

# TMS Planner for FireMonkey
# DEVELOPERS GUIDE

March 2016

Copyright © 2015 - 2016 by tmssoftware.com bvba

Web: http://www.tmssoftware.com
Email: info@tmssoftware.com

## Index

Introduction ..... 4
Organization ..... 5
Modes ..... 7
TimeLine ..... 16
  Display configuration ..... 16
  Appearance ..... 20
Positions / Resources ..... 22
  Display configuration ..... 22
  Appearance ..... 27
Groups ..... 28
  Display configuration ..... 28
  Appearance ..... 29
Grid ..... 32
  Display configuration ..... 32
  Appearance ..... 32
Current time indication ..... 34
Items (events) ..... 36
  Default item ..... 44
  HTML formatted text ..... 46
  Item Linking ..... 47
  Overlapping items ..... 51
  Appearance ..... 53
Interaction ..... 54
  Items ..... 54
  Selection / navigation ..... 57
  Inserting new items ..... 57
Editing ..... 60
Databinding ..... 62
Customization ..... 64
  Setting a specific inactive timeslot ..... 64
  Adding an icon to an item based on the conflict state ..... 67
  Changing the color for a specific timeline unit ..... 70
Styling ..... 71
Demos ..... 73
  Overview ..... 73
  Editing ..... 74
  Custom timeline ..... 75
  vCal adapter ..... 76
  Cloud adapter ..... 77
  Database adapter ..... 79

Properties ..... 81
Events ..... 93
Procedures and functions ..... 96
General FireMonkey component usage guidlines ..... 100
  Visual part ..... 100
  Non-visual part ..... 101
  Naming convention ..... 101
  Styling ..... 102
  Components ..... 105
TMS Mini HTML rendering engine ..... 106

## Introduction

The TMS Planner for FireMonkey offers a wide range of features to enhance your planning and scheduling applications for the Embarcadero cross-platform framework FireMonkey. From simple person PIM applications to schedulers of activities for a group of persons, time planning for resources such as hotel rooms, car rental, university courses and so much more.

It is built from the ground up with a very high customizability and supports a set of pre-defined single-resource views such as day time view, day period view, half-day period view, month view and multi-month view. A multi-resource view is available for day time view, day period view, half day period view and month view and finally for day view, there are also 2 mixed multi day / multi resource views.

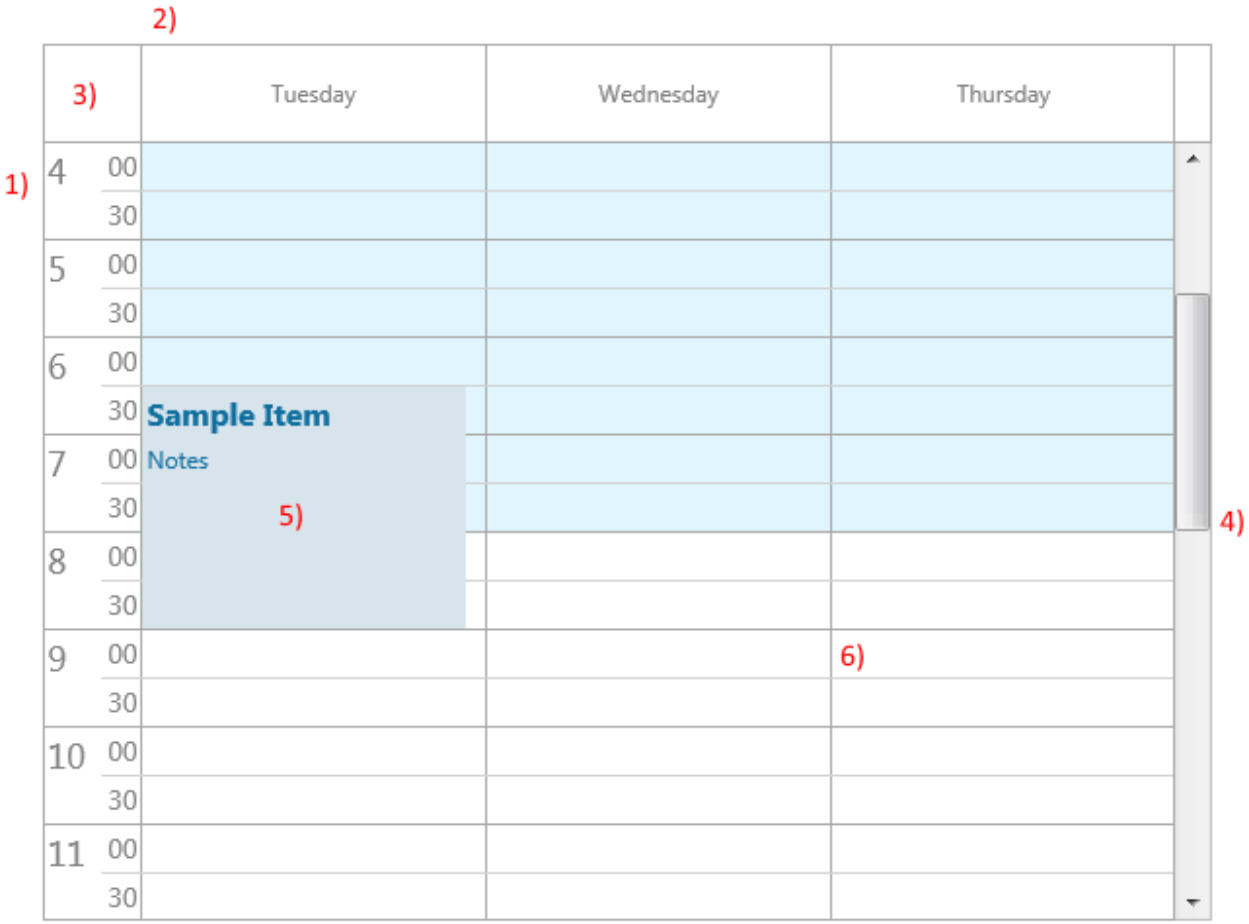
The TMS Planner for FireMonkey is designed for use with Win32, Win64, Mac OS-X, iOS and Android operating systems.



**IMPORTANT NOTICE:**
If the FireMonkey framework is new to you, please see the chapter “General FireMonkey component usage guidelines” that offers an introduction that is recommended to read before you start working with the TMS Planner for FireMonkey. Another interesting source of information is
http://docwiki.embarcadero.com/RADStudio/en/FireMonkey_Application_Platform

## Organization

Below is a quick overview of the most important elements in the planner. This guide will cover all elements in different chapters.




1) The timeline area, which displays a datetime range, set by ModeSettings.StartTime, TimeLine.DisplayStart and TimeLine.DisplayEnd. The timeline area can be set at the left and/or right side or at the top and/or bottom side depending on the orientation. The orientation can be changed with the OrientationMode property.

2) The Positions / Groups area, which displays the positions, set by Positions.Count. Depending on the mode, explained on the previous page, the positions can display datetime values and /or resources. Like the timeline, the positions / groups area can be displayed at all sides depending on the orientation mode.

3) Empty area, used for custom drawing / text.

4) Scrollbars, used to navigate through the planner. The positions /groups are stretched by default but can be configured to have a horizontal scrollbar as well. The scrollbars can be hidden to allow touch-only scrolling on mobile devices.

5) An item, that can be moved, resized and edited depending on the planner settings. Each item can have its own colors for various states and can stretch over multiple positions depending on the mode as explained on the previous page.

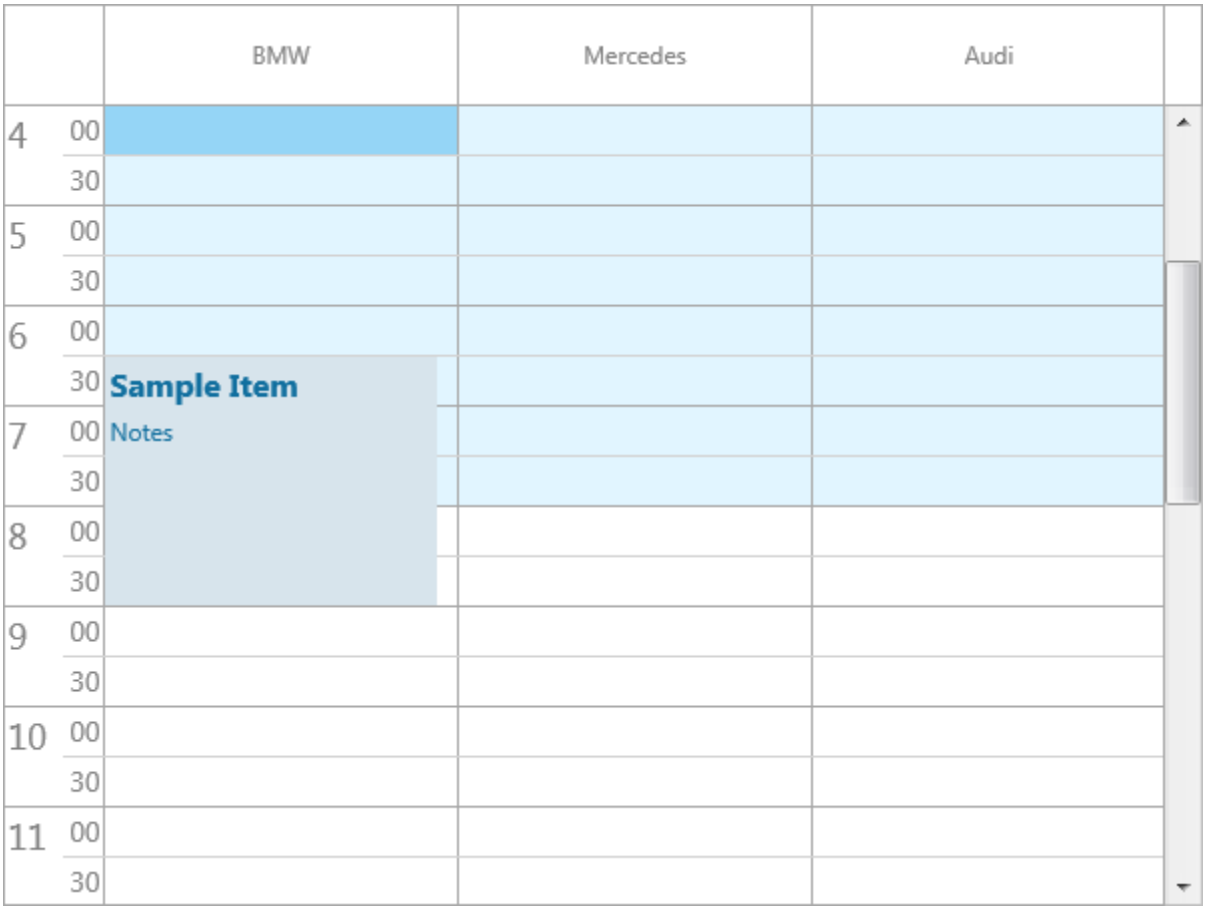
6) The grid / time slots area which is configured with the same settings as the timeline area. It displays the active and inactive datetime values and can be used to select a range of cells or navigate through the planner. The grid / time slots area display the current selection as well.

## Modes

The planner supports a set of predefined modes. In this chapter, we will illustrate and show how you can configure each mode.

**pmDay**:

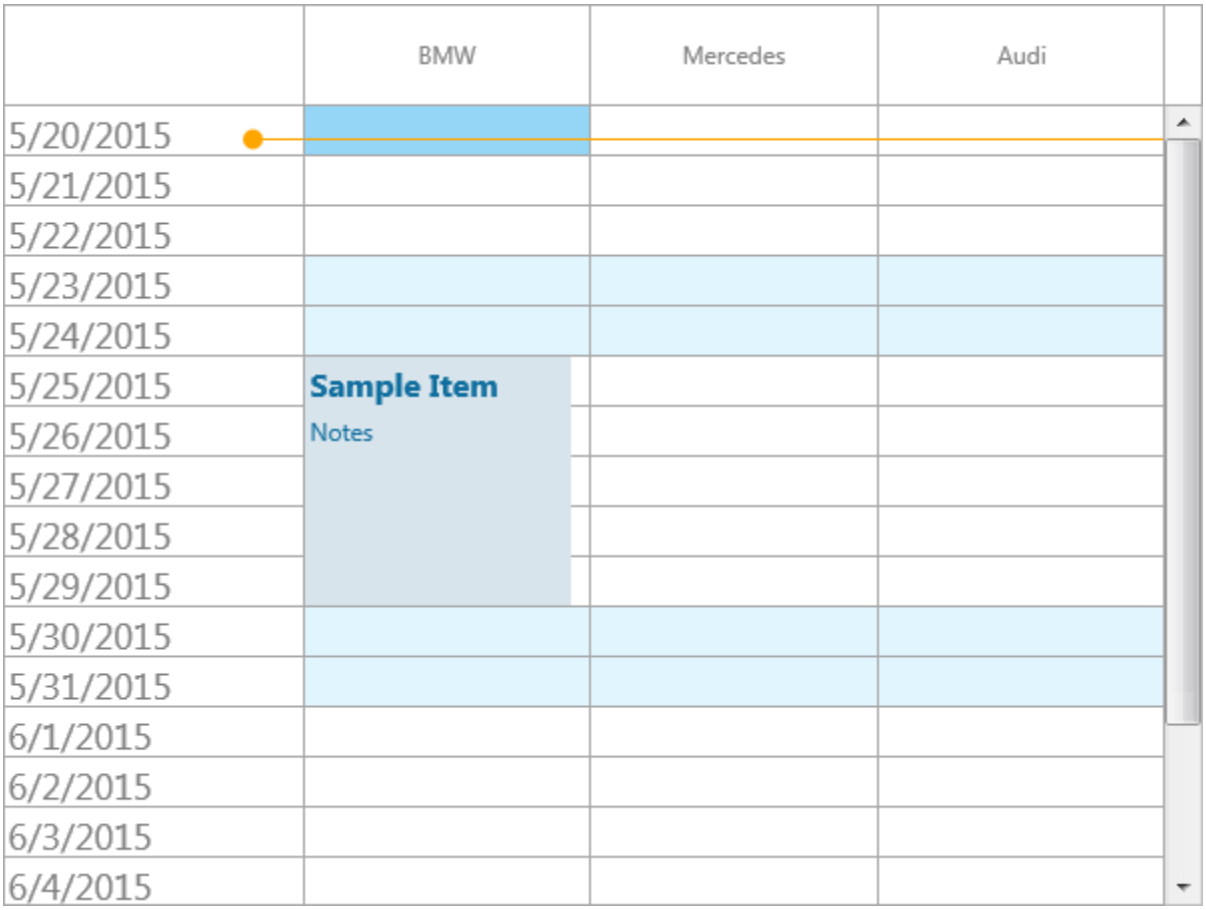
- Timeline: Displays the hours of a single day, customized with ModeSettings.StartTime, TimeLine.DisplayStart and TimeLine.DisplayEnd. Further customization can be done with the additional properties under the TimeLine property.
- Positions: Displays resources, added through the resources collection and based on the Positions.Count property. When no resources are added, the Positions are automatically given a predefined value.




**pmDayPeriod**:

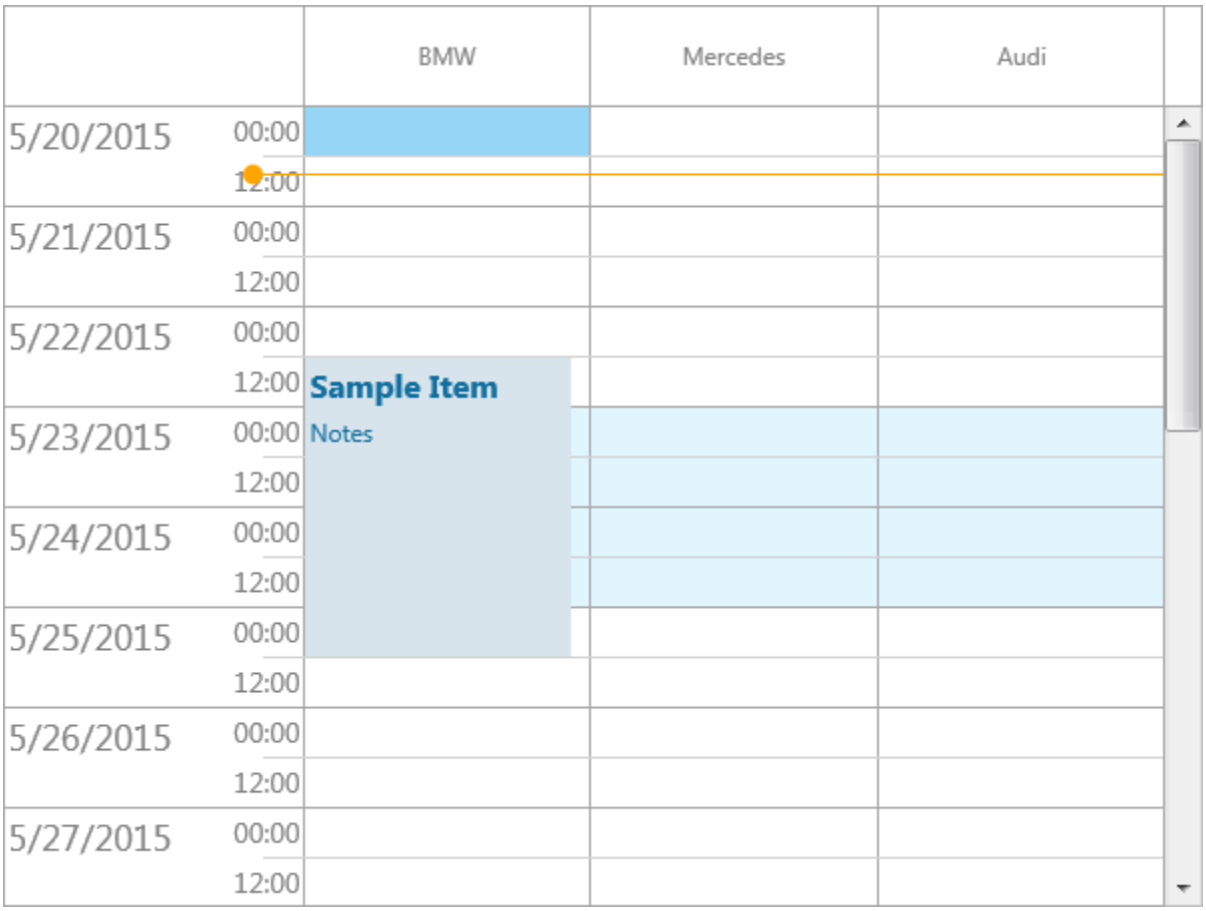
- Timeline: Displays multiple days, customized with ModeSettings.StartTime and ModeSettings.EndTime.

- Positions: Displays resources, added through the resources collection and based on the Positions.Count property. When no resources are added, the Positions are automatically given a predefined value.



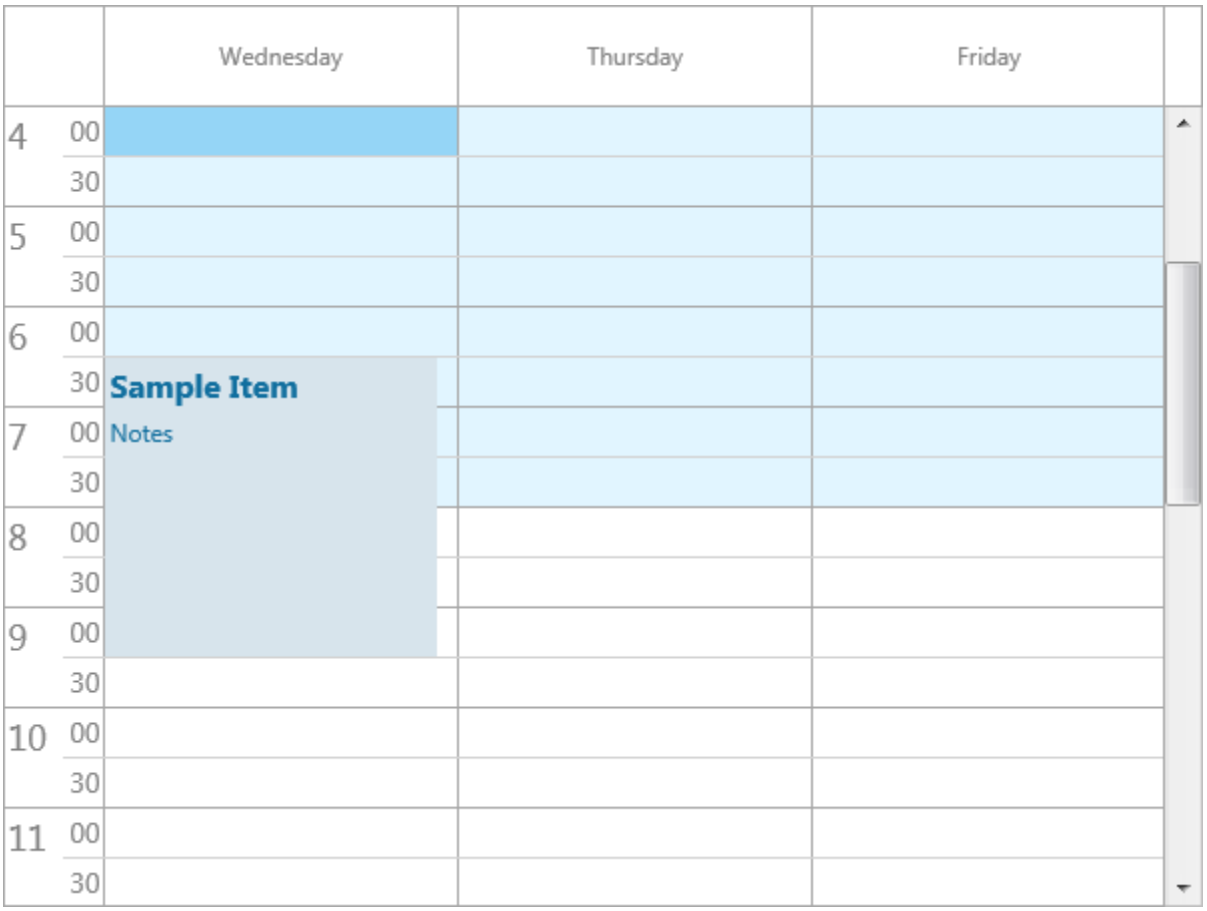

**pmHalfDayPeriod**:

- Timeline: Displays multiple half days, customized with ModeSettings.StartTime and ModeSettings.EndTime.
- Positions: Displays resources, added through the resources collection and based on the Positions.Count property. When no resources are added, the Positions are automatically given a predefined value.



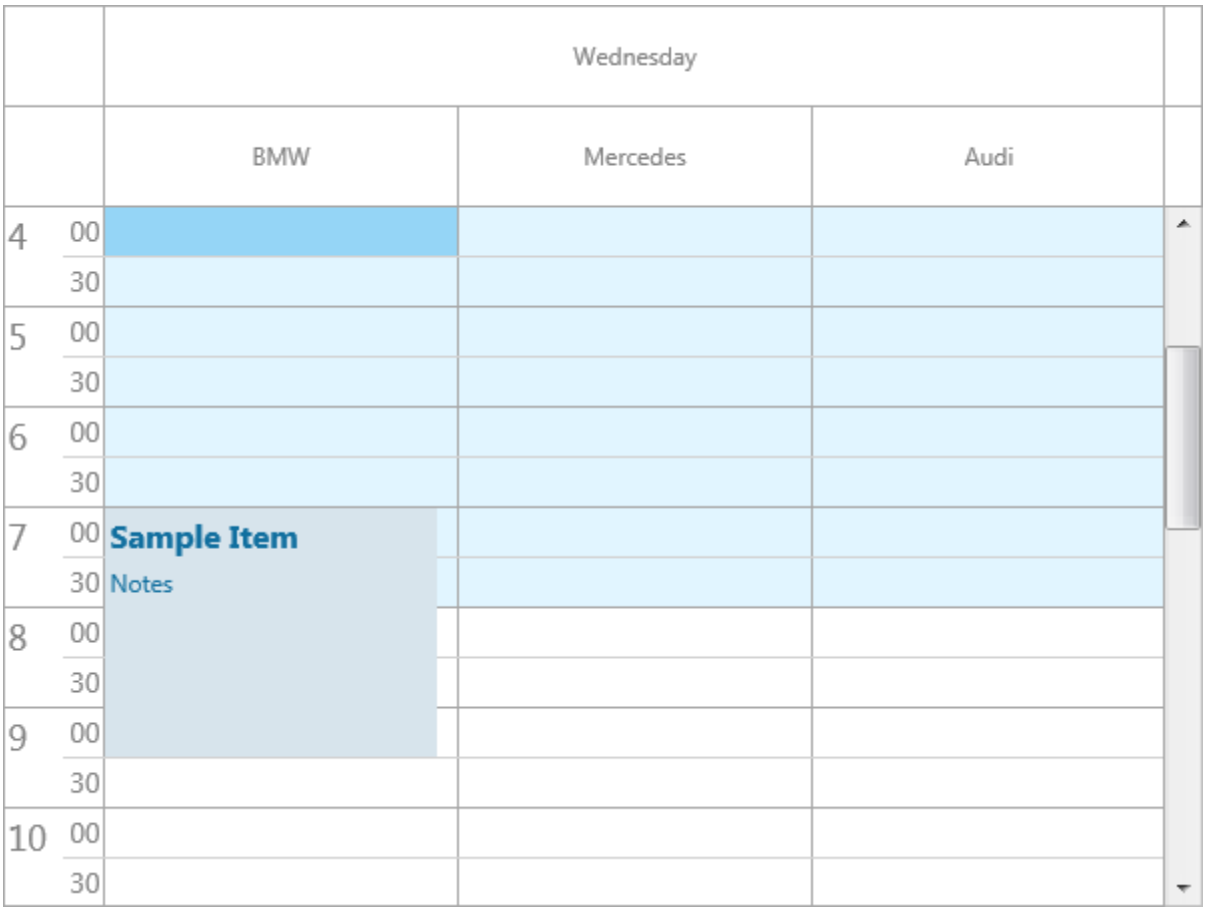

**pmMultiDay**:

- Timeline: Displays the hours of multiple days, customized with ModeSettings.StartTime, TimeLine.DisplayStart and TimeLine.DisplayEnd. Further customization can be done with the additional properties under the TimeLine property.
- Positions: Displays multiple days based on the Positions.Count property.



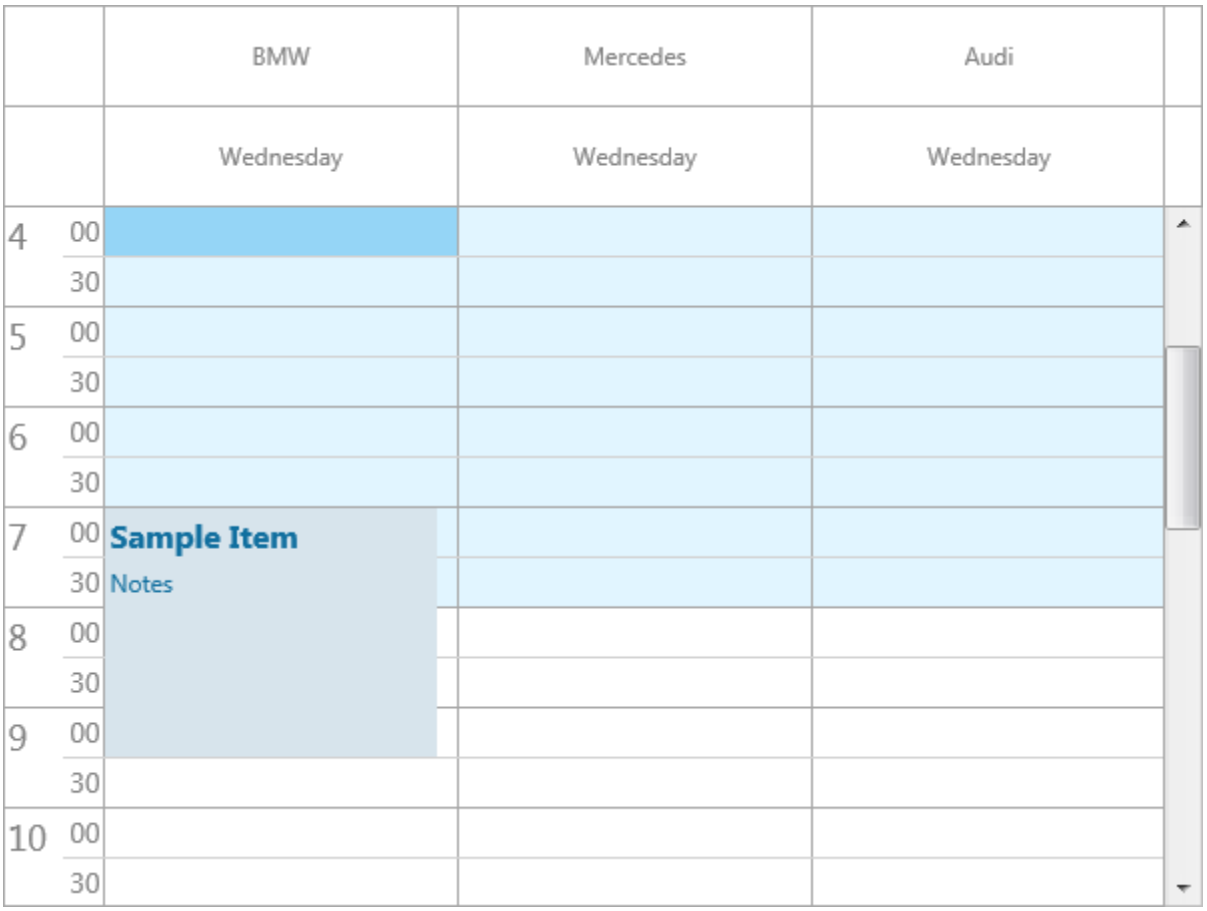

**pmMultiResDay**:

- Timeline: Displays the hours of multiple days, customized with ModeSettings.StartTime, TimeLine.DisplayStart and TimeLine.DisplayEnd. Further customization can be done with the additional properties under the TimeLine property.
- Positions: Displays multiple resources for each day based on the Positions.Count property.



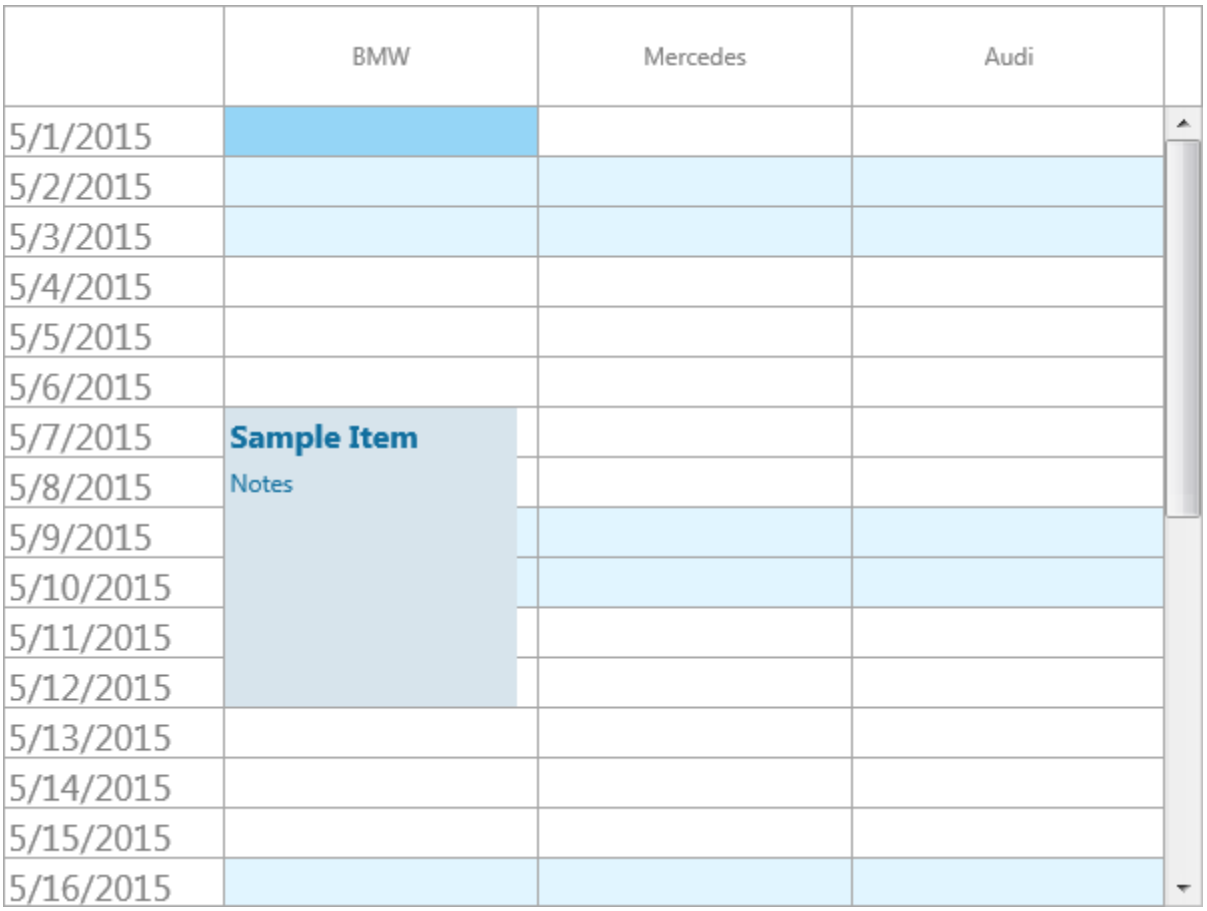

**pmMultiDayRes**:

- Timeline: Displays the hours of multiple days, customized with ModeSettings.StartTime, TimeLine.DisplayStart and TimeLine.DisplayEnd. Further customization can be done with the additional properties under the TimeLine property.
- Positions: Displays multiple days for each resource based on the Positions.Count property.



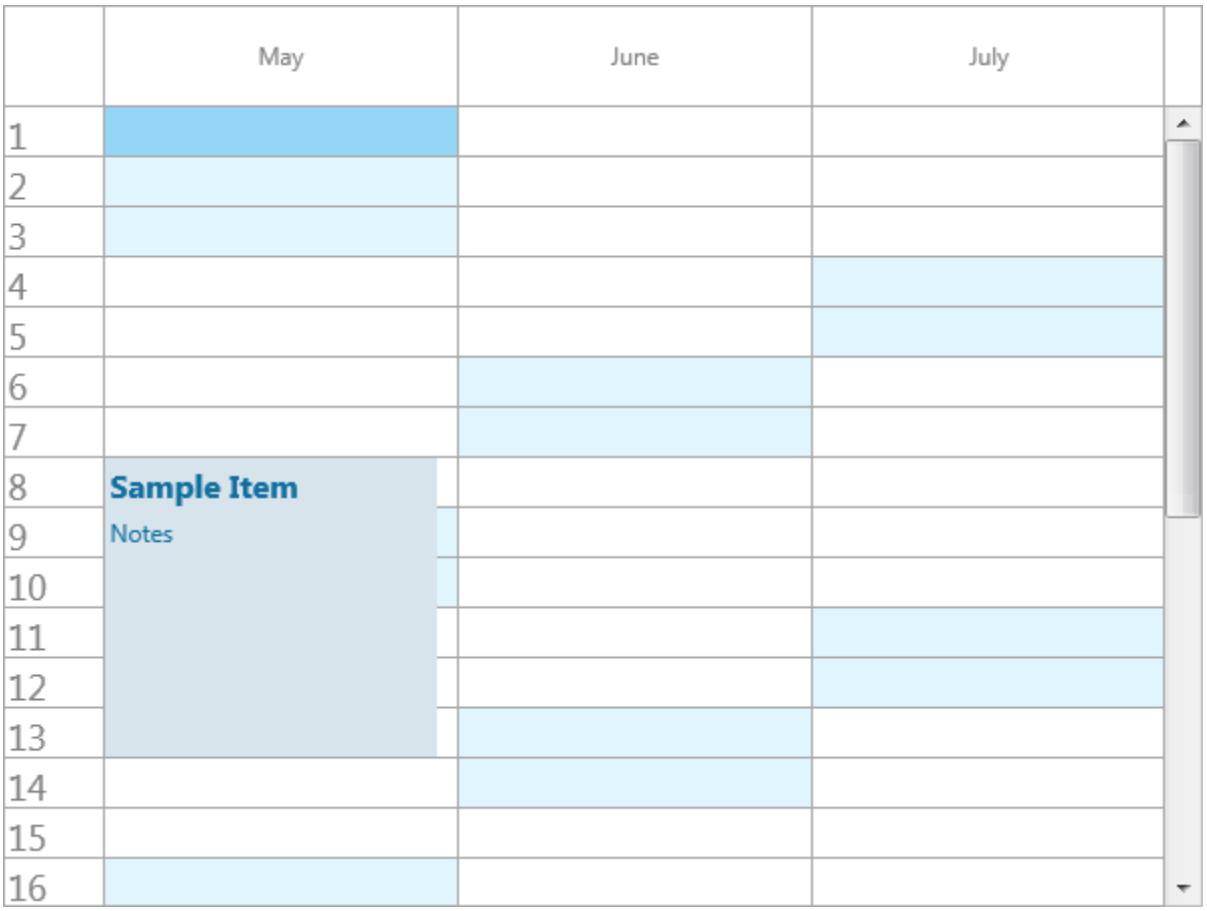

**pmMonth**:

- Timeline: Displays the days of a single month, customized with ModeSettings.StartTime, TimeLine.DisplayStart and TimeLine.DisplayEnd. Further customization can be done with the additional properties under the TimeLine property.
- Positions: Displays resources, added through the resources collection and based on the Positions.Count property. When no resources are added, the Positions are automatically given a predefined value.




