guides

In this section:

Install Guide

How to install TMS FMX Grid Excel bridge.

User Guide

How to use TMS FMX Grid Excel bridge.

TMS FMX Grid Excel bridge Install Guide

To install TMS FMX Grid Excel bridge,you need to install first [TMS FMX UI Pack]https://www.tmssoftware.com/site/tmsfmxpack.asp) and TMS FlexCel.

IMPORTANT

Whenever you reinstall or update either FlexCel or FNC UI Pack, you need to reinstall the bridges, even if the bridges themselves didn't change.

Once you have both dependencies installed, you can now go through the setup and install the bridges.

TMS FMX Grid Excel bridge User Guide

Introduction

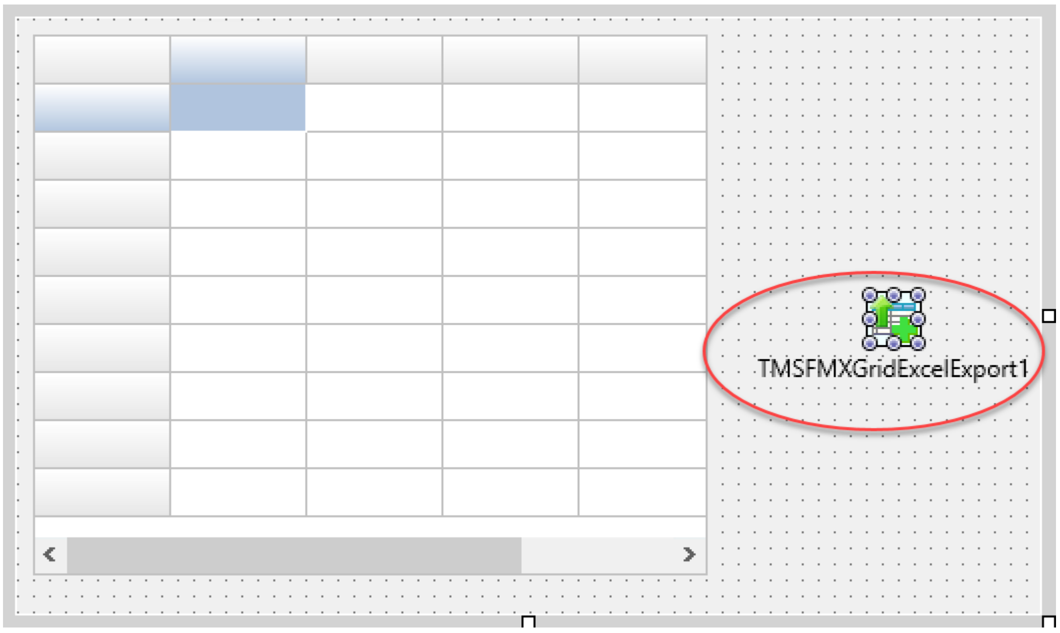
TMS FMX Grid Excel bridge is a distribution with two components: TTMSFMXGridExcelExport and TTMSFMXGridExcelImport.

You can use TTMSFMXGridExcelImport to import an xls or xlsx file to a TMS grid, and you can use TTMSFMXGridExcelExport to export a TMS grid to xls, xlsx, HTML, or PDF. The components use FlexCel Studio to read and write the Excel files, and don't require Excel to be installed in the machine.

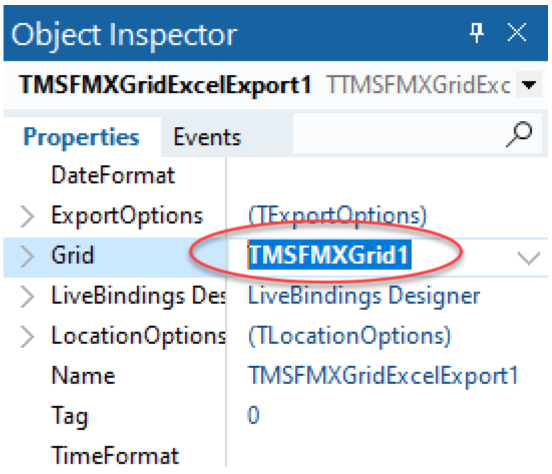
Exporting a grid to Excel

Getting started

Exporting a grid is simple. First, make sure to drop a TTMSFMXGridExcelExport component in the form that has the grid you wish to export:



After that, associate the Grid to the TTMSFMXGridExcelExport in the object inspector:



DateFormat	
ExportOptions	(TExportOptions)
Grid	TMSFMXGrid1
LiveBindings Des	LiveBindings Designer
LocationOptions	(TLocationOptions)
Name	TMSFMXGridExcelExport1
Tag	0
TimeFormat	

Finally, drop a button and write the following code:

```
TMSFMXGridExcelExport1.Export('test.xlsx');
```

If everything went right, you should have created an xlsx file from the grid.

