CUNY Academic Commons - Bug #4438

Events Calendar - Export Recurring Events

2015-08-16 12:45 PM - scott voth

Status:	Assigned	**Start date:**	2015-08-16
Priority name:	Normal	**Due date:**	
Assignee:	Daniel Jones	**% Done:**	0%
Category name:	Events	**Estimated time:**	0.00 hour
Target version:	Future release		

Description

Member Raffi Khatchadourian reports that when he uses the export functionality for a recurrent event, only one instance is copied.

"I tried it on this calendar: http://commons.gc.cuny.edu/groups/cuny-committee-on-academic-technology-62702785/events/cuny-cat-meeting/. I actually just opened the downloaded file in a text editor and noticed no reoccurrences. "

I tried this in cdev - but got a permissions error, so can't say if I can reproduce.

I also tried with external calendars, with mixed results. In MS Outlook, the recurring event worked as expected. In Google Calendar (not exactly sure if I am doing it correctly) but when I imported the ICS file - I only got one event, rather that the whole series.

History

#1 - 2015-08-17 11:08 AM - Boone Gorges

- Status changed from New to Assigned

- Assignee set to Daniel Jones

- Target version set to Future release

Dan, can I ask you to take a look? First, try to reproduce. If you are able to do so, look around the web to see if this is a known issue - either specifically to Event Organiser, or more generally with differences between the way recurring events are represented in iCal files. This could be a place where we could pass a patch upstream to Event Organiser. Thanks!

#2 - 2015-08-26 07:54 AM - Stephen Real

I wanted to let you know that I tested downloading are recurring event into Mac's iCalendar and it worked perfectly.

#3 - 2016-02-18 04:13 PM - Daniel Jones

Yeah wow something very strange is happening here.

From what I was able to tell through testing, there are two separate, but possibly (likely?) related issues:

1. When exporting one recurring event (I tested a weekly and a monthly event), the event is exported as "recurring", but the initial start date is the last in the series. For example, I set up a monthly event that was supposed to start today and go until June, but the recurring event that was exported started in June and went until June. The same happened for weekly recurring events.
2. When exporting a whole Calendar, including recurring events, it exported each of the events in the series as a recurring event, with the same UID, which meant that the last one was the only one saved in the end. So it had basically the same effect. I was able to go in and fix them both pretty easily manually.

I don't think this is an issue about Google Calendar vs. iCal, but I also wasn't able to find another person with this exact issue in the Event Organiser support forums, etc. Should I dive in and try to find what I can? From the changelog, seems like there have been quite a few errors with the iCal feeds in particular. Let me know what you think!

#4 - 2016-02-18 10:30 PM - Boone Gorges

Thanks for digging, Dan. This sounds like an interesting issue. The first thing I'd suggest is getting a fresh installation of Event Organiser, **without** BP Event Organiser on top, to see if you can reproduce the issue there. First, try with whatever version of event-organiser we use on the Commons (2.13.7) to verify that it's not a problem coming from our code. Then, try with the latest version of event-organiser, which is now in the 3.x series https://github.com/stephenharris/Event-Organiser/ It's possible that this has been fixed already.

If the bug is still occurring, please do take some time to try fixing it in Event Organiser. I'm sure the author would be very glad for us to help diagnose and fix a problem like this. Feel free to run your suggested fix by me or Ray before sending a PR to Stephen. Thanks again!

#5 - 2016-02-27 12:37 PM - Daniel Jones

This was a fun one!

So it looks like it is a bug even in the most recent version of the Event Organiser plugin. The issue, as far as I can tell, is with how the plugin handles recurring events at the most basic level, which is by having multiple event rows in the database have the same "post_id". This means that when the database is queried for events by post_id, you can't guarantee which of the rows/occurrences you're going to get. So when events are queried and requested to be grouped by series, you can't guarantee that the one you'll get is the first one, which explains why the exported files often start with the last or one of the last occurrences.