**pmMultiMonth**:

- Timeline: Displays the days of multiple months, customized with ModeSettings.StartTime, TimeLine.DisplayStart and TimeLine.DisplayEnd. Further customization can be done with the additional properties under the TimeLine property.
- Positions: Displays multiple months based on the Positions.Count property.




**pmCustom**:

- Timeline: A custom set of automatically sorted datetime values added through the TMSFMXPlanner.CustomDatesTime property.
- Positions: Displays resources, added through the resources collection and based on the Positions.Count property. When no resources are added, the Positions are automatically given a predefined value.

| | BMW | Mercedes | Audi |
|---|---|---|---|
| 5/30/2015 12:00 AM | | | |
| 6/4/2015 12:00 AM | | | |
| 6/9/2015 12:00 AM | | | |
| 6/14/2015 12:00 AM | | | |
| 6/19/2015 12:00 AM | | | |
| 6/24/2015 12:00 AM | | | |
| 6/29/2015 12:00 AM | **Sample Item** | | |
| 7/4/2015 12:00 AM | Notes | | |
| 7/9/2015 12:00 AM | | | |
| 7/14/2015 12:00 AM | | | |
| 7/19/2015 12:00 AM | | | |
| 7/24/2015 12:00 AM | | | |
| 7/29/2015 12:00 AM | | | |
| 8/3/2015 12:00 AM | | | |
| 8/8/2015 12:00 AM | | | |
| 8/13/2015 12:00 AM | | | |

## TimeLine

### Display configuration

The timeline displays a range of timeslots configured with the properties under TimeLine. The ModeSettings.StartTime is used to set the planner's initial display start time and with the TimeLine.DisplayUnitFormat / TimeLine.DisplaySubUnitFormat the values that are displayed are formatted. The amount of units can be changed with the TimeLine.DisplayUnit property in combination with the TimeLine.DisplayUnitType property.

For the pmMultiDay, pmDay, pmMultiResDay and pmMultiDayRes modes a view of 24 hours is displayed with subunits every 30 minutes (TimeLine.DisplayUnit := 30 and TimeLine.DisplayUnitType := pduMinute). For the pmMonth and pmMultiMonth modes a view per day is shown (TimeLine.DisplayUnit := 1 and TimeLine.DisplayUnitType := pduDay).

Instead of the TimeLine.DisplayStart and TimeLine.DisplayEnd, the ModeSettings.EndTime is used in the pmHalfDayPeriod and pmDayPeriod modes, and these modes display a half day or a full day respectively. The TimeLine.DisplayUnit, TimeLine.DisplayUnitOffset, TimeLine.DisplayUnitType and TimeLine.DisplayUnitOffsetType do not have any effect on these modes.

The displayed time range can be changed with the TimeLine.DisplayStart and TimeLine.DisplayEnd properties. Below are some samples that demonstrate how these properties are used.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.TimeLine.DisplayUnit := 10;
TMSFMXPlanner1.TimeLine.DisplayUnitType := pduMinute;
TMSFMXPlanner1.TimeLine.DisplayStart := 0;
TMSFMXPlanner1.TimeLine.DisplayEnd := 143;
TMSFMXPlanner1.EndUpdate;
```

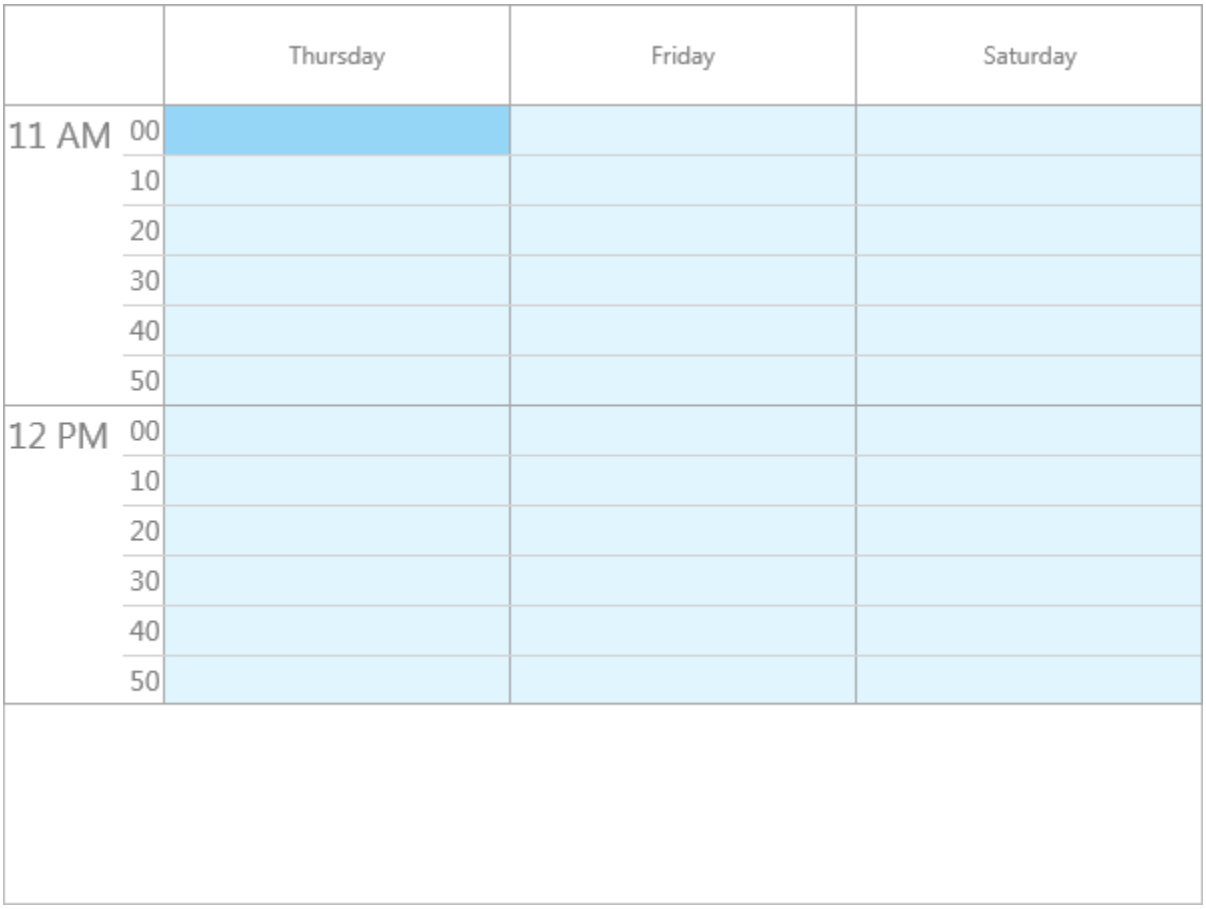
The above code changes the range to display a timeslot every 10 minutes for a full 24 hour range for a single day. The TimeLine.DisplayStart property remains 0, the TimeLine.DisplayEnd value is set to 143 which is based on the following calculation:

Round(MinsPerDay / TMSFMXPlanner1.TimeLine.DisplayUnit) - 1;




The TimeLine.DisplayStart is 0 which displays the initial ModeSettings.StartTime at midnight til midnight of the next day (24 hour range). Below is a sample that changes this to start at 11 PM til 13 AM. (2 hour range). The below code applies this to a pmMultiDay mode and shows how to calculate the TimeLine.DisplayStart and TimeLine.DisplayEnd. Additionally it applies formatting to the units and increases the size of the timeline.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmMultiDay;
TMSFMXPlanner1.TimeLine.DisplayUnit := 10;
TMSFMXPlanner1.TimeLine.DisplayUnitType := pduMinute;
TMSFMXPlanner1.TimeLine.DisplayStart := Round((MinsPerHour * 11) /
TMSFMXPlanner1.TimeLine.DisplayUnit);
TMSFMXPlanner1.TimeLine.DisplayEnd := Round((MinsPerHour * 13) /
TMSFMXPlanner1.TimeLine.DisplayUnit) - 1;
TMSFMXPlanner1.TimeLine.DisplayUnitFormat := 'h AMPM';
TMSFMXPlanner1.TimeLineAppearance.LeftSize := 80;
TMSFMXPlanner1.EndUpdate;
```




In the pmDay mode the TimeLine.DisplayEnd property doesn't have a single day limitation, since the days are continuously displayed along the timeline. After the 24 hour mark of the initial ModeSettings.StartTime, the timeline continues to display the next day. In the pmMultiDay, pmMultiResDay and pmMultiDayRes modes however, the range is limited to display maximum 24 hours. The initial ModeSettings.StartTime is displayed in the first position, the next day in the next position, etc…

The range can be displayed with an offset. The properties TimeLine.DisplayUnitOffset and TimeLine.DisplayUnitOffsetType are used for this purpose. Below is a sample that applies an additional offset of 5 minutes to the pmDay sample code.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.TimeLine.DisplayUnit := 10;
TMSFMXPlanner1.TimeLine.DisplayUnitType := pduMinute;
TMSFMXPlanner1.TimeLine.DisplayStart := 0;
```

```
TMSFMXPlanner1.TimeLine.DisplayEnd := 143;
TMSFMXPlanner1.TimeLine.DisplayOffset := 5;
TMSFMXPlanner1.TimeLine.DisplayOffsetType := pduMinute;
TMSFMXPlanner1.EndUpdate;
```



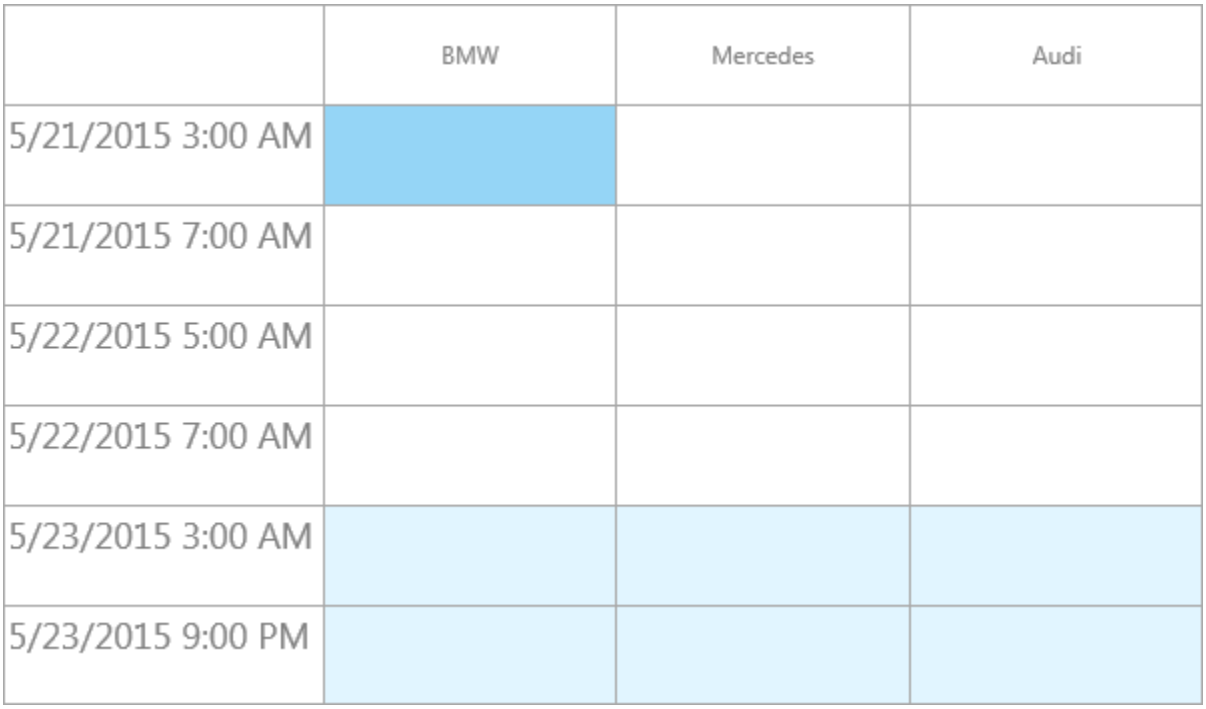

The pmMonth and pmMultiMonth modes are similar to the pmDay and pmMultiDay modes except the range shows all the days for a single month in pmMonth mode and a range from 1 to 31 for the pmMultiMonth mode. The first month in pmMultiMonth mode is displayed in the first position, the next month in the next position.

The difference between pmMultiMonth and pmMultiDay mode is that the TimeLine.DisplayUnit, TimeLine.DisplayUnitOffset, TimeLine.DisplayUnitType and TimeLine.DisplayUnitOffsetType do not have any effect.

The pmCustom mode is based on a public generic TList of TDateTime values (property CustomDateTimes). The timeline configuration is limited to the TimeLine.DisplayUnitFormat property. Below is a sample that demonstrates how to configure a custom timeline.

Additionally it changes the unit size with the TimeLine.DisplayUnitSize property. This is used to change the height / width of a time slot depending on the orientation.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmCustom;
dt := Int(Now);
TMSFMXPlanner1.CustomDateTimes.Add(dt + EncodeTime(3, 0, 0, 0));
TMSFMXPlanner1.CustomDateTimes.Add(dt + EncodeTime(7, 0, 0, 0));
TMSFMXPlanner1.CustomDateTimes.Add(dt + 1 + EncodeTime(5, 0, 0, 0));
TMSFMXPlanner1.CustomDateTimes.Add(dt + 1 + EncodeTime(7, 0, 0, 0));
TMSFMXPlanner1.CustomDateTimes.Add(dt + 2 + EncodeTime(3, 0, 0, 0));
TMSFMXPlanner1.CustomDateTimes.Add(dt + 2 + EncodeTime(21, 0, 0, 0));
TMSFMXPlanner1.TimeLineAppearance.LeftSize := 160;
TMSFMXPlanner1.TimeLine.DisplayUnitSize := 75;
TMSFMXPlanner1.EndUpdate;
```




## Appearance

The look and feel of the timeline can be changed with the TimeLineAppearance properties. These properties can be used for the timeline that is placed left and/or right or top and or bottom in horizontal mode. Below is a sample that configures the timeline to change the font,

font color and fill of a timeslot as well as showing the timeline at the left and right of the planner.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.TimeLineAppearance.Layouts := [ptlLeft, ptlRight];
TMSFMXPlanner1.TimeLineAppearance.LeftFontColor := claSteelblue;
TMSFMXPlanner1.TimeLineAppearance.LeftFill.Color := claAliceblue;
TMSFMXPlanner1.TimeLineAppearance.LeftFill.Kind := TBrushKind.Solid;
TMSFMXPlanner1.TimeLineAppearance.RightFontColor := claOrangered;
TMSFMXPlanner1.TimeLineAppearance.RightFill.Color := claGreenyellow;
TMSFMXPlanner1.TimeLineAppearance.RightFill.Kind := TBrushKind.Solid;
TMSFMXPlanner1.TimeLineAppearance.RightFont.Family := 'Broadway';
TMSFMXPlanner1.TimeLineAppearance.RightSubUnitFontSize := 10;
TMSFMXPlanner1.EndUpdate;
```



## Positions / Resources

### Display configuration

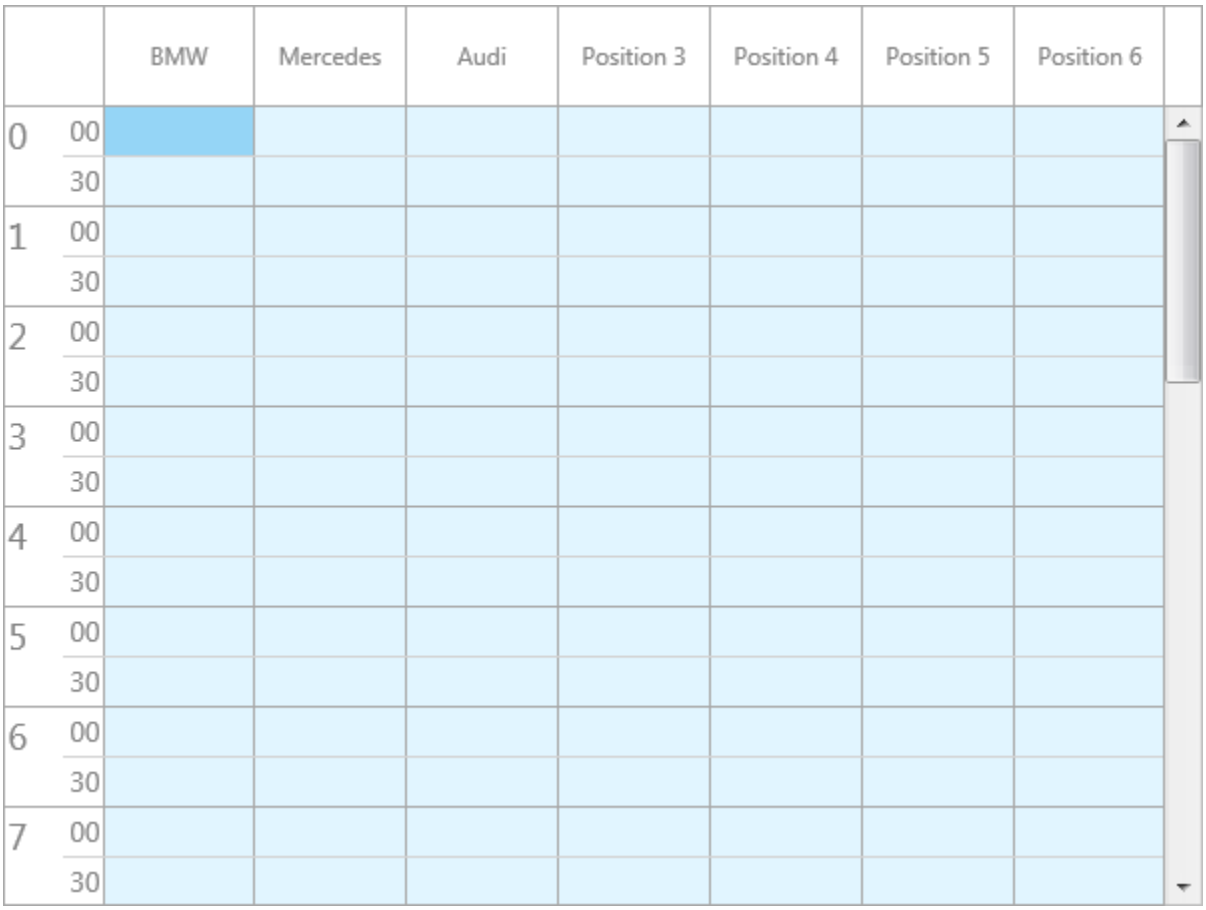
The positions area is designed for multiple purposes. In pmDay, pmHalfDayPeriod, pmDayPeriod, pmMonth and pmCustom the positions area displays the resources, which are added through the Resources collection. When no resources exist, the planner automatically uses a default resource. In these modes, the position to resource and resource to position conversion is one on one.

In pmMultiDay and pmMultiMonth modes, the Resources are not used, instead the configuration of the timeline is no longer limited to the timeline area, but also stretches along the positions area. This view is capable of displaying multiple days / months in multiple positions, where the previous modes where only capable of display a single day / month or a day / month that continuously runs along the timeline.

The special modes that combine resources and multiple days are the pmMultiResDay and pmMultiDayRes modes. Additionally, these modes also make use of the Groups that are explained in a separate chapter.

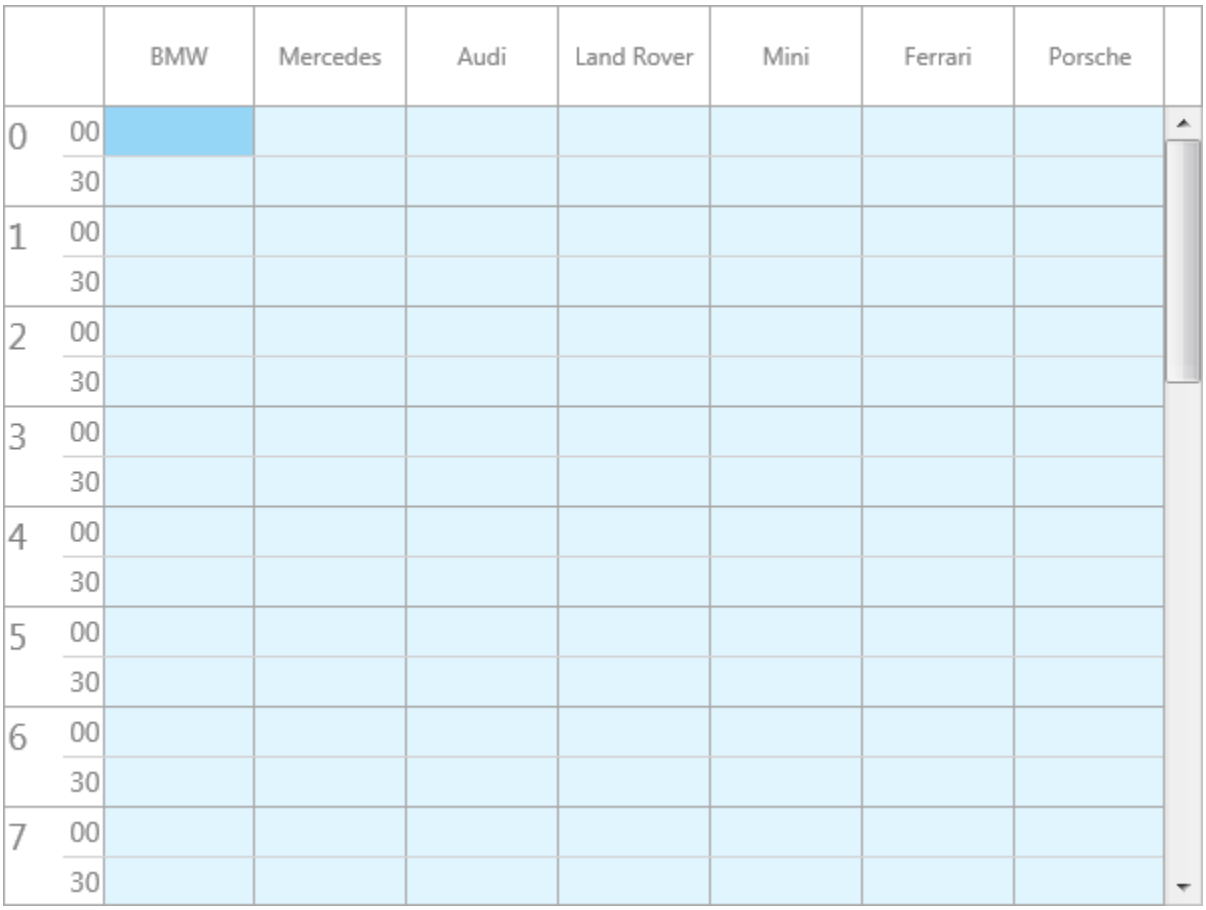
The positions that are drawn are set with Positions.Count as demonstrated in the sample below.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.Positions.Count := 7;
TMSFMXPlanner1.EndUpdate;
```




The planner has 3 resources (“BMW / “Mercedes” / “Audi”) by default. As seen in the screenshot, those default resources are displayed for the first 3 positions. The positions count has been set to 7, and the planner will automatically set a default resource for the remaining positions. To add more resources, use the following code:

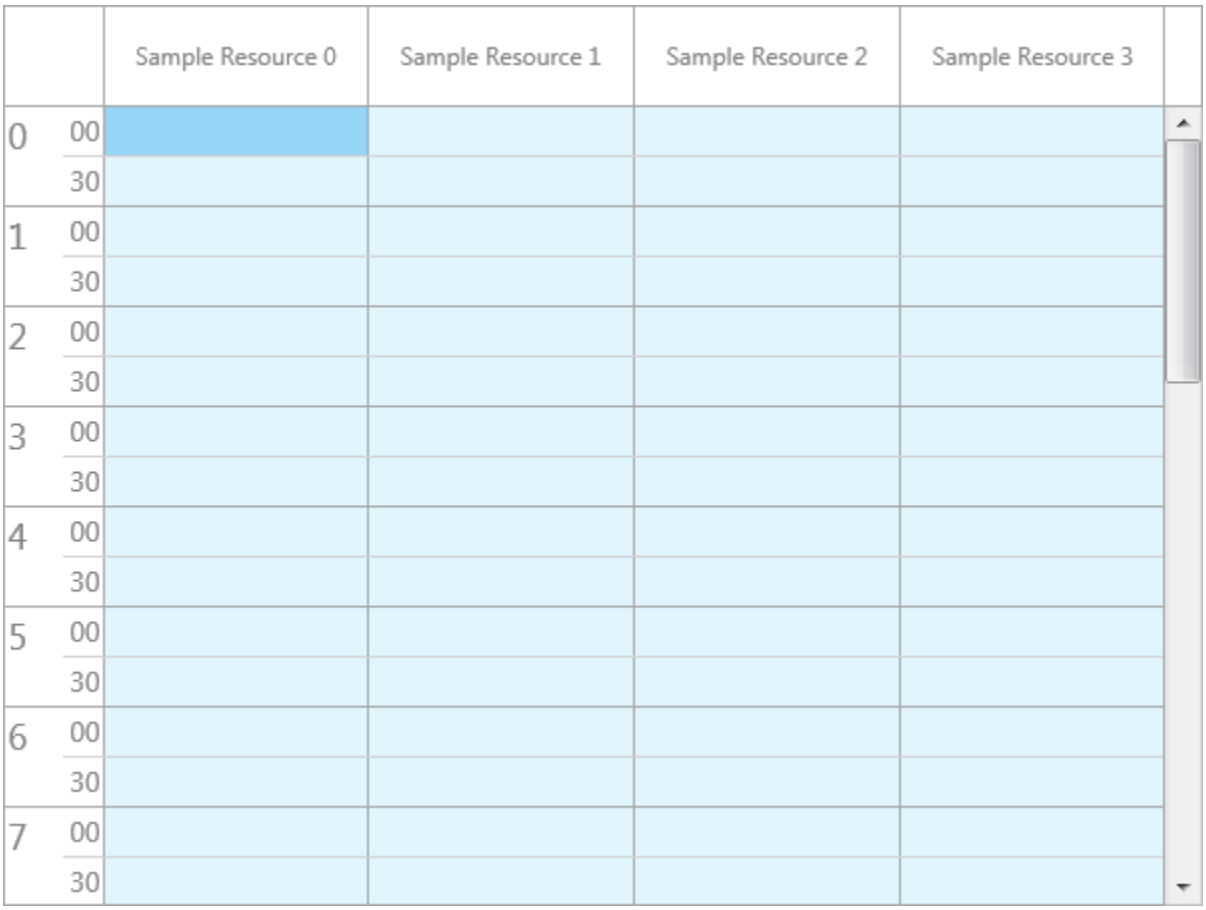
```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.Positions.Count := 7;
TMSFMXPlanner1.Resources.Add.Text := 'Land Rover';
TMSFMXPlanner1.Resources.Add.Text := 'Mini';
TMSFMXPlanner1.Resources.Add.Text := 'Ferrari';
TMSFMXPlanner1.Resources.Add.Text := 'Porsche';
TMSFMXPlanner1.EndUpdate;
```




Using the Resources collection is not obligatory. You can also dynamically set resources by implementing the OnGetPositionText event.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.Positions.Count := 4;
TMSFMXPlanner1.Resources.Clear;
TMSFMXPlanner1.EndUpdate;

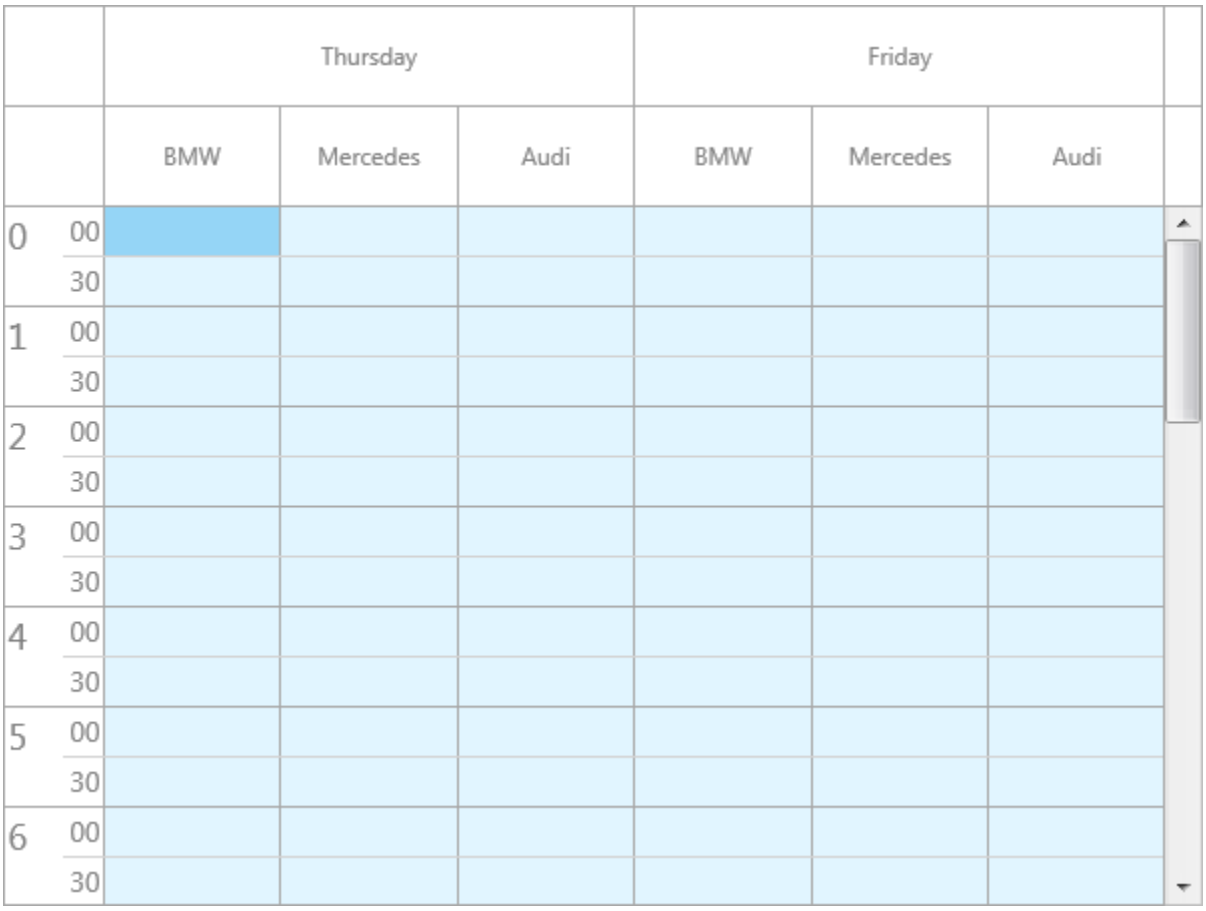
procedure TForm1.TMSFMXPlanner1GetPositionText(Sender: TObject;
  APosition: Integer; AKind: TTMSFMXPlannerCacheItemKind; var AText:
string);
begin
  AText := 'Sample Resource ' + inttostr(APosition);
end;
```




When switching to pmMultiDay or pmMultiMonth mode on a default planner you will notice that the resources will no longer be used. Instead the positions represent days / months respectively. The formatting of the days / months representation is automatically determined by the mode, but can be overriden with the Positions.Format property.

The modes pmMultiResDay and pmMultiDayRes combine both resources and days in the positions / groups area. The initial positions count is set with the property Positions.Count and the Resources collection is filled with resource items. In pmMultiResDay the resources are drawn in the positions area, and the days in the groups area and for the pmMultiDayRes vice versa. In all other modes, the groups area is used for grouping of resources through the Groups collection, which is explained in the next chapter.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmMultiResDay;
TMSFMXPlanner1.Positions.Count := 6;
TMSFMXPlanner1.EndUpdate;
```



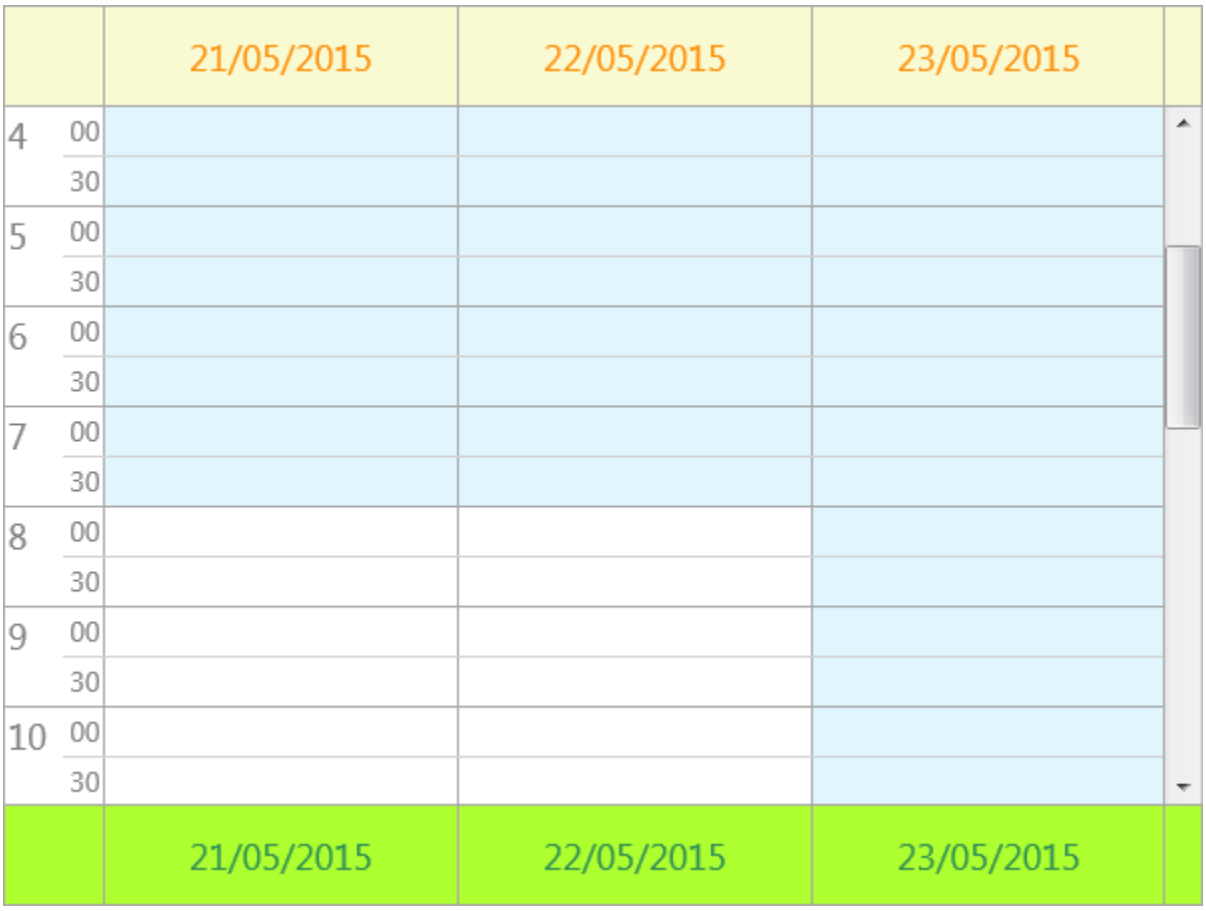

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmMultiDayRes;
TMSFMXPlanner1.Positions.Count := 6;
TMSFMXPlanner1.EndUpdate;
```




## Appearance

The appearance of the positions area is similar to the timeline area, and is found under PositionsAppearance. Below is a sample that demonstrates this property set.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmMultiDay;
TMSFMXPlanner1.Positions.Format := 'dd/mm/yyy';
TMSFMXPlanner1.PositionsAppearance.TopFontColor := claDarkorange;
TMSFMXPlanner1.PositionsAppearance.TopFont.Size := 18;
TMSFMXPlanner1.PositionsAppearance.TopFill.Color :=
claLightgoldenrodyellow;
TMSFMXPlanner1.PositionsAppearance.TopFill.Kind := TBrushKind.Solid;
TMSFMXPlanner1.PositionsAppearance.Layouts := [pplTop, pplBottom];
TMSFMXPlanner1.EndUpdate;
```