Customizing the export

The single-line export methods are the easiest to use. Just write a line of code, check the export option properties in the TTMSFMXGridExcelExport component, and you are ready to go.

But sometimes you need more control over the generated files. Let's imagine for example that you wanted to export the grid into an existing xlsx file that is already customized with your branding. Or that you want to change things not available on the grid, like for example add some conditional formats. For those cases, you can export the grid directly into a FlexCel's TExcelFile object, using code like this:

```
uses ..., FlexCel.Core, FlexCel.XlsAdapter;
...
procedure TExampleForm.ExportGridToXlsFileObject;
var
  xls: TExcelFile;
  Ranges: TArray<TXlsCellRange>;
  Rules: TArray<IConditionalFormatRule>;
  CellIsRule: IConditionalCellIsRule;
  CFDef: TConditionalFormatDefStandard;
begin
  //Open a template with existing branding. We could also create a new file.
  xls := TXlsFile.Create('..\..\MyTemplate.xlsx', true);
  try
    //Export into the 3rd row of the existing template.
    TMSFMXGridExcelExport1.LocationOptions.XlsStartRow := 3;

    //Export the grid to the xls object.
    //We will export insert the rows into an existing template
    TMSFMXGridExcelExport1.Export(xls, TInsertInSheet.InsertRows);

    //Now you can do extra stuff with the xls file.
    //Adding rows, setting options, adding sheets
    //And exporting other grids. The code below
    //was generated with APIMate, and adds a conditional format
    //for column F that will paint the cell red if it is bigger than 3.

    Ranges := TArray<TXlsCellRange>.Create(
      TXlsCellRange.Create(1, 6, 1048576, 6)
    );

    Rules := nil;  //SetLength will resize the array in place. We set it to nil
first to create a new array.
    SetLength(Rules, 1);
    CellIsRule := TConditionalCellIsRule_Create(1, false, TConditionType.GreaterT
han, '=3', '');
    CFDef := CellIsRule.FormatDef;
    CFDef.ApplyFont.Color := true;
    CFDef.Font.Color := TExcelColor.FromArgb($9C, $00, $06);
    CFDef.ApplyFill.BgColor := true;
    CFDef.Fill.BgColor := TExcelColor.FromArgb($FF, $C7, $CE);
    Rules[0] := CellIsRule;

    xls.AddConditionalFormat(TConditionalFormat.Create(Ranges, Rules, false));


    // Finally save the xls object to a real file.
    xls.Save('MyFile.xlsx');

  finally
    xls.Free;
  end;
end;
```

Once you have a TExcelFile object with the grid, you can use the full FlexCel functionality to modify that object as you want.

NOTE
Always remember that you can use APIMate (a tool that is installed with FlexCel) to figure out exactly how to write the code to customize the TExcelFile object. To create the code above that sets the conditional formats, we just created the conditional format in Excel, then opened the file in APIMate and copied the code.

Exporting a grid to PDF or HTML

You can also export to PDF or HTML, using the methods TTMSFMXGridExcelExport.ExportPdf and TTMSFMXGridExcelExport.ExportHtml

Those methods will export to xlsx under the hood and then export the generated xlsx file to PDF or HTML. Same as when exporting to Excel, you can choose between exporting with one line, or using the corresponding FlexCel objects to allow further customization. The linked documentation in the methods contains code examples on how to use them.

Importing an Excel file

Importing an Excel file isn't conceptually different from exporting it. In this case, you drop a TTMSFMXGridExcelImport component instead of TTMSFMXGridExcelExport, and then call TTMSFMXGridExcelImport.Import.

