SM Sync Data Importers Guide

Version: 1.7.0

Last Updated: May 21, 2020

Table of Contents

Change Log **3**

Overview **4**

Data Ingest 4

SM Sync Architecture 4

SM Sync Monitoring and Alerting 5

Coordinator Settings & Requirements 6

Strategy 6

Input File Configuration 6

File Set Configuration 7

Set File Configuration 7

File Type Suffixes Conventions 7

Importers **9**

User Importer 9

Properties 9

Standard User Profile CSV File Format 10

Additional Format Requirements 14

Phone Numbers Field JSON Format 14

Optional JSON File Format 15

User Tag Importer 17

Properties 17

Standard User Tag CSV File Format 17

Sample CSV Input 19

JSON stream with “lookup_key” set to “email” 19

Transaction Data 20

Requirements 20

Standard Transactions JSON File Format 20

Transaction Format 20

Item Format 24

Sample JSON Input File 29

Store Catalog (Product/SKU) Importer 30

Store Catalog 30

Node 30

Child Elements 32

Sample UI 33

Example JSON Format of a Product Hierarchy 33

Venue (Store/Location) 36
Standard Venue Data CSV File Format 36
Additional Format Requirements 37
Venue Tag Importer 38
Standard Venue Tag Data CSV File Format 38
Event Importer 39
Standard Event CSV File Format 39

Appendix A - Sample Customer CSV File 40

Appendix B - Previous Data Importer Versions 41
Purchase Event Importer 41
Event Format 41
Purchase Item Format 42
Add-on Format 43
Discount Format 44
Sample JSON Input File 45
Event Categories Importer (Product/SKU Hierarchy) 46
Node 46
Child Elements 47
Sample UI 47
Example JSON Format of a Product Hierarchy 47
Order Status Importer 49
Standard Order Status CSV File Format 49
Sample JSON Input File 50

Change Log

Version	Changes	Update By
1.5.2	• Updates to User Tag Importer. Inclusion of JSON format. • Updates to User Importer. Inclusion of JSON format and deprecating User Unlinked	Joshua Foster
1.5.3	• Modifications to importer tables. Changed "Required" columns to "Required/Optional." • Changed relevant table cell values of "yes" to "Required" and "no" to "Optional." • Added paragraph to each importer section explaining that optional attributes must be consistently implemented; if included in one payload, must be included in all payloads.	Eric Feingold
1.5.4	• Modifications to Catalog format to account for use in Campaigns.	Dan Ochs
1.5.5	• Formatting changes	Eric Feingold
1.5.6	• Edit to Standard Transactions JSON file format for the store ID.	Dan Ochs
1.5.7	• Edit to reflect internal events NOT fired when loading data with SM Sync.	Eric Feingold
1.5.8	• Edits to date formatting in doc.	Eric Feingold
1.5.9	• New content in select field descriptions. Text colored red made black.	Eric Feingold
1.6.0	• POS changes for transaction data.	Eric Feingold
1.7.0	• DOB/YOB max value changed from 120 to 150. • For CSV format in user tag importer, either external_id or user_id can be used. • Optional fields added to section on event importer.	Eric Feingold

Overview

Data Ingest

SessionM Sync (SM Sync) is designed to rapidly create or update batches of data records for the SessionM Platform, including data associated with users, transactions, events, venues, and product SKUs.

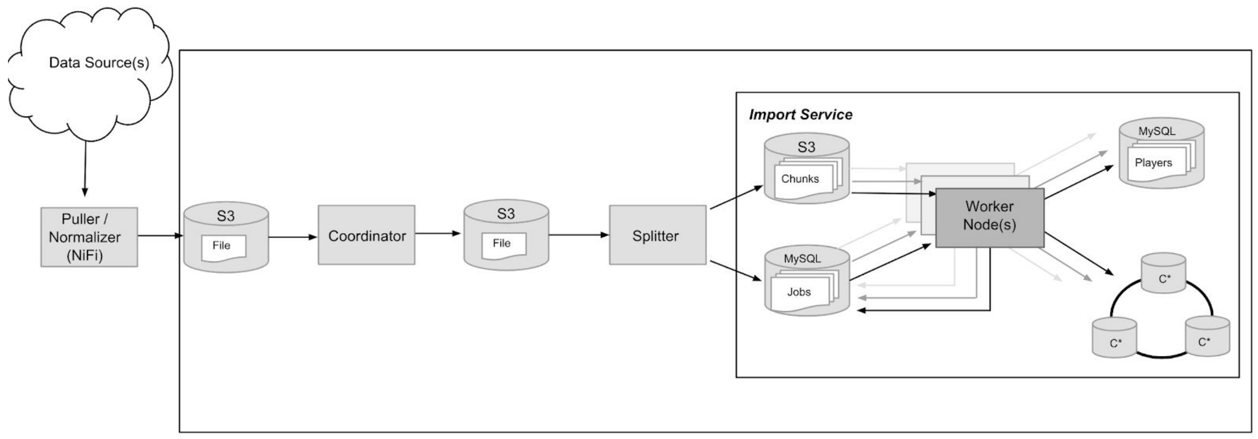
SM Sync jobs can accelerate the rate at which data is ingested by adding workers. The utility can perform both base and customer-defined model validations. One of the primary reasons SM Sync can run at an accelerated rate is its embrace of parallel processing logic. It employs a job splitter to break large jobs into smaller chunks, which are then processed in parallel by workers. Parallelism is provided by “goroutines” within worker nodes as well as by multiple worker nodes running in parallel.

Note that when SM Sync jobs load customer data, the customer *created* and customer *updated* internal events do not fire. Those events can be fired by calling the appropriate SM Sync section “Event Importer” after the customer update completes. This design ensures that data loading can occur as quickly as possible. So, if synchronization with 3rd party systems or events that need to be triggered is required, there is a method to fire them.

SM Sync Architecture

The architecture of the SM Sync utility features a series of processing steps that parse data into manageable "chunks" with corresponding job identifiers. Once split and stored in Amazon S3 (Simple Storage Solutions) buckets, this data can be acted upon by worker nodes to create SessionM user accounts, events, transactions, etc. - all of which are written to a MySQL database as well as to a Cassandra ring.

The following diagram depicts the processing flow of the SM Sync utility:



The flow begins with various data sources which are processed with a normalizer. SessionM currently uses NiFi to transform the data into the prescribed formats and drop it onto an S3 bucket as a CSV file. Using the ETL Coordinator, users can configure the import process and coordinate import jobs (SM Sync job). The Coordinator uses one file set (also known as an SM Sync template) at a time. Each set can contain one or more set files, which are just files within an SM Sync template.

The CSV files are then polled by a splitter built to look for any new files in the bucket. Each of these files must be named with the prefix "new." As it processes each file, the splitter divides the data into "chunks," aggregating every 10,000 lines into a new file. It then writes these files to another S3 bucket dedicated to storing these new chunks. Note that each new file includes the header from the original CSV file.

In addition to splitting out the CSV file into chunks, the splitter creates a job for each chunk which it stores in a MySQL data store. With this transformed user account data, the utility's core worker nodes can begin processing this data into actual SessionM user accounts. Each worker node runs its own “goroutine” that polls the jobs defined in the MySQL data store, sorting out complete jobs from new jobs to run next. Worker nodes run multiple “goroutines” as light threads that can run in parallel to one another.

