

Manual for CalendarX: A Calendar for Plone CMS portals

Lupa Zurven, <http://calendarx.org>
v. 0.6.6-06, 7th public draft, January 3, 2006 for CalendarX-0.6.6(stable)

Table of Contents: (07 - last important update of that section, by draft number)

Part I:	CalendarX Overview and What's New in CalendarX (07)	3
Part II:	A guide for Users: How to browse and add/edit events (06)	11
Part III:	A guide for Administrators: How to configure the calendar (06).....	13
Part IV:	A guide for Developers: Modifying functionality in CalendarX (06)	20
Part V:	Use Case: Creating a Resource Scheduler in CalendarX (06).....	22
Appendix A:	Installing CalendarX (06)	32
Appendix B:	Guide to property sheets in CalendarX (07)	41
Appendix C:	Guide to python scripts in CalendarX (04)	65
Appendix D:	Guide to page template views and macros in CalendarX (06)	71
Appendix E:	Guide to CSS and JavaScript code in CalendarX (06)	76
Appendix F:	About Python, Zope, Plone, and CalendarX branches (06)	77

Note: This draft of the *Manual for CalendarX* contains all material from the /docs folder, but is recompiled and beautified, and expanded a bit, and gradually some screenshots are appearing. But it's still a draft.

The HISTORY.txt and all other important docs will still be available in the /docs folder, just as they always have been, and they are kept updated as well.

Status of the *Manual for CalendarX*:

Part I:	CalendarX Overview.....	(second draft)
Part II:	A guide for Users.....	(just a note)
Part III:	A guide for Administrators:	(real notes)
Part IV:	A guide for Developers:	(brief notes)
Part V:	Use Case: Resource Scheduler.....	(first draft)
Appendix A:	Installing CalendarX	(full notes)
Appendix B:	Guide to property sheets in CalendarX.....	(full notes)
Appendix C:	Guide to python scripts in CalendarX.....	(first draft)
Appendix D:	Guide to page template views and macros in CalendarX	(full draft, pics)
Appendix E:	Guide to CSS and JavaScript code in CalendarX.....	(just a note)
Appendix F:	About Python, Zope, Plone, and CalendarX branches	(brief notes)

Title: Manual for CalendarX: A Calendar for Plone CMS portals
Author: Lupa Zurven, <http://calendarx.org>
Version: v. 0.6.6-06, 7th public draft, January 3, 2006 for CalendarX-0.6.6(stable)
Contents copyright Lupa Zurven, 2004-2006. Some rights reserved.

This manual is made available in electronic form, and may be freely downloaded either with the CalendarX software, or it may be downloaded from the CalendarX.org website, or it may be made available at the SourceForge.net website. In a significant departure from the first public draft, this manual is also released under a much less restrictive license. For details of this license, see the CreativeCommons website, and specifically the 2.0 version of the "Attribution-NonCommercial-ShareAlike 2.0" license:

<http://creativecommons.org/licenses/by-nc-sa/2.0/>

In short, you may redistribute this work or derivative works for non-commercial purposes, as long as you give the original author credit.

In full, you are free to copy, distribute, display, and perform the work and to make derivative works under the following conditions:

- * Attribution: You must give the original author credit.
- * Noncommercial: You may not use this work for commercial purposes.
- * Share Alike: If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.
- * Reuse: For any reuse or distribution, you must make clear to others the license terms of this work.
- * Any of these conditions can be waived if you get permission from the copyright holder.
- * Your fair use and other rights are in no way affected by the above.

I make my poor living as a teacher with a hobby of open source programming, and it takes time and effort to write this manual and this software. I give my software away as an open source project, and I give this manual away to anyone who downloads CalendarX or obtains it from my websites. You can use this manual as a starting point for your own portal users, or deliver copies of it to your development team, whatever.

Just don't sell my words, thank you. If you need to sell a manual to your clients for your version of CalendarX, please write your own, or contact me for licensing this one. I am happy to license the manual or portions of it to organizations for distribution, either on a per piece or site license basis, and I am happy to be hired to modify and customize the manual (and CalendarX itself) to make both of them perfectly suit your users, or you can do the customization of the manual yourself, but a license is absolutely required for such commercial distribution.

I hope you find this *Manual for CalendarX* (and CalendarX itself) useful. If you really like this product, consider clicking the Donate button on the CalendarX.org website to send me a little gift. +lupa+

Part I: CalendarX Overview

CalendarX is a web-based calendar application that is easy for users, allows a great deal of flexibility in configuration options to calendar administrators, and provides a modifiable, open source code base for developers to build on. The standard calendar includes five calendar views (multimonth, month, week by days, week by hours, and daily) and provides metacalendar tools: each event has a subject, and the calendar can be filtered to show only those events belonging to one or more selected subjects. This filtering can be controlled by the calendar user, or a calendar can be configured by an administrator to show only desired subjects to all users. Users may learn more about each event on the calendar by simply rolling the mouse over the event's title (which shows a small popup box containing more information) or by clicking on the event and going directly to a page with the event's details.

CalendarX is built for the Plone content management system (CMS), which allows for a great variety of customization options for developers. In order to make basic customization easier for calendar administrators, CalendarX utilizes handy property sheets to allow access to many configuration options. These property sheets are available via the ZMI (Zope Management Interface), in the `portal_skins/CalendarX` folder.

There are many options that can be configured by the CalendarX administrator through the property sheets. These include controlling the types of events shown on the calendar, the subjects shown, the amount of information shown in the event rollover popup text, etc. There are also many controls on the CSS display of the calendar. First, the many colors of the calendar itself can be adjusted to fit the look-and-feel of the rest of the Plone site. Second, CSS styles can be applied to each event's text on the basis of what its subject is. Third, each event subject can also have a unique icon associated with it, providing rapid visual clues on a busy calendar.

Finally, for developers who wish to modify it, the source code for CalendarX is accessible to allow overrides for each instance of CalendarX in a site. It is even possible to create subcalendars, as is illustrated here in the creation of a basic Resource Scheduling calendar. Custom views can also be created for CalendarX. CalendarX is released as open source software under the General Public License (GPL).

Instructions herein for Plone and ZMI are based on Plone 2.0.5. Some of these instructions will likely not work for Plone 2.1, but the equivalent task should be easily discerned. I've only just started using Plone 2.1.1 myself, and have made a few changes to CalendarX accordingly. Plone 2.1.2 is due out any day now, and then I'll start using Plone 2.1 in earnest.

About this *Manual for CalendarX*:

For the purposes of this manual, I'm considering the needs of three types of people:

Users: visitors and/or members of a Plone portal who may need calendar instruction for browsing and/or for entering Events for the calendar.

Administrators: someone setting up one or more CalendarX instances for a portal and configuring CalendarX to suit the needs of the portal.

Developers: programmers needing a solid calendar application for customization, or for advanced Administrators looking for additional functionality.

Part II is a **User's Guide**, a description (with pictures) on how to use CalendarX. **Part III** of the manual is an **Administrator's Guide**, offering instruction on configuring CalendarX to meet the needs of a portal's visitors. **Part IV** is a **Developer's Guide**, providing a closer look at the inner workings of CalendarX, a brief guided tour to the source code and where to look for the various functions of CalendarX. **Part V** of the manual is a case study in advanced configuration: how to use CalendarX with its subcalendar abilities to create a *Resource Scheduling* calendar suitable for an organization with equipment to loan out to its members. It would be equally suitable for a *Room Scheduling* calendar, I believe, although I personally haven't used it for such yet.

CalendarX Significant Features:

CalendarX has many features that make it a usable web-based calendar right out of the box, but its real strength is in its configurability and customizability. We will first list some of the features that make it useful with no modifications to the configuration property sheets at all.

Major Out-Of-The-Box Features:

1. Comfortable look-and-feel, like a regular calendar with multimonth, month, week-by-day, week-by-hour, and day views.
2. Each event can be assigned one or more Subjects, and the calendar can be sorted by the user to show only events belonging to one or more of those Subjects.
3. Navigation is sensible; users can go to the previous or next month, week or day via a simple text link, or they can jump to a specific date in the past or future.
4. Internationalization. CalendarX displays dates and phrases in eleven languages (ca, cs, da, de, es, en, fr, it, ja, nl, pt-br).

CalendarX has many additional features that can be configured easily by the calendar administrator without changing any of the source code. This is accomplished through the use of property sheets, which contain attributes that can be adjusted through the use of checkboxes and text boxes through the ZMI.

Major Configuration Features:

1. Colors, fonts, font sizes, and calendar cell sizing are all adjustable, as well as the type of information shown to users when they roll the mouse cursor over an event on the calendar.
2. Subjects for the events can be modified. Have as many as you like, and similar subjects may be grouped and given a group name, or nicknames can be applied to subjects.
3. Calendars can be restricted to show a limited type of event in several ways. For example, a Music calendar could show only those events that have Music as their

- subject, or perhaps that have Concert, Recital, Rehearsal or Jam Session as their subject.
4. Users can be shown a link ("Add New Event") that allows them to add new events to the calendar if they have the proper permission to do so (based on their authentication and role in the Plone CMS). You can also control the type of event and where this event will be stored through this link method.
 5. You can manage the workflow of the event publishing method -- should user contributed events go straight up onto the calendar? Or should the events go through a review/publish cycle first? You decide with CalendarX and Plone.
 6. Subcalendars can be created with different properties from the master calendar. This is useful for creation of specialty calendars with many events, such as a Resource Scheduling calendar, for people to reserve rooms or equipment. Read Part V of this manual for an example of using CalendarX as a Resource Scheduling calendar.

CalendarX is further customizable by changing the source code itself. CalendarX is open source software released under the GPL, which gives you a great deal of freedom. You can change anything in the source code of CalendarX that you want, and all of the source code is available to you for this purpose. Here are a few ideas on what can be accomplished through relatively modest changes in the source code itself.

Possible new functionality available by rewriting CalendarX code:

1. Creation of new views, or changing the behavior of existing views.
2. Creation of a customized version of CalendarX to distribute.
3. Creation of custom Event types, perhaps events that include space for an uploaded image, or a sound file, or a digital video clip. Or events that are used to request a reservation for a certain time slot on the calendar. Actually, you don't need to rewrite CalendarX at all for custom Events. For this you would use Archetypes in Plone. CalendarX doesn't have its own Event types.
4. Creating new queries to access outside databases of events.

CalendarX Licensing:

If you do create something really neat with CalendarX, we'd greatly appreciate it if you'd share your work with the CalendarX community. We have a community website at <http://CalendarX.org> where you can add Tutorials or HowTo documents, or enter bug reports or feature requests. If you create interesting modifications to the source code underlying CalendarX, we'd certainly like to hear about those too, so that we can improve CalendarX for everyone. You don't necessarily have to share your modifications, but you should be aware that under certain circumstances you may have to share your code, or else you'll not be in keeping with the license agreement that accompanies CalendarX.

CalendarX is licensed under the General Public License (GPL) to give you a great deal of freedom with the code, and to give the developer (and copyright holder) a great deal of freedom as well. The GPL allows you to customize CalendarX with NO restrictions, and to distribute

your customizations with very few restrictions. In brief, this software is free and you can't license it in any way that takes away its freedom. You can give away your changes to others, even a deep rewriting of the code, but you can't do so under different licensing conditions if they are not compatible with the GPL. So for instance, you are certainly allowed to make changes that would make CalendarX more appropriate for a corporate intranet calendar and install such a program for as many corporations as you can convince to use it. You may even charge the corporations for your services customizing, installing and configuring that program, or for training and maintenance of the program, or for any other ancillary services. But you must license it to those corporations under the GPL or a compatible license, which preserves all these same freedoms for them. You must provide the source code to these clients. They will have the right (under the GPL) to further modify and give away the changes you made for them or to their copy of the code (that's what the GPL gives them). They may well choose not to give these changes away (many don't), but they must be granted the right to do so. So to put it another way, you cannot use CalendarX source code as a basis for a proprietary (non-free) computer program. Use CalendarX for making free software.

Why this license for CalendarX? The GPL is designed for software that we want to keep free for growth and exploration, free to be customized and crafted by anyone who desires to do so. In order to establish this freedom, the GPL says you can change things in the software however you want, but in order to let others use your changes, you must release your code to them with the same freedoms. If you just want to change the code for yourself and not release your changes, that is fine! Go ahead -- you are free to keep any changes to yourself! But in order to share your code that utilizes CalendarX code, you must ensure our CalendarX's freedom by licensing your modifications in a compatible way.

Note: many people misunderstand the GPL in this regard. They seem to think the GPL says that if you make any changes to the source code period, under any circumstances, you have to then make those changes public somehow. The GPL doesn't say that at all. If you put CalendarX on a server in your business and tweak the code, you don't have to release those changes. If you distribute your tweaked code to others, then the GPL kicks in. If you distribute CalendarX, you have to give the recipients the source with changes and you have to give the recipients the right to tweak and redistribute just as I am giving that right to you.

Of course, this GPL is not such an onerous license. IBM identified a tremendous opportunity in selling consulting services based on the Linux operating system; Linux is released under the GPL so IBM doesn't really sell Linux, but it does sell consulting, installation, configuration, training, and of course server hardware all designed to help organizations run computers with Linux. If IBM feels comfortable making money with GPL software, I think the rest of us can find ways to make it work too.

Therefore, you are granted the freedom to make money installing, configuring and customizing CalendarX for others, but you must release to them your code changes under comparably licensed conditions (for simplicity... just stick to the GPL). These are the same terms that apply to Plone, which is also (as of versions 2.0.5 and 2.1) released under the GPL. These are comfortable terms for most people, and work fine for most open source consultants. If you aren't comfortable with those licensing terms, you should work with a different calendar

application, one that allows you to change the source code and release it as your own and sell it as your own. We chose the GPL for CalendarX because we want to build a community around the calendar that feels that their contributions will be kept free for the good of all. Many of the best features of CalendarX come either from code contributions or the freely offered suggestions of our users. If you have any questions about CalendarX and its licensing, feel free to contact me and ask.

Licensing FAQs:

Q: If I want to use CalendarX as a starting point for my proprietary calendar app, will you consider licensing it to me under a non-GPL license?

A: No. We don't want to, and we really can't. CalendarX uses contributed code that has been given under the conditions that CalendarX be a GPL'd product. Also, CalendarX is fairly tightly integrated with Plone, importing key modules and such, and since Plone is GPL'd, CalendarX is best kept a GPL product. Besides, we like it this way, not having to worry about licensing considerations and knowing it will always be around to use as long as it has use for folks.

Q: Has anyone actually asked that question about CalendarX?

A: Yes! Well, no, not exactly that one, but something fairly close to it.

```
<dtml-comment>
```

```
"We"? Why do I keep using "we" when it seems to be just "me"?
Lupa has many friends and helpers and has been working on and off
on CalendarX in Zope for three years before actually releasing it
for Plone in 2004. Ongoing conversation with everyone helps
tremendously. When it seems appropriate, as in the case of
licensing, I'll use the royal "we" to indicate that it's more
than just me that thinks this way. Other times, when I'm writing
for myself, I'll use first person singular. And when I'm feeling
testy, I might write in the third person impersonal or even the
seldom used but often admired first person abusive. So just
watch it.
```

```
</dtml-comment>
```

Q: Are there any caveats that you might add about CalendarX?

A. Of course there are.

Caveats.

0. These caveats have been included here since the first releases of CalendarX and I've never had excuse to need them. But here they are.

1. For your health, and mine, I've tried to document nearly everything you need to accomplish significant things with CalendarX. Please read all the documentation before you ask questions to the list or at CalendarX.org. There's lots of it, even to the point of explaining each configurable attribute in the property sheets. It is not, however, capable of explaining

all the possible things you can do with CalendarX. Be creative, and share your successes and failures with us at CalendarX.org.

2. Part of the code for CalendarX at this point in time comes from the previous products it builds on -- mainly PloneCalendar (see the README). I know most of that code now, and have replaced large parts of it with my own. But I can't vouch for everything it does yet. Do with this product what you will, but use it responsibly.
3. At this point, I have no idea how CalendarX may interact with or interfere with any other Zope/Plone products or other software. There are likely to be name overlaps between objects in this code and those in other apps, and in a world with Spammish Acquisition, many things are possible. I have seen no bad such interactions, although one potential user claimed to be totally unable to import the Fezzik folder (on the 0.2 branch) no matter how he tried (a month later, this user wrote back and said there are no problems now with the most recent version). I think this must have been an interaction with this particular Plone setup, but I have no idea. I've tried it without any problems on several systems and operating systems, but I don't know what will happen on your site, because you may have customizations that do cause problems. Forewarned should be four armed, but you never know. In the meantime, do with this product what you will, and remember to use it responsibly.
4. This product has not been thoroughly tested and is likely to have bugs. In fact, it doesn't even have a unit test suite for it yet. You have been warned. That said, I and many others use this product routinely in production Plone sites, and deliver versions of it to clients for use on their production sites. Do with this product what you will, but use it responsibly. Practice safe Plone.
5. No animals were harmed in the creation of this Product, and no testing on live animals was performed. You should not attempt to use this Product in the manufacture or assembly of weapons of mass dysfunction, as it is likely to function properly, thus causing failure of intended weapons. Any attempt to disassemble or reverse engineer the functionality of this Product is strictly encouraged, and any successes in such endeavor should be shouted from the mountaintops. Or at least let me know.

