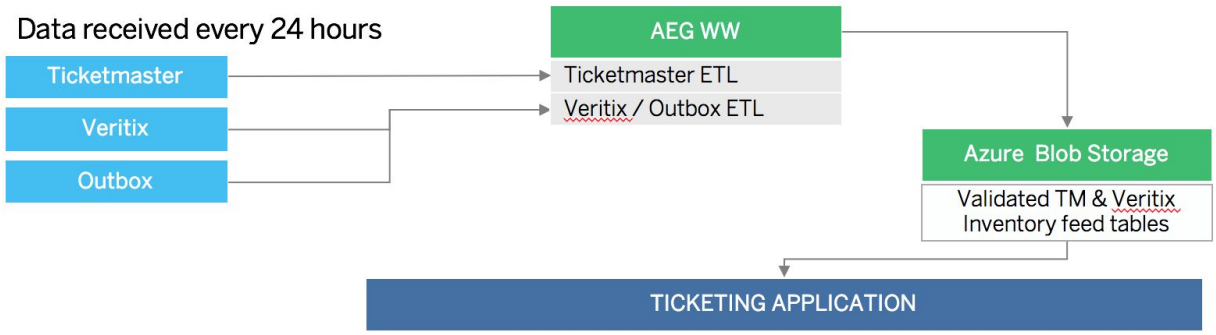
Introduction

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.

Ticketing feeds are provided by 3rd party companies (ticketmaster, axs, ..) they are pulled by AEG WW and stored in their Datahub. From this datahub they will be sent to the AEGP Blob Storage. The Ticketing Application will then will then be responsible for importing the raw inventory feeds and performing an ETL job to transform the data in tables called ticket_events and daily_counts.



Objective

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

Workflow

Step 1: Data is stored raw from AEG WW into the Ticketing Application

Prerequisites

The AXS and TM is imported raw in the ticketing application

Table 1: AXS_RAW

AXS_RAW attribute name	Description	Type
DB Name	Not used, display name for a vax. DB Names have a one to one relationship w/ Context ID and a one to many w/ facility_name.	varchar(128)
Context ID	ID for a given vax. Vax can be thought of as a unique identifier for a ticketing system. Vax/Context ID has a one to many relationship w/ facility_name	Int
Event Internal ID	Not used, I use eventcode or mastereventid in place	int
Event Code	id of the ticketing feed	int
Event Title	name of the event	varchar(128)
Major Category	Similar to TM feed, this is the AXS provided categorization of our events.	varchar(128)
Minor Category	Similar to TM feed, this is the AXS provided categorization of our events.	varchar(128)
Event Date	date of the event	date
Event Time	time of the event	varchar
Onsale Date	onsale date for the event	datetime
Sdate	date the inventory snapshot was taken	date
Price Level	ticket price level, synonymous with price level and id in TM, not standardized	varchar(64)
Restriction	rows with the substring Kill in this field are kills	varchar(64)
Price Code	used to determine price level	varchar(128)
Channel Group	This is the source by which tickets where used	varchar(64)
Seat Status	holds / opens / sold / kills	varchar(20)
Tix #	total ticket movement	int
Unit Price	face price of the ticket	decimal
Avg SCH Price	I believe this is average service charge	varchar(20)
Currency Code	currency name	varchar(10)
Facility Code	facility code	int
Facility Name	venue name	varchar(64)
City	venue city	varchar(64)
Facility State	venue state	varchar(5)
pricing_mode	This has very recently been added to our feed but can be used to determined complimentary tickets. If a seat_status='SOLD' and Pricing_mode = 'comp' the ticket is complimentary. We have only starting receiving this column as of 8-1-2018 so prior to this date comps are seat_status='SOLD' where price_code like '%comp%'	varchar(20)

