ENIGMA2 DATE AND TIME REFORM

Written by IanSav on 19-Dec-2016 for OpenViX and OE-Alliance developers.
Updated by IanSav on 24-Jan-2017.

SUMMARY

Much of Enigma2 has date and time formats hard coded across the code base and skins. Different coders have used different formats. There is no coordination and no option for users to have the dates or times presented in a format with which they are most comfortable.

The original proposal suggested adding a new infrastructure into the Enigma2 code that supports a range of date and time formats that can be selected by users. The nature of the proposal resulted in a more coherent and uniform presentation of dates and times across the Enigma2 user interface (UI).

ADVISORY NOTE

Care should be taken before enabling the various display options, particularly the am/pm and AM/PM time formats. The new time formats can be substantially wider than their 24 hour clock equivalent. (Previously 24 hour time was the only time display option available.) Developers should ensure that their code and skins can acceptably display all of these new formats before the options to select the various formats are enabled and exposed to users. My testing with OpenViX has shown that the 12 hour clock displays are mostly harmlessly trimmed so that the am/pm and AM/PM indicators are lost. This is not optimal but so far I have noted no fatal breakage. The idea was to introduce the underlying infrastructure and ask both code and skin developers to accept and embrace the concept and use these formats in their code rather than hard coding specific formats.

DETAILS

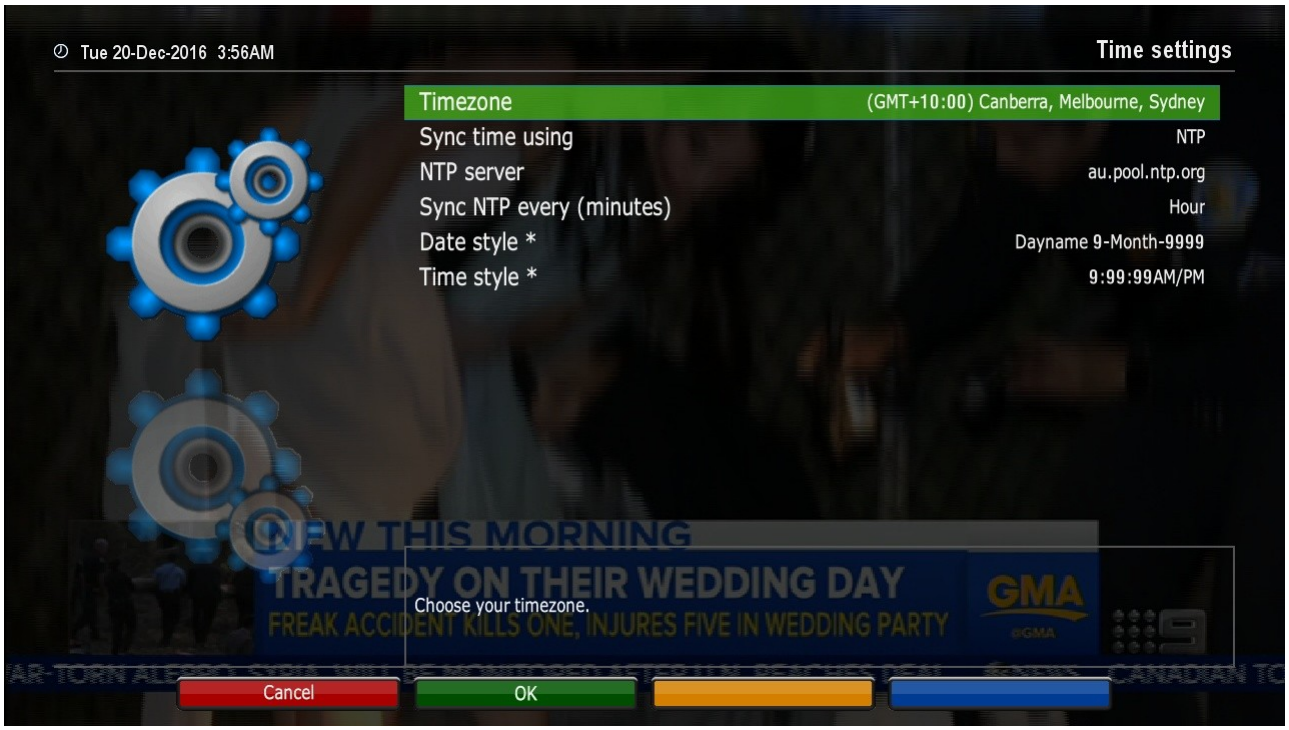
The proposal added code infrastructure for a number of pre coded date and time formats that can be used across Enigma2, plugins, converters and skins. The formats available are based, in part, on those available in the "Components/Converters/ClockToText.py" with a few additions. Each of the date formats have display options for European style DAY-MONTH-YEAR, American style MONTH-DAY-YEAR and ISO / International style YEAR-MONTH-DAY. The time formats have options for 24 hour clock, 12 hour clock with both lower case (am/pm) or upper case (AM/PM) displays.

People who prefer USA style dates will not like Enigma2 in its previous form. Is "11/12" really "11-Dec" or "12-Nov"? It depends on who you are and your background as to how you would read the date. By providing the different style options users, by their selection, can clearly distinguish the date displays in the UI. As shown above the latter versions of the date enable *everyone* to understand the correct date information from anywhere in the UI. Using this proposal coders need not worry who the users are or from where they come. Users can simply select their preferred language and the appropriate date and time formats and then everything should just work.

The idea was to introduce some standardised date and time formats that various developers can access and use rather than hard coding specific, and often differing, strftime() formats in their code. Users can use a simple interface to select the date and time formats they wish to see across all the screens on their UI.

A refactored and enhanced "Components/Converters/ClockToText.py" has been coded that uses these new formats and will allow skin developers to automatically access them. A few other core modules have also been adapted to support these new formats. Now that the proposal has been accepted I would welcome participation to implement this date / time change proposal across all Enigma2 code and all plugins.

The format selection, for end users, has been facilitated by an addition to the date set up menu in "setup.xml". The user is presented with an option to select a single date and time format from the options available (the date and time formats displayed are "config.usage.date.dayfull" and "config.usage.time.long" respectively, see below.). The code will then adapt the selected style to all the other forms of date and time. That is, only one style of the date or time will be selected and all the options will be derived from that selection. I believe that it would be too daunting and confusing to ask users to individually select each of the date and time options (though this can be programmed if required). A sample of the formatting selection UI is included below:



When I first communicated this proposal some concern was expressed with regard to compatibility with existing skins. I have adapted and improved the proposal by implementing a way for all the date and time options to be hidden until such time as a skin author updates their skin to be compatible with the new infrastructure. The new "setup.xml" entries are now conditional on flags that are triggered by a new "<parameter>" entry in the skin. The currently selected and active skin totally controls the feature. If the parameter is missing from the skin then the new date and time selection options will be suppressed. This should be the default case for *ALL* current skins across all the repositories.

For granularity the date and time infrastructure can be independently enabled and/or disabled in each skin. Feedback from team members was that the new 12 hour times were a problem. Now you can enable the new date display while suppressing the new time displays until such time as the skin authors decide to support the new expanded time formats. The idea behind the change was to maximise compatibility of the new

infrastructure without making any skin, supported or unsupported, obsolete or breaking *any* of the skins in any repository.

