# Introduction

The objective of the ticketing application is to get ticket inventory data from 3rd parties and map it to specific events created in Prince or Elvis for booking purposes.

Elvis Event and New Elvis Events are separate tables in AEGP datahub. We will build an ETL script to read from these two sources, as well as the offers table and store event information in the `booking_events` table in the ticketing application.
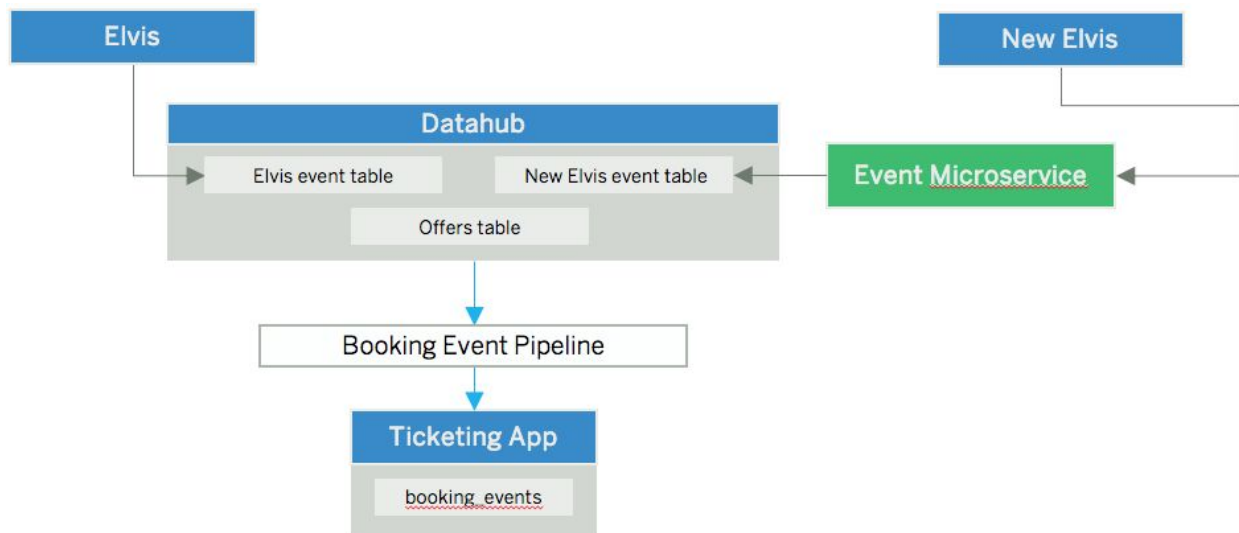
Booking event pipeline: Rake task that runs via cron
PR: https://github.com/kunai-consulting/aeg-prince-api/pull/605