+lupa+

What's New in CalendarX version 0.6.6(stable)

Most recent changes from the HISTORY.txt file (you can read the whole history of changes in the 0.6 software branch in your /docs folder):

v0.6.6(stable)

Code base: v0.6.5(stable)

Status: Stable release, no known bugs, tests as valid XHTML

Transitional 1.0 at w3c. Four small bugfixes from 0.6.5: (1) error using restrictToThisListOfSubjects, (2) repeated event icons shown for some events with multiple Subjects selected, (3) unnecessary span tags within certain CSS spans in some views, and (4) CSS for rollover highlighting not properly set in calendar.js. Bugfix #4 also required addition of one new property in the CX_props_css and some changes to the calendar.js file so that it reads properties for highlighting cells properly (see below).

mod: getEventsBetweenZC.py: changed q_xsub initialization to an empty list ([]) instead of zero (0) to avoid error on restrictToThisListOfSubjects.

mod: getEventsBeforeZC.py: changed q_xsub initialization to an empty list ([]) instead of zero (0) to avoid error on restrictToThisListOfSubjects.

mod: CX_props_macros.pt: Eventlister and MMEventlister macros: removed the offensive tal:repeat="subject event/Subject" command that was in place for useEventTypeIcons option. Bug caused multiple showing of the EventTypeIcon if an event had multiple Subjects selected... which is only appropriate for use with the useSubjectIcons property.

mod: CX_props_macros.pt: prevnextcurrentlinks: added tal:omit-tag="" to join all the i18n components into the control of one class="center" span.

mod: weekbyday.pt: added tal:omit-tag="" to join all the i18n components into the control of one class="day" span.

mod: weekbyhour.pt: added tal:omit-tag="" to join all the i18n components into the control of one class="day" span.

mod: CX_props_css.props: added calTableDataEventHighlightBackgroundColor property to allow changing the highlighted color of the table cell when user mouseovers an event.

mod: calendar.js has been renamed calendar.065.js

new: calendar.js.dtml: this is the old calendar.js file, but converted to a Filesystem DTML Method so that it can read in values from the CX_props_css property sheet. In the skins, it still looks like it is named calendar.js (and it is), but on the filesystem it is called calendar.js.dtml. This now reads the following properties in from CX_props_css.props:

calTableDataEventHighlightBackgroundColor for rollover highlighting color

calTableDataNoEventBackgroundColor for empty calendar cells

calTableDataEventBackgroundColor for non-empty calendar cells

calTableDataOutOfMonthBackgroundColor for out of month calendar cells

NOTE: by default, the new calendar.js.dtml will be used instead of the older calendar.js (renamed to calendar.065.js). To use the old calendar.js, without reading values from the property sheet, change each of the view templates in

the header information from reading "calendar.js" to reading "calendar.065.js" and everything should work fine, just as in version 0.6.5 of this software.

Part II: A guide for Users: How to browse the calendar and add/edit events

CalendarX is easy and intuitive to use. You don't need no stinking guide just to use the calendar. If you simply want to browse through the calendar, there are only a few things you need to know:

- * How to change views (3 Month, Month, Week by Day, Week by Hour and Day).
 - (duh. Click on the tabs that have those names.)
- * How to read details about an Event.
 - (duh. Click on the event, and you'll get a full page with all details.)
- * How to change dates displayed on the calendar (link to Next period, Previous period, or Jump to a specific date)
 - (duh. Click on those links, or set the Date and click the button to Jump to it.)
- * How to look at just one Subject of Events at a time, or a selection of Subjects.
 - (select the checkboxes for the subjects interesting to you. Then "Refresh".)

If your calendar has been set up to allow you to add and/or edit events for the calendar, then there are some extra steps you should learn:

- How to Add an Event (three ways).
- How to enter the Edit mode for an Event.
- What the various parts of an Event entry mean.
- Getting your event "Published".

This User guide will cover the instructions for browsing the calendar first, and then cover the details of adding and editing events. All instructions here will be for use with a standard installation of CalendarX... modifications to the calendar by your website Administrator may cause your calendar to look or act somewhat differently, but it should be similar enough that you don't have much problem following these instructions.

Welcome to CalendarX!

Step One. Let's get to know your calendar. Figure 1 is a

[to be continued, someday. Anybody want to contribute this? actually, I'm not sure I'll ever write it, because each Plone site is set up differently for users, so Site Managers should do this part. And CalendarX is pretty darn easy to use. OK, that's enough. On to more pressing issues....]

[notes from the CalendarX Help Tab - this is an OPTIONAL help tab that you can add to your calendar by selecting the option in the CX_props_calendar property sheet. You can also customize this text for your site.]

Use the tabs across the top of the calendar screen to select different ways of viewing the organization's calendar.

Use the "previous" and "next" links to scroll through the dates to reach the date you are interested in, or alternatively you can select a date from the drop-down menus and click on "jump" to go there directly.

In any of the week/month views you can click on an individual date to view the details of events happening on that particular day.

The default view of the calendar is to show all the important dates in the organization. However, if you are only interested in (for example) dates relevant to one category of events, you can use the tick boxes and "Refresh" button near the top of the page to make the the calendar show only those dates. You can have as many selection criteria as you like, but note that you will need to make sure "view all" is NOT ticked if you want the filtering to work.

On all screens, you can hover the mouse over a particular event to find out more information about it. There may also be supplementary information for the event which you can find by clicking on it.

TO ADD AN EVENT:

Click on your *my folder* link, and select "Event" from the dropdown list. Enter information about your event, and make certain to fill out all the relevant parts like End date and Start date. Make your entries informative so that viewers of your event can find out everything they need to know.

Part III: A guide for Administrators: How to configure the calendar

How to get to the property sheets.

How to change some common properties.

How to migrate from an older version of CalendarX.

[notes from the /docs folder]

From the CUSTOM.txt file:

I. Overview

A. Go to portal_skins/CalendarX. Find one of the seven property sheets (named CX_props_yadayada). Click on one of them to view it, and then click the "Customize" button to move a copy of the property sheet into the /custom folder. Now you can modify the properties (use the Properties tab!) and start changing the calendar behavior. In brief:

CX_props_calendar: controls most basic calendar functionality, including what types of events are shown, how the Subject bar is displayed, etc.

CX_props_css: provides many opportunities for changing the colors and fonts displayed in the calendar.

CX_props_custom: does nothing. If you add new functionality to your calendar and need properties, put them in here.

CX_props_popup: checkboxes to select which text is displayed in the rollover text box associated with each event displayed in the calendar.

CX_props_addeventlink: If you wish to have an "Add New Event" link displayed in the subject bar, configure the link here.

CX_props_subcalendar: If you have subcalendars beneath your main calendar, you will need to configure them here.

CX_props_eventdisplays: allows you to use different icons and CSS classes based on the Subject of each event.

Almost definitely, you will want to customize **CX_props_calendar**. It has many basic options for the calendar.

TIP: In some Plone installs, I've run into the situation where the customized property sheets were being ignored by the skins mechanism. Couldn't find any good reason for it, even though they were located in the /portal_skins/custom folder as usual.

SOLUTION: Cut and paste your customized property sheets into your CalendarX instance folder, using the ZMI to do this. So if your calendar is named "cal", in the ZMI this will be a folder... put the property sheets inside there, and the Zope acquisition mechanism will pick them up without using the skinning mechanism.

B. You can customize anything else in the /CalendarX folder. Page templates, macros, CSS or Python scripts, or Javascript... anything you find there. Usually this means clicking the "Customize" button which puts a copy of the object in the /custom folder for you, where you can change things and refresh your calendar to see the results. TIP: see tip in #A above.

C. To create more than one instance of CalendarX is easy... just add another one from the dropdown list. However, all your calendars will use the same properties from the /portal_skins/custom or /portal_skins/CalendarX folders.

To make each calendar different:

IN ZMI: CUT all the customized objects that you need from the /portal_skins/custom folder, and PASTE them into your CalendarX folder. Now your calendar can be customized locally and independently from any other calendars. I like to use this approach anyway, even if I'm only using ONE CalendarX instance in my Plone site, because it cleans up the /custom folder for other uses.

D. If you want to use Subcalendars, move your customized objects out of the /portal_skins/custom folder and put them directly into your /calendar folder (using the ZMI). You will need a copy of CX_props_subcalendar in each of your MAIN and SUBcalendar folders... and you will very very likely need CX_props_calendar in each of them as well. There's a detailed section later in the manual that provides an example of using Subcalendars (Part V).

E. READ UP.

Also read everything you find in the /docs folder of the CalendarX product distribution or in the Manual. And read the source code in the Python scripts. We've added quite a bit of commentary to most of the Python scripts to make it all much more understandable, and more readily customizable.

II. Tips.

1. Slots/Portlets. For most users, we recommend disabling the slots on your calendar folder, or more specifically, making the slots be empty. If you want portlets appearing in the slots, click on the Properties tab of your calendar folder in the ZMI. Add your portlets into left_slots and right_slots attributes, just like you would in your main Plone site.

Why recommend empty slots? On most browsers, this gives you a lot of extra room for your calendar... CalendarX will expand sideways to take up the extra room if the slots are empty.

1A. I WANT PORTLETS (detail).

If you want portlets appearing in the slots, next to your calendar, use the ZMI

and navigate to your CalendarX folder instance. Click on the Properties tab of your calendar folder in the ZMI. What you need here are two properties called "left_slots" and "right_slots", and they should each be of the "lines" type of properties. In these properties, type in the macro calls that will add portlets in these locations... you can see the examples of how these should look by going to the Properties tab in the root of your Plone site to see how the default ones are set up for your whole Plone portal.

1B. I DON'T WANT ANY PORTLETS TO SHOW (detail)

If you want a full width CalendarX, then you want EMPTY left_slots and right_slots. Follow the instructions above, and make sure that your slots properties are empty. This tells Plone to collapse the left and right columns, allowing CalendarX to expand fully left and right... or nearly fully. Still seems to be a little bit of space reserved on the right side that I haven't explored how to collapse completely. But pretty good.

2. Categories for the Calendar

The calendar uses categories based on the default Event Subjects. Currently these are Appointment, Convention, Meeting, Work, Social Event. You can change these in the portal_metadata/Elements/Subject. If you do this, CalendarX needs to be told about it... this will be fixed in a future version to be automatic. Actually, it is now semi-automatic. If you want CalendarX to choose Subjects based on all the Subjects available, see the instructions in CX_props_calendar_text.txt

So, find the CX_props_calendar sheet, and list the Subjects line by line in the "listOfSubjects" attribute, just like you made them in portal_metadata. And then you're done.

Actually, there's more. You may want/need to change the CSS classes that go along with each category (subject). For this you'll need to customize the calendar.css file. See more about this in Tips #10 below.

Actually, now there's even more. If you have used very LONG subject names, like "French Connection United Kingdom", there is an option to create shorter "nicknames" for each of your subjects, and show them at the top of the calendar instead of the LONG names. For example, you could use the much shorter "FCUK" instead of the long, full name of the Subject used in the earlier example.

And now there's even more! You can make groupings of subjects, using the nicknames approach. First, in the 'listOfSubjects' attribute, list your subject groupings with commas like this:

Mrs Wilson 3rd Grade,Mr Smith 3rd Grade
Mrs Farber 4th Grade,Mrs Jasper 4th Grade
Mr Spinky 5th Grade,Mr Zurven 5th Grade
Field Hockey,Jump Rope,Basketball,Soccer,Chess Club

Then in the 'listOfSubjectTitles' attribute, put the nicknames in that will be shown in the Subject Bar, like so:

3rd Grade
4th Grade
5th Grade
Sports

Now when a user clicks 'Sports', events from all the sports listed in the corresponding listOfSubjects line will be returned in a calendar. This is usually much preferred to listing all 11 Subjects in the Subject Bar of CalendarX.

Also, this functionality makes subcalendar usage very powerful, as you will see if you keep reading this drivel.

3. Basic: Access to your calendar for your visitors

First, publish your /calendar folder from within Plone so that it becomes available on the portlet_navigation. Or not, if you don't want it to show up there.

Second, another nice hook is to create a portal_tab to go to your new Calendar (these instructions are for Plone 2.0.5).

A. In the ZMI, go to portal actions.

B. Add a new action as follows:

Name: Calendar
Id: calendar
Action: string:\${portal_url}/calendar
Condition: (empty or whatever)
Permission: View
Category: portal_tabs
Visible? (checked)

This will display your defaultView (whichever view you have selected in the defaultView property in CX_props_calendar). Default is 'month'.

4. Multiple Calendars.

Now you can have multiple calendars on your site. Place another CalendarX instance wherever you want the calendar to appear. Chances are that if you have multiple calendars, each one will be configured differently. In that case, simply customizing a property sheet won't be enough, because portal_skins/custom can't have two different property sheets both named the same thing. SO... go into the ZMI and cut any CalendarX property sheets, scripts, icons or templates out of portal_skins/custom, and paste them directly into your CalendarX folder instance. Now these customized sheets will ONLY be found by the desired calendar instance, and

you can have as many independent, unique calendars as you wish.

5. Restricting events on your calendars.

A common request is to restrict the types of events shown on a calendar. There are now several ways to do this without customizing the calendar views or macros, by using attributes in `CX_props_calendar` properties.

5a. Subject restrictions.

If you want to display ONLY events that contain certain subjects, you can use the "restrictToThisListOfSubjects" attribute along with the "listOfSubjects" attribute. Read the directions, and simply list the Subjects that you wish to show. For example, this might be a way to have a Music Events calendar on your site, with multiple types of Music events, as well as an Arts Events calendar, with its own set of Arts event categories.

5b. Event Type restrictions.

If you want to display ONLY events of a certain "portal_type", use the "restrictToThisListOfTypes" attribute along with the "listOfTypes" attribute. A "portal_type" is the type of Plone object that you are using for your events. For example, you can go to `portal_types` in the root of your Plone site, copy and paste the existing Event type, and modify it so that its type is a "Staff Event". Then you can create a calendar that will ONLY show Staff Events, just for Staff usage. A good idea is to add a `getNotAddableTypes.py` script in your site that will restrict usage of the "Staff Event" type to just members of your "Staff". See howtos on the Plone.org website for more instructions on how to use `getNotAddableTypes`.

5c. Path restrictions.

If you want to display ONLY events that are found within certain folders of your Plone site, you can use the "restrictToThisListOfPaths" attribute along with the "listOfPaths" attribute. Read the directions, and simply list the full paths (as found in the Path index in the `portal_catalog`) for the folder objects you wish to use. Then ONLY those events found within those folders will display on your calendar.

6. Look at all the calendar properties, and read about them in the `/docs` folder. There are lots of things you can adjust, and they're all documented. Try them and see what you like.

7. Allow Visible Events: There's an attribute in calendar properties that lets 'visible' as well as 'published' events show up on the calendar.

8. CSS made easy: Nearly all the most important CSS attributes are now adjustable from the CSS properties sheet. Font sizes are all expressed in

percentages so that they can vary the way the rest of Plone can, with simple stylesheet changes.

9. "Add New Event" link: A new addition to 0.4 branch is the ability to put a link in the Subject Bar that allows users to click on it and go to an appropriate folder where they can add a new Event. There is a property sheet that controls this behavior (i.e., where the link goes and who gets to see the link at all). In brief, the macro controlling this link first checks to see that the user is Authenticated on the site (few sites allow un-Authenticated users to add Events, and those who do can easily disable this in the macro code). The link then can be set to the following possibilities:

- A. Link to the Member's personal folder.
- B. Link to a specified folder.
- C. Link for certain users to go to different folders (one specified folder per specified user).
- D. Link for users with specific Roles to go to different folders (one specified folder per specified Role).
- E. Link to a specified subfolder within the Member's folder.

If choice C or D is selected, and the current users is not listed among the possible choices (for listed users or Roles), then the link will roll back to choice B or A, if either of these has been checked. If the user is not found in C or D and neither B nor A has been checked, then the "Add New Event" link will NOT be displayed for this user.

NOTE:WARNING Be sure that you configure this option correctly for the way your portal is set up for your users. For example, if you use ROLES to determine where the Events go, then be sure your users have those roles (duh). Misconfiguring this can lead to confusing calendar behavior for your users, and you don't want that.

10. New Icons and CSS classes, selected by Event Subject. Now your calendar can really show its colors by letting you highlight each event on your calendar in color and with icons depending on the Subject. Simply check the appropriate box and follow the examples, listing a single line for each of your Subjects that tells it what icon to use. Add those icons to your /custom folder, or /calendar instance folder, and your calendar will take on a whole new look. Read the details in the CX_props_eventdisplays.txt file in the /docs folder. Or just check the box and try it with the icons included in the default settings.

You'll also need to add special CSS classes for your subjects. The default ones are in calendar.css... look at them to see how to do it for yourself. There are currently two classes for each subject: one class for the link shown within the calendar proper, and one class for the Subject

bar up at the top of the calendar page.

Also: Icons and CSS by Event Type. Same as above, but works for the specified Event types listed. Should come in handy for tricky subcalendars.

11. SPEED TIPS. CalendarX is a busy product, doing lots of queries and lookups in order to display a full-featured calendar. Here are two speed tips to make sure that CalendarX is running as fast as reasonable. The first one is imperative, the second is definitely optional. Neither have been rigorously benchmarked, but informal benchmarks give the first one a potential speed up of as much as five-fold (in Plone 2.0; this whole discussion is less critical for Plone 2.1.x). SO DO #1, if it's not already done for you. It's quite painless.

Speed Tip #1: The skins of Plone create a performance hit to some extent. This tip speeds up the calendar by putting it near the top of the skins list of layers. Moving it up from the bottom of the skin layers list can increase speed severalfold. Go to `portal_skins` in your Plone site, and click on the Properties tab. Then find CalendarX in the listing for your plone skin (usually Plone Default or Plone Tableless) and move CalendarX up to the second spot, right underneath "custom". That's it. Now your calendar skin will be found very quickly.

Note: CalendarX tries to install up at the top below "custom" at installation. So this is probably already done for you. But if you install other products AFTER CalendarX, they may install ahead of CalendarX in your skins. So you may need to go back and move CalendarX up in the layers at some future time.

Speed Tip #2: Customize the query methods from `/portal_skins/CalendarX` folder and move them locally to the location of the calendar instance itself... no more skin-inefficiency, because they are found locally. This should help, but I have no real benchmarks to prove it. In order to really work, you should move virtually all the scripts and property sheets to your CalendarX instance. This should definitely be faster than having your scripts all in the skin layers hierarchy, but probably not much of a boost over Speed Tip #1 and it is more work, so don't bother.

I'll keep looking for other tips and speed ups. I made some code changes in 0.6 branch to speed up the page renderings over the 0.4 branch. The 0.5 branch already contains most of these changes. Also, Plone 2.1.x runs CalendarX faster than Plone 2.0, without bothering with these tweaks... these may help too, but do upgrade if you haven't already.

Part IV: A guide for Developers: Modifying functionality in CalendarX

How CalendarX works -- a brief guided tour.

Contents:

The main folder, and what it does.

The property sheets.

The python scripts.

The page templates.

The macros.

The CSS and Javascript files.

A. The main folder, and what it does. -- For now, read Part III for Administrators over again.

In brief, CalendarX is a folderish Zope product.