Table 2: TM_RAW

TM_RAW attribute name	Description	Type
SYSTEM	VAX name is the system	varchar(20)
VAXEVID	name of the system managing the tickets for the venue	int
EVCODE	id of the ticketing event (not unique)	varchar(20)
SDATE	this is the date that the inventory snapshot was taken	date
OPTYPE	Operator type; does not impact ticket status.	varchar(5)
PRLVLID	represent difference in price change by venue and event typ just go by facevalue don't use this one	int
PRICELEVELNAME	represent difference in price change by venue and event. typ just go by facevalue don't use this one	varchar(3)
SETYPEID	enum: 0 is an open/available ticket, 1 – 7 (inclusive) indicate sold tickets, 8 is a complimentary ticket, 9 – 12 (inclusive) is also a sold ticket, 13 – 25 (inclusive) is a hold ticket, 26 is a production/kill ticket, 27 and above are considered holds as well	int
SEATTYPENAME	holds / opens / sold / kills	varchar(20)
FACEVAL	ticket's face value	decimal
FACCHARGE	Facility charge, TM charge varies by facility and optional	decimal(10,2)
SERVCHARGE	Service charge, TM charge varies and is optional	varchar(10)
PREMIUMAMOUNT	Tm Charge any additional charges on top of face value (optional and varies by venue)	decimal(10,2)
TOTALTICKETS	cummulated inventory movement	int
DAILYTICKETS	daily inventory movement	int
QUALIFIERID	used to determine price level but he gave us logic to use seattypename, used to serve purpose of price level but out dated	int
QUALIFIERNAME	used to determine price level but he gave us logic to use seattypename, used to serve purpose of price level but out dated	varchar(128)
AUCTIONFLAG	In all our records only two rows have an auction flag marked 1. I would say this is not needed	bit
ACTNAME	name of show	varchar(128)
VENUENAME	venue Name	varchar(128)
EPDATE	event Date	date
EPTIME	event Time	varchar(10)
ITVPERFORMER	main performer	varchar(128)
EXCHANGEFLAG	this signifies the tickets belong to a 3rd party ticket exchange (StubHub). Make sure you filter on ExchangeFlag = 0!	binary

INTERNETONSALEDATE	internet on sale date	date
VENUECITY	venue City	varchar(128)
SETTLEMENTCODE	Not sure what this is; I don't use it	varchar(20)
AEGFLAG	I don't use this either, couldn't figure out what it was used for	bit
VENUESTATE	venue State	varchar(3)
VENUEID	Not needed, not unique to a venuename	int
MAJCATNAME	tm's event categorization typically {MISC, FAMILY, ARTS, NONE, SPORTS, NONTKT, MOVIES, CONCERTS}	varchar(128)
MINCATNAME	TM's event sub-categorization. Think music genre's, type of sporting event, etc. Ex would be Latin Rock or Baseball, etc	varchar(128)
FEEDNAME	ticket Feed name	varchar(16)
FILENAME	The name of the file the row was included in	varchar(200)
DH_PROCESSDATE	Timestamp for file ingestion in DataHub	datetime
EN_PROCESSDATE	Timestamp for file landing in blob storage?	datetime
DH_UPDATEDATE	Not sure what this is exactly but I believe it was the time the row was last updated	datetime

Step 2: Ticketing data is stored in Ticket Application

Table 3: AXS_WRAPS

Proposed format for the AXS wrap table

ATTRIBUTE	OPTIONAL	TYPE	DEFAULT	COMMENT
axs_wrap.event_code	NO	varchar(64)		axs_raw.event_code (value to be displayed as event code on FE of application)
axs_wraps.uid	NO	varchar(64)		Concatenation of 'axs_raw.context_id' + 'axs_raw.event_internal_id'
axs_wraps.context_id	NO	int		
axs_wraps.event_datetime	NO	datetime		Time is received as a string from Datahub, and will have to be converted
axs_wraps.price_level	NO	varchar(64)		
axs_wraps.seat_status	NO	varchar(20)		
axs_wraps.price	NO	decimal		
axs_wraps.daily_movement	NO	int		total ticket movement (not a wrap)

axs_wraps.event_name	NO	varchar(128)		
axs_wraps.venue_name	NO	varchar(64)		
axs_wraps.venue_city	NO	varchar(255)		
axs_wraps.venue_state	NO	varchar(5)		
axs_wrap.restriction	NO	varchar(64)		
axs_wrap.price_code	NO	varchar(128)		
axs_wraps.currency	NO	varchar(10)		Ex: USD
axs_wraps.major_category	NO	varchar(128)		
axs_wraps.minor_category	NO	varchar(128)		
axs_wraps.sdate	NO	date		
axs_wraps.pricing_mode	NO			
axs_wraps.facility_code	NO	int		
axs_wraps.on_sale_datetime	NO	datetime		