As was the case with exporting, you have two options to import a file. You can use a one-liner:

```
  TMSFMXGridExcelImport1.Import('test.xlsx', 'my sheet');
```

Or you can import an existing TExcelFile object:

```
uses ..., FlexCel.Core, FlexCel.XlsAdapter;
...
procedure TExampleForm.XlsxObjectImport;
var
  xls: TExcelFile;
begin
  //We will start by opening the file we want to import
  xls := TXlsFile.Create('..\..\MyTemplate.xlsx', true);
  try
    //You can manipulate the xls file before importing it.

    //Select the sheet we want to import.
    xls.ActiveSheetByName := 'My Sheet';

    //Convert the formulas to values, so there aren't formulas in the
    //imported file. Also recalculate the sheet before converting
    //(recalculation is only needed if the original file wasn't calculated)
    xls.ConvertFormulasToValues(false, true);

    TMSFMXGridExcelImport1.Import(xls);
  finally
    xls.Free;
  end;
end;
```

Differences between a TMS Grid and Excel

Missing features

In an ideal world, you could import an xlsx file into the grid, then export the grid back to a separate xlsx file, and both files would be identical. But as you can guess, this is not the case, except maybe for very simple files. Imagine, for example, that the file we are importing has a macro. Once the file is imported into the grid, the macro will be gone. And when we export the file back to xlsx, the new file won't have the macro.

When we import the file into the grid, we will lose any features in the file that don't have a corresponding grid feature. If the file has a chart, the grid won't have it. It is the same for pivot tables, conditional formats, and another thousand things available in Excel but not in the grid. And when we export the grid into a file, we have the reverse problem. Say you have a fixed row in the middle of the grid. You can't have fixed rows in the middle when in Excel, so we can't export that feature.

NOTE

If you want to read a file, modify it and save it back, and you don't need to display it, you will be much better off using FlexCel alone. FlexCel won't do an xlsx->grid->xlsx conversion, and most of the stuff in the original file will be in the modified file. Only use TTMSFMXGridExcelImport and TTMSFMXGridExcelExport if you need to show the grid to the final user.

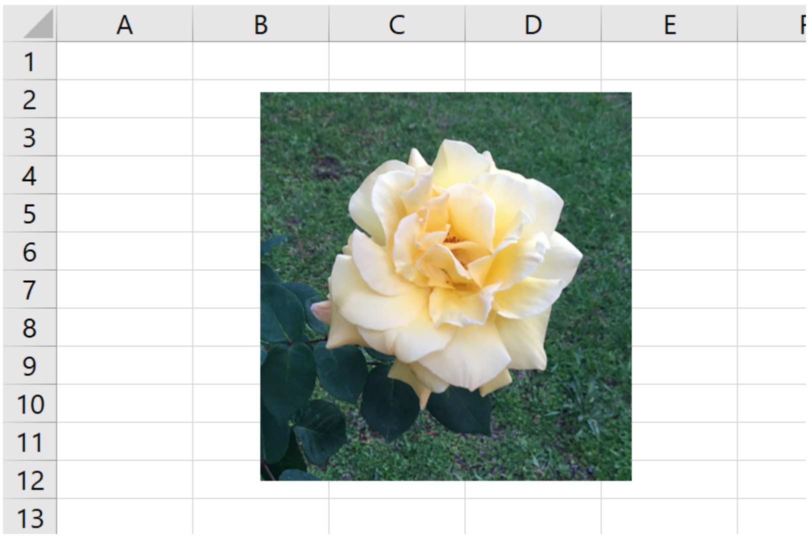
Features that behave differently

Sometimes, both the grid and Excel have a feature, but they aren't exactly compatible. In those cases, the bridge components will try to make their best effort to translate the feature from the grid to Excel or vice-versa, but sometimes it might need additional help. Here we will discuss some of those.

