Operational Feeds Specification

Exchange Solutions User Documentation

Table of contents:

    - Frequency
    - Feed contents
    - Return feed
- Ad hoc reward feed (internal)
    - Feed name and folder
    - Frequency
    - Feed contents
    - Rejected records
- Aggregation override feed
    - Feed name and path
    - Frequency
    - Feed contents
    - Return feed
- Bulk account closure feed
    - Feed name and path
    - Frequency
    - Feed contents
    - Logging of results
- Bulk offer import feed
    - Standard validation rules and defaults
    - Folder and file format
    - Filename
    - Frequency
    - Validations
    - Feed contents
    - Return feed
- Discretionary feed
    - Feed name, folder, and file format
    - Frequency
    - Feed contents
    - Return feed
- Exclusion feed
    - Feed name and folder
    - Frequency
    - Feed contents
    - Exclusion extended data
    - Return feed

Operational Feeds Specification

Version: 4.7.1.01 **Last updated:** 2026-03-03

These are the operational feeds (to and from the data store in AWS). To view the data warehouse feeds (directly to and from Snowflake), get the latest version from the Data Warehouse Documentation - Wizard Analytics - Confluence page.

Includes inbound (from client to Exchange Solutions) and outbound (from Exchange Solutions to client) feeds.

Table of contents

> **Note:** To ensure that all changes are properly tracked and updated in all versions, only the Technical Writer or Data team makes changes to this page.
>
> To view deprecated outbound feeds (not supported for new clients), refer to the Deprecated Feeds page.

Introduction

The ES Loyalty System provides the choice of a file-based integration for batch processing or integration via a real-time API. This document provides the ES Loyalty feed specification.

API documentation

For more documentation on the real-time API, refer to the ES Loyalty API Reference

---

ES Loyalty feed integration

Inbound feeds

The ES Loyalty System accepts the following feeds:

Feed	Description	SFTP folder
`ACTIVITY`	Accepts customer activities and events	`activity/`
`ADHOC_REDEEM` (external)	Accepts ad hoc redemption data by business units or partners	`businessUnit/<businessUnitName>/adhocRe`
`ADHOC_REDEEM` (internal)	Accepts ad hoc redemption data through a Console upload	`internal/esconsole/upload/adhocRedeem/`
`ADHOC_REWARD`	Accepts ad hoc rewards data for rewards given by	`businessUnit/<businessUnitName>/adhocRe`

Feed	Description	SFTP folder
	business units or partners	
`ADHOC_REWARD` (internal)	Accepts ad hoc reward data through a Console upload	`internal/esconsole/upload/adhocReward/`
`AGGREGATION_OVERRIDE`	Provides aggregation value updates on lift and shift	`businessUnit/<businessUnitName>/aggrega`
`BULK_ACCOUNT_CLOSURE`	Closes loyalty accounts in batch mode with options to zero account balances and anonymize member data	S3: `es-cryptography-service-<env>-esi-in` `bucket/member`
`BULK_OFFER_IMPORT`	Imports externally created offers in bulk	S3: `es-cryptography-service-uat-esi-inbo` `bucket/businessUnit/{bu}/promotion`
`DISCRETIONARY`	Accepts no-receipt adjustment transaction information for clawback calculation purposes	`discretionary/`

Feed	Description	SFTP folder
`EXCLUSION`	Establishes a global product exclusion list	`businessUnit/<businessUnitName>/exclusi`
`HISTORICAL_PURCHASE_FEED`	Accepts data for purchase history targeting, analytical segmentation or analysis, and seeding purchase data for ES Loyalty Boost™	`transaction/`
`ISSUANCE`	Accepts data regarding member issuances like the type, amount (dollar value), and expiry date	`businessUnit/<businessUnitName>/issuanc`
`MEMBER`	Accepts member information for enrollment or profile update	`member/`
`MEMBER_BALANCE_UPDATE`	Accepts member point balance information to input point balances for newly imported members or to	`member/`

Feed	Description	SFTP folder
	adjust the point balance post-launch in case of errors	
`MEMBER_CODE`	Provides data for referral programs	`member/`
`MEMBER_EXTENDED_DATA`	Accepts extended member data for extended data profile updates	`member/`
`MEMBER_HOUSEHOLD`	Allows addition/deletion of member households — groups of members that can pool rewards	`member/`
`MEMBERSHIP_TIER`	Accepts program and tier code information to identify a consumer's tier	`membershipTier/`
`PARTNER`	Accepts information relevant, for instance, to a partner account	`partner/<partner_id>`

Feed	Description	SFTP folder
	and/or payment card	
`PARTNER_LINK`	Provides partner linking information for members	`partner/<partner_id>`
`PRODUCT`	Accepts product hierarchy information for offer recognition	`businessUnit/<businessUnitName>/product`
`PROGRAM_TIER`	Accepts program and tier code information to identify a consumer's tier. Previously called the Membership Tier feed.	`programTier/`
`REDEMPTION`	Accepts information regarding redemption of rewards	`redemption/`
`STORE`	Accepts information about client store locations	`businessUnit/<businessUnitName>/store/`
`TRANSACTION`	Accepts POS purchase and	`transaction/`

Feed	Description	SFTP folder
	return information	
`VENDOR`	Accepts vendor data	`vendor/`

Snowflake Data Sharing Capability for Inbound Feeds

Inbound Feed	Snowflake Data Sharing
Product Feed	✅ Supported
Store Feed	✅ Supported
Exclusion Feed	✅ Supported
Historical Transaction / Purchase Feed	✅ Supported
Vendor	✅ Supported
Issuance Feed	❌ Not supported
Aggregation Override Feed	❌ Not supported
Bulk Account Closure Feed	❌ Not supported
Ad-hoc Reward	❌ Not supported
Ad-hoc Redeem	❌ Not supported
Activity	❌ Not supported
Discretionary	❌ Not supported

Inbound Feed	Snowflake Data Sharing
Member Balance	❌ Not supported
Member	❌ Not supported
Transaction	❌ Not supported

> **NOTE**
>
> Feeds marked as not supported can be developed upon client request.

Outbound feeds

The ES Loyalty system provides the following outbound feed:

Feed	Description	SFTP folder
`SCHEDULE_REPORT_REJECT`	Provides information about rejected inbound feed records	`businessUnit/<client name>/reject`

Nuclear updates

Exchange Solutions inbound feeds are processed using "nuclear updates." Nuclear updates mean that when a record is updated, the entire record is replaced. If this was the original record sent:

Member ID	Favourite Colour	Favourite Team	Favourite Movie
123123123	Blue	Leafs	Die Hard

Then this update is sent:

Member ID	Favourite Team
123123123	Bruins

The update results in the member's favourite colour and favourite movie being wiped to null because those attributes and values were not included in the update.

To avoid wiping existing data to null, the update must include all attributes and values, even those that repeat information from the original record:

Member ID	Favourite Colour	Favourite Team	Favourite Movie
123123123	Blue	Bruins	Die Hard

This is the only way to prevent values in the original record from being erased or set to null. All feeds must include information for all attributes that should be in the record.

General file conventions

Filenames

Filenames must follow the naming convention of `FEEDNAME_YYYYMMDD_SEQ_VERSION.EXTENSION`, where:

- `FEEDNAME` is the name of the feed
- `YYYYMMDD` is the current date
- `SEQ` is a sequence number
- `VERSION` is the feed version number (currently all files are version one)
- `EXTENSION` is either `.csv` or `.json` depending on the specific feed

Example: A `MEMBER_FEED` might be named `MEMBER_FEED_20220331_S1_V1.json`

Sequence numbers

Inbound sequence numbers must always be incremented by one for each file of the same type. Each file type maintains its sequence number independently of the other file types.

Outbound sequence numbers are also incremented by one for each subsequent file.

Formats

For JSON feeds, each JSON object must be on its own line followed by a Carriage-Return Linefeed (UNIX-style CRLF). This follows the NDJSON format.

For delimited feeds, the ESI delimiter can be configured, but a pipe (`|`) delimiter is recommended. Any delimiters present in the feed data itself must be escaped in quotes. Those quotes in turn must also be escaped.

Encryption

The overall flow of feed encryption and decryption follows the flow shown above.

Inbound feeds allow encrypted files to be uploaded through the client SFTP. The files are decrypted by the AWS Cryptoservice, then the decrypted files are sent to the inbound S3 bucket and on to the ETL Lambdas. The data is finally redirected to the DynamoDB tables.

Outbound feeds take the reverse direction, sending specified reports through the ETL Lambdas to the S3 bucket, then on to the AWS Cryptoservice for encryption. The encrypted files/reports are sent back through the outbound feed to the client.

PGP Encryption is supported and strongly recommended but is not mandatory. ES Loyalty ensures all data is encrypted in transit or at rest, but PGP provides an additional level of security.

- PGP Keys must be exchanged prior to beginning file transfer.
- MDC must be enabled when encrypting/decrypting. This is enabled by default.
- Incoming file feeds must not use Radix-64 encoding (ASCII-Armor).
- Outgoing feed files use Radix-64 encoding (ASCII-Armor).
- Prior to encryption, the feed filename must be the described name listed in this document.
- Recommended PGP libraries/tools:
  - PgpCore 5.5.0
  - OpenPGP

The encrypted file must have a `.pgp` extension before file transfer. Adding a PGP filename extension (and exchanging public keys) allows the system to identify and decrypt the appropriate file.

Example: `MEMBER_FEED_20220331_S1_V1.json.pgp`

Frequency

Details on the expected frequencies are provided on a feed-by-feed basis. In general, no feed can be accepted at a rate shorter than every 15 minutes.

Capitalization standards

Depending on the feed, field names in JSON can use either camel case or Pascal case as a convention.

Inbound feeds

Activity feed

The `ACTIVITY_FEED` provides information about member activities such as filling out a survey or entering a contest.

Feed name and folder

- **Feed name:** `ACTIVITY_FEED`
- **Format:** NDJSON
- **Example filename:** `ACTIVITY_FEED_20230331_S1_V1.json`
- **SFTP folder:** `activity/`

Frequency

This feed can be accepted once per 15-minute period. Any external member IDs referenced in this file must have arrived previously in a `MEMBER` feed, or the record will be rejected.

Feed contents

The `ACTIVITY_FEED` provides information regarding customer activities or events such as filling out a survey or entering a contest (refer to example for capitalization rules).

Field	Type	Required	Description
`loyaltyID`	string	Required	Unique per card number but not validated for format
`externalIdentifier`	string	Optional	The member that this event pertains to, using client identifier
`correlationID`	string	Required	A unique identifier for the event (your format, must be unique)
`businessUnit`	string	Optional	Business unit ID (must exist and be enabled). If not provided, the activity is associated with the loyalty program itself.
`action`	string	Required	The type of event
`namespace`	string	Required	Where the activity occurs (for example, `"CLIENT"` or `"SYSTEM"`)
`channel`	string	Required	The channel in which it took place (for example, `"STORE"`, `"ONLINE"`, `"APP"`)
`note`	string	Optional	An optional note about the event
`when`	string	Required	The time it occurred, in ISO Date Format
`extendedData`	object	Optional	Additional data about the event

Activity extended data

Extended data can be provided; however, it will not be acted upon without product customization. Any key-value pair is allowable, but the maximum data size for extended data for a given event is five kilobytes, consisting of up to 20 unique root-level keys across all members.

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "loyaltyID": "23828329",
  "correlationID": "",
  "businessUnit": "BUYCO",
  "action": "SURVEY_RESPONSE",
  "namespace": "SYSTEM",
  "channel": "ONLINE",
  "note": "Optional",
  "when": "2022-03-10T13:58:45-05:00",
  "extendedData": {}
}
```

Return feeds

The following outbound feeds are generated as a response if any records meet the criteria:

- **REJECT** — An outbound Schedule Report Reject Feed including a reason code is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.
- **REWARD** — If there are rewards associated with the completion of an activity, details of the rewards are captured and sent back to the client using an outbound REWARD feed. For more information, see Schedule Report Reward Feed in the Outbound feeds section.

Ad hoc redeem feed (external)

The `ADHOC_REDEEM` feed file is used to redeem points on behalf of members in a batch.

Feed name and folder

- **Feed name:** `ADHOC_REDEEM`
- **Format:** JSON
- **Example filename:** `ADHOC_REDEEM_20241015_S1_V2.json`
- **SFTP folder:** `businessUnit/<businessUnitName>/adhocRedeem/`

Frequency

This feed can be accepted once per 15-minute period.

Feed contents

Field	Type	Required	Description
`businessUnit`	string	Required	Business unit ID (must exist and be enabled)
`channel`	string	Required	The channel in which it took place (for example, `"STORE"`, `"ONLINE"`, `"APP"`)
`identifierType`	enum	Required	Type of identifier (for example, `"LOYALTY_ID"`, `"ACCOUNT_ID"`, `"EXTERNAL_ID"`, `"EMAIL"`, `"PARTNER_LINK_ID"`). Note: `"PHONE_NUMBER"` lookups are not allowed.
`identifierValue`	string	Required	Value of the identifier
`correlationId`	string	Required	Session identifier
`externalReferenceId`	string	Optional	Reference identifier for this transaction in the client's system
`redeemerType`	enum	Required	Type of redeemer (for example, `"PARTNER"`, `"BUSINESS_UNIT"`)
`redeemerId`	string	Required	Identifier of redeemer. If `redeemerType` is `"PARTNER"`, then `redeemerId` must be a partner associated with the `businessUnit` in the request. If `redeemerType` is `"BUSINESS_UNIT"`, then `redeemerId` must be the same as the `businessUnit` in the request.

Field	Type	Required	Description
`redemptionId`	string	Required	Identifier of redemption. Either `redeemAmount` or `catalogueItems` (with at least the `id` attribute) or both must be provided for a valid request.
`redeemAmount`	string	Optional	Total redeem amount. If both `redeemAmount` and `catalogueItems` are provided, then `redeemAmount` takes precedence.
`catalogueItems`	object array	Optional	Identifier of catalogue item. If both `catalogueItems` and `redeemAmount` are provided, then `redeemAmount` takes precedence.
`catalogueItems[].id`	string	Optional	Unique identifying name for the catalogue item
`catalogueItems[].quantity`	string	Optional	Quantity of catalogue item. If not provided, assumed to be 1. Used with `catalogueItemId` only.

Example JSON file (formatted for readability only — one object per line followed by CRLF):

```
{
  "channel": "APP",
  "businessUnit": "BUYCO",
  "identifierType": "LOYALTY_ID",
  "identifierValue": "9803101284637281945",
  "correlationId": "12df4-3948e-2938a-29384-2939f",
  "redeemerType": "PARTNER",
  "redeemerId": "MEGABANK",
  "redemptionId": "REDEMPTION_2023",
  "catalogueItems": [
    {
      "id": "TOASTER",
```

```
      "quantity": 5
    },
    {
      "id": "COFFEE_BAG",
      "quantity": 10
    }
  ],
  "redeemAmount": 10000,
  "externalReferenceId": "284948-eu78s9-293he"
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including a reason code is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Ad hoc redeem feed (internal)

The `INTERNAL_ADHOC_REDEEM` feed file is used by Console users to redeem points on behalf of a member.

