

Podcast Producer: Scheduling Podcasts

If you work with podcasts on an ongoing basis, scheduling for the recording and publishing of your podcasts is critical. From an audience perspective, you want to have a regular stream of content to keep people interested in your podcasts. You also want to make sure that you are making the best use of your presenters by giving them the time and flexibility to create a podcast at a time that suits their schedule and yours.

Instead of explicitly recording a podcast and having the information submitted into the system, why not schedule podcasts recordings to take place automatically according to your organizational needs? For example, within a school or university you may have regular class sessions that you want to record and publish as a podcast.

The article demonstrates how to use AppleScript in combination with iCal to automate the recording of podcasts from cameras without having to manually create each recording. The solution will simplify the scheduling of recordings to create a suitable event within iCal and then uses iCal and Podcast Producer handle all of the complexities of the recording process.

Automating Your Recordings

One way to simplify the recording of podcasts in your network is to automate the process of starting and stopping recording on different cameras. To achieve this, you use the standard scheduling and calendar features of iCal to specify the start and stop times for each podcast. The summary, description, and attendee information of the iCal event can be used to set the corresponding title, description and author information within the final podcast.

To handle multiple cameras in different locations, different calendars within iCal can be used to trigger the recording on each corresponding camera. By using different calendars, you gain the ability to schedule and automate the process across your entire organization.

Podcast Producer Remote Control

The first step to setting up the automated recording of podcasts from cameras is to examine the command line interface to Podcast Producer. The podcast tool is installed on all Mac OS X Leopard machines, both the server and client versions, and this provides a command line interface to the Podcast Producer system.

The core of the sample application requires the Podcast Producer server name and the user and password credentials of a user with rights to manage the Podcast Producer system. Thus, all requests using podcast generally start with the following options on the command line:

```
$ podcast --server pcastserver --pass password --user admin
```

You can get a list of cameras registered with the Podcast Producer by using the `--listcameras` command:

```
$ podcast --server pcastserver --pass password --user admin --listcameras
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>cameras</key>
    <array>
      <dict>
        <key>name</key>
        <string>Boardroom Camera</string>
        <key>uuid</key>
        <string>E7352910-AF04-46D9-9C47-EEBCAB426298</string>
        <key>preview_url</key>
        <string>https://boardroom.example.com:8170/agent_previews/E7352910-AF04-46D9-
9C47-EEBCAB426298.jpg</string>
      </dict>
    </array>
  </dict>
</plist>
```

This example shows that there is a single camera, called Boardroom Camera, registered with Podcast Producer.

Starting Recordings

To start a recording, use the `--start` option and specify the name of the camera to use for the recording. For example, to

Related Articles

Podcast Producer: Anatomy of a Workflow

Podcast Producer: Writing Actions

Podcast Producer: Using the Command Line

Podcast Producer: Publishing on YouTube

Leopard Technology Series for Developers: Server Overview

Resources

Reference Library: Mac OS X Server

Mac OS X Server Podcast Producer Workflow Tutorial (PDF, 1MB)

Mac OS X Server Podcast Producer Administration (PDF, 2.1 MB)

Leopard Server QuickTours (in iTunes)

Quick Tips Videos

Podcast Producer Mailing List

Podcast Producer Technology Brief (PDF, 1.2 MB)

start recording to the Boardroom Camera, do the following:

```
$ podcast --server pcastserver --pass password --user admin --start "Boardroom Camera"
Starting recording on 'Boardroom Camera' on https://boardroom:8170/podcastproducer
'Boardroom Camera' recording started
```

This will start the recording. All of the information is recorded locally on the machine that hosts the camera and the recording of the video material will continue until you either cancel the recording, or stop the recording.

Keep in mind that canceling the recording halts the recording process and deletes the recorded material:

```
$ podcast --server pcastserver --pass password --user admin --cancel 'Boardroom Camera'
Canceling recording on 'Boardroom Camera' on https://boardroom:8170/podcastproducer
'Boardroom Camera' recording canceled
```

Stopping Recordings

Stopping the recording using the podcast command saves the recorded material and then submits it as a podcast into Podcast Producer using a workflow that you specify. You also need to submit a metadata file that contains the information about the podcast itself, such as the title, description, author and other information that will be inserted into the podcast when it is published.

The format is the property list file like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>Author</key>
    <string>Any B Author</string>
    <key>Comment</key>
    <string>Podcast Producer allows remote control</string>
    <key>Copyright</key>
    <string>Copyright (c) 2008 Apple</string>
    <key>Description</key>
    <string>Podcast Producer command line interfaces.</string>
    <key>Keywords</key>
    <string>podcast producer</string>
    <key>Title</key>
    <string>Controlling Podcast Producer from the shell</string>
  </dict>
</plist>
```

For example, to stop the recording and submit the recorded material to the Blog workflow, save the metadata information into a file called `podcast_meta.xml`, and then use the podcast to remotely stop the recording:

```
$ podcast --server pcastserver --pass password --user admin --stop
'Boardroom Camera' --workflow 'Blog' --metadata podcast_meta.xml
```

Starting and stopping the recording remotely is only part of the process. To automate the process for recording each podcast, iCal can be used to schedule the recordings and trigger the recording process.

Using iCal to Schedule Podcasts

Using iCal, you can create meetings and appointments, set the date and time when the meeting occurs, assign individual meetings to different calendars and add attendees and other information to each event.

For automating the recording of podcasts, the key feature of iCal that can be used is the ability to add an alarm to an event. Alarms can signal a simple noise or dialog just before an event occurs. But more usefully, iCal alarms can also call AppleScripts that can then do a variety of tasks using the powerful AppleScript language and environment to control different applications.

Scheduling a Podcast Recording

To schedule a Podcast recording, an event must be created in a calendar within iCal. How the calendars are organized is entirely up to your requirements. You can use a single calendar to hold podcasts and use the location field of the event to signify which camera you want to use for the podcast. Alternatively, you can create a different calendar for each camera and use this information to control the cameras that are used for each recording. Using different calendars can be useful if you want to use iCal to help schedule the different podcasts. Given how each calendar can be a different color, this helps to identify each podcast and the schedule. You can see an example of the multi-calendar solution in Figure 1.

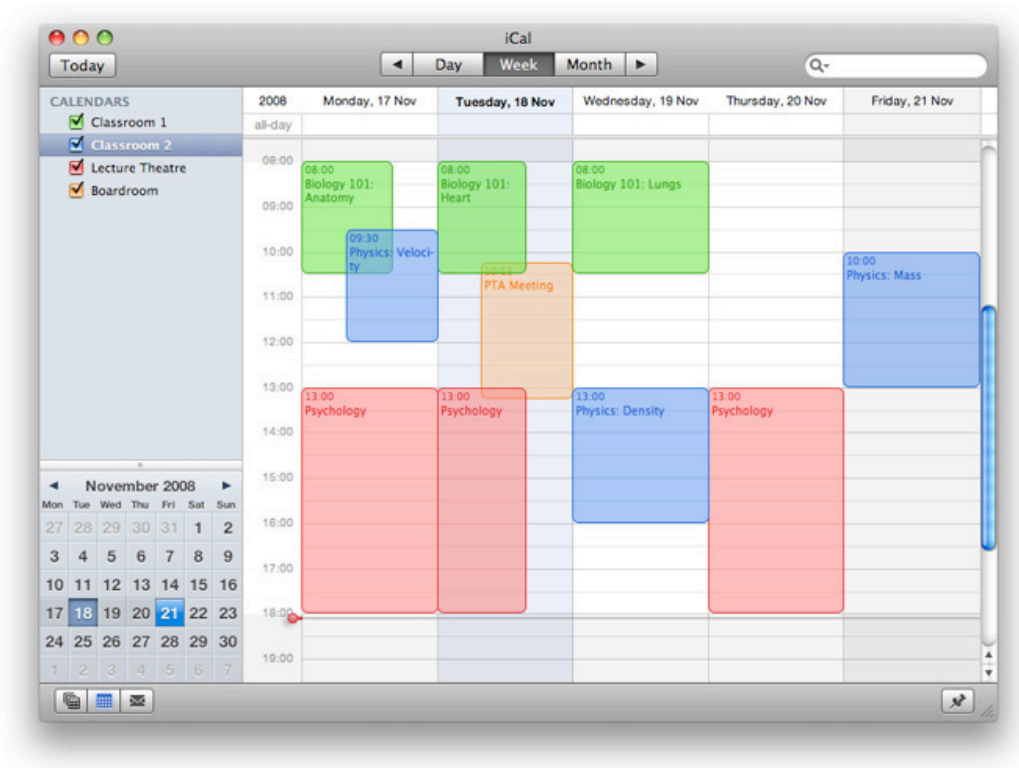


Figure 1: A multi-calendar Podcast schedule

To create a scheduled podcast, you need to create a standard event within a calendar. The event specifies all of the information about the podcast you need, and then selects the AppleScript that will perform the actual recording. An example of the event creation is shown in Figure 2.

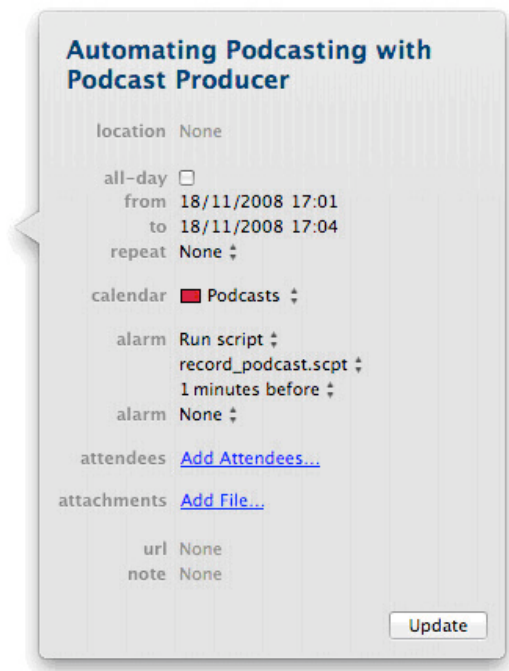


Figure 2: A scheduled podcast event in the calendar

The trigger for the recording process is the AppleScript attached to the alarm. This script takes the iCal event information, starts the recording, creates a suitable metadata file, stops the recording, and then submits the information to Podcast

Producer. You can configure the iCal alarm to use the delay method to start the recording of the podcast before it is actually due to start. For example, you might want to specify that the podcast starts recording 1 minute before the scheduled start time.

Automating Podcasting using AppleScript

The AppleScript that will start the remote recording and then submit the recording to Podcast Producer has a number of key steps. First, the information about the podcast, such as the title and duration, is taken from the event that triggered the alarm. Then the script starts the recording by using the podcast command line tool.

The duration of the event is then used to wait until the podcasts scheduled time has completed. Once the end point has been reached, the metadata file required to submit the podcast to the system is created, and the podcast is stopped, using the created metadata file as the base information for the final podcast.

Here is the full AppleScript for automating the podcasting process.