Objective

The objective of this document is list product requirements around the ETL job that will be performed by Ticketing Application.
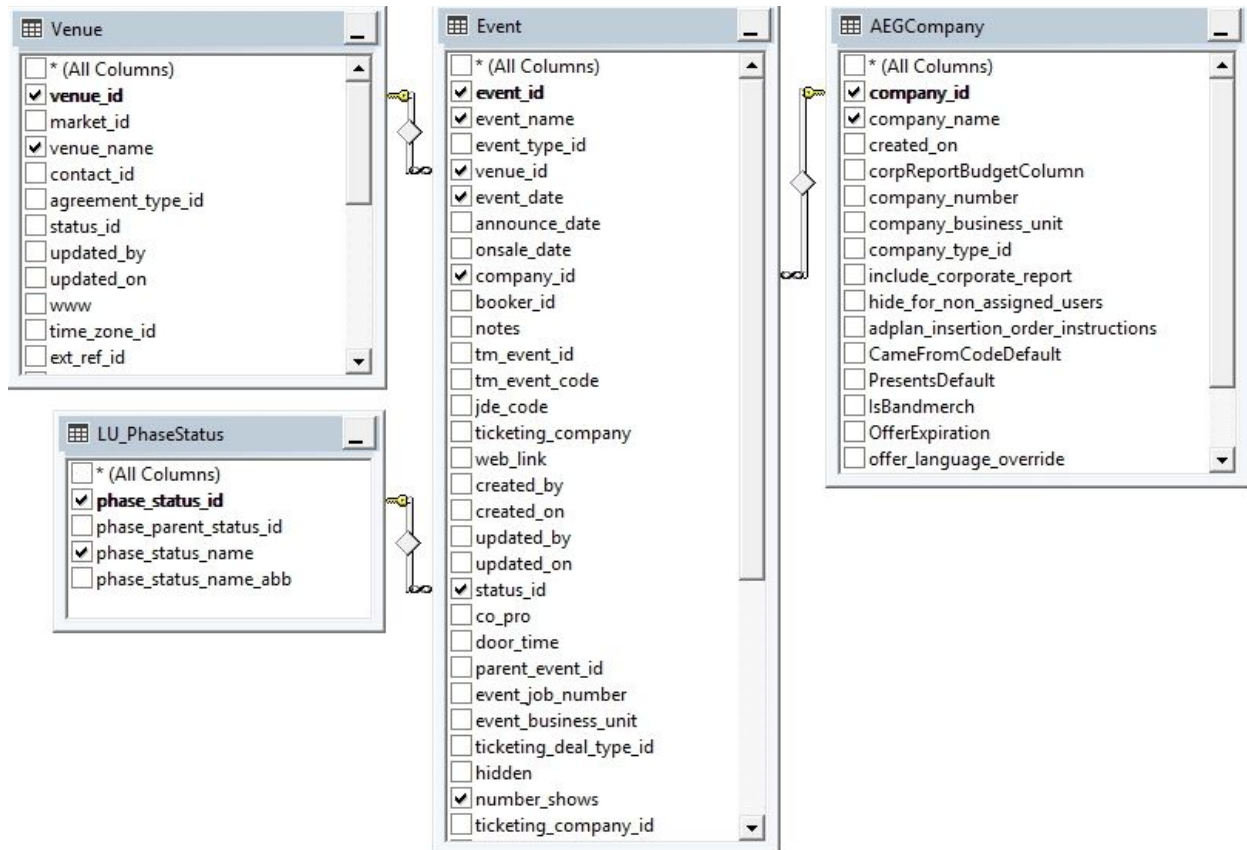
Workflow

Step 1: Read information from AEG datahub and map to booking_events and shows tables

Prerequisites

Booking event pipeline reads raw tables from datahub

Old Elvis Tables and DB view:
- Event
- Venue
- AEGCompany
- LU_PhaseStatus

New Elvis Tables:
- Events
- Shows
- Offers
- Venue

Events

- In New Elvis, the **"Events"** table will be used to create booking events in the ticketing application
- In Old Elvis, the **"Event"** table will be used to create booking events in the ticketing application

booking_events	For New Elvis	For Old Elvis
id	N/A	N/A

event_id	events.id	if (Event.parent_even_id == NULL) {
		    Event.event_id
		} else {
		    Ignore event
		}

name	events.name	Event.event_name
company_name	company.company_name	AEGCompany.company_name
company_id	events.promoter_id	Event.company_id == AEGCompany.company_id
source	new-elvis	old-elvis
venue_name	event.relevant_offer.venue.old_venue_name	Event.venue_name
venue_id	event.relevant_offer.venue.old_contact_id	Event.venue_id

show_count	get number of shows that belong to event.relevant_offer.relevant_forecast	if (Event.parent_id == NULL) {
		    Event.number_shows
		} else {
		    Ignore event
		}

status_id	return the **booking_event_status_id** attribute of the **new_evis_booking_event_status** object where event.status == new_evis_booking_event_status.id	return the **booking_event_status_id** attribute of the **old_evis_booking_event_status** object where event.status_id == old_evis_booking_event_status.id
created_at	N/A	N/A
updated_at	N/A	N/A

Status Mapping

Ticketing Application

Table to add in the ticketing application API with the following fixtures

booking_event_status		
id	status_name	
1	Cancelled	
2	Dead	
3	Draft	
4	Pending Approval	

5	Approved	
6	Sent to Agent	
7	Confirmed	
8	NOS Settlement	
9	Final Settlement	

Old Elvis

Table to add in the ticketing application API with the following fixtures. This objective of this table is to mapp old elvis offer phase status to the statuses used by the ticketing application

old_evis_booking_event_statuses		
id	phase_status_name	booking_event_status_id
1	Deal	NA
2	Show	NA
3	Draft	3
4	Pending approval	4
5	Approved	5
6	Not approved	4
7	Sent to agent	6
8	Revision required	4
9	Agent accepted	6
10	Dead	2
11	Confirmed	7
12	Re-projection requested	7
13	Postponed	7
14	Cancelled	1
15	Initial settlement	8
16	Night of show settlement	8
17	Final settlement	9
18	Accounting settlement	9
19	Settled	9

