Tv Box File Format #00000: Weekly schedule file format

Jonathan Campbell

January 28, 2013

This documentation covers Castus 3.0 or earlier

0.1 Introduction

When I created the Tv Box interface one of the things I wanted to ensure was that it would be possible for both the veejay, UI, and any perl or shell script to parse the schedule easily. After all, if maintaining the schedule software wise required a lot of complicated code there would be a lot of room for bugs and odd behavior. So I decided that the on-disk format used to store the weekly schedule should be text based. That way if something goes wrong, any text editor can be used to fix up the schedule.

0.2 File name convention

The Tv Box metadata design allows your file to be named whatever you want, but it is highly recommended for interchange purposes that you use the file extension .weekly-schedule to indicate a weekly schedule.

0.3 Basic format

A schedule file is made up of groups of name = value pairs. Each group contains name = value pairs associated with that group. The pairs, as well as the starting and ending brackets of the pairs, are each stored as one line of text. This design makes parsing extremely simple: just read one line of text at a time.

0.4 Schedule time slots

A time slot is a group of name = value pairs that describe the item to play, and the start/stop time to play it. Each item starts with an "anonymous" group, a line of text with a lone open curly bracket. The item ends when you reach a line of text with a lone closing curly bracket.

NOTE: By convention all current implementations insert a tab '\t' character prior to each line within the group. This is not required, but it makes it easier to visually work with the schedule in a text editor.

name	*value*
item	Full path of the item scheduled to play in the time slot
start	Weekly starting time
end	Weekly end time

Example schedule time slot:

```
{
        item = /mnt/main/Daily Run/Promotional recycling group 3.vob
        start = monday 5:00 pm
        end = monday 7:30 pm
}
```

0.5 Weekly schedule time encoding

Start and end times in a weekly schedule are specified as a day, hour, minute, and second count since the start of the week (sunday). Parsing the time is deliberately simple; just read the text one word at a time, dividing at each space character, and read sequentially.

When your code starts parsing, assume that the day is sunday, the hour is 0 (12 AM midnight) and that it's the top of the hour (minutes and seconds are 00). As you read through the text, update the variables as you read in keywords and values that you understand.

0.5.1 Days of the week

The day of the week is given either as the full name on the Gregorian calendar, or the first 3 letters (shorthand). Current implementations are actually designed to match by the first three, ignoring the rest, but you will remain most compatible with the Tv Box by generating 3-letter or full names in lowercase text. You convert the day of the week to an integer (0-6) by matching the name.

day of week	*name*	*shorthand name*
0	sunday	sun
1	monday	mon
2	tuesday	tue
3	wednesday	wed
4	thursday	thu
5	friday	fri
6	saturday	sat

0.5.2 Hours, minutes, and seconds

Time is written in *h:m:s* or *h:m* format. The hour field may be in 12 or 24-hour format, depending on whether the timestamp includes the "am" or "pm" tags.

If you are parsing the words from the text and the first character is a number, then that word should be parsed as the hour/minute/second timestamp. Take the word, and divide the string by each occurence of the colon ":". The first number is the hour, the second number is the minute, and the third number is the second. It is expected that if the second or third fields are absent that you act as if that field were zero.

As part of the parsing process, your code must also take note of whether the time is "am", "pm", or not specified. If AM/PM is specified, the hour value is modulated to keep it within the 0...11 range, and then 12 is added if PM. So, 12:00 AM is the 0th hour, and 12:00 PM is the 12th hour.

If AM/PM were NOT specified, then the hour is modulated to keep it within the 0...23 range (24-hour clock).

0.5.3 Conforming time stamp

A conforming time stamp structure has one day of the week, one am/pm specification, and one hour:minute:second specification. An implementation wishing to generate the widest compatible timestamp must conform to that basic pattern. If you do weird things like specify TWO wallclock times or multiple days of the week then the resulting time gathered by an implementation is effectively undefined and unreliable.

0.6 Weekly default items

The default item group is defined by a non-anonymous open curly bracket, explicitly named "defaults, day of the week". Each name = value pair inside specifies what item is associated with a given day of the week.

NOTE: Same rules apply as when parsing time slots, the day of the week is given either as a whole name or the first 3 characters.

NOTE: For maximum compatibility with current implementations, put this group first in the schedule file.

NOTE: Current implementations place a tab '\t' char before every entry in the group. This is not required.

```
defaults, day of the week {
        sunday = /mnt/main/Sunday Funday.vob
        monday = /mnt/main/Educational blues.vob
        tuesday = /mnt/main/Slideshow.ispslideshow
        wednesday = /mnt/main/testing123.vob
        thursday = /mnt/main/thursday.vob
        friday = /mnt/main/tgif.vob
        saturday = /mnt/main/saturday.vob
}
```

0.7 Differentiation from other schedule variations

This form of encoding the schedule is also re-used in current implementations for monthly and yearly schedules. To differentiate between monthly and yearly a line of text is placed at the top of the file that states the variation, in the form "*type" where the first char is an asterisk.

The basic format is the same, but the encoded time stamps have different meaning, and they will be described in another document.