Only when a skin gets the new parameter will it start offering the new features. This is a skin by skin choice. The changes will appear and disappear based on whatever skin is currently loaded. No need to reboot just change the skin and restart the GUI and everything will just adapt / change.

NOTE: When the code changes were implemented some date formatting changes were forced - EVEN IF THE DATE CODE IS DISABLED. The nature of unifying the date formats means that if there are varying hard coded representations of a short date like "17/12" or "17 Dec" then all these displays become unified as the default version of a short date. By default this is "17 Dec" though the default can be changed in "Components/UsageConfig.py". (The change of default must be properly made such that all the various date and time format options must always remain synchronised.)

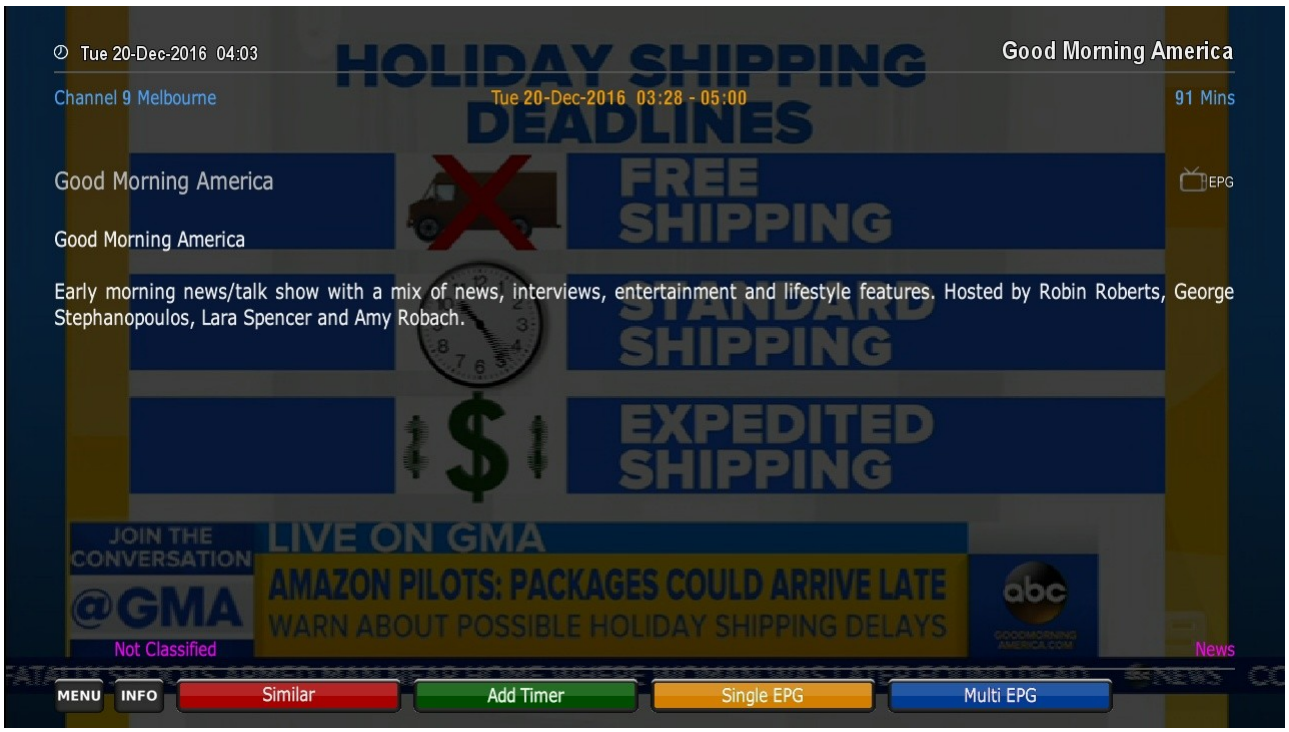
I have written a skin for the Beyonwiz and now OpenViX builds that makes extensive use of the new infrastructure. The skin is called "OverlayHD". What I did for OverlayHD was design all the screens such that there was lots of space for the date and time widgets. The design is such that this space is not obvious but allows for the variable time width formats.

There is *NO* extra special code for virtually anything! That was a major prerequisite and feature of my design. All you need to do in a skin is simply use the ClockToText converter and tell it what type of time or date display you want and it will work out how to deliver the required formats modified to the date or time style that the user chooses.

There is an "enhancement" built into the system. ;) The enhancement is in the EventTime and ClockToText converters. The enhancement is that the start and end times for an event can now be delivered via a single widget. The EventTime converter can be given the options "Times", "NextTimes" and "ThirdTimes" to give the current, next and later event times as a single tuple. The ClockToText converter now knows about this tuple and can format the times as required but in a single widget. This means that the two times can be formatted and aligned as desired by the skin writer without regard to the size of the date and time formats involved. Just make enough space for the single widget, align it as you desire and let the code do all the work.

Example 1:

The "Info" screen in OverlayHD looks like this:



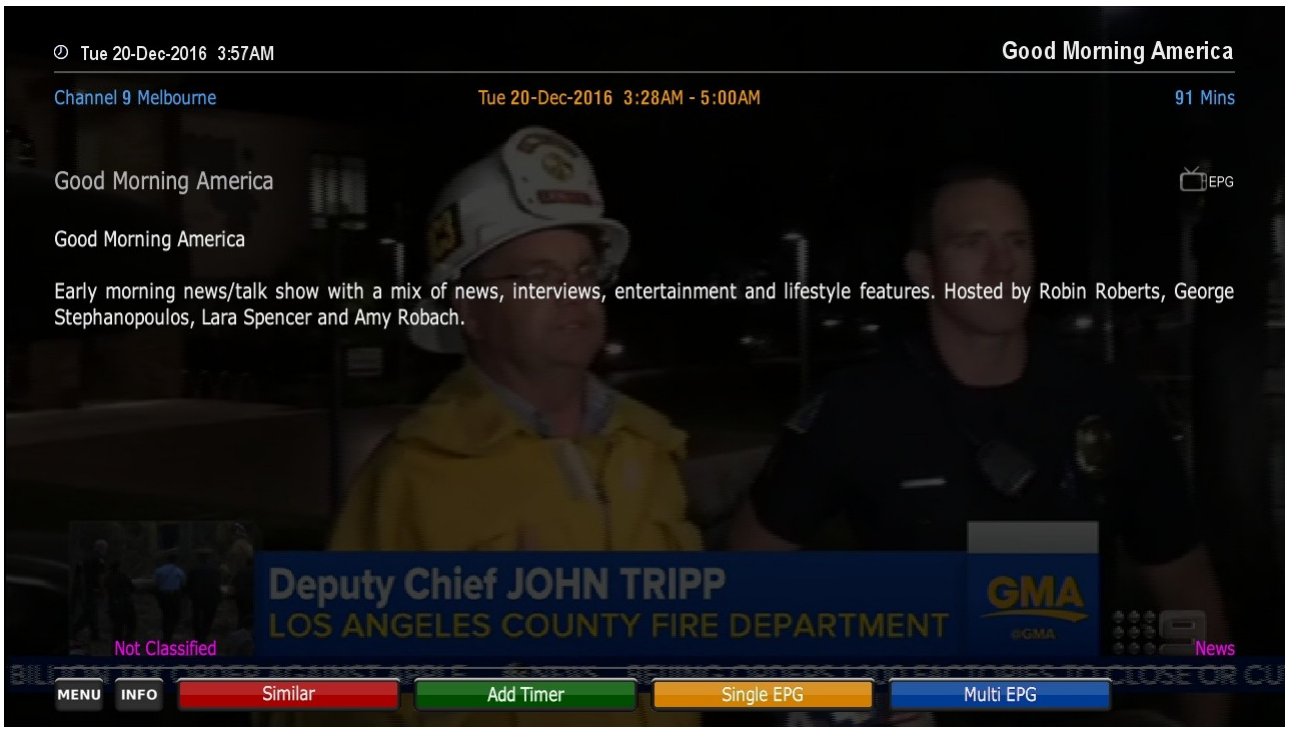
The code to produce this originally looked like:

```
<widget source="Event" render="Label" position="350,80" size="400,25"
backgroundColor="InfoBackground" font="InfoTimeFont;20" foregroundColor="InfoTimesNow"
halign="right" transparent="1">
     <convert type="EventTime">StartTime</convert>
     <convert type="ClockToText">Format:%a %e-%b-%Y  %H:%M -</convert>
</widget>
<widget source="Event" render="Label" position="760,80" size="100,25"
backgroundColor="InfoBackground" font="InfoTimeFont;20" foregroundColor="InfoTimesNow"
transparent="1">
     <convert type="EventTime">EndTime</convert>
     <convert type="ClockToText">Format:%H:%M</convert>
</widget>
```