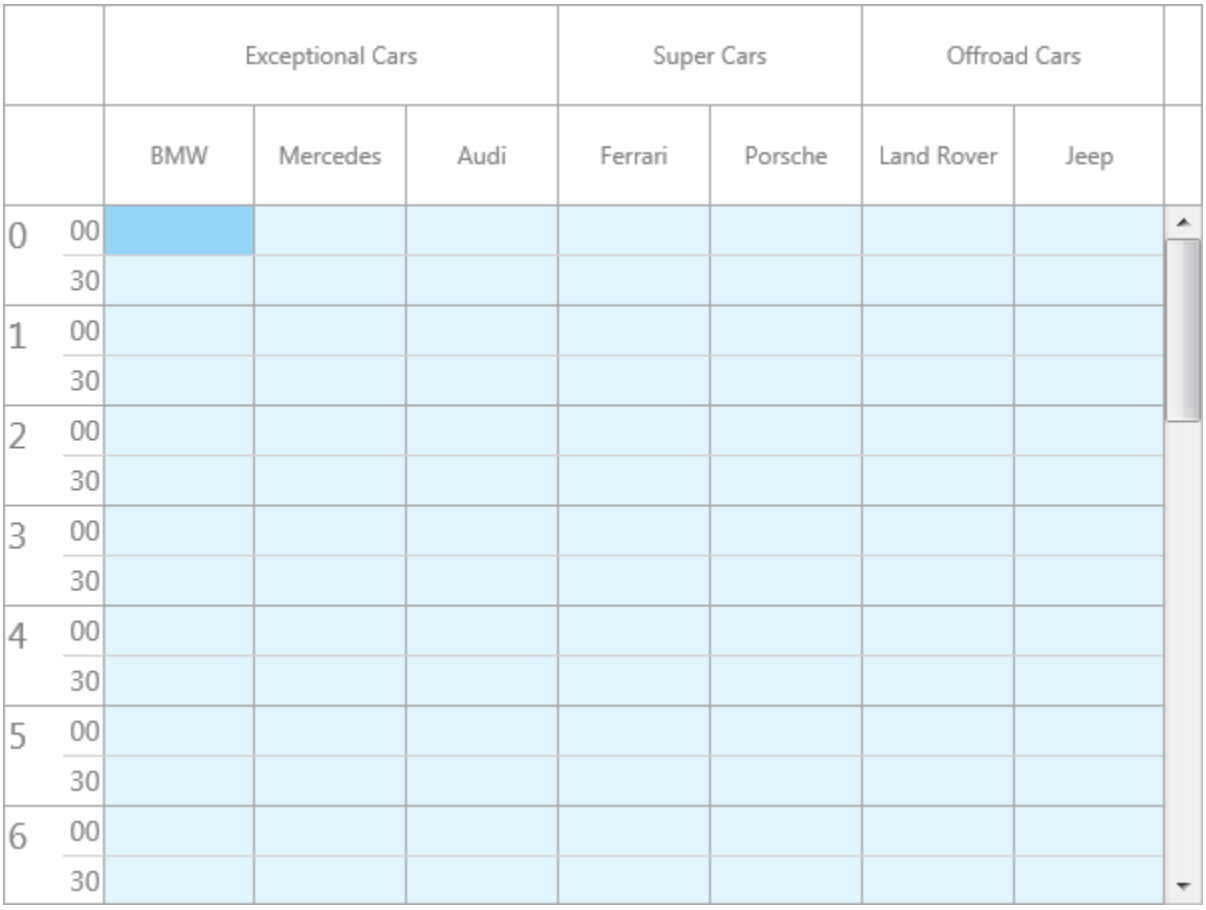

## Groups

### Display configuration

As explained in the previous chapter, groups are used in pmMultiResDay and pmMultiDayRes to indicate days or resources. In all other modes, the groups are only visible in combination with the Groups collection. A group indicates a series of resources. Groups are always placed above (top layout) or below (bottom layout) positions. Below is a sample that demonstrates this.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.Positions.Count := 7;
TMSFMXPlanner1.Resources.Add.Text := 'Ferrari';
TMSFMXPlanner1.Resources.Add.Text := 'Porsche';
TMSFMXPlanner1.Resources.Add.Text := 'Land Rover';
TMSFMXPlanner1.Resources.Add.Text := 'Jeep';
```

```
grp := TMSFMXPlanner1.Groups.Add;
grp.Text := 'Exceptional Cars';
grp.StartPosition := 0;
grp.EndPosition := 2;
grp := TMSFMXPlanner1.Groups.Add;
grp.Text := 'Super Cars';
grp.StartPosition := 3;
grp.EndPosition := 4;
grp := TMSFMXPlanner1.Groups.Add;
grp.Text := 'Offroad Cars';
grp.StartPosition := 5;
grp.EndPosition := 6;
TMSFMXPlanner1.EndUpdate;
```

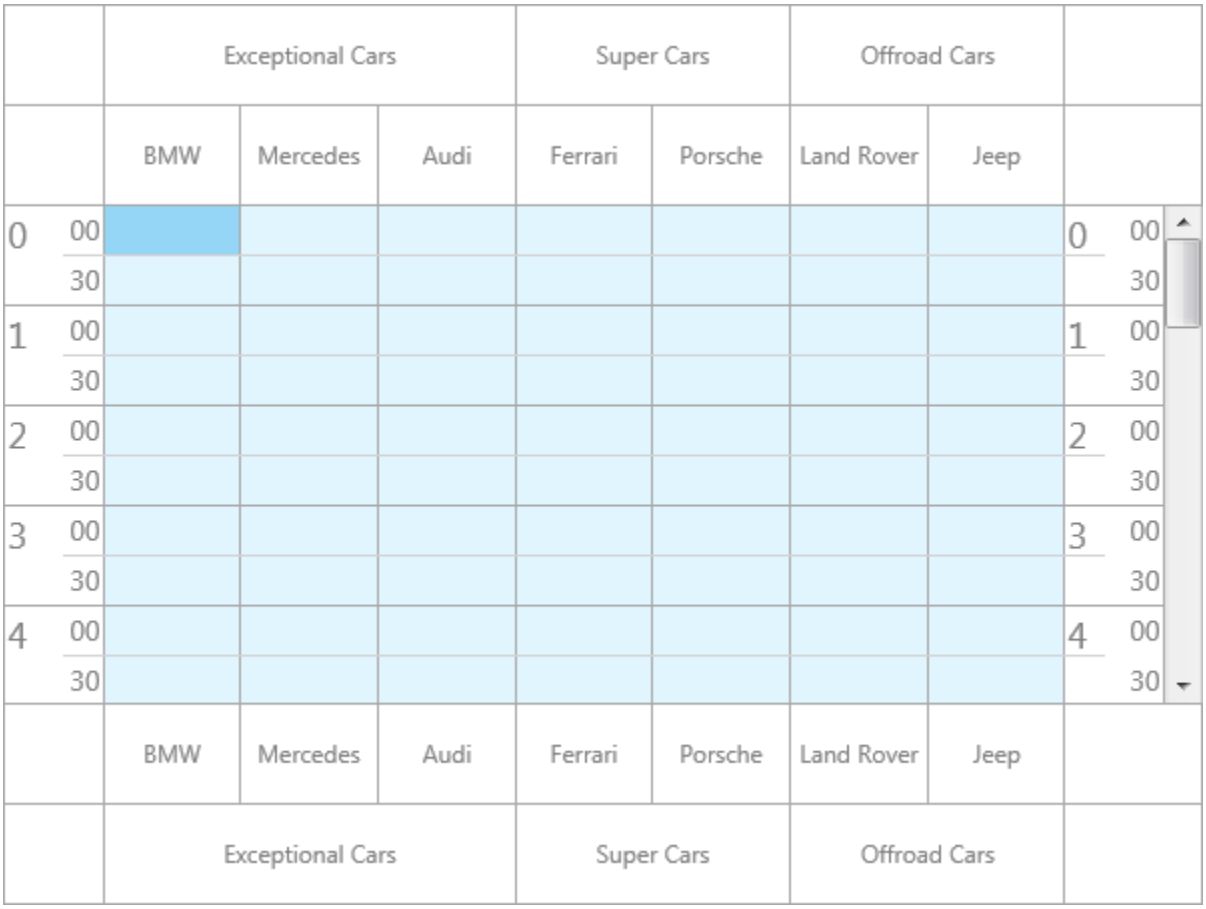



## Appearance

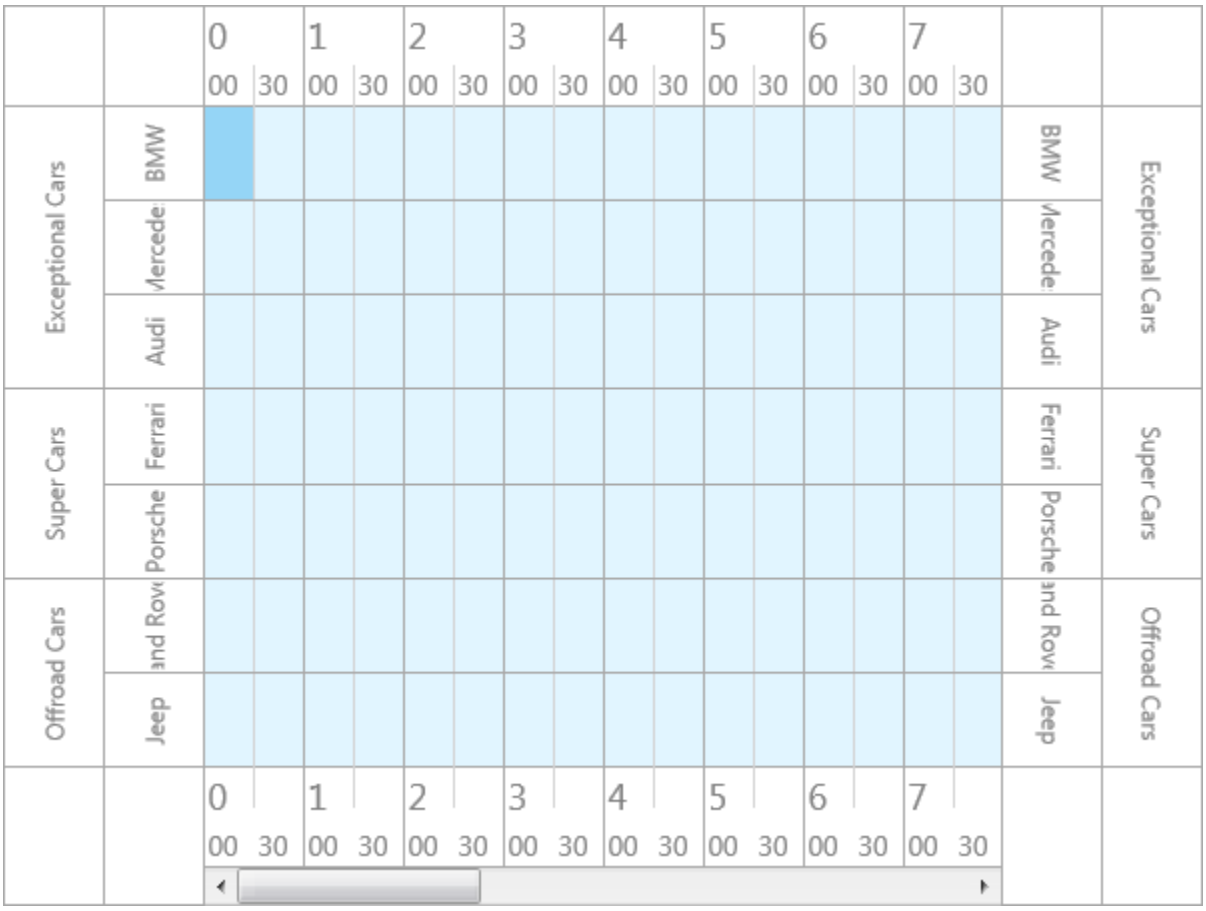
Similar to the positions appearance, the appearance of the groups can be found under GroupsAppearance. The groups can be placed at the top and / or bottom side in vertical mode and the left and / or right side in horizontal mode. Below is a screenshot that shows the groups / positions and timeline in full layout mode in both directions.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.Positions.Count := 7;
TMSFMXPlanner1.Resources.Add.Text := 'Ferrari';
TMSFMXPlanner1.Resources.Add.Text := 'Porsche';
TMSFMXPlanner1.Resources.Add.Text := 'Land Rover';
TMSFMXPlanner1.Resources.Add.Text := 'Jeep';
grp := TMSFMXPlanner1.Groups.Add;
grp.Text := 'Exceptional Cars';
grp.StartPosition := 0;
grp.EndPosition := 2;
grp := TMSFMXPlanner1.Groups.Add;
grp.Text := 'Super Cars';
grp.StartPosition := 3;
grp.EndPosition := 4;
grp := TMSFMXPlanner1.Groups.Add;
grp.Text := 'Offroad Cars';
grp.StartPosition := 5;
grp.EndPosition := 6;
TMSFMXPlanner1.PositionsAppearance.Layouts := [pplTop, pplBottom];
TMSFMXPlanner1.TimeLineAppearance.Layouts := [ptlLeft, ptlRight];
TMSFMXPlanner1.GroupsAppearance.Layouts := [pglTop, pglBottom];
TMSFMXPlanner1.EndUpdate;
```

Vertical mode (`TMSFMXPlanner1.OrientationMode := pomVertical`)




Horizontal mode (`TMSFMXPlanner1.OrientationMode := pomHorizontal`)




## Grid

### Display configuration

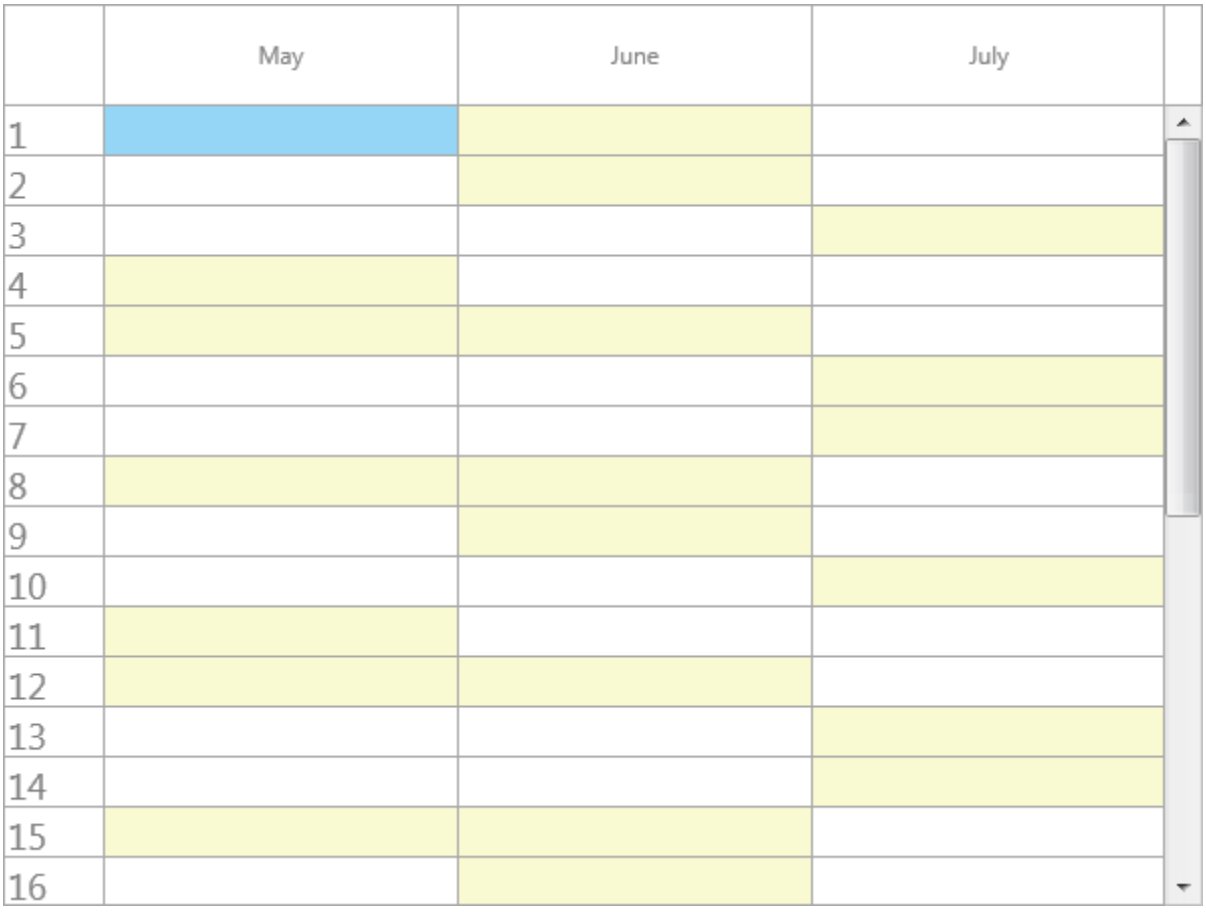
The area between the timeline and the positions area is the grid area. The grid area is scrollable (depending on the positions and timeline configuration), and shows the items (events) along with the active, inactive and disabled time slot values. The inactive time slots can be configured with the TimeLine.ActiveStart and TimeLine.ActiveEnd and the ModeSettings.InactiveDays properties.

The grid also displays the current selected timeslots in a different appearance. The Interaction chapter explains more about selection settings in the grid.

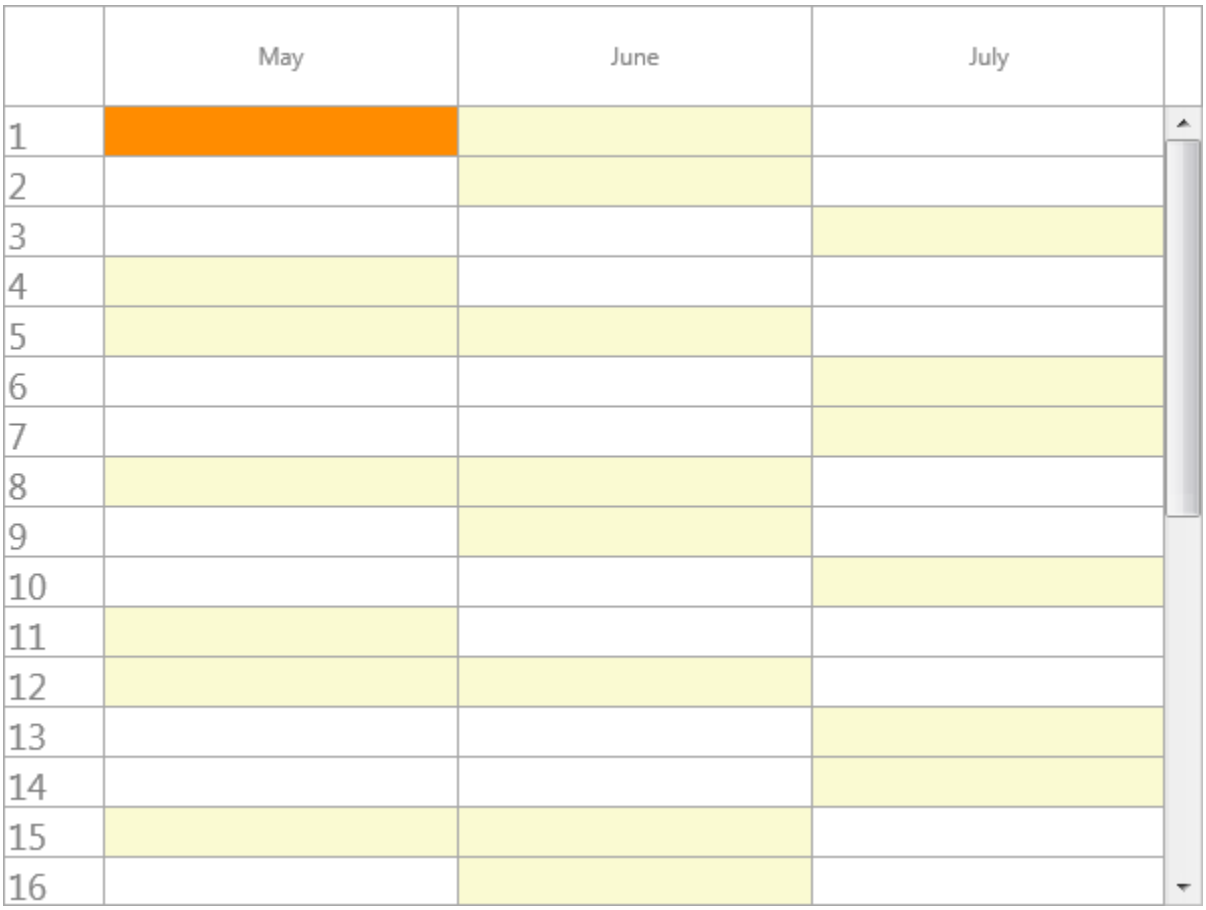
### Appearance

The grid appearance can be changed under the GridCellAppearance property. Below is a sample that changes the inactive days and changes the inactive fill for the pmMultiMonth mode.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmMultiMonth;
TMSFMXPlanner1.GridCellAppearance.InActiveFill.Color :=
claLightgoldenrodyellow;
TMSFMXPlanner1.ModeSettings.InActiveDays := [padMonday, padTuesday,
padFriday];
TMSFMXPlanner1.EndUpdate;
```




The selection is drawn with a fill that can be changed under SelectionAppearance.




## Current time indication

The timeline and the grid area are capable of displaying the current machine/device time. By default, the current time is set to show a line in both grid & timeline areas. The code below shows how to set the start time, configure the timeline to show units of 10 minutes and initialize the planner start view to the hour mark of the current time.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.ModeSettings.StartTime := Now;
TMSFMXPlanner1.TimeLine.DisplayUnit := 10;
TMSFMXPlanner1.TimeLine.CurrentTimeMode := pctmLine;
TMSFMXPlanner1.TimeLine.DisplayEnd := Round(MinsPerDay /
TMSFMXPlanner1.TimeLine.DisplayUnit) - 1;
TMSFMXPlanner1.EndUpdate;
TMSFMXPlanner1.TimeLine.ViewStart :=
Int(TMSFMXPlanner1.ModeSettings.StartTime) +
EncodeTime(HourOf(TMSFMXPlanner1.ModeSettings.StartTime), 0, 0, 0);
```



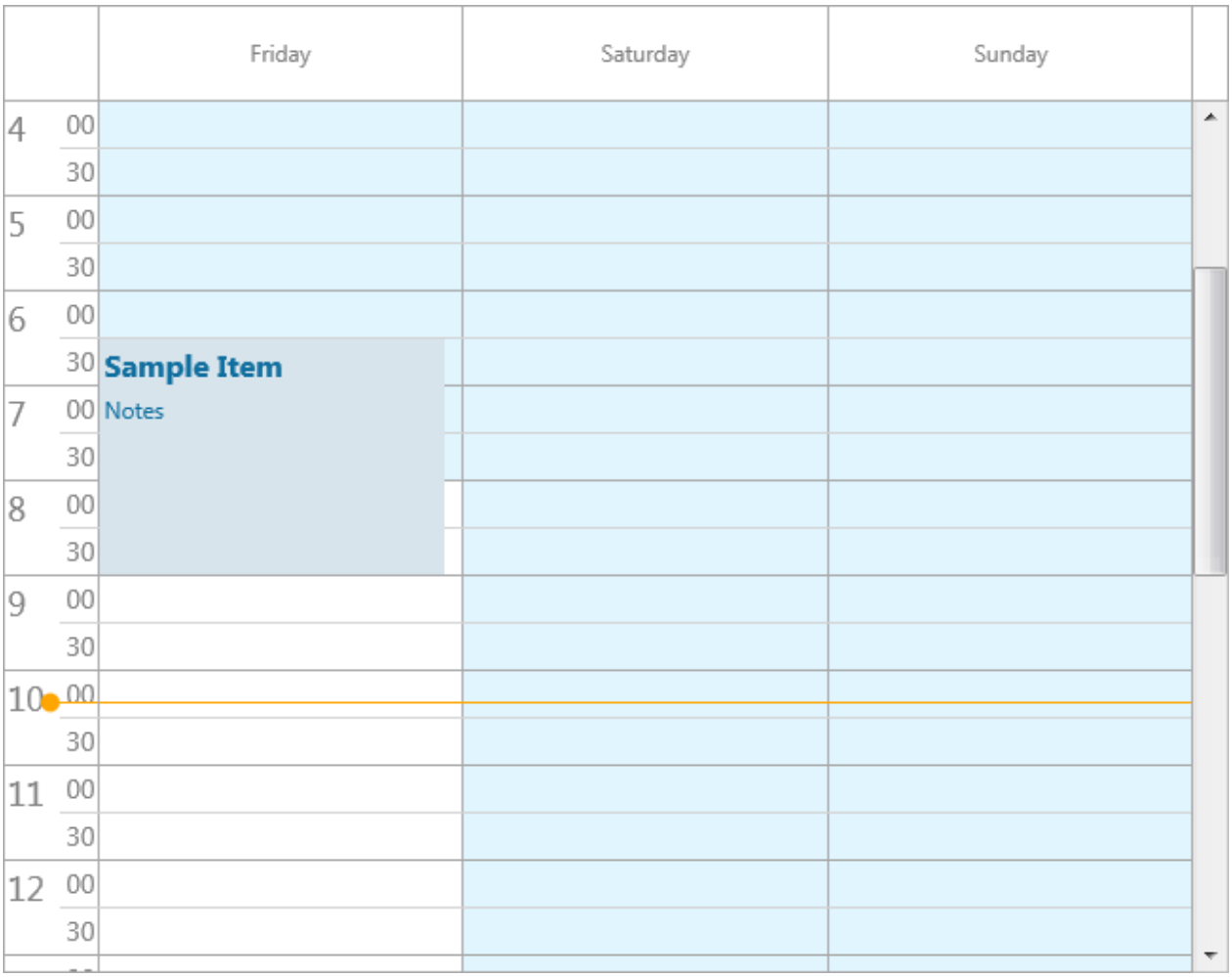

Setting the TimeLine.CurrentTimeMode to pctmText will display text in the timeline area instead. The current time indication can have a different color under TimeLineAppearance.CurrentTimeColor. Further customization can be done with one of the many custom drawing events, which is explained in the Customization chapter.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.ModeSettings.StartTime := Now;
TMSFMXPlanner1.TimeLine.DisplayUnit := 20;
TMSFMXPlanner1.TimeLine.DisplayUnitSize := 50;
TMSFMXPlanner1.TimeLine.CurrentTimeMode := pctmText;
TMSFMXPlanner1.TimeLineAppearance.CurrentTimeColor := claBlue;
TMSFMXPlanner1.TimeLine.DisplayEnd := Round(MinsPerDay /
TMSFMXPlanner1.TimeLine.DisplayUnit) - 1;
TMSFMXPlanner1.EndUpdate;
TMSFMXPlanner1.TimeLine.ViewStart :=
Int(TMSFMXPlanner1.ModeSettings.StartTime) +
EncodeTime(HourOf(TMSFMXPlanner1.ModeSettings.StartTime), 0, 0, 0);
```




## Items (events)

When dropping a new instance of the planner (TTMSFMXPlanner) on the form, you will notice it already has a default item. The item has a title and text area and its position within the grid is based on the StartTime, EndTime and Resource properties. The text area supports HTML formatted text including hyperlink detection. In the pmMultiDay, pmMultiMonth modes, the items can stretch over multiple positions depending on the StartTime and EndTime. In the pmDay, pmHalfDayPeriod, pmDayPeriod, pmMonth and pmCustom modes, the position is set with the Resource property. The pmMultiDayRes and pmMultiResDay modes combine all three properties to position its items.

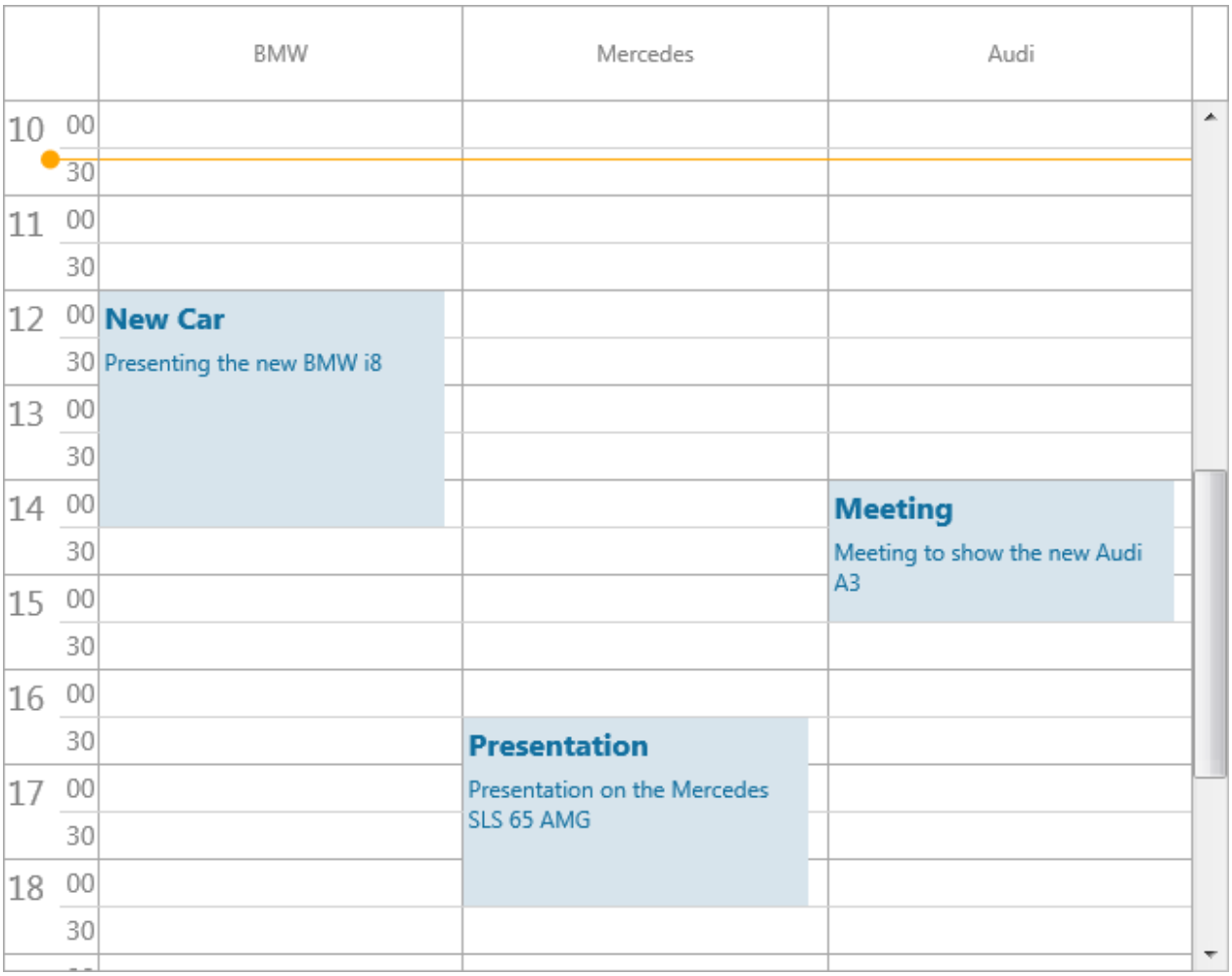



The planner has a DefaultItem property that can be used to preset item property settings that will be applied to all new created items. Adding items can be done with Items.Add or with one of the AddOrUpdateItem overload functions. Below are some samples that demonstrate this in various modes.

The first sample shows the default view for the pmDay mode, displays three resources and adds an item for each resource. Additionally, it initializes the view scrolling position to a specific datetime value.

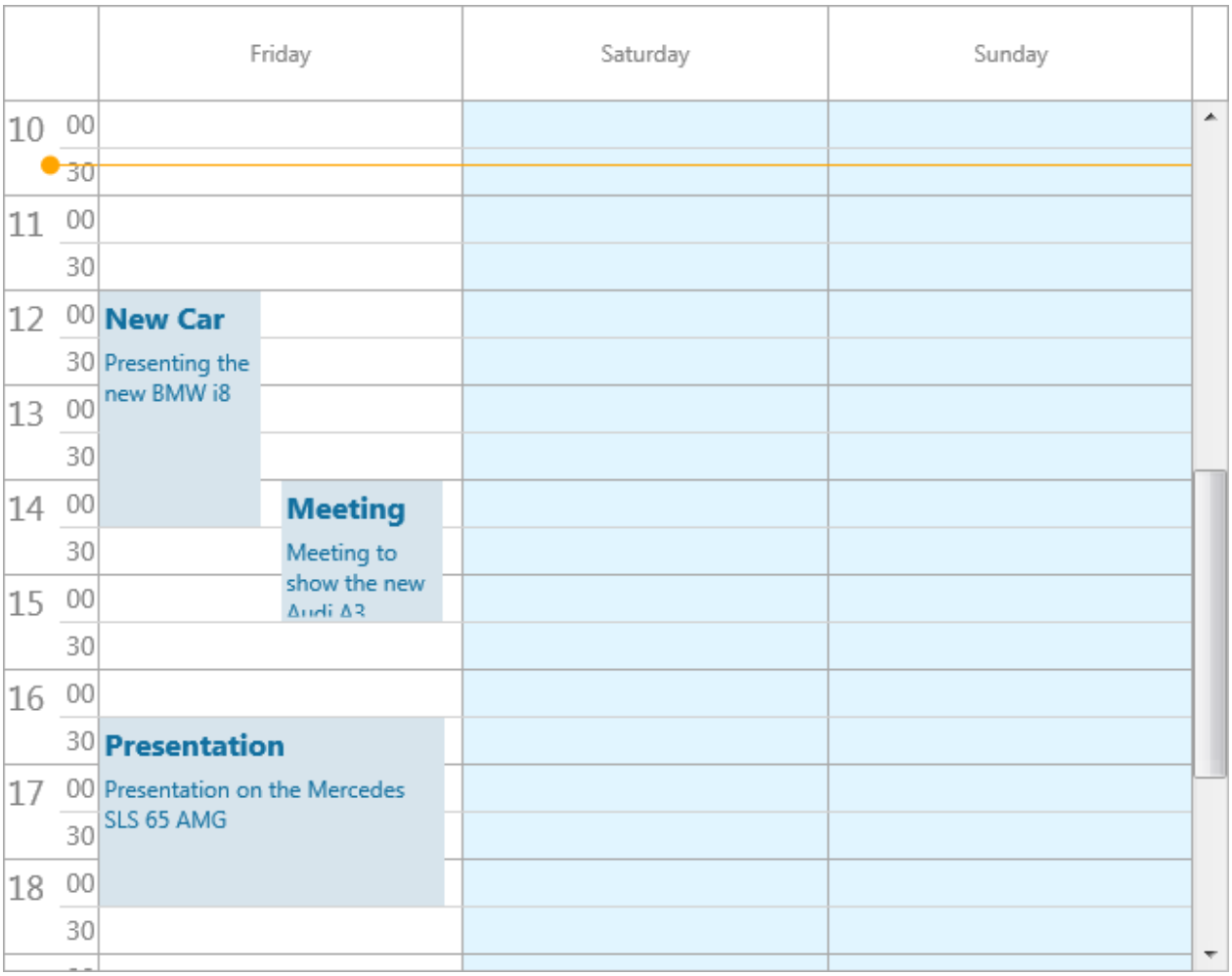
```
dt := Int(Now);
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.ModeSettings.StartTime := dt;
TMSFMXPlanner1.Items.Clear;
```

```
TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(12, 0, 0, 0), dt +
EncodeTime(14, 30, 0, 0), 'New Car', 'Presenting the new BMW
i8').Resource := 0;
TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(16, 30, 0, 0), dt +
EncodeTime(18, 30, 0, 0), 'Presentation', 'Presentation on the
Mercedes SLS 65 AMG').Resource := 1;
TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(14, 0, 0, 0), dt +
EncodeTime(15, 30, 0, 0), 'Meeting', 'Meeting to show the new Audi
A3').Resource := 2;
TMSFMXPlanner1.EndUpdate;
TMSFMXPlanner1.TimeLine.ViewStart := dt + EncodeTime(10, 0, 0, 0);
```




Note that the AddOrUpdateItem function returns an item reference and the Resource property is set to 0, 1 and 2 respectively. If we would add items without setting the Resource property, the items would all be placed on the first position.

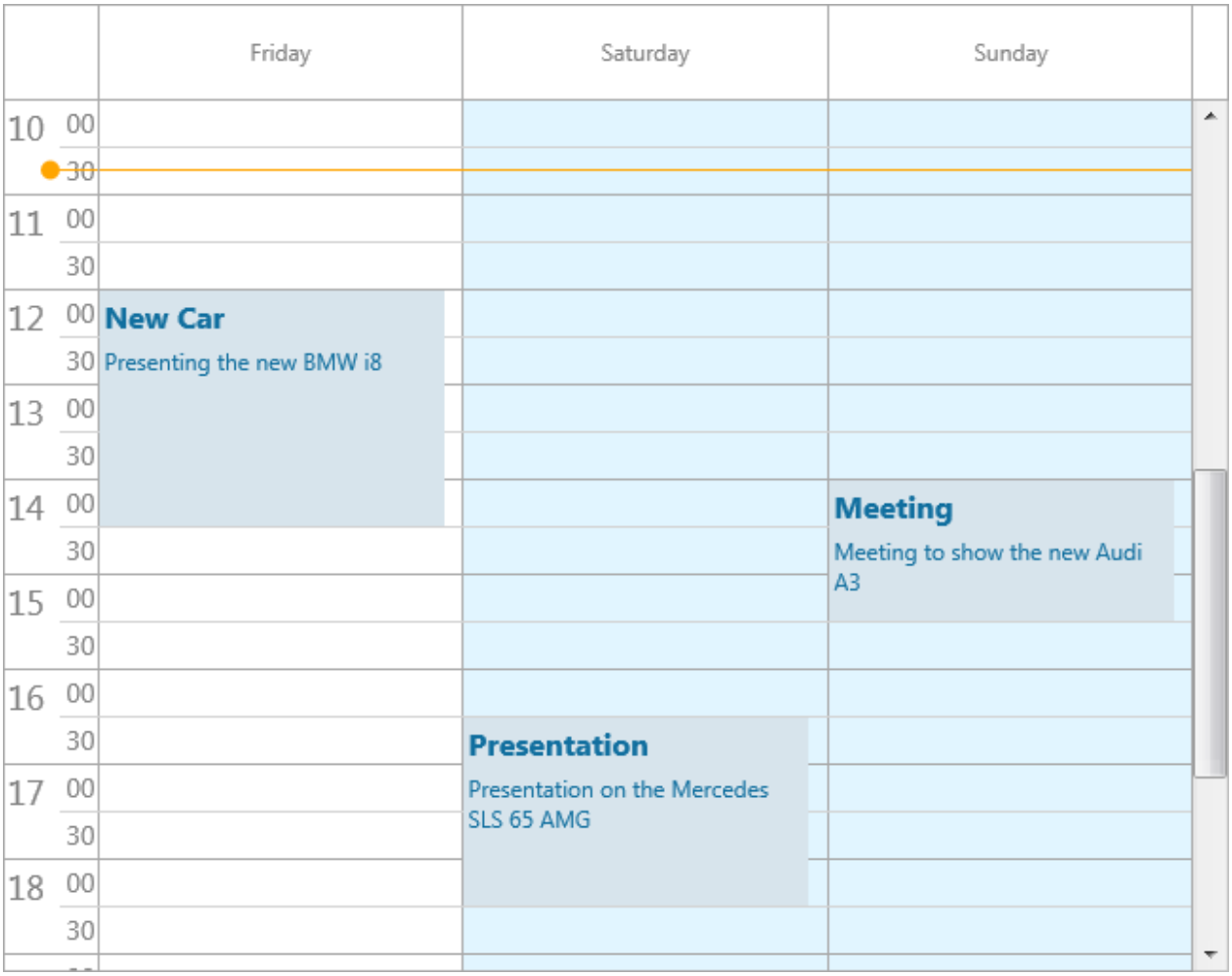
If we now change this to pmMultiDay mode, you will notice that all items will be on the same position. Since all items are added on the same day through the StartTime and EndTime properties.




If we want to change the position of the Mercedes “Presentation” item to Saturday and the Audi “Meeting” item to Sunday, we need to increase the StartTime and EndTime with 1 and 2 days respectively. The sample below demonstrates this.