Once the accounts are created, the worker nodes write them to two places for user ingest: the player’s MySQL store for linked users and the Cassandra ring for unlinked users. For linked accounts, SessionM creates full-fledged accounts for each record. For unlinked accounts, SessionM stores the information in a side table. In unlinked setups, users create very basic accounts which get populated once their accounts are linked.

SM Sync Monitoring and Alerting

SM Sync has a corresponding front end utility called the ETL Controller, which shows the current progress of the ingest process. This is monitored by internal SessionM team members and select integration partners who will also be alerted if failures to the ingest occur. This tool is not yet ready to be shared with clients.

Coordinator Settings & Requirements

Strategy

Strategy is an attribute of a file set. Strategy specifies the order in which multiple individual file sets are handled. The current list of supported strategies are specified below.

Type	Description
All	Allows all file sets to be processed as they show up in the S3 bucket.
Sequential	Allows for sequential processing of the file sets. This processing is dependent on the previous file set being processed.
Stride	Stride is similar to sequential strategy. The additional constraints on Stride derive from the its duration and offset. Duration: Time between two imports. Offset: Percentage of the stride duration when the files can be placed in the S3 bucket for the import to be processed. If all the file set files show up before the stride window starts, the set is marked "failed."

Input File Configuration

Input files can be configured to be part of a specific file set. This is achieved by naming the file in a specific format or using the Coordinator UI to pick it automatically. The input files must be named in the format below:

new_<api_key>_<timestamp>_<file_set_name>_<type_suffix>.<ext>

Consult this table for descriptions of the format attributes:

Attribute Name	Description

new	Input file should begin with the string *new_*.
api_key	The 40 character *api_key* associated with the application.
timestamp	Time stamp specifying the chronological order for processing. Time stamp should be of this format: *YYYYMMDDHHmmSS* Files in the same set job must **all** have the same timestamp value.
file_set_name	The name of the file set. This is currently setup by SessionM via our ETL Coordinator tool internally and needs to be provided to you. This value is mapped by SessionM to a specific SM Sync job type. The current job types are *event_categories*, *event*, *messaging_preference*, *phone_number_deactivation*, *purchase*, *user*, *user_create*, *user_tag*, *user_unlinked*, *user_update*, *venue*, and *venue_tag*. Note that the user job type performs like an "upsert," by which the user is inserted if they are not found or updated if they are found.
type_suffix	Type suffix that identifies the import type of the input file.
ext	The extension matching the file type expected by the importer. Currently, either JSON or CSV. Even if the file is encrypted, this file extension should be used.

File Set Configuration

If a group of input files are to be in the same file set, ensure that they all have the same *api_key*, *timestamp* and *file set name*.

Set File Configuration

Files bearing names that do not adhere to the naming convention mentioned above are ignored. The *api_key*, *timestamp* and *file set name* are configured according to the file set configuration. The import type is specified by the *type_suffix*.

File Type Suffixes Conventions

Consult the table below for a list of file type suffixes:

Name	**Importer**	**File Suffix**	**Description**

User	User	"_user.csv"	Creates new user or updates existing user based on external ID.
CreateUser	User	"_user_create.csv"	Only creates new users based on external ID. Produces errors if users exist.
UpdateUser	User	"_user_update.csv"	Only updates existing users based on external ID. Produces errors if users do not exist.
User Tag	User Tag	"_user_tag.csv"	Tags users (based on external ID) with specified tag.
Transaction	Transaction	"_connect_transaction.json"	Creates transaction-based events for users based on external IDs (in the Connect stack).
Store Catalog	Store Catalog	"_connect_upload_catalog.json"	Creates or replaces a store catalog for a given retailer based on supplied JSON tree data.
Venue	mPlace Venue	"_venue.csv"	Creates or updates venues based on external ID.
Venue Tag	mPlace Venue Tag	"_venue_tag.csv"	Creates or deletes a tag on the *mPlace* venue.
Event	Events	"_event.csv"	Creates events for users based on their external IDs.
Deprecated Importers			
Purchase	Purchase	"_purchase.json"	Creates transaction-based events for users based on their external IDs.

Event Categories (Catalog)	Catalog	"_event_categories.json"	Creates or replaces a catalog, based on supplied JSON tree data.
OrderStatus	Order Status	"_orderstatus.csv"	Updates status of offer orders for orders specified by order ID, offer ID, and external ID.
Unlinked User	User	"_user_unlinked.csv"	Creates unlinked users if the users do not exist based on external ID. Updates existing users.

Importers

User Importer

The User Importer ingests customer profile data, which can consist of both standard and appended data within the SessionM Platform. The example table below shows the start of a customer profile, with *external_ID* being the unique ID of the customer. We recommend encrypting all personally identifiable information (PII) and provide us the key to decrypt the file. The CSV file should be comma delimited, and text qualified for the fields that have the delimiter (comma) included as a value.

Please note that while any importer attribute designated “Optional” is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

Properties

Type	Key	Value	Notes for SessionM
User	*Supported encoding types:*	CSV/JSON‡	
	Suffix:	_user_tag	

	Header options:	lookup_key	Defines a record attribute to be used for user lookups. Can be either "external_id" or "email". Defaults to "external_id" when not set.
User create	*Supported encoding types:*	CSV/JSON‡	
	Suffix:	_user_create	
	Header options:	lookup_key	Defines a record attribute to be used for user lookups. Can be either "external_id" or "email". Defaults to "external_id" when not set.
	Notes:	This importer does not try to find given users before inserting them into the database.	
User update	*Supported encoding types:*	CSV/JSON‡	
	Suffix:	_user_update	
	Header options:	lookup_key	Defines a record attribute to be used for user lookups. Can be either "external_id" or "email". Defaults to "external_id" when not set.
	Notes:	This importer returns an error when the given user does not exist.	

Standard User Profile CSV File Format

The Standard User Profile format is a CSV file containing columns outlined below.

Column	Description	Type	Required/ Optional	Mapping	Notes

external_id	The customer's ID for this user.	GUID	Depends	ID field	For JSON stream format, with "lookup_key" option set to "email", this field is not required. It must be set otherwise.
external_id@<type>	A set of *external_ids* for a particular type (ex: *external_id@facebook*).	JSON array of IDs	Optional		A user may have multiple *external_ids* of any given type. For empty external_id@<type>, the field must be "[]".
email	The user's email address.	String	Depends*	email	Stored as encrypted data. Required only when JSON stream format with "lookup_key" option set to "email" is used. * email field must be provided unless "enable_user_auto_create" is true in the rewards_system settings.
first_name	The user's first name.	String	Optional	first_name	
last_name	The user's last name.	String	Optional	last_name	
age	The age of the user in years.	Number	Optional	Appended Data	**It is strongly suggested that this be provided via either the 'yob' or 'dob' fields.** Directly specified age fields become incorrect over time. Should be between 13 and 150 years ago, if specified.