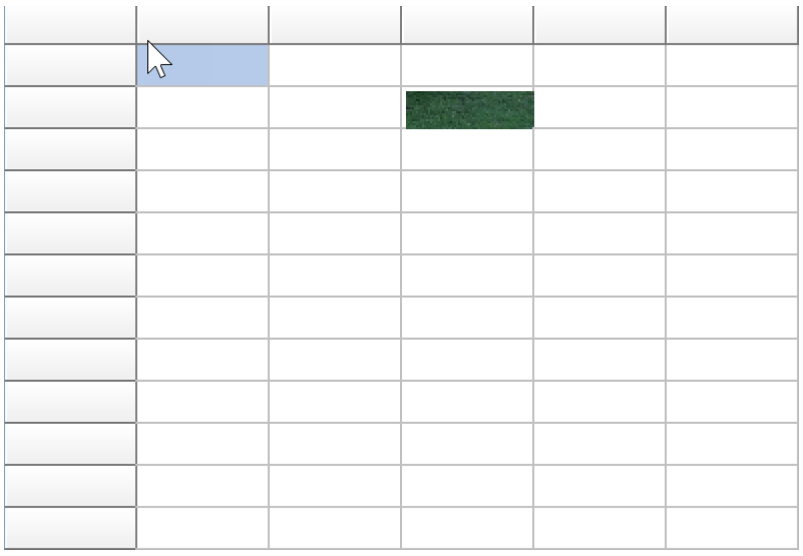
Images

Both Excel and the grid support images, but they behave differently. In Excel, images float over the cells. In the grid, images are inside a cell and can't span to a different cell.

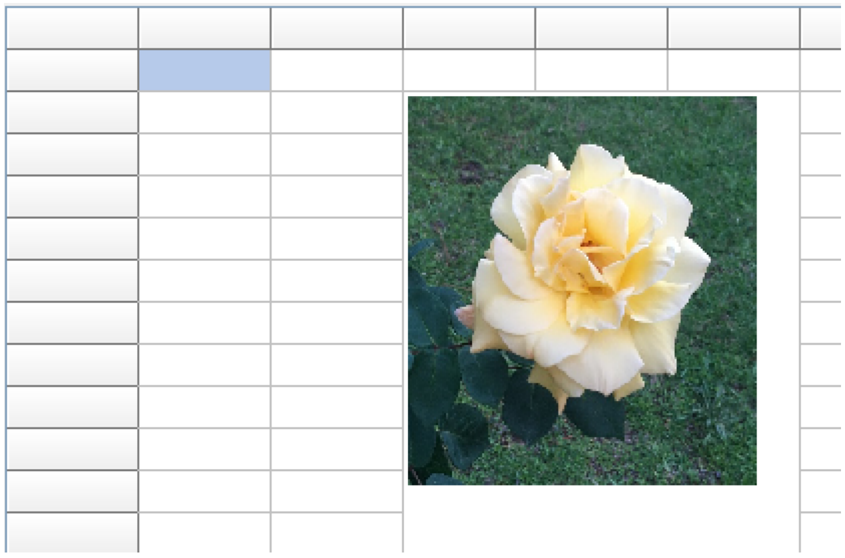
So if you have to import this image from Excel:



we would end up with the following:



To try avoiding this problem, TTMSFMXGridExcelImport will merge the cells beneath the image:



But of course, this is not a 1-to-1 translation. If there were data in any of the cells we had to merge; those cells would be lost. And when you save the file back to xlsx, it will have merged cells behind the image, even if the original file didn't.

Cell Formats

By default, TTMSFMXGrid uses Delphi format strings to format numbers. For example, if you have the number 10 and use the format "%.2f", it will display as "10.00". But in Excel, to show the number 10 as "10.00", you need to use the format string "0.00".

TTMSFMXGridExcelExport doesn't know how to convert "Delphi format strings" to "Excel format strings", or vice-versa. You can use two ways to get your exported data formatted as it is on the grid.

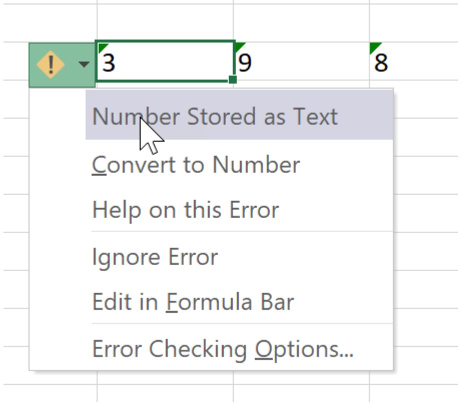
1. You can use the event TTMSFMXGridExcelExport.OnExportCell and manually specify the "Excel format" used for every cell. If the cell is formatted as "%.2f", you can say, in the event, that the Excel format used should be "0.00".

2. **Currently only for TMS AdvStringGrid**: You can set the property `AdvStringGrid.FormatType := vtExcel;` and use Excel formatting in the grid. Instead of specifying "%.2f" in the grid, you set the format to "0.00". TTMSFMXGridExcelExport will realize that you are using vtExcel, and pass that format to Excel when saving the file.