```
set myPCCalendar to "Podcasts"
tell application "iCal"
  activate
  set theEvent to the last event of calendar myPCCalendar
  set theDescription to summary of theEvent
  set theDuration to (end date of theEvent) - (start date of theEvent)
  set theAuthor to display name of (first attendee of theEvent)
  set theTitle to summary of theEvent
  do shell script "podcast --server pcastserver --pass password --user admin
--start 'Boardroom Camera'
delay (theDuration + 30)
tell application "System Events"
  set the parent_dictionary to make new property list item with
properties {kind:record}
  set the plistfile_path to "~/Desktop/podcastmeta-" & myPCCalendar & ".plist"
  set this_plistfile to
  make new property list file with properties {contents:parent_dictionary,
name:plistfile_path}
  make new property list item at end of property list items of contents
of this_plistfile
  with properties {kind:string, name:"Author", value:theAuthor}
  make new property list item at end of property list items of contents
of this_plistfile
  with properties {kind:string, name:"Comment", value:theDescription}
  make new property list item at end of property list items of contents
of this_plistfile
  with properties {kind:string, name:"Copyright", value:"Copyright The
iCal Podcast"}
  make new property list item at end of property list items of contents
of this_plistfile
  with properties {kind:string, name:"Description", value:theDescription}
  make new property list item at end of property list items of contents
of this_plistfile
  with properties {kind:string, name:"Keywords", value:"podcast producer
ical" & myPCCalendar}
  make new property list item at end of property list items of contents
of this_plistfile
  with properties {kind:string, name:"Title", value:theTitle}
end tell
do shell script "podcast --server pcastserver --pass password --user admin --stop
'Boardroom Camera' --workflow 'Blog' --metadata " & plistfile_path
end tell
```

First, the name of the calendar that you want to extract the information from is set at the top of the script shown in Listing 1. The event information is not automatically provided to AppleScript when the alarm is triggered. Instead, you need to use a trick to get the event from iCal. You do this by identifying the last event in a specific calendar. You will also use the calendar name as part of the filename for the metadata file that you will create. This allows you to create different metadata files for different podcasts on different calendars simultaneously:

```
set myPCCalendar to "Podcasts"
```

Next, the script communicates with the iCal application to get the last event object in the calendar:

```
tell application "iCal"
  activate
  set theEvent to the last event of calendar myPCCalendar
```

Now, the information about the event is extracted from the event. This is performed at the start of the process, not the end, to ensure that you are pulling the right information from the right calendar event:

```
set theDescription to summary of theEvent
set theDuration to (end date of theEvent) - (start date of theEvent)
set theAuthor to display name of (first attendee of theEvent)
set theTitle to summary of theEvent
```

The duration of the podcast can be calculated by looking at the start and end times of the iCal event. By subtracting the end time from the start time, AppleScript will return the number of seconds of duration for the event, which is required so that the AppleScript can pause until the recording needs to be stopped.

Next, the recording is started by calling the command line podcast tool demonstrated earlier. To record the full podcast, the AppleScript is paused for the duration, plus a grace period of 30 seconds to ensure that the podcast has fully completed by the time the recording stops:

```
do shell script "podcast --server pcastserver --pass password --user admin
--start 'Boardroom Camera'"
delay (theDuration + 30)
```

The next code block performs the key stage required to actually submit the podcast recording for processing into Podcast Producer. You create the metadata XML file (or Property List) that defines the podcast parameters that will be used during publication. AppleScript can create property lists using the System Events interface:

```
tell application "System Events"
    set the parent_dictionary to make new property list item with
    properties {kind:record}
    set the plistfile_path to "~/Desktop/podcastmeta-" &map; myPCCalendar & ".plist"
    set this_plistfile to
        make new property list file with properties {contents:parent_dictionary,
name:plistfile_path}
    make new property list item at end of property list items of contents
of this_plistfile
        with properties {kind:string, name:"Author", value:theAuthor}
    make new property list item at end of property list items of contents
of this_plistfile
        with properties {kind:string, name:"Comment", value:theDescription}
    make new property list item at end of property list items of contents
of this_plistfile
        with properties {kind:string, name:"Copyright", value:"Copyright The iCal Podcast"}
    make new property list item at end of property list items of contents
of this_plistfile
        with properties {kind:string, name:"Description", value:theDescription}
    make new property list item at end of property list items of contents
of this_plistfile
        with properties {kind:string, name:"Keywords", value:"podcast producer
ical" & myPCCalendar}
    make new property list item at end of property list items of contents
of this_plistfile
        with properties {kind:string, name:"Title", value:theTitle}
    end tell
```

With the podcast over, and the metadata file with the required information written, you just need to stop the recording and submit the information into Podcast Producer, using the same command line tool as before:

```
do shell script "podcast --server pcastserver --pass password --user admin
--stop 'Boardroom Camera' --workflow 'Blog' --metadata " & plistfile_path
end tell
```

Now when the alarm calls a given event, the camera and recording will automatically stop according to the event that you scheduled within iCal.

The process is so automatic that in most cases you only need to create the schedule within iCal and attach the appropriate alarm and script for your podcasts to be automatically recorded and scheduled into the system.

Summary

In this article we have examined a method for automating the recording of podcasts by using a combination of the Podcast Producer command line interface and iCal. The command line interface enables you to control individual cameras registered with Podcast Producer, including starting and stopping recordings automatically and remotely. From iCal, you can craft a podcasting schedule by creating an event and attaching an AppleScript that uses the podcast command line tool to start and stop the recording process. All of the information about the podcast, including the author, title and description information, in addition to the start and stop times, can be taken from the iCal event, making the scheduling of podcasts as simple as creating an event in your iCal schedule.

Posted: 2008-12-16