Feed name and folder

- **Feed name:** `INTERNAL_ADHOC_REDEEM`
- **Format:** CSV
- **Example filename:** `adhoc_redeem.csv`
- **SFTP folder:** `internal/esconsole/upload/adhocRedeem`

Frequency

This feed can be accepted once per 15-minute period.

Feed contents

Field	Type	Required	Description
`channel`	string	Required	The channel in which it took place (for example, `"STORE"`, `"ONLINE"`, `"APP"`)
`businessUnit`	string	Required	Business unit ID (must exist and be enabled)
`identifierType`	enum	Required	Type of identifier (for example, `"LOYALTY_ID"`, `"ACCOUNT_ID"`, `"EXTERNAL_ID"`, `"EMAIL"`, `"PARTNER_LINK_ID"`). Note: `"PHONE_NUMBER"` lookups are not allowed.
`identifierValue`	string	Required	Value of the identifier
`correlationId`	string	Required	Session identifier
`externalReferenceId`	string	Optional	Reference identifier for this transaction in the client's system
`redeemerType`	enum	Required	Type of redeemer (for example, `"PARTNER"`, `"BUSINESS_UNIT"`)
`redeemerId`	string	Required	Identifier of redeemer. If `redeemerType` is `"PARTNER"`, then `redeemerId` must be a partner associated with the `businessUnit` in the request. If `redeemerType` is

Field	Type	Required	Description
			`"BUSINESS_UNIT"`, then `redeemerId` must be the same as the `businessUnit` in the request.
`redemptionId`	string	Required	Identifier of redemption
`redeemAmount`	string	Optional	Total redeem amount. Either `redeemAmount` or `catalogueItemId` must be provided, or both can be provided. If both are provided, `redeemAmount` takes precedence.
`catalogueItemId`	object array	Optional	Identifier of catalogue items. Either `catalogueItemId` or `redeemAmount` must be provided.
`catalogueItemId[].id`	string	Required (if `catalogueItemId` is used)	Unique identifying name for the catalogue item
`catalogueItemId[].quantity`	number	Optional	Quantity of catalogue item. If not provided, assumed to be 1. Used with `catalogueItemId` only.

Example CSV file:

```
channel,businessUnit,identifierType,identifierValue,correlationId,redeemerType,redeem
APP,BUYCO,LOYALTY_ID,8135997160218672743,sessionID1000,BUSINESS_UNIT,SELLCO,REDEMPTI
APP,BUYCO,LOYALTY_ID,8135099716218672743,sessionID1001,BUSINESS_UNIT,SELLCO,REDEMPTI
```

Rejected records

If records are rejected, this information is shown on the same page in the Console where the CSV file was uploaded, after the upload file is processed.

Ad hoc reward feed (external)

The `ADHOC_REWARD` feed file is used to reward members in a batch.

Feed name and folder

- **Feed name:** `ADHOC_REWARD`
- **Format:** NDJSON
- **Example filename:** `ADHOC_REWARD_20241015_S1_V2.json`
- **SFTP folder:** `businessUnit/<businessUnitName>/adhocReward/`

Frequency

This feed is a scheduled feed and runs every 15 minutes.

Feed contents

The `ADHOC_REWARD` feed is used by client partners and business units to issue ad hoc bonus points to members without running a promotion inside ES Loyalty. The issuers (client partners or business units) associate the rewards with a reward identifier for reporting purposes. The list of valid reward identifiers is maintained through the Console.

Only one of the following identifiers must be provided:

Field	Type	Required	Description
`businessUnit`	string	Required	The business unit associated with this ad hoc reward
`channel`	string	Required	The channel through which the ad hoc rewards are provided (may be `"APP"`, `"ONLINE"`, or `"SYSTEM"`)

Field	Type	Required	Description
`loyaltyId`	string	One identifier required (mutually exclusive)	Unique per card number but not validated for format
`externalIdentifier`	string	One identifier required (mutually exclusive)	The member that this event pertains to, using client identifier
`accountId`	string	One identifier required (mutually exclusive)	Unique per account but not validated for format
`emailId`	string	One identifier required (mutually exclusive)	The member's email address
`correlationId`	string	Required	UUID for this session
`issuerId`	string	Required	Unique identifier for the issuer of these reward points
`rewardID`	string	Required	Unique identifier for this specific reward
`rewardAmount`	number	Required	The quantity of points to be added or removed, formatted as a positive (no sign) or negative (`-`) number

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "channel": "APP",
  "businessUnit": "GASCORP",
  "externalIdentifier": "700034343223232",
  "correlationId": "<uuid>",
```

```
  "issuerId": "MEGABANK",
  "rewardId": "SURVEY_2023",
  "rewardAmount": 500
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

The possible reasons for a rejected record for this feed include:

Rejection message	Reason for rejection
Ad hoc Reward feature is disabled.	Feature switch disabled
Identifier not provided.	Missing identifier/accountId
Reward Amount is required and must be non-empty.	Missing `rewardAmount`
Correlation ID is required and must be non-empty.	Missing `correlationId`
Invalid business unit.	Empty `businessUnit`
Transaction is created based on folder name where file is uploaded.	Missing `businessUnit`
Issuer ID is required and should be non-empty.	Missing/empty `issuerId`
Reward ID is required and should be non-empty.	Missing/empty `rewardId`
Reward amount is more than the maximum allowed limit.	Rewarding more than the limit set in `ADHOC_REWARD` config

Rejection message	Reason for rejection
Reward id not found.	`rewardID` doesn't exist in the system

Ad hoc reward feed (internal)

The `INTERNAL_ADHOC_REWARD` feed file is used by Console users to issue points on behalf of a member.

Feed name and folder

- **Feed name:** `INTERNAL_ADHOC_REWARD`
- **Format:** CSV
- **Example filename:** `adhoc_reward.csv`
- **SFTP folder:** `internal/esconsole/upload/adhocReward/`

Frequency

This feed can be accepted once per 15-minute period.

Feed contents

Only one of the following identifiers must be provided:

Field	Type	Required	Description
`channel`	string	Required	The channel through which the ad hoc rewards are provided (may be `"APP"`, `"ONLINE"`, or `"SYSTEM"`)
`businessUnit`	string	Required	The business unit associated with this ad hoc reward

Field	Type	Required	Description
`loyaltyId`	string	One identifier required (mutually exclusive)	Unique per card number but not validated for format
`externalIdentifier`	string	One identifier required (mutually exclusive)	The member that this event pertains to, using client identifier
`accountId`	string	One identifier required (mutually exclusive)	Unique per account but not validated for format
`emailId`	string	One identifier required (mutually exclusive)	The member's email address
`correlationId`	string	Required	UUID for this session
`issuerId`	string	Required	Unique identifier for the issuer of these reward points
`rewardID`	string	Required	Unique identifier for this specific reward
`rewardAmount`	number	Required	The quantity of points to be added or removed, formatted as a positive (no sign) or negative (`-`) number

Example CSV file:

```
channel,businessUnit,loyaltyId,correlationId,issuerId,rewardID,rewardAmount
APP,GASCORP,700034343223232,8135997160218672743,MEGABANK,SURVEY_FEB,500
APP,GASCORP,700034432052047,5256202346504652402,CARDCO,SURVEY_MAR,1000
```

Rejected records

If records are rejected, this information is shown on the same page in the Console where the CSV file was uploaded, after the upload file is processed.

Aggregation override feed

The Aggregation Override feed is used to provide proper aggregation values for lift and shift (data migration) activities.

Feed name and path

- **Feed name:** `AGGREGATION_OVERRIDE`
- **Format:** JSON
- **Example filename:** `AGGREGATION_OVERRIDE_S1V1.json` (`S` refers to the serial number and `V` to the version number)
- **S3 file path:** `es-cryptography-service-<env>-esi-inbound-bucket/businessUnit/<businessUnitName>/aggregationOverride`

Frequency

The file can be uploaded as required.

Feed contents

The `AGGREGATION_OVERRIDE` feed is used to update aggregated values during a lift and shift operation.

Field	Type	Required	Description
`aggregationName`	string	Required	The type of aggregation value provided. Valid values: `"REWARDS_YEAR"`, `"REWARDS_QUARTER"`, `"REWARDS_MONTH"`, `"REWARDS_DAY"` (aggregate of rewards accumulated during the specified period); `"SPEND_YEAR"`, `"SPEND_QUARTER"`, `"SPEND_MONTH"`, `"SPEND_DAY"` (aggregate of spending for the specified period); `"REDEEM_YEAR"`, `"REDEEM_QUARTER"`, `"REDEEM_MONTH"`,

Field	Type	Required	Description
			`"REDEEM_DAY"` (aggregate of redemptions for the specified period).
`periodStartDate`	string	Required	The starting date and time of the year (UTC timestamp in ISO 8601 format)
`identifierType`	string	Required	The type of unique identifier for the member account. Valid values: `"ACCOUNT_ID"`, `"LOYALTY_ID"`, `"EMAIL"`.
`identifierValue`	string	Required	Unique identifier value for the account corresponding to the `identifierType`
`newValue`	number	Required	New aggregated value to update the account for the relevant aggregation as per the `aggregationName`

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "aggregationName": "REWARDS_YEAR",
  "periodStartDate": "2024-01-01T06:00:00.000Z",
  "identifierType": "ACCOUNT_ID",
  "identifierValue": "cf49b100-a08a-4051-9fbe-6b80b5d87a25",
  "newValue": 20000
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Bulk account closure feed

The Bulk Account Closure feed is used to close a number of accounts in a single operation. The client provides a JSON file listing loyalty IDs for accounts they want to close. Using these loyalty IDs, the account is closed, any related loyalty cards are cancelled, partner links are removed (if applicable), and the member is removed from a household (if required). Optionally (as controlled by configuration), PII data is also anonymized and points are zeroed in the account.

Feed name and path

- **Feed name:** `ACCOUNT_CLOSURE`
- **Format:** JSON
- **Example filename:** `ACCOUNT_CLOSURE_20241122_S1_V1.json`
- **S3 file path:** `es-cryptography-service-<env>-esi-inbound-bucket/member`

Frequency

The file can be uploaded as required.

Feed contents

The `loyaltyID` is always provided in this feed. Note that the default options are set in the configuration files but can be overridden for a specific member record using the options settings specified in the record as shown below.

By configuration, the options to `zeroBalances` and `anonymizeMemberData` can be set in the feed for each record. If `zeroBalances` and/or `anonymizeMemberData` are not specified in the record, they are set to the default for that record. If provided in the record, the provided value overrides the default.

All of the following examples are valid record versions provided in the file:

```
{"loyaltyID": "60524253345"}
{"loyaltyID": "60524253346", "options": {"zeroBalances": false}}
{"loyaltyID": "60524253347", "options": {"anonymizeMemberData": true}}
{"loyaltyID": "60524253348", "options": {"zeroBalances": false,
"anonymizeMemberData": true}}
```

Logging of results

Accounts that don't have an `ACTIVE` status are not processed. Accounts that fail processing are included in the `REJECT` outbound feed.

Bulk offer import feed

This feature enables clients to seamlessly import externally created offers (such as flyers) into ES Loyalty™ for behavior recognition. Instead of manually creating up to 150 offers daily in the Console, clients can export offers from their system in a supported format and integrate them efficiently.

This scalable and maintainable solution benefits both Exchange Solutions and its clients by streamlining offer ingestion, reducing manual effort, and expanding clients' opportunities to reuse their offers from an external flyer management application in ES Loyalty to allow consumer behavior recognition. The process primarily focuses on flyer offers but may also support other offer types based on scope and some customizations.

To use this feature, client system administrators upload a JSON file containing offer data which Exchange Solutions processes for integration.

The system operates in two modes: **Strict Validation** and **Flexible Validation**, controlled by a configuration setting. This configuration defines the validation rules for the ingestion process, including mandatory fields, product validity, and store existence.

- **Strict mode** — All validations must pass for every record.
- **Flexible mode** — Allows partial ingestion, skipping invalid entries while processing the rest. Flexible mode allows unknown products and stores and can be further customized by relaxing other validation rules.

Standard validation rules and defaults

Standard validation rules:

1. Offers must be future-dated. The effective date for the offer must be at minimum the current date plus one day (at least 24 hours in advance).
2. The offer's effective date must be before the expiry date.
3. The status of uploaded offers can be `active`, `disabled`, or `delete`. `active` means the offers are ready for use on the effective date; `disabled` means the offers must be enabled before use. The `delete` status is used if any offers are uploaded in error.
4. Tiers are evaluated to have all attributes in ES Loyalty's internal schema.

Defaults (most are in the code example at the end of this section):

1. The expiry date defaults to no expiry (evergreen) if no expiry date for the offer is provided.
2. If `offerActivationType` is not provided, it is set to `"MASS"` (available to all consumers).
3. If a reward control limit is not provided (`maxUses = null`), it defaults to user level 1 and the offer is unlimited.
4. If the status is not provided, it is set to `"ACTIVE"`.
5. If `publishedStatus` is not provided, it defaults to `"PUBLISHED"`.
6. The default for the `isDynamic` attribute is `false`, meaning that targeting is static and carried out when the offer is ingested into ES Loyalty.
7. By default, the channel is set to `"EXTERNAL"` unless there is a different setting in configuration (for example, `"FLYER"`).

Folder and file format

The JSON file is uploaded to the following location:

SFTP folder: `es-cryptography-service-uat-esi-inbound-bucket/businessUnit/{bu}/promotion`

> **Note:** As per the existing flow, from the client side, the file is uploaded to the client bucket. The value of `{bu}` is the business unit within the client's business, if used.

Filename

Filenames must follow this naming convention: `FEEDNAME_YYYYMMDD_SEQ_VERSION.EXTENSION`

Where `FEEDNAME` is the name of the feed (for example, `PROMOTION`), `YYYYMMDD` is the current date, `SEQ` is a sequence number, `VERSION` is the feed version number (currently all files are version one), and `EXTENSION` is `.json`.

Example filename: `BULKOFFERS_20250311_S1_v1.json`

Frequency

No feed can be accepted at a rate shorter than every 15 minutes.

Validations

Offer ID and Offer Name must conform to the following patterns:

- **Offer ID** — Must be between 3 and 128 characters, consisting solely of alphanumeric characters or underscores (`a-z`, `A-Z`, `0-9`, `_`).
- **Offer Name** — Must be between 3 and 128 characters, allowing alphanumeric characters along with specific symbols (`a-z`, `A-Z`, `0-9`, `_`, `.`, `@`, `+`, `-`, `$`, `&`, `'`), French letters, and spaces.

Feed contents

Mandatory or required fields are set in the configuration file. The attributes indicated as Required in this table are the typical required attributes. Attribute values may also be restricted by the type of offer being validated.