Table 4: TM_wraps

Proposed format for the Ticketmaster wrap table

ATTRIBUTE	OPTIONAL	TYPE	DEFAULT	COMMENT
tm_wraps.event_code	NO	varchar(20)		Concatenation of ‘tm_raw.system’ + ‘tm_raw.evcode`
tm_wraps.vaxeid	NO	?		
tm_wraps.event_datetime	NO	datetime		Time is received as a string from Datahub, and will have to be converted
tm_wraps.price_level	NO	int		enum: 0 is an open/available ticket, 1 – 7 (inclusive) indicate sold tickets, 8 is a complimentary ticket, 9 – 12 (inclusive) is also a sold ticket, 13 – 25 (inclusive) is a hold ticket, 26 is a production/kill ticket, 27 and above are considered holds as well
tm_wraps.seat_status	NO	varchar(20		holds / opens / sold / kills
tm_wraps.price	NO	decimal		
tm_wraps.daily_movement	NO	int		
tm_wraps.event_name	NO	varchar(128)		
tm_wraps.venue_name	NO	varchar(128)		
tm_wraps.venue_city	NO	varchar(128)		
tm_wraps.venue_state	NO	varchar(3)		
tm_wrap.headliner	NO	varchar(128)		

tm_wrap.exchange_flag	NO	bit		this signifies the tickets belong to a 3rd party ticket exchange (StubHub). Make sure you filter on ExchangeFlag = 0!
tm_wraps.total_tickets	NO	int		
tm_wraps.feed_name	NO	varchar(16)		
tm_wraps.major_category	YES	varchar(128)		
tm_wraps.minor_category	YES	varchar(128)		
tm_wraps.sdate	NO	date		
tm_wraps.on_sale_datetime	NO	datetime		

Step 3: Create Ticket Event ticket feed

Note:

- only one ticket event can be created per event code
- when a wrap is received for a new event code, the api should be responsible for creating a new ticket_event object

ticket_event	**axs_wraps**	**tm_wraps**
id	N/A	N/A
collection_id	N/A	N/A
name	axs_wraps.event_name	tm_wraps.event_name
event_name	axs_wraps.event_name	tm_wraps.event_name
event_code	axs_wraps.event_code	tm_wraps.event_code
fcontext_id	axs_wraps.context_id	NA
vaxe_id	NA	tm_wraps.vaxevid
ticketing_company_id	48	1
venue_code	N/A	N/A
venue_name	axs_wraps.venue_name	tm_wraps.venue_name
venue_city	axs_wraps.venue_city	tm_wraps.venue_city
currency	axs_wraps.currency	Only reports on US venues so only deals with USD, may have to ask Richard for a work around here
on_sale_datetime	axs_wraps.on_sale_datetime	tm_wraps.on_sale_datetime
event_datetime	axs_wraps.event_datetime	tm_wraps.event_datetime
timezone	Times for events are giving in local time zone of the venue (try getting it from venues.time_zone + venues.time_zone_id)	Times for events are giving in local time zone of the venue (try getting it from venues.time_zone + venues.time_zone_id)

booking_event_id	N/A	N/A
created_at	autogenerated	autogenerated
updated_at	autogenerated	autogenerated

Step 4: Daily Count from a ticket feed

Note:

- A daily count object uses summarises information coming from all wraps having the same ticket event id and snap shot date
- For example, lets assume a feed was received from AEG WW datahub and 1000 tm_wraps were created.
- Let's assume that among all the tm_wraps received 50 of them share the same event code and snapshot date (sdate)
- In this example, 1 daily_count object will be created to capture information relevant to the ticketing application for this 50 tm_wraps

daily_counts	**axs_wraps**	**tm_wraps**
id	autogenerated	autogenerated
ticket_event_id	ticket_events.id (id of the ticket event created by the api for the axs_wraps.event_code)	ticket_events.id (id of the ticket event created by the api for the axs_wraps.event_code)
date	axs_wraps.sdate	tm_wraps.sdate
open	sum of axs_wraps.daily _movements where the following conditions are met: • condition 1: daily_counts.date == axs_wraps.sdate • condition 2: axs_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: axs_wraps.seat_status == "OPEN" && (axs_wraps.restrictions INCLUDES one of the following string: "OPEN", "AVAIL" or "PUBLIC")	sum of tm_wraps.daily _movements where the following conditions are met: • condition 1: daily_counts.date == tm_wraps.sdate • condition 2: tm_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: tm_wraps.price_level == 0 and tm_wraps.exchange_flag == 0
hold	sum of axs_wraps.daily _movements where the following conditions are met: • condition 1: daily_counts.date == axs_wraps.sdate • condition 2: axs_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: axs_wraps.seat_status == "OPEN" && (axs_wraps.restriction does not INCLUDE any of the following strings: "KILL", "OPEN", "PUBLIC", "AVAIL")	sum of tm_wraps.daily _movements where the following conditions are met: • condition 1: daily_counts.date == tm_wraps.sdate • condition 2: tm_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: [(tm_wraps.price_level >= 13 && tm_wraps.price_level <= 25) || (tm_wraps.price_level > 27)] && tm_wraps.exchange_flag == 0
sold	sum of axs_wraps.daily _movements where the following conditions are met:	sum of tm_wraps.daily _movements where the following conditions are met:

	• condition 1: daily_counts.date == axs_wraps.sdate • condition 2: axs_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: axs_wraps.seat_status == “SOLD” && axs_wraps.pricing_code does not contain the string “COMP”	• condition 1: daily_counts.date == tm_wraps.sdate • condition 2: tm_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: (tm_wraps.price_level >= 1 && tm_wraps.price_level <=7) or (greater than or equal to 9 and less than or equal to 12) and tm_wraps.exchange_flag == 0
gross	sum of (axs_wraps.daily _movements * axs_wraps.price) where the following conditions are met: • condition 1: daily_counts.date == axs_wraps.sdate • condition 2: axs_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: axs_wraps.seat_status == “SOLD” && axs_wraps.pricing_code does not contain the string “COMP”	sum of (tm_wraps.daily _movements * tm_wraps.price) where the following conditions are met: • condition 1: daily_counts.date == tm_wraps.sdate • condition 2: tm_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: tm_wraps.price_level is (greater than or equal to 1 and less than or equal to 7) or (greater than or equal to 9 and less than or equal to 12) and tm_wraps.exchange_flag == 0
kills	sum of axs_wraps.daily _movements where the following conditions are met: • condition 1: daily_counts.date == axs_wraps.sdate • condition 2: axs_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: axs_wraps.restriction == ‘kill’	sum of tm_wraps.daily _movements where the following conditions are met: • condition 1: daily_counts.date == tm_wraps.sdate • condition 2: tm_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: tm_wraps.price_level == 26 and tm_wraps.exchange_flag == 0
comps	sum of axs_wraps.daily _movements where the following conditions are met: • condition 1: daily_counts.date == axs_wraps.sdate • condition 2: axs_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: axs_wraps.seat_status == “SOLD” && axs_wrap.price_code contains the string “COMP”	sum of tm_wraps.daily _movements where the following conditions are met: • condition 1: daily_counts.date == tm_wraps.sdate • condition 2: tm_wraps.event_code == daily_counts.ticket_events.event_code • condition 3: tm_wraps.price_level == 8 and tm_wraps.exchange_flag == 0
capacity	Check definition with Nate	Cap of venue (guessed based on using elvis) we should ask Nate how they calculate this
wrap	daily_counts.wrap = daily_movements.sold (day n) - daily_movements.sold (day n - 1)	daily_counts.wrap = daily_movements.sold (day n) - daily_movements.sold (day n - 1)
user_added	N/A	N/A
is_adjusted	N/A	N/A
created_at	autogenerated	autogenerated
updated_at	autogenerated	autogenerated

Testing

Step 1: Select 20 ticketing events 10 from Elvis Ticketing App Ticketmaster feed and 10 from AXS feed

Prerequisites:

- They have an elvis_id

Step 2: Share ticket event_ids with Drew and ask for the raw inventory feeds for these data sets