Each view (month, day, etc) is a page template which can be located any of these places:

1. In the /portal_skins/CalendarX folder
2. In the /portal_skins/custom folder
3. In the filesystem /zopeinstancehome/Products/CalendarX/skins/CalendarX folder
4. In the (portal_root)/calendarxinstance folder.
5. In the portal_root folder (not recommended, but it works).

Each view template is structured similarly: it queries the portal_catalog for appropriate content objects and displays them as a calendar-style table. The code for doing these tasks is complex and repetitive, so a great deal of the code has been removed from the templates and aggregated as macros (for repetitive page template code, stored in the CX_props_macros.pt page template) or as Python scripts (for logic as Python code).

In order to make it easy to customize the calendar, many options have been built into the templates and are controlled by attributes on the property sheets (names that start with CX_props_ ...). These property sheets can be customized by the developer in the same places that the templates can be customized (1-4 above).

ALL the scripts and macros and templates in CalendarX can be customized. So can the CSS and Javascript code. You can go deeper and get into the filesystem Python code of the product itself, but there is not much that you'd want to do there... I've kept it all in the filesystem as easily accessed scripts instead. There are plusses and minuses to this approach, but so far CalendarX has remained fairly easy to fix when it breaks and readily customized for most purposes.

B. The property sheets. -- For now, read Appendix B.

One thing to note is that there is a monkey patch in the CalendarX Product code that fixes a Zope bug/omission for FSPropertySheets. In short, the Zope code never allowed for a FSPropertySheet to have lines and text properties... you could add such properties after the FSPropertySheet was working, but there was no mechanism to put a "lines" property in the FSPropertySheet on the filesystem and have it read into Zope properly. My monkey patch fixes that.

C. The python scripts. -- For now, read Appendix C.

D. The page templates. -- For now, read Appendix D.

E. The macros. -- Also covered (briefly) in Appendix B.

F. The CSS and Javascript files. -- For now, the miniscule Appendix E must suffice.

Part V: Use Case: Creating a Resource Scheduler with CalendarX

NOTE: updated for both 0.4 and 0.6 branch. Previous versions had incomplete instructions regarding use of "listOfSubjects" property, which didn't properly work. Follow all the instructions here for your first time through. THEN to make your own resource scheduling calendar, go back and make a copy of it and start making modifications. That's my recommendation, YMMV. I tried following these step-by-step for CalendarX-0.6.5(stable) on Plone 2.1.1 and they work.

A common application for calendar applications is the scheduling and reservations of available equipment, rooms, and facilities for an organization. CalendarX can handle the basic functionality of a Resource Scheduling application through the use of subcalendars. Here we will go through the steps you need to create a Resource Scheduling calendar application for an environmental group that has a number of Beamers (LCD projectors), Bicycles, Cameras, Laptops, and Rooms at their headquarters that are made available to their members on a first-come-first-served basis.

The calendar will be organized into one Main calendar with several nested Subcalendars. We'll name the Main calendar "Resources", and we'll name each of the Subcalendars for its primary resource: Beamers, Bicycles, Cameras, Laptops, and Rooms.

Imagine that you are in this organization and you have an important meeting to attend in Antwerp next week. It's a last minute invitation to speak to the executive board of another environmental group, and you'll need to take along with you both a beamer and a laptop. You should be able to quickly examine the calendar to see only the laptop and beamer schedules, so that you know whether you can borrow ones from the group, or if you might have to make other arrangements.

That's an easy one. The Resources calendar will display ALL the resources that are being used on each day, but it will have the ability to allow users to filter the Subjects so that only select resource categories are visible. This is just like the regular Subject filtering done by CalendarX elsewhere, except here our Subjects of interest will be Laptops and Beamers. So you would just check the boxes by Beamers and Laptops (and uncheck the one that says View All), and hit the Refresh button. Now you can see which beamers and laptops are checked out on the days you need them for Antwerp.

Overall, this one view should work fine for most uses. Use the Month view to start, then zoom in to the week or day views. Even with just one Subject category chosen (e.g., Laptops), the calendar would show all the scheduled Laptop reservations, and a quick rollover with the mouse could display which laptops are reserved, etc. But sometimes a more detailed view is desired, or perhaps you ONLY want to see the reserved times for the really nice laptops, the ones with 1 Gb of RAM and the 17" wideview monitors. To accomplish this, you would click on the Subject link for Laptops and you can enter the Laptops subcalendar. Here only laptop reservations are shown, and the Subject bar now lists all the available laptops. To reduce the clutter on the screen, select just the few laptops that you are interested in, and hit the Refresh button to show their availability.

So let's build this resource scheduling calendar. Right now. It should take about an hour or so, if you have a clean Plone portal to work on (as you should). The first step will be to create some custom Event types for the various resources we'll be scheduling. The second step will be to build the Main calendar and configure it, and finally we'll add the subcalendars and configure them. We use custom Event types because it makes it much easier to restrict our Resources calendar to only see the events that we're interested in, and they won't interfere with other calendars in this Plone portal. [Note: It took me an hour and 15 minutes to run through this just now on a fresh Plone 2.1.1 with CalendarX-0.6.5 and rewriting parts of the instructions as I went. So it's pretty easy.]

Step 1. Create New Event Types.

First you need to create content types for each resource event. We do this because restricting CalendarX by event type is a handy way to create a nicely organized resource scheduling calendar. An easy way to get new content types is to "repurpose" the existing Plone Event type (these instructions are for Plone 2.0.5, with any special Plone 2.1.1 notes added in square brackets and asterisks like this: [*2.1*: more words here...]):

- A. In the ZMI for your portal, click on *portal_types*.
- B. Select the checkbox beside the *Event* type. Then hit the COPY button at the bottom of the page, and then hit the PASTE button. This will create a duplicate of the *Event* type called *copy_of_Event*, and we can rename this for our first Resource type.
- C. Select the checkbox next to *copy_of_Event* and then click the RENAME button at the bottom of the page. In the form, give it the new name *Reserve a Beamer*.
- D. Click on the new *Reserve a Beamer* type, which will open up the form for editing it. Now type "Reserve a Beamer" in the form for the *Product meta type* property, which will nicely identify this special Event type in the portal_catalog (this will show up in the catalog metadata as both "portal_type" and as "Type"). [*2.1*: Also, be sure to change the *Title* to match ("Reserve a Beamer") because Title is used elsewhere in Plone to identify these new types unambiguously.] Optionally, you can type in a new description here too ("Use this Event type to Reserve a Beamer on our Resource Scheduling Calendar"). Hit SAVE at the bottom of the form.
- E. Repeat this process: start with the *Reserve a Beamer* type, COPY, PASTE, RENAME several times to create event types for Bicycles, Cameras, Laptops and Rooms. Be sure and edit each one to change its *Product meta type* property, and perhaps the Description as well. Now you have all the Event types you need.

Before we go on, we should also set up the proper subject categories for these new event types. For resource scheduling, the Subject will be the identifiers for the specific resources that will be available to schedule. In this case, we'll add Lumina Zoom 2400L, Toshiba NoZoom 1500L, and Sharp Mini 1200 as our three possible Subjects for the *Reserve a Beamer* event type (we only have three beamers available, and that's what we call them, by their brand names):

F. Three clicks in the ZMI:

1. In the ZMI for your portal, click on *portal_metadata* (one of the Plone tools).
2. Click on the *Elements* tab at the top, to take you to the Update Element Metadata Policies tab.
3. Click on the *Subject(s)* element type in the menu list of metadata elements available.

G. Now you can see a list of Content Types (such as *Event*, usually the only content type here in a stock Plone installation) and a number of options associated with their Subject metadata. Subjects are created specifically for each content type, using this form. Since your content types are new, they aren't yet listed here. Scroll down toward the bottom of this form to the section for adding a new type to the roster... it's an empty form for a **<new type>** (displayed in the gray bar). Use the dropdown list for *Content type* to select your new *Reserve a Beamer* content type, and in the Vocabulary textbox, list your three Subjects, one per line:

Lumina Zoom 2400L
Toshiba NoZoom 1500L
Sharp Mini 1200

(these names are arbitrary. But they are the names that will show up when your users click to add a "Reserve a Beamer" event for the scheduler! So my advice is to choose sensible names that your users will understand. You'll also need to re-enter these names exactly in another place, so remember them. I'll remind you when the time comes.)

Also check the box that says *Enforce vocabulary* (yes). This means that users must choose one of the Beamers when they fill out this *Reserve a Beamer* form. Then click **ADD** button to add these subjects to the metadata. I don't care about the other choices, you can try them if you'd like.

And now your *Reserve a Beamer* type is listed, along with its subjects.

H. Repeat step G for each of the other *Reserve a Resource* types you've created. Give them each a vocabulary of resources available for checkout. For the *Rooms*, list the room numbers or room and building numbers, whatever is appropriate for your users to readily identify each resource. If Camera type or other properties are important for your resources, name your resources accordingly (*Canon Rebel*, *Nikon F*, etc.) so that your users will find it easy to choose the resource that they really want.

Now your new Resource content types are ready. Let's make the calendar.

I. SOMETHING MISSING... Steps above don't quite do it all for a fresh Plone 2.1.1 installation. That's because of the Plone 2.1.1 bug that I wrote about in INSTALL.txt (and in Appendix A of this manual). Read about it, and patch it.

Step 2. Create the Main Resources calendar.

A. Add a CalendarX to your site.

In Plone, add a CalendarX Content the usual way that you add content. I put mine in the

portal root, but you might want yours elsewhere. Name your calendar appropriately (like "Resource Schedule" for the Title and "resources" for the Id [*2.1*: or *Short Name*, as it is called in Plone 2.1. You may have to go back to the Contents view of your portal root to find your new Resources Schedule object and check the box to Rename it, and give it a new *Short Name*.].

It is possible to do this step (create the resources calendar) through the ZMI as well, but is not advisable because officially a CalendarX instance is a type of Plone content, and should be treated as such (ONLY create and delete content through the Plone interface, not the Zope interface). This "resources" calendar will be your MAIN calendar. We'll create your Subcalendars in a bit. For the remainder of these instructions I'll assume that you put your resources calendar in the portal root, and called it "resources" (or sometimes MAIN).

NOTE! You will also want to get rid of the right slots here, which in a default Plone site consists of the Calendar portlet and a few others. Right now, in the ZMI, go to the Properties tab of your "resources" CalendarX instance. Add a new property called "right_slots" and make sure that it is of type "lines" (not the default "string"). Leave it blank, empty, nada. Now when you view this resources calendar, no portlets will appear on the right side.

B. Customize your Resources calendar.

In the ZMI, go to /portal_skins/CalendarX and one at a time, click "customize" on each of these three property sheets: CX_props_calendar, CX_props_popup, and CX_props_subcalendar.

Then in the /portal_skins/custom folder, select all three of them and click the CUT button. Navigate to your new MAIN ("resources") calendar instance and then click the PASTE button, so that all three of these fresh property sheets are now inside your MAIN calendar folder.

NOTE! You want to use CUT and PASTE so that these property sheets are removed from the portal_skins/custom folder and placed in the resources calendar. This way these properties will ONLY be used by the resources calendar and its subcalendars.

NOTE! These aren't Plone content, so you don't (can't) do this through the Plone interface. Using the ZMI is the only way.

C. Set the following configuration settings in this MAIN calendar

(we'll set up the subcalendars afterward). So click on one of the new property sheets in your "resources" folder in the ZMI, and then click on the "Properties" tab to make these changes, for all three property sheets.

MAIN calendar:

config: CX_props_calendar:

1. useMultiSubjects: checked. The subcalendar menuing options

ONLY will work if you have this option selected (and it should already be selected by default).

2. listOfSubjects: List names of the resources as defined in each of the subcalendars, listed *each as a single line with commas*. For example, I'm creating calendars for the following topics: Beamers, Bicycles, Cameras, Laptops, and Rooms. In the Beamers subject setup (part 1G above) we

created entries for Beamer1, Beamer2, Beamer3, etc. So here we want to list all of those. So for this sample here are the five lines to add in listOfSubjects (I'm using the names I added: you use yours):

Lumina Zoom 2400L,Toshiba NoZoom 1500L,Sharp Mini 1200
Trail bike blue,Trail bike red,Bike with large basket,Road bike 10 speed
Nikon F,Canon EOS,Minolta SRT101
Toshiba 1200s,Sony 13mon,Old Dell laptop
Conference 1120,Meeting 1136,Freds office 1148

These are the lists of all subjects used for the topics listed above (Beamers, Bicycles, etc.). They don't show up as the subcalendar names because we give each grouping of subjects a nickname in parts 6 and 7 below. They are comma-separated names: Don't use any extra spaces between the commas and names!

NOTE! Put ALL the names for one group on ONE line, separated by commas. The line may wrap in the textarea in your ZMI form, but it should still be one line PER SUBJECT GROUPING, as above.

NOTE! Use the exact names that you gave your event subjects in Step 1G and 1H above (in the portal_metadata). Otherwise you won't see your events show up properly. This is a common mistake. Just follow the directions.

3. restrictToThisListOfSubjects: (broken before 0.6.5) Optional.

You may check this if you want to restrict your calendar further, but it is unnecessary in this setup because we are restricting the calendar usage to these few event types (below).

4. eventTypes: I'm using five new Event types created by simply repurposing the Event type, with new names and subjects. So list these here, ONE PER LINE:

Reserve a Beamer
Reserve a Bicycle
Reserve a Camera
Reserve a Laptop
Reserve a Room

5. restrictToThisListOfTypes: checked, because I want this calendar to ONLY show these special reservation events, not regular events.

6. useCalendarHelp: checked. I think calendar help will be needed because subcalendars may present conceptual problems for some users.

7. listOfSubjectTitles: names of the subcalendars I'm using: Beamers, Bicycles, Cameras, Laptops, Rooms. These names will take the place of the comma-separated list of subjects that we completed above, displayed in the calendar header.

8. useSubjectTitles: checked, so that these nicknames in #7 will be used.

9. includeReviewStateVisible: checked. This means that events will show in the resources calendar as soon as they are added. I'm using this on the demo resources calendar on CalendarX, but you can choose for your site whether this is a good idea or not.

Save your changes before going on.

MAIN calendar:

config: CX_props_subcalendar:

1. useSubCalendarSubjectMenu: checked, because you need that to properly navigate to your subcalendars.

2. listOfSubCalendars: fill this with the list of IDs for the subcalendars. Here this is: beamers, bicycles, cameras, laptops, and rooms. [*2.1*: in Plone 2.1.x these are called Short Names instead of IDs, remember?]

3. isSubCalendar - nope, this is the MAIN calendar, not a subcalendar.

4. nameOfSubCalendar - nope.

MAIN calendar:

config: CX_props_popup:

1. Go crazy. Expanded the extent of the popup coverage by checking several options, including: showPOPTitle, Type, Subject, Start, End, Creator, Description, etc. Or not. This is optional at this point, but I'm adding it here so that you try it (and there's a point to make with this later on in this tutorial). Try things with CalendarX, because you can learn a lot just trying things out.

Step 4. In the ZMI, inside the MAIN calendar, add a new "CalendarX Content" instance using the normal Zope way to add objects (a dropdown, next to the button called "Add"), and select CalendarX (the only option) and name this one what you want for the Id of one of your Subcalendars. For this example, name this first one "beamers" for the Id. Eventually, we'll be adding 4 more of these subcalendars (named as above: bicycles, cameras, etc.).

TECH NOTE: Normally you add a new CalendarX instance from Plone, not from the ZMI. But CalendarX tries to prevent normal use of the calendar as a folder from within Plone. So we use the ZMI to add the subcalendars, and it works fine and is easy. It does

mean that the subcalendars are probably not going to be indexed in your portal_catalog, but they don't really need to be indexed there under normal circumstances (events are not stored here, and events will be cataloged normally), and if you do have some reason why the subcalendars need to be indexed, then you can reindex your catalog so that they are.

Step 5. Go to /portal_skins/CalendarX and hit customize on each of these two property sheets: CX_props_calendar and CX_props_subcalendar. Then in the /portal_skins/custom folder, select both of them and click the CUT button. Navigate to your new Subcalendar instance (/resources/beamers) and then click the PASTE button, so that these two fresh property sheets are now inside your Subcalendar (beamers) folder.

Step 6. Set the following configuration settings in this Subcalendar (we'll set up the other subcalendars you want afterward)...

SUB calendar:

config: CX_props_calendar:

- 1. useMultiSubjects: checked.** The cool subcalendar menuing options ONLY will work with this option selected.
- 2. listOfSubjects:** names of the subjects desired for this subcalendar.
For example, in the Beamers subcalendar I'm using:

Lumina Zoom 2400L
Toshiba NoZoom 1500L
Sharp Mini 1200

There are the names you used earlier, the same ones listed all in ONE LINE with commas back in the MAIN calendar CX_props_calendar property sheet. Now they are on separate lines, so that you'll be able to examine each one separately on this resources scheduler.

- 3. eventType:** I list the one type of Event that I want to show in this subcalendar because I'm restricting my events this way: for my Beamer subcalendar, this uses: *Reserve a Beamer*.
- 4. restrictToThisListOfTypes: checked,** because I want this calendar to ONLY show these special reservations for bicycle events.
- 5. useCalendarHelp: checked.** I think calendar help will be needed because subcalendars may present conceptual problems for some users.
6. Any other properties that you checked in the main calendar should also be set here with the same properties. You may have good reason for using different property values here, and that's ok too, but probably mostly these should be the same.

7. We DON'T check the useSubjectTitles here, because the Subjects themselves are what we want to display for our users.

SUB calendar:

config: CX_props_subcalendar:

1. **useSubCalendarSubjectMenu: UNchecked.** ONLY for main calendars.
2. **listOfSubCalendars: Unused.** Empty. ONLY for main calendars.
3. **isSubCalendar - Checked.** Yes, because this is a subcalendar.
4. **nameOfSubCalendar** - A name (Beamers, or Bicycles, etc.), which will be displayed in the Subject choices header on your subcalendar. If you do not choose a name here, the default will be the ID of the subcalendar, changed to ALLCAPITALS.

Step 6b. Note that we did NOT include a CX_props_popup in the subcalendar.

That's because we want the Subcalendar behavior to inherit the values of the MAIN calendar for these popups. ONLY add property sheets into a subcalendar if you DON'T want to inherit the values of the properties from down in the MAIN resources calendar. Or, to put it another way, ONLY add property sheets in a subcalendar in order to override the acquisition from the MAIN calendar.