Field	Type	Required	Description
`offerCode`	string	Required	Unique identifier for the offer
`name`	string	Required	Name of the offer
`dateRange`	string	Required	Contains the relevant dates for the offer
`dateRange.effective`	ISO 8601 timestamp	Required	Date/time at which the offer starts
`dateRange.expiry`	ISO 8601 timestamp	Optional	Date/time at which the offer expires
`dateRange.display`	ISO 8601 timestamp	Optional	Date/time at which the offer is displayed, which may be a few days before the offer starts
`dateRange.target`	ISO 8601 timestamp	Optional	Date/time at which offer targeting is

Field	Type	Required	Description
			carried out
`offerActivationType`	string	Optional	Type of offer to be activated (`LTC` or `MASS`)
`rank`	number	Optional	The priority rank of the offer
`offerType`	string	Required	Type of offer within the loyalty program (for example, `PROMO`)
`offerSubType`	string	Required	Subtype of offer (`BASE` or `BONUS`)
`businessUnit`	string	Optional	Name of business unit for this offer within the loyalty program. Required only if there is more than one business unit.
`enabled`	boolean	Required	Flags whether the offer is enabled for use (`true` or `false`)
`channel`	string	Optional	The valid channel for the offer. Set to `EXTERNAL` by default. Valid values: `POS`,

Field	Type	Required	Description
			`WEBSITE`, `CONSOLE`, `APP`, `STORE`, `PARTNER`, `EXTERNAL`, `FLYER`, `null`.
`tiers`	array	Required	Contains an object for each tier in the loyalty program
`tiers[].name`	string	Required	Name of this member tier in the loyalty program
`tiers[].precedence`	number	Required	Where the tier ranks in the precedence order
`tiers[].messages`	object	Required	Contains messages organized by language (`en` or `fr`)
`tiers[].messages.{{language}}.reward`	object	Required	Contains details of the reward
`tiers[].messages.{{language}}.reward.amount`	number	Required	Amount of the reward in points
`tiers[].messages.{{language}}.reward.type`	string	Required	Type of rewards (`FLAT` or `MULTIPLIER`)
`tiers[].thresholds`	array	Required	Contains data relevant to the

Field	Type	Required	Description
			thresholds of the offer such as minimum spend
`tiers[].thresholds[].thresholdId`	string	Optional	Unique identifier for this set of thresholds
`tiers[].thresholds[].category`	string	Required	Type of transaction required (`SPEND` or `UNIT`)
`tiers[].thresholds[].subcategory`	string	Required	Qualifying metric (`AMOUNT` or `NUMBER`)
`tiers[].thresholds[].type`	string	Required	Type of threshold (`MIN`, `PER`, or `BONUS`)
`tiers[].thresholds[].value`	number	Required	Value of the threshold
`funding`	object	Optional	Contains data about whether the offer is vendor-funded
`funding.vendorFunding`	boolean	Optional (required if `funding` is used)	Whether the offer is funded by a vendor (`true` or `false`)
`targetingCriteria`	object	Optional	Contains objects which combined

Field	Type	Required	Description
			create the offer targeting
`targetingCriteria.cart`	object	Optional	Contains additional objects relative to targeting on cart attributes
`targetingCriteria.cart.store`	object	Optional	Contains categories of targeting based on store attributes
`targetingCriteria.cart.product`	object	Optional	Contains categories of targeting based on product attributes
`ignoreGlobalExclusions`	boolean	Required	Whether to ignore the list of global product exclusions for this offer (`true` or `false`)
`metadata`	object	Optional	Contains additional data relevant to the client about the offer
`metadata.standard`	object	Optional	Contains standard attributes for metadata
`metadata.standard.reportingIdentifier`	string	Optional	Reporting identifier associated with the

Field	Type	Required	Description
			offer
`metadata.standard.reportingIdentifier2`	string	Optional	A second reporting identifier associated with the offer
`metadata.standard.vehicleNumber`	string	Optional	Identifier for a vehicle
`metadata.standard.vehicleDescription`	string	Optional	Description of the vehicle
`metadata.standard.eventNumber`	string	Optional	Related identifier for an event
`metadata.custom`	object	Optional	Contains custom attributes for metadata
`display`	object	Optional	Contains data for messaging and visualization of the offer, categorized by localized language
`display.{{language}}.boilerPlate`	string	Optional	Generic text for the promotion
`display.{{language}}.headline`	string	Optional	Headline for the promotion
`display.{{language}}.shortDescription1`	string	Optional	First line of short description

Field	Type	Required	Description
`display.{{language}}.shortDescription2`	string	Optional	Second line of short description
`display.{{language}}.longDescription`	string	Optional	Long description for this offer
`display.{{language}}.smallImageURI`	string	Optional	URI for small image
`display.{{language}}.largeImageURI`	string	Optional	URI for large image
`display.{{language}}.smallHDImageURI`	string	Optional	URI for small HD image
`display.{{language}}.largeHDImageURI`	string	Optional	URI for large HD image
`limits`	object	Optional	Contains additional objects to categorize offer limits by user, offer, and transaction
`limits.user.maxUses`	number	Optional	Limit on maximum number of uses of this offer per user (can be `null`)
`limits.user.maxPoints`	number	Optional	Limit on maximum number of points per user (can be `null`)

Field	Type	Required	Description
`limits.user.rewardValue`	number	Optional	Limit on maximum rewards per user (can be `null`)
`limits.offer.maxUses`	number	Optional	Limit on maximum number of uses of this offer (can be `null`)
`limits.offer.maxPoints`	number	Optional	Limit on maximum number of points associated with this offer (can be `null`)
`limits.offer.rewardValue`	number	Optional	Limit on maximum rewards through this offer (can be `null`)
`limits.transaction.maxUses`	number	Optional	Limit on maximum number of uses per transaction (can be `null`)
`limits.transaction.maxPoints`	number	Optional	Limit on maximum number of points per transaction (can be `null`)
`limits.transaction.rewardValue`	number	Optional	Limit on maximum rewards per transaction (can be `null`)

Field	Type	Required	Description
`status`	string	Required	Status of the offer (`ACTIVE`, `DISABLED`, or `DELETED`)
`publishedStatus`	string	Optional	Published status of the offer (`PUBLISHED` or `DRAFT`)
`isDynamic`	boolean	Optional	Whether targeting should take place when the offer starts (`true` or `false`). Default is `false`.
`isEmployeeDiscountCompatible`	boolean	Optional	Whether the offer can be used along with an employee discount (`true` or `false`)
`controlPercentage`	number	Optional	Percentage of members that should be in the control group
`promptMessages`	object	Optional	Contains messages that can be displayed to the member at different points in

Field	Type	Required	Description
			the offer fulfilment process
`promptMessages.nearPromptMessages`	string array	Optional	Messages displayed to members that are close to offer fulfilment
`promptMessages.anonymousMessages`	string array	Optional	Messages displayed to members that have not yet registered

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):


```
{
  "offerCode": "03VK11",
  "name": "Big Points HealthGlo BOGO",
  "dateRange": {
    "effective": "2025-05-31T06:00:00.000Z",
    "expiry": "2025-06-31T06:00:00.000Z"
  },
  "offerActivationType": "LTC",
  "offerType": "PROMO",
  "offerSubType": "BONUS",
  "enabled": true,
  "channel": "",
  "tiers": [
    {
      "messages": {
        "en": {
          "LOCALE": "LOCALE",
          "OFFER": "OFFER",
          "DISPLAY": "DISPLAY",
          "TRANSACTION": "TRANSACTION",
          "REWARD": "REWARD"
        }
      },
```

```
      "name": "Tiername 1",
      "precedence": 0,
      "reward": {
        "amount": 15000,
        "type": "FLAT"
      },
      "thresholds": [
        {
          "category": "SPEND",
          "subcategory": "AMOUNT",
          "type": "MIN",
          "value": 1
        }
      ]
    }
  ],
  "funding": {
    "vendorFunded": true
  },
  "targetingCriteria": {
    "cart": {
      "product": {
        "$or": [
          {
            "category": {
              "$in": ["ANL"]
            }
          },
          {
            "subcategory": {
              "$in": ["DELRV"]
            }
          }
        ]
      }
    }
  },
  "ignoreGlobalExclusions": true,
  "metadata": {
    "standard": {
      "reportingIdentifier": "reportingIdentifier",
      "reportingIdentifier2": "reportingIdentifier2",
      "vehicleNumber": "123",
      "vehicleDescription": "Test",
      "eventNumber": "123"
```

```
    },
    "custom": {}
  },
  "display": {
    "en-CA": {
      "boilerPlate": "Boilerplate text",
      "promotionHeadline": "Earn 5,000 Points",
      "shortDescription1": "When you buy 2 HealthGlo Vitamins",
      "shortDescription2": "When you buy 2 HealthGlo Vitamins",
      "longDescription": "Get 5,000 bonus points when you purchase two HealthGlo
Vitamins or Herbal Remedies in a single transaction.",
      "smallImageURI": "https://example.com/offers/25PP01.png",
      "largeImageURI": "https://example.com/offers/25PP02.png",
      "smallHDImageURI": "https://example.com/offers/25PPHD01.png",
      "largeHDImageURI": "https://example.com/offers/25PPHD02.png"
    }
  },
  "limits": {
    "user": {
      "maxUses": 1,
      "maxPoints": null,
      "rewardValue": null
    },
    "offer": {
      "maxUses": 10,
      "maxPoints": null,
      "rewardValue": null
    },
    "transaction": {
      "maxUses": 10,
      "maxPoints": null,
      "rewardValue": null
    }
  },
  "status": ""
}
```


Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Discretionary feed

To be used in conjunction with Exchange Solution's bulk discretionary loader. Used to increment or decrement the point balances in a number of member accounts in a single operation.

Feed name, folder, and file format

- **Feed name:** `DISCRETIONARY`
- **Format:** Pipe-delimited
- **Example filename:** `DISCRETIONARY_20220331_S1_V1` (can be without a file extension, or `.txt` or `.csv`, as long as the contents are pipe-delimited values)
- **SFTP folder:** `discretionary/`

The format for the pipe-delimited file must include all pipes, including those for missing values.

Example pipe-delimited records:

```
704826434834|10000|R1|2024-07-11T15:03:57.730Z|a2cc5dab-1ed4-4983-bd29-
eb0469a4349c|PATCH|Epsilon_514355278
705355452502|15000|MISSING_POINTS||||
|200|Missing rewards from promo||||12345
```

Note that headers are not used.

Frequency

This feed can be accepted daily but may be sent less frequently (weekly or monthly).

Feed contents

Field	Type	Required	Description
`loyaltyId`	string	Required (at least one of `loyaltyId` or `externalIdentifier` must be provided)	Unique per card number but not validated for format

Field	Type	Required	Description
`pointAmount`	number	Required	The quantity of points to be awarded (or clawed back if a negative number)
`reasonCode`	string	Required	Code for the reason for the increment or decrement. Values are not validated, so they can be values that are meaningful to the client or just placeholders, including blanks or nulls.
`transactionDate`	string	Optional (required if `mode` is `PATCH`)	The date/time in ISO 8601 format of the original transaction (if mode is `STANDARD`) or the current transaction (if mode is `PATCH`)
`sessionId`	string	Optional (required if `mode` is `PATCH`)	The current unique session identifier or correlation ID, in GUID format

Field	Type	Required	Description
`mode`	string	Optional	Identifies whether the discretionary transaction is created with the current transaction date and a randomized `sessionId` (`STANDARD` — updates the balance without hold) or with the provided `sessionId` and transaction date/time (`PATCH` — used to update information used to explain balance changes; balance is not updated). `STANDARD` is the default.
`externalIdentifier`	string	Optional (at least one of `loyaltyId` or `externalIdentifier` must be provided)	Unique identifier used by the client to identify the account. Must be the last value in the feed record if used.

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Exclusion feed

The `EXCLUSION` feed establishes and updates a global product exclusion list for a loyalty program.

Feed name and folder

- **Feed name:** `EXCLUSION`
- **Format:** NDJSON

- **Example filename:** `EXCLUSION_20230331_S1_V1.json`
- **SFTP folder:** `businessUnit/<businessUnitName>/exclusions/`

Frequency

This feed can be accepted once per day, but is generally updated rarely.

Feed contents

The `EXCLUSION` feed provides product exclusion data (using Pascal case capitalization rules as specified in the conventions section of this document):

Field	Type	Required	Description
`BusinessUnit`	string	Required	The business unit associated with the excluded product
`ExclusionType`	string	Required	The level of the product hierarchy at which the exclusion is targeted (for example, `"subCategory"` or `"category"`)
`ExclusionCode`	string	Required	The product hierarchy code matching the type
`ExclusionName`	string	Optional	Human-readable name for the exclusion
`EligibleBase`	boolean	Required	Indicates whether base points are eligible (`true` or `false`)
`EligibleBonus`	boolean	Required	Indicates whether bonus points are eligible (`true` or `false`)
`EligibleRedemption`	boolean	Required	Indicates whether bonus points can be redeemed for this kind of product (`true` or `false`)
`ProvinceState`	string	Optional	A provincial or state code limiting the exclusion to retail locations in that province, state, or

Field	Type	Required	Description
			territory. If the two-letter code is incorrect and cannot be validated, the record is rejected.
`RegionType`	string	Optional	Used to denote the country (currently `"Province"` for Canada or `"State"` for the United States). If a value is not provided, the country is determined via configuration. If an incorrect value is provided, the record is rejected.
`ExtendedData`	object	Optional	Extended data about the exclusion (only if exclusions are province-specific based on store location)

Exclusion extended data

Extended data can be provided; however, it will not be acted upon without product customization. Any key-value is allowable, but the maximum data size for extended data for a given location is five kilobytes, consisting of up to 20 unique root-level keys across all members.

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "BusinessUnit": "KINGREX",
  "ExclusionType": "subCategory",
  "ExclusionCode": "32150",
  "ExclusionName": "SMOKING ACCESSORIES",
  "EligibleBase": false,
  "EligibleBonus": false,
  "EligibleRedemption": false,
  "ProvinceState": "AB",
  "RegionType": "Province",
  "ExtendedData": {
    "franchise": true,
    "superStore": false
  }
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Historical transaction feed

There are several purposes for loading historical transaction data into the ES Loyalty™ platform: purchase history targeting, analytical segmentation or analysis, seeding purchase data for ES Loyalty Boost™, and viewing the transaction history for redemptions (online, partner, and POS), donations, discretionary (earn and burn), online earn, and partner earn. All these use cases require the historical data to be in Snowflake instead of the Exchange Solutions operational system, so that is the destination for this data.

Supported inputs

The solution supports two ways to deliver historical data. The historical data can either be sent via a JSON flat file through an FTP site or through Snowflake Data Sharing.

Data feed option

If the data feed option is selected, the historical purchase feed is sent via Exchange Solutions' FTP site. The file goes through the normal cryptography service and is transferred in the Snowflake ADS ingestion S3 bucket. From there, the normal process for loading data using Snowpipe is used.

