PEL - SportSG data transformation (HPBPPH-15728)

[HPBPPH-16423] [T] SportSG Job - get all programmes Created: 02/Jun/25 Updated: 25/Jul/25

Status:	Testing
Project:	HPB-PPH
Components:	None
Affects versions:	None
Fix versions:	None
Parent:	PEL - SportSG data transformation

Type:	Story	**Priority:**	High
Reporter:	Irma Rahardja	**Assignee:**	Rahul Saine
Resolution:	Unresolved	**Votes:**	0
Labels:	FY25-PMO2, PPHTP-FY25		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original estimate:**	Not Specified

Sub-tasks:

Key	Summary	Type	Status	Assignee
HPBPPH-16571	Implement recursive function	Sub-task	Open	
HPBPPH-16572	Add validation	Sub-task	Open	
HPBPPH-16573	Update job data	Sub-task	Open	

Epic Link:	PEL - SportSG data transformation
Sprint:	PPHTP FY25 Iteration 123
Story Points:	3
Rank:	0|i5s1tu:

Description

As a **System Administrator** I want **to implement a recursive job to fetch all available programmes from the Sportsg API and store them in a temporary database table** So that **we can have a complete, up-to-date dataset of programmes ready for further processing and integration into the H365 application.**

Business Information

To provide a comprehensive view of all available sports programmes, the H365 application must retrieve the entire dataset from the Sportsg API. The API uses pagination to deliver large datasets in manageable chunks. This job will implement the necessary logic to traverse all pages of the programme data, aggregate the results, and store them in a temporary location for subsequent processing.

Client Requirement

The client requires a robust, recursive process within the Sportsg Job that can handle the paginated responses from the `/programmes` API. This process must continue to call the API until all programmes have been retrieved, using the cursor provided in each response. The collected programme data should then be saved in a temporary table within our event database.

In-scope

- Design and implement a recursive or iterative loop to handle pagination for the `GET /api/v1/programme` endpoint.
- The logic must use the `after_cursor` from each API response to request the subsequent page.
- The job will utilize the `limit` parameter to control the number of items fetched per API call, with a maximum of 100.
- All programmes fetched from all pages will be aggregated in memory in a variable. PPH to log the count of retrieved programmes to represent the valid programmes retrieved from SportSG.
- After fetching all programmes, perform validation on the mandatory fields (fields to be saved in the events db, Refer https://sgtechstack.atlassian.net/wiki/spaces/HPBPPH/pages/574989777/PEL+Sportsg+API+Spec for data type and optional)
- Apply registration_start_date and registration_end_date filters as follow:
  - registration_start_date = today's date
  - registration_end_date = (today's date + 29 days)

Out of scope

- Fetching session data from the {{GET /api/v1/programme/ {programme_id}

/session}} endpoint.
- Error handling for individual API call failures within the recursive loop (this will be handled in a separate story).
- Storing the data in the Event table. Covered in https://sgtechstack.atlassian.net/browse/HPBPPH-16427
- Handling of API calls based on rate limiting.
- Batch processing of programmes to be saved in database.

Questions

1. What is the desired `limit` to be used for each API call? Should it be the maximum of 100? (default: 100)

Assumptions

- The event database and the necessary permissions to write to it are in place.
- The logic will know it has reached the end of the data when the `has_more` flag in the pagination object is `false`.
- The recursive logic will have a failsafe (e.g., maximum recursion depth) to prevent infinite loops in case of an API issue.
- The volume of programme data can be safely aggregated in the application's memory before being saved to the database.
- With the implementation of registration date filters, some events will naturally be excluded from the response, even when these events are still active and sessions are in the future. Based on event cancellation logic, these excluded events' status will be set to `Cancelled`. PAHA is aware that PPH event status will not reflect accurate status and is agreeable to this approach. However, if in the future PPH is required to reflect accurate status, registration date filters will need to be updated and performance/load testing might be required to cater to the number of programmes returned from SportSG. cc: Kenneth Ng
- API failure retry mechanism covered as part of https://sgtechstack.atlassian.net/browse/HPBPPH-16459

UI Reference (Figma / Screenshots)

Not applicable for this backend story.

Acceptance Criteria

Given	When	Then
PPHTP changes is completed	PPHTP calls GET `/api/v1/programme` API	by default, it will pass in below values in the request: • registration_start_date = today's date • registration_end_date = (today's date + 29 days) • limit = 100 • after_cursor (passed from 2nd call onwards)
The GET `/api/v1/programme` API has multiple pages of data.	The SportSG Job is executed.	The job calls the `/programmes` API repeatedly, passing the `after_cursor` from the previous response in each subsequent call until `has_more` is false.
The GET `/api/v1/programme` API returns only one page of data (`has_more` is false in the first response).	The SportSG Job is executed.	The job calls the `/programmes` API once and the count of the retrieved programme data (minus the invalid records) is logged
The job has successfully fetched all pages of programme data.	there are records with missing mandatory fields as per mentioned in Sportsg API spec document	PPH will not process these records, and will log the corresponding programme ID AND PPH will proceed to validate the next programme/event
The job has successfully fetched all pages of programme data.	there are records with invalid datatype	PPH will not process these records, and will log the corresponding programme ID AND PPH will proceed to validate the next programme/event
The job has successfully fetched all pages of programme data.	The recursive fetching process is complete.	the count of the retrieved programme data (minus the invalid records) is logged
The GET `/api/v1/programme` API returns no data.	The SportSG Job is executed.	The job completes with an error and error is logged
The GET `/api/v1/programme` API returns data for some pages	encounters errors when fetching the rest of the pages	PPH will process retrieved events data, and log this partial retrieval error

Tech notes

Implementation

- pph-svc-events:
  - Docs:
    - API: https://sgtechstack.atlassian.net/wiki/spaces/HPBPPH/pages/574989777/PEL+Sportsg+API+Spec
    - Tech Approach: https://sgtechstack.atlassian.net/wiki/spaces/HPBPPH/pages/607358477/Tech+Approach

- Base Job created in https://sgtechstack.atlassian.net/browse/HPBPPH-16146
- Add limit as job data param (default value as 100)
- Implement recursive logic to call /programmes API
  - Based on response pagination.has_more boolean value call the API will query params and updated after_cursor value.
  - If pagination.has_more is not true exit the recursive API call logic
  - Call /api/v1/programme API (Mockserver), Pass following query params
    - registration_start_date
      - date value from job data, default value as current date
    - registration_end_date based
      - calculate date from days value from job data, default value today + 29 days
    - limit
      - limit value from job data, default value as 100
    - after_cursor
      - Cursor value used for pagination. Would be passed from second call onwards to the API. Value to be extracted from previous API calls response pagination.after_cursor
  - Append the programmes retrieved from each API response in an in memory variable.
  - If unable to retrieve any programme (0 programmes retrieved), throw error and fail the job
  - If unable to retrieve all programme due to any error in retrieval, proceed to validate the programmes successfully retrieved.
- Validate the programmes (events)
  - filter out the programmes which do not pass the validation rules. Log the programme id and the error encountered during validation, continue the execution to validate next programme
    - Job will not be failed for invalid programmes.
    - If No programmes are valid, throw error and fail the job.
    - return back a boolean flag highlighting if all programmes are valid or not (would be used to determine event cancellation stage)
- Note: As the programme (events) are not being persisted in database, log the number of events to be saved ?

Showcase approach

- Test data
  -
- Execution
  -
  - ....
- Clean up
  -

Feature Toggle / Environment Variables

Checklist

✓ ✗ –

Ready for development

Item	Status / Remarks
Estimation	
User persona definition	
UI Reference: Figma / Screenshots	
In-scope & out-scope	
External/Third party systems	
Scope: Assumptions	
Scope / ACs: UI - Loading/Empty/Error states are in	
Scope / ACs: happy & error scenarios	
NFR: Performance Implications	
NFR: Security Implications	• Role based access requirements • Any change in access controls
Operations: Logs & Alerts 2024 Logging Strategy	• How do we know this is always working as intended? • How do we know this is not working?
Release: Feature toggle and environment variables	
App: Force upgrade requirement	
Showcase approach	

DB schema changes	• performance analysis

Ready for deskcheck

Item	Status / Remarks
Changes are in trunk	• peer reviewed (pairing / MR) • pipeline is green • deployed to SIT
Set environment variables and deploy to UAT	
UI: Add/Remove Translation key AND add to main sheet	
Operations: Appropriate logs added	
NFR: Execute performance tests	
Release: backward compatibility - changes can be deployed to production	
Release: add / clean up feature toggle story	• add to Feature Toggle change
Backend API: Update API security check sheet	
Backend DB: Update schema log, update confluence SOP for Schema Changes	• inform PPH to PHDH • message chat group (https://chat.google.com/room/AAAAqMJgqyQ?cls=7) • email PHDH • new Table requires permissions to be given to PHDH • Migration scripts for new schema migrations should work without manually editing the database
Documentation	• Update API collection repo • ADRs and discussions in Confluence • Update credentials repo
UI: Verify translations are reflecting correctly	

Desk check

Item	Status / Remarks
Acceptance criteria	
With QA, TA, BA, XD present	
On SIT	
Alerts	• How do we know this is always working as intended? • How do we know this is not working?

Ready for showcase

Item	Status / Remarks
SIT sanity test	
Test data setup	

Ready for UAT

Item	Status / Remarks
Set environment variables and deploy to UAT	
Test data setup	
UAT sanity test	

Ready for Prod Beta Release / Deployment behind toggle

Item	Status / Remarks

Item	Status / Remarks
In deployment request	• environment variables • services
After deployment	• third party are connectable • inbound • outbound

Ready for Prod Release

Item	Status / Remarks
In deployment request	• environment variables • services

Comments

Comment by Tong YuZuo Sergio [01/Jul/25]

For comprehensiveness, can we include acceptance criteria if the number of pages exceeds the threshold.

For clarity in AC4, the list of mandatory fields is referenced inside the API specifications itself with nullable indicators?

Comment by Raja M [03/Jul/25]

Tong YuZuo Sergio

1. Threshold of number of pages is not considered. Do we want to confirm with Sportsg before implementing this restriction?
2. Yes, Mandatory fields as mentioned in spec document. Updated ACs
3.

Comment by Tong YuZuo Sergio [03/Jul/25]

Raja M Noted. Ok to proceed.

Comment by Irma Rahardja [14/Jul/25]

added AC8 for partial retrieval scenario

Generated at Sat Jul 26 07:17:13 UTC 2025 by Bishnu Prasad using Jira 1001.0.0-SNAPSHOT#100287-rev:0139ea21e0f8b5dafbd2e1eb33923e0c468b7f69.