I wasn't sure how I could change the behavior of their functions hooked the pre_get_posts to fix this, so instead I took an easier route that might not really be usable: when you start an export, the plugin uses query_posts (there's a comment saying that this might be one of the only justifiable uses of query_posts...) to get a list of events. I run this code to modify the resulting $wp_query global, just making sure that the occurrences included as stand-ins for reoccurring events are the earliest ones. I included their surrounding code, too:

```
//Exporting events
//mmm... maybe finally a legitimate use of query_posts
query_posts(array(
    'post_type'=>'event',
     'showpastevents'=>true,
     'group_events_by'=>'series',
     'posts_per_page'=>-1,
));
// We can't be sure that the occurrences returned for recurring events are the earliest ones, so do that now.
global $wp_query;
$num_queried_events = count($wp_query->posts);
for( $index = 0; $index < $num_queried_events; $index++ ) {
     $queried_event = $wp_query->posts[$index];
     // Don't do anything if this isn't a reocurring event, or if the first occurrence is already in place
     if ( isset( $queried_event->event_occurrence ) && $queried_event->event_occurrence != 0 ) {
         $all_occurrences_query = new WP_Query( array( 'post_type' => 'event', 'p' => $queried_event->ID ) );
        $occurrences = $all_occurrences_query->posts;
        // Don't do anything if there's only one event in this series
        if ( count( $occurrences ) <= 1 ) {
            continue;
         }
        // Find the earliest occurrence, that isn't in the past, for this series
        $earliest_occurrence = $occurrences[0];
         foreach( $occurrences as $occurrence ) {
            if ( $occurrence->event_occurrence < $earliest_occurrence->event_occurrence &&
                    strtotime( $occurrence->StartDate ) > time() ) {
                $earliest_occurrence = $occurrence;
            }
        }
         // Replace the queried event with its earliest recurrence
        $wp_query->posts[$index] = $earliest_occurrence;
    }
}
$this->get_export_file();
```

I tested it and it works, including with exporting a calendar of mixed recurring and non-recurring events. I see two glaring issues:
1. Definitely is not recommended to just mess with $wp_query like that..but maybe it's okay here because we aren't changing the number of queried items, just swapping them out for ones that are almost identical?
2. My test for weeding out occurrences in the past is pretty sloppy. I don't know a whole lot about date manipulation and comparison - any recommendations there?

Let me know what you think about this.

#6 - 2016-02-29 10:54 PM - Boone Gorges

Thanks for looking into this, Dan!

I've been looking at your proposed fix, but I'm having a hard time reproducing the bug. Both in the case of global calendar feeds (eg example.com/feed/eo-events) and single-event ical exports (example.com/events/event/my-event/feed/eo-events), the recurrence feature seems to be working properly. The relevant parts of the .ical seem to be as follows:

```
DTSTART;TZID=Europe/Warsaw:20160301T043000
DTEND;TZID=Europe/Warsaw:20160301T053000
RRULE:FREQ=DAILY;INTERVAL=1;UNTIL=20160331T023000Z
```

DTSTART and DTEND correctly specify the **first** item in the series, while RRULE describes the recurrence interval and end date.

I've been trying to get a better grasp of what EO is doing here, and here's how I understand it:
- In eventorganiser_pre_get_posts(), a bunch of query clauses are filtered to add EO-specific clauses. In our case, we're specifically concerned about 'group_events_by=series'.
- First, eventorganiser_join_tables() joins against the result of a subquery of wp_eo_events. The subquery is ordered by StartDate ASC, StartTime ASC, so it should be correct.

- Second, eventorganiser_event_groupby() sets the GROUP BY clause of WP_Query to wp_posts.ID.

It sounds to me like you're correctly seeing the second of these, but the first isn't working correctly. I have a feeling that something weird is happening with the way MySQL is grouping results. See eg http://stackoverflow.com/questions/13459516/php-mysql-group-by-to-get-latest-record-not-first-record - the impression here is that "the result is unspecified" if the SELECT fields are not the same as those listed in the GROUP BY clause.