The ticketing application will need to assign a specific status when importing events from datahub that belongs to New Elvis. To assign such events a status, the Ticketing Application will need to use the mapping described below.
When performing the ETL, assume that new_evis_booking_event_statuses.id should be matched to against event.status

Table to add in the ticketing application API with the following fixtures. This objective of this table is to mapp new elvis event status to the statuses used by the ticketing application

new_evis_booking_event_statuses		
id	status_name	booking_event_status_id
1	Cancelled	1
2	Dead	2
3	Draft	3
4	Pending Approval	4
5	Approved	5
6	Sent to Agent	6
7	Confirmed	7
8	NOS Settlement	8
9	Final Settlement	9

Shows

- In New Elvis, the **"Shows"** table will be used to create shows in the ticketing application
  - For every event imported from New Elvis to the ticketing app, the ETL should import shows that are related to this Event. A show is said to be related to an event, if the following condition is meant.
    - show.forecast = event.relevant_offer.relevant_forecast
- In Old Elvis, the **"Event"** table will be used to create shows in the ticketing application.
  - What defines a show in Old Elvis is an object from the "**Event**" were the following condition is met

■ event.parent_event_id != NULL
○ The ticketing app should create a show anytime the above condition is met in Old Elvis

shows	For New Elvis	For Old Elvis
id	N/A	N/A
booking_event_id	Should match booking_event.id from the related event	Should match booking_event.id from the related event
event_date	show.event_date	should be the date part of the datetime attribute: event.event_date
event_time	show.event_time	should be the time part of the datetime attribute: event.event_date
created_at	N/A	N/A
updated_at	N/A	N/A

Step 2: How to select show information from the correct offer (prince) or event (elvis)

For New Elvis

In this example, I have one event with 3 offers:
● offer_1 has 2 shows
● offer_2 has 3 shows
● offer_3 has 3 shows

If an event is created in
Which **event_date** should we display?
- event.relevant_offer.relevant_forecast, then select show.event_date

Which **event_date** should we display?
- event.relevant_offer.relevant_forecast, then select show.event_date
- If there is more than one show that belongs to event.relevant_offer.relevant_forecast display show.event_date for all shows

Which **event_time** should we display?
- event.relevant_offer.relevant_forecast, then select show.event_time
- If there is more than one show belongs to event.relevant_offer.relevant_forecast display show.event_time for all shows

Which **show_count** should we display?
- get number of shows that belong to event.relevant_offer.relevant_forecast

Which **venue_name and venue_id** should we display?
- event.relevant_offer.venue_id
- event.relevant_offer.venue_name

For Elvis

Query for gathering information for booking_events table

```
SELECT
    dbo.Event.event_id,
    dbo.Event.event_name,
    dbo.AEGCompany.company_name,
    dbo.Event.company_id,
    dbo.Venue.venue_name,
    dbo.Venue.venue_id,
    dbo.Event.event_date,
    dbo.Event.number_shows,
    dbo.Event.status_id,
    dbo.LU_PhaseStatus.phase_status_name

FROM
    dbo.Event INNER JOIN
    dbo.Venue ON dbo.Event.venue_id = dbo.Venue.venue_id INNER JOIN
    dbo.AEGCompany ON dbo.Event.company_id = dbo.AEGCompany.company_id INNER JOIN
    dbo.LU_PhaseStatus ON dbo.Event.status_id = dbo.LU_PhaseStatus.phase_status_id

WHERE Event.event_name IN ('Mount Kimbie,Jessy Lanza', 'Katy Perry UK 2018', 'Dora the Explorer
Live!' )
```

[Output](#)

How to display show information for Elvis

Example using [`Dora the Explorer Live!`](#) event

booking_event	
id	integer
event_id	853

name	Dora the Explorer Live!
company_name	Verizon Theatre Grand Pr.
company_id	11
source	TBD
venue_name	Verizon Theatre Grand Pr.
venue_id	5294
show_count	5
status	Accounting settlement
created_at	N/A
updated_at	N/A

shows	Show #1	Show #2	Show #3	Show #4	Show #5
id	integer	integer	integer	integer	integer
booking_event_id	853	853	853	853	853
event_date	7/23/2005	7/23/2005	7/23/2005	7/24/2005	7/24/2005
event_time	11:00:00	14:00:00	17:00:00	14:00:00	17:00:00
created_at	N/A	N/A	N/A	N/A	N/A
updated_at	N/A	N/A	N/A	N/A	N/A

Appendix: