AutoSync

Introduction

MiPlanIt provides auto sync for the calendar events. It is applicable for both Google and Outlook calendars. When an event is added users social calendar it automatically synchronizes to Miplanit calendar. All details like reminder organizer..etc in the event is synced. This feature helps users automatically generate the events data without any refresh or else.

Change Logs

- AutoSync not working for the Outlook calendar.
  - Fixed issue in special characters in deltaLink format - `updateNextLinkInCalendar() function.`
  - `Also fixed the date format of calendar StartDate in Calendar.py`

Business Details

Sl.no.	Rule	Type	Implementation
1	Get details of external Calendar	Google → calendartype -1 Outlook → calendartype - 2	Query: **SELECT** **"calendarId", "userId", "extCalendarId",** **"token", "refreshToken", "lastSyncDate", "deltaLink",** **"accessTokenExpirationDate", "socialAccountId"** **FROM calendar** **WHERE "status" = true AND "calendarType" IN ({})**

<table>
<tr><td rowspan="2">2</td><td rowspan="2">Refresh Access Token</td><td>Google</td><td>URL:

<u>https://oauth2.googleapis.com/token</u>

Parameter:

{
"client_id": MIPLANIT_GOOGLE_CLIENT_ID,
"client_secret": "",
"refresh_token": sRefreshToken,
"grant_type": "refresh_token"
}</td></tr>
<tr><td>OutLook</td><td>URL:

<u>https://oauth2.googleapis.com/token</u>

Parameter:

{
"client_id" : MIPLANIT_OUTLOOK_CLIENT_ID,
"grant_type" : "refresh_token",
"scope" : MIPLANIT_OUTLOOK_SCOPE,
"refresh_token" : sRefreshToken,
"redirect_uri" : "",
"client_secret" : MIPLANIT_OUTLOOK_CLIENT_SECRET,
}</td></tr>
</table>

<table>
<tr>
<td>3</td>
<td>Update Token in Calendar DB</td>
<td>Google → calendarType - 1

OutLook → calendarType - 2</td>
<td>Query:


```
UPDATE calendar SET
"token" = '{}',
"accessTokenExpirationDate" = NULLIF('{}', '')::timestamp
WHERE "socialAccountId" = '{}' AND "calendarType" = {} AND "userId" = {};
```

</td>
</tr>
<tr>
<td>4</td>
<td>Calendar Request</td>
<td>Google</td>
<td>URL:

https://www.googleapis.com/calendar/v3/calendars/

Parameter:


```
updatedMin    =  datetime.datetime.now() +      (weeks=-4)
dUpdatedMin   =  datetime.datetime.strftime(updatedMin, '%Y-%m-%dT%H:%M:%S.%fZ')

{
 "timeZone"      : "UTC",
 "access_token"  : sToken,
 "showDeleted"   : True,
"updatedMin"     : dUpdatedMin,
 "maxResults"    : 2000
}
```

</td>
</tr>
</table>

		OutLook	URL():
			
			https://graph.microsoft.com/v1.0/me/calendars/{}/events
			
			Headers:
			
			Host = 'graph.microsoft.com'
			
			{
			'Host' : Host,
			'Authorization' : "Bearer {}".format(token)
			}
			
			Delta URL(Delete Removed OutLook Events):
			
			https://graph.microsoft.com/v1.0/me/calendars/{}/calendarView/delta
			
			Parameter:
			
			{
			"startDateTime" : datetime.datetime.strftime(startDateTime, '%Y-%m-%dT%H:%M:%S.%fZ'),
			"endDateTime" : datetime.datetime.strftime(endDateTime, '%Y-%m-%dT%H:%M:%S.%fZ')
			}
			
			URL(Fetching recurring events):
			
			https://graph.microsoft.com/v1.0/me/calendars/{}/events
			
			Parameter:
			
			{
			"$filter" : "type eq 'seriesMaster' or end/dateTime ge '{}'".format(timeMin),
			"$top" : 500