Non-POS transactions

The current design was intended to support historical purchase transactions but, to have a complete view of transaction history, the following transaction types also need to be supported:

- Redemption (online, partner, and POS)
- Donation
- Discretionary (earn and burn)
- Online earn
- Partner earn

POS transactions

The current design doesn't include points calculated by an external loyalty program. The base points, bonus for each promotion, and redeemed points for each redemption must be included for POS transactions. The partner information must also be included within the transactions, including the `partner_id` and/or `promotion_id` list and the `partner_link_id` (if the client has the hash of the `link_id`). For CPL (cents per litre) and gift card redemptions, the expectation is that these are sent as tender.

Historical purchase feed layout

The retail `HISTORICAL_TRANSACTION` feed receives information from the point-of-sale and/or eCommerce solution containing the historical purchase and cart information of each retail transaction. This data contains all the purchase information received before the loyalty program was initiated.

Feed name and folder

- **Feed name:** `HISTORICAL_TRANSACTION`
- **Format:** NDJSON
- **Example filename:** `HISTORICAL_TRANSACTION_20230331_S1_V1.json`
- **SFTP folder:** `transaction/`

Frequency

This feed can be accepted as a one-time load before the program is initialized. It can also be received at regular intervals to include historical purchase information for members as they are onboarded, if applicable to the program.

Feed contents

If a `MEMBER` record is provided that doesn't exist in ES Loyalty, a placeholder account is created that will be linked to the corresponding ES Loyalty account when it's created.

Purchases

The `HISTORICAL_TRANSACTION` feed provides standard transaction data (using camel case capitalization rules as specified in the conventions section of this document):

Field	Type	Required	Descripti
`businessUnit`	string	Required (if business unit is applicable)	The unit of t overall busin associated w the purchase transaction
`action`	string	Required	Type of transaction: `"PURCHASE"` `"ADJUSTMEN` `"REDEMPTIO` `"DONATION"` `"DISCRETIO` or `"TRANSFE`
`channel`	string	Required	Channel thro which the transaction occurred: `"STORE"`, `"ECOMMERCE` `"ONLINE"`, `"PARTNER"`, `"CALL_CENT`
`correlationID`	string	Required	The transacti associated w the transacti there is a cha of overlap w real-time transaction, t `transactio` must match

Field	Type	Required	Descripti
			real-time transaction.
`externalTransactionID`	string	Optional	The transacti that the clier associates w transaction
`transactionDate`	string	Required	A datetime v the transacti took place
`retailBanner`	string	Optional	A store grou or type
`storeNumber`	string	Required	Referencing store/ecomn site from the location feed
`tillNumber`	string	Optional	Referencing specific till ir store
`deviceType`	string	Optional (strongly recommended for analytics)	The type of d used to creat transaction: `"INSTORE PO` `"SELF-CHEC` `POS"`, `"PUMP` `"APP"`, `"IPH` `APP"`, `"ANDR` `APP"`, `"ONLI` `"CALL CENT`

Field	Type	Required	Descripti
`employeeCode`	string	Optional (strongly recommended for analytics)	Referencing employee wh handled the transaction
`externalAccountID`	string	Required (optional if `loyaltyID` is used)	The external account ID o customer
`loyaltyID`	string	Required (optional if `externalAccountID` is used)	The loyalty identifier of t customer
`partnerID`	string	Optional	The partner identifier link the transacti
`discretionary_agent_Id`	string	Optional (strongly recommended for discretionary transactions)	The call cent agent who performed th transaction
`transactionPointTypes`	array	Optional	A JSON array containing a further break of the types points includ a transaction Required onl the net point earned or bu are not zero.
`transactionPointTypes[].pointType`	string	Optional	The type of p being applie

Field	Type	Required	Descripti
			`"BASE"`, `"BO` `"REDEEM"`, `"DISCRETIO` `"EXPIRY"`, `"DONATION"`
`transactionPointTypes[].amount`	integer	Optional	The number points. Inclu negative sig when points removed or
`transactionPointTypes[].rewardID`	string	Required (for `BONUS` or `DISCRETIONARY` type)	When `point` is `BONUS` or `DISCRETIONA` include the identifier for type of bonus/prom or a reason f discretionary points given `redemption` goes into `rewardID` w the `pointTy` `BONUS`.
`transactionPointTypes[].rewardName`	string	Required (when `pointType` is `BONUS` or `DISCRETIONARY`)	Human-read name for eac distinct `rewa`

Field	Type	Required	Descripti
`transactionPointTypes[].redemptionTypeID`	string	Required (when `pointType` is `REDEEM`)	When `point` is `REDEEM`, ir the redempt identifier
`cart`	object	Required	A JSON obje containing th
`cart.saleLineItems`	array	Required	A JSON array individual sa items
`cart.saleLineItems[].subcategory`	string	Required	Unique ident for the produ subcategory relevant to th SKU
`cart.saleLineItems[].sku`	number	Required	The product identifier
`cart.saleLineItems[].quantity`	number	Required	The number SKUs of this in the basket
`cart.saleLineItems[].originalSaleAmount`	number	Required	Undiscounte amount in ce
`cart.saleLineItems[].saleAmount`	number	Required	The discount amount
`cart.saleLineItems[].itemDiscount`	number	Required	Discounts ap in cents

Field	Type	Required	Descripti
`cart.saleLineItems[].itemTax`	number	Optional	Applicable sa tax
`cart.saleLineItems[].finalSaleAmount`	number	Required	Final discour amount with tax on the ite
`cart.saleLineItems[].tags`	array	Optional	An array containing extended da about the lin
`cart.saleLineItems[].redemptionFlag`	boolean	Required	`true` for iter marked as redeemed ite For dollar-of redemptions flag is `false` is not applie directly to ar individual pr
`cart.totalSaleAmount`	number	Required	The total sal amount of al items exclud tax
`currency`	string	Required	The currency transaction i (for example `"CAD"`)

Field	Type	Required	Descripti
`tender`	map object	Optional (strongly recommended for analytics)	A map conta tender information. map key can either a cent or a more de tender objec an array of t objects.

Tender object attributes:

Field	Type	Description
`idType`	enum	Type of ID used: `"TOKEN"` (default), `"LAST4"`, or `"PREFIX_LAST4"`
`id`	string	ID used to identify the card (for example, `"1234"`, `"123456TOKEN1234"`). TOKEN IDs are expected to be unique hashed or tokenized payment card numbers. Exchange Solutions is not responsible for clear-text data passed into this field. Required for TOKEN.
`prefix`	string	BIN of the card involved (for example, `"123456"`). Required for `PREFIX_SUFFIX`.
`suffix`	string	Last four digits (including the check digit) of the card (for example, `"1234"`). Required for `PREFIX_SUFFIX`.
`amount`	number	Tender amount in the required currency format, in cents (for example, `4000`)

Example purchase NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "action": "PURCHASE",
  "channel": "STORE",
  "correlationID": "29Ee-399RD-293-2",
  "externalTransactionID": "client_trans_00001",
  "transactionDate": "2022-03-10T13:58:45-05:00",
  "retailBanner": "OUTLET",
  "storeNumber": "2378",
  "tillNumber": "04",
  "deviceType": "INSTORE POS",
  "employeeCode": "11120",
  "externalAccountID": "23442344432",
  "loyaltyID": "602778234423444329",
  "partnerID": "PARTNER_1",
  "transactionPointTypes": [
    {
      "pointType": "BASE",
      "amount": 100
    },
    {
      "pointType": "BONUS",
      "amount": 100,
      "rewardID": "2XPOINTS",
      "rewardName": "2X Points"
    },
    {
      "pointType": "REDEEM",
      "amount": -1000,
      "redemptionTypeID": "$10OFF"
    }
  ],
  "cart": {
    "saleLineItems": [
      {
        "subCategory": "44100",
        "sku": "73385499157",
        "quantity": 1,
        "originalSaleAmount": 4500,
        "saleAmount": 4500,
        "itemDiscount": 0,
        "itemTax": 0,
        "finalSaleAmount": 2000,
        "tags": []
      }
```

```
    ],
    "totalSaleAmount": 2000,
    "currency": "CAD"
  },
  "tender": {
    "CASH": "250",
    "VISA": {
      "idType": "PREFIX_LAST4",
      "prefix": "123456",
      "suffix": "3312",
      "amount": "250"
    },
    "MC": [
      {
        "idType": "TOKEN",
        "id": "123456Token1234",
        "amount": "250"
      },
      {
        "idType": "TOKEN",
        "id": "123123Token1231",
        "amount": "250"
      }
    ],
    "LOYALTY": "1000"
  },
  "businessUnit": "BUYCO"
}
```

Adjustments

> **Note:** Adjustments are optional if the purchase contains the final state of the transaction after an adjustment is performed — in that case, including the final state is preferred.

For adjustments, the remaining keys (other than the `"action"` key) are:

Field	Type	Required	Description
`originalCorrelationID`	string	Required	The unique correlation/transaction ID of the previous

Field	Type	Required	Description
			transaction this transaction is adjusting
`currentCorrelationID`	string	Required	The unique correlation/transaction ID of this adjustment
`channel`	string	Required	Channel through which the original transaction was made (`"STORE"` or `"ECOMMERCE"`)
`retailBanner`	string	Required	The banner of the store (for example, `"OUTLET"`)
`storeNumber`	string	Required	The code of the store
`tillNumber`	number	Optional	An identifier for a specific till
`employeeCode`	number	Optional	A code identifying the employee at the till
`transactionPointTypes`	array	Optional	A JSON array containing a further breakdown of the types of points included in the transaction. Required only if the net points earned or burned are not zero.
`cart`	object	Required	A JSON object containing the cart

Field	Type	Required	Description
`cart.reversalLineItems`	array	Required	—
`cart.reversalLineItems[].sku`	number	Required	The SKU being reversed
`cart.reversalLineItems[].quantity`	number	Required	The number of SKU items being reversed
`cart.totalSaleAmount`	number	Optional	The updated total sale amount after removing the reversed line items from the basket
`currency`	string	Optional	The currency for the sale amount fields
`tender`	object	Optional (strongly recommended for analytics)	Method of payment

> **Note:** Exchanges of different items are expected to come as an adjustment removing the exchanged item, followed by a purchase transaction of the new items. If an exchange consists of an identical cart (for example, different sizes), a purchase can be sent with `$0` items to net 0 points.

Example adjustment NDJSON object (formatted for readability only — one object per line followed by CRLF):


```
{
  "action": "ADJUSTMENT",
  "originalCorrelationID": "2342334",
  "currentCorrelationID": "2344442433",
  "channel": "STORE",
  "transactionDate": "2022-03-10T13:58:45-05:00",
  "retailBanner": "OUTLET",
  "externalAccountID": "23442344432",
```

```
  "storeNumber": "0034",
  "tillNumber": "04",
  "employeeCode": "11120",
  "transactionPointTypes": [
    {
      "pointType": "BASE",
      "amount": -100
    },
    {
      "pointType": "BONUS",
      "amount": -100,
      "rewardID": "2XPOINTS",
      "rewardName": "2X Points"
    }
  ],
  "cart": {
    "reversalLineItems": [
      {
        "sku": "73385499157",
        "quantity": 1
      }
    ],
    "totalSaleAmount": 2260,
    "currency": "CAD"
  },
  "tender": {
    "VISA": {
      "idType": "PREFIX_LAST4",
      "prefix": "123456",
      "suffix": "3312",
      "amount": "500"
    },
    "LOYALTY": "0"
  }
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

The possible reasons for a rejected record for this feed include:

Rejection message	Reason for rejection
`<Required_Field>` is mandatory and required to be non-empty.	If any of the required fields are not present in the request, the request is rejected. Required fields: `transactionID`, `mappedRequestAction`, `identifierType`, `identifierValue`, `netAmount`, `transactionParts`, `channel`, `processedDate`, `transactionDate`.
Invalid processed date.	Invalid `processedDate`.
Invalid transaction date.	Invalid `transactionDate`.
Invalid request action.	Invalid `mappedRequestAction`. Valid values: `FINALIZE`, `PPE`, `SAF`, `DISCRETIONARY`, `ACTIVITY`, `VOID`, `ADHOC_REWARD`, `ADHOC_REDEEM`.
Invalid channel.	Invalid channel. Valid values: `POS`, `APP`, `WEBSITE`, `CONSOLE`, `STORE`, `PARTNER`.
Invalid transaction part action.	Invalid `transactionParts` action. Valid values: `ADJUSTMENT`, `BASE`, `BONUS`, `TARGETED`, `DISCRETIONARY`, `EXPIRY`, `TRANSFER_IN`, `TRANSFER_OUT`, `REDEEM`, `BALANCE_UPDATE`.
Unknown loyalty identifier type provided.	Invalid `identifierType`. Valid values: `LOYALTY_ID`, `ACCOUNT_ID`, `EXTERNAL_ID`, `PARTNER_LINK_ID`.
Invalid account identifier.	Invalid `LOYALTY_ID`, `ACCOUNT_ID`, `EXTERNAL_ID`, or `PARTNER_LINK_ID`.
Discretionary amount is mandatory and must be a valid non-zero number.	If `transactionParts` has `action = DISCRETIONARY`, then `partDetails` must have a valid numeric value in the `amount` attribute.
`reasonCode` is mandatory and required to be non-empty.	If `transactionParts` has `action = DISCRETIONARY`, then `partDetails` must have `reasonCode` inside `metaData`.

Rejection message	Reason for rejection
Invalid overall request format.	If `netAmount` is not of type Number.
Invalid overall request format.	If the request has `transactionAmount` and it is not of type Number.
Invalid overall request format.	If the request has `channelDetails` and it is not of type Object.

Issuance feed

The `ISSUANCE` feed is used to provide information from the client about member issuances such as voucher information. For instance, if a client issues vouchers quarterly, they would use this feed to report those voucher issuances back into the Exchange Solutions system.

Feed name and folder

- **Feed name:** `ISSUANCE`
- **Format:** NDJSON
- **Example filename:** `ISSUANCE_20230331_S1_V1.json`
- **SFTP folder:** `businessUnit/<businessUnitName>/issuance/`

Frequency

This feed can be accepted once per day. It provides information regarding member issuances such as the type, amount (dollar value), and expiry data of rewards issued, such as vouchers.

Feed contents

Field	Type	Required	Description
`loyaltyID`	string	Required (mutually exclusive with `externalIdentifier`)	Unique per card number but not validated for format
`externalIdentifier`	string	Optional (mutually exclusive with `loyaltyID`)	The member that this event pertains to, using client identifier
`businessUnit`	string	Optional	Business unit ID (must exist and be enabled). Falls back to default if not provided.
`issuanceType`	string	Optional	Type of issuance (currently supported value: `VOUCHER`)
`issuanceCode`	string	Optional	The unique code per issuance type
`issuedDate`	string	Required	The date/time when the issuance is issued, in ISO 8601 format (for example, `2023-01-10T13:58:45-05:00`)
`expiryDate`	string	Required	The date/time when the issuance expires, in ISO 8601 format (for example, `2023-12-10T13:58:45-05:00`)
`redemptionDate`	string	Optional	The date/time when the issuance is redeemed, in ISO 8601 format (for example, `2023-01-16T13:58:45-05:00`), if available

Field	Type	Required	Description
`issuanceStatus`	string	Required	Status of the issuance: `"AVAILABLE"`, `"REDEEMED"`, or `"EXPIRED"`
`issuanceRewardType`	string	Required	Type of reward: `"POINTS"` or `"DOLLARS"`
`issuanceRewardValue`	number	Required	Reward amount
`notes`	string	Optional	Passes additional information about the issuance (for example, `"Q1 2023"` as a note in a voucher). Format: Only alphanumeric characters and symbols `_.@+-$&'`, French letters, and spaces are allowed; length must be 3 to 128 characters.
`extendedData`	object	Optional	Additional data about the issuance

Extended data

Extended data can be provided; however, it will not be acted upon without product customization. Any key-value pair is allowable, but the maximum data size for extended data for a given event is five kilobytes, consisting of up to 20 unique root-level keys.

