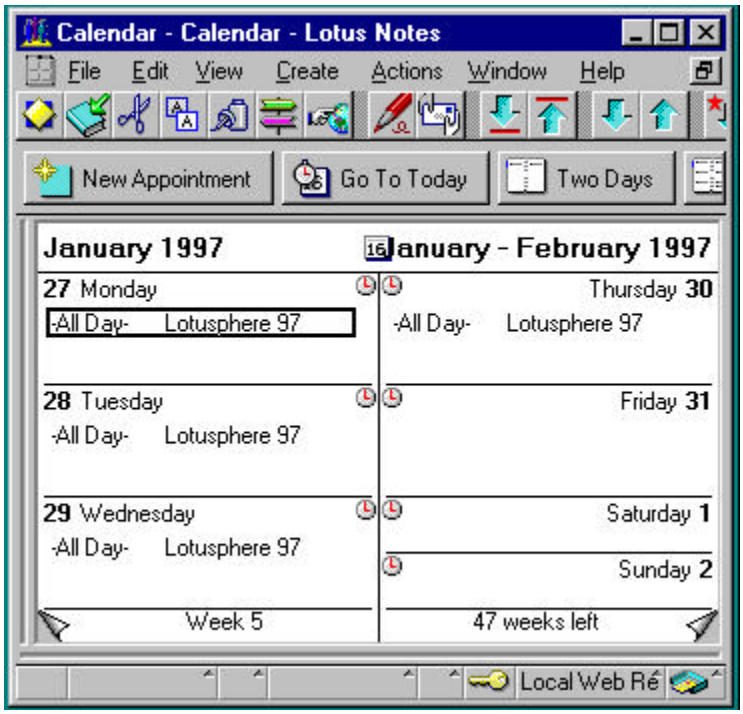
Creating a Calendar View in Notes

by Cathy Duffy

[Editor's note: This article resides in "Notes Today", the technical Webzine located on the http://www.notes.net Web site produced by Iris Associates, the developers of Domino/Notes.]

Overview

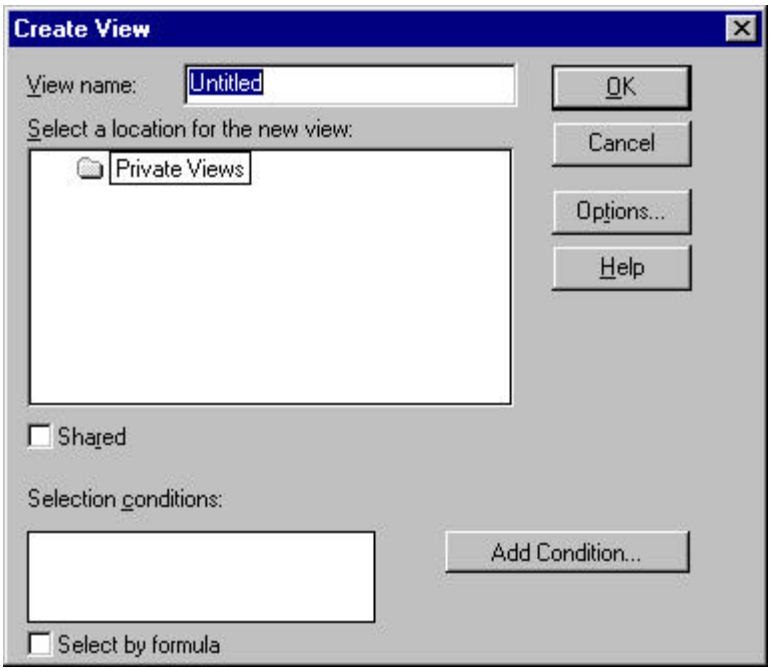
The Notes 4.5 Calendar view allows you to display documents on a calendar, rather than in a traditional tabular format. Calendar views are used extensively in Notes 4.5 Calendaring and Scheduling (C&S) applications, but that shouldn't stop you from using Calendar views in any application that would benefit from a display organized by date and time.