This required skin based format descriptors and careful positioning of all the elements. With the new converters the code becomes:

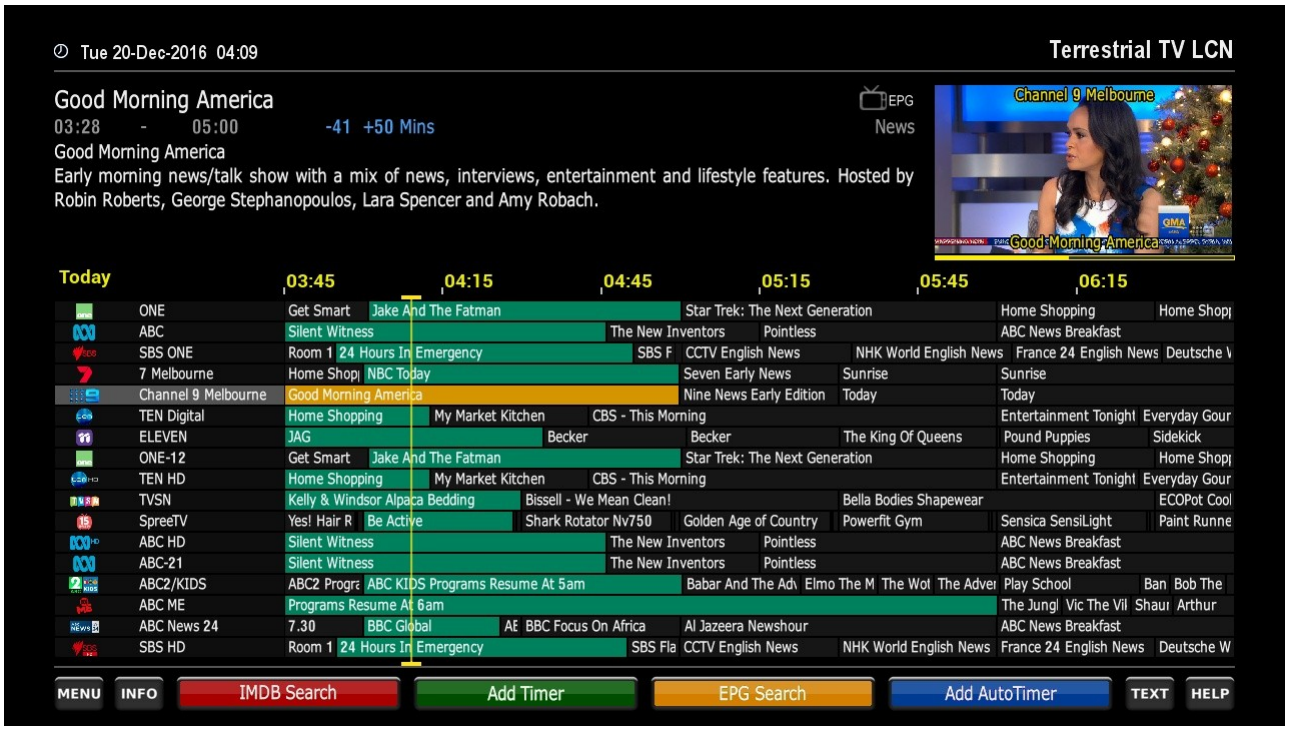
```
<widget source="Event" render="Label" position="365,80" size="500,25"
backgroundColor="InfoBackground" font="InfoTimeFont;20" foregroundColor="InfoTimesNow"
halign="center" transparent="1">
     <convert type="EventTime">Times</convert>
     <convert type="ClockToText">LongFullDate,Default</convert>
</widget>
```

In the earlier version the areas need to be appropriately sized and positioned to make the screen look correct. Note that we now have a single large and simple widget and all the heavy lifting is done by the converters. With the new code you just provide a width large enough to hold any of the user chosen styles and let the new formats and options do all the work for you. Whatever date or time styles the user selects to be used they are automatically applied and aligned by the single widget. A sample of the new 12 Hour version is as follows:



Example 2:

The "EPG" screens all feature start and end times for the event detail displays. The screens previously looked like:



but now looks like:

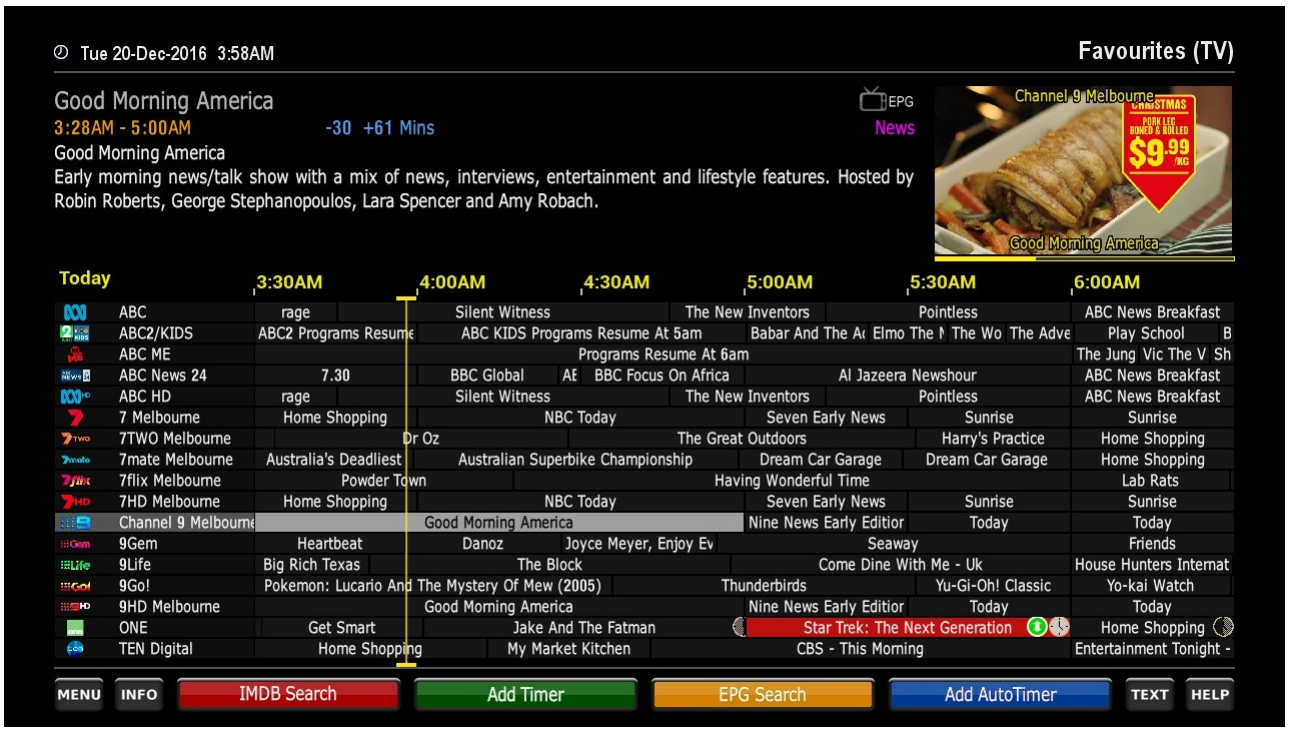
The old code to produce this looked like:

```
<widget source="Event" render="Label" position="0,30" size="80,25"
backgroundColor="EPGBackground" font="InfoTimeFont;20" foregroundColor="InfoTimesNow"
transparent="1">
      <convert type="EventTime">StartTime</convert>
      <convert type="ClockToText">Default</convert>
</widget>
<eLabel text="-" position="80,30" size="20,25" backgroundColor="EPGBackground"
font="InfoTimeFont;20" foregroundColor="InfoTimesNow" halign="center" transparent="1" />
<widget source="Event" render="Label" position="105,30" size="80,25"
backgroundColor="EPGBackground" font="InfoTimeFont;20" foregroundColor="InfoTimesNow"
halign="right" transparent="1">
      <convert type="EventTime">EndTime</convert>
      <convert type="ClockToText">Default</convert>
</widget>
```

Note that to allocate the potential space required for the two times there is a lot of space wasted in the display. It is not optimal. With the new converters the code becomes:

```
<widget source="Event" render="Label" position="0,30" size="180,25"
backgroundColor="EPGBackground" font="InfoTimeFont;20" foregroundColor="InfoTimesNow"
noWrap="1" transparent="1">
      <convert type="EventTime">Times</convert>
      <convert type="ClockToText">Default</convert>
</widget>
```

The times now automatically float to the correct place. Any styles can be used with plenty of space for all. Samples of the new 12 and 24 Hour versions are:



As a side note, in the screen images the date and time display on the banner used to be:

```
<ePixmap pixmap="icons/clock.png" position="0,4" size="14,14" alphatest="blend" />
<widget source="global.CurrentTime" render="Label" position="25,0" size="280,25"
borderColor="BannerBorder" borderWidth="1" font="ClockFont;20"
foregroundColor="BannerClock" transparent="1">
     <convert type="ClockToText">Format:%a %-e-%b-%Y  %R</convert>
</widget>
```

With the new converter is now simplified to:

```
<ePixmap pixmap="icons/clock.png" position="0,4" size="14,14" alphatest="blend" />
<widget source="global.CurrentTime" render="Label" position="25,0" size="280,25"
borderColor="BannerBorder" borderWidth="1" font="ClockFont;20"
```

```
foregroundColor="BannerClock" transparent="1">
     <convert type="ClockToText">LongFullDate</convert>
</widget>
```

The date and time simply reformat as specified by the date and time configuration options.

TECHNICAL DETAILS

SKIN PARAMETER

To enable the new date and time formats a skin author simply needs to add a tag like the following "<parameter>" tag to the "<parameters>" section of their skin:

```
<skin>
     <parameters>
          <parameter name="AllowUserDatesAndTimes" value="1,0" />
     </parameters>
</skin>
```

The first number in the value controls the new date formats. The second number in the value controls the new time formats. In both cases a value of "0" disables the feature and a value of "1" enables the feature.

The default if the <parameter> tag is not found is:

```
<parameter name="AllowUserDatesAndTimes" value="0,0" />
```

Given that most skins do not yet support the AM/PM times but the new dates appear to work okay the following parameter could be used:

```
<parameter name="AllowUserDatesAndTimes" value="1,0" />
```

When a skin has been adapted to support the wider times then the following form should be used:

```
<parameter name="AllowUserDatesAndTimes" value="1,1" />
```

CLOCKTOTEXT CONVERTER

With the new "ClockToText" converter developers have the following options (which can be expanded if required!):

OPTION	DISPLAY FORMAT
Date	Sunday 18 December 2016
Default	16:28
Full	Sun 18 Dec 16:28
FullDate	Sun 18 December 2016
LongDate	Sunday 18 December
LongFullDate	Sun 18 Dec 2016 16:28
Mixed	16:28:42 for 24 hour clocks; 4:28am for 12 hour clocks
ShortDate	Sun 18 Dec
ShortFullDate	Sun 18 Dec 2016
VFD	18Dec16:28
VFD08	16:28
VFD12	18Dec16:28
VFD14	18 Dec 16:28

VFD18 ..... Sun 18 Dec 16:28
WithSeconds..... 16:28:42

These options are constructed from the primatives defined in the "UsageConfig.py" code as described below.

The "AsLength", "AsLengthHours", "AsLengthSeconds", "Format", "InMinutes" and "Timestamp" options still exist and are unchanged by the new system.

The OpenViX "NoSpace" and "Proportional" options are no longer needed but are accepted, for legacy compatibility, and ignored.

The following options have been added:

OPTION	**PURPOSE**
"Parse"	This option MUST be first and is immediately followed by the new parsing separator character. The default separators are ";" and ",".
"Separator"	This is the separator string that will be used between the times in the "EventList" tuple.

If the ClockToText converter does not have enough format option to match the incoming tuple then the last format option is reused until all times in the tuple have been displayed.

Example 1:

```
<convert type="ClockToText"></convert> or <convert type="ClockToText" /> ......... "16:28"
```

This is the same as:

```
<convert type="ClockToText">Default</convert> -> "16:28"
```

Example 2:

```
<convert type="ClockToText">LongFullDate</convert> ............... "Sun 18 Dec 2016  16:28"
```

Example 3:

```
<convert type="EventTime">Times</convert>
<convert type="ClockToText">Default,Default</convert> ................... "16:30 - 17:30"
```

This can also be written as:

```
<convert type="EventTime">Times</convert>
<convert type="ClockToText">Default</convert> ............................ "16:30 - 17:30"
```

Example 4:

```
<convert type="EventTime">Times</convert>
<convert type="ClockToText">Separator ~ ,Default</convert> ............... "16:30 ~ 17:30"
```

Example 5:

```
<convert type="EventTime">Times</convert>
<convert type="ClockToText">LongFullDate,Default</convert>
     ................................................ "Sun 18 Dec 2016  16:30 - 17:30"
```

Example 6:

```
<convert type="EventTime">NextTimes</convert>
<convert type="ClockToText">Parse=Separator-,#;-=LongFullDate=Default</convert>
     ..................................................... "Sun 18 Dec 2016  17:30-,#;-18:30"
```

All these examples are based on the default date and time styles. The actual displays will adjust based on the style selected by the user.

When designing or modifying skins developers should allocate about 50% to 60% more widget width for each of the time fields than previously designed. If the widget isn't made wider the 12 hour times will be truncated. The AM/PM or am/pm will be outside the widget display area and simply dropped / hidden. This is the most significant issue / risk with the new system infrastructure. That is why I suggested for legacy skins the option to enable new dates but not the new times may be a workable option.

NOTE: The original "VFD" format was inconsistent and became "18Dec16:28". That is, the date is now shown *before* the time.

EVENTTIME CONVERTER

For the "EventTime" converter developers have the following new options:

```
<convert type="EventTime">Times</convert>
<convert type="EventTime">NextTimes</convert>
<convert type="EventTime">ThirdTimes</convert>
```

These output the start and end times for the nominated event as a tuple that can be used by the "ClockToText" converter.

The "Duration", "Elapsed", "EndTime", "NextEndTime", "NextStartTime", "Progress", "Remaining", "StartTime", "ThirdEndTime", "ThirdStartTime", "VFDElapsed" and "VFDRemaining" options still exist and are unchanged by the new system.

The "NextDurartion" and "ThirdDurartion" options in the original converter are miss-spelt and were decoded but the code to support the options was never written! I have added the correctly spelt options "NextDuration" and "ThirdDuration" and coded up support for these options.

NEW PYTHON CONFIG VARIABLES

The new date and time formats are defined in a series of "config.usage" variables that are defined in "Components/UserConfig.py". The variables and their defaults are:

Date Formats:

VARIABLE	DEFAULT DISPLAY	FORMAT EXPLANATION
config.usage.date.dayfull	Dayname 9 Month 9999	Day and month names in full words
config.usage.date.shortdayfull	Day 9 Month 9999	Day name in short word and month name full word
config.usage.date.daylong	Day 9 Mon 9999	Day and month names in short words
config.usage.date.dayshortfull	Dayname 9 Month	Day and month names in full words and no year
config.usage.date.dayshort	Day 9 Mon	Day and month names in short words and no year

config.usage.date.daysmall.....Day 9 ..... Day name in short word and no month or year
config.usage.date.full .....9 Month 9999..... Month name in full word and no day
config.usage.date.long.....9 Mon 9999..... Month name in short word and no day
config.usage.date.short.....9 Mon ..... Month name in short word and no day or year
config.usage.date.compact .....9Mon ..... Month name in short word and no day or year or spacing

Time Formats:

config.usage.time.long ..... 99:99:99 .....Time in hours minutes and seconds
config.usage.time.mixed ..... 99:99:99 / 99:99am .....Time in hours minutes and seconds for 24 hour clocks; hours:minutes for 12 hour clocks
config.usage.time.short..... 99:99 .....Time in hours and minutes only

Status Settings:

config.usage.date.enabled ... Set by the first "AllowUserDatesAndTimes" parameter value
config.usage.time.enabled ... Set by the second "AllowUserDatesAndTimes" parameter value
config.usage.time.wide ..... Set to true if an "AM/PM" or "am/pm" style is chosen (see below)

Each of the date formats is available in a number of styles. Using "config.usage.date.dayfull" as the example, the following styles are available:

DISPLAY STYLE	**STRFTIME()**	**STYLE EXPLANATION**
Dayname 99 Month 9999.....	%A %d %B %Y"	Day month year date with full word day and month and a two digit day number with leading zero
Dayname 9 Month 9999"	%A %-e %B %Y"	Day month year date with full word day and month and a single or two digit day number
Dayname 9-Month-9999	%A %-e-%B-%Y	Day month year date with full word day and month and a single or two digit day number with hyphen separators
Dayname 9/99/9999"	%A %-e/%m/%Y.....	Day month year date with full word day, two digit month number and one or two digit day number with slash separators
Dayname Month 99 9999.....	%A %B %d %Y	Month day year date with full word day and month and a two digit day number with leading zero
Dayname Month 9 9999.....	%A %B %-e %Y	Month day year date with full word day and month and a single or two digit day number
Dayname Month-9-9999	%A %B-%-e-%Y	Month day year date with full word day and month and a single or two digit day number with hyphen separators
Dayname 9/9/9999"	%A %-m/%-e/%Y....	Month day year date with full word day, one or two digit month number and one or two digit day number with slash separators
Dayname 9999 Month 99.....	%A %Y %B %d	Year month day date with full word day and month and a two digit day number with leading zero

Dayname 9999 Month 9.....	%A %Y %B %-e	Year month day date with full word day and month and a single or two digit day number
Dayname 9999-Month-99	%A %Y-%B-%d	Year month day date with full word day and month and a two digit day number with hyphen separators
Dayname 9999-Month-9	%A %Y-%B-%-e.....	Year month day date with full word day and month and a single or two digit day number with hyphen separators
Dayname 9999/99/9.....	%A %Y/%m/%-e.....	Year month day date with full word day, two digit month number and one or two digit day number with slash separators

Each of the time formats is available in a number of styles. Using "config.usage.time.long" as the example, the following styles are available:

DISPLAY STYLE	**STRFTIME()**	**STYLE EXPLANATION**
99:99:99.....	%T	24 hour time
99:99:99AM/PM	%I:%M:%S%p.....	12 hour time with two digit hour and upper case AM/PM
9:99:99AM/PM	%-I:%M:%S%p.....	12 hour time with one or two digit hour and upper case AM/PM
99:99:99am/pm.....	%I:%M:%S%P	12 hour time with two digit hour and lower case am/pm
9:99:99am/pm.....	%-I:%M:%S%P	12 hour time with one or two digit hour and lower case am/pm

CODE EXAMPLES

For the purposes of this section I will use code examples from the OE-Alliance "EPGSearch.py" plugin and the Beyonwiz version of "EpgList.py". The "EpgList.py" on the Beyonwiz has been modified such that for every event the first column of the display is not the day but the day and date and the second column is not the date and time but the start and end times. This can be seen in the images associated with the examples above.

In the "EPGSearch.py" plugin when building an event line the following code was used:

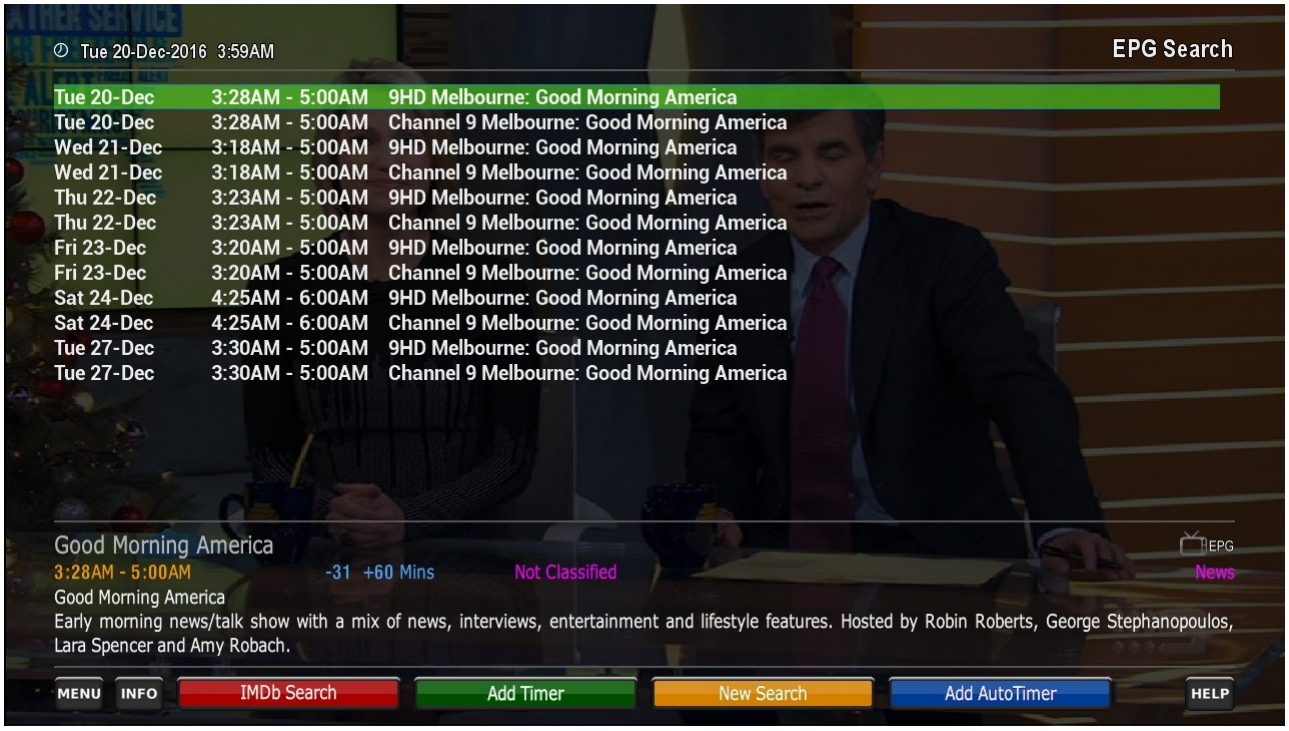
```
res = [
     None, # no private data needed
     (eListboxPythonMultiContent.TYPE_TEXT, r1.x, r1.y, r1.w, r1.h, 0,
RT_HALIGN_LEFT|RT_VALIGN_CENTER, strftime("%a", t)),
     (eListboxPythonMultiContent.TYPE_TEXT, r2.x, r2.y, r2.w, r2.h, 0,
RT_HALIGN_LEFT|RT_VALIGN_CENTER, strftime("%e/%m, %H:%M", t))
]
```

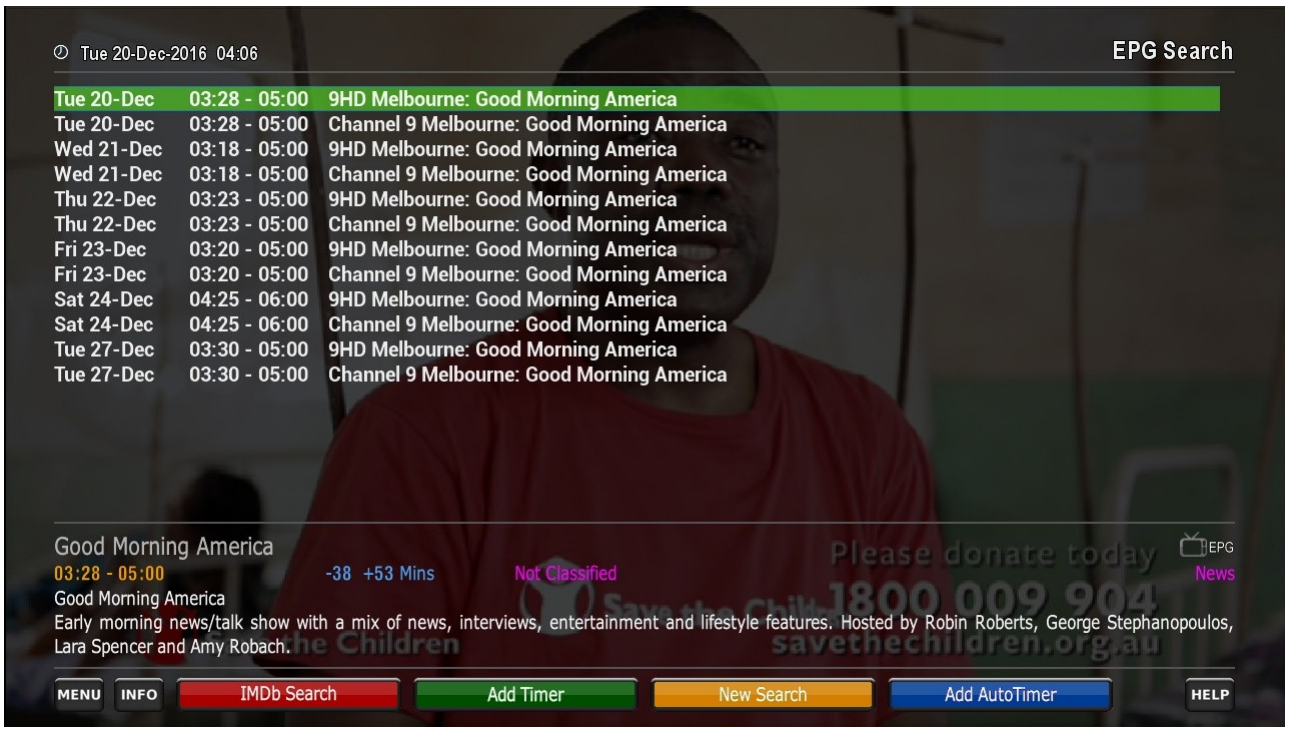
This has now been rewritten as:

```
if hasattr(self, "showend"):
     et = localtime(beginTime + duration)
     align = RT_HALIGN_RIGHT
     if hasattr(config.usage, "time"):
          weekday = strftime(config.usage.date.dayshort.value, t)
          datetime = "%s - %s" % (strftime(config.usage.time.short.value, t),
strftime(config.usage.time.short.value, et))
     else:
          weekday = strftime("%a %d %b", t)
          datetime = "%s ~ %s" % (strftime("%H:%M", t), strftime("%H:%M", et))
else:
     align = RT_HALIGN_LEFT
     weekday = strftime("%a", t)
     if hasattr(config.usage, "time"):
          datetime = "%s, %s" % (strftime(config.usage.date.short.value, t),
strftime(config.usage.time.short.value, t))
     else:
```

```
            datetime = strftime("%e/%m, %H:%M", t)
res = [
      None,  # no private data needed
      (eListboxPythonMultiContent.TYPE_TEXT, r1.x, r1.y, r1.w, r1.h, 0, RT_HALIGN_LEFT |
RT_VALIGN_CENTER, weekday),
      (eListboxPythonMultiContent.TYPE_TEXT, r2.x, r2.y, r2.w, r2.h, 0, align |
RT_VALIGN_CENTER, datetime)
]
```

In the revision the hard coded strings were replaced with their "config.usage.x.y" equivalents. This is a plugin and there can be no way to predict if these new options are available in the hosting build. To protect the plugin it tests to see if the "config.usage.time" items have been defined via the hasattr() function. If so, use them, if not fall back to the original hard coded values. Samples of the new 12 and 24 Hour versions are in the following images respectively:





Notes:

1. The "_(strftime("%a", t)" in the original code was inappropriate as strftime is already language corrected and has been corrected by the above change.
2. The r2 and r3 rectangle heights were defined as the r1 height. This worked but is also incorrect and has been corrected.
3. To save code I only test for the existence of the "config.usage.time" definition. This could equally have been for the "config.usage.date" definition. With this revised proposal the date and time changes can be independently controlled. That said both definitions should exist. The enable and disable code simply forces the unused date or time values back to their default values without regard to what the user may have selected and/or saved their preferred values.
4. The attribute "showend" is a special flag that indicates that this plugin is being called from a Beyonwiz (or other compatible build) where the event lines have been restructured so that the start and end times for each event are listed.

For the next code example let us look at the proposed change for the Beyonwiz version of "Components/EpgList.py". As mentioned before the Beyonwiz version of this EPG code has a modified layout. The modification to use both the start and end event times in a single column provides me with a perfect way to show what could/should be done to cope with the variable size of the time strings.

In "EpgList.py" the EPG timeline tick times were created using the following code:

```
ttime = localtime(time_base + x * timeStepsCalc)
if (self.type == EPG_TYPE_GRAPH and config.epgselection.graph_timeline24h.value) or
(self.type == EPG_TYPE_INFOBARGRAPH and config.epgselection.infobar_timeline24h.value):
     timetext = strftime("%H:%M", ttime)
else:
     timetext = strftime("%-I:%M%P", ttime)
```

This has now been rewritten as:

```
timetext = strftime(config.usage.time.short.value, localtime(time_base + x *
timeStepsCalc))
```

The code is simplified yet gains the ability to change the time formats.

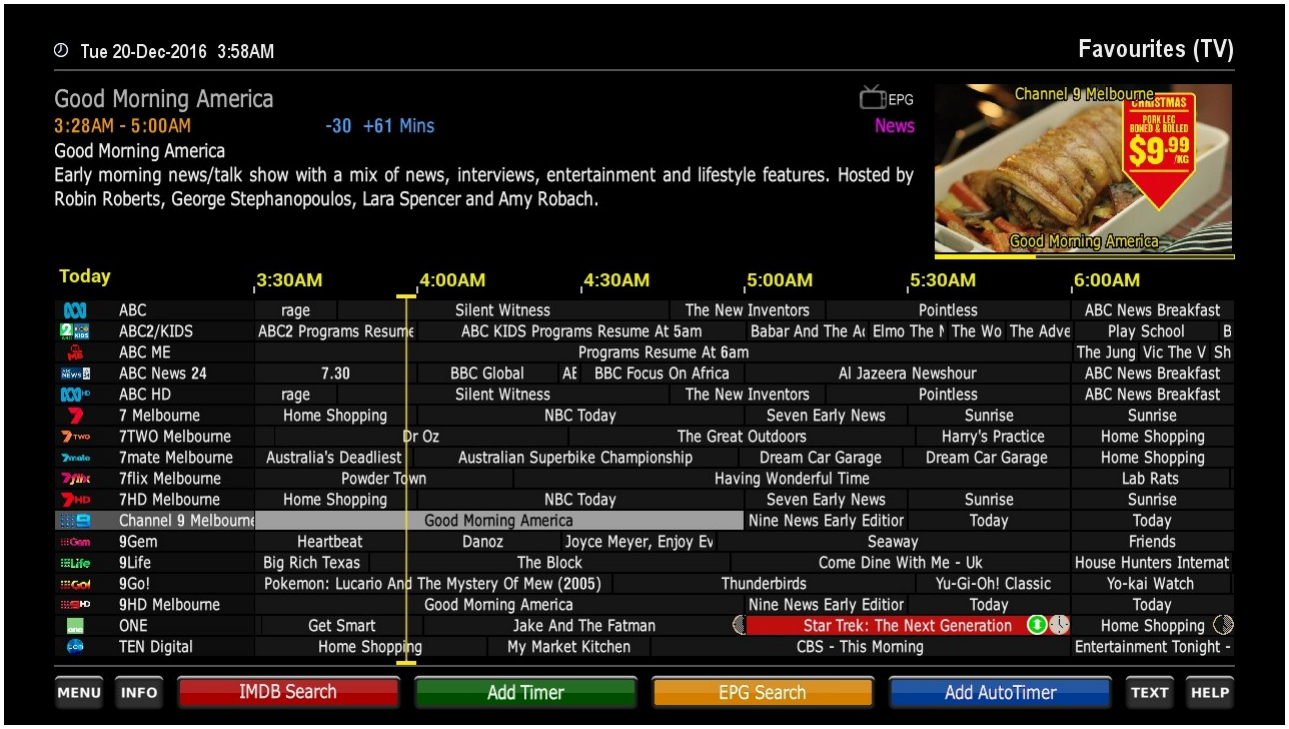
Similarly in "EpgList.py" the EPG grid time display was created using the following code:

```
if nowTime.tm_year == begTime.tm_year and nowTime.tm_yday == begTime.tm_yday:
	datestr = _("Today")
else:
	if serviceWidth > 179:
		datestr = strftime("%A %d %B", begTime)
	elif serviceWidth > 139:
		datestr = strftime("%a %d %B", begTime)
	elif serviceWidth > 129:
		datestr = strftime("%a %d %b", begTime)
	elif serviceWidth > 119:
		datestr = strftime("%a %d", begTime)
	elif serviceWidth > 109:
		datestr = strftime("%A", begTime)
	else:
		datestr = strftime("%a", begTime)
```

This has now been rewritten as:

```
if nowTime.tm_year == begTime.tm_year and nowTime.tm_yday == begTime.tm_yday:
	datestr = _("Today")
else:
	if serviceWidth > 179:
		datestr = strftime(config.usage.date.daylong.value, begTime)
	elif serviceWidth > 129:
		datestr = strftime(config.usage.date.dayshort.value, begTime)
	elif serviceWidth > 79:
		datestr = strftime(config.usage.date.daysmall.value, begTime)
	else:
		datestr = strftime("%a", begTime)
```

Again the date and times displayed can use the configuration variables to reflect the user's wishes. Samples of the new 12 and 24 Hour versions are as follows:



Also in "EpgList.py" when building the event line format rectangles for the Single and Similar EPG screens the following code was used:

```
fontSize = self.eventFontSizeSingle + config.epgselection.enhanced_eventfs.value
self.weekday_rect = Rect(0, 0, int(fontSize * 5.8), height)
self.datetime_rect = Rect(self.weekday_rect.width(), 0, int(fontSize * 6.5), height)
self.descr_rect = Rect(self.datetime_rect.left() + self.datetime_rect.width(), 0, width -
self.datetime_rect.left() - self.datetime_rect.width(), height)
```

The Single EPG entries were drawn as:

```
res = [
     None,  # no private data needed
     (eListboxPythonMultiContent.TYPE_TEXT, r1.x, r1.y, r1.w, r1.h, 0, RT_HALIGN_LEFT |
RT_VALIGN_CENTER, strftime("%a %d %b", t)),
     (eListboxPythonMultiContent.TYPE_TEXT, r2.x, r2.y, r2.w, r2.h, 0, RT_HALIGN_LEFT |
RT_VALIGN_CENTER, "%s ~ %s" % (strftime("%H:%M", t), strftime("%H:%M", et)))
]
```

The Similar EPG entries were drawn as:

```
res = [
     None,  # no private data needed
     (eListboxPythonMultiContent.TYPE_TEXT, r1.x, r1.y, r1.w, r1.h, 0, RT_HALIGN_LEFT |
RT_VALIGN_CENTER, strftime("%a %d %b", t)),
     (eListboxPythonMultiContent.TYPE_TEXT, r2.x, r2.y, r2.w, r2.h, 0, RT_HALIGN_LEFT |
RT_VALIGN_CENTER, "%s ~ %s" % (strftime("%H:%M", t), strftime("%H:%M", et)))
]
```

These code fragments can be rewritten as:

```
fontSize = self.eventFontSizeSingle + config.epgselection.enhanced_eventfs.value
dateW = int(fontSize * 5.4)
timesW = int(fontSize * 5.6)
dateLeft, dateWidth, timesLeft, timesWidth, descOffset =
skin.parameters.get("EPGSingleEPGColumnFormats", (0, dateW, dateW + 10, timesW, 20))
if config.usage.time.wide.value:
     timesWidth = int(timesWidth * 1.5)
self.weekday_rect = Rect(dateLeft, 0, dateWidth, height)
```

```
self.datetime_rect = Rect(timesLeft, 0, timesWidth, height)
descLeft = timesLeft + timesWidth + descOffset
self.descr_rect = Rect(descLeft, 0, width - descLeft, height)
self.showend = True
```

and

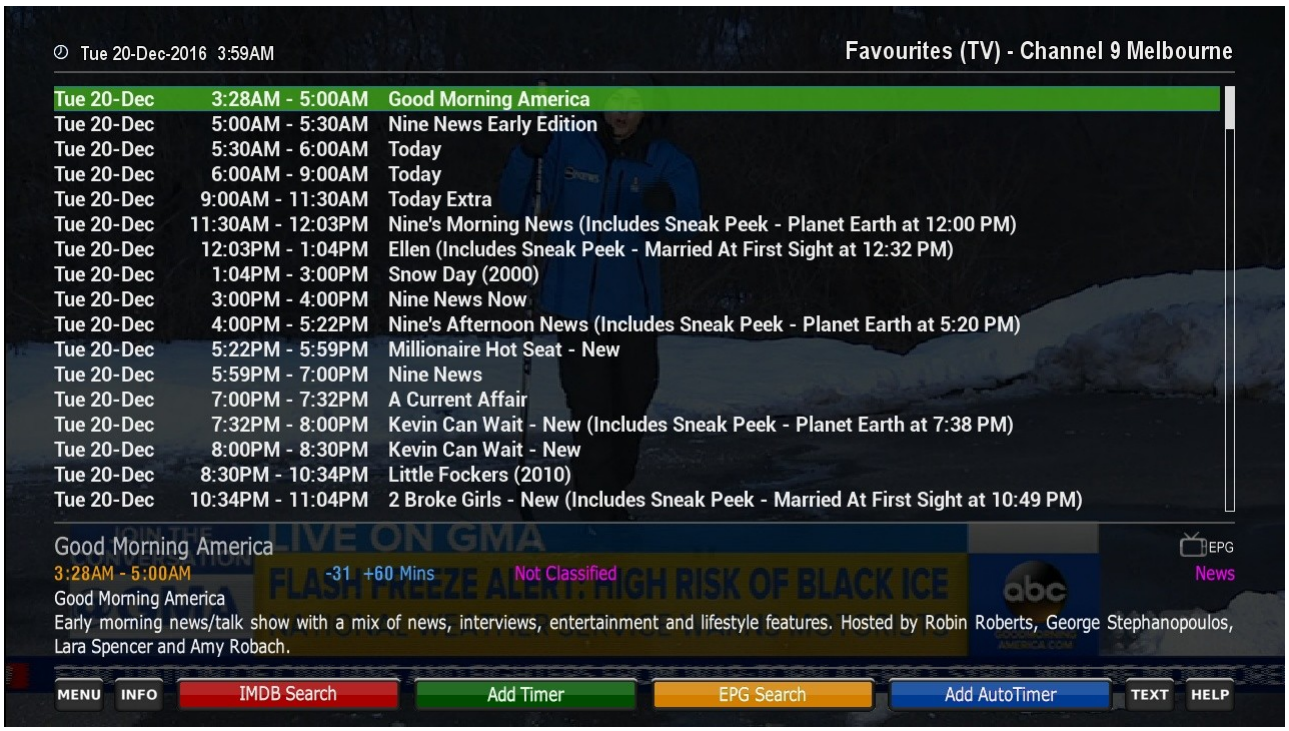
```
res = [
     None,  # no private data needed
     (eListboxPythonMultiContent.TYPE_TEXT, r1.x, r1.y, r1.w, r1.h, 0, RT_HALIGN_LEFT |
RT_VALIGN_CENTER, strftime(config.usage.date.dayshort.value, t)),
     (eListboxPythonMultiContent.TYPE_TEXT, r2.x, r2.y, r2.w, r2.h, 0, RT_HALIGN_CENTER
| RT_VALIGN_CENTER, "%s - %s" % (strftime(config.usage.time.short.value, t),
strftime(config.usage.time.short.value, et)))
]
```

and

```
res = [
     None,  # no private data needed
     (eListboxPythonMultiContent.TYPE_TEXT, r1.x, r1.y, r1.w, r1.h, 0, RT_HALIGN_LEFT |
RT_VALIGN_CENTER, strftime(config.usage.date.dayshort.value, t)),
          (eListboxPythonMultiContent.TYPE_TEXT, r2.x, r2.y, r2.w, r2.h, 0,
RT_HALIGN_CENTER | RT_VALIGN_CENTER, "%s - %s" % (strftime(config.usage.time.short.value,
t), strftime(config.usage.time.short.value, et)))
]
```

To help developers identify when more screen space may be required for a time display a Boolean "config.usage.time.wide" has been created. When True the currently active time format selected uses an "am/pm" or "AM/PM" format and will require more screen space. In the format rectangle code a scaling factor is used to set the width of the column. If the two dates used one of the 12 hour clock formats then more space will be required. If "config.usage.time.wide" is true then the scaling factor is increased to account for the new wider space requirement. (In the previous code there was padding in the date and time widths to keep the columns apart. This has now been changed such that the rectangles are separated by fixed protective 20 pixel padding. The time rectangles can now be sized to properly reflect the content and not have to worry about the padding.)

The EPG entry drawing code samples have had the fixed date and time formatting strings replaced by the proposed variables. Samples of the new 12 and 24 Hour versions are in the following images:



I believe that all these changes can be implemented in all external plugins in a legacy sensitive way, as demonstrated, such that builds that don't want to adopt the changes can still be fully supported.

CONCLUSION

For all the Enigma2 builds there is a significant legacy of code and skins that needs to be preserved and protected. The proposal outlined what I believe to be a conservative approach to making a significant but beneficial change in unifying and standardising the date and time presentations across the Enigma2 UI.

--- End ---