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "loyaltyID": "23828329",
  "businessUnit": "BUYCO",
  "issuanceType": "VOUCHER",
  "issuanceCode": "WELCOME20PERCENTOFFER",
```

```
  "issuedDate": "2023-01-10T13:58:45-05:00",
  "expiryDate": "2023-12-10T13:58:45-05:00",
  "issuanceStatus": "AVAILABLE",
  "issuanceRewardType": "DOLLARS",
  "issuanceRewardValue": 200,
  "extendedData": {},
  "notes": "This is a Q3 2023 voucher."
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Member feed (enrollment/update)

The `MEMBER` feed contains customer enrollment information and first-class data fields for ES Loyalty. This feed is for adding and updating members. Deleting and disabling users can be achieved via the ES Loyalty Console.

This feed is used to pass in the unique identifier. If the ID — for instance, a `loyaltyID` — doesn't match an existing client loyalty card, then the record is rejected with a message such as: `"Identifier $(loyaltyID) does not exist or is not registered."`

An optional `registrationDate` attribute is supported. If present, this data is used for `registrationDate` in the Account record (`Registration.Details.registrationDate`); otherwise, the system-generated date is used.

The results for account creation from ingesting this feed are shown in the table below:

SOR configuration	Only LoyaltyID in request	Exists in card-pool table	isClaimed	isAllocated	Result
`loyaltyID: false` `pointsBank: false` `inTransition: true`	Yes	No	N/A	N/A	Create an unregistered account
	Yes	Yes	N/A	true	Create an unregistered account and update `isClaimed` with a CardID
	Yes	Yes	N/A	N/A	Create an unregistered account, set `isClaimed` and `isAllocated` to `true`
	No	No	N/A	N/A	Create an active account with details provided and take the `registrationDate` provided in the request
	No	Yes	N/A	true	Create an active account with details provided and set `isClaimed` to `true`; also take the `registrationDate` provided in the request

SOR configuration	Only LoyaltyID in request	Exists in card-pool table	isClaimed	isAllocated	Result
	No	Yes	N/A	N/A	Create an active account with details provided and set `isClaimed` and `isAllocated` to `true`; also take the `registrationDate` provided in the request
	No	Yes	true	true	Existing account is updated
	Yes	Yes	true	true	If only the `loyaltyID` is in the request and it was already claimed, this account is updated. Since only the `loyaltyID` is passed, `firstName`, `lastName`, etc. are set to null, even if they had been set to other values earlier.
`loyaltyID: true` `pointsBank: true` `inTransition: false`	Yes	Yes	false	true	Create a new active account and set `isClaimed` to `true`

SOR configuration	Only LoyaltyID in request	Exists in card-pool table	isClaimed	isAllocated	Result
	No	Yes	true	true	Update the existing account
	Yes	Yes	true	true	If only the `loyaltyID` is in the request and it was already claimed, this account is updated. Since only the `loyaltyID` is passed, `firstName`, `lastName`, etc. are set to null, even if they had been set to other values earlier.
	Yes	Yes	true	true	Includes `loyaltyID` and other information such as `emailAddress` and `firstName`. A new account is created, `isClaimed` is set to `true`, and `registrationDate` is set as the system date.

Feed name and folder

- **Feed name:** `MEMBER`
- **Format:** NDJSON
- **Example filename:** `MEMBER_20240331_S1_V1.json`
- **SFTP folder:** `member/`

Frequency

This feed can be accepted once per 15-minute period. Any transactions referencing newly enrolled members must arrive after the receipt of this file. Transactions for members without loyalty IDs can be sent with the special `loyaltyID` keyword `"ANONYMOUS"`.

Feed contents

The `MEMBER` feed provides standard customer profile data (using camel case capitalization rules as specified in the conventions section of this document):

Field	Type	Required	Description
`loyaltyId`	string	Required (mutually exclusive with `externalIdentifier`)	Unique per card number but not validated for format
`externalIdentifier`	string	Required (mutually exclusive with `loyaltyId`)	The member that this event pertains to, using client identifier
`registrationDate`	string	Optional	The date the member registered in the loyalty program, in ISO 8601 format
`businessName`	string	Optional	The name of a business associated with the member
`emailAddress`	string	Required (must be non-empty)	Validated as unique per member, case-insensitive
`firstName`	string	Optional	First name of the member
`lastName`	string	Optional	Last name of the member
`birthYear`	number	Optional for operations, required for analytics	Birth year of the member

Field	Type	Required	Description
`birthDate`	string	Optional for operations, required for analytics	Birth date of the member (format: `"MM-DD"`)
`gender`	string	Optional for operations, required for analytics	Gender of the member: `"M"`, `"F"`, `"Other"`, or `"PreferNotToSay"`
`address1`	string	Optional	Address of the member
`address2`	string	Optional	Additional address information
`city`	string	Optional for operations, required for analytics	City of residence of the member
`provinceState`	string	Optional for operations, required for analytics	Province or state of residence for the member
`postalZipCode` or `postalZip`	string	Required for analytics (must be non-empty)	Postal or Zip code of the member
`phoneNumber`	string	Optional	Phone number of the member. See the Phone Number Locking and Account Association section below.
`languagePreference`	string	Optional	Preferred language of the member for communications: `"EN"`, `"FR"`, or `"ES"`
`extendedDataSource`	string	Optional (used with multi-source EMD only)	Identifies the source of the `extendedData`. Example values: `"MARKETING"`, `"PARTNERSHIP"`, `"SUPPLY_MANAGER"`, `"ES-`

Field	Type	Required	Description
			`LOYALTY"`, `"PPN"`, `"BANKING_INFO"`.
`extendedData`	object	Optional	Additional data that the client wants included in reporting
`flags`	object	Optional	Additional flags for the member
`referralCode`	string	Optional	Unique code that a member can provide to a potential member for enrollment. If passed, the code is validated, the referrer and referee relationship object is created, and the referrer's counter for tracking purposes is updated.
`emailAddressBook`	object	Optional	Allows store owners' email addresses to be used in place of the loyalty member account

Using externalIdentifier with registrationDate

For some clients, accounts can be registered and made active (`ACTIVE` account status and `ACTIVE` loyalty status with a digital card number pulled from the card pool) if there is a value for `externalIdentifier` AND a valid value for `registrationDate`. The `registrationDate` value must be what is provided by the client in the feed. If an `externalIdentifier` is passed in the feed but a `registrationDate` is not passed, then the account status is `UNREGISTERED` but the loyalty status is `NON-LOYALTY`. On registration, the account status changes to `ACTIVE`. The `externalIdentifier` must be unique for each account.

Phone number locking and account association

When a phone number is associated with an account, a lock is placed on the phone number to prevent its use for another account. If an account with a new phone number is added, then the number is locked on that account. If a new account is added with an existing phone number already

associated with another account, the new account gets the phone number and the lock, and the old account loses both (it will have no phone number). If a phone number is deleted from an account, then the lock and the phone number are removed.

Member extended data

Extended data is optional program-specific data about the customer that can be used for offer targeting (for example, an analytical segment) or accessed by a customer service agent or salesperson from the console (for example, a shirt size). Any key-value pair is allowable (camel case naming convention is strongly recommended), but the maximum data size for extended data for a given customer is five kilobytes, consisting of up to 20 unique root-level keys across all members.

Here is an example of extended data fields that might be used with `extendedDataSource` set to `"BANKING_INFO"`:

```
"extendedDataSource": "BANKING_INFO",
"extendedData": {
  "accountNumber": "accountNumber001",
  "accountHolderType": "VIP",
  "bankName": "MEGABANK",
  "institutionNumber": "001",
  "transitNumber": "03438"
}
```

Email address book

An object named `EmailAddressBook` is used to allow alternate or additional email addresses for notifications about member redemptions to be stored as a top-level attribute in the Account object. This feature provides the ability for a client's relevant internal contacts, such as store owners, to receive notification about member redemptions.

If used, the structure of `EmailAddressBook` within the feed is:

```
{
  // Account object
  "EmailAddressBook": {
    "fulfillmentEmails": {
      "<identifier>": "<valid_email_address>",
      "owner_1": "owner@clientUS.com",
      "owner_2": "owner2@clientCA.com"
```

```
    }
  }
}
```

Flags

The `flags` property is an object that contains optional flags about the customer. Currently, only one flag is supported: `ghostRedeemer`.

- If the member is a ghost account and `Flags.ghostRedeemer` is `true`, `ghostRedeemer` is saved in the member's profile and the member can redeem.
- If the member is a ghost account and `Flags.ghostRedeemer` is `false`, `ghostRedeemer` is saved in the member's profile and the member cannot redeem.
- If the member is not a ghost account, the attribute is ignored regardless of the value. If the feed is updating a member's profile, this attribute is kept in the member's profile if it already exists.

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):


```
{
  "loyaltyId": "7069000000012602",
  "registrationDate": "2023-03-09T21:00:00Z",
  "businessName": "BUYCO",
  "firstName": "Bob",
  "lastName": "Smith",
  "emailAddress": "bob@domain.com",
  "birthYear": "YYYY",
  "birthDate": "MM-DD",
  "gender": "M",
  "address1": "123 Any Street",
  "address2": "Unit 123",
  "city": "Boston",
  "provinceState": "MA",
  "postalZip": "02101",
  "phoneNumber": "555123432",
  "languagePreference": "en-CA",
  "extendedDataSource": "MARKETING",
  "extendedData": {
    "listOfShirtSizes": ["medium", "large"],
    "coefficientX": 0.023,
    "memberSince": "2020-07-10T13:58:45-05:00",
    "segment1": "cherry_pickers"
```

```
  },
  "flags": {
    "ghostRedeemer": true
  },
  "referralCode": "X4G7B8QP",
  "emailAddressBook": {
    "fulfillmentEmails": {
      "owner_1": "harrypotterbuyco1@buyco.com",
      "owner_2": "harrypotterbuyco2@buyco.com"
    }
  }
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Member balance update feed

The `MEMBER_BALANCE_UPDATE` feed is used to set up the point balance for members imported into the system. It can also be used to update the balance post-launch in case of balance errors.

Feed name and folder

- **Feed name:** `MEMBER_BALANCE_UPDATE`
- **Format:** NDJSON
- **Example filename:** `MEMBER_BALANCE_UPDATE_20230331_S1_V1.json`
- **SFTP folder:** `member/`

Frequency

This feed can be accepted once per 15-minute period. One of the following fields must be sent to identify an account: `loyaltyID` or `externalIdentifier`. If the account is not found, the record is rejected.

Feed contents

Field	Type	Required	Description
`loyaltyId`	string	Required (mutually exclusive with `externalIdentifier`)	Unique per Exchange Solutions account but not validated for format
`externalIdentifier`	string	Required (mutually exclusive with `loyaltyId`)	The member that this event pertains to, using client identifier
`correlationId`	string	Required	Unique identifier for the transaction
`pointsAmount`	number	Required	Points to be added or removed to/from the member's balance; negative values are accepted
`balanceDate`	string	Required	Date (ISO 8601) when the member had the related "points" as balance

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "loyaltyId": "2382832944324324",
  "correlationId": "PB1234567",
  "pointsAmount": 30000,
  "balanceDate": "2023-03-09T21:00:00Z"
}
```

Reject feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

The possible reasons for a rejected record for this feed include:

Rejection message	Reason for rejection
Identifier not provided.	Neither `loyaltyId` nor `externalIdentifier` provided
Correlation ID not provided.	`correlationId` not provided
Points Amount not provided.	`pointsAmount` not provided
Balance Date not provided.	`balanceDate` not provided
Account not found.	Account not found by given identifier
Skipped	`transactionId` with same `correlationId` exists

Member code feed

Provides referral codes to specified accounts for a referral program.

Feed name and folder

- **Feed name:** `MEMBER_CODE`
- **Format:** NDJSON
- **Example filename:** `MEMBER_CODE_12_11_V2.json`
- **SFTP folder:** `member/`

Frequency

This feed is run when required by the client to add referral codes to existing members.

Feed contents

Field	Type	Required	Description
`identifierType`	string	Required	Identifier for the type of information contained in the `identifierValue`. Must be `"ACCOUNT_ID"`.

Field	Type	Required	Description
`identifierValue`	string	Required	`ACCOUNT_ID` identifier value
`program`	string	Required	Name of the referral program. Unless changed, the value is `"REFERRAL_PROGRAM"`.