<table>
<tr><td>5</td><td>Sync back</td><td>Google and OutLook</td><td>Query:

INSERT INTO temp_log ("userId", "data") VALUES (275, '{"aData":""}');</td></tr>
<tr><td rowspan="2">6</td><td rowspan="2">Delete Event</td><td>Google</td><td>URL:

https://www.googleapis.com/calendar/v3/calendars/{}/events/{}

Parameter:

{
'sendNotifications' : True,
'access_token' : token
}</td></tr>
<tr><td>OutLook</td><td>URL:

https://graph.microsoft.com/v1.0/me/calendars/{}/events/{}

Parameter:

Host = 'graph.microsoft.com'

{
'Host' : Host,
'Authorization' : "Bearer {}".format(token)
}</td></tr>
</table>

7	Upsert Event	Google	Fetch User Basic Details - Query:
			
			SELECT ces."sharedStatus" as "sharedStatus",um.email as "email"
			FROM cal_event_share ces
			LEFT JOIN user_main um ON ces."sharedUser" = um."userId"
			WHERE ces."eventId" = {0} AND ces.status = true AND ces."sharedStatus" <> 2 AND "sharedUser" NO
			(SELECT "createdBy" FROM cal_event ce2 WHERE "eventId" = {0})
			UNION
			SELECT nrus."sharedStatus" AS "sharedStatus", nrus.email AS "email"
			FROM non_reg_user_share nrus
			WHERE nrus."activityId" = {0} AND "activityType" = 2
			
			URL:
			
			https://www.googleapis.com/calendar/v3/calendars/{}/events
			
			Parameter:
			
			{
			'sendNotifications' : True,
			'access_token' : token
			}
			
			Fetch Calendar Events - Query:
			
			SELECT
			ce."eventId",
			ce.description,
			ce."location",
			ec."calendarId",
			ce."extEventId",
			to_char(ce."startDate", 'YYYY-MM-DD') AS "startDate",
			to_char(ce."endDate", 'YYYY-MM-DD') AS "endDate",
			to_char(ce."startTime", 'HH24:MI:SS') AS "startTime",
			to_char(ce."endTime", 'HH24:MI:SS') AS "endTime",
			ce."eventName",

		OutLook	Fetch User Basic Details - Query:
			
			ELECT ces."sharedStatus" as "sharedStatus",um.email as "email"
			FROM cal_event_share ces
			LEFT JOIN user_main um ON ces."sharedUser" = um."userId"
			WHERE ces."eventId" = {0} AND ces.status = true AND ces."sharedStatus" <> 2 AND "sharedUser" NO
			(SELECT "createdBy" FROM cal_event ce2 WHERE "eventId" = {0})
			UNION
			SELECT nrus."sharedStatus" AS "sharedStatus", nrus.email AS "email"
			FROM non_reg_user_share nrus
			WHERE nrus."activityId" = {0} AND "activityType" = 2
			
			URL:
			
			https://graph.microsoft.com/v1.0/me/calendars/{}/events
			
			Headers:
			
			Host = 'graph.microsoft.com'
			
			{
			'Host' : Host,
			'Authorization' : "Bearer {}".format(token)
			}

Main Source Code References

Sl.no.	File name	Path	Type of Script	Remarks
1	Autosync	backend-master\development\lambda	Python with PosetgreSQl	
2	Calendar	backend-master\development\lambda	Python with PosetgreSQl	
3	CommonFunctions	backend-master\development\lambda	Python with PosetgreSQl	
4	syncBack	backend-master\development\lambda	Python with PosetgreSQl	
5	OtherNotifications	backend-master\development\lambda	Python with PosetgreSQl	
6	CalendarEvents	backend-master\development\lambda	Python with PosetgreSQl	
7	Configurations	backend-master\development\lambda	Python with PosetgreSQl	

Event Details

AutoSyncing Calendar

Method: POST

Request:

```
event = {
"field": "AutoSync",
"arguments": {
"userId": 282,
"silentNotification" : 1,
"calendarType" : 1
}
```

```
}
```

Response:

```
{'Status': 'OK', 'Statuscode': 'SU001', 'Message': 'Data sync successfully'}
```

Details

If arguments passed in events, it means that synchronization is needed for a specific user. Therefor the calendar will be synchronized specifically for the user according to the calendar type

If arguments doesn't passed the userId will be None and the data according to the calendar type will be saved as an array of calendars for each external calendars.