Step 7. In the ZMI, go back to the MAIN calendar. Now we're going to save some time by duplicating the beamers calendar four times to make our other subcalendars. Select the checkbox for your Subcalendar that you just created (beamers) , and click the COPY button. Then hit the PASTE button four times, to create four copied subcalendars. Then RENAME each one of these pasted subcalendars (bicycles, cameras, etc.) Each one will already have the property sheets inside them, although they each need some configuration, as follows (leave other choices as they were set for the first subcalendar):

SUB calendar: (each of bicycles, cameras, etc.)

config: CX_props_calendar:

1. **useMultiSubjects: checked.** LEAVE THIS ALONE
2. **listOfSubjects:** names of the subjects desired for each of these additional subcalendars.
For example, in the Cameras subcalendar I'm using: Canon EOS, Nikon F, etc. Whatever you called them in the MAIN calendar configs.
3. **eventTypes:** I list the one type of Event that I want to show in this subcalendar because I'm restricting my events this way: for my Camera subcalendar, this uses: Reserve a Camera.

4. restrictToThisListOfTypes: checked. LEAVE THIS ALONE

5. useCalendarHelp: checked. LEAVE THIS ALONE

6. Any other properties... LEAVE THESE ALONE

SUB calendar:

config: CX_props_subcalendar:

1. useSubCalendarSubjectMenu: UNchecked. LEAVE THIS ALONE

2. listOfSubCalendars: Unused. LEAVE THIS ALONE

3. isSubCalendar - Checked. LEAVE THIS ALONE

4. nameOfSubCalendar - A name (ex. Our Cameras). Again, this name will only be displayed in the Subject choices header on your subcalendar.

Step 7b. You make those changes for EACH of the subcalendars.

Step 8. That's it, you're done.

Step 9. Go add some of your new resource scheduling events. See them on your calendar. You'll need to publish them, of course. That's fairly comparable in workflow terms to asking permission to schedule a given resource, so I think that's a reasonable default setup. Alternatively, you can check the `includeReviewStateVisible` property in `CX_props_calendar`, and then all entries will be shown as soon as they are entered in the Visible state (as described in step #9 above).

Optionally, you can use some of the other property sheets to tweak your calendar further. For example, you probably will want to skin the calendar's CSS properties so that the colors and fonts match the rest of your website. So when you do Step 2B above (or later, after you've thought about it), repeat Step 2B for the `CX_props_css` and `CX_props_addeventlink` property sheets into the Main calendar folder, and configure them appropriately. You won't need to copy them into the subcalendar folders unless you want different CSS options for the subcalendars... I can see when this might provide a nice touch, but not on most calendars.

A good reason to use the `CX_props_addeventlink` property sheet here is if you want your Users to put all their *Reserve a Resource* events into one folder. That might make more sense for some organizations, and will certainly make cleanup of old Resource events easier for a calendar Administrator. In that case, configure the *Add New Event* link to direct all users into a single Plone folder where events will be stored. Make sure that the folder has permissions set properly so that all appropriate Users can add events there.

That's all. Now go and add lots of events and see how it all works. See how the navigation works, and allows you to filter according to the types of resources you may want, or lets you drill down and only look at one resource at a time in the subcalendars.

Go crazy and dream up new uses for subcalendars. I think it should work great for school calendars, and I'll probably be setting one up that way soon. Let me know what great calendars with subcalendars you create. Go to CalendarX.org and add a *CalendarX Sighting* document with a link to your CalendarX calendar so the community can see what you've done.

Step 10. If anything went wrong... A common mistake is to not have the same properties checked in your `CX_props_calendar` sheet in the subcalendars that you do in the main resources calendar. For example, if you want Visible events to show in all of them, then you really need to have that selected in the `CX_props_calendar` property sheet in ALL the subcalendars, as well as in the MAIN calendar. And if you're still broken, it's probably because you went too fast and tried to make changes along the way, playing instead of following the directions... that's ok, I don't mind. But before you write me to say "Hey, it doesn't work," go back and do it again from scratch and follow the directions to the letter. That'll teach you to mess around in MY class.

Another common problem: You go to view your new Resources calendar and instead you get an error. Two errors that sometimes occur due to a misconfiguration are these:

TypeError: iteration over non-sequence (happens in one of the query scripts).
IndexError: tuple index out of range (happens in one of the macros)

I've trapped most of these, but sometimes a messed up configuration in the calendar or subcalendar property sheets will throw one of these errors. Both of these tend to result from not having a proper match between the number of subcalendars with the number of Subject listings, or other properties where CalendarX needs to match lines properties with each other. In short, take your time and make sure that everything is configured correctly and try again.

Appendix A: Installing CalendarX for an existing Plone portal

I am not going to tell you how to install Plone (or Python or Zope) here. I have opinions on that but I'm not going to tell you them here. So I am assuming that you have a reasonably current (version 2.0.x or 2.1.x) Plone installation, and that's why you want CalendarX. If you have not used Plone before and you just want to try CalendarX out, then I'd recommend getting one of the easy Plone installers (the binary Plone installer for Windows machines works alright for me) and follow the instructions first, and then come back here.

I. Software requirements (Updated for 0.6.5)

II. Instructions for Installation for the first time. (0.6.5)

III. i18n: Internationalization instructions. WE NEED TRANSLATIONS! (0.6.5)

IV. Documentation available.

V. Caveats.

Old Instructions for upgrading to new versions of CalendarX

(MOVED these instructions to MIGRATE.txt)

Plone BUGFIXES: There are two known Plone bugs that should be fixed in order for everything to work right in CalendarX... one in Plone 2.0.5 and another in Plone 2.1.1. Details are below.

I. Software requirements

!!! DON'T USE Python 2.4 !!! Well, ok, it does run on Python 2.4 but there have been some peculiar problems, especially when a file accidentally had DOS carriage returns in it. Talk about picky! Oh well.

Software Requirements for CalendarX (0.6 branch):

1. Plone 2.0+ (also tested working in 2.1.1).
2. Python 2.2 or 2.3, NOT 2.4.

Software Suggestions (NOT REQUIRED):

1. Zope 2.7.x+ (may work on earlier Zopes, but untested. Python 2.2+ is not standard until Zope 2.7+)
2. AdvancedQuery [this is available at Dieter's site:
<http://www.dieter.handshake.de/pyprojects/zope/>]
AdvancedQuery is one of two query techniques used. By default, the traditional ZCatalog query method is used, but using AQ is simply a switch option in the skins property sheet.

II. Instructions for Installation for the first time.

NOTE these instructions were written for Plone 2.0.5. They still work roughly the same in Plone 2.1.1, but there may be small differences. I've only just begun working in Plone 2.1.1 myself and haven't a lot of experience here yet. But it does work, no fear about that.

1. Make sure you have the required software and additional Zope Products in place. Additionally, there may be bugfixes you need to apply to your Plone install. Here are the two I'm aware of, one for Plone 2.0 and one for Plone 2.1.

#NOTE: IMPORTANT BUGFIX TO Plone version 2.0.5#

There is a bug in Plone 2.0.5 that will bite you if you change your Events from 24 hour clock to the 12am,pm style. The bug is documented here on the Plone.org website:

<http://trac.plone.org/plone/ticket/3641> or
<http://plone.org/collector/3641> (older link, reroutes to above)

and is also described on the CalendarX website here:

<http://calendarx.org/issues/talkback/1104958175>

You can patch the bug either on the filesystem, or through the ZMI in your portal_skins/custom folder.

#NOTE: IMPORTANT BUGFIX TO Plone version 2.1.1#

There is a bug in ATContentTypes, event.py distributed with Plone 2.1.1 that forces the Event to keep the Event vocabulary, even if you have repurposed an Event in portal_types and given it a new Type name. The bug is documented here on the Plone.org website:

<http://trac.plone.org/plone/ticket/5307>

and also is described on the CalendarX website here:

<http://calendarx.org/issues/talkback/1135185404>

and must be patched on the filesystem. In brief:

In YourZopeInstance/Products/ATContentType/content/event.py
change the definition of 'getEventTypes' from

```
metatool = getToolByName(self, "portal_metadata")
events = metatool.listAllowedSubjects(content_type = "Event")
return events
```

to

```
metatool = getToolByName(self, "portal_metadata")
my_type = self.getPortalTypeName()
events = metatool.listAllowedSubjects(content_type = my_type)
```

return events

"Event" was hardcoded into the event.py, so that even repurposing it failed to get the Vocabulary info from portal_metadata, and there may be other side effects. FAILURE to fix this one means that the Resources Calendar exercise in the Manual will NOT work properly.

2. Acquire the CalendarX-0.6.x.tar.gz file from sourceforge. This unzips as a tar file, which untars as a single folder, called CalendarX-0.6.x.

3. RENAME! In your Zope INSTANCE, place this folder in your INSTANCE_HOME/Products/ folder and rename the folder to /Products/CalendarX instead of /Products/CalendarX-0.6.x.

4. Restart Zope.

Note: If you are Reinstalling CalendarX, read MIGRATE.txt in the docs. There's really nothing to do if you haven't customized it at all, but nearly everyone does some customizing, at least for the property sheets.

5. Go to your Plone portal, and log in as a manager. Go to plone_setup and Add New Products. Find CalendarX in the list and check it, and click the Install button. This is equivalent to navigating in the ZMI to your Plone instance, and using the portal_quickinstaller to install CalendarX.

6. IN PLONE: Navigate to where you want a CalendarX instance, and add a CalendarX in the normal fashion. Name it "calendar" or whatever you want. It creates an "empty" folder, but when you now navigate to it in Plone, it will display the Calendar, set to the default view (Month view).

NOTE: CalendarX is *not* an ordinary Plone folder. The folder_listing and folder_contents have been disabled. Do attempt to store Plone content inside your CalendarX instance. Certain advanced features of CalendarX require you to use this folderish behavior to store CalendarX scripts and other files within it (such as having multiple calendars on one site, and the use of subcalendars). See step #9 below for more information.

7. That's it. Navigate in Plone to your calendar folder and the default (month) view should appear. You may then publish (or not) your calendar as you wish.

8. To customize your CalendarX, go to portal_skins/CalendarX, and customize any of the files there (property sheets, scripts, page templates, etc.). For more information on customizing CalendarX, read the CUSTOM.txt file

in the /docs folder, and each of the separate text documents that describe the attributes in each property sheet.

9. To create more than one instance of CalendarX is easy... just add another one from the dropdown list. However, all your calendars will use the same properties from the /portal_skins/custom or /portal_skins/CalendarX folders.

To make each calendar different:

IN ZMI: CUT all the customized objects that you need from the /portal_skins/custom folder, and PASTE them into your CalendarX folder. Now your calendar can be customized locally and independently from any other calendars. I like to use this approach anyway, even if I'm only using ONE CalendarX instance in my Plone site, because it cleans up the /custom folder for other uses. To further customize it, find the object in /portal_skins/CalendarX that you want, hit the Customize button, then CUT/PASTE it into your CalendarX instance, and then make your changes. Only the objects that you want to customize need be CUT/PASTED to your CalendarX instance. All the other ones needed will be acquired from the /portal_skins/CalendarX folder in a normal fashion.

III. i18n: Internationalization instructions. WE NEED TRANSLATIONS!

CalendarX-0.6 branch has i18n capability. There are now eleven translations (as of 0.6.4 release). If you can help, please do! Here's how:

- 1. Go to the i18n folder.** This is where the translations are stored.
- 2. Choose one of the translated files as a starting point;** any of them will do for getting started.
- 3. Copy the file and rename it:** here are the translations that are standard in Plone that we don't have yet, and would LOVE to have for CalendarX so that we're compatible in all these. And we would love to have even more than these... as many as we have languages for folks using CalendarX! So copy one of the translations we have, and translate all the terms into your language of choice.

Thirty-some translations and status:

calendarx-af.po	- needed
calendarx-ar.po	- needed
calendarx-bg.po	- needed
calendarx-ca.po	- FINISHED
calendarx-cs.po	- FINISHED
calendarx-da.po	- FINISHED
calendarx-de.po	- FINISHED
calendarx-el.po	- needed

calendarx-eo.po - needed
calendarx-es.po - FINISHED
calendarx-es-ar.po - needed
calendarx-es-es.po - needed
calendarx-et.po - needed
calendarx-eu.po - needed
calendarx-fa.po - needed
calendarx-fi.po - needed
calendarx-fr.po - FINISHED
calendarx-he.po - needed
calendarx-hr.po - needed
calendarx-hu.po - needed
calendarx-hy.po - needed
calendarx-it.po - FINISHED
calendarx-ja.po - FINISHED
calendarx-ka.po - needed
calendarx-ko.po - needed
calendarx-lt.po - needed
calendarx-nl.po - FINISHED
calendarx-nn.po - needed
calendarx-no.po - needed
calendarx-pl.po - needed
calendarx-pt.po - needed
calendarx-pt-br.po - FINISHED
calendarx-ro.po - needed
calendarx-ru.po - needed
calendarx-sv.po - needed
calendarx-tr.po - needed
calendarx-uk.po - needed
calendarx-zh.po - needed
calendarx-zh-cn.po - needed
calendarx-zh-hk.po - needed
calendarx-zh-tw.po - needed

Other language .po files gladly accepted! This is the list that is present in Plone 2.0.5. If more will be available for Plone 2.1, let's get those all in here too.

4. Translate all the strings and the few date and time formatting strings that are present. Also, if your translation should serve as a fallback translation, include those translations that should fallback here... follow the other calendarx-xx.po files or (really) the examples in CMFPlone/i18n for a guideline.

5. Test your translation: put it in your /Products/CalendarX/i18n folder, and restart your Zope. Actually RESTART your Zope, don't just refresh the

CalendarX product -- that's not enough, because the i18n initialization takes place at Zope startup. After the initial restart, you can test your new translations or modifications you make by going in your ZMI to the Control Panel, select PlacelessTranslationService, find the po file you're working on click it and hit the Refresh Catalog button to update the translation catalog with your new work. Then email your translated po files to lupa.

6. Check the results in your browser. Set your browser's default language for your desired translation and see the difference. Also, if desired, you can download PloneLanguageTool and install it, and use it to make a language switcher in your site... that makes it easy to test different translations and see what they look like. I use PLT on the Crashtest site where you can try out CalendarX-0.6.x translations.

7. Pay particular attention to getting the formatting strings localized... do you want to say "April 2005" or "2005 Avril"? There are more lots of these formatting strings... make them the way YOU think your users visiting in your language translation would be comfortable seeing these strings. Then be sure to test these all out.

IV. Documentation available:

Contents of the CalendarX.0.6.x /docs folder:

- CREDITS.txt - some people who deserve our thanks.
- CUSTOM.txt - description of how to customize CalendarX with skins.
- CX_props_calendar_text.txt - instructions for the calendar properties
- CX_props_css_text.txt - instructions for the CSS properties
- CX_props_custom_text.txt - instructions for the Custom property sheet
- CX_props_popup_text.txt - instructions for the popup properties
- CX_props_addeventlink_text.txt - how to control the Add New Event link
- CX_props_subcalendar_text.txt - instructions for subcalendar properties
- HISTORY.txt - all the changes in this CalendarX branch so far.
- INSTALL.txt - this file, describing the installation.
- LICENSE.txt - brief synopsis licensing for CalendarX (GPL).
- LICENSE.GPL - a current copy of the General Public License.
- MIGRATE.txt - a quick how-to on migrating/upgrading CalendarX.
- OVERVIEW.txt - synopsis of what CalendarX is, and does.
- README.txt - brief message identifying CalendarX.
- TODO.txt - some things to finish before this branch becomes (stable).
- VERSIONS.txt - explains version numbers used for CalendarX releases.
- CalendarXManual-065-draft06.pdf - a manual with all the goodies from these documents, and some more besides.

V. Caveats.

0. These caveats have been included here since the first releases of CalendarX and I've never had excuse to need them. But here they are.
1. For your health, and mine, I've tried to document nearly everything you need to accomplish significant things with CalendarX. Please read all the documentation before you ask questions to the list or at CalendarX.org. There's lots of it, even to the point of explaining each configurable attribute in the property sheets. It is not, however, capable of explaining all the possible things you can do with CalendarX. Be creative, and share your successes and failures with us at CalendarX.org.
2. Part of the code for CalendarX at this point in time comes from the previous products it builds on -- mainly PloneCalendar (see the README). I know most of that code now, and have replaced large parts of it with my own. But I can't vouch for everything it does yet. Do with this product what you will, but use it responsibly.
3. At this point, I have no idea how CalendarX may interact with or interfere with any other Zope/Plone products or other software. There are likely to be name overlaps between objects in this code and those in other apps, and in a world with Spammish Acquisition, many things are possible. I have seen no bad such interactions, although one potential user claimed to be totally unable to import the Fezzik folder (on the 0.2 branch) no matter how he tried (a month later, this user wrote back and said there are no problems now with the most recent version). I think this must have been an interaction with this particular Plone setup, but I have no idea. I've tried it without any problems on several systems and operating systems, but I don't know what will happen on your site, because you may have customizations that do cause problems. Forewarned should be four armed, but you never know. In the meantime, do with this product what you will, and remember to use it responsibly.
4. This product has not been thoroughly tested and is likely to have bugs. In fact, it doesn't even have a unit test suite for it yet. You have been warned. That said, I and many others use this product routinely in production Plone sites, and deliver versions of it to clients for use on their production sites. Do with this product what you will, but use it responsibly. Practice safe Plone.
5. No animals were harmed in the creation of this Product, and no testing on live animals was performed. You should not attempt to use this Product in the manufacture or assembly of weapons of mass dysfunction, as it is likely to function properly, thus causing failure of intended weapons. Any attempt to disassemble or reverse engineer the functionality of this Product is

strictly encouraged, and any successes in such endeavor should be shouted from the mountaintops. Or at least let me know.

VI. Upgrading versions of CalendarX.

- A. Upgrading from 0.6 branch versions of CalendarX (updated 0.6.4)
- B. Upgrading from 0.4 branch versions of CalendarX (updated 0.6.4)

A. Upgrading to 0.6.x from 0.6 branch versions of CalendarX.

1. Replace the old version of CalendarX in your INSTANCE_HOME/Products folder with the new version of CalendarX (0.6 or higher).
2. Restart Zope. DON'T JUST REFRESH IT, I don't think, because there are probably changes in the i18n folder, and without Restarting Zope, these changes won't be picked up properly. Or try it: ZMI, Control Panel, Product Management, CalendarX, and in the Refresh tab, click Refresh. Or in Plone (2.0+), go to Plone Setup, Add/Remove Products, and under "Installed Products", look for CalendarX and click the link that says "This product has been upgraded, click here to reinstall." BUT I THINK YOU REALLY NEED TO RESTART ZOPE. Or you can try refreshing the ... no, JUST RESTART ZOPE.

This is important! If you just refresh the CalendarX product in your quickinstaller, you will NOT get the updated translation files indexed properly. Alternatively, if you really don't want to restart Zope, after refreshing the CalendarX product you must go in the ZMI to Control Panel, Placeless Translation Service, and find the several translation (po) files for CalendarX... then click on each one of them and then click the Reindex button for it. See? I told you it was easier to just restart your Zope!

3. Any parts of CalendarX that you have customized should be compared with fresh versions (and the HISTORY.txt file) to see if there are changes that you should be aware of in your customizations. If all you have changed are the CX_props_XXX property sheets, then there is little chance of any problems.

TIP: Just do it. Make a new instance of CalendarX, use all new property sheets from the upgraded distribution. Go through each one to make sure the values are set how you want them. Read the HISTORY.txt file to see if there are any changes in this version that you need. Then it will work as it should... best not to bring old property sheets from an older CalendarX install, because these frequently change as development occurs.

4. There is not currently, nor immediate plans to create, an automated upgrade feature that will migrate your configurations to the new versions of the property sheets, etc. This is because CalendarX is designed to allow easy TTW access to all the scripts and templates for customization, and any one CalendarX installation will be expected to have many customization choices that an automated upgrade mechanism will not properly handle. Document the changes you make so that you can more easily upgrade to newer versions of CalendarX.

B. Upgrading to 0.6 from 0.4 branch versions of CalendarX.

1. Replace the old version of CalendarX in your INSTANCE_HOME/Products folder with the new version of CalendarX (0.6 or higher).
2. Restart Zope. REALLY! Do it! See why above.
3. Get all new property sheets from the new CalendarX skins folder. They will be similar to the old ones, but just do it anyway. Then go through them and set the properties the way you want. CalendarX will likely break if you just try to use your old (0.4) property sheets, so don't do it. Any parts of CalendarX that you have customized should be compared with fresh versions (and the HISTORY.txt file) to see if there are changes that you should be aware of in your customizations. If all you have changed are the CX_props_XXX property sheets, then there is little chance of any problems.
4. There is not currently, nor immediate plans to create, an automated upgrade feature that will migrate your configurations to the new versions of the property sheets, etc. This is because CalendarX is designed to allow easy TTW access to all the scripts and templates for customization, and any one CalendarX installation will be expected to have many customization choices that an automated upgrade mechanism will not properly handle. Document the changes you make so that you can more easily upgrade to newer versions of CalendarX.

Appendix B: Guide to property sheets in CalendarX

Property sheets are the tools that allow calendar administrators to configure CalendarX easily, without touching the source code. Easy access to these properties allows a great deal of flexibility in how CalendarX looks and what events it displays. Here is a guide to each of the properties associated with the property sheets, what you can accomplish with it and how to set it properly.

Administrators: Changing the properties associated with each property sheet is best performed in the usual Zope/Plone way... navigate in the ZMI to your /portal_skins/CalendarX folder and click on the name of the property sheet of interest to take you to a view of the properties on the sheet. Then click the button to "customize" that property sheet, which will place a copy of the property sheet in the /custom folder. To change property values, click on the **Properties** tab of the property sheet, and adjust values as desired, then click the Save Changes button at the bottom of the page.

If there are multiple CalendarX instances in a portal, then a property sheet in the /custom folder will override default values established in the filesystem. If different settings are required for just one CalendarX instance, then a copy of the appropriate property sheet can be placed inside the desired CalendarX folder (using the ZMI top copy/paste the property sheet there).

Developers: Property sheets are found in the /Products/CalendarX/skins/CalendarX folder of your distribution.

Officially, these property sheets are objects of a CMF class called FSPropertySheets (File System Property Sheets). The FSPropertySheet class currently (as of Zope 2.7.3) has deficiencies in its ability to handle lines and text properties, so CalendarX patches the methods used to handle these using the so-called "monkey patch" method so that FSPropertySheets can use lines and text properties. The monkey patch can be examined in the `__init__.py` file (the part that says "fixing the 'lines' property for CMF FSPropertiesObject"); this monkey patch provides the bulk of the `__init__.py` code. The patch is also described online in the CMF issue collector here: <http://www.zope.org/Collectors/CMF/271>, where the status is at this writing (Sept 18, 2005) "Accepted", but CMF is not yet patched to fix this, afaik.

If you want to change the default settings for CalendarX for your entire portal, or for all your portals (if running more than one Plone on your server), you can change the settings in the property sheets in the CalendarX folder in your INSTANCE_HOME/Products folder. Then all instances of CalendarX reading from that /Products folder will receive those defaults, unless they are overridden locally as described above in the Administrator's section.

Overview: Here are the names and basic function of the property sheets. The remainder of this appendix will cover details of the properties of each sheet in this order. I've described these in alphabetical order, but #2 (CX_props_calendar) is the most commonly used one.

1. **CX_props_addeventlink**: If you wish to have an "Add New Event" link displayed in the subject bar, configure the link here.
2. **CX_props_calendar**: controls most basic calendar functionality, including what types of events are shown, how the Subject bar is displayed, etc.
3. **CX_props_css**: provides many opportunities for changing the colors and fonts displayed in the calendar.
4. **CX_props_custom**: does nothing. If you add new functionality to your calendar and need properties, put them in here.
5. **CX_props_eventdisplays**: allows you to use different icons and CSS classes based on the Subject of each event.
6. **CX_props_popup**: checkboxes to select which text is displayed in the rollover text box associated with each event displayed in the calendar.
7. **CX_props_subcalendar**: If you have subcalendars beneath your main calendar, you will need to configure them here.

1. CX_props_addeventlink

The "Add New Event" link is a means of making it easier for calendar users to add new events to your calendar. It places a link in the SubjectLinks bar that takes users to an appropriate place to add events. Because there are many places/ways that Plone allows you to add events, we've provided several options for how to control who gets what link. Your suggestions (and code) for more options are gratefully accepted.

WARNING Misconfiguration of these options can cause error messages for your users... and it's not my fault. Make sure that your calendar users have the correct permissions to add events in the folders where you point them using this link. Read all the directions. *YOU HAVE BEEN WARNED.*

=== List of Attributes ===

title string

Leave this title attribute alone.

showAddEventLink boolean

Check this to include an "Add New Event" link in the SubjectLinks bar.

Controls for this link are below. If more than one of the boolean controls below are checked, the ones below will take priority over the ones above. For example, if both useANEFolder and useRolesAndFolders are checked, but the current user does not have one of the specified Roles, then the link target will fall back to the specified ANEFolderPath. The order of these priorities is determined by the code in the Python script "getAddNewEventURL". If no match is found, then a blank string will be returned to the macro, and a condition there will cause NO "Add New Event"

link to be shown. This way you can restrict display of this link to only certain users or to users with certain roles. The first two choices (useMemberFolder and useANFolder) are shown to all Authenticated users, if selected.

useCreateObjectOnClick **boolean**

createObjectOnClickCommand **string**

Together, these two properties tell the link to instantiate a new Event object in the target folder. Check this if you want to have the link automatically initiate editing of a new event for the user. Uncheck this property if you want the Add New Event link to simply take the user to a target folder without starting a new Event object automatically.

The createObjectOnClickCommand string is the command that is carried in the query string of the link's URL target if you are using the useCreateObjectOnClick property. The default string is:

```
createObject?type_name=Event
```

which will create a new Event object in the target folder of the link.

If you have a different event type that you would like to create, replace the meta_name "Event" with the appropriate meta_name of your desired event type.

If you use this feature, it is advisable to also set your portal to use the portal_factory for initiating Events, so that if a user clicks on the link to start a new event but then decides not to finish it, the event will not be abandoned half-finished. Portal_factory will simply create a temporary version and then delete it if left unfinished by the user.

useMemberFolder **boolean**

Check this so that the link will take users to their default Member folder where they can add Events.

useMemberSubfolder **boolean**

memberSubfolderPath **string**

Check this property if you want events to be instantiated in a subfolder of a user's Member folder. For example, if all your users are musicians in bands (or groupies perhaps) and they post their band gigs on the calendar, then you might want all the events to be saved in a specific subfolder, such as "/Members/username/gigs". The proper format for the target folder is relative to the /Members/username folder and should start with a slash, e.g.:

"/gigs"

NOTE: ONLY use this if you are certain that users WILL have the named subfolder in their user folder. Otherwise it will return 404, page not found error, or something closely related.

useANFolder **boolean**

ANFolderPath **string**

Together, these allow you to specify a single folder that will be the target of the link. The proper format for the target folder is relative to the portal_root and should start with a slash, e.g.:

"/somefolderintheroot/thefolderforevents"

useUsersAndFolders **boolean**

listOfUsersAndFolders **lines**

Together, these allow you to specify a combination of a username and a corresponding single folder that will be the target of the link for that specific user. The proper format for each line is as follows:

"username|folderpath"

where the "pipe" character (a vertical slash) is used as a separator between the username and folder path. The proper format for the target folder is relative to the portal_root and should start with a slash, e.g.:

"/somefolderintheroot/thefolderforevents"

An example with two possible role|folder lines:

lupa|calendar/specialevents

davos|calendar/drearyevents

If no matching username is found, the priority rules described above will take over to find a suitable target for the user.

useRolesAndFolders **boolean**

listOfRolesAndFolders **lines**

Together, these allow you to specify a combination of a Role and a corresponding single folder that will be the target of the link for all users with that Role. The proper format for each line is as follows:

"rolename|folderpath"

where the "pipe" character (a vertical slash) is used as a separator between the role name and folder path. The proper format for the target folder is relative to the portal_root and should start with a slash, e.g.:

"/somefolderintheroot/thefolderforevents"

An example with two possible role|folder lines:

Manager/calendar/specialevents
Member/calendar/ordinaryevents

The lookup stops when a matching role is found for the user. For example, if the Manager logs in, the link for the Manager will target the folder called "specialevents", even though the Manager is also (likely) a Member of the site. If no matching role is found, the priority rules described above will take over to find a suitable target for the user.

2. CX_props_calendar

CX_props_calendar controls most basic calendar functionality, including what types of events are shown, how the Subject bar is displayed, and many other attributes of the calendar's look and feel.

New attributes added in 0.6.x branch releases

showPrivateEventsToGroupMembers (0.6.0)
useHalfHours (0.6.2)
numMonthsForMultiMonthView (0.6.2)
earlyDayEventHour (0.6.4)

=== List of Attributes ===

title string

Leave this title attribute alone.

dayOfWeekToStart int

Value indicates what day of the week the Month and Week views begin on. Sunday = 0, Monday = 1, etc.

defaultView string

Name of the default view to be displayed: month, weekbyday, weekbyhour, or day.

useAdvancedQuery boolean

If checked, CalendarX will use the AdvancedQuery product for making queries to the catalog to find events. Use this if you want to override those query methods in your skin folder. I find AdvancedQuery significantly easier to use in building complex queries, but it offers no other advantages for general use in CalendarX.

dayViewStartHour string

Hour of day for CalendarX to BEGIN display for day view and for weekbyhour view. 8 = 8am = 08:00, 20 = 6pm = 20:00.

For a 24 hour calendar, set this to 0 (zero).

dayViewEndHour string

Hour of day for CalendarX to END display for day view and for weekbyhour view. 8 = 8am = 08:00, 20 = 6pm = 20:00.

Must be later than dayViewStartHour.

For a 24 hour calendar, set this to 24.

earlyDayEventHour string

Hour of day for CalendarX to use for deciding between events being classed as "later" or "earlier" in the Day and Weekbyhour views.

Previous behavior was that events up to midnight showed as "later" events and "earlier" events from midnight to dayViewStartHour showed up as "continuing". This property is the hour that now serves as the boundary between these types of events.

Default value is 0, representing 12 midnight, meaning there are no "later" events.

Setting value to 3, will represent 3am. Seems a reasonable boundary to me, meaning that events running 1am-2am will show up on the previous day, say Saturday night instead of Sunday morning.

Use Case: Calendar of events for a city where there are commonly late night movies or concerts after midnight... consider Las Vegas, for example. Now you can set the boundary to be 3 or 4am, so that when viewing the Day view for Friday, the later events as late as 3am on Saturday morning will show up on the Friday view.

Limitations: ONLY applies to Day and WeekByHour views, not to other views.

Therefore this could create some confusion among viewers who see a late night event on Saturday night on the Day view, but finding that same event displayed on Sunday on the month view. A fully consistent fix is for this situation is difficult, and I'm unwilling to program it at this time. I tried, and hence the long delay between the 0.6.4 and 0.6.5 releases. Instead I'm leaving it at this for now.

hoursDisplay string

Code to tell the "hoursdisplay" macro how to display the hours in your calendar views. Currently two possibilities and they only affect the left column display of hours:

"12ampm" = 12 hour display, with am or pm. Ex. 6 pm

"24.00" = 24 hour display, with period. Ex. 20.00

useHalfHours boolean

If checked, CalendarX will display half-hour increments on the day and weekbyhour views instead of the default one-hour increments. There

remains NO capability of starting the day on the half-hour... the `dayViewStartHour` and `dayViewEndHour` settings must be integer hour values. The default value is blank (false), meaning Hourly periods will be used. There is a modest performance penalty associated with this option, use at your discretion.

numMonthsForMultiMonthView string

How many months to display in the month view. Default value is 1, but may be extended up to 12 months. Next buttons move forward by one month, not by the number of months in the view. Querying time to generate several months at once may be excessive if many events are to be shown. A future TODO might be to allow the user a widget to select the number of months on-the-fly. However, this will necessitate changes in managing the URL querystring to accommodate this new parameter. At present, this feature is left as an exercise for the gentle reader.

showHeaderTitleAndIcons boolean

If checked, CalendarX will display the calendar title and description and email/print/favorites Action icons as it does for other Plone content.

showHighlightFullEvent boolean

If checked, CalendarX will show the full extent of Events on the calendar, even without rolling over with the mouse. Default is a blue color, can be changed in CSS property sheet.

NOTE! If you use this, you might also want to disable the `labelEventsOnlyAtStart` property. Disabling `labelEventsOnlyAtStart` means that the events in the Month view that span several days will show labels for each of those days, instead of only on the first day of the event.

showJumpToDateWidget boolean

If checked, a date-picking widget will show up at the top and bottom of the calendar near the Next/Previous links. This widget lets users pick a date and jump to it, instead of using multiple Next, Next, Next click, or manually typing the date into the URL querystring.

NOTE: this widget has been removed from the bottom of each of the views, where it has been for many months, in order to avoid a viewing Bug in IE6. It can be returned by changing the macro call at the bottom of each of the view templates from "`prevnextcurrentlinks_nojump`" to the previous "`prevnextcurrentlinks`" macro.

useNumericMonthInJumpToDateWidget boolean

If checked, the Jump-To-Date widget will show a numeric month value (ex. "2")

instead of an abbreviation of the month (ex. "Feb"). These abbreviations are pulled from the python DateTime module, not coded into CalendarX code, in the getMonthName.py script.

showPublicPrivateLink boolean

If checked, the *Public* vs *My Events* link will be shown in the Subject Bar. This link allows users to switch between viewing all the published events, or ONLY their own private events. If your calendar is mainly for viewing by anonymous users, you probably don't need this. Default is OFF because this is a nice feature, but not a commonly chosen one.

useMultiSubjects boolean

If checked, the Subject category picker is a checkbox-style form, allowing users to select multiple subjects for viewing. If unchecked, it switches to (an older) single subject chooser that only allows one Subject category at a time. Default is ON for this new-style Multi-Subject chooser, because it is ever so much nicer.

showSubjectBar boolean

If checked, the Subject bar is shown, and if unchecked, it will disappear from view. Default is ON. Decomplicates the calendar if you don't want to use Public/Private or Subject categories.

useCalendarHelp boolean

Check this attribute to show a View tab for "Calendar Help". This brings up a new view page that is intended for you to use for help in case you have neophyte users who could use some calendar help. I've added some code that brings up one page of help for "Members" and a different page of help for "Anonymous" users. This could easily be extended to show different help screens for other Roles. See the "help" view page template for more information. The help text is quite minimal, so feel free to expand it for your users. Feel free to send me a copy of your nice help files, too!

includeReviewStateVisible boolean

Check this attribute to include events where the review state is 'visible' as well as 'published'. This is useful for calendars where the only users are trusted users and going through the publishing workflow only adds unnecessary complication.

In particular, this could work well even on a site with many untrusted users. In that case, create a calendar for the trusted users that uses a repurposed Event with a new portal_type name.

Use the 'restrictToThisListOfTypes' attribute to make this new calendar ONLY read this one type of Event. Then use a `getNotAddableTypes.py` script to restrict the use of this type of Event to your trusted users (as a role, or a group, or whatever). See the [HowTo](#) on [plone.org](#) for use of `getNotAddableTypes`.

showPendingLink boolean

Check this attribute to show a link in the subjectbar that, when clicked, tells the calendar to display events with "pending" state as well as the other events (published, and visible if `includeReviewStateVisible` has been selected). The link is not a toggle; to get out of the mode where the pending events are showing, simply click any other link on the calendar.

This link ONLY shows up for Calendar Managers. Who is a Calendar Manager? User status as a Calendar Manager is determined by the `isCalendarManager.py` script. It is easily customized, but as a default is set to allow users with the "Manager" role. If this role is adequate for you, leave this script as is. An example is included in the script to show how to look up group membership to determine Calendar Manager status.

showPrivateEventsToGroupMembers boolean

Check this attribute to allow PRIVATE events to show up to anyone with the Plone privilege of viewing your PRIVATE events. In short, what this does is allow you to give one of your Groups or some other user a proxy ownership role for one of your `review_status=Private` events. That is enough to allow them to see it in Plone. But this attribute must be TRUE (checked) if you want such events to show up on a `CalendarX` instance.