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "identifierType": "ACCOUNT_ID",
  "identifierValue": "4c81d334-cb77-4285-a2c4-1e24cda88bc5",
  "program": "REFERRAL_PROGRAM"
}
```

Reject feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Member extended data feed (changes only)

Contains only extended member data for ES Loyalty. This feed updates extended member data and doesn't affect other first-class attributes. Deleting and disabling users can be achieved via the ES Loyalty Console.

Feed name and folder

This feed can be uploaded either as a JSON file through the regular feeds process or as a CSV file in the Console.

JSON format:

- **Feed name:** `MEMBER_EXTENDED_DATA`
- **Format:** NDJSON
- **Example filename:** `MEMBER_EXTENDED_DATA_20230331_S1_V2.json`

- **SFTP folder:** `member/`

CSV format:

The EMD data can be uploaded as a CSV file through the Console (Data Management > Extended Member Data).

- **Feed name:** `MEMBER_EXTENDED_DATA`
- **Example filename:** `MEMBER_EXTENDED_DATA_20231106_S1_V1.csv`

Frequency

JSON format: This feed can be accepted daily. Any transactions referencing newly enrolled members must arrive after the receipt of this file.

CSV format: The CSV file can be uploaded in the Console as required.

Feed contents (JSON format)

Field	Type	Required	Description
`loyaltyID`	string	Required (mutually exclusive with `externalIdentifier` and `accountID`)	Unique per card number but not validated for format
`externalIdentifier`	string	Required (mutually exclusive with `loyaltyID` and `accountID`)	The member that this event pertains to, using client identifier
`accountID`	string	Required (mutually exclusive with `loyaltyID` and `externalIdentifier`)	Unique per account
`extendedDataSource`	string	Optional (used with multi-source EMD only)	Identifies the source of the `extendedData`. Example values: `"MARKETING"`, `"PARTNERSHIP"`, `"SUPPLY_MANAGER"`, `"ES-`

Field	Type	Required	Description
			`LOYALTY"`, `"PPN"`, `"BANKING_INFO"`.
`extendedData`	object	Optional	Additional data about the member (key-string values inside)

Extended data

This data is optional program-specific data about the consumer that can be used for offer targeting (for example, an analytical segment) or accessed by a customer service agent or salesperson from the console (for example, a shirt size). Any key-value is allowable (camel case naming convention is strongly recommended), but the maximum data size for extended data for a given consumer is five kilobytes, consisting of up to 20 unique root-level keys across all members.

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "loyaltyID": "28282828282",
  "extendedDataSource": "SUPPLY_MANAGER",
  "extendedData": {
    "label": "Supply Manager",
    "default": true
  }
}
```

Example NDJSON object (for passing subscription blocking status for email, phone calls, and SMS messages):

```
{
  "loyaltyID": "28282828282",
  "extendedData": {
    "casl": "false",
    "doNotEmail": "false",
    "doNotPhone": "false",
    "doNotText": "false"
```

```
  }
}
```

Example NDJSON object (providing member banking data using `"BANKING_INFO"` as external data source with external identifier as the unique identifier):

```
{
  "externalIdentifier": "50fad209-18dd-4df0-8c6c-49b92e31580a",
  "externalDataSource": "BANKING_INFO",
  "extendedData": {
    "accountNumber": "account001",
    "accountHolderType": "VIP",
    "bankName": "MEGABANK",
    "institutionNumber": "001",
    "transitNumber": "38564"
  }
}
```

Reject feed (JSON format)

Rejection message	Reason for rejection
`UNKNOWN_LOYALTY_ID`	Member extended data feed with invalid `loyaltyId`
`EXTENDED_DATA_REQUIRED`	Member extended data feed without any extended data
`LOYALTY_ID_REQUIRED`	Member extended data feed without any `loyaltyId`
Exceeded root level key limit for extended data: `{x}` keys	Limit on keys or attributes for extended data exceeded

Feed contents (CSV format)

Field	Type	Required	Description
`identifierType`	string	Required (mutually exclusive with `externalIdentifier`)	Identifies the type of the unique identifier per row of data
`identifierValue`	string	Required	Provides a value for the unique identifier; identifies the member record to be updated
`source`	string	Optional (used with multi-source EMD only)	Identifies the source of the `extendedData`. Example values: `"MARKETING"`, `"PARTNERSHIP"`, `"SUPPLY_MANAGER"`, `"PPN"`.
`{attribute}`	string	Optional	EMD attribute name containing attribute values. If there is more than one business unit, the attribute format is `{attribute::buName}` using the delimiter to separate the source BU (for example, `attr1::BUYCO`). There can be up to 20 attributes, each separated by a comma.

Example CSV file:

```
identifierType,identifierValue,source,attr1::BUYCO,att2::BUYCO,att3,attr4,att5,attr6
loyaltyId,6105700000653433457,,attr1Value,,,,,,attr7Value
loyaltyId,6105700000653433457,PARTNERSHIP,,attr2Value,,,,,,
loyaltyId,6105700000653433457,INTERNAL,,,TestSeg123,TestSeg123,,,
externalIdentifier,b0543a9e-0283-4d8a-a100-0dcf98cfb506,XYZ,,,TestSeg1,TestSeg2,,,
loyaltyId,6105700000653433457,PPN,attr1_Updated_Value,,,,,,attr1_Updated_Value
loyaltyId,LoyaltyID456,MARKETING,,,,,,,Yes
loyaltyId,6105700000653433457,SUPPLY_MANAGER,,,attr3Value,,attr5Value,attr6Value,,
```

Reject feed (CSV format)

The following reject information can be accessed in the Console from the Select File menu (Select File > Reject File) after the CSV upload has the status of PROCESSED. If there are no rejected files, the Select File menu is not displayed.

Rejection message	Reason for rejection
`UNKNOWN_LOYALTY_ID`	Member extended data feed with invalid `loyaltyId`
`EXTENDED_DATA_REQUIRED`	Member extended data feed without any extended data
`LOYALTY_ID_REQUIRED`	Member extended data feed without any `loyaltyId`
Exceeded root level key limit for extended data: `{x}` keys	Limit on keys or attributes for extended data exceeded

---

Member household feed

Used to add or remove member households — groups of members that allow them to aggregate rewards for redemption.

There is one Primary member and additional members are designated as Secondary. The Primary member may view information about the household that may not be available to Secondary members (depending on settings). In addition, if the Primary member is removed from the household, the household is disbanded (deleted), but if a Secondary member is removed, the household continues.

Feed name and folder

- **Feed name:** `MEMBER_HOUSEHOLD`
- **Format:** NDJSON
- **Example filename:** `MEMBER_HOUSEHOLD_20230331_S1_V2.json`
- **SFTP folder:** `member/`

Frequency

This feed can be accepted every 15 minutes.

Feed contents

Field	Type	Required	Description
`partitionKey`	string	Optional	Unique identifier for the household
`primaryLoyaltyID`	string	Required (mutually exclusive with `primaryExternalIdentifier`)	Unique per card number but not validated for format, for the Primary member of the household
`primaryExternalIdentifier`	string	Required (mutually exclusive with `primaryLoyaltyID`)	The member that this event pertains to, using client identifier, for the Primary member of the household
`secondaryLoyaltyID`	string	Optional	Loyalty ID for the Secondary member (non-Primary member) to be added to the household
`secondaryExternalIdentifier`	string	Optional	External ID for the Secondary

Field	Type	Required	Description
			member to be added to the household
`action`	string	Required	Determines whether the action is to create or to remove a household (`ADD` or `REMOVE`)
`joinedDate`	string	Optional	Date in ISO 8601 format. Defaults to current date/time if not provided.

Examples of use cases

The identifiers for both Primary and Secondary members of the household are validated. Neither type of member may be associated with a ghost (unregistered) account.

- **Creating the household** — To create the household, only the `primaryLoyaltyID` is passed with the `ADD` action; the household is then created and the member becomes the Primary.
- **Adding a Secondary member** — If both the `primaryLoyaltyID` and the `secondaryLoyaltyID` are passed along with the `ADD` action, then the Secondary member is added to the household associated with the Primary member.
- **Removing a member from the household** — If both primary and secondary IDs are passed along with the `REMOVE` action, then only the Secondary member is removed and the member account is decremented by one. In this instance, the Primary member's household ID is checked against the Secondary member's household ID and rejected if not a match.
- **Disbanding the household** — If only the `primaryLoyaltyID` is passed along with the `REMOVE` action, then the household is disbanded/deleted.

The actions in this feed are processed in a set order:

1. `REMOVE` with `primaryIdentifier` (disbanding household)
2. `REMOVE` with `primaryIdentifier` and `secondaryIdentifier` (Secondary member leaving the household)
3. `ADD` with `primaryIdentifier` (creating household)
4. `ADD` with `primaryIdentifier` and `secondaryIdentifier` (adding Secondary member to household)

Implementation is based on grouping the `primaryIdentifier` actions for each of these operations.

Example NDJSON object — creating a household:

```
{
  "primaryLoyaltyID": "28282",
  "action": "ADD",
  "joinedDate": "2023-03-10T13:58:45-05:00"
}
```

Example NDJSON object — adding a Secondary member:

```
{
  "primaryLoyaltyID": "28282",
  "secondaryLoyaltyID": "96711",
  "action": "ADD",
  "joinedDate": "2023-03-10T13:58:45-05:00"
}
```

Example NDJSON object — removing a Secondary member from the household:

```
{
  "primaryLoyaltyID": "28282",
  "secondaryLoyaltyID": "96711",
  "action": "REMOVE"
}
```

Example NDJSON object — removing a Primary member and disbanding the household:

```
{
  "primaryLoyaltyID": "28282",
  "action": "REMOVE"
}
```

Reject feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

The possible reasons for a rejected record for this feed include:

Rejection message	Reason for rejection
Mandatory fields errors: `action`	Action is mandatory and required to be non-empty
Invalid action passed in request	Invalid action provided
Primary LoyaltyId / Primary ExternalIdentifier does not pass	Either Primary LoyaltyID or Primary ExternalIdentifier is required
Invalid primary loyaltyId	Primary `loyaltyId` `99999XXXXX` is not the default card for any account
Invalid Secondary loyaltyId	Secondary `loyaltyId` `99999XXXXX` is not the default card for any account
Invalid Primary ExternalIdentifier	Primary account not found
Invalid Secondary ExternalIdentifier	Secondary account not found
Invalid joined date passed	Invalid joined date
If primary member household already exists	Primary member is already part of a household

Rejection message	Reason for rejection
If member is already part of household	This account is already a member of a household
Secondary member add/delete but primary member household does not exist	Household does not exist
Unregistered primary account provided	Primary Identifier `xxx` does not exist or is not registered
Unregistered secondary account provided	Secondary Identifier `xxx` does not exist or is not registered

---

Membership tier feed

Refer to the Program tier feed.

Partner feed

The `PARTNER` feed is used for identifying the partner with which a member is associated.

Feed name and folder

- **Feed name:** `PARTNER`
- **Format:** NDJSON
- **Example filename:** `PARTNER_20220331_S1_V1.json`
- **SFTP folder:** `partner/<partner_id>`

Frequency

This feed can be accepted once per 15-minute period. Transactions for members without loyalty IDs can be sent with the special `loyaltyID` keyword `"ANONYMOUS"`.

Feed contents

Field	Type	Required	Description
`accountID`	string	Required	Unique per account
`cardType`	string	Required	The type of payment card involved
`loyaltyID`	string	Required (mutually exclusive with `externalIdentifier`)	Unique per card number but not validated for format
`externalIdentifier`	string	Required (mutually exclusive with `loyaltyID`)	The member that this event pertains to, using client identifier
`last4`	string	Required	The last four digits of the payment card number
`cardHolderType`	string	Required	Identifies the class of cardholder associated with this card
`mode`	string	Required	Partner card linking activity such as `"Add"` or `"Delete"`
`eventTimeStamp`	string	Required	The date/time when the transaction occurred
`partner`	string	Required	The name of the partner providing the payment card

Example JSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "accountID": "6701847578341",
```

```
  "cardType": "MYCREDIT",
  "loyaltyID": "28282",
  "last4": "0795",
  "cardHolderType": "Platinum",
  "mode": "Add",
  "eventTimeStamp": "2022-05-10T13:58:45-05:00",
  "partner": "BIGBANK"
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed. Note that if the account is closed, the error message reads: `"ACCOUNT_STATUS_IS_CLOSED."`

Partner links feed

The `PARTNER_LINK` feed is used for partner linking data to be imported so that bulk partner linking data for multiple members can be stored in the system.

Feed name and folder

- **Feed name:** `PARTNER_LINK`
- **Format:** NDJSON
- **Example filename:** `PARTNER_LINK_20230331_S1_V1.json`
- **SFTP folder:** `partner/<partner_id>`

Frequency

This feed can be accepted daily. It provides information regarding partner linking for members, including bulk partner linking data for multiple members.

Feed contents

Field	Type	Required	Description
`accountId`	string	Optional	Unique identifier for the account
`loyaltyId`	string	Optional	Unique identifier for the loyalty card
`identifierType`	string	Required	Enumeration for the type of identifier to be used for looking up an account. Allowed values: `"ACCOUNT_ID"`, `"LOYALTY_ID"`, `"EXTERNAL_IDENTIFIER"`, `"EMAIL"` (possibly supported in future). Default: `"ACCOUNT_ID"`.
`identifierValue`	string	Required	Value of the member identifier
`linkedCardType`	string	Required	Defines the partner card type. Allowable values are defined per client by Exchange Solutions working with the client. There may be one or more partner card types (for example: `"BANKCO"`, `"SHOPCO"`, `"GASCO"`).
`linkId`	string	Optional	Full identifier for the linked partner card. Either `"linkId"` or `"last4"` must be set in the feed file (which one is used depends on `PARTNER_CONFIG`).
`bin`	string	Optional	Bank Identification Number (BIN) of the partner card
`last4`	string	Optional	Last four digits of the partner card number. Either `"linkId"` or `"last4"` must be set in the feed file (which one is used depends on `PARTNER_CONFIG`).
`mode`	string	Required	Identifier for the operation to be performed for this linking record. Allowed values: `"ADD"`, `"REMOVE"`, `"UPDATE"`, `"NOOP"`.
`metadata`	object	Optional	Container to pass optional metadata related to this partner card link. Must respect the limits configured

Field	Type	Required	Description
			in `EXTENDED_DATA Config`. These are pass-through attributes and are only persisted with no processing.
`eventTimestamp`	string	Optional	Timestamp of the linking event in the default timezone configured for the system. Also used to sort records for each operation and process in sequence. Expected format: `YYYYMMDDHHmmss`.
`timeStampJoined`	string	Optional	Timestamp when the partner card is linked to the loyalty account in the default timezone configured for the system. Expected format: `YYYYMMDDHHmmss`.

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "identifierType": "ACCOUNT_ID",
  "identifierValue": "7b402123-88a5-4d19-a7fe-6d1956a32773",
  "linkedCardType": "BANKCO",
  "linkId": "123456Token1234",
  "bin": "123456",
  "last4": "1122",
  "mode": "ADD"
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed. Note that if the account is closed, the error message reads: `"ACCOUNT_STATUS_IS_CLOSED."`

Product feed

The `PRODUCT` feed contains information about products (which need to be defined to allow loyalty offers to be targeted against them), their categorization (for category, brand, or department offers),

and their price and cost (for offer economics).

Feed name and folder

- **Feed name:** `PRODUCT`
- **Format:** NDJSON
- **Example filename:** `PRODUCT_20230331_S1_V1.json`
- **SFTP folder:** `businessUnit/<businessUnitName>/product`

Frequency

This feed can be accepted once per day.

Feed contents

Field	Type	Required	Description
`productCode`	string	Required	Could be a string of numbers, an alphanumeric code, or another unique identifier
`productCodeType`	string	Required	The type of standard identifier used to identify products: `"GTIN"`, `"UPC"`, or `"PRODUCT NUMBER"`
`productName`	string	Required	The name of the product
`productStatus`	string	Optional (recommended)	Whether the product is in stock: `"IN_STOCK"` or `"OUT_OF_STOCK"`
`categoryCode`	string	Required	The code for the top-level product category
`categoryName`	string	Required	The human-readable category name

Field	Type	Required	Description
`subcategoryCode`	string	Required	The code for the product subcategory
`subcategoryName`	string	Required	The human-readable subcategory name
`productDescription`	string	Optional (recommended)	Description of the product
`itemCost`	number	Optional (strongly recommended)	Cost of item expressed as a number in pennies (for example, $34.23 = 3423)
`itemRetailPrice`	number	Optional (strongly recommended)	Price of the item expressed as a number in pennies
`brandCode`	string	Optional for operations, required for analytics	The brand code for the product
`brandName`	string	Optional for operations, required for analytics	The human-readable name for the product
`departmentCode`	string	Optional for operations, required for analytics	The code for this product's department
`departmentName`	string	Optional for operations,	The human-readable department name

Field	Type	Required	Description
		required for analytics	
`groupCode`	string	Optional for operations, required for analytics	The code for this product's grouping
`groupName`	string	Optional for operations, required for analytics	The human-readable group name
`reportingIdentifierCode`	string	Optional for operations, required for analytics	The code for the unique identifier used for reporting
`reportingIdentifierName`	string	Optional for operations, required for analytics	The name for the unique identifier used for reporting
`vendorCode`	string	Optional for operations, required for analytics	The code for this product's associated vendor
`vendorName`	string	Optional for operations, required for analytics	The human-readable vendor name
`activeDate`	string	Optional for operations,	The date this product became available for sale in the catalog

Field	Type	Required	Description
		required for analytics	
`lastUpdatedDate`	string	Optional	The last time this product information was updated
`alternativeProductCode`	string	Optional	To provide alternative product codes if needed. An object that can contain multiple aliases.
`extendedData`	object	Optional	Additional data about the product that the client may want to have in reports
`secondaryAssociations`	array	Optional	An array to provide additional associations for the product. A maximum of 20 secondary associations may be provided. Attributes within `secondaryAssociations` include: `category`, `subCategory`, `department`, `group`, `brand`, and `reportingIdentifier`, each of which are name/code pairs. Duplicates are ignored.
`businessUnit`	string	Optional	The business unit associated with the product
`productCodeAliases`	object	Optional	Key-value pairs that provide a `productCode`, such as `"GTIN"`, and the associated value

Product extended data

Extended data can be provided; however, it will not be acted upon without product customization. Any key-value is allowable, but the maximum data size for extended data for a given product is five kilobytes, consisting of up to 20 unique root-level keys across all members.

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):


```
{
  "productCode": "2323333",
  "productCodeType": "UPC",
  "productName": "Wilson Raptors Game Size BasketBall",
  "productDescription": "Official Size Raptors BasketBall",
  "productStatus": "IN_STOCK",
  "itemCost": 1432,
  "itemRetailPrice": 4000,
  "brandCode": "39239",
  "brandName": "Wilson",
  "categoryCode": "34242",
  "categoryName": "Team Sports",
  "subcategoryCode": "23932",
  "subcategoryName": "Basketball",
  "departmentCode": "23",
  "departmentName": "Sporting Goods",
  "groupCode": "23442",
  "groupName": "NBA Licensed Goods",
  "reportingIdentifierCode": "SPORTS",
  "reportingIdentifierName": "Sports",
  "vendorCode": "34242",
  "vendorName": "Wilson",
  "activeDate": "2020-01-01",
  "lastUpdatedDate": "2023-05-10",
  "productCodeAliases": {
    "UPCE": "2323333",
    "GTIN": "ASDFA332"
  },
  "extendedData": {},
  "secondaryAssociations": [
    {
      "category": {
        "name": "Sporting Goods",
        "code": "SPORTING_GOODS"
      }
    },
    {
```

```
      "subCategory": {
        "name": "Team Sports",
        "code": "TEAM_SPORTS"
      }
    },
    {
      "department": {
        "name": "Official Merchandise",
        "code": "OFFICIAL_MERCHANDISE"
      }
    },
    {
      "group": {
        "name": "All Ages",
        "code": "ALL_AGES"
      }
    },
    {
      "brand": {
        "name": "Official Sub-Licensing Co.",
        "code": "OFFICIAL_SUB_LIC"
      }
    },
    {
      "reportingIdentifier": {
        "name": "Sports Gear",
        "code": "SPORTS_GEAR"
      }
    }
  ]
}
```

Reject feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

The possible reasons for a rejected record for this feed include:

Reason for rejection	Rejection message
Missing `productCode`	Product Code not found
Missing `productCodeType`	Product Code Type not found
Missing `productName`	Product Name not found
Missing `categoryCode`	Category Code not found
Missing `categoryName`	Category Name not found
Missing `subcategoryCode`	SubCategory Code not found
Missing `subcategoryName`	SubCategory Name not found
Missing `productStatus`	Product Status not found
Size of `extendedData` more than configured max value	Exceeded data size limit for extended data: `${dataSize}` Bytes
Number of keys in `extendedData` more than configured max value	Exceeded root level key limit; allowed is up to `${numberOfKeys}`
Number of `secondaryAssociations` more than configured max value	Secondary Associations size limit; allowed is up to `${maxSecondaryAssociationSize}`
Missing `secondaryAssociations[:].code`	Code is required
Invalid format of `secondaryAssociations[:].code`	Code length exceeded or invalid characters

---

Program tier feed

The `PROGRAM_TIER` or `MEMBERSHIP` feed is used for identifying the tier with which a member is associated.

> **Note:** The Program Tier feed is used by most current and new clients. The Membership Tier feed is used by some legacy clients.

Feed name and folder

- **Feed name:** `PROGRAM_TIER` or `MEMBERSHIP`
- **Format:** NDJSON
- **Example filenames:** `PROGRAM_TIER_20250331_S1_V1.json` or `MEMBERSHIP_20250331_S1_V1.json`
- **SFTP folder:** `membershipTier/` or `programTier/`

Frequency

This feed can be accepted once per 15-minute period. Transactions for members without loyalty IDs can be sent with the special `loyaltyID` keyword `"ANONYMOUS"`.

Feed contents

Field	Type	Required	Description
`LoyaltyId`	string	Required (primary key, more than one identifier may be used)	Unique per card number but not validated for format

Field	Type	Required	Description
`ExternalIdentifier`	string	Required (primary key, more than one identifier may be used)	The member that this event pertains to, using client identifier
`AccountId`	string	Optional	Unique per account
`ProgramCode`	string	Required	Tier program setting (for example, `"MOST_VALUABLE_CUSTOMER"`)
`TierCode`	string	Required	Designation for member tier rank (for example, `"TIER1"`)
`BenefitStartDate`	number	Optional	The date epoch when the tier benefits commence(d). Note: Whether this field is required depends on configuration settings. If configuration is set not to allow updates to past benefit periods (the default), this field cannot be included in the feed. If configuration does allow updating past periods and this field has no value, the value is set to the beginning of the current period. The value must align with the program start date and cadence.
`OverrideCurrent`	boolean	Optional	Whether the current tier associated with the member should be overridden by the `tierCode` (`true` or `false`)
`EligibleContribution`	number	Optional	Used to help determine when a member should be awarded membership in the next program tier level, based on the eligible

Field	Type	Required	Description
			spend in dollars or units in litres. Note: If this field and value are provided in the feed but are not within the range of values for the specified `TierCode`, an error is returned. If this field is not included, the value corresponding to the lower range for the `TierCode` is returned.

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "LoyaltyId": "245246686469",
  "AccountId": "6701847578341",
  "ProgramCode": "MOST_VALUABLE_CUSTOMER",
  "TierCode": "TIER1",
  "BenefitStartDate": "1651436407000",
  "OverrideCurrent": false,
  "EligibleContribution": 500
}
```

Differences in handling between household and individual member tiers

Household member tiers are handled a bit differently than individual member tiers when updating data with this feed:

- If it's an individual user who doesn't have a `MEMBERSHIP_TIER` object, it should also be initialized.
- The feed cannot be used to update individual tier objects if the member is part of a household.
- If the member is the Primary member of a household, update the household tier objects. The individual tier object of the Primary member remains unaffected.
- If the member is a Secondary member of a household, reject the request. Only the Primary member can be used to update household tier objects.
- If the member is not part of a household, manual tier updates should only update the member's individual tier objects.
- If any of a household member's `MEMBERSHIP_TIERS/ASSESSMENT` objects don't exist while processing the feed and the Primary member does an update to the Household object, those

users' objects should also be created.

How household changes affect tier status

A household's aggregated balance determines tier status, so anything that affects the balances of members within the household potentially affects the household tier status. This includes purchases, returns/adjustments, and members joining or leaving the household. If tier status is set to roll over from one period to the next, then balance changes can affect tier status in the current period and potentially (in the event of adjustments to a member's balance in the last period) tier status in both the current and the last period.

In terms of members joining or leaving the household when rollover is enabled:

1. When the household is created, set the `AchievedDate` to the `achievedDate` of the individual `membershipTier` for the Primary members, as they are the lone member in the household at that time.
2. Whenever any member joins the household and it causes the current household tier status to change, update the `AchievedDate` to `benefitStartDate` if `tierStatus` is due to the last period spend (placement). Otherwise, set the `AchievedDate` to `currentDate`.
3. Whenever any member leaves the household and it causes the current household tier status to change, update the `AchievedDate` to `benefitStartDate` if `tierStatus` is due to the last period spend (placement). Otherwise, set the `AchievedDate` to `currentDate`.
4. After any type of transaction, if there is a change in the `tierStatus` of the `membershipTier/householdTierStatus` object, update the `AchievedDate` to `benefitStartDate` if `tierStatus` is due to last period spend (placement). Otherwise, set the `AchievedDate` to `currentDate`.

If rollover is disabled, as with individual members, then the next tier is set to zero. The household begins to accumulate contributing spend again at the beginning of the next period.

Reject feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

The possible reasons for a rejected record for this feed include:

Rejection message	Reason for rejection
`DOWNGRADING_TIER_RANK`	Feed uploads rank that downgrades tier or changes tier to unranked for the same account with `override false`
`INVALID_ACCOUNT_ID`	Membership tier with invalid account ID
`UNKNOWN_LOYALTYID`	Membership tier with invalid loyalty ID
`INVALID_PROGRAM_CODE`	Membership tier with invalid program code
`INVALID_TIER_CODE`	Membership tier with invalid tier code
`INVALID_BENEFIT_START_DATE`	Membership tier with invalid benefit start date. For example, `BenefitStartDate` is not allowed in feed requests when `AllowPastPeriodUpdated` is disabled.
`ACCOUNT_ID_OR_LOYALTY_ID_REQUIRED`	Membership tier without account ID or loyalty ID
`PROGRAM_CODE_REQUIRED`	Membership tier with empty program code
`TIER_CODE_REQUIRED`	Membership tier with empty tier code
`PROGRAM_DISABLED`	Membership tier with disabled program
`SECONDARY_HOUSEHOLD_MEMBER_NOT_ALLOWED`	If there is a household, a Secondary household member cannot initiate a redemption
`PARSE_ERROR`	The code is unable to understand or interpret a script
`INVALID_ELIGIBLE_CONTRIBUTION`	The value for `EligibleContribution` is outside the range of allowed values for the designated `TierCode`

Redemption feed

The `REDEMPTION` feed is used to provide information about rewards redemptions.

Feed name and folder

- **Feed name:** `REDEMPTION`
- **Format:** NDJSON
- **Example filename:** `REDEMPTION_20230331_S1_V1.json`
- **SFTP folder:** `redemption/`

Frequency

This feed can be accepted once per 15-minute period. Transactions for members without loyalty IDs can be sent with the special `loyaltyID` keyword `"ANONYMOUS"`.

Feed contents

Field	Type	Required	Description
`LoyaltyID`	string	Required (mutually exclusive with `ExternalIdentifier`)	Unique per card number but not validated for format
`ExternalIdentifier`	string	Required (mutually exclusive with `LoyaltyID`)	The member that this event pertains to, using client identifier
`RedeemableBalance`	number	Required	The quantity of rewards available for redemption
`ReasonCode`	string	Required	Identifies the reasons for the redemption
`Tender`	object	Optional	Method of payment (object containing value pairs such as cash (`"CASH"`), credit card name

Field	Type	Required	Description
			(`"VISA"`, `"MC"`, `"AMEX"`), and/or reward redemption (`"LOYALTY"`, `"GIFTCARD"`), each with a corresponding value). If tender is not used, then the default tender is the `redeemableBalance`.
`BusinessUnit`	string	Optional	The unit of the overall business associated with the redemption transaction

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "loyaltyID": "72631872631237",
  "ExternalIdentifier": "879182739871938",
  "RedeemableBalance": "25000",
  "ReasonCode": "OFFER",
  "Tender": {
    "LOYALTY": 1000
  },
  "BusinessUnit": "BUYCO"
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Store location feed

The `STORE` feed provides information about the various store locations which can be used to target the applicability of specific offers. There are two versions of the `STORE` feed (v1 and v2) to accommodate different client needs.

Feed name and folder

- **Feed name:** `STORE`
- **Format:** NDJSON
- **Example filename:** `STORE_20230331_S1_V1.json`
- **SFTP folder:** `businessUnit/<businessUnitName>/store/`

Frequency

This feed can be accepted once per day.

Feed contents

STORE feed v1:

Field	Type	Required	Description
`storeName`	string	Required	Human-readable name of the store
`countryCode`	string	Required	Country code
`province`	string	Optional	Province code
`retailBanner`	string	Required	If the retailer has different store banners or divisions (for example, `"Outlet"`)
`storeCode`	string	Required	Unique identifier for the store
`extendedData`	object	Optional	Additional data about the store that the client may want to have in reports, consisting of key-value pairs within the object
`businessUnit`	string	Optional	The unit of the overall business associated with the store location

Extended data can be provided; however, it will not be acted upon without product customization. Any key-value is allowable, but the maximum data size for extended data for a given location is five kilobytes, consisting of up to 20 unique root-level keys across all members.

Example NDJSON object — STORE feed v1 (formatted for readability only — one object per line followed by CRLF):

```
{
  "storeName": "ORLEANS GARDEN OTTAWA",
  "countryCode": "CA",
  "province": "ON",
  "retailBanner": "OutletStore",
  "storeCode": "0095",
  "extendedData": {},
  "businessUnit": "BUYCO"
}
```

STORE feed v2:

Field	Type	Required	Description
`storeName`	string	Required	Human-readable name of the store
`countryCode`	string	Required	Country code
`province`	—	Deprecated	Replaced by `provinceState`
`provinceState`	string	Required	Province or state code. The value is validated based on the `addressType` (CA or US) used in configuration.
`retailBanner`	string	Required	If the retailer has different store banners (for example, `"Outlet"`)
`storeCode`	string	Required	The store code
`extendedData`	object	Optional	Additional data about the store that the client may want to have in reports
`postalZipCode`	string	Optional	The postal or Zip code from the store's address

Field	Type	Required	Description
`businessUnit`	string	Optional	The unit of the overall business associated with the store location

Extended data can be provided; however, it will not be acted upon without product customization. Any key-value is allowable, but the maximum data size for extended data for a given location is five kilobytes, consisting of up to 20 unique root-level keys across all members.

Example NDJSON object — STORE feed v2 (formatted for readability only — one object per line followed by CRLF):