- Then the synchronization process of social calendars and events starts with,
- For each object in an array, each and every details about a calendar like user id, refresh token, external calendar id, last sync date, token, access token expiration date, social account id, etc
- if refresh token exist, then the access token expiration date is compared with the current date, if it is over new access token is requested by providing the refresh token. The new access token will be returned with expiration data and all. It will be saved calendar with respect to social account id, calendar type and user id.
- With Token API Parameter will be passed and requested. If the response status_code is 200 which is OK, then response is collected in JSON format. Items will be updated.
- For each items a single event will be fetched with data like event name, event id, location, recurrence etc.
- If the original start time referred in events the the single event will be updated with the original start time with respect to date or date time.
- If conference data is referred in events the single event will be updated with conference type and URI, and also full conference data which is the entire thing about conference data.

- if organizer is referred in events and it is "self" then the single event will be updated organizer name and email with no value, else organizer email id and name will be fetched from email and display name in organizer. if organizer is not provided single event will be updated organizer name and email with no value.
- If the status of event is cancelled, then the updated date will be no valued.
- If the html or hangout link is provided in event, it will be saved as it is.
- Then the original start and end time zone saved.
- Then the attendees data like email id and status is saved to single events attendees data, if attendees in events
- If attachments is provided in events it will be also saved.
- If reminder data is provided, then the whole data like reminder type and minutes will be updated
- Each single events will be updated to events in the Google's calendar details where token, refresh token and all are saved.
- These calendar details will be passed to calendar's import calendar data function with respect to userid. From the response calendar id is took and direct it to sync back function.
- If the event status is 1 or False then the event will be deleted else, event will be upserted.

For outlook

- Get response in JSON, then startdate and enddate is mentioned.
- For each value in the response, if id is provided and type is seriesMaster, then the request is send with exception date as parameter. If the response got has status_code as 200 i.e. OK, then check for error. If error is not in the response, then
- If cancelled occurrences or exception occurrences are provided, it is saved in a variable as it is or an empty list is saved.
- If isCancelled is true in event, then the master data will have remove child instance and remove cancelled instances will be set True. Else the start date and all day constraint will be set.
- After all cancellation request is checked, then single event is taken from the whole outlook calendar and all the data like event name, event id, etc. all other details will be saved
- If the original start time referred in events the the single event will be updated with the original start time.
- If excluded dates are provided in the event, the single event will be updated with excluded date.
- If remove Child Instances are provided in the event and it is True, the single event will be updated with remove child as True.
- If remove Cancelled Instances are provided in the event and it is True, the single event will be updated with remove cancelled instances as True.
- If is Online Meeting is provided in the event and it is True, the single event will be updated as conference data with conference type and URI.
- If organizer is referred in events, the single event will be updated with organizer email and organizer name.
- If cancelled, status will be cancelled, else status will be confirmed

- If attachments is provided in events it will be also saved.
- If series Master id is referred in event, the single event will be updated as recurring Event id
- If reminderMinutesBeforeStart is provided, the single event will be updated as reminder with type and minutes.
- If attendees data provided in event, it will also be updated with single event.
- Each single events will be updated to events in the Google's calendar details where token, refresh token and all are saved.
- These calendar details will be passed to calendar's import calendar data function with respect to userid. From the response calendar id is took and direct it to sync back function.
- If the event status is 1 or False then the event will be deleted else, event will be upserted.