dob	The date of the user's birth in YYYY-MM-DD format.	Date	Optional	dob	Should be between 13 and 150 years ago, if specified. Stored as encrypted data.
yob	The year of the user's birth in YYYY format.	Numb er	Optional	yob	Should be between 13 and 150, if specified.
state	The state or province in which the user resides. Should be an ISO-3166-1 Alpha-2 code.	String	Optional	state	
city	The name of the city in which the user resides.	String	Optional	city	
zip	The ZIP or postal code associated with the user's residence. Should follow the standards for the specific country. For US, this is 5-digit or zip+4. For Canada, 6 chars. For Japan, 7-digit.	String	Optional	zip	
country	The country of the user in ISO-3166 format.	String	Optional	country	
address	The user's street address.	String	Optional	address	
address2	The user's street address (cont'd).	String	Optional	address2	

gender	The gender of the user. Either 'm' for male, 'f' for female, or '' for unspecified.	String	Optional	gender	
hhi	The user's household income. Currently one of the following: "<$25K", "$25K-$50K", "$51K-$75K", "$76K-$100K", "$101K+", or "" for unspecified.	String	Optional	hhi	
locale	An IETF language tag, composed of a 2-letter ISO 639-1 language name and 2-letter ISO 3166-2 country subdivision (ex: en-US).	String	Optional	locale	
opted_out	Specified if the user is *opted_out* of the loyalty program.	Boolean	Optional	opted_out	If the column is not specified, it defaults to *false*.
phone_numbers	The user's phone numbers.	JSON array of Phone Number Fields	Optional	phone_numbers and player_phone_numbers	If the column is specified, it must be a valid JSON array string. This means that for empty phone numbers it must be either "null" or "[]". Look for more details at the phone numbers discussion below.

anything_else	Whatever other data the customer wishes to store.	Any allowed appended data type	Optional	anything_else (in Appended Data)	Any field not listed above is assumed to be an Appended Data field by the ingest process. These fields, like any appended data fields, must be properly defined in the *user_profile* for them to be successfully imported. Data specified in these fields must adhere to the type (and validation) requirements specified there.

Additional Format Requirements

- All fields have a maximum length of 256 characters.
- Empty optional fields are OK.
- Any additional fields that are not specified above are acceptable, but **must** be added to the Appended Data for the user according the conventions we have outlined here: https://docs.sessionm.com/server2server/#create-a-custom-profile. This **must** be done prior to ingestion as a setup step.

Phone Numbers Field JSON Format

Column	Description	Type	Required/ Optional	Notes
phone_number	A phone number for the user.	String	Required	
phone_type	The type of phone number.	String	Optional	Must be one of the following: ("home", "mobile", "office", “fax”, and “other”)
preference_flags	Array of flags related to the phone number.	JSON Array	Optional	Array can contain only one or more of the values: ("primary")

verified_ownership	Indicates if ownership of the phone number has been verified.	Boolean	Optional	Defaults to *false*.

Optional JSON File Format

If it is easier for the customer to use the JSON format, they can format their request like this:

```
[
    {
        "external_id": "1234",
        "first_name": "John",
        "last_name": "Doe",
        "gender": "m",
        "dob": "1987-01-01",
        "hhi": "$25K-$50K",
        "ethnicity": null,
        "locale": "EN_US",
        "country": "usa",
        "state": "massachusetts",
        "city": "Boston",
        "zip": "02115",
        "appended_data": {
            "user_profile": {
                "some_string_field": "string",
                "some_int_filed": 64
            },
            "different_org_model": { // these match request_key in
organization_model table.
                "some_string_field": "string",
                "some_int_filed": 64
            }
        }
    },
    {
        ...
    }
]
```

User Tag Importer

The User Tag Importer ingests expiring user tags against an existing customer profile, which can then be leveraged for building audiences or targeting for campaigns. In the SessionM Platform, tags are arbitrary strings that act as keyword classifiers used for targeting and can be broken into two types: Counter-based tags or Expiring tags. This importer supports ONLY expiring tags at this time, where a defined end date is configured against the tag that serves as its TTL (Time to Live). The tag is always updated with the last updated date to the tag. So to force a tag to expire earlier, you can set *end_date* to a date in the past.

While any importer attribute designated “Optional” is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

NOTE: The APIs support both counter-based and expiring tags, if a counter-based method is required.

Properties

Type	Key	Value	Notes for SessionM
User Tag	*Supported encoding types:*	CSV/JSON‡	
	Suffix:	_user_tag	
	Header options:	lookup_key	Defines a record attribute to be used for user lookups. Can be either "external_id", “user_id”, or "email". Defaults to "external_id" when not set.
	Notes:		

Standard User Tag CSV File Format

The Standard User Tag format is a CSV file containing columns outlined below.

Column	Description	Type	Required/ Optional	Mappin g	Notes
external_ id	The unique external customer's ID for this user.	GUID	Depends	external_ id	For JSON stream format, with "lookup_key" option set to "email", this field is not required. Otherwise, It must be set to either “external_id” or “user_id”.
user_id	The unique internal customer’s ID for this user.	GUID	Depends	user_id	For JSON stream format, with "lookup_key" option set to "email", this field is not required. Otherwise, It must be set to either “external_id” or “user_id”.
email	The user’s email address	String	Depends	email	Required only when JSON stream format with "lookup_key" option set to "email" is used.
end_date	The time in RFC3339 format for when the tag should expire.	String	Optional	end_dat e	The format for the string is YYYY-MM-DDTHH:M MSSZHH:MM. If end_date is not specified, it is calculated as 20 years from now.
tag	The tag name.	String	Required	tag	

Sample CSV Input

1	external_id,end_date,tag
2	ext_id_1,2020-01-01T00:00:00-00:00,tag_111
3	ext_id_3,2020-01-01T00:00:00-00:00,tag_333
4	ext_id_2,2020-01-01T00:00:00-00:00,tag_222

JSON stream with “lookup_key” set to “email”

```
{
        “lookup_key”: “email”
}
[
        {
                “tag”: “tag_111”,
                “end_date”: “2020-01-01T00:00:00-00:00”,
                “email”: “email1@sessionm.com”
        },
        {
                “tag”: “tag_222”,
                “end_date”: “2020-01-01T00:00:00-00:00”,
                “email”: “email2@sessionm.com”
        }
]
```

Transaction Data

Importing transaction data allows SessionM to perform many different functions within the SessionM Platform, including but not limited to the following: tracking purchase attributions for points based economies, tier calculation, product and offer recommendations, and for calculating RFM (recency, frequency, and monetary spend) scores while also presenting an ongoing purchase history for the customer within their individual profile.

While any importer attribute designated “Optional” is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

NOTE: If you need access to the older version of this documentation, it has been moved to Appendix B.

Requirements

- Transaction files will be processed as they arrive in the SM Sync imports directory.
- Transaction files must confirm to the Standard Transaction File Format documented below.
- SM Sync will post the listed transactions to the Transactions Domain utilizing the Connect Service “Send Transaction” API routes (/api/2.0/send_transaction & /api/2.0/send_batch_transaction)

Standard Transactions JSON File Format

The Transaction file format is a JSON file containing an array of Transactions to process. Transactions are JSON hashes containing attributes which are outlined below.

Transaction Format

Attribute	Description	Type	Required/Optional	Notes
store_id	The SessionM store ID where the transaction took place.	String	Required	
user_id	The SessionM user ID of the person making the transaction.	String	Optional	

request_id	The request ID.	String	Optional	If unspecified, SM Sync generates one using a hash of the filename and the transaction offset within the file.
pos_employee_id	A client system employee_id. Maximum of 20 characters.	String	Required	ID of the employee who should be associated to this transaction.
sm_employee_id	The SessionM user_id for the employee, which is only required for clients that have an employee reward program.	String	Optional	If transaction ID is not available from the customer, one can be generated by combining StoreID, RegisterID and Receipt Number.
table_id	The client system table_id for where the user was seated. Maximum of 20 characters.	String	Optional	Not needed for retail or e-commerce.
guest_count	The number of guests serviced by this transaction.	Int	Optional	

is_closed	Whether or not this is the final state for the transaction.	Boolean	Required	Should only be closed once the check is finalized and closed, also when any locked discounts will be applied.
is_voided	Whether or not this transaction was voided.	Boolean	Required	if true, any and all loyalty earnings gained from association to this transaction will be removed.
transaction_id	The unique identifier for the transaction. Maximum of 64 characters.	String	Required	Needed for any and all future updates for this transaction.
from_transaction_id	The parent transaction ID. Maximum of 64 characters.	String	Optional	For split transactions, this references the parent transaction that this transaction was split from
subtotal	The subtotal without tax included of the transaction. This subtotal should reflect all of the applied discounts.	Decimal	Required	
tax_total	The total amount of tax for the transaction.	Decimal	Required	

open_time	A UTC time in RFC3339 format for when this transaction was opened or created.	String (DateTime)	Required	This value should remain consistent for all correspondence regarding this transaction.
modified_time	A UTC time in RFC3339 format for when this transaction was last modified and triggered the update call to SessionM.	String (DateTime)	Required	
guest_receipt_code	The globally unique across all client transactions receipt code printed on the guest check.	String	Optional	If no guest check is printed, this can be null.
channel	The channel that this transaction originated from. This could be something like “STORE” or “ECOM” or “KIOSK”. This is only used for client aggregation reporting, these values can be whatever the customer would like, just must be consistent across all instances of the same channel. Example values could be: “IN-STORE”, “CARRY OUT”, and “DRIVE THROUGH”.	String	Required	
items	The current collection of item objects that make up this transaction.	Array of Items	Required	
payments	The current collection of payment objects for this transaction. Note, if there are no payments applied, this should be an empty array.	Array of Payments	Required	

discounts	The current collection of discount objects for this transaction. Note, if there are no discounts applied this should be an empty array.	Array of Discounts	Required	
culture	The culture at the point-of-sale.	String	Optional	

Item Format

Attribute	Description	Type	Required/ Optional	Notes
line_id	The unique - within this transaction- ID for this transaction line item. This is used for reference for modifiers and line item discounts. These values should remain consistent across subsequent requests.	String	Required	
item_id	The client system unique item_id. Maximum of 45 characters.	String	Required	
quantity	The quantity purchased for this item.	Decimal	Required	
unit_price	The unit price for 1.0 quantity for this item.	Decimal	Required	
subtotal	The subtotal for this line. This should reflect unit_price * quantity = subtotal.	Decimal	Required	
tax_included	The amount of tax represented in the subtotal. This should only be used when the unit_price includes tax and should not be used for check level tax such as sales tax.	Decimal	Required	
modifies_line_id	If this item is a modifier for another item, this is the line_id of the item it modifies. Modifier prices should not be reflected	String	Optional	

	within the unit_price or subtotal of the item being modified.			

Payment Format

Attribute	Description	Type	Required/ Optional	Notes
payment_id	The unique, to this transaction, payment ID assigned by the client system. Maximum of 64 characters.	String	Required	
amount	The total amount of this payment applied to the transaction.	Decimal	Required	
type	The type of this payment. Example values: “Cash”, “Credit Card”, “Gift Card”, “Certificate”. Maximum of 20 characters.	String	Required	
payment_time	The date and time when this payment was applied. This should be JavaScript JSON date time format. (yyyy-MM-ddTHH:mm:ss.fffZ) and be passed as the UTC date and time.	String (DateTime)	Required	
user_id	The user ID that is applied to this payment.	String	Required	
user_id_type	Required when user_id is passed. This is used to determine the lookup type for a loyalty account. The lookup type provided must be configured as a required field	String	Required	

	and guaranteed unique within the program configuration. Legal values: “SessionM_ID” and “External_ID”.			
additional_user_id	An additional user ID that is applied to this payment. Depending on program structure, this could be used for reporting purposes or where multiple user accounts within a hierarchy should both be rewarded.	String	Optional	
additional_user_id _type	Required when additional_user_id is passed. This is used to determine the lookup type for a loyalty account. The lookup type provided must be configured as a required field and guaranteed unique within the program configuration. Legal values: “SessionM_ID” and “External_ID”.	String	Optional	
receipt_code	The globally unique, across all client transactions, receipt code printed on the payment check. If a user was looked up and a user_id was provided, this can be null.	String	Optional	

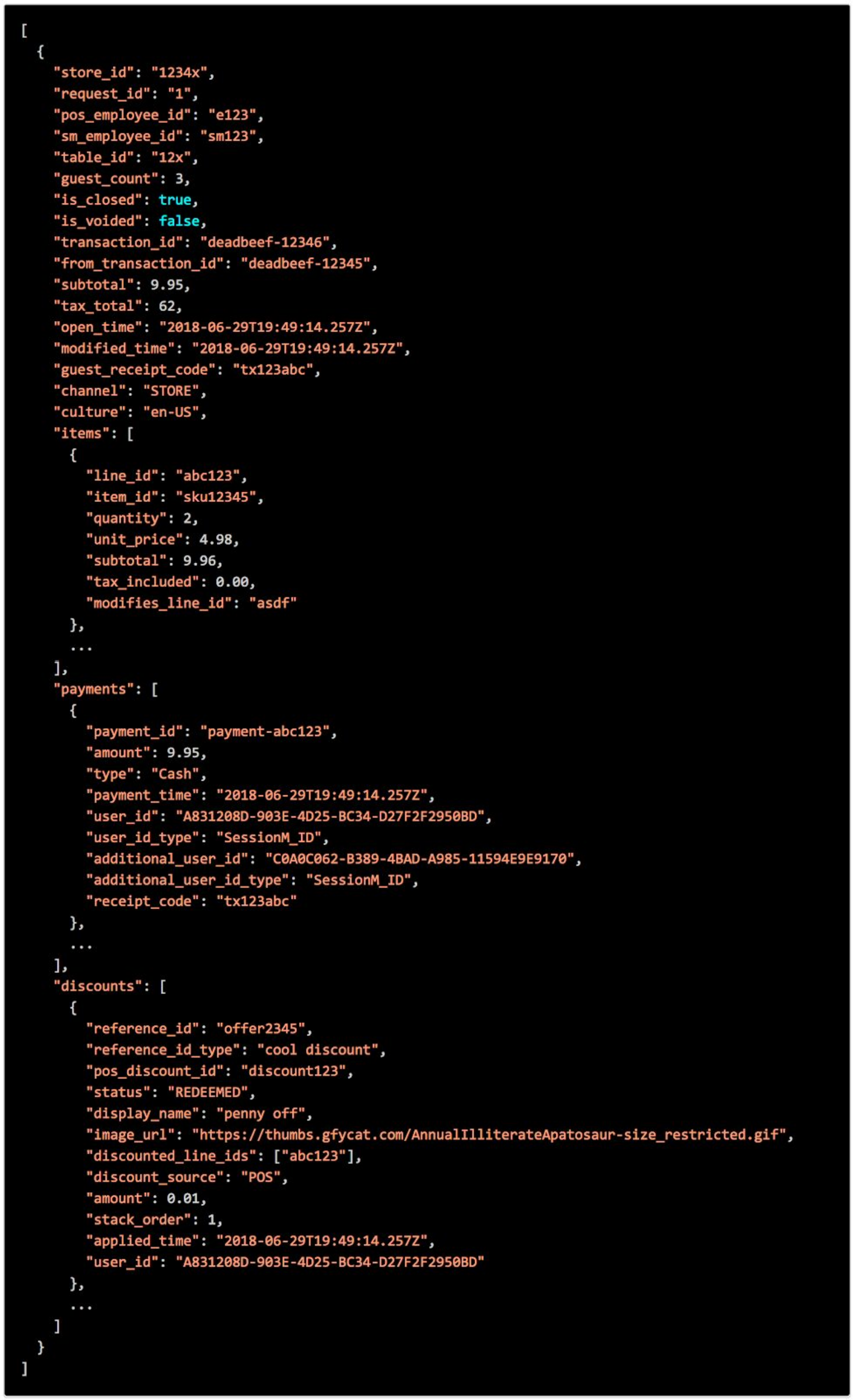
Discount Format

Attribute	Description	Type	Required/ Optional	Notes

reference_id	The reference id of the discount. If this is a SessionM provided discount, this should be the provided user offer ID. If this a discount provided from another system, it should be the identifier that uniquely identifies this discount within the transaction.	String	Required	
reference_id_type	Will be used to be echoed back in Discounting APIs.	String	Required	
pos_discount_id	If passed, can be used to apply the discount from the point of sale side linking the discount details below to the discount on the point of sale for reporting purposes.	String	Optional	
status	When this is provided in the response from any of the Discounting APIs it should be echoed back in subsequent transaction updates. Valid values are "LOCKED" or "REDEEMED" or "INFO" or "OFFLINE". Note: The OFFLINE status should ONLY be used for offline transactions. This skips any validation/restriction checks and forces the offer to be removed from the user's wallet. In the event the offer is no longer available due to expiration or being used by another online transaction, this fails without notification back to the point of sale. OFFLINE should only ever be used to ensure an offer is no longer available in a user's wallet.	String	Optional	
display_name	The name of the discount.	String	Required	

image_url	The image URL associated with this discount.	String	Optional	
discounted_line_ids	If this discount is applied to one or more items within the transaction, this should contain the line_id of what items were discounted. If this is a check level discount not applied to any specific items, this can be null.	Array of Strings	Optional	
discount_source	The source system for the discount. Example: If this discount was provided by SessionM, the source should be “SessionM”. If this is a discount, such as an employee discount, the source could be “POS”.	String	Required	
amount	The amount of the discount applied to the check.	Decimal	Required	
stack_order	The order in which this discount was applied. If it is the only discount on the check, this value should be 0. (For multiple discounts, use order example: 0 => 1 => 2 => 3 => etc.)	Int	Required	
applied_time	The date and time that this discount was applied to the transaction. This should be JavaScript JSON date time format (yyyy-MM-ddTHH:mm:ss.fffZ) and be passed as the UTC date and time.	String (DateTime)	Required	
user_id	Returned from the Lock and Redeem calls, this is the SessionM user’s identification.	String	Optional	

Sample JSON Input File



```
[
  {
    "store_id": "1234x",
    "request_id": "1",
    "pos_employee_id": "e123",
    "sm_employee_id": "sm123",
    "table_id": "12x",
    "guest_count": 3,
    "is_closed": true,
    "is_voided": false,
    "transaction_id": "deadbeef-12346",
    "from_transaction_id": "deadbeef-12345",
    "subtotal": 9.95,
    "tax_total": 62,
    "open_time": "2018-06-29T19:49:14.257Z",
    "modified_time": "2018-06-29T19:49:14.257Z",
    "guest_receipt_code": "tx123abc",
    "channel": "STORE",
    "culture": "en-US",
    "items": [
      {
        "line_id": "abc123",
        "item_id": "sku12345",
        "quantity": 2,
        "unit_price": 4.98,
        "subtotal": 9.96,
        "tax_included": 0.00,
        "modifies_line_id": "asdf"
      },
      ...
    ],
    "payments": [
      {
        "payment_id": "payment-abc123",
        "amount": 9.95,
        "type": "Cash",
        "payment_time": "2018-06-29T19:49:14.257Z",
        "user_id": "A831208D-903E-4D25-BC34-D27F2F2950BD",
        "user_id_type": "SessionM_ID",
        "additional_user_id": "C0A0C062-B389-4BAD-A985-11594E9E9170",
        "additional_user_id_type": "SessionM_ID",
        "receipt_code": "tx123abc"
      },
      ...
    ],
    "discounts": [
      {
        "reference_id": "offer2345",
        "reference_id_type": "cool discount",
        "pos_discount_id": "discount123",
        "status": "REDEEMED",
        "display_name": "penny off",
        "image_url": "https://thumbs.gfycat.com/AnnualIlliterateApatosaur-size_restricted.gif",
        "discounted_line_ids": ["abc123"],
        "discount_source": "POS",
        "amount": 0.01,
        "stack_order": 1,
        "applied_time": "2018-06-29T19:49:14.257Z",
        "user_id": "A831208D-903E-4D25-BC34-D27F2F2950BD"
      },
      ...
    ]
  }
]
```

Store Catalog (Product/SKU) Importer

The Store Catalog importer supports multiple functions within the SessionM Platform, including but not limited to: setup of targeted promotional campaigns based on purchase events, managing offer restrictions and eligibility, product and offer recommendation data, and storing the transactions against the customer profile. It's a best practice to provide a hierarchical set of product data, with multiple levels of categorization. Doing so provides a simpler user experience for marketers looking to select products or groups of products. This file format is also not a *.csv* file, but a *.json* file.

While any importer attribute designated "Optional" is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

NOTE: If you need access to the older version of this documentation, it has been moved to Appendix B.

Store Catalog

Key	Type	Description	Required/ Optional	Notes
store_id	String	Store ID.	Required	Valid store_id that the catalog belongs to.
nodes	Array	Node array.	Required	An array of child nodes.

Node

Key	Type	Description	Required/ Optional	Notes
name	String	Category name. While category names can differ when under different parent categories, the category name must be unique for each category that resides within the	Required	

		same level of the catalog hierarchy.		
category_id	String	Category ID. If existing catalog does not use “Category IDs,” IDs should be generated for each category prior to import. Category IDs should remain consistent upon subsequent uploads. Generated IDs should be provided by the client to ensure consistency when catalogs are updated.	Required	
description	String	Description for category.	Optional	
children	Array	Child array.	Depends	An array of child nodes, or nil only if subcategories are populated. Contains at least one child node, if no subcategories.
subcategories	Array	Node array.	Depends	An array of child nodes, or nil only if children are populated. Contains at least one subcategory, if no children.

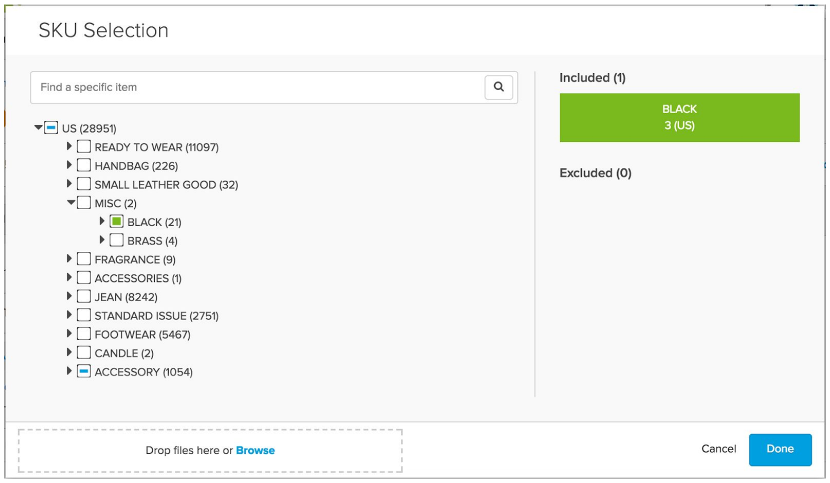
A Node represents a node of a tree. A Node is a node with a non-null array of children and/or non-null array of subcategories. Unlike the Node in the Catalog Event Categories Importer, a Node object here can have **both** child nodes and subcategories.

Child Elements

Key	Type	Description	Required/ Optional
id	String	Item ID. Item IDs (that represent the same item) can duplicate within the same catalog, but must not duplicate within the same parent category.	Required
sku	String	Stock Keeping Unit - A universal value across the client/brand to identify a product. This value should be the same for an item across all stores/channels within a client's catalogs. While required, in special use cases the SKU requirement can be overridden upon ingest.	Required (default)
name	String	Name of the item.	Required
description	String	Description of the item.	Optional
is_modifier	Bool	Indicates if the item modifies another item within the transaction. For example, in the restaurant industry, food toppings are typically modifiers. If the item is a modifier, field set to "true".	Optional
items_modified	[]String	Array of Item IDs that this item is a modifier for. Only used when the item is a modifier.	Optional

Sample UI

Child elements hold a leaf's data.


SKU Selection

Find a specific item

- US (28951)
  - READY TO WEAR (11097)
  - HANDBAG (226)
  - SMALL LEATHER GOOD (32)
  - MISC (2)
    - BLACK (21)
    - BRASS (4)
  - FRAGRANCE (9)
  - ACCESSORIES (1)
  - JEAN (8242)
  - STANDARD ISSUE (2751)
  - FOOTWEAR (5467)
  - CANDLE (2)
  - ACCESSORY (1054)

Included (1)

BLACK
3 (US)

Excluded (0)

Drop files here or Browse

Cancel Done

Example of Product Hierarchy in SessionM for Retail Clothing

Example JSON Format of a Product Hierarchy

```
[
  {
    "store_id": "BD3BABD0-0CB6-41D9-AAE0-B7B7F58EE5E5",
    "nodes":
    [
      {
        "name":"Wine",
        "description":"Delicious fermented grape product",
        "category_id":"8ecce9a1f30df32d3708f030b37a9002",
        "subcategories":
        [
          {
            "name":"White Wine",
            "description":"Wine from white grapes",
            "category_id":"9df49509fec363f6827525ed7304e6fd",
            "children":
            [
              {
                "id":"8",
                "name":"Standing Stone Chardonnay Ice Wine"
              },
              {
                "id":"10",
                "name":"Standing Stone Riesling Ice Wine"
              },
              {
                "id":"13",
                "name":"Standing Stone Riesling Ice Wine"
              }
            ]
          },
          {
            "name":"Fortified Wine",
            "description":"Wine with extra punch",
            "category_id":"382a98704e73dc8782eeebac14d7f1da",
            "children":
            [
              {
                "id":"9",
                "name":"Standing Stone Gewurzdraminer Ice Wine"
              },
              {
                "id":"11",
                "name":"Standing Stone Vidal Ice Wine"
              }
            ]
          },
          {
            "name":"Red Wine",
            "description":"Wine from red grapes",
            "category_id":"9df49509fec363f6827525ed7304e6f0",
            "children":
            [
              {
                "id":"12",
                "name":"Star Lane Cabernet Sauvignon"
              },
              {
                "id":"16",
                "name":"Chateau Ste. Michelle Merlot"
              },
              {
                "id":"3452",
                "name":"Chateau Ste. Michelle Merlot"
              },
              {
                "id":"18",
                "name":"Steak House Cabernet Sauvignon"
              },
              {
                "id":"19",
                "name":"Stefano Farina Barbera"
              }
            ]
          }
        ]
      },
      {
        "name":"Cheese",
        "category_id":"8ecce9a1f30df32d3708f030b37a9003",
        "subcategories":[
          {
            "name":"Gouda",
            "category_id":"9df49509fec363f6827525ed7304e6f1",
            "children":
            [
                    {
                        "id":"1",
                        "name":"Strong Gouda"
                    },
                    {
                        "id":"2",
                        "name":"Medium Gouda"
                    },
                    {
                        "id":"3",
                        "name":"Mild Gouda"
                    }
                ]
            },
            {
                "name":"Brie",
                "category_id":"382a98704e73dc8782eeebac14d7f1d2",
                "children":
                [
                    {
                        "id":"44",
                        "name":"Hard Brie"
                    },
                    {
                        "id":"55",
                        "name":"Soft Brie"
                    }
                ]
            },
            {
                "name":"Havarti",
                "category_id":"9df49509fec363f6827525ed7304e6f3",
                "children":
                [
                    {
                        "id":"666",
                        "name":"Grape Havarti"
                    },
                    {
                        "id":"777",
                        "name":"Apple Havarti"
                    },
                    {
                        "id":"888",
                        "name":"Melon Havarti"
                    }
                ]
            }
        ]
    }
  ]
 }
]
```

Venue (Store/Location)

The Venues data set supports multiple functions within the SessionM Platform, including but not limited to the following: setting up targeted promotional campaigns with location-based geofence triggering, managing offer restrictions and eligibility, segmenting audiences for campaign eligibility, and storing venue information associated with transactions against the customer profile. Although there are few fields that are required for locations, the more data provided, the simpler it will be for marketers to search and select from available locations.

Please note that while any importer attribute designated "Optional" is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

Standard Venue Data CSV File Format

The Standard Venue Data format is a CSV file containing columns outlined below.

Column	Description	Type	Required/ Optional
external_id	The customer's external unique ID for this venue.	String	Required
store_id	The customer's ID for this venue (unique per brand).	String	Optional
brand	The name of the associated *mplace_brand*.	String	Required
name	The display name of the venue.	String	Optional
lat	The latitude of the venue. Format: 7 decimal points, such as 42.1231231.	Float	Required
lng	The longitude of the venue. Format: 7 decimal points, such as -71.3213213.	Float	Required
address	The street address of the venue.	String	Required
city	The city of the venue.	String	Required
state	The state of the venue. Should be an ISO-3166-1 Alpha-2 code.	String	Required
zip	The ZIP or postal code of the venue. Should follow the standards for the specific country. For US, this is 5-digit or zip+4. For Canada, 6 chars. For Japan, 7-digit.	String	Required

country	The 3-letter country of the venue in ISO-3166 format.	String	Required
phone_number	The phone number of the venue. No format requirement.	String	Optional
time_zone	The time zone of the venue. TZ environment variable format, such as “America/New_York.”	String	Optional
dma	The DMA code of the venue. Note: the default is derived from the zip code.	String	Optional
status	The activation status of the venue. Could be one of the following: *Active*, *Inactive*, *Hidden*.	String	Optional
website_url	The website URL of the venue.	URL	Optional
geofence_radius	The radius of the geofence around the location in meters.	Float	Optional

Additional Format Requirements

- All fields have a maximum length of 256 characters.
- Empty optional fields are OK.

Venue Tag Importer

The Venues Tag data set allows for the addition of appended data relevant to a venue or location that can be used for search and targeting purposes. An example of this is to tag select quick-service restaurants to identify those venues with a drive-through.

Please note that while any importer attribute designated “Optional” is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

Standard Venue Tag Data CSV File Format

- The Standard Venue Tag Data format is a CSV file containing columns outlined below.
- Venue data files should be full dumps. The venues for each brand specified fully replace the existing venues for the given brand.
- In the case of a bad/invalid data load, reversion is handled by manually reloading the last known good data dump.
- Venue updates must preserve venue IDs when reloading data for an existing external ID.

Column	Description	Type	Required/ Optional
external_id	The customer's external unique ID for this venue.	GUID	Required
brand	The name of the associated mplace_brand.	String	Required
tag	The tag name to add.	String	Required
action	Action to take ('Add', 'Remove').	String	Optional

Example set of location based data

Event Importer

The Event importer supports many different functions within the SessionM Platform, including but not limited to the following: tracking engagement attributions for triggering messaging campaigns, points based economies, and tier calculations. In addition, the importer presents an ongoing engagement history for the customer within their individual profile.

While any importer attribute designated “Optional” is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

NOTE: This is not for tracking transactions. Please refer to the Transaction importer for those activities.

Standard Event CSV File Format

The Standard Event format is a CSV file containing columns outlined below.

Column	Description	Type	Required/ Optional	Mapping	Notes
external_id	The unique customer's ID for the user who has the event fired.	GUID	Required	external_id	
event_name	The event name.	String	Required	event_name	
occurred_at	The time in RFC3339 format for when the event occurred.	String	Required	occurred_at	The format for the string is YYYY-MM-DDTHH:MMSSZHH:MM. If occurred_at is more than 10 minutes in the future, an error is thrown.

transaction_id	Unique transaction identifier.	String	Optional	transaction_id	
context	Arbitrary piece of data related to the given transaction ID.	JSON object	Optional	context	The JSON object must be properly CSV escaped.

Appendix A - Sample Customer CSV File

* Note that the *locale*, *active_flag*, *size*, and *affinity* columns are all custom appended data fields.

```
external_id,email,first_name,last_name,gender,dob,locale,country,state,city,zip,active_flag,size,affinity
99000000,fake99000000@example.com,Julia,Cox,,1952-10-22,en_US,USA,OH,morro bay,55942,false,,1.0
99000001,fake99000001@example.com,Timothy,Jackson,m,1932-05-07,en_US,USA,TX,elk grove,false,false,,1.0
99000002,fake99000002@example.org,Phillip,Campbell,f,1979-03-28,en_US,USA,VA,morro bay,false,false,,1.0
99000003,fake99000003@example.com,Stephen,Rogers,m,1966-06-16,en_US,USA,OR,burbank,false,false,,1.0
```

Appendix B - Previous Data Importer Versions

Purchase Event Importer

While still used in select instances, the Purchase Event importer is being deprecated in favor of the new Transaction importer. Unless explicitly instructed by the SessionM Integration team, please use the Transaction importer documentation above.

Please note that while any importer attribute designated “Optional” is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

The following is an example set of data.

Event Format

Column	Description	Type	Required/ Optional	Notes
external_id	The unique customer ID of the user for whom the event has fired.	String	Required	
transaction_id	A unique transaction identifier.	String	Required	If transaction ID is not available from the customer, one can be generated by combining *StoreID*, *RegisterID* and *Receipt Number*.
occurred_at	The time in RFC3339 format for when the event occurred.	String	Optional	The format for the string is YYYY-MM-DDTHH:MMSSZ HH:MM. occurred_at must be no more than 10 minutes in the future.
transaction_type	Type of transaction.	String	Optional	Purchase/Refund etc.

purchase	Information about the item(s) involved in the purchase transaction.	Array of purchase items.	Required	
subtotal_amount	The sum of amounts.	Uint	Required	
channel	Store channel used to submit the transaction.	String	Optional	Example "instore", "online".
sub_channel	Store sub channel used to submit the transaction.	String	Optional	Example "app", "web".
store	Unique identifier of the store where the transaction occurred.	String	Optional	
pos_transaction_no	Customer facing receipt number.	String	Optional	It is not guaranteed unique and not the same as *transaction_id.*
register_no	Register ID.	String	Optional	
sales_associate_no	Sales associate ID.	String	Optional	

Purchase Item Format

Column	**Description**	**Type**	**Required/ Optional**	**Notes**
name	Unique identifier for the item in the product catalog. If one is not available, use event name.	String	Required	

description	Product description of the item.	String	Optional	
currency	Currency used for the item transaction.	String	Optional	
qty	Quantity of item.	Uint	Optional	
price_amount	Unit cost * qty in cents.	Uint	Optional	Total cost for an item type. Ex: an item costs 199 and a quantity of 3 are purchased; the value here would be 597.
amount	Actual amount the user paid for this item type.	Uint	Required	*price_amount - total discounts*. (Ex - If 3 of an item was purchased, this would be 3*unit cost - discounts.)
promotion_name	Name of the promotion on the item.	String	Optional	Used for lookups
discounts	Discount(s) applied to the purchase of the item.	Array of discount	Optional	
children	Add-on purchases associated with the purchase of an item.	Array of add-on	Optional	Example: flavor shot added to a beverage.

Add-on Format

Column	Description	Type	Required/ Optional	Notes
name	Unique identifier for the add-on item.	String	Required	
description	Product description of the add-on item.	String	Optional	

qty	Quantity of item.	Uint	Optional	
price_amount	Unit cost * qty in cents.	Uint	Required	Total cost for an item type. Ex: an item costs 199 and a quantity of 3 are purchased, the value here would be 597.
override_price_amount	Actual amount the user paid for this item.	Uint	Optional	Useful if price is overridden at the cash register
discounts	Discount(s) applied to the purchase of the item.	Array of discount.	Optional	

Discount Format

Column	Description	Type	Required/ Optional	Notes
amount	Amount of the item discount in cents.	Uint	Required	The format for the string is YYYY-MM-DDTHH:MMSS ZHH:MM. If occurred_at is more than 10 minutes in the future, an error is thrown.
description	Description of the item discount	String	Required	
promotion_name	Name of the promotion providing the discount	String	Required	
promotion_code	Code of the promotion providing the discount	String	Required	

Sample JSON Input File

```
[
 {
   "external_id":"654321",
   "occurred_at": "2017-01-02T12:34:56-04:00",
   "transaction_id" : "54621415-93EF-4B98-BF4B-617F14A9B456",
   "transaction_type": "buy",
   "channel": "Retail",
   "sub_channel": "Lunch",
   "store": "Joe's Burger Joint #123",
   "subtotal_amount": 179,
   "purchase":
   [
      {
         "name": "Cola",
         "description": "A cold, refreshing carbonated beverage",
         "currency": "USD",
         "qty": 1,
         "price_amount": 199,
         "amount": 149,
         "discounts":
         [
            {
               "amount": 50,
               "description": "50 cent coupon",
               "promotion_name": "cola discount",
               "promotion_code": "50COLA"
            }
         ],
         "children":
         [
            {
               "name": "Cherry Flavor Shot",
               "description": "Cherry Yum",
               "qty": 1,
               "price_amount": 20
            }
         ]
      }
   ]
 }
]
```

Event Categories Importer (Product/SKU Hierarchy)

While still used in select instances, the Event Categories (product/SKU hierarchy) importer is being deprecated in favor of the new Store Catalog importer. Unless explicitly instructed by the SessionM Integration team, please use the Store Catalog importer documentation above. This file format is also not a *.csv* file, but a *.json* file.

Please note that while any importer attribute designated "Optional" is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

Node

Key	Type	Description	Required/ Optional	Notes
name	String	Category name.	Required	
category_id	String	Category ID.	Required	
children	Array	Child array.	Depends	An array of child nodes, or nil. Either children or subcategories should be populated, but not both.
subcategories	Array	Node array.	Depends	An array of child nodes, or nil. Either children or subcategories should be populated, but not both.

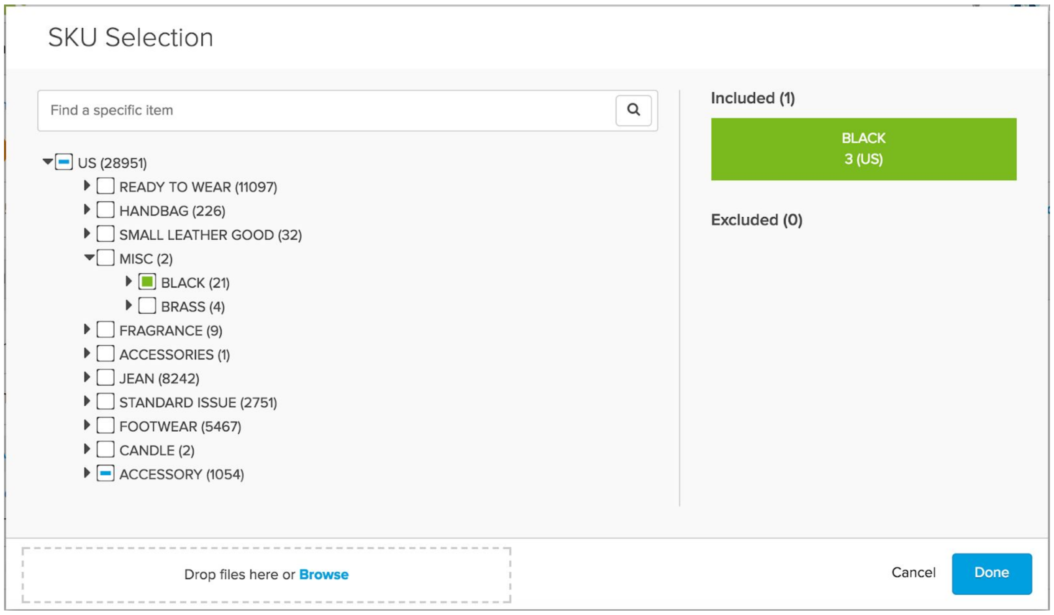
A Node represents a node of a tree (*event_categories*). A leaf node is a node with a non-null array of children. A branch node is a node with a non-nil subcategories array.
All nodes must have either their children **or** subcategories elements be non-null, but not both. Exactly one must be populated for any given node.

Child Elements

Key	Type	Description	Required/ Optional
id	String	event_data name	Required
display_name	String		Required

Sample UI

Child elements hold a leaf's data.



Example of Product Hierarchy in SessionM for Retail Clothing

Example JSON Format of a Product Hierarchy

```
[
 {
          "name": "Wine",
          "category_id": "8ecce9a1f30df32d3708f030b37a9002",
          "children": null,
          "subcategories": [
          {
          "name": "White Wine",
          "category_id": "9df49509fec363f6827525ed7304e6fd",
          "children": [
          {
          "id": "8",
          "display_name": "Standing Stone Chardonnay Ice Wine"
```

```
            },
            {
            "id": "10",
            "display_name": "Standing Stone Riesling Ice Wine"
            },
            {
            "id": "13",
            "display_name": "Standing Stone Riesling Ice Wine"
            }
            ],
            "subcategories": null
            },
            {
            "name": "Fortified Wine",
            "category_id": "382a98704e73dc8782eeebac14d7f1da",
            "children": [
            {
            "id": "9",
            "display_name": "Standing Stone Gewurzdraminer Ice Wine"
            },
            {
            "id": "11",
            "display_name": "Standing Stone Vidal Ice Wine"
            }
            ],
            "subcategories": null
            },
            {
            "name": "Red Wine",
            "category_id": "81092801b861076308da6e29b945ffcd",
            "children": [
            {
            "id": "12",
            "display_name": "Star Lane Cabernet Sauvignon"
            },
            {
            "id": "16",
            "display_name": "Chateau Ste. Michelle Merlot"
            },
            {
            "id": "3452",
            "display_name": "Chateau Ste. Michelle Merlot"
            },
            {
            "id": "18",
            "display_name": "Steak House Cabernet Sauvignon"
            },
            {
            "id": "19",
            "display_name": "Stefano Farina Barbera"
            }
            ],
            "subcategories": null
            }
            ]
  }
]
```

Order Status Importer

While still used in select instances, the Order Status importer is being deprecated as it is being replaced with recent changes to the Offers domain. Unless explicitly instructed by the SessionM Integration team, please do not use this importer.

Please note that while any importer attribute designated “Optional” is not required in a payload, its inclusion or exclusion must be applied to ALL instances of the attribute. In short, if an attribute is included in one payload, it must be included in all payloads.

Standard Order Status CSV File Format

The Standard OrderStatus format is a CSV file containing columns outlined below.

Column	**Description**	**Type**	**Required/ Optional**	**Notes**
transaction_id	The ID of the offer_order to update.	int32	Required	
offer_id	The ID of the offer in the offer_order.	int32	Required	
external_id	The unique customer's ID for the user who has the event fired.	GUID	Required	
review_state	The review_state of the offer_order.	review_state	Required	One of the following: “approved”, “rejected”, or “redemption_error”.
reviewed_at	The time in RFC3339 format for when the offer_order was reviewed.	String	Required	The format for the string is YYYY-MM-DDTHH:MMSSZHH:MM.

data	A JSON hash of metadata to store with the offer_order.	JSON hash	Optional	A hash of string → string values.

Sample JSON Input File

```
transaction_id,offer_id,external_id,review_state,reviewed_at,details
1234,2345,asdfasdf,approved,2017-10-01T12:34:56-05:00,"{""foo"":""bar""}"
```