```
{
  "storeName": "MONTPELIER SUPERSTORE",
  "countryCode": "US",
  "provinceState": "VT",
  "postalZipCode": "05401",
  "retailBanner": "OutletStore",
  "storeCode": "0095",
  "businessUnit": "BUYCO",
  "extendedData": {
    "Store Size": "Large",
    "Legal Entity": "MSS",
    "Address": "285 Church Street",
    "City": "Burlington",
    "Store Open Date": "2015-01-01",
    "Store Close Date": ""
  }
}
```

Reject feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

The possible reasons for a rejected record for this feed include:

Reason for rejection	Rejection message
Missing `storeName`	Store Name not found
Missing `countryCode`	Country Code Type not found
Missing `retailBanner`	Retail Banner not found
Missing `StoreCode`	Store Code not found
Missing `provinceState`	Province/State not found
Invalid `provinceState` value (for given `countryCode` or default `countryCode=CA`)	Invalid Province/State
Size of `extendedData` more than configured max value	Exceeded data size limit for extended data: `${dataSize}` Bytes
Number of keys in `extendedData` more than configured max value	Exceeded root level key limit for extended data: `${numberOfKeys}` keys

---

Transaction (purchase) feed

The `TRANSACTION` feed (the retail purchase feed) receives information from the point-of-sale and/or eCommerce solution containing the purchase and cart information of each retail transaction. It handles both purchase and adjustment records.

Feed name and folder

- **Feed name:** `TRANSACTION`
- **Format:** NDJSON
- **Example filename:** `TRANSACTION_20220331_S1_V1.json`
- **SFTP folder:** `transaction/`

Frequency

This feed can be accepted once per 15-minute period. Any external member IDs referenced in this file must have arrived previously in a `MEMBER` feed, or the record will be rejected.

Feed contents

If a `TRANSACTION` record is provided that doesn't exist in ES Loyalty, that record is rejected (and you are notified in the `REJECT` feed).

Purchases

Field	Type	Required	Descript
`loyaltyID`	string	Required (mutually exclusive with `externalIdentifier`)	Unique per c number but validated for format
`externalIdentifier`	string	Required (mutually exclusive with `loyaltyID`)	The membe this event pe to, using clie identifier
`action`	string	Optional	Type of tran `"PURCHASE"` `"SAF"` (to complete transaction), `"ADJUSTMEN` return or pa return), or `"ADJ_NO_RE` (an adjustme without a re that doesn't reference th original transaction)

Field	Type	Required	Descript
`channel`	string	Required	Channel thro which the transaction occurred: `"S` or `"ECOMMER`
`correlationID`	string	Required	The transact associated w current trans (your format be unique)
`transactionDate`	string	Required	A datetime v the transacti took place (I 8601 format
`retailBanner`	string	Required	A store grou or type
`priceMatrix`	string	Required	Code identif the proper s prices to app
`storeNumber`	string	Required	Referencing store/ecomm site from the location feed
`tillNumber`	string	Required	Referencing specific till in store
`employeeCode`	string	Optional (strongly recommended for	Referencing employee w

Field	Type	Required	Descript
		analytics)	handled the transaction
`cart`	object	Required	A JSON obje containing t
`cart.saleLineItems`	array	Required	A JSON arra individual sa items
`cart.saleLineItems[].subCategory`	string	Optional (can be null)	Product subcategory
`cart.saleLineItems[].sku`	string	Required	The product identifier
`cart.saleLineItems[].quantity`	string	Required	The number SKUs of this the basket
`cart.saleLineItems[].originalSaleAmount`	string	Required	Undiscounte amount in c
`cart.saleLineItems[].saleAmount`	string	Required	The amount the discount cents
`cart.saleLineItems[].storeCoupon`	string	Optional (can be null)	Coupon pro in the store
`cart.saleLineItems[].mnfCoupon`	string	Optional (can be null)	Manufacture coupon in c
`cart.saleLineItems[].promoCoupon`	string	Optional (can be null)	Promotional coupon in c

Field	Type	Required	Descript
`cart.saleLineItems[].itemDiscount`	string	Required	Discounts ap in cents
`cart.saleLineItems[].itemTax`	string	Required	Applicable s
`cart.saleLineItems[].itemCost`	string	Required	Cost of item
`cart.saleLineItems[].finalSaleAmount`	string	Required	Final discou amount with on the item
`cart.saleLineItems[].tags`	array	Optional	An array cor extended da about the lir
`cart.totalSaleAmount`	string	Required	The total sal amount of a excluding ta
`currency`	string	Optional	The currency transaction i (for example `"CAD"`)
`tender`	object	Optional	Object conta value pairs s cash (`"CASH` credit card r (`"VISA"`, `"MASTERCAR` `"AMEX"`), an reward rede (`"LOYALTY"` `"CARD"`). Ea key can have

Field	Type	Required	Descript
			a cent value more detaile tender objec array of tend objects.
`tender.LOYALTY`	number	Required	Reserved ke denotes in c the dollars s by exchangi loyalty point
`businessUnit`	string	Optional	The unit of t overall busir associated w purchase transaction
`externalTransactionID`	string	Optional	Used to carr client transa identificatio within the p data
`extendedData`	object	Optional	Contains ke pairs that all client to add own data int feed

Tender object attributes:

Field	Type	Description
`idType`	enum	Type of ID used: `"TOKEN"` (default), `"LAST4"`, or `"PREFIX_LAST4"`
`id`	string	ID used to identify the card (for example, `"1234"`, `"123456TOKEN1234"`). TOKEN IDs are expected to be unique hashed or tokenized payment card numbers. ESI is not responsible for clear-text data passed into this field. Required for TOKEN.
`prefix`	string	BIN of the card involved (for example, `"123456"`). Required for `PREFIX_SUFFIX`.
`suffix`	string	Last four digits (including the check digit) of the card (for example, `"1234"`). Required for `PREFIX_SUFFIX`.
`amount`	number	Tender amount in the required currency format. Depending on system configuration, this may be in dollars or cents.

Example purchase NDJSON object (formatted for readability only — one object per line followed by CRLF):


```
{
  "action": "SAF",
  "channel": "STORE",
  "correlationID": "29Ee-399RD-293-2",
  "externalTransactionID": "f32-934rt-28eud77-23wer77",
  "transactionDate": "2022-03-10T13:58:45-05:00",
  "retailBanner": "OUTLET",
  "priceMatrix": "R",
  "storeNumber": "2378",
  "tillNumber": "04",
  "employeeCode": "11120",
  "loyaltyID": "23442344432",
  "cart": {
    "saleLineItems": [
      {
        "subCategory": "44100",
        "sku": "73385499157",
        "quantity": "1",
```

```
        "originalSaleAmount": "4500",
        "saleAmount": "4500",
        "mnfCoupon": "",
        "promoCoupon": "",
        "itemDiscount": "0",
        "itemTax": 0,
        "itemCost": "3300",
        "finalSaleAmount": "2000",
        "tags": []
      }
    ],
    "totalSaleAmount": "2260",
    "currency": "CAD"
  },
  "tender": {
    "CASH": "1000",
    "VISA": {
      "idType": "PREFIX_LAST4",
      "prefix": "123456",
      "suffix": "3312",
      "amount": "500"
    },
    "MASTERCARD": [
      {
        "idType": "TOKEN",
        "id": "123456Token1234",
        "amount": "500"
      },
      {
        "idType": "TOKEN",
        "id": "123123Token1231",
        "amount": "160"
      }
    ],
    "LOYALTY": "0"
  },
  "businessUnit": "SELLCO",
  "externalTransactionID": "ry6537-rwry9959",
  "extendedData": {
    "airMilesSwipe": "0",
    "salesPoolID": "Contractor",
    "program_1": "value_45",
    "program_2": "value9000"
```

```
  }
}
```

Adjustments

For Adjustments, the remaining keys (other than the `"action"` key) are:

Field	Type	Required	Description
`loyaltyID`	string	Required (mutually exclusive with `externalIdentifier`)	Unique per card number but not validated for format
`externalIdentifier`	string	Required (mutually exclusive with `loyaltyID`)	The member that this event pertains to, using client identifier
`originalCorrelationID`	string	Required	The unique correlation/transaction ID of the previous transaction this transaction is adjusting
`currentCorrelationID`	string	Required	The unique correlation/transaction ID of this adjustment
`externalTransactionID`	string	Optional	Used to carry a client transaction identification code within the purchase data
`transactionDate`	string	Required	A datetime when the transaction took place (ISO 8601 format)

Field	Type	Required	Description
`priceMatrix`	string	Required	Code identifying the proper set of prices to apply
`channel`	string	Required	Channel through which the original transaction was made (`"STORE"` or `"ECOMMERCE"`)
`retailBanner`	string	Required	The banner of the store (for example, `"OUTLET"`)
`storeNumber`	string	Required	The code of the store
`tillNumber`	number	Required	An identifier for a specific till
`employeeCode`	number	Required	A code identifying the employee at the till
`reversalLineItems`	array	Required	—
`reversalLineItems[].sku`	number	Required	The SKU being reversed
`reversalLineItems[].quantity`	number	Required	The number of SKU items being reversed
`reversalLineItems[].saleAmount`	number	Optional	Should be included only in cases where there could potentially be purchases of the same SKU at different

Field	Type	Required	Description
			prices or by different/additional measures (for example, rope sold by the spool and also by the foot). This ensures that ES Loyalty can discriminate between instances of the same SKU. Value is in cents.
`tender`	object	Optional	Method of payment
`extendedData`	object	Optional	Contains key-value pairs that allow the client to add their own data into the feed

> **Note:** Exchanges of different items are expected to come as an adjustment removing the exchanged item, followed by a purchase transaction of the new items. If an exchange consists of an identical cart (for example, different sizes), a purchase can be sent with `$0` items to net 0 points.

Example adjustment NDJSON object (formatted for readability only — one object per line followed by CRLF):


```
{
  "action": "ADJUSTMENT",
  "originalCorrelationID": "2342334",
  "currentCorrelationID": "2344442433",
  "externalTransactionID": "10000001",
  "externalIdentifier": "983ddb80-2a87-41b6-8875-0c040f55e6a7",
  "transactionDate": "2024-03-10T13:58:45-05:00",
  "priceMatrix": "R",
  "channel": "STORE",
  "retailBanner": "OUTLET",
  "storeNumber": "0034",
```

```
  "tillNumber": "04",
  "employeeCode": "11120",
  "reversalLineItems": [
    {
      "sku": "73385499157",
      "quantity": 1,
      "saleAmount": 1999
    },
    {
      "sku": "73385499157",
      "quantity": 0.1,
      "saleAmount": 500
    }
  ],
  "tender": {
    "VISA": "2260",
    "LOYALTY": "0"
  },
  "extendedData": {
    "airMilesSwipe": "0",
    "salesPoolID": "Contractor",
    "abccvv": "value_45",
    "abcnnvv": "value9000"
  }
}
```


Return feeds

The following outbound feeds are generated as a response if any records apply:

- **REJECT** — An outbound Schedule Report Reject Feed including a reason code is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.
- **REWARD** — If there are rewards associated with the completion of an activity, details of the rewards are captured and sent back to the client using an outbound REWARD feed. For more information, see Schedule Report Reward Feed in the Outbound feeds section.

Vendor feed

The `VENDOR` feed lists third-party vendors that supply products to the business and may participate in the loyalty program. This feed is used for reporting purposes only.

Feed name and folder

- **Feed name:** `VENDOR`
- **Format:** NDJSON
- **Example filename:** `VENDOR_20230331_S1_V1.json`
- **SFTP folder:** `vendor/`

Frequency

This feed can be accepted once per day.

Feed contents

Field	Type	Required	Description
`VendorId`	—	Required	Vendor ID
`VendorName`	—	Required	Human-readable vendor name
`BusinessUnit`	string	Required (if business units are activated)	The unit of the overall business associated with the vendor

Example NDJSON object (formatted for readability only — one object per line followed by CRLF):

```
{
  "VendorId": "BUYCO",
  "VendorName": "VENDOR BUYING COMPANY LTD",
  "BusinessUnit": "HOUSE OF GARDENS"
}
```

Return feed

REJECT — An outbound Schedule Report Reject Feed including reason codes is sent back to the client when there are rejected files resulting from operations with an inbound feed. For more information, see Schedule Report Reject Feed.

Outbound feeds

Schedule report reject feed

The `REJECT` feed provides information about any records that were rejected for any other feed.

Feed name and path

- **Feed name:** `REJECT_{original file name}`
- **Format:** NDJSON
- **Example filenames:** `REJECT_TRANSACTION_20230331_S1_V1.json` or `REJECT_ACTIVITY_20230331_S1_V1.json`
- **Feed path:** `businessUnit/<client name>/reject/REJECT_<original file name>`

Frequency

This feed is typically generated after the receipt of each inbound feed if there are errors.

Feed contents

The `REJECT` feed is an outbound feed that provides offer information.

For inbound feeds in which some processed files fail, a header is included for each rejected file with:

Field	Type	Required	Description
`fileName`	string	Required	The filename of the file that was processed
`success`	number	Required	The number of successfully processed records in the file
`unprocessed`	number	Required	The number of unprocessed records in the filename

Information about each row that failed from the original file is also provided, optionally including the original request and the failed record name:

Field	Type	Required	Description
`originalRequest`	object	Optional	JSON object to pull in the original request made in the inbound feed
`failedRecord`	string	Optional	Identifier of the specific record that was erroneous
`eslProcessing`	object	Required	Information about why the row was rejected
`eslProcessing.status`	string	Required	Always `REJECTED`
`eslProcessing.fileOrigin`	string	Required	The file this row came from
`eslProcessing.reasonCode`	number	Required	Code for the reason the row failed
`eslProcessing.reasonMessage`	string	Required	A more human-friendly message about the failure

For files on inbound feeds that fail completely, the following is returned:

Field	Type	Required	Description
`fileProcessed`	string	Required	The file ES Loyalty attempted to process
`processFailure`	object	Required	A JSON object containing the reason for the file failure

Partially-processed file example (expanded for readability):

```
{
  "fileProcessed": "TRANSACTION_20220331_S1_V1.json",
  "success": 1,
  "unprocessed": 2
}
{
  "recordId": "id1",
```

```
  "attribute1": "attributeValue",
  "attribute2": "attributeValue2",
  "eslProcessing": {
    "status": "REJECTED",
    "fileOrigin": "TRANSACTION_20220331_S1_V1.json",
    "reasonCode": "FAILED",
    "reasonMessage": "Some reason it failed to process"
  }
}
{
  "recordId": "id2",
  "attribute1": "attributeValue",
  "attribute2": "attributeValue2",
  "eslProcessing": {
    "status": "REJECTED",
    "fileOrigin": "TRANSACTION_20220331_S1_V1.json",
    "reasonCode": "FAILED",
    "reasonMessage": "Some reason it failed to process"
  }
}
{
  "eslProcessing": {
    "status": "REJECTED",
    "fileOrigin": "TRANSACTION_20220331_S1_V1.json",
    "reasonCode": "FAILED",
    "reasonMessage": "Some reason it failed to process"
  },
  "failedRecord": "\"recordId\":\"id2\",
\"attribute1\":\"attributeValue\",\"attribute2\":\"attributeValue2\""
}
```

Full file failure example:

```
{
  "fileProcessed": "TRANSACTION_20220331_S1_V1.json",
  "processFailure": {
    "reasonCode": "MALFORMED_REQUEST",
    "reasonMessage": "Original file could not be read due to formatting failure.",
    "trace": "expected ' at line 89"
  }
}
```

*Last updated on **Oct 14, 2018***

(Simulated during dev for better perf)