```
dt := Int(Now);
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmMultiDay;
TMSFMXPlanner1.ModeSettings.StartTime := dt;
TMSFMXPlanner1.Items.Clear;
```

```
TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(12, 0, 0, 0), dt +
EncodeTime(14, 30, 0, 0), 'New Car', 'Presenting the new BMW i8');
TMSFMXPlanner1.AddOrUpdateItem(dt + 1 + EncodeTime(16, 30, 0, 0), dt +
1 + EncodeTime(18, 30, 0, 0), 'Presentation', 'Presentation on the
Mercedes SLS 65 AMG');
TMSFMXPlanner1.AddOrUpdateItem(dt + 2 + EncodeTime(14, 0, 0, 0), dt +
2 + EncodeTime(15, 30, 0, 0), 'Meeting', 'Meeting to show the new Audi
A3');
TMSFMXPlanner1.EndUpdate;
TMSFMXPlanner1.TimeLine.ViewStart := dt + EncodeTime(10, 0, 0, 0);
```

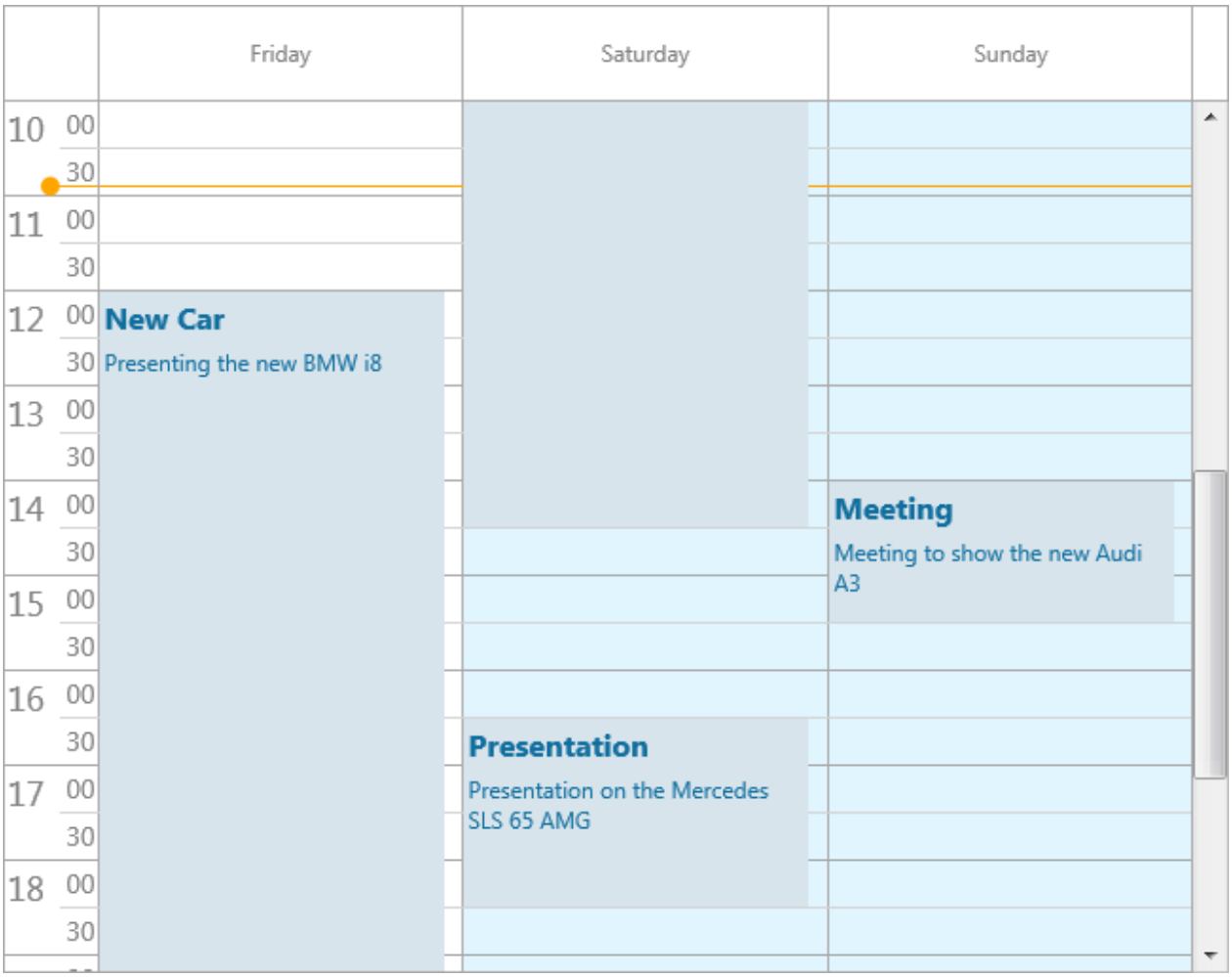



The items are now shown in a similar way as pmDay mode, but with the difference that they are not linked to any resource, but instead are placed in the position that displays the day.

As explained in the beginning of this chapter, the item can stretch over multiple positions in some modes. When we change the code to allow an item to have an EndTime that ends on the next day, we get an item that is stretched over 2 positions. Important to know is that the item repeats the title and text for every position it is stretched on. When we change the ViewStart property to scroll to the beginning of the display, you will notice the BMW item will be drawn in the same day as the Mercedes item, which starts at a later time.

Below is a sample that demonstrates this.

```
dt := Int(Now);
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmMultiDay;
TMSFMXPlanner1.ModeSettings.StartTime := dt;
TMSFMXPlanner1.Items.Clear;
TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(12, 0, 0, 0), dt + 1 +
EncodeTime(14, 30, 0, 0), 'New Car', 'Presenting the new BMW i8');
TMSFMXPlanner1.AddOrUpdateItem(dt + 1 + EncodeTime(16, 30, 0, 0), dt +
1 + EncodeTime(18, 30, 0, 0), 'Presentation', 'Presentation on the
Mercedes SLS 65 AMG');
TMSFMXPlanner1.AddOrUpdateItem(dt + 2 + EncodeTime(14, 0, 0, 0), dt +
2 + EncodeTime(15, 30, 0, 0), 'Meeting', 'Meeting to show the new Audi
A3');
TMSFMXPlanner1.EndUpdate;
TMSFMXPlanner1.TimeLine.ViewStart := dt + EncodeTime(10, 0, 0, 0);
```

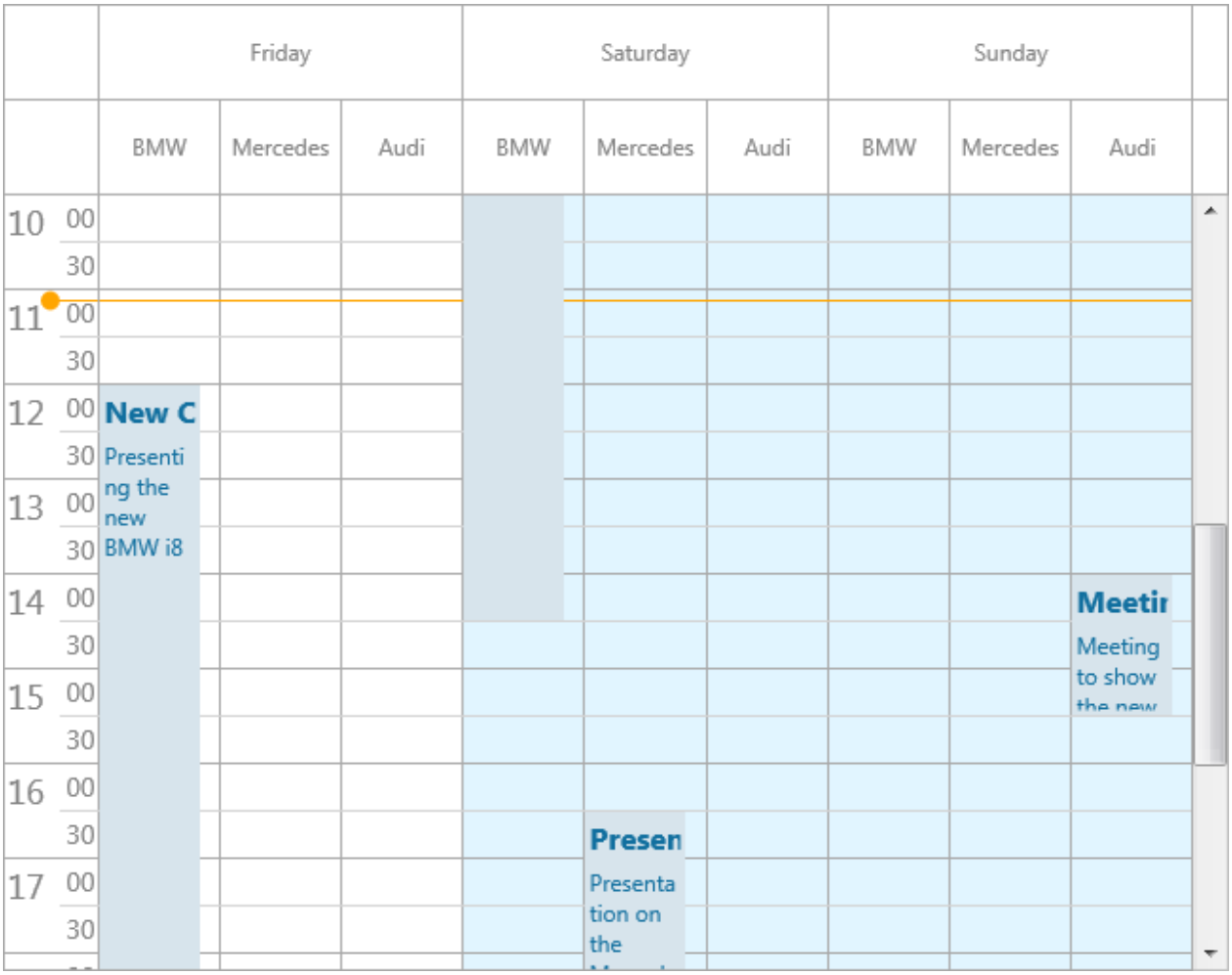



The pmDayPeriod, pmHalfDayPeriod, pmMonth, pmMultiMonth and pmCustom modes can use the same approach as the pmDay and pmMultiDay mode samples in this chapter, but with different settings of StartTime and EndTime.

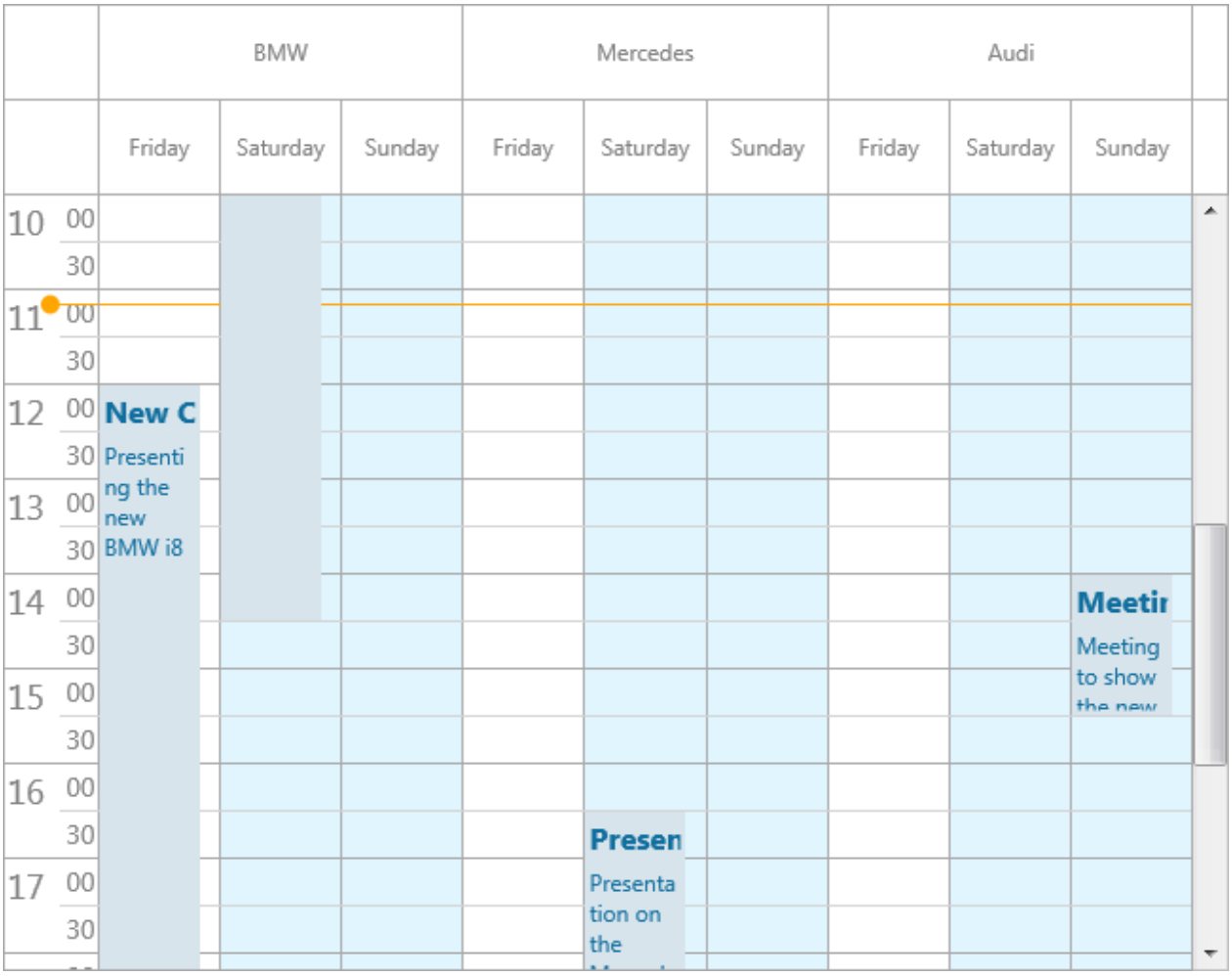
As explained, the pmMultiResDay and pmMultiDayRes modes combine the StartTime, EndTime and Resource properties into a single view. Below is a sample that demonstrates this. To display all items we will need to increase the positions count.

```
dt := Int(Now);
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmMultiResDay;
TMSFMXPlanner1.ModeSettings.StartTime := dt;
TMSFMXPlanner1.Items.Clear;
```

```
TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(12, 0, 0, 0), dt + 1 +
EncodeTime(14, 30, 0, 0), 'New Car', 'Presenting the new BMW
i8').Resource := 0;
TMSFMXPlanner1.AddOrUpdateItem(dt + 1 + EncodeTime(16, 30, 0, 0), dt +
1 + EncodeTime(18, 30, 0, 0), 'Presentation', 'Presentation on the
Mercedes SLS 65 AMG').Resource := 1;
TMSFMXPlanner1.AddOrUpdateItem(dt + 2 + EncodeTime(14, 0, 0, 0), dt +
2 + EncodeTime(15, 30, 0, 0), 'Meeting', 'Meeting to show the new Audi
A3').Resource := 2;
TMSFMXPlanner1.Positions.Count := 9;
TMSFMXPlanner1.EndUpdate;
TMSFMXPlanner1.TimeLine.ViewStart := dt + EncodeTime(10, 0, 0, 0);
```



Note how the BMW item still stretches over multiple positions, but keeps displaying in the same resource. The other items are placed on Saturday and Sunday in their respective resource. When we change to the other mode pmMultiDayRes, we get a different view.



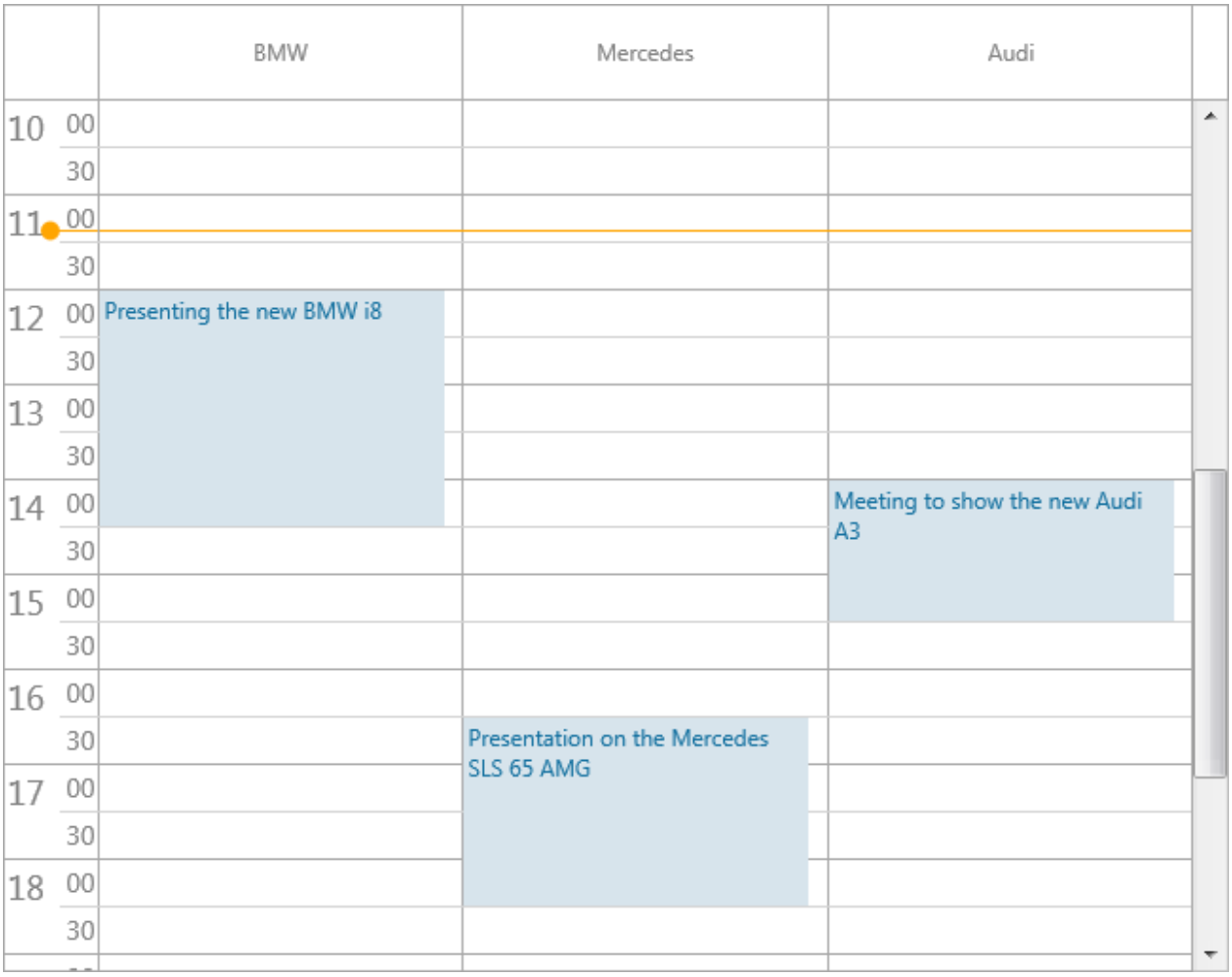

The BMW item is stretched over multiple days, but the days are repeated for each resource in this mode.

## Default item

The DefaultItem property can be used to completely preset how an item should look and feel before adding it. Below is a sample that demonstrates this on one of the previous samples.

```
dt := Int(Now);
TMSFMXPlanner1.BeginUpdate;
```

```
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.ModeSettings.StartTime := dt;
TMSFMXPlanner1.Items.Clear;
TMSFMXPlanner1.DefaultItem.ShowTitle := False;
TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(12, 0, 0, 0), dt +
EncodeTime(14, 30, 0, 0), 'New Car', 'Presenting the new BMW
i8').Resource := 0;
TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(16, 30, 0, 0), dt +
EncodeTime(18, 30, 0, 0), 'Presentation', 'Presentation on the
Mercedes SLS 65 AMG').Resource := 1;
TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(14, 0, 0, 0), dt +
EncodeTime(15, 30, 0, 0), 'Meeting', 'Meeting to show the new Audi
A3').Resource := 2;
TMSFMXPlanner1.EndUpdate;
TMSFMXPlanner1.TimeLine.ViewStart := dt + EncodeTime(10, 0, 0, 0);
```

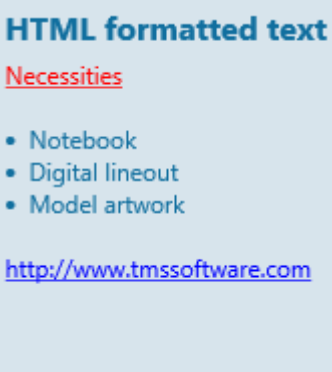


The sample code sets the ShowTitle property of the DefaultItem to False. Each item that is created takes over the DefaultItem settings.

## HTML formatted text

The item text area is capable of displaying HTML formatted text with hyperlink detection. As soon as HTML tags are detected, the text will be rendered in HTML. Below is a sample that demonstrates this.

```
TMSFMXPlanner1.Items[0].Title := 'HTML formatted text';
TMSFMXPlanner1.Items[0].Text := '<u><font
color="#FF0000">Necessities</font></u> <br><ul><li>Notebook<li>Digital
lineout<li>Model artwork</ul><br><br><a
href="http://www.tmssoftware.com">http://www.tmssoftware.com</a>';
```



## Item Linking

A planner item can be linked in various ways to another planner item. Linking two items means that if the user will move or size one item, the linked item can also move or size, depending on the link type. A link is a relationship between two items. It is not possible to link one item to more than one other item but chained linking is possible, even circular chained linking. Linking is achieved through 2 planner item properties:

TTMSFMXPlannerItem.**LinkedItem**: TTMSFMXPlannerItem; defines to which the item is linked
TTMSFMXPlannerItem.**LinkType**: TTMSFMXPlannerItemLinkType; defines the type of the link

The LinkType can be:

**iltFull**: both StartTime and EndTime are linked. This means that item duration is always synchronised between the items. When the item moves or sizes, both begin and end of the linked item will do the same move or size.
**iltStartEnd**: StartTime of the item is linked to the EndTime of the linked item. This means that if the StartTime of the item changes, the EndTime of the linked item will change with the same delta
**iltEndStart**: EndTime of the item is linked to the StartTime of the linked item
**iltEndEnd**: EndTime of the item is linked to the EndTime of the linked item
**iltStartStart**: StartTime of the item is linked to the StartTime of the linked item
**iltNone**: the items are linked but in a loose relationship. This means that moving or sizing of linked items will not affect the size or position of other items.

With linked items, it is possible that when selecting one item in a chain of linked items, all linked items will become selected automatically. To enable this, set Planner.Interaction.MultiSelect = True and Planner.Interaction.AutoSelectLinkedItems = true.

**Example:**

```
var
  dt: TDateTime;
  it1, it2: TTMSFMXPlannerItem;
begin
  TMSFMXPlanner1.BeginUpdate;
  TMSFMXPlanner1.Items.Clear;
  TMSFMXPlanner1.Mode := pmMultiResDay;
  dt := Int(Now);
```

```
  it1 := TMSFMXPlanner1.AddItem(dt + EncodeTime(10, 0, 0, 0), dt +
EncodeTime(12, 0, 0, 0));
  it1.Title := 'Sample';
  it1.Resource := 0;

  it2 := TMSFMXPlanner1.AddItem(dt + EncodeTime(8, 0, 0, 0), dt +
EncodeTime(10, 30, 0, 0));
  it2.Title := 'Linked Item';
  it2.Resource := 1;

  it1.LinkedItem := it2;
  it1.LinkType := iltFull;

  TMSFMXPlanner1.ViewRow := 14;
  TMSFMXPlanner1.Interaction.MultiSelect := True;
  TMSFMXPlanner1.Interaction.AutoSelectLinkedItems := True;
  TMSFMXPlanner1.EndUpdate;
end;
```

Some additional methods are available on Planner level to facilitate handling item linking:

procedure **LinkItems(AItems: TTMSFMXPlannerLinkedItemArray; ACircular: Boolean = false; ALinkType: TTMSFMXPlannerItemLinkType = ltLinkNone);**

Sets up a link between all items in the array. By default, this is a chained link from item 0 in the array to the last item. When parameter ACircular = true, a circular chained link is created. The last parameter sets the link type.

procedure **LinkItems(AItems: TTMSFMXPlannerLinkedItemArray);**

Breaks the link between all items in the array.

procedure **SelectedLinkedItems(AItem: TTMSFMXPlannerItem)**

Selects all items that are linked (in chain) to AItem

function **FindItemLinkedTo(AItem: TTMSFMXPlannerItem): TTMSFMXPlannerItem;**

Returns the item that is linked to APlannerItem.

Optionally, the Planner can also visually show linked items by drawing an interconnection line between linked items. This featured is enabled by setting Planner.ItemsAppearance.ShowLinks = true. The color of the interconnection line between two items is set by PlannerItem.LinkColor.

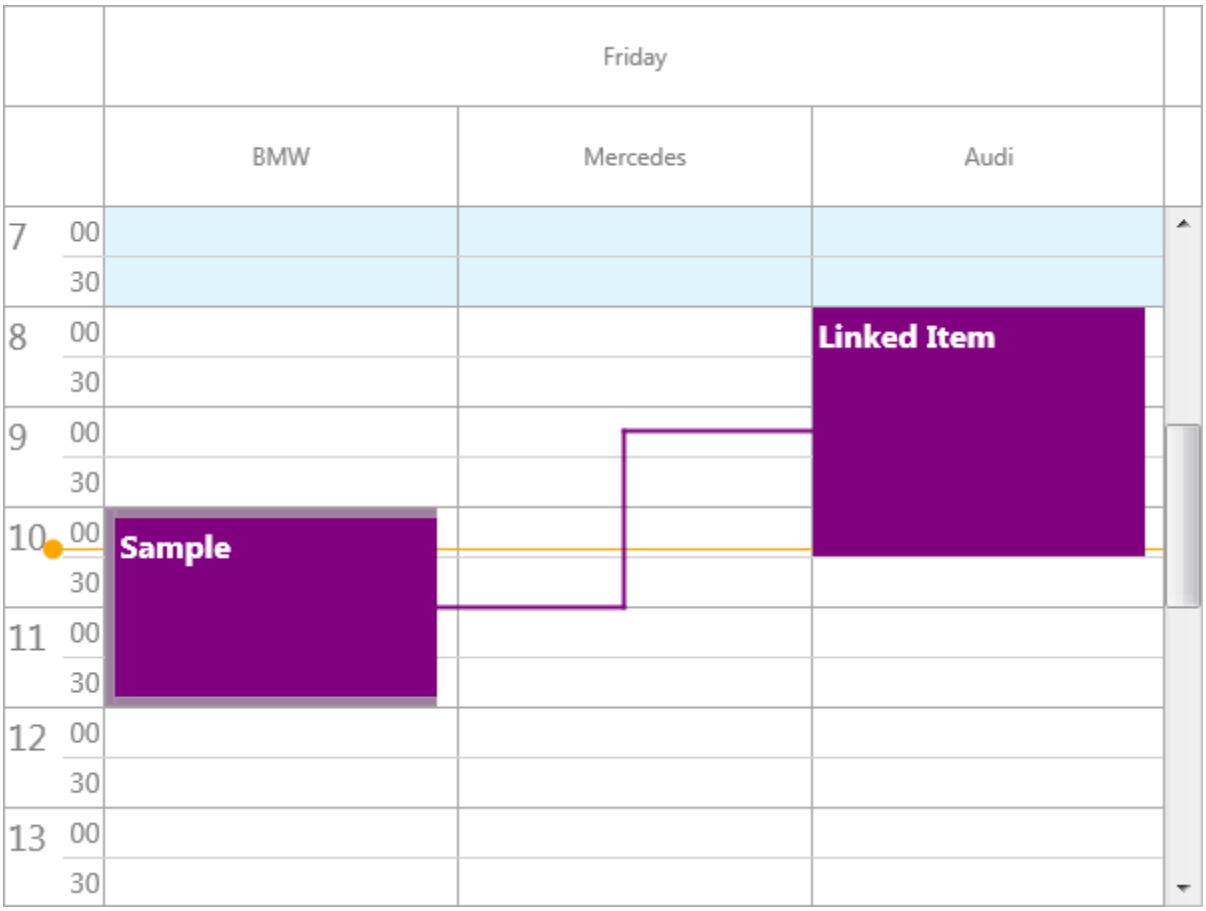
**Example:**

```
var
  dt: TDateTime;
  it1, it2: TTMSFMXPlannerItem;
begin
  TMSFMXPlanner1.BeginUpdate;
  TMSFMXPlanner1.Items.Clear;
  TMSFMXPlanner1.Mode := pmMultiResDay;
  dt := Int(Now);
  it1 := TMSFMXPlanner1.AddItem(dt + EncodeTime(10, 0, 0, 0), dt +
EncodeTime(12, 0, 0, 0));
  it1.Title := 'Sample';
  it1.Resource := 0;
  it1.ActiveColor := claPurple;
  it1.SelectedColor := claPurple;
  it1.SelectedLinkColor := claPurple;

  it2 := TMSFMXPlanner1.AddItem(dt + EncodeTime(8, 0, 0, 0), dt +
EncodeTime(10, 30, 0, 0));
  it2.Title := 'Linked Item';
  it2.Resource := 2;
  it2.ActiveColor := claPurple;
  it2.SelectedColor := claPurple;
  it2.SelectedLinkColor := claPurple;

  it1.LinkedItem := it2;
  it1.LinkType := iltFull;

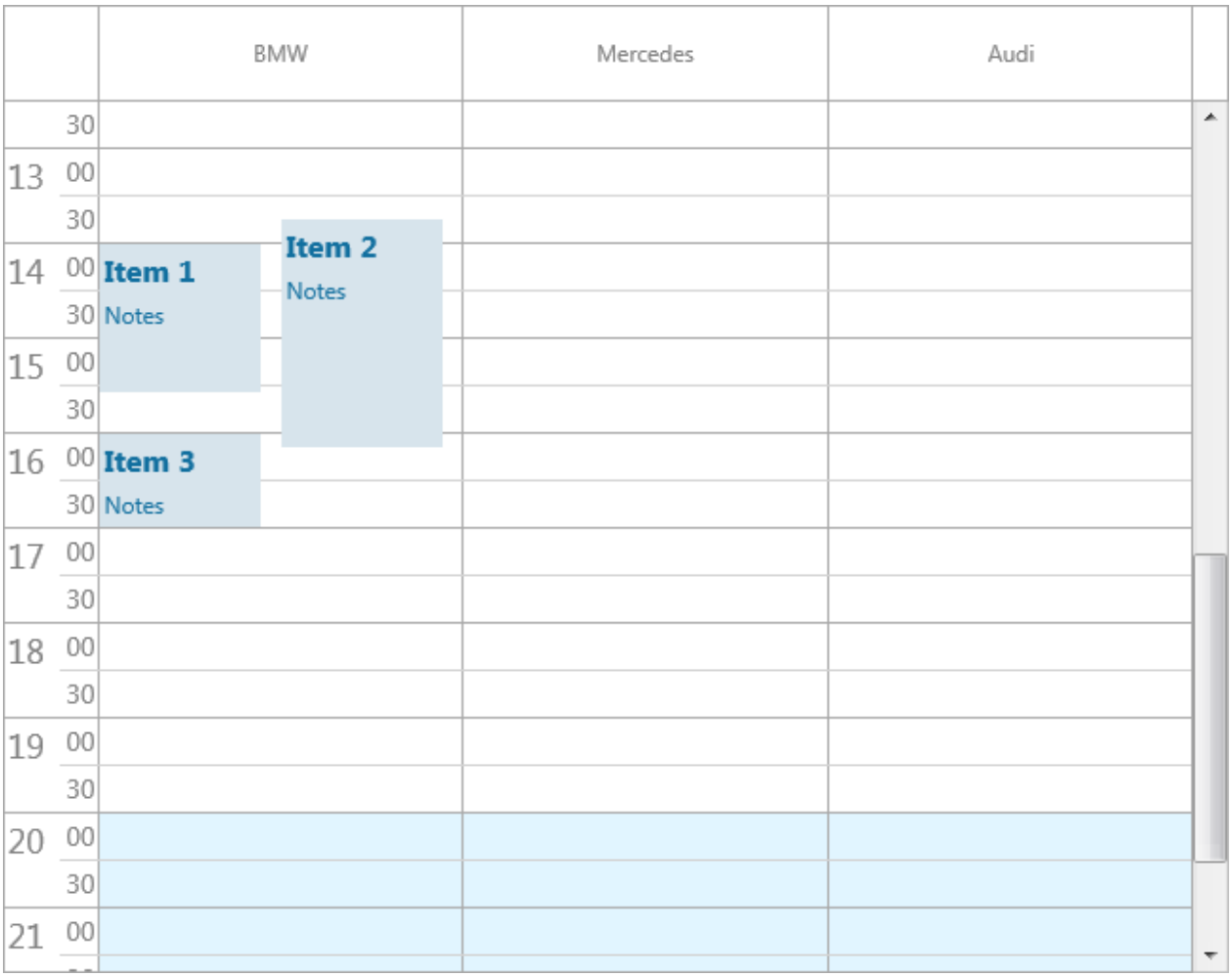
  TMSFMXPlanner1.ItemsAppearance.ShowLinks := True;
  TMSFMXPlanner1.Interaction.AutoSelectLinkedItems := True;
  TMSFMXPlanner1.Interaction.MultiSelect := True;
  TMSFMXPlanner1.ViewRow := 14;
  TMSFMXPlanner1.EndUpdate;
end;
```


Friday
BMW
Mercedes
Audi
7
8
9
10
11
12
13
00
30
Linked Item
Sample

## Overlapping items

Items that are placed at the same position at the same time interval are overlapping items. Overlapping is enabled by default for all items and can be turned off globally with the property ModeSettings.OverlappableItems. Each item has a property Overlappable which can be used to control per item if the item is overlappable or not. If the items overlap, they contain a list of conflicts and conflict positions. As the item can be stretched over multiple positions, the count of conflicts can be retrieved by passing the position as a parameter to the ConflictsForPosition function on item level. Below is a sample that demonstrates this.

```
  TMSFMXPlanner1.BeginUpdate;
  TMSFMXPlanner1.Mode := pmDay;
  TMSFMXPlanner1.Items.Clear;
  dt := Int(Now);
  TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(14, 0, 0, 0), dt +
EncodeTime(15, 35, 0, 0) , 'Item 1', 'Notes');
  TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(13, 45, 0, 0), dt +
EncodeTime(16, 10, 0, 0) , 'Item 2', 'Notes');
  TMSFMXPlanner1.TimeLine.CurrentTimeMode := pctmNone;
  TMSFMXPlanner1.EndUpdate;
  TMSFMXPlanner1.TimeLine.ViewStart := dt + EncodeTime(12, 30, 0, 0);
```




The `AItem.ConflictsForPosition(0)` will return 2 for the first and the third item. The second item will have 3 conflicts. To know the position of the item if it has a conflict is to use the `AItem.ConflictsPosForPosition` function.

In the above sample, moving the items is possible, but by setting the overlappable property to false for an item, that particular item is not overlappable. An item that might possibly have a conflict with an item that is not overlappable will not be able to move to that position. The item that is not overlappable can be moved anywhere since all the other items are overlappable.

Note that while the planner will control that with the user moving items, the rules for overlap will be respected, the planner does not perform checks when programmaticalyl inserting items. If an item cannot be programmatically created because it would overlap with an existing non-overlappable item, this should be checked at application level and when needed

and appropriate warning should be given to the user for the reason the item cannot be created.

## Appearance

The overall appearance of the item can be set with the ItemsAppearance properties. The ItemsAppearance properties define the kind of fill / stroke that is used for various states (which are explained in the Interaction chapter).

Each item has color and font properties for the title and text area for different states. Each state has its own set of properties. By default the item takes over the default color settings from the DefaultItem property, but after the items are added, the items can be further customized. We take the first pmDay sample again, which shows three items placed on different resources. The AddOrUpdateItem returns a reference to the newly created item. In this sample we define a TTMSFMXPlannerItem variable and then set the Color, FontColor and TitleFontColor for each item.

```
dt := Int(Now);
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmDay;
TMSFMXPlanner1.ModeSettings.StartTime := dt;
TMSFMXPlanner1.Items.Clear;
it := TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(12, 0, 0, 0), dt
+ EncodeTime(14, 30, 0, 0), 'New Car', 'Presenting the new BMW i8');
it.Resource := 0;
it.Color := claLightSteelBlue;
it.FontColor := claWhite;
it.TitleFontColor := claWhite;
it := TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(16, 30, 0, 0), dt
+ EncodeTime(18, 30, 0, 0), 'Presentation', 'Presentation on the
Mercedes SLS 65 AMG');
it.Resource := 1;
it.Color := claLightgoldenrodyellow;
it.FontColor := claRed;
it.TitleFontColor := claRed;
it := TMSFMXPlanner1.AddOrUpdateItem(dt + EncodeTime(14, 0, 0, 0), dt
+ EncodeTime(15, 30, 0, 0), 'Meeting', 'Meeting to show the new Audi
A3');
it.Resource := 2;
it.Color := claGreenyellow;
it.FontColor := claGreen;
```

```
it.TitleFontColor := claGreen;
TMSFMXPlanner1.EndUpdate;
TMSFMXPlanner1.TimeLine.ViewStart := dt + EncodeTime(10, 0, 0, 0);
```




In the Interaction chapter, you will see that each item has a normal, disabled, selected and active state. The properties in this sample are based on the normal state, but other properties can be used to give the item a unique look and feel for all states.

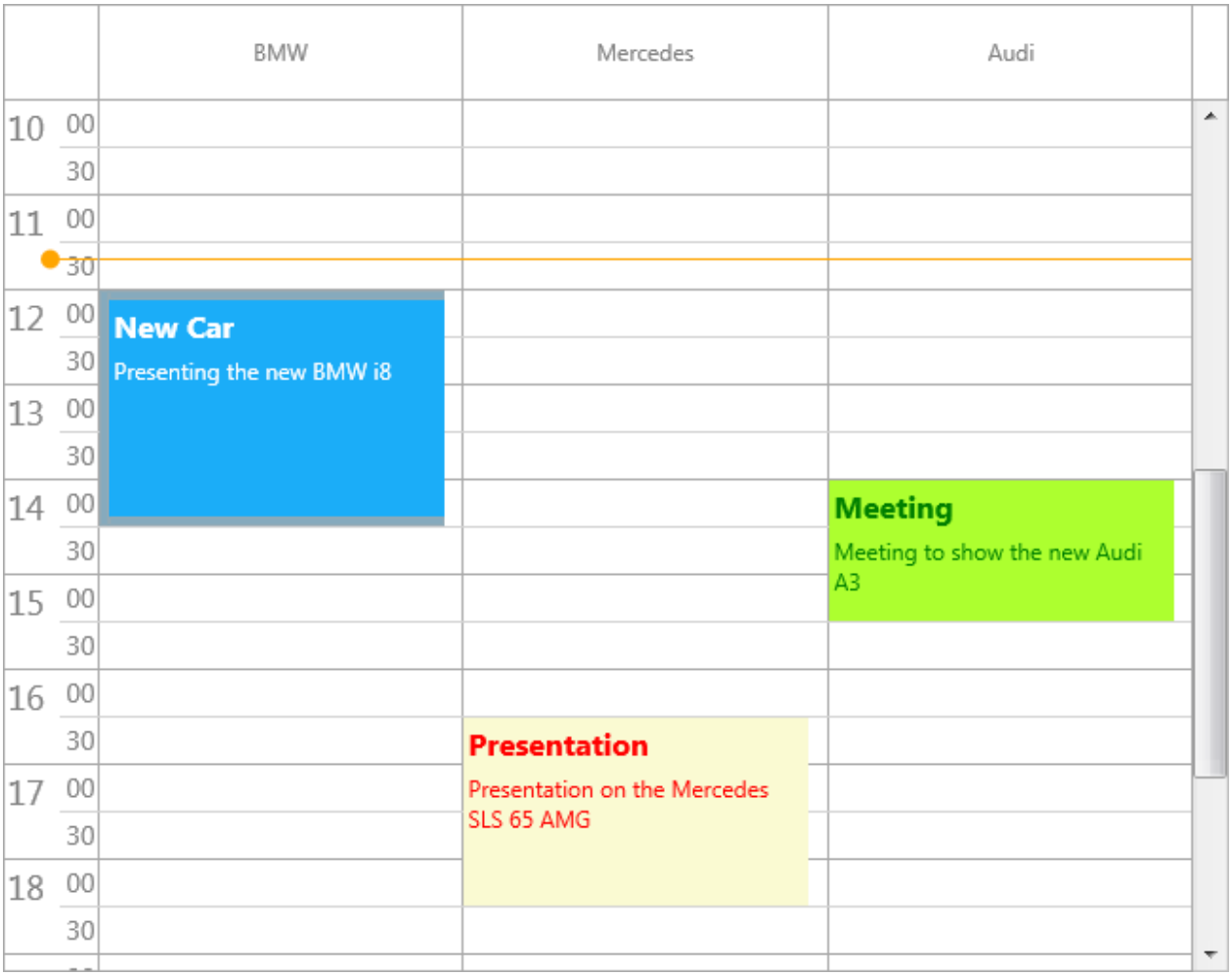
## Interaction

### Items

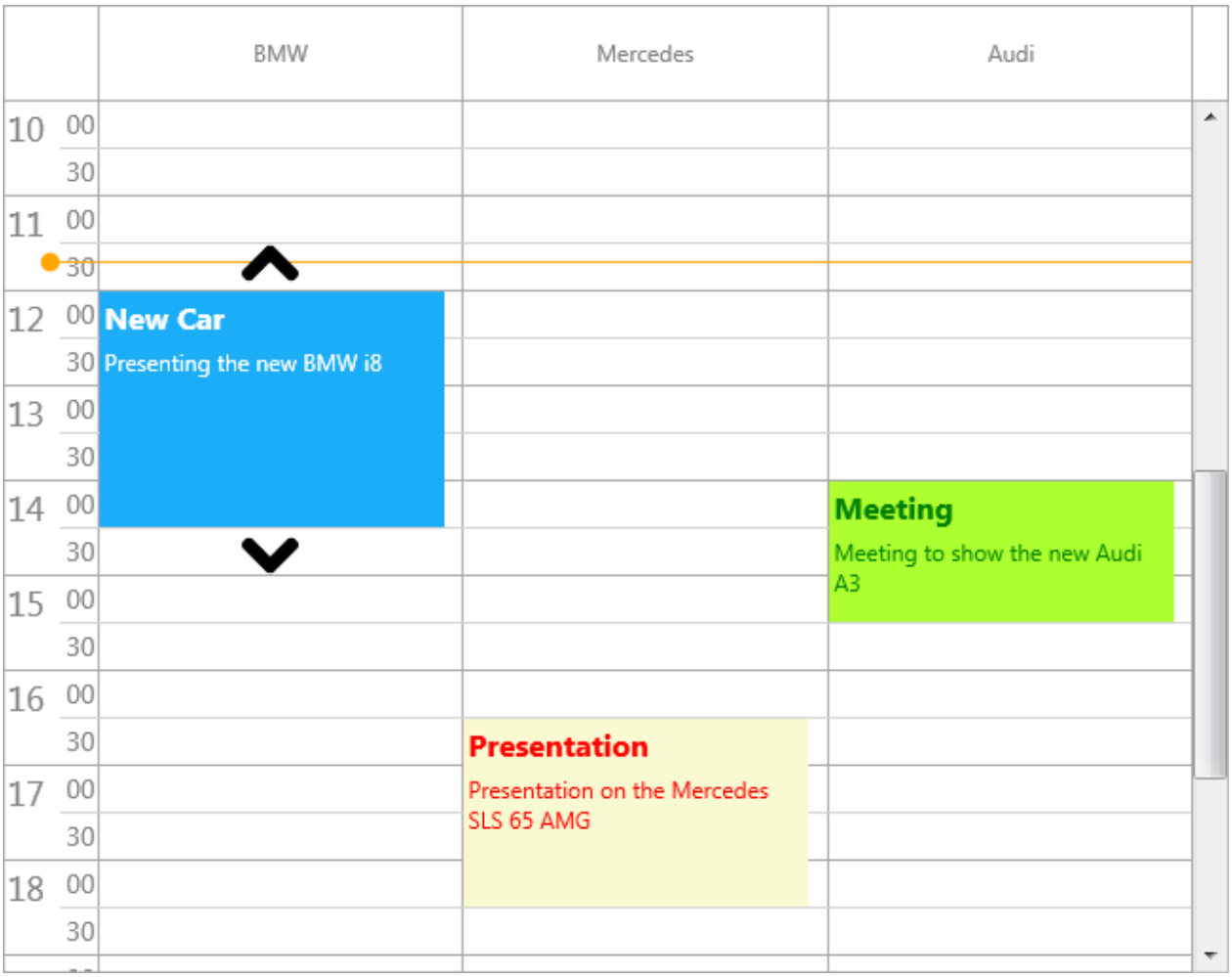
We continue with the previous sample, which shows three items in pmDay mode. The planner supports selection, moving and sizing of items. These interaction modes can be configured

per item. Each item has a Movable, Sizeable and Selectable property. When we click on the BMW item, you will notice it changes the color settings to the active state. The item also shows a move and size helper which can be used to change the position, start and end time of the item. On mobile devices, the move and size helper areas are replaced with customizable arrows as seen in the screenshots below:

Desktop

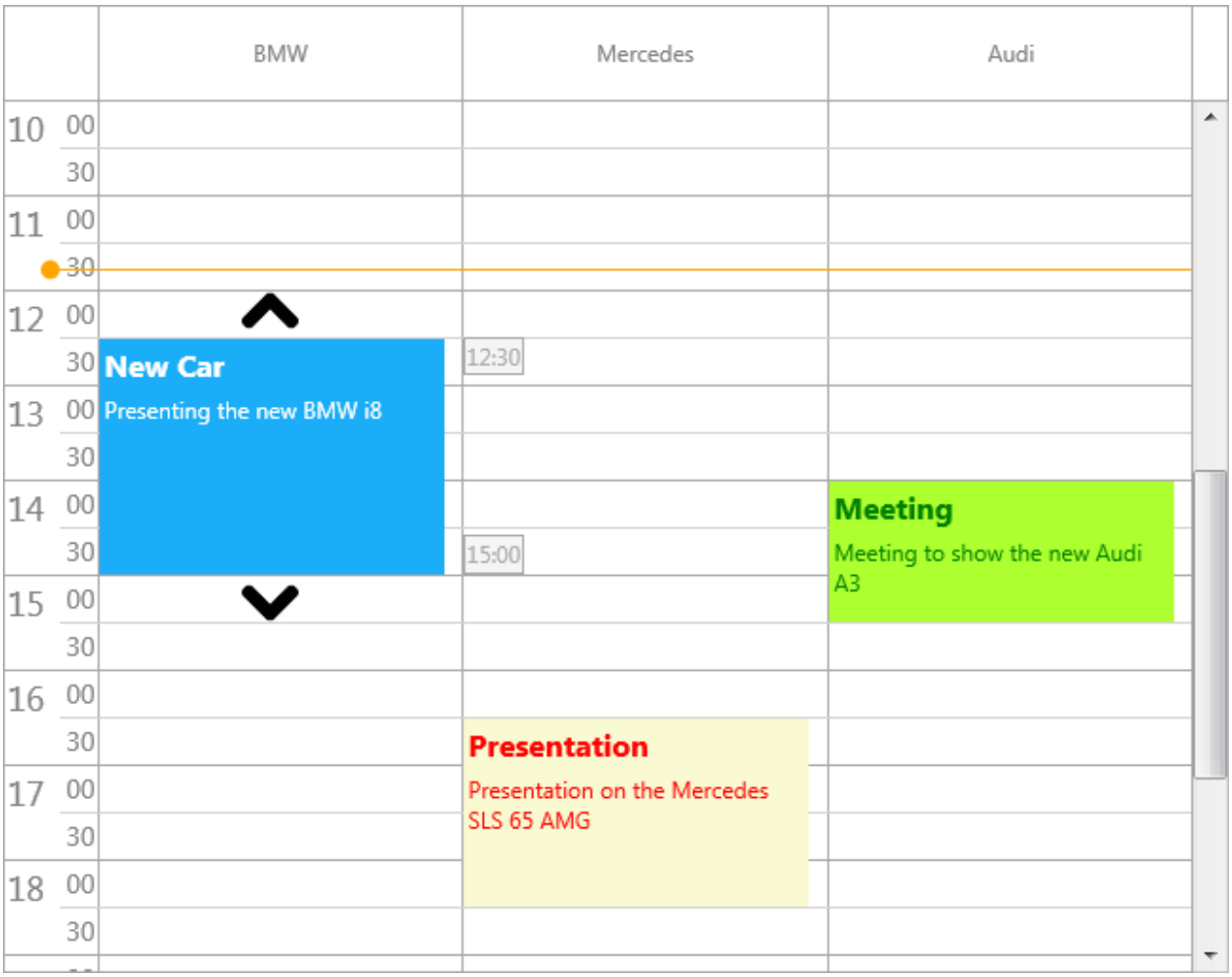



Mobile




The behavior of sizing and moving can be changed with the Interaction.SizeMode and Interaction.MoveMode properties. In mobile mode, sizing is done by clicking and dragging the arrows and moving is done by tapping and holding the finger down on an item. You will notice the sizing or moving operation is active when 2 additional helper controls are visible that indicate the start and end time of the item. These helper controls are optional.

In desktop mode, moving and sizing can also be done with the keyboard. The arrow keys can be used to move the item and the shift key to size the end time of the item. The start time can be changed by arrow key and holding the ctrl key in combination with the shift key.




## Selection / navigation

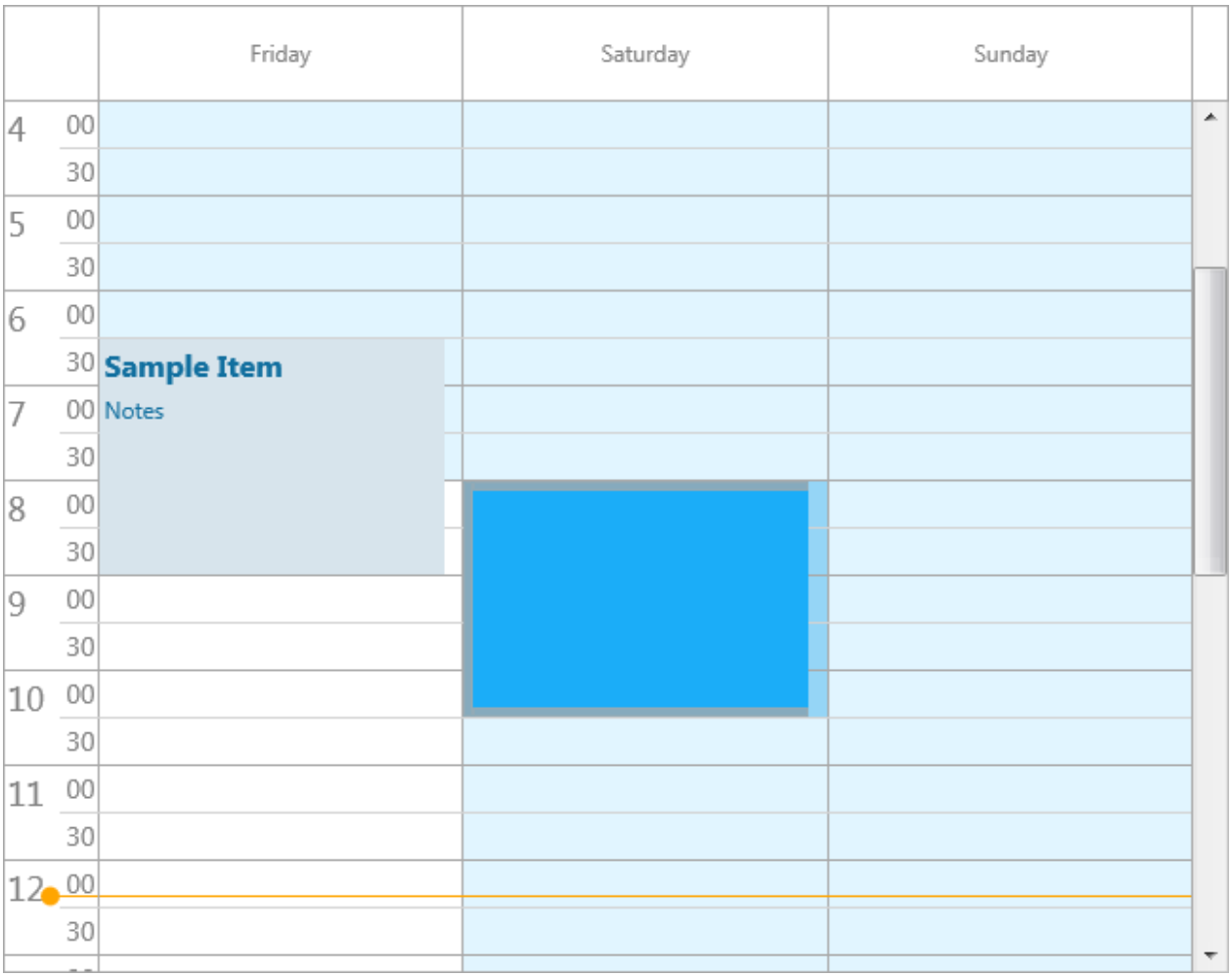
When clicking next to the item, on the grid area, the item will be unselected again. Selection and navigation can be done with the mouse / finger and keyboard (desktop only). On mobile operating systems, a single timeslot selection can be done by tapping. On desktop operating system, the same operating can be done with the mouse.

On mobile operating systems, tapping and holding the finger down will start a range selection. On desktop operating systems, the same keyboard shift, ctrl and arrow keys combination can be used to move the selection or change the selection range.

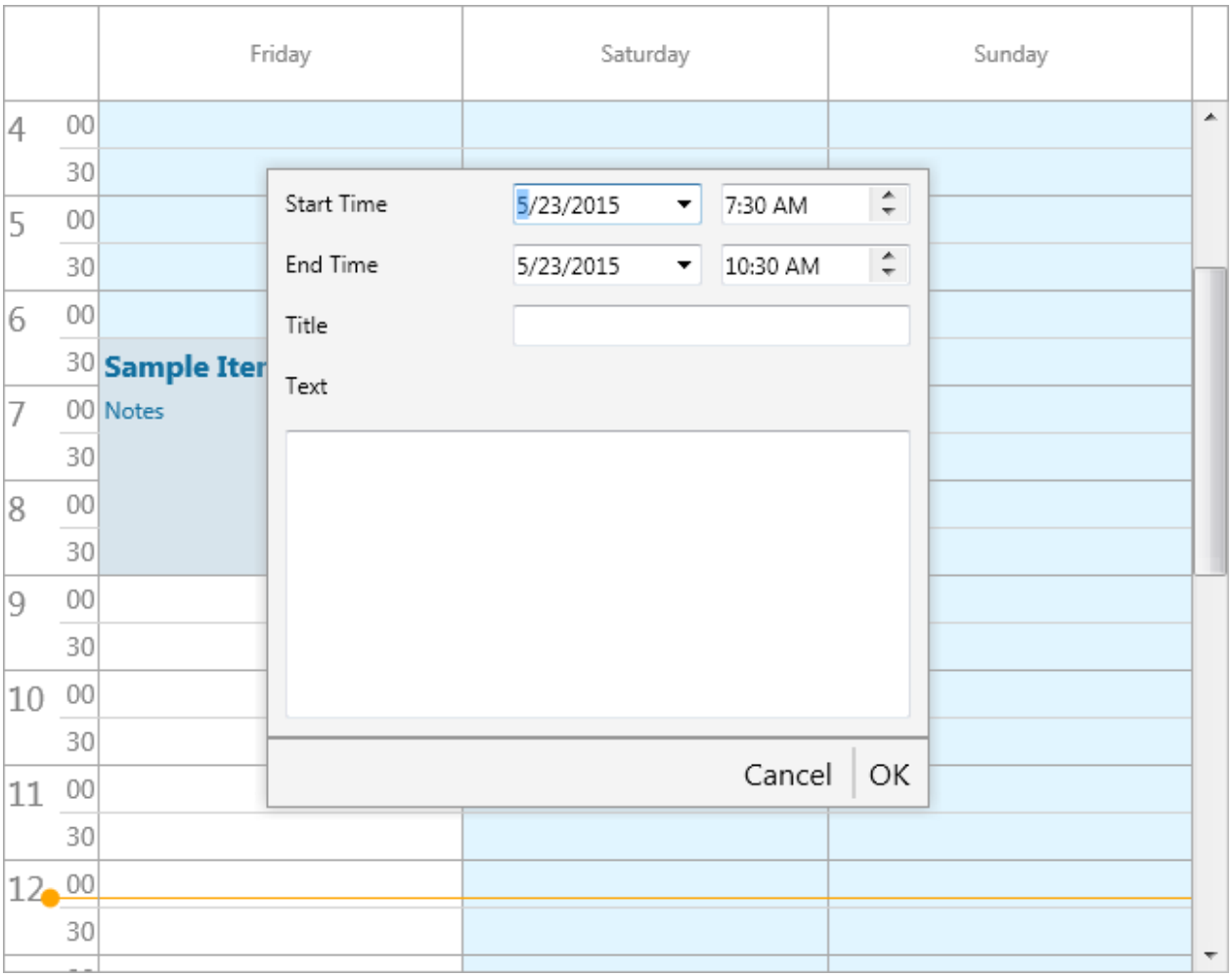
## Inserting new items

As demonstrated in the Items chapter, items can be added with Items.Add or with one of the AddOrUpdateItem overloads. Adding items can also be done after selecting a range of cells with the mouse or the finger, or when pressing insert on the keyboard. By default this way of adding items is disabled but can be enabled by changing the Interaction.MouseInsertMode and Interaction.KeyboardInsertMode. If the Interaction.MouseInsertMode is pmimAfterSelection, the item will be added immediately after a selection is made. If the Interaction.MouseInsertMode is pmimDialogAfterSelection, a built-in dialog is presented before the item is added. The dialog offers a way to customize the item before it is inserted at the selected timeslot range. The same action applies to the Interaction.KeyboardInsertMode after pressing the insert key. Below is a sample that adds a new item in both modes.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Interaction.KeyboardInsertMode := pkimSelection;
TMSFMXPlanner1.Interaction.MouseInsertMode := pmimAfterSelection;
TMSFMXPlanner1.EndUpdate;
```



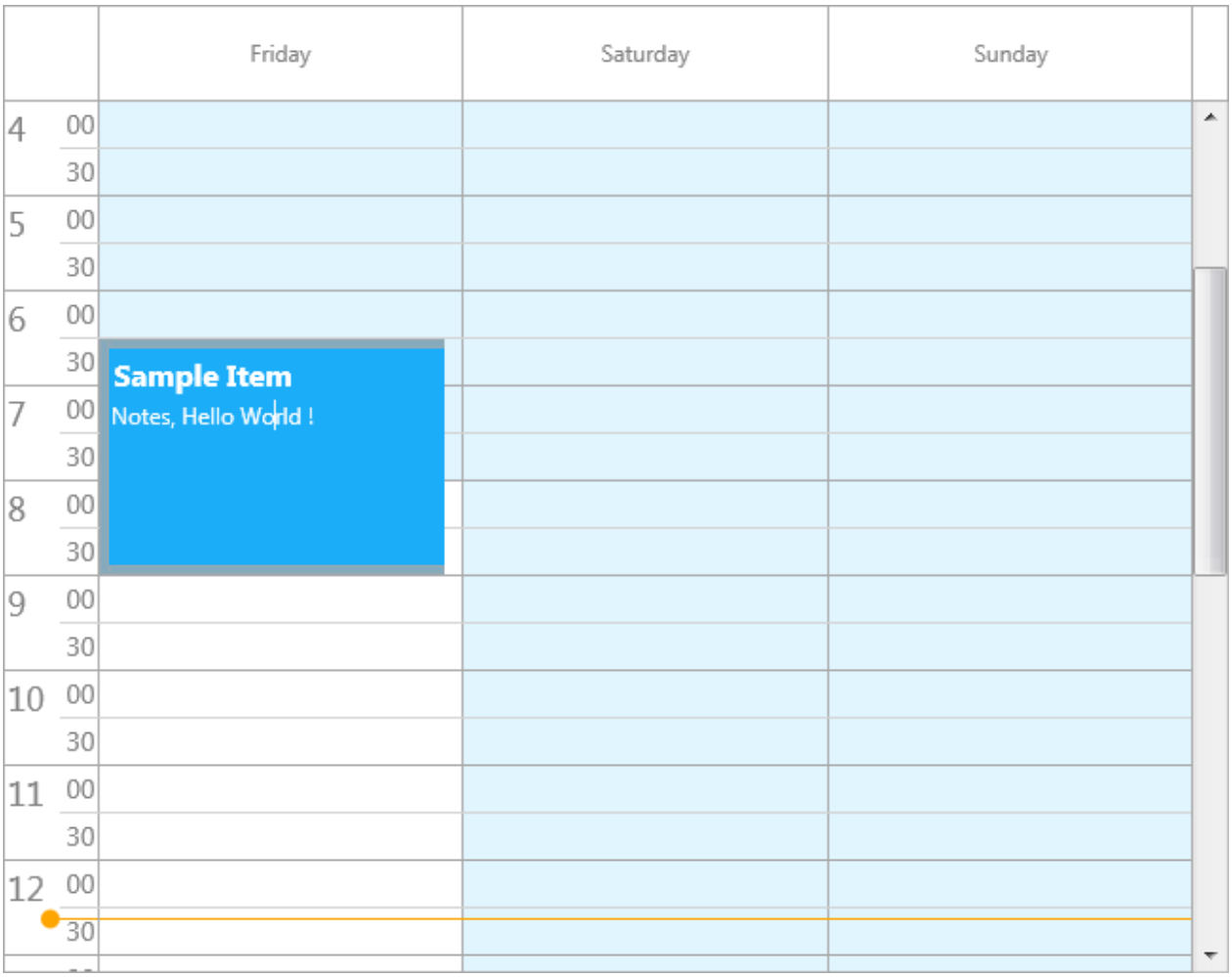

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Interaction.KeyboardInsertMode := pkimSelectionDialog;
TMSFMXPlanner1.Interaction.MouseInsertMode :=
pmimDialogAfterSelection;
TMSFMXPlanner1.EndUpdate;
```




The dialog can also be used for updating items, which is explained in the Editing chapter.

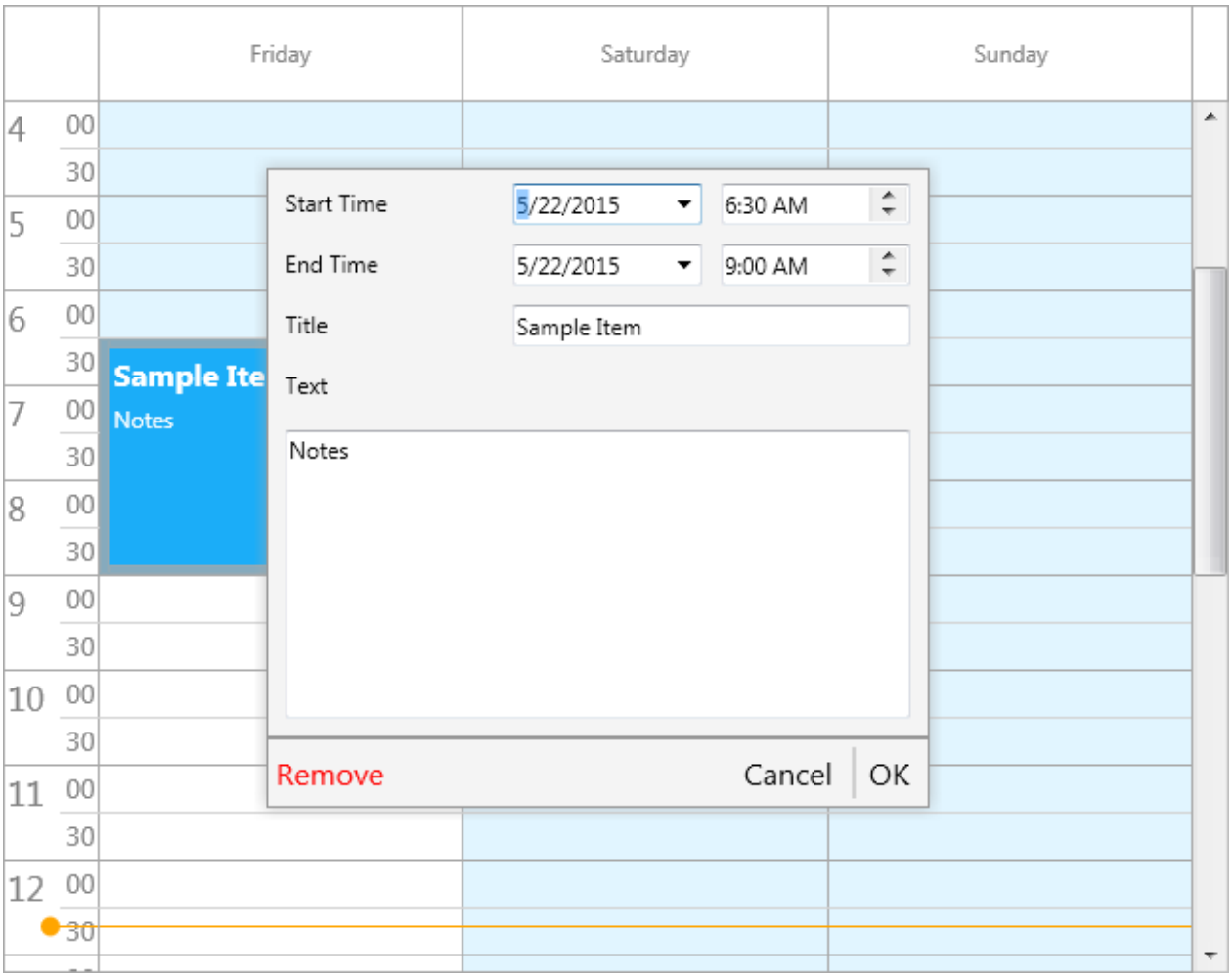
## Editing

Editing can be done in several ways, with the mouse / finger and the keyboard. The properties under Interaction are set to allow editing by default. Clicking or tapping a second time on an active item will start editing. Editing can also be started by pressing F2 or Enter on the keyboard. To stop editing, click next to the item or press F2 again on the item. By default, a built-in TMemo is shown in the area of the item.




The editor can be changed to the built-in editor dialog as we have shown in the previous chapter when insert a new item.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Interaction.UpdateMode := pumDialog;
TMSFMXPlanner1.EndUpdate;
```




For each action, be it sizing, moving, selection or editing an event is triggered. An explanation of each event / property / function and procedure that can be used to further customize the planner and to handle interaction / editing actions can be found under the Properties, Events and Procedures and functions chapters.

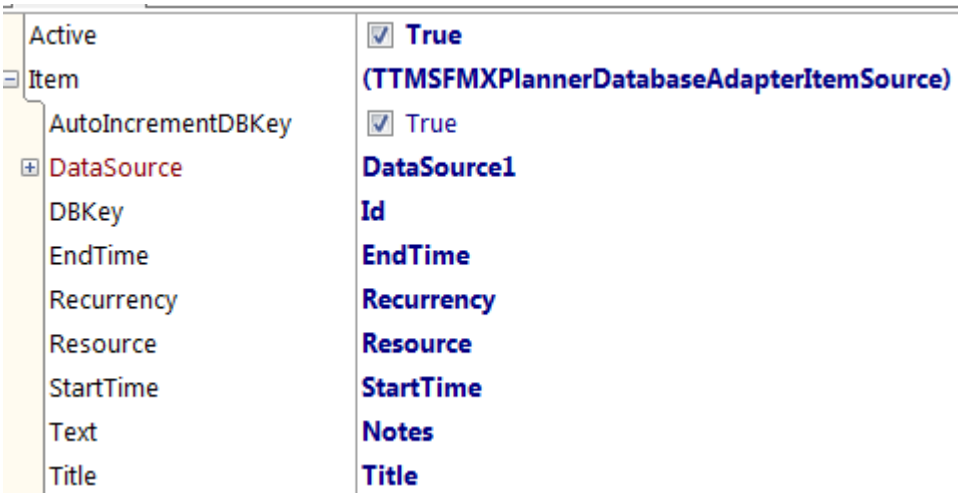
## Databinding

The planner supports databinding through a non-visual component called TTMSFMXPlannerDatabaseAdapter. As with the other adapters (explained in the 'Demos' chapter) it is a simple as connecting the adapter to the planner, filling in the database fields and set the Active property to True.

Below is a screenshot and sample code how binding is done at designtime and at runtime.

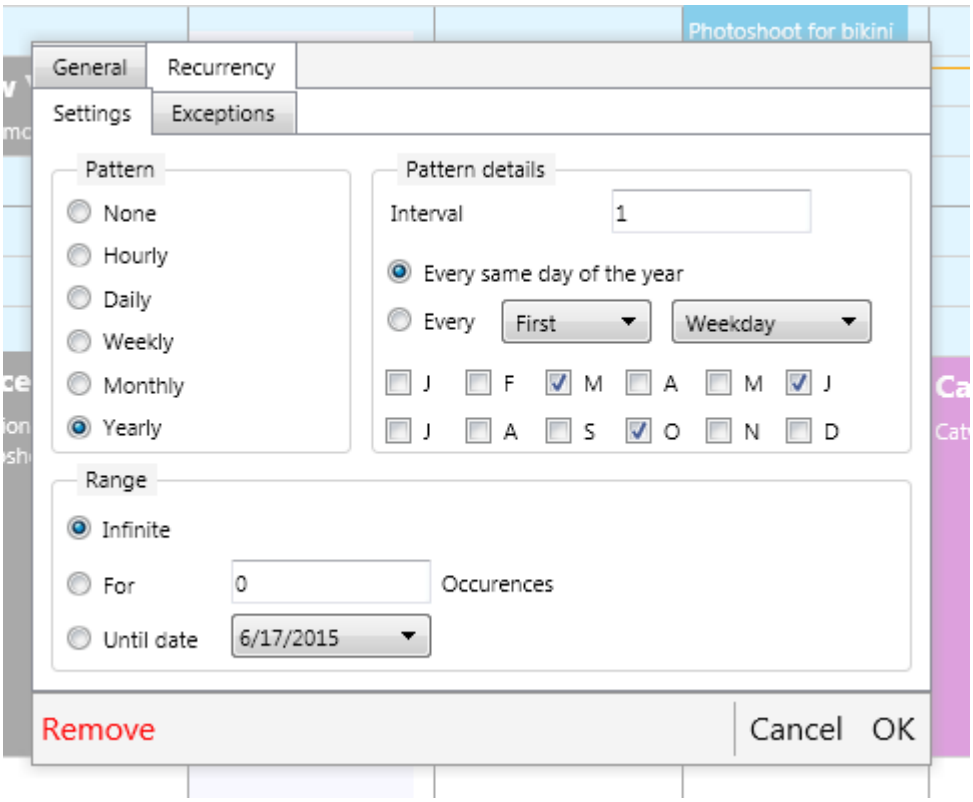
```
TMSFMXPlanner1.Adapter := TMSFMXPlannerDatabaseAdapter1;
```

```
TMSFMXPlannerDatabaseAdapter1.Item.DataSource := DataSource1;
TMSFMXPlannerDatabaseAdapter1.Item.DBKey := 'Id';
TMSFMXPlannerDatabaseAdapter1.Item.StartTime := 'StartTime';
TMSFMXPlannerDatabaseAdapter1.Item.EndTime := 'EndTime';
TMSFMXPlannerDatabaseAdapter1.Item.Title := 'Title';
TMSFMXPlannerDatabaseAdapter1.Item.Text := 'Text';
TMSFMXPlannerDatabaseAdapter1.Item.Resource := 'Resource';
TMSFMXPlannerDatabaseAdapter1.Item.Recurrency := 'Recurrency';
```




After filling the mandatory (DBKey, StartTime and EndTime) and optional fields the planner will automatically load the items from the dataset and display the items that are visible in the current configuration. The items can be re-loaded at any time using TMSFMXPlannerDatabaseAdapter1.LoadItems for instance when the starttime of the planner is changed.

The planner database adapter also supports item recurrency. When adding a new or editing an existing item, the (optional) built-in editor dialog is triggered with a replacement for the content. This replacement is coming from the TTMSFMXPlannerItemEditorRecurrency component that can be connected to the ItemEditor property of the planner. When starting the application after everything is setup, the custom editor appears after editing an item.

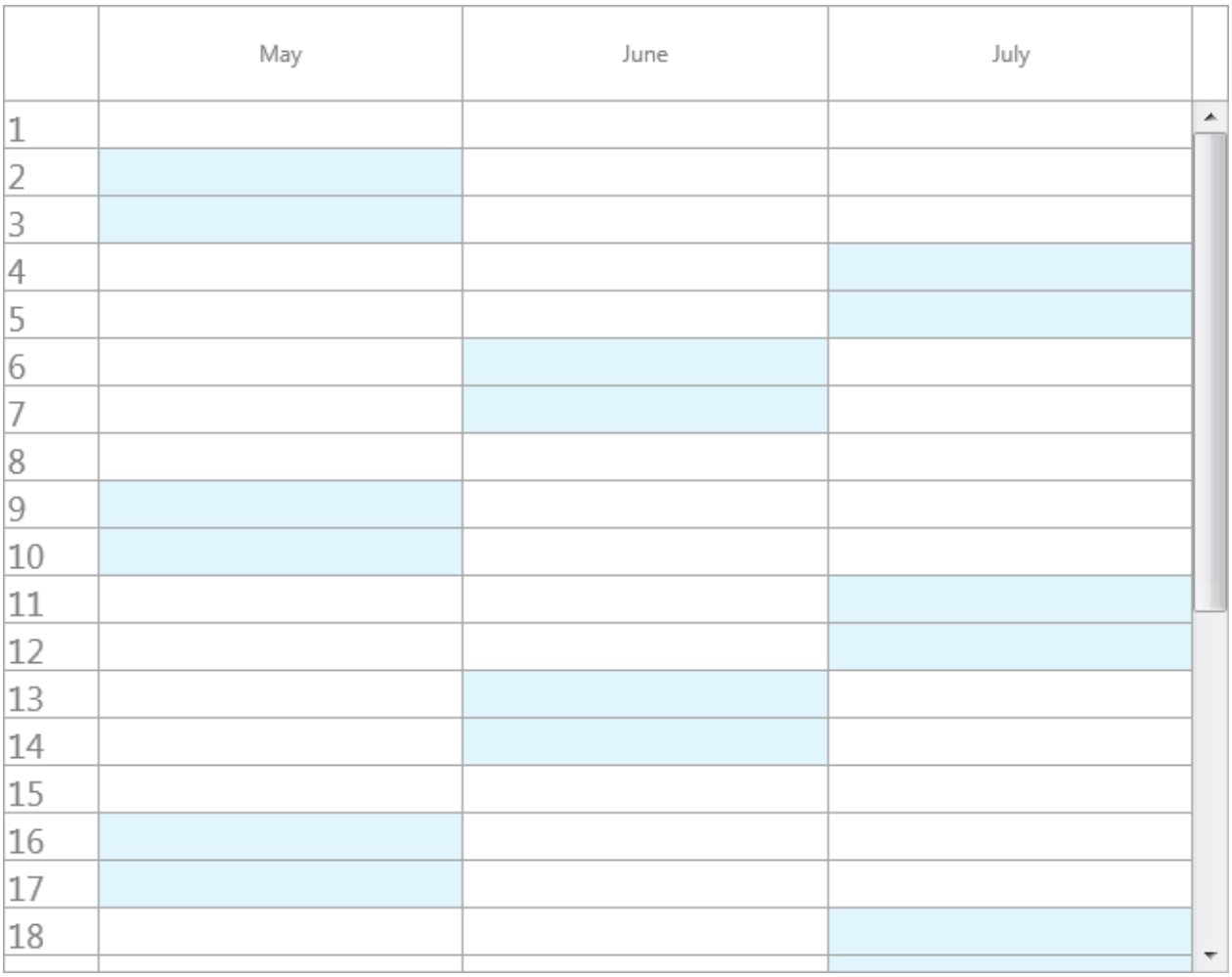



## Customization

The planner is highly customizable. There are a lot of events that you can use for further customization, such as changing the color of a specific timeslot, adding an icon to an item, dynamically changing the formatting of the positions, customize the built-in or inplace editor, etc... Below are some samples that demonstrate these events.

### Setting a specific inactive timeslot

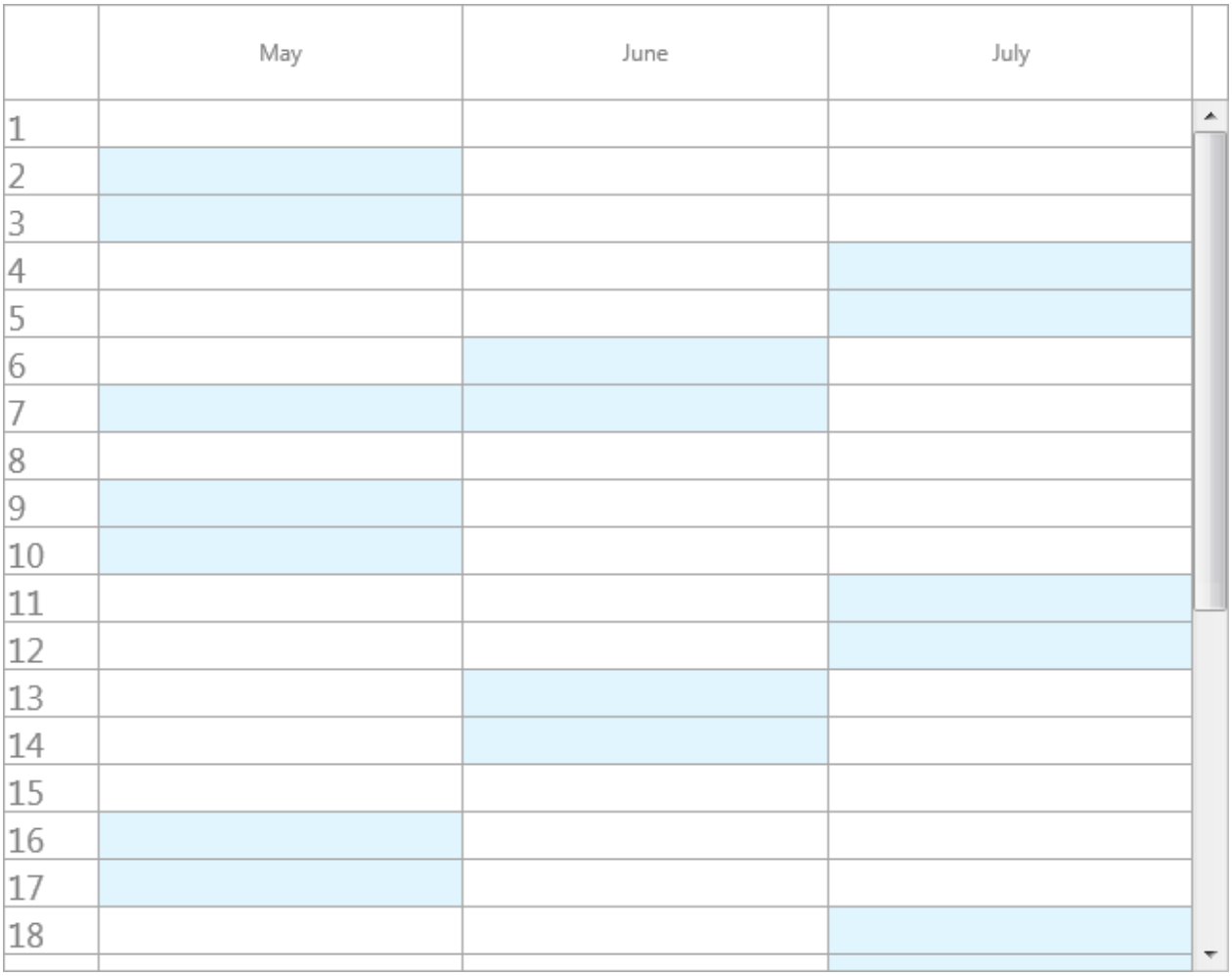
The properties for setting a timeslot inactive is by adding or removing a value in the ModeSettings.InactiveDays set or, in some modes, by changing the TimeLine.ActiveStart and TimeLine.ActiveEnd. This behaviour is shared for all timeslots. What if you want to set an additional timeslot inactive? The OnIsDateTimeInactive event can be used to accomplish this. In this sample, we take the pmMultiMonth mode, which clearly demonstrates the inactive timeslot state.




As you can see in the above screenshot, the inactivedays are set with the ModeSettings.InactiveDays property and by default limited to Saturday and Sunday. If we want to add for example the 7th of May as inactive day, we use the following code:

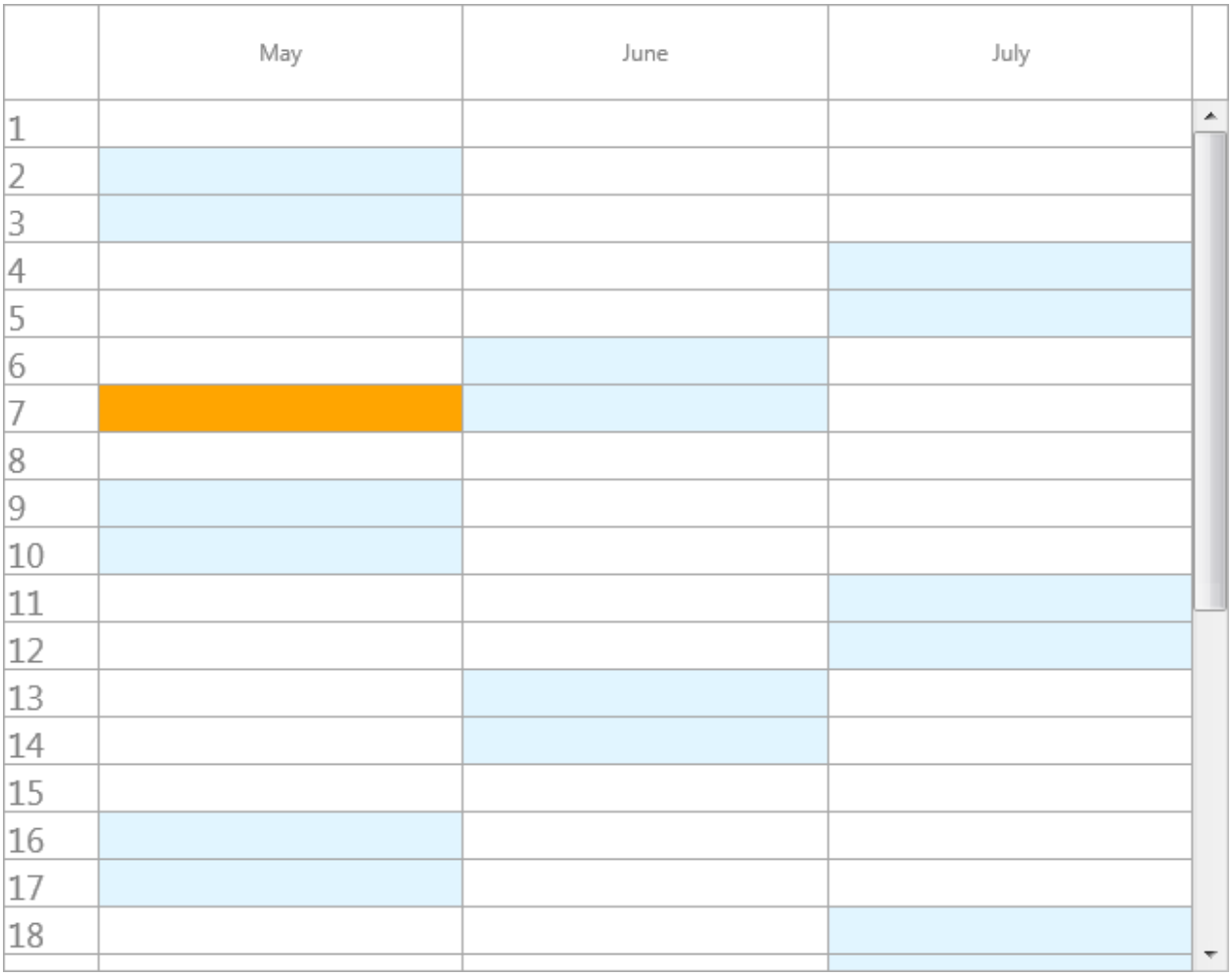
```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.Mode := pmMultiMonth;
TMSFMXPlanner1.Interaction.ShowSelection := False;
TMSFMXPlanner1.EndUpdate;

procedure TForm1.TMSFMXPlanner1IsDateTimeInActive(Sender: TObject;
  ADateTime: TDateTime; APosition: Integer; var AInActive: Boolean);
begin
  AInActive := AInActive or (CompareDate(ADateTime, EncodeDate(2015,
5, 7)) = EqualsValue);
end;
```




As you can see, the 7th of May is set to an inactive state. If we want to change the color for a specific cell we can use the OnBeforeDrawCell event and change the fill settings. If we apply this to the above sample, we can color the 7th of May which is inactive.

```
procedure TForm1.TMSFMXPlanner1BeforeDrawCell(Sender: TObject;
  ACanvas: TCanvas; ARect: TRectF; ACol, ARow: Integer; AStartTime,
  AEndTime: TDateTime; APosition: Integer; AKind:
TTMSFMXPlannerCacheItemKind;
  var AAllow, ADefaultDraw: Boolean);
begin
  if CompareDate(AStartTime, EncodeDate(2015, 5, 7)) = EqualsValue
then
    ACanvas.Fill.Color := claOrange;
end;
```




## Adding an icon to an item based on the conflict state

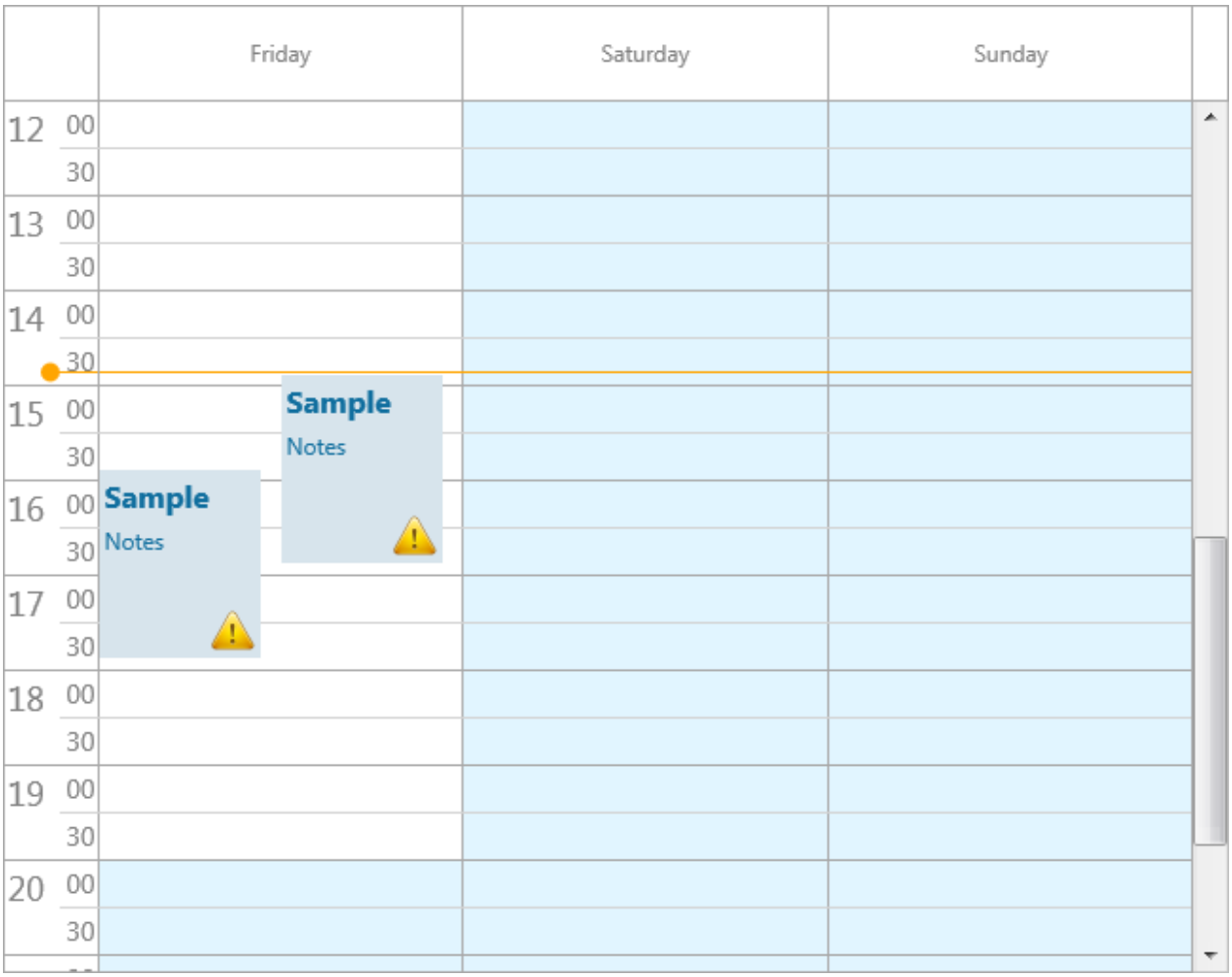
When adding items that overlap, you might need to show a notification that they are overlapping. This sample adds 2 overlapping items and custom draws an additional icon in the lower left corner to indicate the item is in conflict with another item. To do this, we add a TTMSFMXBitmapContainer instance on the form. Below is the code that demonstrates this. Comparing this to the previous sample, we do not want to change the default drawing, but want to add an additional element after the item is drawn. The difference here is that we make use of the OnAfterDrawItem event.

```
TMSFMXPlanner1.BeginUpdate;
TMSFMXPlanner1.items.Clear;
TMSFMXPlanner1.DefaultItem.Title := 'Sample';
```

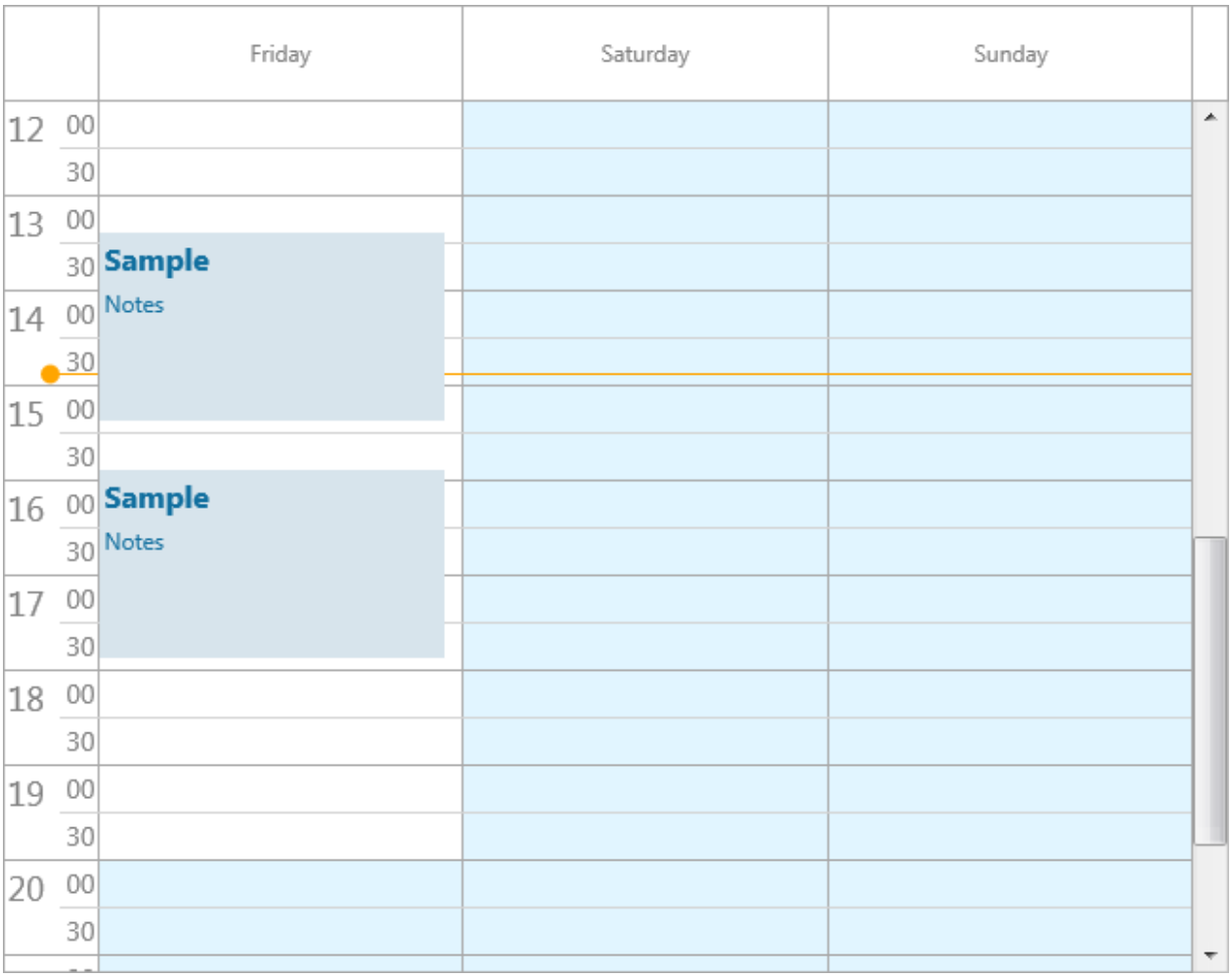
```
TMSFMXPlanner1.DefaultItem.Text  := 'Notes';
it := TMSFMXPlanner1.Items.Add;
it.StartTime := Now + EncodeTime(1, 0, 0, 0);
it.EndTime := Now + EncodeTime(3, 0, 0, 0);

it := TMSFMXPlanner1.Items.Add;
it.StartTime := Now;
it.EndTime := Now + EncodeTime(2, 0, 0, 0);
TMSFMXPlanner1.EndUpdate;

procedure TForm1.TMSFMXPlanner1AfterDrawItem(Sender: TObject; ACanvas:
TCanvas;
  ARect: TRectF; AItem: TTMSFMXPlannerItem);
var
  bmp: TBitmap;
  rbmp: TRectF;
begin
  if AItem.ConflictsForPosition(0) > 1 then
  begin
    bmp := TMSFMXBitmapContainer1.FindBitmap('warning');
    if Assigned(bmp) then
    begin
      rbmp := RectF(ARect.Right - 26, ARect.Bottom - 26, ARect.Right -
2, ARect.Bottom - 2);
      ACanvas.DrawBitmap(bmp, RectF(0, 0, bmp.Width, bmp.Height),
rbmp, 1);
    end;
  end;
end;
```




As soon as we move the item so it is not in conflict with the other item, the icons will disappear.

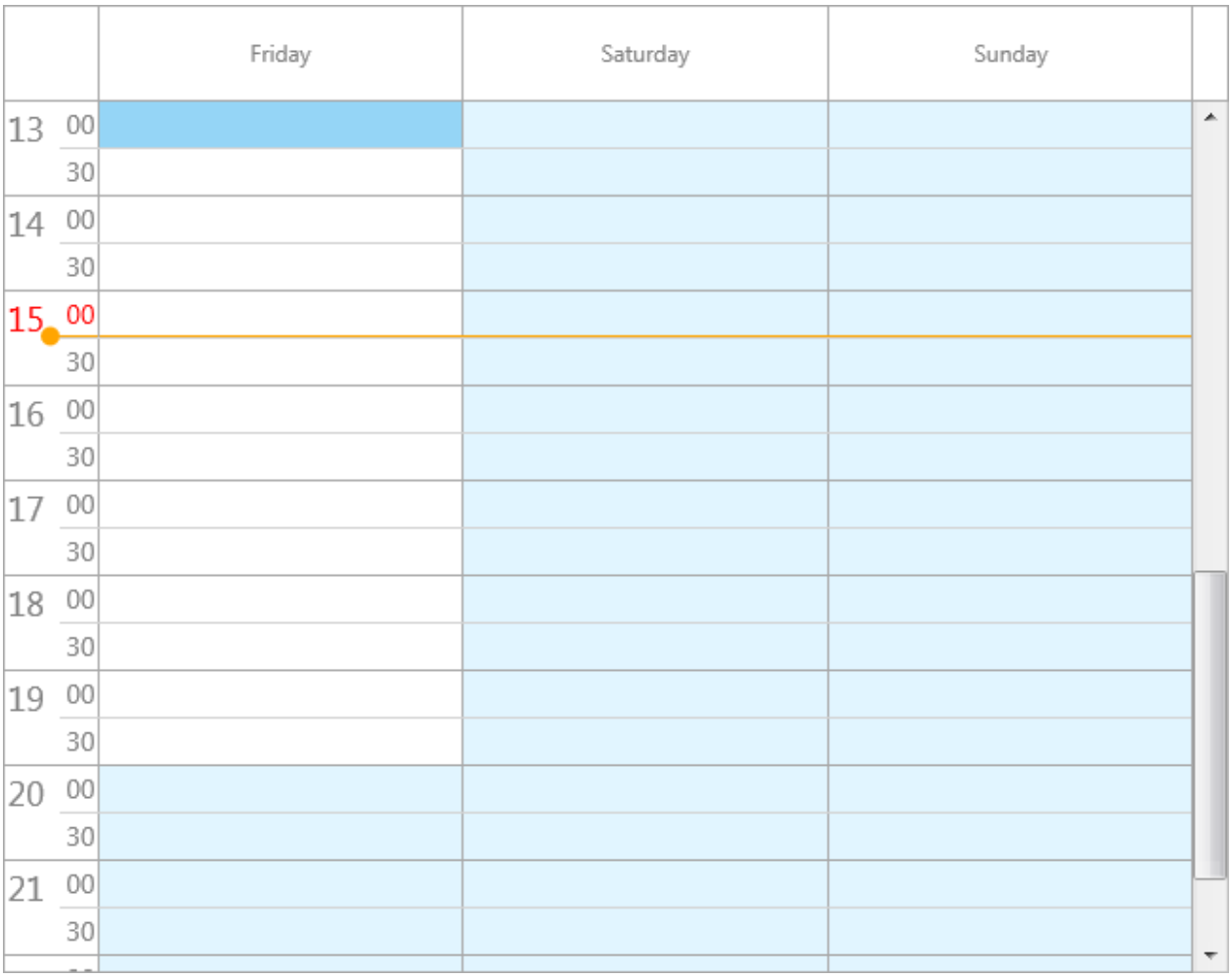



## Changing the color for a specific timeline unit

The appearance of the timeline can be changed with the TimeLineAppearance properties. The appearance of the units and subunits are the same for all timeslots except for the font size. The sample below is demonstrating how to change the font color of the current time timeslot.

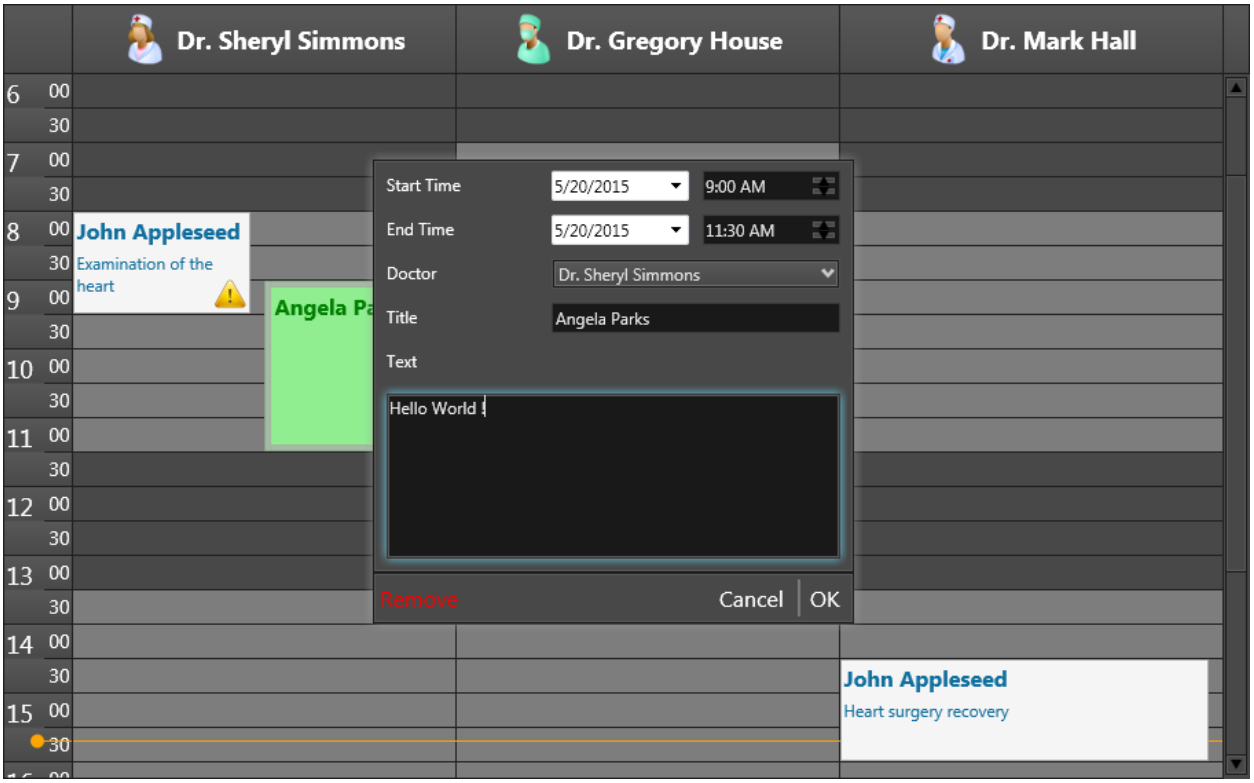
```
procedure TForm1.TMSFMXPlanner1BeforeDrawTimeText(Sender: TObject;
  ACanvas: TCanvas; ARect: TRectF; AValue: Double; ARow: Integer;
  ASubUnit: Boolean; AKind: TTMSFMXPlannerCacheItemKind; AText:
string;
  var AAllow: Boolean);
var
  du: TDateTime;
begin
```

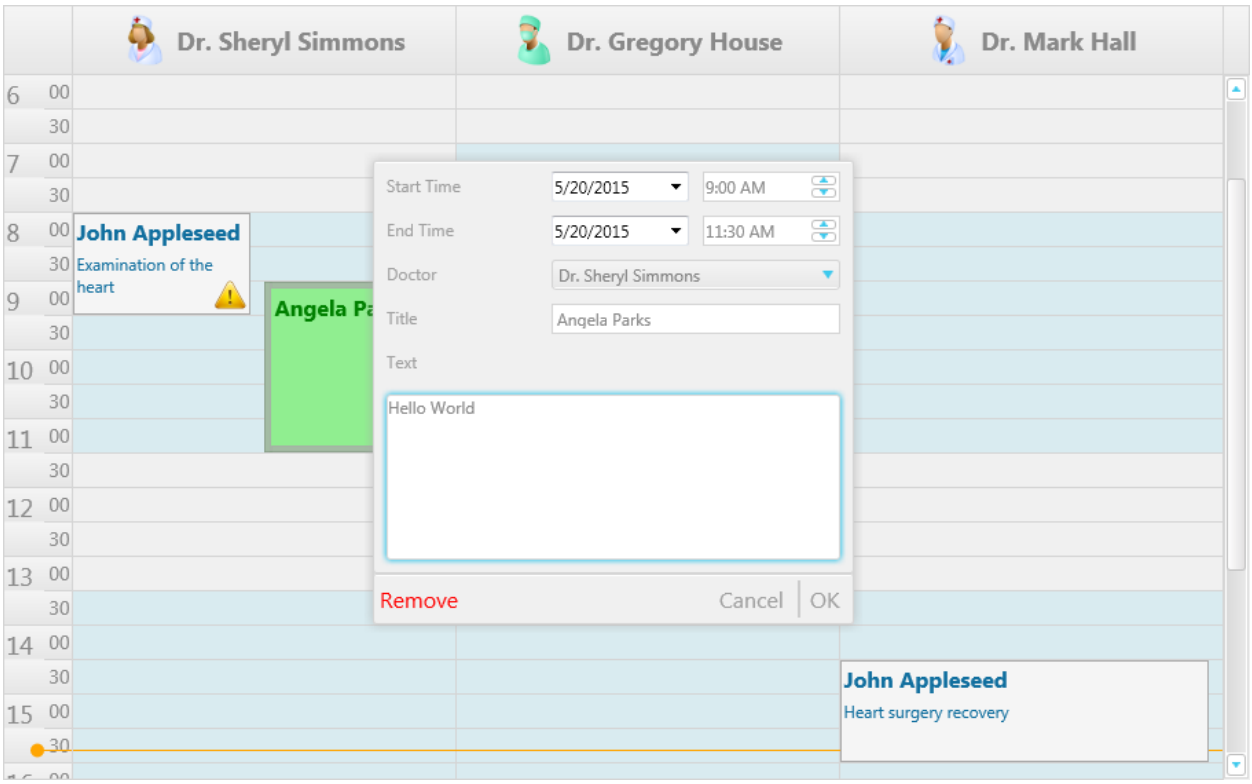
```
  du := EncodeTime(0, 30, 0, 0);
  if (CompareDateTime(AValue, Now) = LessThanValue) and
(CompareDateTime(AValue + du, Now) = GreaterThanValue) then
    ACanvas.Fill.Color := claRed;
end;
```




## Styling

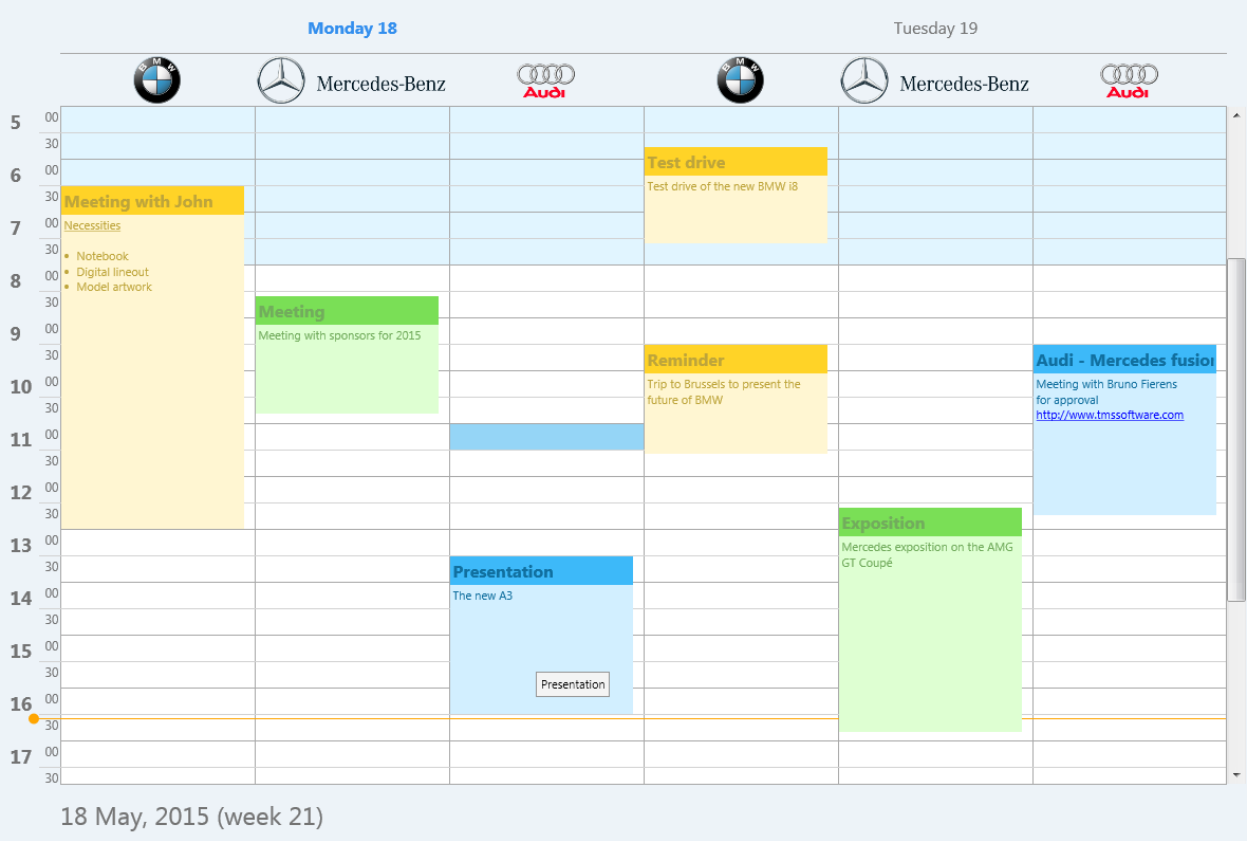
The planner supports FireMonkey Styles. Simply put a StyleBook one the form and load on of the default or premium FireMonkey Styles. After assigning the StyleBook to the form, the AdaptToStyle property will then automatically adapt to the style loaded in the StyleBook. Below are 2 samples of styles that are applied to the planner.


Dr. Sheryl Simmons
Dr. Gregory House
Dr. Mark Hall
John Appleseed
Examination of the heart
Start Time
5/20/2015
9:00 AM
End Time
5/20/2015
11:30 AM
Doctor
Dr. Sheryl Simmons
Title
Angela Parks
Text
Hello World !
Remove
Cancel
OK
John Appleseed
Heart surgery recovery




## Demos

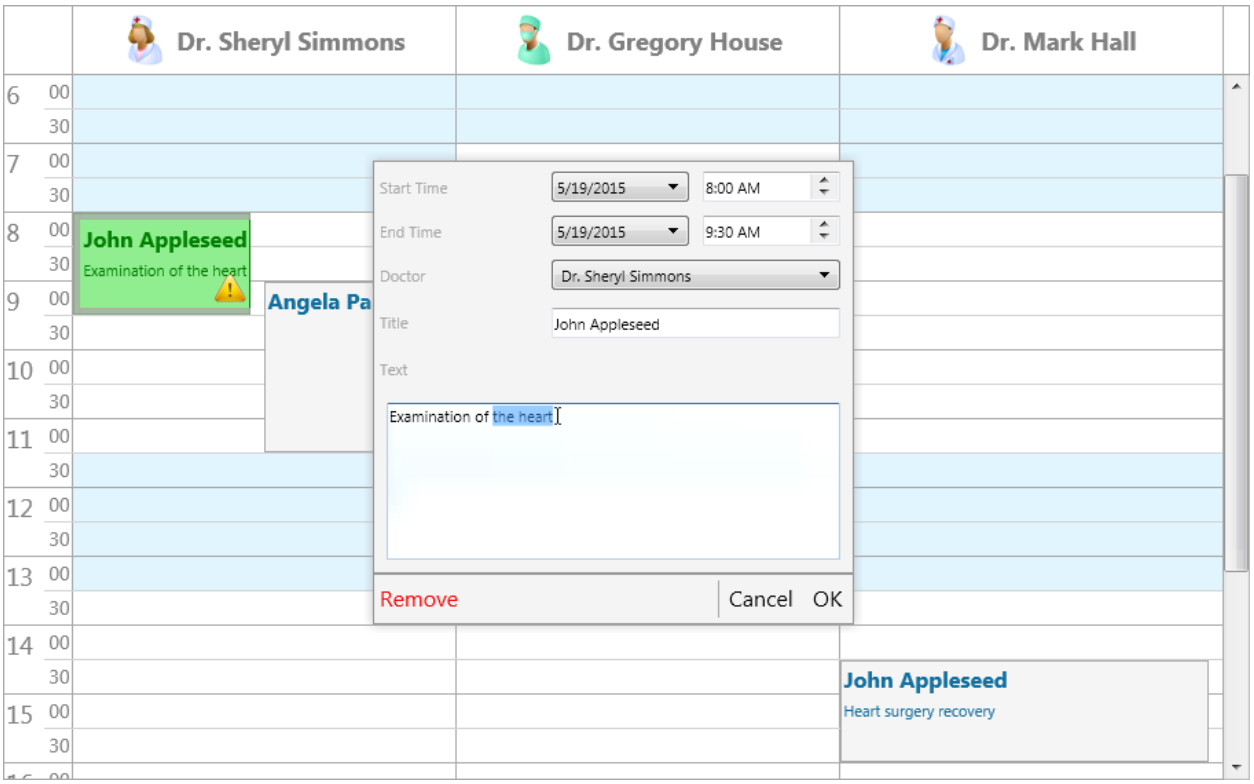
## Overview




The overview demo demonstrates the most important elements of the planner.

- HTML rendering in the positions area and the item text area.
- Changing the appearance of an item.
- Custom drawing
- Styling of the timeline, positions and groups area.
- Navigation through the different modes.

## Editing




The editing demo focuses on built-in inplace and dialog editing.

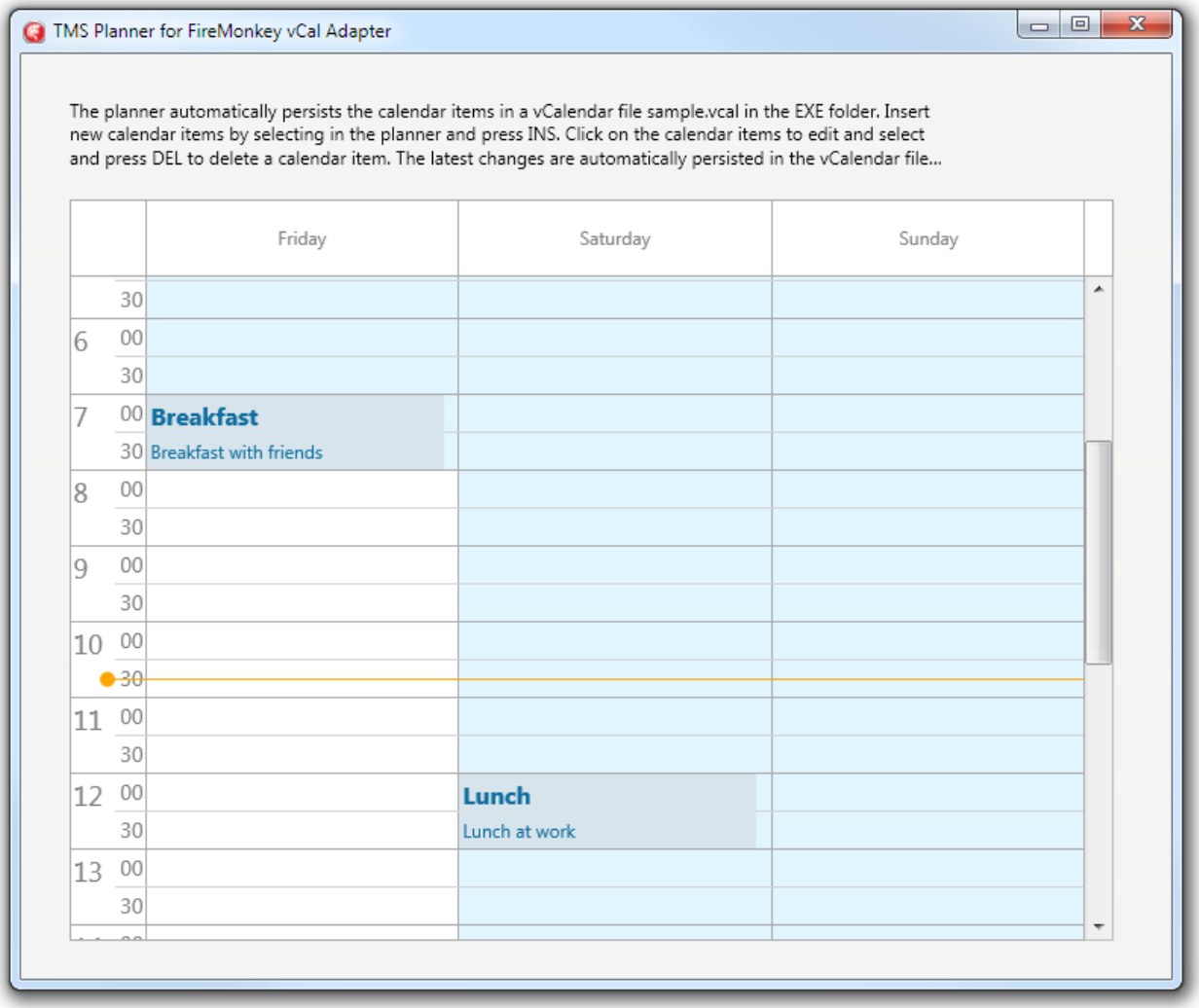
- How to assign a custom content instead of the standard content for the built-in editor dialog for a specific item.
- HTML text in the positions area.
- Custom drawing of items.
- Difference between inplace and dialog editing.
- Movable and sizeable items.

## Custom timeline



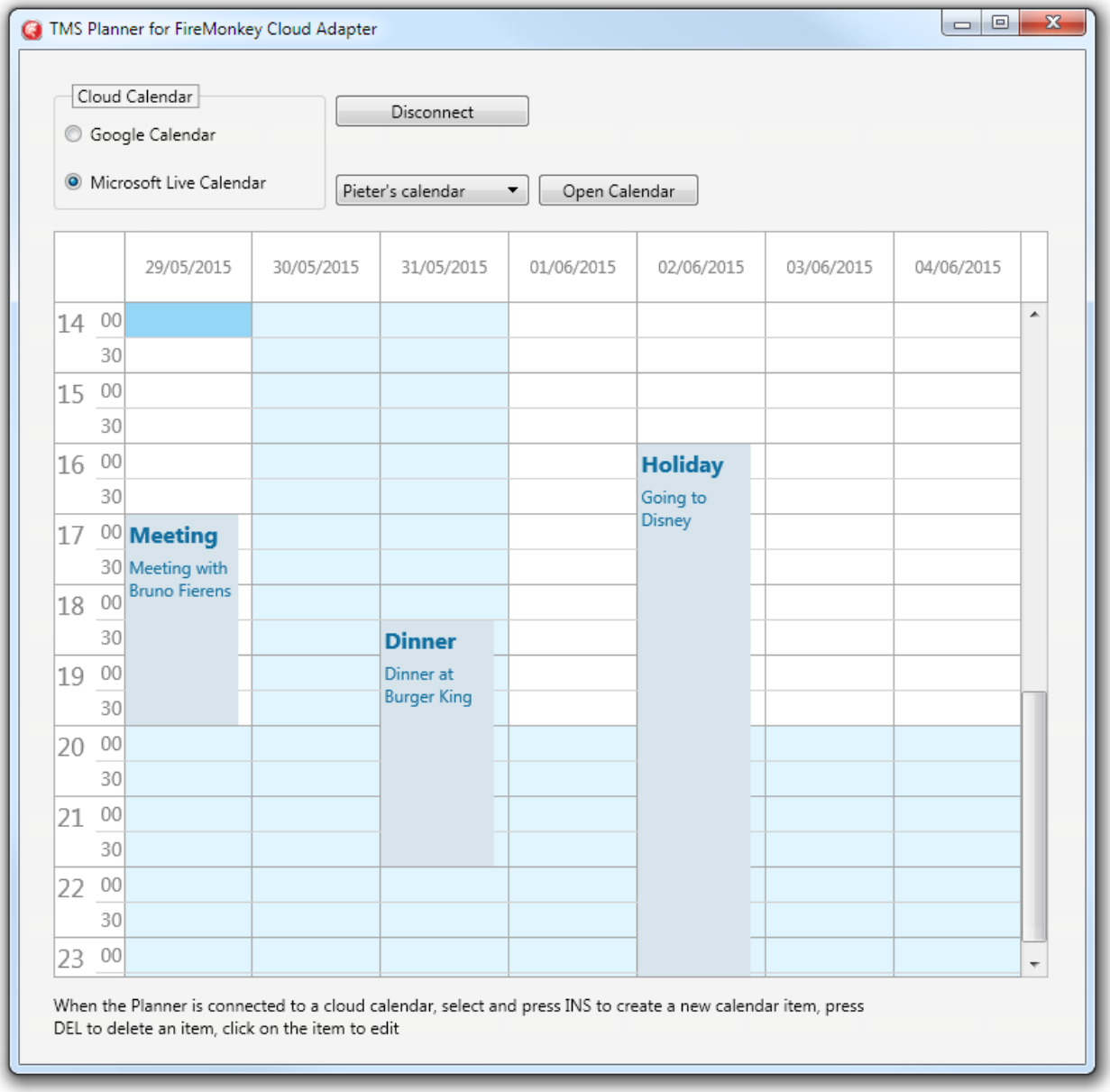

The custom timeline demo shows how to configure the planner in horizontal mode as well as programmatically adding custom datetime values to show a series of items with HTML formatted text.

## vCal adapter




The vCal adapter demo demonstrates how to save and load items from a vCal file. When inserting and updating items, the changes are automatically saved to the vCal file.

## Cloud adapter




The cloud demo demonstrates how to connected to the Google Calendar and Microsoft Live Calendar and display the items. Changing the position, start/end time and text will also automatically update the item through the cloud service. This demo requires the TMS Cloud Pack for FireMonkey and a separate package to install the 2 adapter components. Below are instructions on how to configure this.

Make sure the latest version of TMS Pack for FireMonkey is installed.

Then install TMS Cloud Pack for FireMonkey
(http://www.tmssoftware.com/site/tmsfmxcloudpack.asp)

From the IDE, create a new package and add the files FMX.TMSPlannerGoogleAdapter.pas, FMX.TMSPlannerLiveAdapter.pas, FMX.TMSPlannerAdapterReg.pas from the TMS Pack for FireMonkey source folder. Then compile & install this package in the IDE. The IDE should have automatically added references to the TMS Pack for FireMonkey and TMS Cloud Pack for FireMonkey package files to the requires list. If the IDE did not do this automatically, add TMSFMXPackPkgDXE*.dcp and TMSFMXCloudPackPkgDXE*.dcp to the requires list (* = 5,6,7 or 8 depending on the version of the IDE being used)

When this package is compiled & installed in the IDE, the components TTMSFMXPlannerGoogleAdapter & TTMSFMXPlannerLiveAdapter should be available on the tool palette and the demo can be opened & used.

Registering the application for Google Calendar and/or Microsoft Live Calendar
---
Follow the steps in the TMS Cloud Pack for FireMonkey documentation to register for an application key & secret for Google Calendar and/or Microsoft Live Calendar and set this key in the file APPIDS.INC in the demo source folder:

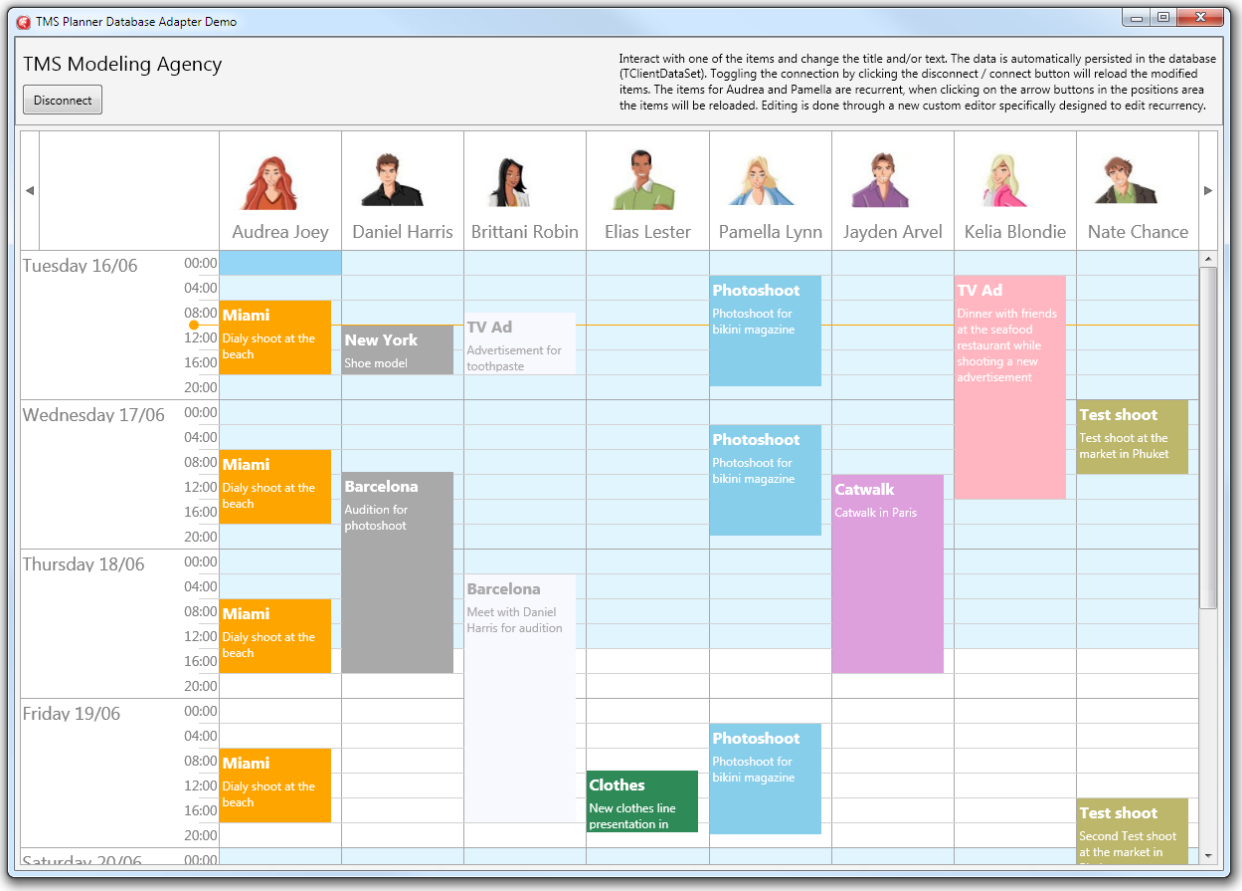
```
//please specify the keys here

const
  GAppkey = ;
  GAppSecret = ;

  LiveAppkey = ;
  LiveAppSecret = ;
```

With these IDs correctly specified, the demo is ready to run and connect to a cloud calendar.

## Database adapter




The database adapter demo shows how to bind data through the TTMSFMXPlannerDatabaseAdapter component to an instance of the TTMSFMXPlanner. It also shows to edit the recurrency of an item through a custom editor specifically designed to edit recurrency.

## Properties

| | |
|---|---|
| ActiveItem | Gets or Sets the Active Item. |
| Adapter | Property to connect to an instance of TTMSFMXPlannerAdapter. |
| AdaptToStyle | When set to true, and an FMX style is applied, the planner takes over the style. When set to false, the planner applies the default style. |
| BitmapContainer | A container that contains a set of images to be used in combination with HTML drawing. |
| CustomDateTimes | Property to add datetime values when the mode is set to pmCustom. The values that are added will automatically be sorted. |
| DefaultItem | The default item that is applied when creating a new item in the planner either programmatically or at runtime through the planner interaction.<br><br>These properties also apply to all items that are created, which are not repeated in this table for the items collection. |
| DefaultItem -> ActiveColor | The color of the item when the item in active state. |
| DefaultItem -> ActiveFontColor | The font color of the text of the item in active state. |
| DefaultItem -> ActiveTitleColor | The color of the title of the item in active state. |
| DefaultItem -> ActiveTitleFontColor | The font color of the title of the item in active state. |
| DefaultItem -> Color | The color of the item in normal state |
| DefaultItem -> Deletable | Determines if an item can be deleted and if delete indicator is showing when ItemsAppearance.ShowDeleteArea is true. |

| | |
|---|---|
| DefaultItem -> DisabledColor | The color of the item in disabled state. |
| DefaultItem -> DisabledFontColor | The font color of the text of the item in disabled state. |
| DefaultItem -> DisabledTitleColor | The color of the title of the item in disabled state. |
| DefaultItem -> DisabledTitleFontColor | The font color of the title of the item in disabled state. |
| DefaultItem -> Editable | Sets whether an item is editable. |
| DefaultItem -> Enabled | Sets whether an item is enabled or disabled. |
| DefaultItem -> EndTime | The date / time the item ends |
| DefaultItem -> FixedResource | Sets whether an item has a fixed resource. |
| DefaultItem -> FontColor | The font color of the item in normal state. |
| DefaultItem -> Hint | The hint of the item. |
| DefaultItem -> Moveable | Sets whether an item is moveable. |
| DefaultItem -> Overlappable | Sets whether an item is overlappable. |
| DefaultItem -> Resource | The resource of the item; |
| DefaultItem -> Selectable | Sets whether an item is selectable. |
| DefaultItem -> SelectedColor | The color of the item in selected state but not active state. |
| DefaultItem -> SelectedFontColor | The font color of the item in selected but not active state. |
| DefaultItem -> SelectedTitleColor | The color of the title of the item in selected but not active state. |
| DefaultItem -> SelectedTitleFontColor | The font color of the title of the item in selected but not active state. |
| DefaultItem -> ShowDelete | Shows the delete icon if the item is deletable. |
| DefaultItem -> ShowTitle | Shows or hides the title area and title text. |
| DefaultItem -> Sizeable | Sets whether an item is sizeable. |
| DefaultItem -> StartTime | The date / time the item starts. |
| DefaultItem -> Text | The text of the item. |
| DefaultItem -> Title | The title of the item. |
| DefaultItem -> TitleColor | The color of the title in normal state. |
| DefaultItem -> TitleFontColor | The font color of the title in normal state. |

| | |
|---|---|
| DefaultItem -> Visible | Sets whether an item is visible or invisible. |
| GridCellAppearance | The appearance of the cells (timeslots) in the planner. |
| GridCellAppearance -> DisabledFill | The fill of a cell in disabled state, which can be set automatically (in multi month mode) and/or programmatically, with the OnIsDateTimeDisabled event. |
| GridCellAppearance -> Fill | The fill of a cell in normal state. |
| GridCellAppearance -> HorizontalStroke | The horizontal stroke of the cell. |
| GridCellAppearance -> InActiveFill | The fill of a cell in inactive state, which can be set with the properties ModeSettings -> InActiveDays and the TimeLine.ActiveStart and ActiveEnd and/or through the OnIsDateTimeInActive event. |
| GridCellAppearance -> SubHorizontalStroke | The horizontal stroke of the cell for the sub units. The sub units are automatically calculated based on the display settings under the TimeLine property. |
| GridCellAppearance -> VerticalStroke | The vertical stroke of the cell. |
| Groups | A collection of custom groups that are added on top or at the bottom relative to the positions. In multi resource / multi day mode the groups are replaced by days or resources but the appearance of the groups is applied. |
| GroupsAppearance | The appearance of the groups. |
| GroupsAppearance -> BottomFill | The fill of the bottom groups. |
| GroupsAppearance -> BottomFont | The font of the bottom groups. |
| GroupsAppearance -> BottomFontColor | The font color of the bottom groups. |
| GroupsAppearance -> BottomHorizontalTextAlign | The horizontal text align of the bottom groups. |
| GroupsAppearance -> BottomSize | The size of the bottom groups. |
| GroupsAppearance -> BottomStroke | The stroke of the bottom groups. |
| GroupsAppearance -> BottomVerticalTextAlign | The vertical text align of the bottom groups. |

| | |
|---|---|
| GroupsAppearance -> Layouts | Shows the groups on top and/or bottom. This can be the left and/or right side in horizontal mode. |
| GroupsAppearance -> TopFill | The fill of the top groups. |
| GroupsAppearance -> TopFont | The font of the top groups. |
| GroupsAppearance -> TopFontColor | The font color of the top groups. |
| GroupsAppearance -> TopHorizontalTextAlign | The horizontal text align of the top groups. |
| GroupsAppearance -> TopSize | The size of the top groups. |
| GroupsAppearance -> TopStroke | The stroke of the top groups. |
| GroupsAppearance -> TopVerticalTextAlign | The vertical text align of the top groups. |
| HorizontalScrollBarVisible | Sets whether the horizontal scrollbar is visible or not. |
| Interaction | The interaction options of the planner. |
| Interaction -> BottomNavigationButtons | The bottom navigation buttons used to navigate to the next or previous start time of the planner. |
| Interaction -> InPlaceEditorMode | The mode of the inplace editor when the UpdateMode property is set to use an inplace editor. The mode of the inplace editor can be set to edit the title, the text or the item. In the item mode, the text is edited, but the inplace editor takes on the dimensions of the item instead of the text. |
| Interaction -> KeepSelection | Determines whether the selection is removed or retained after selecting an item. |
| Interaction -> KeyboardDelete | Enables keyboard deletion of the active item. |
| Interaction -> KeyboardEdit | Enables keyboard editing of the active item. |
| Interaction -> KeyboardInsertMode | Sets the keyboard insert mode. After selection, the insert key can be used to insert an item. Additionally the mode can be set to first show the built-in editor dialog before inserting the item. |

| | |
|---|---|
| Interaction -> MouseEditMode | Sets the mouse edit mode. Editing is started after a single or a double click on the item. Additionally, the mode can be configured to first select an item before editing can be started with a single click. |
| Interaction -> MouseInsertMode | Sets the mouse insert mode. After selection the item is automatically inserted. Additionally the mode can be set to first show the built-in editor dialog before inserting the item. |
| Interaction -> MoveMode | The move mode of the item. Defaults to automatically determine the mode. The mode on mobile operating systems is to tap and hold on the item area to move the item. The mode on desktop operating systems is to use the mouse and click on the move area at the edge of the item to move the item. |
| Interaction -> MultiSelect | Allows multiple item selection. Multiple items can be selected, but only one item can be active and selected simultaneously. |
| Interaction -> ReadOnly | When set to true, disables updating inserting and editing of items. Selection, scrolling and navigation is still possible. |
| Interaction -> ShowSelection | Shows or hides selection. |
| Interaction -> SizeMode | The move mode of the item. Defaults to automatically determine the mode. The mode on mobile operating systems is to use the size handlers at the outside of the item area to move the item. The mode on desktop operating systems is to use the mouse and click on the size area at the edge of the item to move the item. |
| Interaction -> SwipeToNextDateTime | Activates the possibility to swipe on the positions area to navigate to the next start time. |

| | |
|---|---|
| Interaction -> SwipeToPreviousDateTime | Activates the possibility to swipe on the positions area to navigate to the previous start time. |
| Interaction -> TopNavigationButtons | The top navigation buttons used to navigate to the next or previous start time of the planner. |
| Interaction -> TouchScrolling | Enables or disables touch scrolling. Touch scrolling can be used to navigate through the planner on all areas except for the positions area. Enabled by default, but on desktop system it might be preferable to set to false. |
| Interaction -> UpdateMode | The mode to update the item. When editing with the keyboard or the mouse, the update mode determines whether an inplace editor is shown, or the built-in editor dialog. |
| ItemEditor | Property to connect to an instance of TTMSFMXPlannerItemEditor for replacing the built-in editor dialog content with custom content. |
| Items | The items collection, the explanation of the properties of the item can be found under the DefaultItem property in this table. |
| ItemsAppearance | The general appearance of the item. |
| ItemsAppearance -> ActiveFill | The fill of the item in active state. |
| ItemsAppearance -> ActiveFont | The font of the item in active state. |
| ItemsAppearance -> ActiveStroke | The stroke of the item in active state. |
| ItemsAppearance -> ActiveTitleFill | The fill of the title of the item in active state. |
| ItemsAppearance -> ActiveTitleFont | The font of the title of the item in active state. |
| ItemsAppearance -> ActiveTitleStroke | The stroke of the title of the item in active state. |
| ItemsAppearance -> DeleteAreaColor | The color of the delete icon in normal state. |
| ItemsAppearance -> DeleteAreaSize | The size of the delete area. |
| ItemsAppearance -> DisabledFill | The fill of the item in disabled state. |

| | |
|---|---|
| ItemsAppearance -> DisabledFont | The font of the item in disabled state. |
| ItemsAppearance -> DisabledStroke | The stroke of the item in disabled state. |
| ItemsAppearance -> DisabledTitleFill | The fill of the title of the item in disabled state. |
| ItemsAppearance -> DisabledTitleFont | The font of the title of the item in disabled state. |
| ItemsAppearance -> DisabledTitleStroke | The stroke of the title of the item in disabled state. |
| ItemsAppearance -> Fill | The fill of the item in normal state. |
| ItemsAppearance -> Font | The font of the item in normal state. |
| ItemsAppearance -> Gap | The gap of the item used to allow selection next to the item. |
| ItemsAppearance -> MoveAreaColor | The color of the move area of the item in desktop interaction mode. |
| ItemsAppearance -> MoveAreaSize | The size of the move area of the item in desktop interaction mode. |
| ItemsAppearance -> SelectedFill | The fill of the item in selected state. |
| ItemsAppearance -> SelectedFont | The font of the item in selected state. |
| ItemsAppearance -> SelectedStroke | The stroke of the item in selected state. |
| ItemsAppearance -> SelectedTitleFill | The fill of the title of the item in selected state. |
| ItemsAppearance -> SelectedTitleFont | The font of the title of the item in selected state. |
| ItemsAppearance -> SelectedTitleStroke | The stroke of the title of the item in selected state. |
| ItemsAppearance -> ShowDeleteArea | Shows a delete icon in the top right corner of the item if the item deletable property is true. |
| ItemsAppearance -> ShowItemhelpers | Shows helpers on the item when interacting with the item. |
| ItemsAppearance -> ShowMoveArea | Show the move area on the item. |
| ItemsAppearance -> ShowSizeArea | Show the size area on the item. |
| ItemsAppearance -> SizeAreaColor | The color of the size area. |
| ItemsAppearance -> SizeAreaSize | The size of the size area. |
| ItemsAppearance -> Stroke | The stroke of the item. |
| ItemsAppearance -> TextHorizontalTextAlign | The horizontal text align of the item. |
| ItemsAppearance -> TextVerticalTextAlign | The vertical text align of the item. |
| ItemsAppearance -> TitleFill | The fill of the title of the item. |

| | |
|---|---|
| ItemsAppearance -> TitleFont | The font of the title of the item. |
| ItemsAppearance -> TitleHorizontalTextAlign | The horizontal text align of the title. |
| ItemsAppearance -> TitleStroke | The stroke of the title. |
| ItemsAppearance -> TitleVerticalTextAlign | The vertical text align of the title. |
| ModeSettings | The initial settings to configure the planner. |
| ModeSettings -> EndTime | The end time in case day period or half day period view is used. |
| ModeSettings -> InActiveDays | The days that are drawn with the inactive fill. |
| ModeSettings -> OverlappableItems | A general setting to allow / disallow overlappable items. |
| ModeSettings -> StartTime | The start time for all the views except for the custom view. |
| OrientationMode | The orientation of the planner. The default mode is vertical. In horizontal mode the planner automatically rotates text and applies the opposite settings from vertical mode where necessary. |
| Positions | The positions in the planner. |
| Positions -> Count | The count of positions in the planner. The positions are used in all views and can be combined with resources |
| Positions -> Format | The format of the positions when days / months are displayed. The positions are automatically converted to datetime values in views that combine multi days / months. When this property value is an empty string, default datetime formatting is applied depending on the chosen view. |
| Positions -> ViewStart | The initial position that is shown when starting the application. |
| PositionsAppearance | The appearance of the positions. |
| PositionsAppearance -> BottomFill | The fill of the bottom positions. |
| PositionsAppearance -> BottomFont | The font of the bottom positions. |
| PositionsAppearance -> BottomFontColor | The font color of the bottom positions. |
| PositionsAppearance -> BottomHorizontalTextAlign | The horizontal text align of the |

| | |
|---|---|
| | bottom positions. |
| PositionsAppearance -> BottomSize | The size of the bottom positions. |
| PositionsAppearance -> BottomStroke | The stroke of the bottom positions. |
| PositionsAppearance -> BottomVerticalTextAlign | The vertical text align of the bottom positions. |
| PositionsAppearance -> Layouts | Shows the positions on top and/or bottom. This can be the left and/or right side in horizontal mode. |
| PositionsAppearance -> Size | The size of a single position when no stretching is turned off. |
| PositionsAppearance -> Stretch | Applies automatic stretching on the positions. |
| PositionsAppearance -> TopFill | The fill of the top positions. |
| PositionsAppearance -> TopFont | The font of the top positions. |
| PositionsAppearance -> TopFontColor | The font color of the top positions. |
| PositionsAppearance -> TopHorizontalTextAlign | The horizontal text align of the top positions. |
| PositionsAppearance -> TopSize | The size of the top positions. |
| PositionsAppearance -> TopStroke | The stroke of the top positions. |
| PositionsAppearance -> TopVerticalTextAlign | The vertical text align of the top positions. |
| Resources | A collection combined with items in views that support resources. Each resource has a text and a name property to uniquely identify each resource. |
| SelectedItems | Returns a list of selected items when the planner is configured for multi-select. This list also includes the active item. |
| Selection | Read-only property to retrieve the current selection cell range. The selection can set with the method TMSFMXPlanner.SelectCells(AStartCell, AEndCell: TTMSFMXPlannerCell); |
| SelectionAppearance | The Appearance of the selection. |
| TimeLine | The settings of the timeline. |
| TimeLine -> ActiveEnd | The active end time. The time values that exceed the end time are drawn in the inactive state. |

| | |
|---|---|
| TimeLine -> ActiveStart | The active start time. The time values that are prior to the active start time are drawn in inactive state. |
| TimeLine -> CurrentTimeMode | The mode of the current time indicator, which is drawn in the timeline and on the grid depending on the mode. |
| TimeLine -> DisplayEnd | The actual display end time based on the ModeSettings.StartTime, DisplayUnit and DisplayUnitType properties. |
| TimeLine -> DisplayOffset | The offset applied to the display start and end time. |
| TimeLine -> DisplayOffsetType | The display offset type. |
| TimeLine -> DisplayStart | The actual display start time based on the ModeSettings.StartTime, DisplayUnit and DisplayUnitType properties. |
| TimeLine -> DisplaySubUnitFormat | The format for the sub units that are displayed in the timeline. |
| TimeLine -> DisplayUnit | The timeline unit used to indicate a time slot. Used in combination with the DisplayUnitType property. |
| TimeLine -> DisplayUnitFormat | The format for the units that are display in the timeline. |
| TimeLine -> DisplayUnitSize | The size of the time slots. |
| TimeLine -> DisplayUnitType | The unit type of the display. |
| TimeLine -> ViewStart | The initial start time that is shown when starting the application. |
| TimeLineAppearance | The appearance of the timeline. |
| TimeLineAppearance -> CurrentTimeColor | The color of the current time indication. |
| TimeLineAppearance -> Layouts | Shows the timeline at the left and/or the right side. This can be the top and/or bottom side in horizontal mode. |
| TimeLineAppearance -> LeftFill | The fill of the left timeline. |
| TimeLineAppearance -> LeftFont | The font of the left timeline. |
| TimeLineAppearance -> LeftFontColor | The font color of the left timeline. |
| TimeLineAppearance -> LeftHorizontalTextAlign | The horizontal text align of the left |

| | |
|---|---|
| | timeline. |
| TimeLineAppearance -> LeftSize | The size of the left timeline. |
| TimeLineAppearance -> LeftStroke | The stroke of the left timeline. |
| TimeLineAppearance -> LeftSubHorizontalTextAlign | The horizontal text align of the left timeline for sub units. |
| TimeLineAppearance -> LeftSubUnitFontSize | The font size of the left timeline for sub units. |
| TimeLineAppearance -> LeftSubVerticalTextAlign | The vertical text align for the left timeline for sub units. |
| TimeLineAppearance -> LeftSubVerticalTextMode | The vertical text mode for the left timeline for the sub units. |
| TimeLineAppearance -> LeftVerticalTextAlign | The vertical text align for the left timeline. |
| TimeLineAppearance -> LeftVerticalTextMode | The vertical text mode for the left timeline. |
| TimeLineAppearance -> RightFill | The fill of the right timeline. |
| TimeLineAppearance -> RightFont | The font of the right timeline. |
| TimeLineAppearance -> RightFontColor | The font color of the left timeline. |
| TimeLineAppearance -> RightHorizontalTextAlign | The horizontal text align of the right timeline. |
| TimeLineAppearance -> RightSize | The size of the right timeline. |
| TimeLineAppearance -> RightStroke | The stroke of the right timeline. |
| TimeLineAppearance -> RightSubHorizontalTextAlign | The horizontal text align of the right timeline for sub units. |
| TimeLineAppearance -> RightSubUnitFontSize | The font size of the left timeline for sub units. |
| TimeLineAppearance -> RightSubVerticalTextAlign | The vertical text align for the right timeline for sub units. |
| TimeLineAppearance -> RightSubVerticalTextMode | The vertical text mode for the right timeline for the sub units. |
| TimeLineAppearance -> RightVerticalTextAlign | The vertical text align for the right timeline. |
| TimeLineAppearance -> RightVerticalTextMode | The vertical text mode for the right timeline. |
| TimeLineAppearance -> Stretch | Stretches the timeline. False by default, and uses the TimeLine.DisplayUnitSize for a single timeslot. |
| VerticalScrollBarVisible | Sets whether the vertical scrollbar is visible or not. |

| | |
|---|---|
| ViewCol / ViewRow | Properties to initialize the first visible Column and Row. Can be used in combination with TimeLine.ViewStart and Positions.ViewStart. |

## Events

| | |
|---|---|
| OnAfterDeleteItem | Event called after an item is deleted. |
| OnAfterDrawBottomNavigationButton | Event called after a bottom navigation button is drawn. |
| OnAfterDrawCell | Event called after the cell is drawn. |
| OnAfterDrawCellHorizontalLine | Event called after the horizontal line in a cell is drawn. |
| OnAfterDrawCellVerticalLine | Event called after the vertical line in a cell is drawn. |
| OnAfterDrawCurrentTimeInGrid | Event called after the current time indication is drawn in the grid. |
| OnAfterDrawCurrentTimeInTimeLine | Event called after the current time indication is drawn in the timeline. |
| OnAfterDrawDeleteArea | Event called after the delete area in desktop mode is drawn. |
| OnAfterDrawGroup | Event called after a group is drawn. |
| OnAfterDrawGroupEmptySpace | Event called after the empty space next to the group area is drawn. |
| OnAfterDrawGroupText | Event called after the group text is drawn. |
| OnAfterDrawItem | Event called after an item is drawn. |
| OnAfterDrawItemHelper | Event called after an item helper is drawn. |
| OnAfterDrawItemHelperText | Event called after an item helper text is drawn. |
| OnAfterDrawItemText | Event called after an item text is drawn. |
| OnAfterDrawItemTitle | Event called after an item title is drawn. |
| OnAfterDrawItemTitleText | Event called after an item title text is drawn. |
| OnAfterDrawMoveArea | Event called after the move area in desktop mode is drawn. |
| OnAfterDrawPosition | Event called after a position is drawn. |
| OnAfterDrawPositionEmptySpace | Event called after an empty space next to the position area is drawn. |
| OnAfterDrawPositionText | Event called after a position text is drawn. |
| OnAfterDrawSizeArea | Event called after the size area in desktop mode is drawn. |
| OnAfterDrawTime | Event called after a time is drawn. |
| OnAfterDrawTimeStroke | Event called after a time stroke is drawn. |
| OnAfterDrawTimeText | Event called after a time text is drawn. |
| OnAfterDrawTopNavigationButton | Event called after a top navigation button is drawn. |
| OnAfterInsertItem | Event called after an item is inserted. |
| OnAfterMoveItem | Event called after an item is moved. |

| | |
|---|---|
| OnAfterNavigateToDateTime | Event called after the planner is navigated to a new datetime through the navigation buttons or through the swipe gesture. |
| OnAfterOpenInplaceEditor | Event called after the inplace editor is opened. |
| OnAfterOpenInsertDialog | Event called after the built-in editor dialog is shown in insert mode for a new item. |
| OnAfterOpenUpdateDialog | Event called after the built-in editor dialog is shown in update mode for an existing item. |
| OnAfterSelectItem | Event called after an item is selected. |
| OnAfterSizeItem | Event called after an item is sized. |
| OnAfterUpdateItem | Event called after an item is updated through the in place editor or built-in editor dialog. |
| OnBeforeDeleteItem | Event called before an item is deleted. |
| OnBeforeDrawBottomNavigationButton | Event called before a bottom navigation button is drawn. |
| OnBeforeDrawCell | Event called before the cell is drawn. |
| OnBeforeDrawCellHorizontalLine | Event called before the horizontal line in a cell is drawn. |
| OnBeforeDrawCellVerticalLine | Event called before the vertical line in a cell is drawn. |
| OnBeforeDrawCurrentTimeInGrid | Event called before the current time indication is drawn in the grid. |
| OnBeforeDrawCurrentTimeInTimeLine | Event called before the current time indication is drawn in the timeline. |
| OnBeforeDrawDeleteArea | Event called before the delete area in desktop mode is drawn. |
| OnBeforeDrawGroup | Event called before a group is drawn. |
| OnBeforeDrawGroupEmptySpace | Event called before the empty space next to the group area is drawn. |
| OnBeforeDrawGroupText | Event called before the group text is drawn. |
| OnBeforeDrawItem | Event called before an item is drawn. |
| OnBeforeDrawItemHelper | Event called before an item helper is drawn. |
| OnBeforeDrawItemHelperText | Event called before an item helper text is drawn. |
| OnBeforeDrawItemText | Event called before an item text is drawn. |
| OnBeforeDrawItemTitle | Event called before an item title is drawn. |
| OnBeforeDrawItemTitleText | Event called before an item title text is drawn. |
| OnBeforeDrawMoveArea | Event called before the move area in desktop mode is drawn. |
| OnBeforeDrawPosition | Event called before a position is drawn. |
| OnBeforeDrawPositionEmptySpace | Event called before an empty space next to the position area is drawn. |
| OnBeforeDrawPositionText | Event called before a position text is drawn. |

| | |
|---|---|
| OnBeforeDrawSizeArea | Event called before the size area in desktop mode is drawn. |
| OnBeforeDrawTime | Event called before a time is drawn. |
| OnBeforeDrawTimeStroke | Event called before a time stroke is drawn. |
| OnBeforeDrawTimeText | Event called before a time text is drawn. |
| OnBeforeDrawTopNavigationButton | Event called before a top navigation button is drawn. |
| OnBeforeInsertItem | Event called before an item is inserted. |
| OnBeforeMoveItem | Event called before an item is moved. |
| OnBeforeNavigateToDateTime | Event called before the planner is navigated to a new datetime through the navigation buttons or through the swipe gesture. |
| OnBeforeOpenInplaceEditor | Event called before the inplace editor is opened. |
| OnBeforeOpenInsertDialog | Event called before the built-in editor dialog is shown in insert mode for a new item. |
| OnBeforeOpenUpdateDialog | Event called before the built-in editor dialog is shown in update mode for an existing item. |
| OnBeforeSelectItem | Event called before an item is selected. |
| OnBeforeSizeItem | Event called before an item is sized. |
| OnBeforeUpdateItem | Event called before an item is updated through the in place editor or built-in editor dialog. |
| OnCloseInplaceEditor | Event called when the inplace editor is closed. |
| OnCloseInsertDialog | Event called when the built-in editor is closed after inserting a new item. |
| OnCloseUpdateDialog | Event called when the built-in editor is closed after updating an existing item. |
| OnCustomPanelToItem | Event called when the editor dialog is closed and the contents will be transferred to the item. |
| OnGetCustomContentPanel | Event called when the editor dialog is created and asks for the content panel for a particular item. |
| OnGetGroupText | Event called when the group text is retrieved. |
| OnGetInplaceEditor | Event called before the inplace editor is created, to customize the built-in editor for each item. |
| OnGetItemHelperText | Event called when the helper text for an item is retrieved. |
| OnGetItemText | Event called when the item text is retrieved. |
| OnGetItemTitleText | Event called when the item title text is retrieved. |
| OnGetPositionText | Event called when the position text is retrieved. |

| | |
|---|---|
| OnGetTimeText | Event called when the time text is retrieved. |
| OnHasDateTimeSub | Event called to determine if the current mode supports drawing of sub datetime values. |
| OnHScroll | Event called when the planner scrolls horizontally. |
| OnIsDateTimeDisabled | Event called to retrieve which datetime value is inactive. |
| OnIsDateTimeInActive | Event called when an anchor is clicked inside an item text. |
| OnIsDateTimeSub | Event called to determine if a datetime value is a sub value or not. |
| OnIsItemDeletable | Event called to determine if an item is deletable. |
| OnItemAnchorClick | Event called when a time slot is being selected. |
| OnItemChanged | Event called when the item is updated after moving, sizing and editing. |
| OnItemToCustomPanel | Event called when the editor dialog is being opened and the data of the item will be transferred to the content panel. |
| OnSelectCell | Event called when the selected cell range has changed |
| OnSelectingCell | Event called when the selected cell range is changing. |
| OnSelectingTime | Event called when a time slot is selected. |
| OnSelectTime | Event called when the planner is scrolled vertically |
| OnVScroll | Event called when the planner scrolls vertically. |
| StretchScrollBars (public) | When true: stretches the scrollbars to the total height / width of the planner. When false (default): the scrollbars are limited to the grid area. |

## Procedures and functions

### Planner

| | |
|---|---|
| AddOrUpdateItem(…): TTMSFMXPlannerItem + overloads | Adds a new or updates an existing item with the parameters passed through this function. Returns the item that has been created or updated. |
| ApplyDefaultStyle | Applies the default style to the planner. |
| CancelEditing | Stops editing the active item and cancels the changes. |
| CellToDateTime(ACell: | Converts the cell to a datetime value. |

| | |
|---|---|
| TTMSFMXPlannerCell): TDateTime | |
| CellToEndDateTime(ACell: TTMSFMXPlannerCell): TDateTime | Converts the cell to an end datetime value. |
| CellToStartDateTime(ACell: TTMSFMXPlannerCell): TDateTime | Converts the cell to a start datetime value. |
| CloseEditingDialog(ACancel: Boolean) | Closes the editing dialog and commits or cancels the changes made to the active item. |
| CloseEditingDialogAndRemoveItem | Closes the editing dialog when the dialog is active and removes the item. This action is triggered from the Remove button in the lower left corner of the dialog. |
| DateTimeToCell(ADateTime: TDateTime; AEndDateTime: Boolean = False): TTMSFMXPlannerCell | Converts the datetime value to a cell. |
| DateTimeToPosition(ADateTime: TDateTime; AEndDateTime: Boolean = False; ACheckBounds: Boolean = True): Integer | Converts a datetime value to a position. Additional parameters can be passed to limit the value within the scrollable area or to get the value as an end datetime instead of a start date time. |
| DateTimeToValue(ADateTime: TDateTime; AEndDateTime: Boolean = False; ACheckBounds: Boolean = True): Double | Converts the datetime value to an x (horizontal orientation) or y (vertical orientation) pixel value. Additional parameters can be passed to limit the value within the scrollable area or to get the value as an end datetime instead of a start date time. |
| EditItem(AItem: TTMSFMXPlannerItem) | Start editing an item. Depending on the properties, inplace editing or dialog editing will be started. |
| FindFirstItem(AStartTime, AEndTime: TDateTime; APosition: Integer): TTMSFMXPlannerItem | Returns the first item with a specific start time, end time and position. |
| FindGroupByName(AName: String): TTMSFMXPlannerGroup | Returns a group with a specific name. |
| FindGroupIndexByName(AName: String): Integer | Returns a group index with a specific name. |
| FindItemWithDBKey(ADBKey: String): TTMSFMXPlannerItem | Returns the item with a specific DBKey property. |
| FindNextItem(AStartTime, AEndTime: TDateTime; APosition: Integer): TTMSFMXPlannerItem | Returns the next item with a specific start time, end time and position based on the results of the FindFirstItem. |
| FindResourceByName(AName: String): TTMSFMXPlannerResource | Returns a resource with a specific name. |
| FindResourceIndexByName(AName: String): Integer | Returns a resource index with a specific name. |
| GetEditingDialog(AItemIndex: Integer = -1): TTMSFMXPlannerEditingDialog | Returns the editing dialog for further customization, optionally based on the item |

| | |
|---|---|
| | index. |
| GetEndTimeSizeHandler: TTMSFMXPlannerSizeHandler | Returns the end time size handler for further customization in case mobile sizing is used. |
| GetHintPopup: TTMSFMXPlannerHintPopup | Returns the item hint popup for further customization. |
| GetInplaceEditor: TTMSFMXPlannerInplaceEditor | Returns the inplace editor for further customization. |
| GetStartTimeSizeHandler: TTMSFMXPlannerSizeHandler | Returns the start time size handler for further customization in case mobile sizing is used. |
| HasItem(AStartTime, AEndTime: TDateTime; APosition: Integer; ACompareWithItemIndex: Integer = -1; ACheckOverlap: Boolean = True): Boolean | Returns a Boolean if the planner has another item within a specific position. Additional parameters can be used to compare with a specific item and check if an item overlaps without checking the ModeSettings.OverlappableItems property or the Overlappable property per item. |
| InitSample | Initializes a sample with 3 resources and 1 item. |
| IsCellDisabled(ACell: TTMSFMXPlannerCell): Boolean | Returns a Boolean if the cell is disabled. This converts the cell to a datetime and uses the same approach as the IsDateTimeDisabled function. |
| IsCellInActive(ACell: TTMSFMXPlannerCell): Boolean | Returns a Boolean if the cell is inactive. This converts the cell to a datetime and uses the same approach as the IsDateTimeInactive function. |
| IsDateTimeDisabled(ADateTime: TDateTime; APosition: Integer = -1): Boolean | Returns a Boolean if the datetime that is passed as a parameter is disabled. The disabled state is determined automatically and can be overridden in the OnIsDateTimeDisabled event. |
| IsDateTimeInActive(ADateTime: TDateTime; APosition: Integer = -1): Boolean | Returns a Boolean if the datetime that is passed as a paremeter is inactive. The inactive state is determined through the ModeSettings.InActiveDays and the TimeLine.ActiveStart and TimeLine.ActiveEnd properties. The state can be overridden by the OnIsDateTimeInActive event. |
| IsEditing: Boolean | Returns a Boolean to indicate the planner is in edit mode. |
| IsValidItem(AItem: TTMSFMXPlannerItem): Boolean | Returns a Boolean whether the item is valid or not. A valid item is an item that lies within the display start time and display end time and within the defined resources. |
| ItemToEndCell(AItem: TTMSFMXPlannerItem): | Returns the end cell of the item. |

| | |
|---|---|
| TTMSFMXPlannerCell | |
| ItemToStartCell(AItem: TTMSFMXPlannerItem): TTMSFMXPlannerCell; | Returns the start cell of the item. |
| Navigate(ACell: TTMSFMXPlannerCell; AForceScroll: Boolean = False) | Navigate to a specific cell in range and optionally force the actual scrolling position to the cell. |
| NavigateToNextDateTime | Navigates to the next datetime depending on the mode. |
| NavigateToPreviousDateTime | Navigates to the previous datetime depending on the mode. |
| OpenEditingDialog(AStartTime, AEndTime: TDateTime; AResource: Integer, ATitle, AText: String; AUpdateItem: Integer = -1) | Opens the editing dialog programmatically with a set of initialization parameters and the ability to start as an insert dialog or an update dialog with the AUpdateItem parameter. |
| PositionToDateTime(APosition: Integer): TDateTime | Converts a position to a datetime value. |
| PositionToResource(APosition: Integer): Integer | Returns the resource for a specific position. |
| ResourceToPosition(AResource: Integer): Integer | Returns the position for a specific resource. |
| RestoreScrollPosition | Restores the previous vertical and horizontal scroll position. Needs to be combined with SaveScrollPosition. |
| SaveScrollPosition | Saves the current vertical and horizontal scroll position. Needs to be combined with RestoreScrollPosition. |
| SelectCells(AStartCell, AEndCell: TTMSFMXPlannerCell) | Select and navigate to a range of cells. |
| SelectedEndDateTime | Returns the datetime of the selected end cell. |
| SelectedResource | Returns the resource of the selected cell. |
| SelectedStartDateTime | Returns the datetime of the selected start cell. |
| SelectItem(AItem: TTMSFMXPlannerItem) | Selects a specific item and makes it active. |
| SelectItem(AItemIndex: Integer) | Selects a specific item through the item index and makes it active. |
| SelectItems(AItems: TTMSFMXPlannerItemArray) | Selects a range of items. |
| SelectNextItem: TTMSFMXPlannerItem | Selects the next item. |
| SelectPreviousItem: TTMSFMXPlannerItem | Selects the previous item. |
| StopEditing | Stops editing the active item and commits the changes. |
| ValueToDateTime(AValue: Double; APosition: Integer = -1): TDateTime; | Converts an x (horizontal orientation) or y (vertical orientation) pixel value to a datetime |

| | |
|---|---|
| | value. |
| XYToCacheItem(X, Y: Double): TTMSFMXPlannerCacheItem | Returns the cached item at X and Y. An Item can consist of multiple rectangles (if the item stretches over multiple columns due to the time difference between start time and end time). Each rectangle represents a cache. |
| XYToCell(X, Y: Double): TTMSFMXPlannerCell | Returns the cell at X and Y. |
| XYToCell(X, Y: Double): TTMSFMXPlannerCell | Returns a cell at X and Y. |
| XYToItem(X, Y: Double): TTMSFMXPlannerItem | Returns the item at X and Y regardless of how many rectangles are drawn. |
| XYToItemAnchor(AItem: TTMSFMXPlannerItem; X, Y: Single) | Returns the anchor at X and Y for a specific item. |
| XYToTime(X, Y: Double): TTMSFMXPlannerTime | Returns the time at X and Y. |

**Item**

| | |
|---|---|
| ConflictsForPosition(APosition: Integer): Integer | Returns the count of conflicts for a specific position. The item can be stretched over multiple positions depending on the mode. |
| ConflictsPosForPosition(APosition: Integer): Integer | Returns the conflict position for a specific position. The item can be stretched over multiple positions depending on the mode. |
| GetFirstRect: TRectF | The first rectangle of the item. |
| GetLastRect: TRectF | The last rectangle of the item. |
| GetRect(AIndex: Integer = -1): TRectF | Returns the rectangle of the item, if multiple rectangles are present due to stretching of the item over multiple positions, the AIndex parameter can be used to retrieve the rectangle of choice. The index of the rectangle lies within the count of rectangles retrieved with the RectCount function. |
| RectCount: Integer | The count of rectangles of an item. |

## General FireMonkey component usage guidlines

With the new FireMonkey framework, the methodology to create and use components has dramatically changed. A component now exists of 2 parts.

### Visual part

The visual part is stored in a .style file, which is compiled to a .res file through an .rc file. The .rc file is included in the package and must be recompiled whenever a change is made to

the .style file. For each component in this set you will find a .style file. In this file, the default layout of the component is stored.

You will notice different elements, basic elements such as an arc, ellipse, rectangle … The elements combine and define the layout of a control. The basic elements are called shapes, and are already available by default. The TMS Instrumentation Workshop for FireMonkey takes this new way of styling one step further: custom shapes. In several components you will find custom shapes registered and useable in a new application, and used in the component by default.

Each shape or element can have a StyleName, which is used in the non-visual part of the control for interaction. This name is key in the relationship or "style-contract" between style resource and component code.

## Non-visual part

The non-visual part of the component interacts with the shapes defined in the .style file. This is a normal .pas unit file as was used for VCL component, yet little to no painting is done in code. As explained above, the visual part is already defined by the style.

The component defined in this unit needs to inherit from the TStyledControl class, which can be styled at designtime. This is the base class for all styleable controls, just like the TCustomControl class was the base class for most controls in the VCL framework.

## Naming convention

It is always good practice to handle a consistent naming convention, therefore all .rc, .pas files and .style files should start with the FireMonkey unit scope name "FMX.", such as the units: FMX.Types, FMX.Dialogs, FMX.Objects …

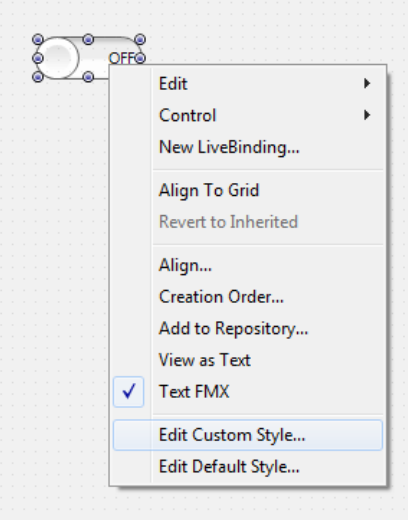
Inside the style file each element can have a StyleName, which can be used in the non-visual part to address the resource. Make sure each element has a unique StyleName to avoid mistakes when interacting with the component. All combinations of elements must be encapsulated within a rectangle element that is invisible by default (through the Fill.Kind and Stroke.Kind = bkNone), and has the StyleName of the component.

If you have a component named TFMXMyFirstControl, the the StyleName of the rectangle encapsulating all other elements must be set to FMXMyFirstControlStyle. The "T" is removed and "Style" is added.

## Styling

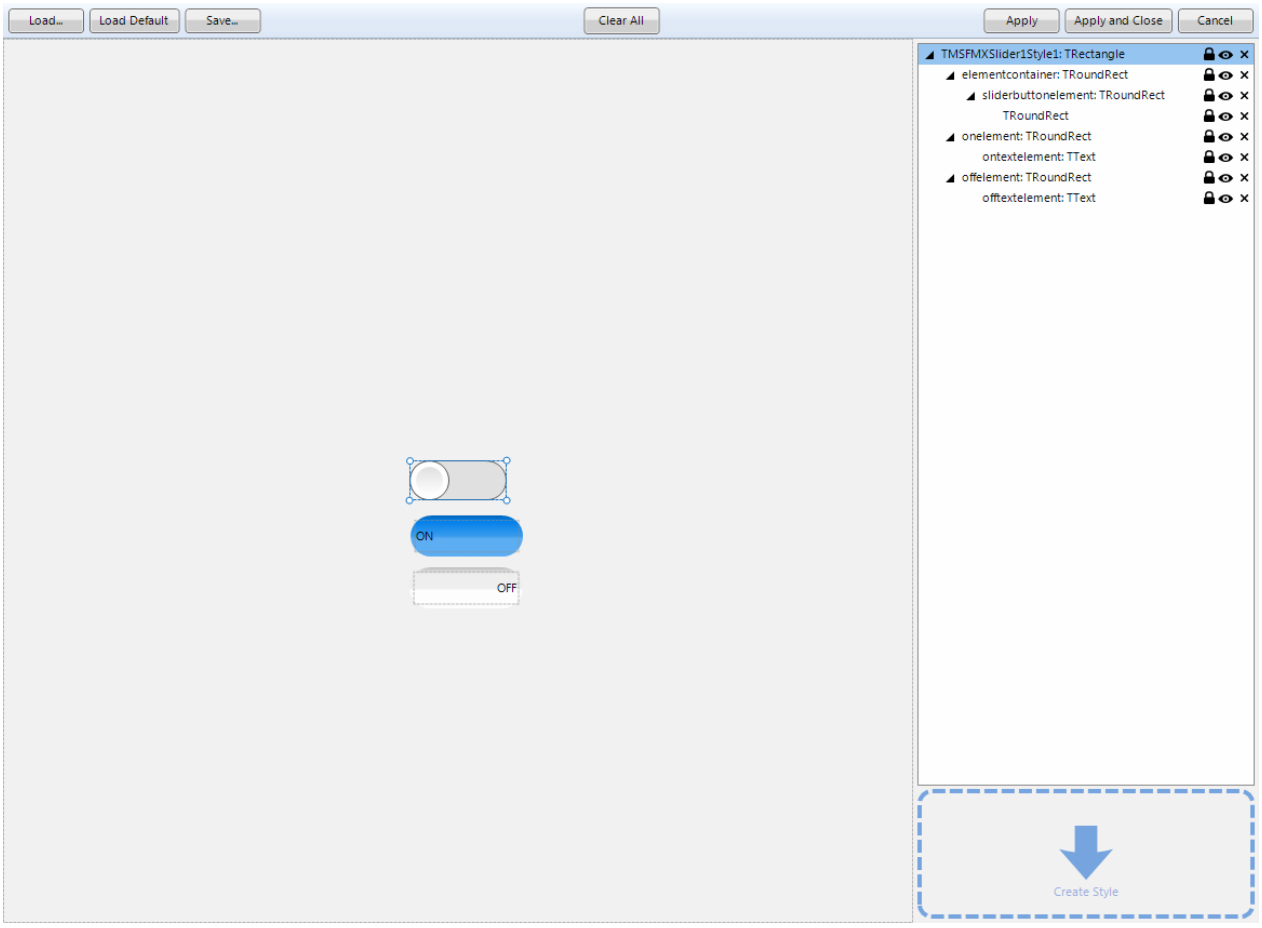
Each component inherits from TTMSFMXBaseControl which implements a basic Fill and Stroke, and handles the style resource files that define the default layout of the component. To change the visuals of the component you no longer have corresponding properties in the object inspector. Right-clicking on the component provides two extra menu items that can be used to edit the style of the component.

Clicking either of these items will automatically drop a StyleBook component on the form when there is not yet one available. A StyleBook holds custom and default styles. When the default style is changed, dropping a new component of the same class will automatically get this changed style as defined in the default style.

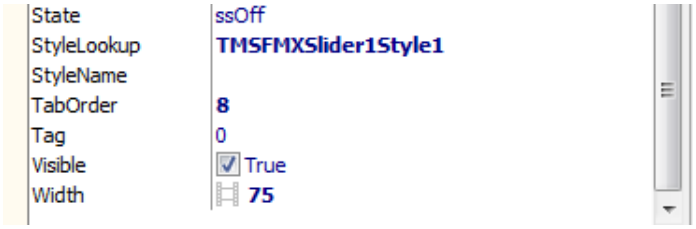



- Edit Custom Style: Clicking on this item starts the IDE style editor and copies the default style of the component. The name of the style is set to the component name on the form and appended with 'Style1'. After changing properties through the editor, the style is then applied to the component. You will notice that the StyleLookUp property is set to the name of the custom style in the stylebook.

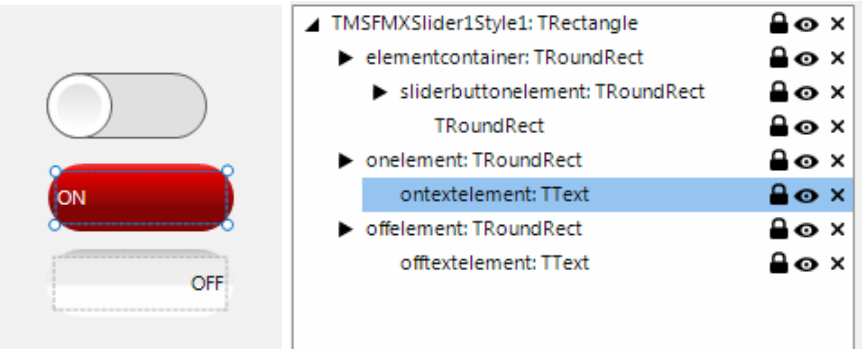
- Edit Default Style: Clicking on this item starts the IDE style editor and uses the default style of the component. As with the Edit Custom Style option, the name of that style is set. The difference between these 2 options is that the default style has a generic name and is applied to all new instances of the component that are dropped on the form. The StyleLoopup property is not set.



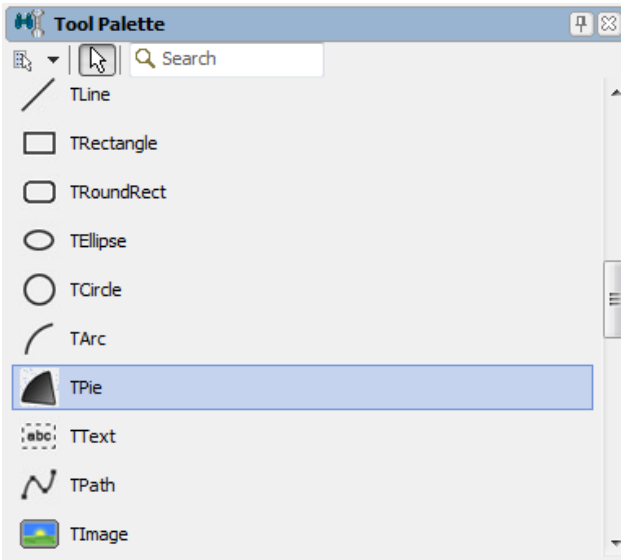

The IDE style editor can be started with these 2 options, or by double-clicking on the StyleBook editor icon on the form. In this example we have a TTMSFMXSlider component that will be altered with a custom style. Notice the TMSFMXSlider1Style1 name that is used for this style. When applying this style, you will also notice the StyleLookup property is set to TMSFMXSlider1Style1.



Each component exists of different styleable elements. Simple click on an element in the editor to change the appearance.



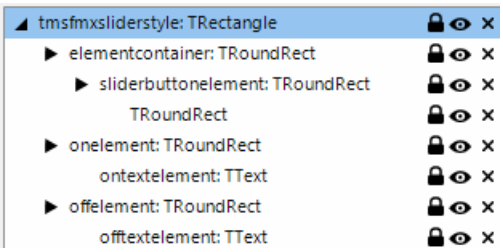

You can also add new elements from the Tool palette.




After applying the Style, the component will have the new custom style.




Dropping a new TTMSFMXSlider component on the form will not adopt this custom style and will have the default style applied. Editing the default style is done in the same way, yet the name of the style differs and each new instance of the TTMSFMXSlider adopts the edited default style.



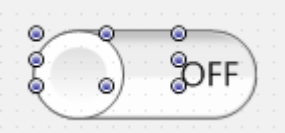

General component properties that do not directly define a visual appearance of the component are still displayed in the Object Inspector. Note though that some properties will affect what is available in the style editor! For example, if a component provides a collection of visible items displayed in the control and it is desirable that the visual appearance of each item can be customized, style elements (shapes) will be dynamically added or removed and be available in the IDE style editor.
In other cases, it is desirable that the appearance for a given type of items in a control is identical. This can be represented as a single style element in the style editor. The component will then internally copy the settings of the style element and apply it to each item displayed in the control.

## Components

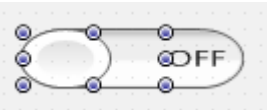
Most of the components in the FireMonkey framework can be scaled and rotated without loss of functionality and quality. As our base control implementation inherits from a base class which supports these features, all of the controls inside the TMS Instrumentation WorkShop set support scaling and rotation.

Scaling: With the Scale property you can specify how large the component must be. The default value of the X and Y property of the Scale is 1. This means that the default component layout is set at one, if you have a component which has 100 pixels width and height dimensions, setting the scale X and Y properties to 1.5 will automatically increase the width and height to 150 pixels. Below are some examples at designtime, which shows the capability of this property.

Scale 1.5



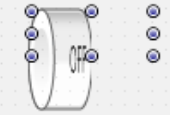

Scale X 1.5 Y 1




Scale 0.5

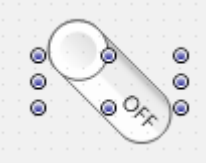
Scale X 0.5 Y 2




Rotation: The rotation property rotates the component around the center by default, which can be changed with the rotationcenter property. Rotating the component does not limit interaction capabilities and functionality.

45°




## TMS Mini HTML rendering engine

Another core technology used among many components is a small fast & lightweight HTML rendering engine. This engine implements a subset of the HTML standard to display formatted text. It supports following tags :

**B : Bold tag**
<B> : start bold text
</B> : end bold text

Example : This is a <B>test</B>

**U : Underline tag**
<U> : start underlined text
</U> : end underlined text

Example : This is a <U>test</U>

**I : Italic tag**
<I> : start italic text
</I> : end italic text

Example : This is a <I>test</I>

**S : Strikeout tag**

<S> : start strike-through text
</S> : end strike-through text

Example : This is a <S>test</S>

**A : anchor tag**
<A href="value"> : text after tag is an anchor. The 'value' after the href identifier is the anchor. This can be an URL (with ftp,http,mailto,file identifier) or any text.
If the value is an URL, the shellexecute function is called, otherwise, the anchor value can be found in the OnAnchorClick event </A> : end of anchor

Examples : This is a <A href= "mailto:myemail@mail.com ">test</A>
This is a <A href="http://www.tmssoftware.com">test</A>
This is a <A href="somevalue">test</A>

**FONT : font specifier tag**
<FONT face='facevalue' size='sizevalue' color='colorvalue' bgcolor='colorvalue'> : specifies font of text after tag.
with

- face : name of the font
- size : HTML style size if smaller than 5, otherwise pointsize of the font
- color : font color with either hexidecimal color specification or color constant name, ie claRed,claYellow,claWhite ... etc
- bgcolor : background color with either hexidecimal color specification or color constant name </FONT> : ends font setting

Examples : This is a <FONT face="Arial" size="12" color="clared">test</FONT>
This is a <FONT face="Arial" size="12" color="#FF0000">test</FONT>

**P : paragraph**
<P align="alignvalue" [bgcolor="colorvalue"] [bgcolorto="colorvalue"]> : starts a new paragraph, with left, right or center alignment. The paragraph background color is set by the optional bgcolor parameter. If bgcolor and bgcolorto are specified,
a gradient is displayed ranging from begin to end color.
</P> : end of paragraph

Example : <P align="right">This is a test</P>
Example : <P align="center">This is a test</P>
Example : <P align="left" bgcolor="#ff0000">This has a red background</P>
Example : <P align="right" bgcolor="claYellow">This has a yellow background</P>
Example : <P align="right" bgcolor="claYellow" bgcolorto="clared">This has a gradient

background</P>*

**HR : horizontal line**
<HR> : inserts linebreak with horizontal line

**BR : linebreak**
<BR> : inserts a linebreak

**BODY : body color / background specifier**
<BODY bgcolor="colorvalue" [bgcolorto="colorvalue"] [dir="v|h"] background="imagefile specifier"> : sets the background color of the HTML text or the background bitmap file

Example : <BODY bgcolor="claYellow"> : sets background color to yellow
<BODY background="file://c:\test.bmp"> : sets tiled background to file test.bmp
<BODY bgcolor="claYellow" bgcolorto="claWhite" dir="v"> : sets a vertical gradient from yellow to white

**IND : indent tag**
This is not part of the standard HTML tags but can be used to easily create multicolumn text
<IND x="indent"> : indents with "indent" pixels

Example :
This will be <IND x="75">indented 75 pixels.

**IMG : image tag**
<IMG src="specifier:name" [align="specifier"] [width="width"] [height="height"] [alt="specifier:name"] > : inserts an image at the location

specifier can be: name of image in a BitmapContainer

Optionally, an alignment tag can be included. If no alignment is included, the text alignment with respect to the image is bottom. Other possibilities are: align="top" and align="middle"

The width & height to render the image can be specified as well. If the image is embedded in anchor tags, a different image can be displayed when the mouse is in the image area through the Alt attribute.

Examples :
This is an image <IMG src="name">

**SUB : subscript tag**

<SUB> : start subscript text
</SUB> : end subscript text

Example : This is <SUP>9</SUP>/<SUB>16</SUB> looks like 9/16

**SUP : superscript tag**
<SUP> : start superscript text
</SUP> : end superscript text

**UL : list tag**
<UL> : start unordered list tag
</UL> : end unordered list

Example : <UL>
<LI>List item 1
<LI>List item 2
<UL>
<LI> Sub list item A
<LI> Sub list item B
</UL>
<LI>List item 3
</UL>

**LI : list item**
<LI [type="specifier"] [color="color"] [name="imagename"]>: new list item specifier can be "square", "circle" or "image" bullet. Color sets the color of the square or circle bullet. Imagename sets the PictureContainer image name for image to use as bullet

**SHAD : text with shadow**
<SHAD> : start text with shadow
</SHAD> : end text with shadow

**Z : hidden text**
<Z> : start hidden text
</Z> : end hidden text

**Special characters**
Following standard HTML special characters are supported :
< : less than : <

> : greater than : >
& : &
" : "
  : non breaking space
™ : trademark symbol
€ : euro symbol
§ : section symbol
© : copyright symbol
¶ : paragraph symbol