For a fuller explanation, what this does is change the standard query to allow searching for `review_state "private"` events as well as "published" and "visible" (if `includeReviewStateVisible` is checked). But they will ONLY be shown for those users (or group members) where they have expressly been given the right to do so, usually via use of the Sharing tab on the event. So to use this, we recommend the following:

1. Use a recent version of GRUF (`GroupUserFolder`) in your Plone (version 3.0 or higher, not the 2.x that comes with Plone 2.0.x).
2. Make a group and enter your event in the group folder (usually something like `/groups/mygroup/myevent`).
3. Make the event a "private" event (instead of submitting it for review for publishing).
4. Use the sharing tab, and give the group proxy ownership of the event.

Now, with the use of this attribute, your group members can have a calendar that shows group-only events.

showOnlyEventsInMonth boolean

Check this attribute to restrict the Month view to display events ONLY in the current calendar month, and NOT those events that occur in the days before or after the month begins and ends (ie., if the month view shows the 30th and 31st of the previous month on the calendar, events will NOT be displayed for those dates).

labelEventsOnlyAtStart boolean

Check this attribute to put labels on the month view ONLY on the first day of an event that lasts multiple days. Default is SELECTED. Unselect this attribute if you'd like the event title and datestring to appear on each calendar day that the event is on (ie., a four day event will have the label show up on the calendar four times, on each of the four days of the event).

NOTE! Disabling this (to show events on EVERY day) will only work if the showHighlightFullEvent property is turned ON (selected). It would make no sense (to me) to have the event labeled, but not highlighted. So make sure you use these together. No harm if you don't, but it won't behave the way you might have expected. It pays to read the documentation.

listOfSubjects lines

restrictToThisListOfSubjects boolean

Together, these two attributes allow you to control the choice of what categories of events to display on your calendar.

List of the Subjects in your CalendarX, for use in creating the macro that displays them on your calendar.

1. LEAVE "listOfSubjects" BLANK, if you want to just use the list of Subjects that is available from already created Events your Plone site.
2. **LIST SUBJECTS ONE PER LINE**, exactly as they are present in your portal_metadata, in the order you want them displayed. The default values included here are the default values that come with CMF Event and AT Event types.

If "restrictToThisListOfSubjects" is checked, a query for "ALL" subjects is restricted to the Subjects in your listOfSubjects attribute.

If unchecked, "ALL" will return all events found, regardless of their Subject.

Use of this feature allows you to segregate certain events pertaining to certain Subjects to unique calendar instances.

This also means that if checked, the calendar WILL NOT pick up events that do not have a Subject selected.

ADVANCED FEATURE: Each line in the listOfSubjects can also be a Comma-Separated-Values list (CSV) where each line becomes a list of subjects for viewing. This becomes very useful in the case where you have many Subjects, but would like to combine several of them at a time and use a Nickname (or abbreviation, or acronym) to show up in the Subject menu.

For example: In a calendar for a school with five grade levels, and four classes of children in each grade level, you could try something like this in your listOfSubjects:

```
Class1a,Class1b,Class1c,Class1d
Class2a,Class2b,Class2c,Class2d
Class3a,Class3b,Class3c,Class3d
Class4a,Class4b,Class4c,Class4d
Class5a,Class5b,Class5c,Class5d
```

and then use this in the listOfSubjectTitles below:

```
Class 1
Class 2
Class 3
Class 4
Class 5
```

In this way, your subject menu is less cluttered, but it is easy for your users to check one of these to filter and see only the events for children of Class 1 age. You may also want to use SubCalendars with this, so that users can drill all the way down and see ONLY events associated with Class1c, Class4b, etc.

eventTypes list

restrictToThisListOfTypes boolean

Together, these two allow you to restrict what types of content objects will be picked up on your calendar. Put one portal_type per line in the eventTypes attribute, and check "restrictToThisListOfTypes" if you wish this feature to be activated. If unchecked, no check is done on the Type index, regardless of the content of the eventTypes attribute.

Usage: For example, this feature means you can create a new Event type for certain users, and then use getNotAddableTypes.py to restrict which users can add those special Event types, which gives even more control over different calendar instances in your Plone site.

This is very useful in subcalendars, where each subcalendar may have a different event Type.

listOfPaths list

restrictToThisListOfPaths boolean

Together, these two allow you to restrict where (the paths) to event content objects that will be picked up on your calendar. This means you can restrict viewing to Events found in certain folders. To use this feature, use a full path exactly as found in your path index. An example:

```
/clients/companyplonesite/Members/fred  
/clients/companyplonesite/staff
```

These two paths represent folders where Events can be stored that will show up on the calendar, if restrictToThisListOfPaths is checked. ONLY those events in these two paths will be found. Events in fred's personal folder and events in the staff folder, and any folders deeper than that will be picked up for display on this calendar. For example, if there is a meetings folder inside the staff folder, events inside that folder will also be displayed. If you are having any trouble with this property, please go to the portal_catalog, click on the Catalog tab, and find one of the events that *should* show up on the calendar. Look near the bottom of the page to see what path is being indexed by the "path" index, and use that as the path to the folder that you will use in listOfPaths.

restrictToThisFolder boolean

This property restricts the calendar so that events are ONLY shown if they are found within or beneath the parent folder of the CalendarX instance. Example: add a CalendarX instance as /Members/lupa/cal. If you set this property to true, then only events found within /Members/lupa and any subfolders therein will be shown on this calendar instance.

This could also be accomplished manually with the restrictToThisListOfPaths property, but this property helps in the special case where you want to allow your users to create a private calendar for their own area. In that case, you should probably set this property to "1" in the property sheet on the filesystem, so that all calendars created by your users will have this property by default.

ADVANCED NOTE: as currently implemented, the restrictToThisFolder option trumps (overrides) the similar restrictToThisListOfPaths property. In other words, in the four query scripts, the restrictToThisListOfPaths property is evaluated first, and if restrictToThisFolder is also selected, the second one (restrictToThisFolder) takes precedence and overrides the

first property. To change this precedence, simply go into these query scripts and rearrange the two-line calls for each one so that their order is reversed.

listOfSubjectTitles list

useSubjectTitles boolean

Together, these two attributes allow you to use sensible (e.g., shorter) titles on the Subject Bar for your Subject categories. If your Subject category names are long, three or four subjects can produce an unwieldy list for your users to select from. Instead, use these attributes to include a list of shorter titles for each of your Subjects.

IMPORTANT Be sure to use this feature in conjunction with the "listOfSubjects" attribute above. Be certain that both lists have the exact same number of entries, so that there is a single corresponding SubjectTitle for each Subject. If these do not match, the calendar may show an error. Simply test your calendar after any changes to this attribute to be certain that your calendar is working without error.

3. CX_props_css

CX_props_css controls the many adjustments that you can make to control the colors and fonts for displaying text in various parts of the calendar. These properties are used in the calendar.css and other CSS files.

Use the Properties tab to adjust the colors and fonts and such throughout the calendar.css and other CSS files.

Default values (include below) are set to nearly match those of a stock Plone 2.0.x install. Or if not match, at least nicely complement. It isn't perfect, but overall things work OOTB. The font sizes are all in percentages instead of fixed sizes, so that the calendar fonts can resize with the rest of the Plone site if the small Normal LARGE style-sheet widget is used.

Has NOT been adjusted at all for Plone 2.1. If you find problems related to Plone 2.1 and this CSS, please email Lupa and we'll chat.

==== List of Attributes ====

title string

Don't mess with this title. Leave it alone. [actually, I don't care.]

**** Section 1: ****

VIEW: These attributes control aspects of the "view" tabs, (month, week, day, etc. view template names) right at the top where you choose which view of the calendar is showing.

viewTabsBorderColor default = #8cacbb

View tabs border color.

viewTabsBackgroundColor default = #dee7ec

View tabs background color.

viewFontBaseSize default = 95%

View tabs font size.

viewFontFamily default = "Lucida Grande", Verdana, Lucida, Helvetica, Arial, sans-serif

View tabs font family.

viewTabsFontColor default = #436976

View tabs font color.

**** Section 2: ****

Subject Bar: These attributes control aspects of the subject bar, where the Private/Public and MetaCalendar Subject choices are found.

subjectBarBorderColor default = #8cacbb

Color of the border around the subject bar, set in calendar.css. Also goes around the My:Public choices.

subjectBarBackgroundColor default = #dee7ec

Color of the background for subject bar and My:Public bar.

subjectFontFamily default = "Lucida Grande", Verdana, Lucida, Helvetica, Arial, sans-serif

Font family for the subject bar and My:Public bar.

subjectFontSize default = 97%

Font size for the subject bar and My:Public bar.

subjectBarFontColor default = #436976

Font Color for the for subject bar and My:Public bar.

**** Section 3: ****

Header: These attributes control aspects of the header area, where the previous and next date arrows, and the calendar date

are displayed, at the bottom and top of the calendar. Code for this is generated in the "prevnextcurrentlinks" macro.

headerCenterFontSize default = 135%

"July 2004" header font size.

headerSideFontSize default = 93%

"previous" and "next" links font size.

headerFontFamily default = Verdana, Helvetica, Arial, sans-serif

Font Family name for "prevnextcurrentlinks" macro (prev, next, date header, footer)

headerFontColor default = #436976

Color of the font for header.

headerHeight default = 0px (formerly 35px)

Height added to base for "prevnextcurrentlinks" macro (prev, next, date header, footer)

headerMarginBottom default = 0px (formerly 15px)

Bottom margin pixels for "prevnextcurrentlinks" macro (prev, next, date header, footer)

headerMarginTop default = 0px (formerly 5px)

Top margin pixels for "prevnextcurrentlinks" macro (prev, next, date header, footer)

headerPadding default = 3px

Padding size for "prevnextcurrentlinks" macro (prev, next, date header, footer)

**** Section 4: ****

Continuing: These attributes control aspects of the Continuing Events section, where events that start before or extend across the entire period of view are shown.

continuingHeaderFontSize default = 90%

Continuing events box, header font size

continuingHeaderFontFamily default = Verdana, Helvetica, Arial, sans-serif

Continuing events box, header font family

continuingOuterBorderColor default = #B3CFD9

Continuing events box, outer border color

continuingOuterBorderWidth default = 1px

Continuing events box, outer border width

continuingHeaderBorderColor default = #436976

Continuing events box, inner border color

continuingHeaderBorderWidth default = 1px

Continuing events box, border width

continuingHeaderBackgroundColor default = #8CACBB

Continuing events box, background color

continuingRowEventBackgroundColor default = #DEE7EC

Continuing events box, color if an event is present

continuingRowNoEventBackgroundColor default = #F7F9FA

Continuing events box, color if no event

continuingRowHeight default = 5px

Continuing events box, row height, added to event bottom

**** Section 5: ****

Cal: These attributes control aspects of the Main Calendar display.

***CONTROL OF THE MAIN CALENDAR

calBorderColor default = #B3CFD9

Main calendar, border color

calBorderWidth default = 1px

Main calendar, border width

***CONTROL OF THE TR TAGS

calTableRowOddBackgroundColor default = #F7F9FA

Main calendar, for certain views where odd/even vary in color,
this controls the ODD row (in TR tags).

calTableRowEvenBackgroundColor default = #DEE7EC

Main calendar, for certain views where odd/even vary in color,
this controls the EVEN row (in TR tags).

***CONTROL OF THE TH TAGS

calTableHeaderBackgroundColor default = #8CACBB

Main calendar, TH tags background color.

calTableHeaderBorderColor default = #436976

Main calendar, TH tags border color.

calTableHeaderBorderWidth default = 1px

Main calendar, TH tags border width.

calTableHeaderFontColor default = #FFFFFF

Main calendar, TH tags font color.

***CONTROL OF THE EVENT FONTS

calEventFontSize default = 85%

Main calendar, TH tags font size.

calEventFontFamily default = Verdana, Helvetica, Arial, sans-serif

Main calendar, font family for the event listings.

calEventPendingTextColor default = #436976

Main calendar, text color for the pending event listings.

calEventPrivateTextColor default = #821513

Main calendar, text color for the private event listings.

calEventPublishedTextColor default = #466A06

Main calendar, text color for the published event listings.

calEventVisibleTextColor default = #436976

Main calendar, text color for the visible event listings.

***CONTROL OF THE TD TAGS (daily cells in the month view, etc)

calTableDataFontColor default = #000000

Main calendar, TD tag text color, but I don't know if it actually controls anything at this time.

calTableDataBorderColor default = #DEE7EC

Main calendar, TD tag border color.

calTableDataBorderWidth default = 1px

Main calendar, TD tag border width.

calTableDataNoEventBackgroundColor default = #F7F9FA

Main calendar, color when a cell has NO EVENT.

calTableDataEventBackgroundColor default = #DEE7EC

Main calendar, color when a cell has an EVENT. Also used in calendar.js.shtml for rollover highlighting return value.

calTableDataOutOfMonthBackgroundColor default = #FFFFFF

Main calendar, in the MONTH view when the day shown is NOT a part of the month, this controls the background color. Also used in calendar.js.shtml for rollover highlighting return value.

calTableDataOutOfMonthBorderColor default = #F7F9FA

Main calendar, in the MONTH view when the day shown is NOT a part of the month, this controls the border color.

calTableDataOutOfMonthBorderWidth default = 1px

Main calendar, in the MONTH view when the day shown is NOT a part of the month, this controls the border width.

calTableDataSpanDayFontColor default = #000000

Main calendar, in the MONTH view, text color of the date (ie., "3" on June 3 cell).

calTableDataHeightMonthView default = 105px

Main calendar, in the MONTH view, this controls the empty height of a daily cell.

calTableDataHeightDayView default = 35px

Main calendar, in the DAY view, this controls the empty height of a daily cell.

calTableDataHeightWeekbydayView default = 105px

Main calendar, in the WEEKBYDAY view, this controls the empty height of a daily cell.

calTableDataHeightWeekbyhourView default = 30px

Main calendar, in the WEEKBYHOUR view, this controls the empty height of a daily cell.

calTableDataFontSizeHour default = 130%

Main calendar, in the DAY and WEEKBYHOUR views, this controls the font size of the HOUR displayed (ie., "8am" or "13:00")

calTableDataEventHighlightBackgroundColor default = #FFE7C4

Main calendar, in the view when an event has a mouseover (rollover) event, this controls the rollover background color. Read into calendar.js.shtml for this control.

4. CX_props_custom

CX_props_custom is an empty property sheet that you can use to add new properties if you are a developer customizing

CalendarX. By using this sheet, rather than adding new properties to the existing sheets, you should find it easier to upgrade to new versions of CalendarX because your special settings will be here, rather than in one of the new, updated property sheets of the new CalendarX version. The CX_props_custom property sheet will always be empty in the CalendarX distribution.

=== List of Attributes ===

title *string*

Don't mess with this title. Leave it alone. [actually, I don't care.]

[that's all. add new properties yourself inside this property sheet.]

5. CX_props_eventdisplays

Use the Properties tab to adjust the attributes of the Event as displayed on the calendar views. These properties control the icons available for display and also the CSS classes. You can choose to control icons and CSS classes according to the Subject of the Event, or according to the Type of the Event.

=== List of Attributes ===

title *string*

Leave this title attribute alone.

useSubjectIcons *boolean*

listOfSubjectIcons *lines*

If checked, this will cause the views to choose an icon for each event based on the Subject names found in the list. The list consists of a lines attribute where each line consists of a Subject and an icon ID, separated by a pipe (|) character. For example:

Work|event_work_icon.gif where Work is the subject.

Your subject names should (must!) match the actual subjects you use for your events, or this method will not work well. Actually, it will just pull the default event_icon.gif from the Plone skin if there is any problem finding a matching subject name or icon ID.

This property is handy for making your events more visibly recognizable in your calendar page. The default icon size is 16x16 pixels, with some white (or clear) pixel space on the right and left sides. I haven't tested it with larger icons, but keeping to a modest size might be a good idea.

Add your event icons into your /portal_skins/custom folder, or put them directly into your calendar instance folder for (slightly) better performance.

useSubjectCSSClasses boolean

listOfSubjectCSSClasses lines

If checked, this will cause the views to choose a CSS class for each event based on the Subject names found in the list. The list consists of a lines attribute where each line consists of a Subject and a CSS class name, separated by a pipe (|) character. For example:

```
US Holiday|event_usholiday  where Work is the subject, and
                             event_usholiday is the CSS class name.
```

Your subject names should (must!) match the actual subjects you use for your events, or this method will not work well. Actually, it will just pull the default event_published CSS class from the Plone skin if there is any problem finding a matching subject name or icon ID.

This nicely allows you to apply styles like font color to your event listings according to the Subject of the event. Put your custom styles into your calendar.css stylesheet, or into a customized version of the ploneCustom.css stylesheet if you prefer (the sample ones I've created for default use are found in calendar.css). An example of a CSS class listing I used to use a blue text color for the "Appointment" event subject:

```
A.event_appointment {
  COLOR: #0000CC;
  TEXT-DECORATION: none;
}
A.event_appointment:hover {
  COLOR: #0000FF;
  TEXT-DECORATION: none;
}
```

Additionally, if you want the Subjects in the Subject listing at the top of the calendar to reflect these same CSS classes, you have to add these too. Example ones (for Appointment, etc.) are in calendar.css for you to customize. They are in a SPAN tag and look like this:

```
TABLE.caltabs TD.barright2 SPAN.event_appointment {
  COLOR: #0000CC;
  TEXT-DECORATION: none;
}
```

***** ONE MORE NOTE about these.** Make sure in your `listOfSubjectCSSClasses` and `listOfSubjectIcons` lines properties that you use the actual SUBJECT name and not any Subject nicknames you might use for display (as defined in the `listOfSubjectTitles` property in `CX_props_calendar`. Those labels won't work here, only the actual subject will work.

***** AND ONE MORE NOTE.** The following two properties (`useEventTypeIcons` and `useEventTypeCSSClasses`) take precedence over `useSubjectTypeIcons` and `useSubjectCSSClasses` if, for unknown reasons, BOTH have been selected. There's no real reason to select both... only one can work at a time, and I chose to make the EventType ones take priority. Go figure.

useEventTypeIcons boolean

listOfEventTypeIcons lines

If checked, this will cause the views to choose an icon for each event based on the Event Type (`portal_type`). The list consists of a lines attribute where each line consists of an Event Type and an icon ID, separated by a pipe (|) character. For example:

Event|event_icon.gif where Event is the `portal_type`.

useEventTypeCSSClasses boolean

listOfEventTypeCSSClasses lines

If checked, this will cause the views to choose a CSS class for each event based on the Event Type (`portal_type`). The list consists of a lines attribute where each line consists of a Subject and a CSS class name, separated by a pipe (|) character. For example:

AT Event|atevent_class where AT Event is the `portal_type`, and `atevent_class` is the CSS class name.

6. CX_props_popup

Use the Properties tab to adjust the attributes of the rollover PopUp text that displays when you rollover each event displayed on the calendar. Just check off the ones that you want to show up. There's a howto on the CalendarX.org website that explains how you can add other attributes as well.

Technical Note: These values represent the metadata that are stored in the `portal_catalog` for each event. If you want to display other information about the event that is NOT in the metadata, you can read the howto on the CalendarX.org website that explains how

you can add other attributes as well.

=== List of Attributes ===

title string

Leave this title attribute alone.

showPOPTitle boolean Checked = Show this info.

Whether to show the "Title" of the event.

showPOPType boolean Checked = Show this info.

Whether to show the "Type" string, telling what type of event object is showing up in the calendar.

showPOPSubject boolean Checked = Show this info.

Whether to show the "Subject" of the event. Currently implemented to show only the first Subject, not multiple events in the case that some of your events have multiple Subjects chosen. Subjects are separated by a comma and a space, so that they wrap nicely if you have a long list of subjects.

showPOPStart boolean Checked = Show this info.

Whether to show the "Start" date and time of the event.

showPOPEnd boolean Checked = Show this info.

Whether to show the "End" date and time of the event.

showPOPCreator boolean Checked = Show this info.

Whether to show the "Creator" (Plone username) for this event.

showPOPCreated boolean Checked = Show this info.

Whether to show the "Created" date of the event.

showPOPModified boolean Checked = Show this info.

Whether to show the "Modified" date of the event, when the last time this event was edited.

showPOPState boolean Checked = Show this info.

Whether to show the review "State" of the event, such as published, visible, etc.

showPOPDescription boolean Checked = Show this info.

Whether to show the "Description" of the event.

7. CX_props_subcalendar

NOTE: These instructions are greatly supplemented by Part V in this Manual, a Use Case describing subcalendar use for a Resource Scheduler. Even if you don't want a resource scheduler, I'd advise you to read it and/or try it out if you want to use CalendarX subcalendars.

NOTE: Plone 2.1.1 has a bug that affects the use of CalendarX Resource Scheduler. Read about it (and the patch) in INSTALL.txt.

Subcalendars allow for interesting possibilities for special calendars, especially very busy ones. A subcalendar is a folder inside (nested) in a Main calendar folder. I have only tried a single level of nesting (one main calendar with multiple subcalendars), and these controls focus on menu behavior reflecting that use case.

The properties in this sheet control the basic behavior of subcalendars.

The first two properties are used by the main calendar, and the second two are used by subcalendars. These properties have been developed with a Resource Scheduling calendar application in mind, but other possibilities are certainly possible. Your suggestions (and code) for more options are gratefully accepted.

ALSO: You will need to carefully consider what properties to use in your CX_props_calendar property sheets for both the MAIN and SUB calendars. Examine the Resource Scheduling calendar example provided in order to see what some of the possibilities are.

FINALLY: Read the "Manual for CalendarX" document. We provide a CASE STUDY of the use of Subcalendars to create a RESOURCE SCHEDULING calendar. It will help you more fully grok subcalendars and their usefulness.

=== List of Attributes ===

title string

Leave this title attribute alone.

useSubCalendarSubjectMenu boolean

For MAIN Calendars: Check this property to signal that (1) there are subcalendars below and (2) hence, use the special Subcalendar Menu for the Subject Links that allows you to both (either) filter on the subcalendars as well as click on the Subject (subcalendar name) to drill down and view that subcalendar alone.

listOfSubCalendars lines

For MAIN Calendars: This is a list (one per line) of the names of the subcalendars. The menu choices uses this list for display of links to the

subcalendars.

isSubCalendar **boolean**

For SUB Calendars: Check this property if this calendar folder is a subcalendar. This controls the style of menu displayed for subcalendars versus non-subcalendars.

nameOfSubCalendar **string**

For SUB Calendars: The name of this subcalendar. This is displayed in the Subject Links area.

Appendix C: Guide to python scripts in CalendarX

Python scripts perform the bulk of the logic in handling calendar queries, manipulation of objects, and so on. A Python script is conveniently customizable by Developers from within the ZMI, or they can be customized on the filesystem. For example, if desired a Developer can create a customized filesystem version of CalendarX complete with changes to the Python scripts that control where and how the queries are made to find events for the calendar. Then any (all) CalendarX instances within the portals using this custom CalendarX product will utilize the custom queries. If the Developer wants one instance of CalendarX to use a different version of the query script, then by simply placing a customized version of the desired Python script inside the CalendarX folder, this local script instance inside the CalendarX folder will override the default behavior and use the new Python code instead.

We'll start with (1) a full listing of all the Python scripts used in CalendarX, followed by (2) examination of how a page template uses one Python script to return values, and then (3,4,5) we'll take a closer look at a couple of the CalendarX Python scripts and see how they work. By examining them closely you can see how queries are assembled before calling the portal_catalog, and you can learn how we call the CalendarX property values out of the property sheets and macros out of other page templates.

1. Alphabetical listing of Python scripts in CalendarX, with brief job descriptions:

getAddNewEventURL.

Returns a URL for the Add New Event link based on property sheet values

getContLaterEventsWBH.py

Returns a list of continuingEvents or laterEvents for the WeekByHour view.

getCXAttribute

Returns an attribute (property) from the appropriate CalendarX property sheet.

getCXEventsBefore

Chooses which means of querying the catalog for Events before a given date.

getCXEventsBetween

Chooses which means of querying the catalog for Events between two dates.

getCXMacro

Returns a path to the macros from within the CalendarX default skin property sheets.

getDaysOfMonth

Returns the number of days in a given month.

getDaysOfTheWeek

Returns list of names of the seven days of the week, in a proper order for use on the calendar month view, or week view. Starting day is set in CX_props_calendar.

getDictCommon

Returns a dictionary of useful objects for the calendar views

getDictDay

Returns a dictionary of useful objects for the Day view

getDictMonth

Returns a dictionary of useful objects for the Month view

getDictMultiMonth

Returns a dictionary of useful objects for the MultiMonth view

getDictWeekbyday
Returns a dictionary of useful objects for the Weekbyday view

getDictWeekbyhour
Returns a dictionary of useful objects for the Weekbyhour view

getEndOfDay
Returns a DateTime object for the end of the Day view showing on the calendar (defaults to one second before midnight). Hour range: 1-24.

getEndOfMonth
Returns a DateTime object for one second before midnight on last day of the Month

getEndOfWeek
Returns a DateTime of the last second of the day of the end of the calendar week.

getEventDictDay
Returns a dictionary of useful objects for Events for the Day view

getEventDictMMonth
Returns a dictionary of useful objects for Events for the MultiMonth view

getEventDictMonth
Returns a dictionary of useful objects for Events for the Month view

getEventDictWeekbyday
Returns a dictionary of useful objects for Events for the Weekbyday view

getEventDictWeekbyhour
Returns a dictionary of useful objects for Events for the Weekbyhour view

getEventIcons
Returns an icon object from the skin based on the Subject or Event Type

getEventsBeforeAQ
Queries catalog to retrieve events between two dates using AdvancedQuery.

getEventsBeforeZC
Queries the catalog for Events before a start date using ZCatalog query dictionary.

getEventsBetweenAQ
Queries catalog to retrieve events between two dates using AdvancedQuery.

getEventsBetweenZC
Queries the catalog for Events between two dates using the ZCatalog query.

getEventTypeCSSClasses
Returns a CSS class name based on the event Type (portal type)

getListOfSubjectTitles
Get listOfSubjectTitles from property sheet, safely

getMonthName
Returns a month name given a month integer, or a month list

getNumOfDays
Returns an int for number of days or dates between start and end. a "day" is 24 hours, a "date" is calendar day (midnight)

getNumOfHours
Returns integer number of hours from start of calendar to event. works for weekbyhour or day view.

getNumOfPeriods
Returns integer number of periods (hours or halfhours) from start of

calendar to event. works for weekbyhour or day view.

`getNumWeeksInMonthToShow`
Returns an int number of weeks showing in a Month view showing on the calendar for any given DateTime object (num in that month)(from 4 to 6 weeks possible). Cleans some code up (as of 0.6.1).

`getStartOfDay`
Returns a DateTime object for the beginning of the Day view showing on the calendar (defaults to midnight).

`getStartOfMonth`
Returns a DateTime object for midnight on 1st day of the Month

`getStartOfMonthToShow`
Returns a DateTime for the first of the Month view showing on the calendar (end of the previous month, usually).

`getStartOfWeek`
Returns a DateTime object for the start of the first day of the week, for use on the calendar month view.

`getSubjectCSSClasses`
Returns a CSS class name based on the event Subject

`isCalendarManager`
Returns True (1) if current user is considered a "Calendar Manager"

`listSortByStart`
Returns list of sorted events from a list of events not sorted. Added 0.6.4.

`listUnique`
Returns list of unique values from a list of values not unique and if `listOfSubjects` is True, creates a unique list of subjects from that attribute

`makeCSV`
Returns a CSV (comma separated variables) string from a list.

`queriesSubtract`
subtracts q2 from q1: looks for objects in q1 that are NOT in q2 and builds a q1new list of those objects, and returns q1new

`queriesUnique`
Returns Brain of unique events from a Brain of events not unique

2. A brief overview of Python script use in CalendarX view templates:

CalendarX strives to make as much of the functionality of the calendar closely accessible to the Plone developer for customization. That's why so little of the functional Python code is located in the `CalendarXFolder` class, and so much of it is present in the `/portal_skins/CalendarX` skins folder. From there, you can easily customize the behavior of your CalendarX instances, and even make each of several CalendarX instances behave differently from the others.

Here's the basic story of how CalendarX creates a view.

1. Call the URL
(e.g., <http://mycalendar.biz/calendar/month?currentDate=2005/01/10&xmy=0&xsub=ALL>)

This calls the month.pt template and initiates the view generation process with three calendar parameters passed through the URL: currentDate, xmy, and xsub.

2. The month page template gathers a number of variables together at the beginning that can be used throughout the page template ('global' variables). Code for this looks like this:

```
<body>
<!-- Defining global variables -->
<div metal:fill-slot="main"
    tal:define="
        MODIFIED string:mod 0.4.12 use calendarPrint.css;
        viewname string:month;
        DateTime python:modules['DateTime'].DateTime;
        Dict python:here.getDictMonth();
        url here/absolute_url;
        ampm Dict/ampm;
        etc....
```

One by one, here's what those lines do:

1. MODIFIED defines a string... in this case it serves simply as a comment. Each of the page templates has a similar string at the beginning to show how it differs from previous versions of the template. Another common string I use in page template code is "COMMENT".

2. viewname defines a string 'month'. This parameter is passed to other scripts for 'month' specific needs.

3. DateTime defines a Python module called 'DateTime'. This is used as an abbreviation so that wherever else (such as macro code) that we need to call a DateTime function, the module will be available for us.

4. Dict defines a dictionary of useful information. There are many parameters that get defined in each of the templates, and some of these are common to all the templates. What we do here is call a template-specific Python script called getDictMonth.py that in turn calls another script called getDictCommon.py, and this returns a dictionary of many variables, as you will see in the following statements.

5. url defines the URL of the current page by calling here/absolute_url.

6. ampm defines a Boolean (True/False) value based on the value of the 'hoursDisplay' property found in the CX_props_calendar property sheet (True if hoursDisplay = 'ampm', and False if hoursDisplay equals anything else). This value is returned to the month template from 'Dict', the dictionary we got from getDictMonth script (and in turn, this value comes from the getDictCommon script). In earlier versions of CalendarX, these variables were all defined right here in the page template code. In this case the code would have looked like this:

```
ampm python:test(here.getCXAttribute('hoursDisplay') == '12ampm',1,0)
```

and there would have been many, many such definitions one after the other, some more complex than others. Instead, all (well, at least much) such Python logic has been moved out of the template and into a few scripts to clean up the template code and so that such code only needs to be changed in one place (rather than in many page templates).

Cleaning and rearranging the code in CalendarX is an ongoing battle/chore/drama. Currently, there are still several places where code could and should be moved out of the page templates and into other Python scripts, but hasn't yet. It doesn't mean CalendarX is bad... it just means I haven't gotten to everything yet. CalendarX is not, by the way, model code for how to build Products in Plone.... it is full of less-than-best practices. But with each new branch of the code I try to make improvements in the architecture, and get more good advice. Meanwhile, it works, and I'm happy to maintain it and make gradual improvements.

3. getCXAttributes.py -- How CalendarX gathers properties from the property sheets.

In brief, I wanted the property sheets in CalendarX to be very flexible in where each property was located. So I created this script to go through a list of all the property sheets and find the property no matter which sheet it is located on. So to call a property, you use the following approach in a python script:

```
propval = context.getCXAttributes('propertyName')
```

and it will run through the list of property sheets until it finds the property requested.

4. getCXMacro.py -- How CalendarX finds its macros.

In brief, I wanted the macros to be similarly flexible. Currently there is only one sheet of macros, but in the 0.5 branch there are two and there could be more. To call a macro from a page template, we use the following tal approach:

```
<metal:block metal:use-macro="python:here.getCXMacro('subjectlinks')"/>
```

This is a bit more convoluted than the typical Plone macro call, but it has flexibility. In particular, the getCXMacro script looks in any listed macros pages in order to find the right macro... if you write custom macros, you can even put them in a new page of macros just your own. Also, if you customize any of the CalendarX macros, you can put them in another page of macros and list that page first... getCXMacro will look through the macro sheets in order that they are listed (in the Properties tab of the CalendarX instance folder in the ZMI). So just put your custom macro sheet first in the order, and your customized ones will be read instead of the default CalendarX versions in CX_props_macros.pt. That makes upgrading to new releases of CalendarX easier, because you haven't changed the default macros in CalendarX. You'll still need to make sure that your modified macros are compatible with any changes to CalendarX -- use the HISTORY.txt file to see whether one of your customized macros has been affected.

5. How CalendarX queries for events (getCXEventsBetween.py and friends).

Later. I'm getting tired of writing right now. I think I won't write this part up until I redo it for the 0.5 branch, which is quite different, and faster.

Appendix D: Guide to page template views and macros in CalendarX

There are five views in CalendarX: MultiMonth, Month, WeekByDay, WeekByHour and Day. Each template has a similar overall style, with a number of defined variables at the top, followed by calls to CSS and Javascript files, and then code to generate the views. Much of the detailed code has been pulled out of these templates into macros and Python scripts to make it easier to maintain and manage. Additionally, most of the important features of the views that can be made configurable have been pulled out and put into configurable properties to minimize the amount of customization required to create a unique calendar for your Plone site. These properties are described in Appendix B. The Python scripts are described separately in Appendix C.

Macros are simply page template code snippets that are called in from the view templates -- this separates the code so that it only needs to be changed in one place, rather than in each template, when a common change should propagate through all the views. The macros for CalendarX are all located in `CX_props_macros.pt`, in the `/skins/CalendarX` folder of your CalendarX product distribution, and on your site this can be found and customized from the `/portal_skins/CalendarX` folder. Here is a brief description of these macros in the order they are found in `CX_props_macros`, with a brief description of each macro's purpose.

1. `samplemacronamehere` - a sample macro that is unused anywhere. Copy and paste this to use as a starting point for macro code of your own.
2. `caltabsforviews` - the view tabs at the top of the calendar: month, weekbyday, weekbyhour, day, help.
3. `prevnextcurrentlinks` - the text and links at the top/bottom of calendar: previous month, next month, the Date string (e.g., June 2006), and the Jump To Date widget. There is also a near duplicate macro called `prevnextcurrentlinks_nojump` (added in 0.6.5) that is used at the bottom of each view, and that displays no Jump To Date widget to avoid an IE browser bug.
4. `monthdaysofweek` - orders days of the week properly for the month view
5. `hoursdisplay` - displays hours on the day, weekbyhour views.
6. `popuptextbox` - generates the rollover popup text boxes for each event. You can control what appears here through the `CX_props_popup` property sheet, and you can further customize it here, if desired.
7. `eventlister` - displays the event text and link in the calendar cells for each event. This macro runs once for each event listed on a view.
8. `copyright` - Powered by CalendarX link... comment the code out here if you don't want to show this on your website ;-(
9. `subjectlinks` - shows choices for Subject categories, plus other widgets. This is a long, complicated macro. In the new 0.5 branch I've created several versions of this macro so that you can choose different layouts just using the property sheets. You could do the equivalent here... just copy the entire macro and rename the original one `subjectlinks_ORIGINAL`. Then modify your new `subjectlinks` macro and the views will pick up the new one here.

Read through Appendix C on Python scripts to learn how `getCXMacro.py` helps find the macro needed.

Month view:

The Month view displays a month of events at one time, with a standard calendar appearance (4 to 6 weeks of seven days in each week). Events within each day are arranged in order of starting time.

There are several configurable properties that ONLY affect the month view, using the CX_props_calendar property sheet (see Appendix B for details on each):

- showOnlyEventsInMonth: this controls whether out-of-month events are shown or not (e.g., events on January 31 for a February calendar view)
- labelEventsOnlyAtStart: this controls whether the title of a multi-day event is shown on each of the days, or just on the first day of the event.

Here is a typical month view (from 0.4.12, but nearly identical to 0.6.5):

The screenshot displays the CalendarX for iPhone web interface. At the top, it shows the logo and version information: "CalendarX for iPhone Running: 0.4.12(stable)". A search bar is located in the top right corner. Below the header, there are navigation links: "home", "the guide", "news", "forums", "issues", "links", "about", and "members". A secondary navigation bar includes "home", "my folder", "my preferences", "junks", "phone setup", and "log out".

The main content area is titled "January 2005" and shows a calendar grid. The days of the week are listed at the top: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday. The calendar shows events for the month of January 2005, with the current date set to January 22, 2005. Events are listed in a list view on the right side of the calendar grid. The events include:

- CalendarX 0.4.12(stable) is released (01:17 am - 02:45 am)
- 1st. Release Happy New Year (06:27 pm - 08:00 pm)
- Meeting with David (12:00 am - 11:00 am)
- CalendarX 0.4.12(stable) (01:30 am - 01:30 am)
- CalendarX 0.3.9(release) is released (06:15 pm - 07:40 pm)
- Workshop (11:00 am - 11:00 am)

At the bottom of the calendar grid, there are navigation controls: "<< previous month", "January 2005", "2005", "Jan", "22", "Jump", and "next month >>".

Powered by [CalendarX, a iPhone Calendar](#).

WeekByDay view:

The WeekByDay view displays a week of events at one time, with each day having one block of events, arranged in the same order as the starting time of each event. The starting day of the week (Sunday, Monday, etc.) is configurable by the Administrator, using the dayOfWeekToStart property of the CX_props_calendar property sheet.

Here is a typical weekbyday view (from 0.4.12, but nearly identical to 0.6.5):

The screenshot shows the CalendarX for Home web application interface. At the top left, it says "CalendarX for Home" and "Running: 0.4.12(stable)". There are navigation links for "home", "the goods", "news", "howto", "issues", "faq", "links", "sites", and "members". A search bar is located at the top right. Below the navigation, the current view is "week by day" for the week of "January 9, 2005 - January 15, 2005". The interface includes a sidebar with "new & updated" items, a main event list for each day, and a footer that says "Powered by CalendarX, a Free! Calendar".

CalendarX for Home
Running: 0.4.12(stable)

home | the goods | news | howto | issues | faq | links | sites | members

you are here: home » calendar:0.4.12(stable) » weekbyday

month | week by day | week by hour | day

Categories: view All Appts Conversion Meetings Parties WorkWorkWorks

« previous week: **January 9, 2005 - January 15, 2005** 2005 Jan 15 Jump next week »

Tuesday, January 9, 2005

Monday, January 10, 2005

- CalendarX 0.4.11(stable) (start: 01:20 pm - Jan 10, 2005 | end: 01:55 pm - Jan 10, 2005)
- CalendarX 0.3.4(dev) is released (start: 09:15 pm - Jan 10, 2005 | end: 07:48 pm - Jan 10, 2005)

Tuesday, January 11, 2005

Wednesday, January 12, 2005

Thursday, January 13, 2005

Friday, January 14, 2005

- aaa (start: 08:47 am - Jan 14, 2005 | end: 08:58 am - Jan 15, 2005)

Saturday, January 15, 2005

- aaa (start: 08:47 am - Jan 14, 2005 | end: 08:58 am - Jan 15, 2005)

« previous week: **January 9, 2005 - January 15, 2005** 2005 Jan 15 Jump next week »

Powered by CalendarX, a Free! Calendar.

WeekByHour view:

The WeekByHour view is the most complicated view... it displays a week of events arranged hour by hour through each day. In this respect it is a bit like seven day views arranged side by side. The view can be set to show the entire 24 hour day (midnight to midnight), or in the default setting from 8am to 8pm, or for any set interval (using the dayViewStartHour and dayViewEndHour properties in the CX_props_calendar property sheet). The listing is arranged in one-hour blocks, and multiple events within the hour are arranged in order of starting time.

The starting day of the week (Sunday, Monday, etc.) is configurable by the Administrator, using the dayOfWeekToStart property of the CX_props_calendar property sheet. The hour display on the left side of the view can be configured as 12 hour (am/pm) display, or as a 24 hour display (using the hoursDisplay property in CX_props_calendar). Events that start on this day but before the starting display time (e.g., a meteor shower event at 4am) will appear in the Continuing events block at the top, and later events (late parties) will show up at the bottom of the page in a Later events block. Rolling over these events with the mouse will highlight the appropriate day they start on.

Here is a typical weekbyhour view (from 0.4.12, but nearly identical to 0.6.5):

The screenshot shows the CalendarX for Home web application. The top navigation bar includes links for home, the goods, news, faq, issues, links, about, and members. The main content area is titled "week by hour" and displays a calendar grid for the week of January 9, 2005, to January 15, 2005. The grid shows events for each hour of the day. A tooltip is visible over an event on Monday, January 10, 2005, at 1:30 pm. The tooltip text is as follows:

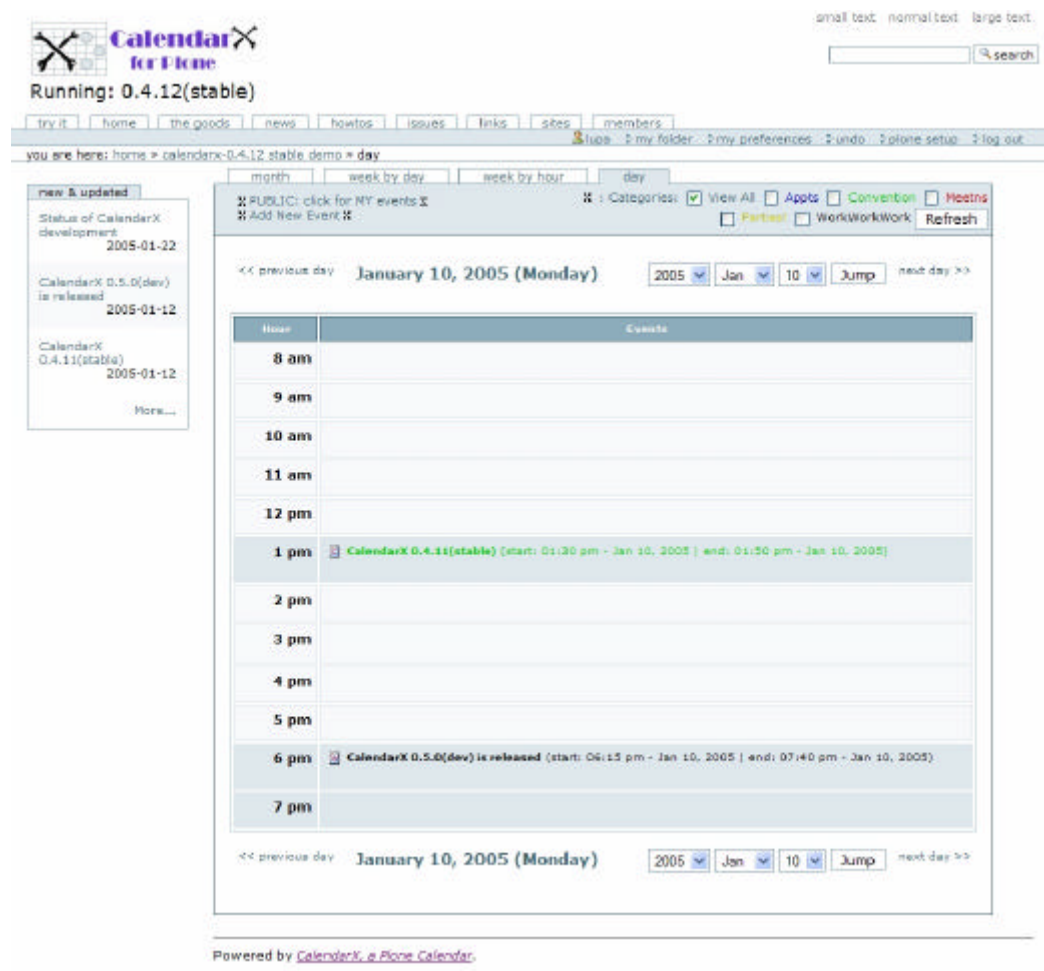
subject: correction
Description: CalendarX-0.4.11[stable] was released Monday. It provides non-critical bugfixes that allow users to show up if naming a Personal calendar, and that also certifies in which view CSS elements to be properly rendered by the user. Feedback from the CSS property sheet. That's all we could find broken in the first 10 days of 0.4.11.

Day view:

The Day view displays one day's worth of events. The view can be set to show the entire 24 hour day (midnight to midnight), or in the default setting from 8am to 8pm, or for any set interval (using the `dayViewStartHour` and `dayViewEndHour` properties in the `CX_props_calendar` property sheet). The listing is arranged in one-hour blocks, and multiple events within the hour are arranged in order of starting time.

The starting day of the week (Sunday, Monday, etc.) is configurable by the Administrator, using the `dayOfWeekToStart` property of the `CX_props_calendar` property sheet. The hour display on the left side of the view can be configured as 12 hour (am/pm) display, or as a 24 hour display (using the `hoursDisplay` property in `CX_props_calendar`). Events that start on this day but before the starting display time (e.g., a meteor shower event at 4am) will appear in the Continuing events block at the top, and later events (late parties) will show up at the bottom of the page in a Later events block.

Here is a typical day view (from 0.4.12, but nearly identical to 0.6.5):



small text normal text large text

CalendarX for iPhone

Running: 0.4.12(stable)

try it home the goods news howtos issues links sites members

you are here: home > calendarx-0.4.12 stable demo > day

month week by day week by hour day

PUBLIC: click for NY events

Add New Event

Categories: View All Appts Convention Meetings Parties WorkWorkWork Refresh

<< previous day January 10, 2005 (Monday) 2005 Jan 10 Jump next day >>

Hour	Events
8 am	
9 am	
10 am	
11 am	
12 pm	
1 pm	CalendarX 0.4.11(stable) (start: 01:30 pm - Jan 10, 2005 end: 01:50 pm - Jan 10, 2005)
2 pm	
3 pm	
4 pm	
5 pm	
6 pm	CalendarX 0.5.0(dev) is released (start: 06:15 pm - Jan 10, 2005 end: 07:40 pm - Jan 10, 2005)
7 pm	

<< previous day January 10, 2005 (Monday) 2005 Jan 10 Jump next day >>

Powered by [CalendarX, a Plone Calendar](#).

Appendix E: Guide to CSS and JavaScript code in CalendarX

CSS is used in CalendarX. So is JavaScript. This is good.

Without touching the page templates, you can configure nearly every visible color and font shown in CalendarX. That's what the plethora of properties in `CX_props_css` is all about. Try it, you'll see.

The JavaScript is much more complicated. Even I'm not sure what everything does, and I've been over it all and through it more than twice or thrice.

For more good information, read the source code. Thank Limi for Plone CSS, and Oliver for PloneCalendar CSS, and thank Oliver for PloneCalendar JavaScript, which together make nice popup text boxes and rollover highlighting quite possible.

Recently (December 2005) I was asked how to change the rollover (orange) highlighting color that is shown in the event boxes when you mouseover an event. Duh! I didn't know! Even though there are more than 50 configurable CSS properties in the property sheet, this is not there. Then I realized why (once I looked)... this variable is NOT used in the `calendar.css` or any of the other `view.css` files. Instead, it is defined at the top of the `calendar.js` JavaScript code. It is easy to change, just put your own hexadecimal value in, and the rollover highlighting works with the new color.

Maybe someday I'll get ALL the variables in one place.

[[more to come...]]

Appendix F: About Python, Zope, Plone, and CalendarX branches

Python is a programming language designed for clean object oriented programming with high-level commands, a rich collection of libraries, and an elegant syntax that allows rapid development. Python is an open source project.

Zope is an application server built on the strength of the Python language, with authentication management and an object database that is ideal for web development. Zope provides a powerful base for rapid and maintainable web application development that scales well with inexpensive hardware. Zope is an open source project.

Plone is an object-oriented content management system (CMS) built on the powerful Zope application server, and featuring rapid development of custom content types and a flexible, standards-compliant web interface. Because of its strong base in Zope and Python, there simply is no other professional quality CMS more flexible and powerful than Plone. Plone is an open source project.

CalendarX is a web-based calendar product that works with Plone portals. CalendarX is pretty good, and it's getting better all the time. CalendarX is an open source project.

And as one raving fan puts it:

"i love calendarx coz...
it does what it says on the tin."

What more can you ask for?

OK... maybe iCalendar integration, hotsync with your Palm or Outlook or Evolution, recurring events, better group functionality, a portlet version, lots more new kinds of views to choose from, clickable icons to let you start a new event from any date or time within the calendar, easily customizable new Event types, PopUp event detail windows instead of the default Plone views, skinning for a standalone calendar that really speeds things up by peeling away some of the Plone extras, maybe a few others.

But, of course, it doesn't say any of those things on the tin. Someday it might say all of those things, but not yet.

CalendarX Branches:

Each time I start a new version of CalendarX for development purposes, I call it a "branch". Each branch has some defined goals, and sometimes a few undefined hopes or ideas to test out. I am dedicated to several things about these branches... like taking each one from a developmental stage where everything works, but the features are changing in the early releases, to a stable release where the only changes will be bugfixes.

Currently there are three stable branches: 0.1, 0.2, 0.4 and 0.6. The 0.1 and 0.2 branches were the first versions of CalendarX and can be run on any version of Plone, including the older Plone 1.0 releases. The 0.4 branch is stable and fairly bugfree. The latest stable branch is 0.6, which is what is documented (mostly) in this manual. The current development branch is 0.5, which adds significant new functionality in the querying area to CalendarX.

0.6 Branch: i18n.

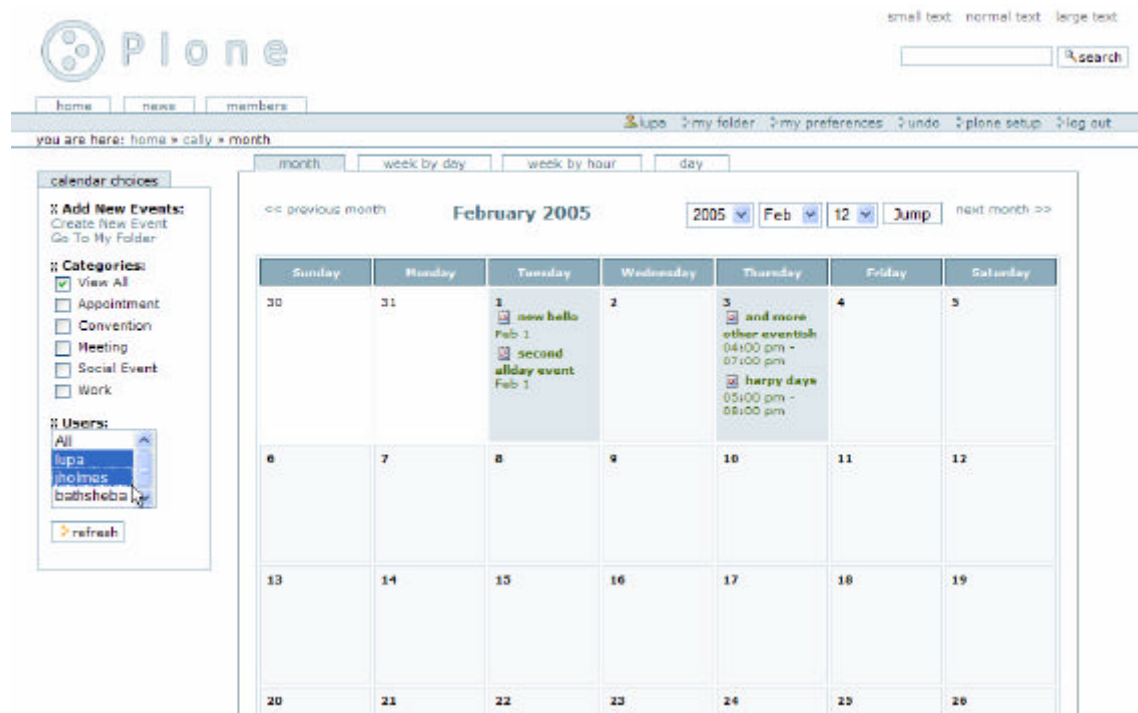
The 0.6 branch is built on the code base from stable release 0.4.15, and adds i18n functionality. As a user, you will simply see the calendar in the appropriate language if your browser's default language setting is set to one of the supported languages. At the 0.6.5-stable release, there are eleven languages supported (including English). In the INSTALL.txt document (and repeated in Appendix A of this manual), there are directions on how to make a translation of your own for CalendarX, and there is a listing of all the translations we would like to have (and more can be added beyond these).

In addition to internationalization, this branch also offers several small feature improvements over the 0.4 branch, some code speedups that improve rendering times, the ability to display the Day and WeekByHour views in half-hour increments or hourly, and a new MultiMonth view that shows from 2-12 months (default is 3).

0.5 Branch: One future.

So, now we introduce the 0.5 branch of CalendarX. Nearly all new development is currently focused on the 0.5 branch, currently (since mid-April 2005) at version 0.5.3(dev). Each release of the development branch is as bugfree as possible (I don't release a tarball until all critical bugs have been squashed), and many times development releases are used in production Plone sites (that was certainly the case for the 0.4 and 0.6 branches, and it is the case for some 0.5 branch sites). However, I don't recommend investing significant work customizing a production calendar based on the development branches simply because I make no guarantee that the API and internals of the development branch will remain stable into the future... once I make it up to the alpha or beta stage, you can be pretty certain that nothing major is going to change and any changes to the CalendarX branch beyond that will be bugfixes and non-critical feature improvements. That's my caveat... but that said, there are folks out there (me included) that use the development branch releases in production use all the time, and aren't afraid to jump in and modify things to make it work the way we want it to. Go for it!

Here's a screen shot of 0.5.1 branch of CalendarX, default release with portlet_cx_choices:



Some of the new features in 0.5 branch of CalendarX:

1. Control of calendar choices can use a portlet, cleaning up the calendar header nicely.
2. A month-view portlet_calendarx is available to replace the stock portlet_calendar of Plone, and it draws its behavior from any designated CalendarX instance in your portal.
3. Greatly expanded property selection (nearly 100 new properties) for controlling what events are displayed on your calendar, and who can see them, including some group functionality.
4. A User-Picker widget (shown above) to let you select a few Users and see only their events on the calendar. Group-Picker widget to follow sometime in the future.
5. Speed. CalendarX-0.5 branch is now significantly faster than the 0.4 branch, but not any faster than the 0.6 branch (although this is not thoroughly benchmarked yet).

No manual yet for the 0.5 branch, but the new features are documented in the /docs folder, just as they are for the other branches.

0.7 Branch: merge stable 0.6 with best-of from 0.5.

No work has been done on this yet, but the next release will likely be a fork of 0.6 that includes some of the most useful features of 0.5.

Happy scheduling!

+lupa+