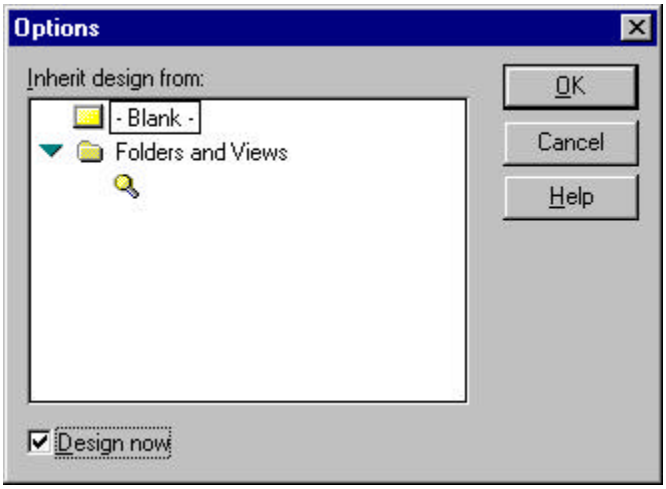
Creating a Calendar View

In order to create a calendar view, you must do the following:

1. From the menu, select Create, Design, View.
2. Check Shared or leave as Private.
3. Click the Options button.

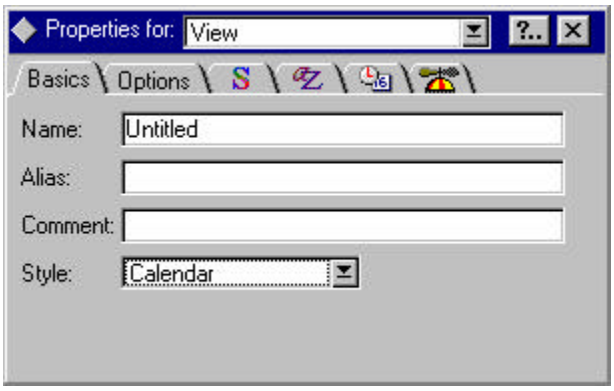


4. For "Inherit design from:" select Blank and click the Design now box.



5. Click OK on each dialog box until you find yourself in view design mode.

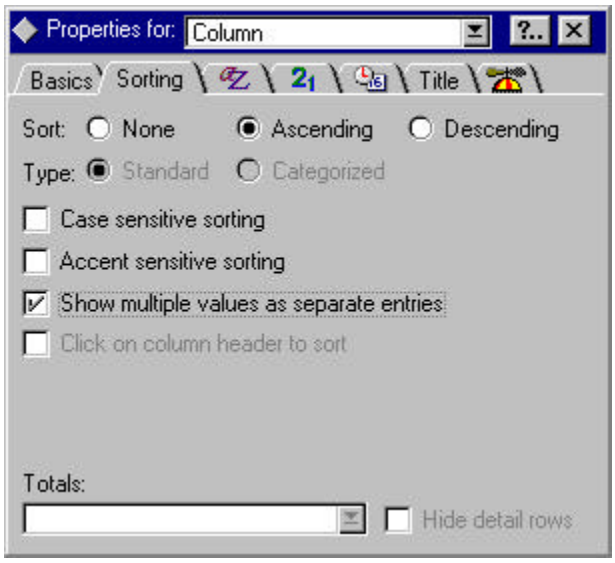
6. Under View, Properties, choose Calendar as the style.



Designing the columns

Designing the first two columns is the most important aspect of a Calendar view's design. These columns **must** be designed as follows in order for the calendar to function properly. (Notes 4.5 won't let you save the view without these two columns.)

Column 1: Make it hidden and sorted and include a Time/Date value, or list of Time/Date values. The time portion is required because it is used for indexing. If you are using a list of Time/Date values, be sure that "Show multiple values as separate entries" is turned on in the properties for Column 1. This will allow the document to appear on more than one day in the calendar. For correct usage of Time/Date fields and values in Calendar Views, see Ryan Jansen's recipe for "**Using Time/Date values in LotusScript**."



It is also a good idea to have your Calendar view's selection formula refer to the date/time field used in Column 1. For example, the formula for this column might be as follows. (AppointmentType "2" is an event that might span multiple days, so the column value is a list.)

```
StartTime := @Time(CalendarDateTime);
DateList := @Explode(@TextToTime(@Text(CalendarDateTime) + "-" + @Text(EndDateTime)));
@If(AppointmentType = "2"; @TextToTime(@Text(DateList) + " " + @Text(StartTime)); CalendarDateTime)
```

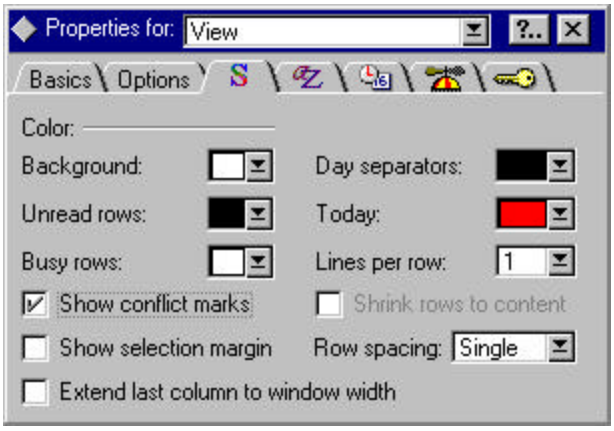
In this case the selection formula is @IsAvailable(CalendarDateTime), because the CalendarDateTime field is required to calculate Column 1.

Column 2: This should also be hidden and must be a duration in minutes (for example, (EndDateTime-StartDateTime)/60). If duration is not relevant, this column should be 0 (zero).

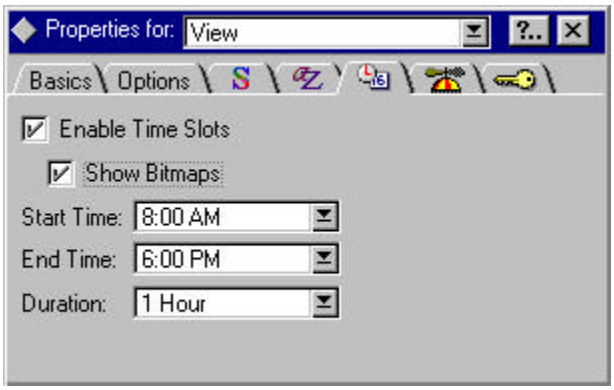
Customizing Calendar Views

Keep these tips in mind when you design customized calendar views:

- Column titles are irrelevant because they are never displayed in Calendar Views.
- You might want to show conflict marks, a vertical line that indicates that you have more than one appointment for the same time.
- You might not want to show the selection column because it takes up extra room on the screen. If the selection column is hidden, it will appear automatically if you have selected documents in the view.
- You might want to change the default colors (like the color of "Today"). The following panel in the view's property box that allows you to do this:



- If you mark a document in a calendar view for deletion, it appears with a strike-through.
- You might want to show the time slots on the calendar. If so, set the defaults through the following pane in the calendar view's property box:



- If you have a time slot column in your calendar view, you should be sure to go to the date/time pane in the column's property box and set the display for Time Only. You will probably also just want to display hours and minutes instead of hours, minutes, and seconds.
- You may want to include an action bar on your calendar view, which allows the user to change the display of the view from two-day (@Command([CalendarFormat]; "2")), to one-week (@Command([CalendarFormat]; "7")), to two-week (@Command([CalendarFormat]; "14")), to one-month (@Command([CalendarFormat]; "30")).

Calendar Views and LotusScript Events

The Calendar view also has LotusScript events associated with it. These events are as follows.

RegionDoubleClick - Allows you to cause something to happen when the user double clicks on a date or a time slot. In the example below, on RegionDoubleClick you create an Appointment document that includes the date/time that the user double clicks on as StartDateTime. If you click somewhere that does not have a date/time associated with it, Notes does not compose the document. The ws.CalendarDateTime must be used in the QueryOpen event; it is not supported in PostOpen.

```
(This is from the calendar view.  ws was declared elsewhere as NotesUIWorkspace.)
Sub Regiondoubleclick(Source As Notesuiview)
   Set ws = New NotesUIWorkspace
   If source.CalendarDateTime <> "" Then Call ws.ComposeDocument("", "", "Appointment")
End Sub

(This is from the Appointment document.  ws was declared elsewhere as NotesUIWorkspace, and
SelectedDate is a global variable that can be used to set a field such as StartDateTime.)
Sub Queryopen(Source As Notesuidocument, Mode As Integer, Isnewdoc As Variant, Continue As Variant)
   Set ws = New NotesUIWorkspace
   SelectedDate = ws.CurrentCalendarDateTime
End Sub
```

QueryPaste - Allows you to check something when a user attempts to paste a doc on a time slot. In the example below, QueryPaste prevents users from pasting documents onto a weekend day.

```
(The CalendarDateTime property of the NotesUIView class indicates the time slot they are attempting to
paste to.)
Sub Querypaste(Source As Notesuiview, Continue As Variant)
   DayOfWeek = Weekday(source.CalendarDateTime)
   If DayOfWeek = 1 or DayOfWeek = 7 then Continue = False
End Sub
```

PostPaste - Allows you to cause something to happen when a doc is pasted into a time slot. In the example below, PostPaste changes the StartDateTime to the value of the new time slot. (Note that this does not happen automatically.)

```
(You are pasting a collection of documents, because a person could select more than one document,
copy/cut, then paste into this time slot.  documents and note were declared elsewhere as
NotesDocumentCollection and NotesDocument.)
Sub Postpaste(Source As Notesuiview)
   Set documents = source.documents
   Set note = documents.GetFirstDocument
   While not(note is Nothing)
      note.StartDateTime = source.CalendarDateTime
      note.Save True, True, True
      Set note = documents.GetNextDocument(note)
   Wend
End Sub
```

QueryDragDrop - Allows you to check something when a user attempts to drop a doc onto a time slot. In the example below, QueryDragDrop prevents users from drag/dropping documents if they are not at least Designers in the db's ACL.

```
(session and db are declared elsewhere as NotesSession and NotesDatabase.)
Sub Querydrapdrop(Source As Notesuiview, Continue As Variant)
   Set session = New NotesSession
   Set db = session.CurrentDatabase
```

```
    If db.CurrentAccessLevel < ACLLEVEL_DESIGNER then Continue = False
End Sub
```

PostDragDrop - Allows you to modify the appropriate fields when a doc is dragged onto a time slot. This is very similar to PostPaste, and you might want both events to call the same "put" routine. In the example below, the PostDragDrop event calls a second routine, which changes the StartDateTime to the value of the new time slot, but only if the Status does not equal "Complete".

```
(Pass the subroutine the UIView object.)
Sub Postdragdrop(Source As Notesuiview)
    Call PutDocument(Source)
End Sub

(You are putting a collection of documents, because a person could select more than one document and
drag them into this time slot.  documents and note were declared elsewhere as NotesDocumentCollection
and NotesDocument. 'source refers to the NotesUIView.)
Sub PutDocument(source)
    Set documents = source.documents
    Set note = documents.GetFirstDocument
    While not(note is Nothing)
        If note.Status(0) <> "Complete" then
            note.StartDateTime = source.CalendarDateTime
            note.Save True, True, True
        End If
        Set note = documents.GetNextDocument(note)
    Wend
End Sub
```

Caveats to Using Calendar View Events

When designing using calendar view events, keep the following in mind:

- In both QueryPaste and QueryDragDrop, you probably do not want to be checking something about the document because the user may have selected several documents to drop or paste. If one of the documents fails the criteria, what will you do? Setting the Continue flag to False will terminate the entire operation, which is probably not what you want. Therefore, if you want to check something about a document, you should do so in the Post event and simply not process the offending document.
- If you have a PostPaste, you will probably also have a PostDragDrop. If you want to have one but not the other you should disable the other by setting Continue to False in its Query event.
- Remember to do a NotesDocument.Save in PostPaste and PostDragDrop, or your changes will not be saved.
- Remember to do a NotesUIWorkspace.Refresh at the end of PostPaste and PostDragDrop so that the changes will be visible immediately.

Iris Sample Calendar Database

You can find examples of calendar views, LotusScript events and forms in the Sample Calendar database that you can later use as you develop your own applications with Calendar views. You can **view** the sample database now, or download it:

Download Iris Sample Calendar database (224Kb – available in Notes Today on http://www.notes.net)

ABOUT THE AUTHOR

Cathy joined Iris in May 1995 to develop Notes Release 4 templates (approve4.ntf, discuss4.ntf, doclib4.ntf, doclibm4.ntf, and doclibl4.ntf). She has been working on the mail template since October 1995 and has been developing applications (both Notes and otherwise) for 15 years. Her area of expertise is workflow applications.