Step 3: Process data sets in TA and compare results with Elvis ticket data

Appendix

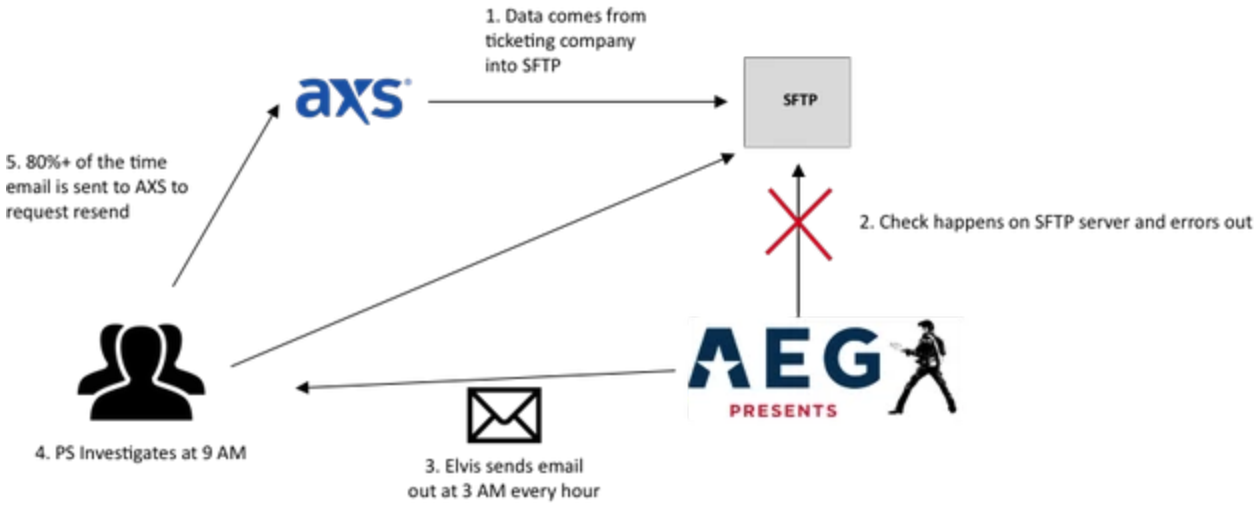
Issues with inventory feeds from TM and Veritix/AXS

ETL

1.) AXS is claiming that date is always consistent, but based on ETL error logs we can show that isn't true
dd/mm/yyyy is expected but sometimes they return mm/dd/yyyy
2.) Blank rows, not always an entire row but sometimes there are empty fields
This causes break in ETL and inventory feed isn't updated
3.) There are delimiter characters in event name (possible other fields), this can cause the ETL to split a single field into multiple
Ex: Event Name `Head;Event Name"
4.) Test events are causing issues and AXS has mentioned that they cannot resolve for these issues
5.) Scheduler is running every hour, the counter checks what is the last import date for a given event:
Problem some events aren't updated everyday, in this case a file will not be caught by the counter and it will sit in STP
6.) Decide on frequency that we will ingest files, and confirm file names we should be expecting
7.) We need to be able to re-ingest data, because there are instances were AXS will send the same inventory feed file more than once

Current ETL scraps previous records and saves the values in duplicate inventory feed
Existing pipeline, AXS is directly dropping these files to AEG ww
If ww find issues with feed they go back to AXS

Notifications



Currently there are two notification systems in place

“Missing Ticketing Company Files Warning”

- Automated sent on a schedule every day, listing any missing files (According to image from Elvis documentation)
- Runs 3am - 7am
- Very basic just list Vax’s missing (example below)
- Steve’s team has built Notification email that runs at the end of each batch process

The ticketing company files are missing. Please resend the following files:

Vax	Missing Sale Date
AEGNorthwest	5/27/2019
Arlington	5/27/2019
bancofcastdm	5/27/2019
LALivePremium	5/27/2019
LasViejasArena	5/27/2019
losangeles	5/27/2019
MountainWinery	5/27/2019
Ncalifornia	5/27/2019
RabobankArena	5/27/2019
sbbowl	5/27/2019
StaplesCenter	5/27/2019

“Error Ticketing Company Files Warning”

The ticketing company files can not be imported because there are the errors. Please fix and resend the following files.

ID: 0 FileName: AXS_Elvis_SprintCenter_20190526*.csv RowNumber: 0
CreateAt: 5/26/2019 8:05 AM Error Type: Unexpected Error
Raw Row: The wait operation timed out

ID: 0 FileName: AXS_Elvis_StaplesCenter_20190526*.csv RowNumber: 0
CreateAt: 5/26/2019 8:06 AM Error Type: Unexpected Error
Raw Row: The wait operation timed out

VAX Feed:

-

Error Handling

When inventory feed data is missing or incorrect what do Admins do?

- **Step 1:** Identify if the feed was delivered
    - Done by checking vax feed or checking Elvis STP
- **Step 2:** If the file wasn't delivered contact TM or Veritix and asks them to resend the missing files
    - Once missing files are received they are given to existing support team to reingest
- **Step 3:** File was delivered to Elvis STP, but has not been processed
    - Admins contact support team and asks to process file (note: admins are able to see if a file in FTP has been processed or not)

Inventory File Delivery Schedule

Batch 1 Files
Received between 1:30AM to 1:45AM PST

* AXS
- AXS_Elvis_AEGMidwest_*.csv
- AXS_Elvis_AEG_Southwest_*.csv
- AXS_Elvis_AEG_TheMidland_*.csv
- AXS_Elvis_asburylanes_*.csv
- AXS_Elvis_BoweryBoston_*.csv
- AXS_Elvis_BoweryNY_*.csv
- AXS_Elvis_ForestHills_*.csv
- AXS_Elvis_GulfCoast_*.csv
- AXS_Elvis_keswicktheatre_*.csv
- AXS_Elvis_masonic_*.csv
- AXS_Elvis_playstationtheatre_*.csv
- AXS_Elvis_Pompano_*.csv
- AXS_Elvis_southeast_*.csv
- AXS_Elvis_SprintCenter_*.csv
- AXS_Elvis_starlandballroom_*.csv
- AXS_Elvis_targetcenter_*.csv
- AXS_Elvis_virginia_*.csv

* Outbox

- ProdAukTicketing*.csv
- ProdLasTicketing*.csv
- ProdLonTicketing*.csv
- ProdLv1Ticketing*.csv
- ProdMgmTicketing*.csv
- ProdSweTicketing*.csv
- ProdWemTicketing*.csv

1:50 AM PST, Batch 1 processing starts
Between 1:50 and 2:00 AM PST, Batch 1 files will be delivered to Elvis destination files

Batch 2 Files
Received between 3:00AM to 3:20AM PST

* AXS
- AXS_Elvis_CityOfDenver_*.csv
- AXS_Elvis_rockymountain_*.csv

* Ticketmaster
- AEGAGF*.dat
- AEGATL*.dat
- AEGCAR*.dat
- AEGCH1*.dat
- AEGCH2*.dat
- AEGCH3*.dat
- AEGCH5*.dat
- AEGCH6*.dat
- AEGCH7*.dat
- AEGDAL*.dat
- AEGDET*.dat
- AEGDL2*.dat
- AEGFL2*.dat
- AEGFLO*.dat
- AEGINT*.dat
- AEGLON*.dat
- AEGMTN*.dat
- AEGNY1*.dat
- AEGNY2*.dat
- AEGNY4*.dat
- AEGNY5*.dat
- AEGNY6*.dat
- AEGNY7*.dat
- AEGNY8*.dat

- AEGPHI*.dat
- AEGQUE*.dat
- AEGSCS*.dat
- AEGTOR*.dat
- AEGUK3*.dat
- AEGUK4*.dat
- AEGUK5*.dat
- AEGUK6*.dat
- AEGWDC*.dat

3:25 AM PST, Batch 2 processing starts
Between 3:25 and 3:35 AM PST, Batch 2 files will be delivered to Elvis destination files

Batch 3 Files
Received between 4:00AM to 4:20AM PST

* AXS
- AXS_Elvis_AEGNorthwest_*.csv
- AXS_Elvis_Arlington_*.csv
- AXS_Elvis_bancofcastdm_*.csv
- AXS_Elvis_LALivePremium_*.csv
- AXS_Elvis_LasViejasArena_*.csv
- AXS_Elvis_losangeles_*.csv
- AXS_Elvis_mountainwinery_*.csv
- AXS_Elvis_Ncalifornia_*.csv
- AXS_Elvis_RabobankArena_*.csv
- AXS_Elvis_sbbowl_*.csv
- AXS_Elvis_StaplesCenter_*.csv
- AXS_Elvis_tmobile_*.csv
- AXS_Elvis_ValleyView_*.csv

* Ticketmaster
- AEGARZ*.dat
- AEGLA1*.dat
- AEGLA2*.dat
- AEGNCA*.dat
- AEGNEV*.dat
- AEGNV2*.dat
- AEGNV3*.dat
- AEGSEA*.dat
- AEGT44*.dat
- AEGVAN*.dat
- AEGWES*.dat

4:25 AM PST, Batch 3 processing starts
Between 4:25 and 4:35 AM PST, Batch 3 files will be delivered to Elvis destination files

NOTE: We have been collecting files above as of the 23rd of May

AXS SQL Example

```
AXS_KILL AS (
	-- KILLS ARE DETERMINED BY THE PRESENCE OF SUBSTRING KILL IN RESTRICTION FIELD
	SELECT
		EVENT_CODE
		,SDATE
		,SUM(TIX_NUMBER) AS "KILL"
	FROM aeg.AXS_INVENTORY
	WHERE
		RESTRICTION LIKE '%KILL%'
	GROUP BY EVENT_CODE, SDATE
), AXS_SOLD AS (
	-- SOLD TICKETS HAVE STATUS SOLD LESS WHERE PRICE_CODE IS COMP
	SELECT
		EVENT_CODE
		,SDATE
		,SUM(SOLD) AS SOLD
		,SUM(SOLD * TICKET_PRICE) AS TICKETREVENUE
	FROM (
		SELECT
			EVENT_CODE
			,SDATE
			,UNIT_PRICE AS TICKET_PRICE
			,SUM(TIX_NUMBER) AS SOLD
		FROM aeg.AXS_INVENTORY
		WHERE
			SEAT_STATUS = 'SOLD'
			AND PRICE_CODE NOT LIKE '%COMP%'
		GROUP BY EVENT_CODE, SDATE, UNIT_PRICE
	) S
	GROUP BY EVENT_CODE, SDATE
), AXS_COMP AS (
	-- COMPS ARE SOLDS W/ COMP STRING IN PRICE_CODE
	SELECT
		EVENT_CODE
		,SDATE
		,SUM(TIX_NUMBER) AS COMP
	FROM aeg.AXS_INVENTORY
	WHERE
		SEAT_STATUS = 'SOLD'
		AND PRICE_CODE LIKE '%COMP%'
	GROUP BY EVENT_CODE, SDATE
), AXS_OPEN AS (
	-- OPEN ARE  SEATSTATUS = OPENS - HOLDS
	-- HOLDS ARE RESTRICTIONS NOT IN (OPEN, AVAIL, PUBLIC)
```

```
	SELECT
		EVENT_CODE
		,SDATE
		,SUM(TIX_NUMBER) AS "OPEN"
	FROM aeg.AXS_INVENTORY
	WHERE
		-- AXS OPEN LOGIC: TICKETS ARE CONSIDERED SOLD IF SEATSTATUS IS SOLD BUT
COMP IS NOT IN PRICECODE FIELD
		SEAT_STATUS = 'OPEN'
		AND (
			RESTRICTION LIKE '%OPEN%'
			OR RESTRICTION LIKE '%AVAIL%'
			OR RESTRICTION LIKE '%PUBLIC%'
		)
	GROUP BY EVENT_CODE, SDATE
), AXS_HOLD AS (
	-- HOLDS ARE OPEN TICKETS W/ RESTRICTIONS NOT IN (OPEN, AVAIL, PUBLIC)
	SELECT
		EVENT_CODE
		,SDATE
		,SUM(TIX_NUMBER) AS "HOLD"
	FROM aeg.AXS_INVENTORY
	WHERE
		SEAT_STATUS = 'OPEN'
		AND RESTRICTION NOT LIKE '%KILL%'
		AND RESTRICTION NOT LIKE '%OPEN%'
		AND RESTRICTION NOT LIKE '%PUBLIC%'
		AND RESTRICTION NOT LIKE '%AVAIL%'
	GROUP BY EVENT_CODE, SDATE
```

TM SQL Example

```
TM_KILL AS (
	SELECT
		EVCODE
		,SDATE
		,SUM(TOTALTICKETS) AS "KILL"
	FROM aeg.TICKETMASTER_INVENTORY
	WHERE
		SETYPEID = 26
		AND EXCHANGEFLAG = 0
	GROUP BY EVCODE, SDATE
)

TM_SOLD AS (
	SELECT
		EVCODE
		,SDATE
		,SUM(SOLD) AS SOLD
		,SUM(SOLD * TICKET_PRICE) AS TICKETREVENUE
	FROM (
		SELECT
			EVCODE
			,SDATE
			,FACEVAL AS TICKET_PRICE
			,SUM(TOTALTICKETS) AS SOLD
		FROM aeg.TICKETMASTER_INVENTORY
		WHERE
			((SETYPEID >= 1 AND SETYPEID <= 7)
			OR (SETYPEID >= 9 AND SETYPEID <= 12))
			AND EXCHANGEFLAG = 0
		GROUP BY EVCODE, SDATE, FACEVAL
	) S
	GROUP BY EVCODE, SDATE
)

TM_COMP AS (
	SELECT
		EVCODE,
		SDATE,
		SUM(TOTALTICKETS) AS COMP
	FROM aeg.TICKETMASTER_INVENTORY
	WHERE
		SETYPEID = 8
		AND EXCHANGEFLAG = 0
	GROUP BY EVCODE, SDATE
)

TM_OPEN AS (
	SELECT
		EVCODE,
		SDATE,
		SUM(TOTALTICKETS) AS "OPEN"
	FROM aeg.TICKETMASTER_INVENTORY
	WHERE
```

```
                SETYPEID = 0
                AND EXCHANGEFLAG = 0
        GROUP BY EVCODE, SDATE
)

TM_HOLD AS (
        SELECT
                EVCODE,
                SDATE,
                SUM(TOTALTICKETS) AS HOLD
        FROM aeg.TICKETMASTER_INVENTORY
        WHERE
                (SETYPEID >= 13 AND SETYPEID <= 25) or (SETYPEID > 27)
                AND EXCHANGEFLAG = 0
        GROUP BY EVCODE, SDATE
```

// OUTDATED

AXS

AXS attribute name	Description	Type
Context ID		int
Event Code	id of the ticketing feed	varchar(64)
Event Title	name of the event	varchar(128)
Event Date	date of the event	date
Event Time	time of the event	varchar
Onsale Date	onsale date for the event	datetime
Sdate	date the inventory snapshot was taken	date
Price Level	ticket price level, synonymous with price level and id in TM, not standardized	varchar(64)
Restriction	rows with the substring Kill in this field are kills	varchar(64)
Price Code	used to determine price level	varchar(128)
Seat Status	holds / opens / sold / kills	varchar(20)
Tix #	total ticket movement	int
Unit Price	face price of the ticket	decimal
Currency Code	currency name	varchar(10)
Facility Name	venue name	varchar(64)
City	venue city	varchar(64)
Facility State	venue state	varchar(5)
Pricing Mode	helps determine which sold tickets are comps	
Facility Code	facility code	

Ticketmaster

TM attribute name	Description	Type
EVCODE	id of the ticketing event (not unique)	varchar(20)
VAXEVID		int
SDATE	this is the date that the inventory snapshot was taken	date
SETYPEID	enum: 0 is an open/available ticket, 1 – 7 (inclusive) indicate sold tickets, 8 is a complimentary ticket, 9 – 12 (inclusive) is also a sold ticket, 13 – 25 (inclusive) is a hold ticket, 26 is a production/kill ticket, 27 and above are considered holds as well	int
SEATTYPENAME	holds / opens / sold / kills	varchar(20)
FACEVAL	ticket's face value	decimal
TOTALTICKETS	cummulated inventory movement	int
DAILYTICKETS	daily inventory movement	int

ACTNAME	name of show	varchar(128)
VENUENAME	venue Name	varchar(128)
EPDATE	event Date	date
EPTIME	event Time	varchar(10)
ITVPERFORMER	main performer	varchar(128)
EXCHANGEFLAG	this signifies the tickets belong to a 3rd party ticket exchange (StubHub). Make sure you filter on ExchangeFlag = 0!	binary
INTERNETONSALEDATE	internet on sale date	date
VENUECITY	venue City	varchar(128)
VENUESTATE	venue State	varchar(3)
MAJCATNAME	tm's event categorization typically {MISC, FAMILY, ARTS, NONE, SPORTS, NONTKT, MOVIES, CONCERTS}	varchar(128)
MINCATNAME	minor Category Name? ask Drew	varchar(128)
FEEDNAME	ticket Feed name	varchar(16)