Maybe I'm trying something different from what you're doing?

I have a feeling that something like your suggested fix is probably OK, but I feel uneasy moving forward with it until we have a better sense of what's going on under the hood (ie, why GROUP BY is acting unpredictably). Can you try to wrap your head around this?

#7 - 2016-03-04 03:39 PM - Daniel Jones

I think you were right about all of this Boone. I got to do some serious brushing-up on my SQL and came up with what I think is a solution.

First - I think it's worth noting that "GROUP BY" isn't really built for this kind of use. It's better for creating aggregate columns.

Second - looks like this is a common frustration among SQL users, and there's a lot out there about the best way to do "group-wise maximums". We're trying to do something relatively unusual here, looking for a "group-wise minimum". I was able to adapt one of the recommended techniques from the MySQL documentation, which in our case means using a nested JOIN. Here's how I change the eventorganiser_join_tables function.

The original is:

```
LEFT JOIN
( SELECT * FROM {$wpdb->eo_events} ORDER BY {$wpdb->eo_events}.StartDate ASC, {$wpdb->eo_events}.StartTime ASC
 )
AS {$wpdb->eo_events} ON $wpdb->posts.id = {$wpdb->eo_events}.post_id
```

The version that seems to be working for me is:

```
LEFT JOIN
( SELECT events1.* FROM {$wpdb->eo_events} events1
JOIN
( SELECT post_id, MIN(event_occurrence) as min_occurrence
FROM wp_eo_events
GROUP BY post_id
) AS events2
ON events1.post_id = events2.post_id AND events1.event_occurrence = events2.min_occurrence
ORDER BY events1.StartDate ASC, events1.StartTime ASC
)
AS {$wpdb->eo_events}
ON $wpdb->posts.id = {$wpdb->eo_events}.post_id
```

Here's the MySQL documentation I adapted it from: http://dev.mysql.com/doc/refman/5.7/en/example-maximum-column-group-row.html

Let me know what you think about it!

#8 - 2016-03-07 02:11 PM - Boone Gorges

- File eo-recurrences.patch added

Awesome! Thanks for continuing to dig into this, Dan.

This looks like it's getting close. What you've proposed should reliably ensure that the StartDate of the event matches the earliest occurrence, as long as the occurrence with the lowest "event_occurrence" is, in fact, the earliest one. But what if it's not? I spent some time trying to write a unit test that described the bug, and I was only able to reproduce it reliably by manually modifying the first occurrence so that it no longer had the first StartDate-StartTime. (See the attached patch.)

Ideally, we'd join against the occurrence with the first start date/time. But this is complicated by the fact that StartDate and StartTime are separate columns. (Think about an hourly event, for example.) I don't know off the top of my head how to do this in MySQL, but I feel like it must be possible. We might need a temporary table?

#9 - 2016-03-31 04:06 PM - Daniel Jones

What about replacing event_occurence entirely by converting the date and time columns into a timestamp and picking the earliest one? I'm not a SQL expert so it's possible that this'll run really slowly with large tables. Do you have a sense of that?

```
LEFT JOIN
( SELECT events1.* FROM {$wpdb->eo_events} events1
JOIN
( SELECT post_id, MIN(TIMESTAMP(CONCAT(StartDate, ' ', StartTime))) as earliest_datetime
FROM wp_eo_events
GROUP BY post_id
```

```
) AS events2
ON events1.post_id = events2.post_id AND TIMESTAMP(CONCAT(events1.StartDate, ' ', events1.StartTime)) = events
2.earliest_datetime
ORDER BY events1.StartDate ASC, events1.StartTime ASC
)
AS {$wpdb->eo_events}
ON $wpdb->posts.id = {$wpdb->eo_events}.post_id
```

#10 - 2016-03-31 09:56 PM - Boone Gorges

At a glance, it looks like this is more accurate. I don't know if it will be slow, but better slow than wrong :)

Another possibility is to lobby for an upstream improvement: a new column in the EO table that is a full timestamp for the start of the occurrence. This would be annoying in the sense that you'd need to keep the columns in sync, but it would make it far easier to sort. (As a side note, I don't know why he has two columns to begin with. This is only useful if you're trying to see, for example, which of next month's events will be the earliest in the day.) What do you think?

Were you able to test your idea against the unit test described in my patch?

#11 - 2016-04-17 01:48 PM - Daniel Jones

Yeah I agree it would be better to just have the column generated by EO. How would we go about asking for that?

To be totally honest, I'm totally sure how to do unit testing - also the version of the plugin I downloaded didn't have any tests included with it - should I get them from the Github repository instead of the WP plugins repository for that?

#12 - 2016-04-21 12:39 AM - Boone Gorges

> Yeah I agree it would be better to just have the column generated by EO. How would we go about asking for that?

I think the first step is probably to open a ticket on the github repo https://github.com/stephenharris/Event-Organiser/ where we explain the issue and ask Stephen if he thinks it's something worth pursuing. If he says no, there's no point in writing any code. If he says yes, you and I can work together on writing the code that would be necessary to make this work in Event Organiser (I have a feeling it's going to be complicated, and involve a migration routine), and then send a PR.

> To be totally honest, I'm totally sure how to do unit testing - also the version of the plugin I downloaded didn't have any tests included with it - should I get them from the Github repository instead of the WP plugins repository for that?

Yeah, get it from Github - use his develop branch, if in doubt. You'll need PHPUnit, which you can get as a phar: https://phpunit.de/manual/current/en/installation.html Then run the bin/install-wp-tests.sh script included in the Event-Organiser repo. Once that's done, you should be able to run 'phpunit' in the event-organiser directory and see the magic happen. Once you get to that point, and once Stephen gives a green light on the feature, we can work through the process of actually writing the tests.

#13 - 2016-05-16 07:49 PM - Daniel Jones

Sorry for the delay here. Learning curves...

So I got to the point of running phpunit on the dev repository. It gave me 1 failure, 1 error, 1 skipped, and 2 incomplete. Is that about par for the course on a plugin of this complexity, or should all the tests be passing?

This is without making our changes to the export, or apply your patch.

When I do try to apply your patch it doesn't look like any changes happen. When I open it in a text editor, it looks like there aren't any changes, just some lines that get deleted and replaced with the same line. What do you think I'm missing?

#14 - 2016-05-23 04:25 PM - Boone Gorges

> So I got to the point of running phpunit on the dev repository. It gave me 1 failure, 1 error, 1 skipped, and 2 incomplete. Is that about par for the course on a plugin of this complexity, or should all the tests be passing?

I'm getting one intermittent failure, which appears to be due to an HTTP timeout:

```
.................I.............................................  63 / 215 ( 29%)
................................................I......I.....  126 / 215 ( 58%)
...............F.......................S.......................  189 / 215 ( 87%)
.........................                                        215 / 215 (100%)

Time: 7.73 seconds, Memory: 34.00Mb
```

```
There was 1 failure:

1) iCalTest::testWebCalProtocol
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
-'http_request_failed'
+'unable-to-fetch'

/home/bgorges/sites/cmaster/plugins/event-organiser/tests/unit-tests/iCalTest.php:828
```

Is that what you're seeing? If so, I think it's safe to ignore.

> When I do try to apply your patch it doesn't look like any changes happen. When I open it in a text editor, it looks like there aren't any changes, just some lines that get deleted and replaced with the same line. What do you think I'm missing?

Oy, it looks like I generated the patch incorrectly. I don't have the old test anymore. I'll look into rebuilding the test at some point soon.

Files

eo-recurrences.patch	59.7 KB	2016-03-07	Boone Gorges