3. Finally, and as a last resort, you could export everything as strings. To do so you would use the property TExportOptions.CellsAsStrings

Exporting as strings is the simplest way to get the exact formatting as the grid, but it comes with significant caveats.

- First, the Excel cells won't have actual numbers. If in cell A1 you write the string "30", and in cell A2 you write "=Sum(A1)" the result will be 0, not 30.
- Second, you might run into localization issues. If you write the string "3,000", this might mean 3 or 3000 depending on the decimal separator used in the country. While if you stored the number 3000 in the cell, it will always be 3000 and will display with "," or "." depending on the machine locale.
- Finally, Excel will warn with a green triangle that the number is stored as text:



Cell Margins

TTMSFMXGrid has margins around the cells, while Excel has not. So if a cell is 20 pixels in Excel, to fit the text in the grid, we might need 26 pixels. But now the column is 26 pixels wide, instead of the original 20. If you keep repeating import/export cycles, the column width will grow indefinitely. You can control the cell margin behavior with the property TExportOptions.CellMargins

Hyperlinks

In Excel, hyperlinks apply to the whole cell. In the grid, hyperlinks apply to the text. This means that you can't have more than one link per cell in Excel, while you can have multiple links inside a single cell in the grid. If a cell contains more than one link, only the first will be exported to Excel.

About

This documentation is for TMS FMX Grid Excel bridge v3.1.0

In this section:

What's new

Everything that changed since the last release.

Copyright information

Information about the third party copyrights and licenses of code used by TMS FMX Grid Excel bridge.

What's new in TMS FMX Grid-Excel Bridge

New in v 3.1 - September 2021

- **Rad Studio 11 support.** Added Rad Studio 11 Support.

New in v 3.0.1 - April 2021

- **BugFix.** The installer wouldn't install in macOS 64bit

New in v 3.0

- **Unified versioning.** Now we have a single versioning between VCL/FMX/FNC Grid-Excel bridges
- **New documentation system.** There are new and improved documentation and examples.

New in v 2.3.2

- **Bug Fix.** Fixed setup errors.

New in v 2.3.1

- **Bug Fix.** The installer could throw an exception when installing in old Delphi versions.

New in v 2.3

- **Support for Rad Studio 10.4 Sydney.** Now the setup will install in Rad Studio 10.4 Sydney.

New in v 2.2.1

- **Fixed install error.** The setup could fail to find the component pack even if it was installed.

New in v 2.2

- **Support for Rad Studio 10.3 Rio.** Now the setup will install in Rad Studio 10.3 Rio.

New in v 2.1

- **Support for Rad Studio 10.2 Tokyo.** Now the setup will install in Rad Studio 10.2 Tokyo.

New in v 2.0

- **Adapted for TMS UI Pack 3.5.** The component TTMSFMXFlexCelPDFRenderLib for exporting natively to pdf using FlexCel was removed. Now you can use the new pdf engine in the pack directly.

New in v 1.9

- **Setup modified to install all supported platforms.** Setup used to install only packages for win32 and the other platforms had to be manually installed. Now they are also installed in the setup.

New in v 1.8

- **Rad Studio 10.1 Berlin support.** Added support for Rad Studio 10.1 Berlin.

New in v 1.7

- **Rad Studio 10 Seattle Support.** Added support for Rad Studio 10 Seattle.
- **Improved image exporting.** The size of the images could be wrong in some cases. Now we also export the images if they are not in a BitmapContainer.
- **Bug Fix.** Images could be exported in the wrong sizes sometimes.

New in v 1.6

- **XE8 Support.** Added support for Rad Studio XE8.

New in v 1.5

- **XE7 Support.** Added support for Rad Studio XE7.

New in v 1.4

- **XE6 Support.** Added support for Rad Studio XE6.

New in v 1.3

- **XE5 Support.** Added support for Rad Studio XE5.

New in v 1.2.1

- **Bug Fix.** C++ Builder header files required explicit namespaces. Now namespaces are implicit.

New in v 1.2

- **XE4 Support.** Rad Studio XE4 is now supported.

New in v 1.1

- **Export grid headers and footers.** Now when ExportPrintSettings is true, the grid title will be exported as headers or footers in Excel.

New in v 1.0

- **First release.** This is the first release.

Copyright information