API Workflow: User Points Management

Table of Contents

Summary

This workflow features the UserPoints API, which is part of the Incentives domain. It details the primary transactions that can be performed for tasks related to managing a user's point balances.

Use Case Sequence

The sequence for this use case is:

1. Fetch customer's point balance:
    a. From default account.
    b. From specified point account(s).
    c. From all of their existing accounts.
2. Increase or decrease the point balance from the account(s).

Data Configuration Prerequisites

This flow assumes that at least one point source and point account have been configured within the SMP for a given retailer.

Step by Step with Examples

Featured in this workflow is the UserPoints API. It allows you to fetch, decrease (spend), or increase (deposit) a customer's point balance.

When issuing curl commands for these transactions, adhere to the following syntax:

- Begin each curl command with either `POST`.
- Specify: `-H 'Content-Type: application/json' -H 'authorization: Basic AUTH_ID'`
- Begin URL with same endpoint: `https://[ENDPOINT]/api/1.0/user_points`

Step 1: Retrieve Available Balance for Customer Point Accounts

The first step in managing a customer's point balance is to retrieve it. You can do this using one of the two endpoints available in the API.

Fetching a Balance for Specified or Default Point Accounts

This endpoint allows you to specify fetch a balance from one or more point accounts. If your loyalty program requires only a single point account, then the API returns the current balance and lifetime historical balance for a given point account.

```
POST /api/1.0/user_points/balance
```

The endpoint passes in a request object that identifies which point account to use.

Request

```
{
  "retailer_id": "string",
  "user_id": "string",
  "point_account_ids": [
    "00000000-0000-0000-0000-000000000000"
  ]
}
```

The request object can contain one or multiple point account IDs. The key attribute in this object is *point_account_ids*. If you don't specify any IDs, the operation fetches the balance from the customer's default point account.

The platform returns a response object:

Response

```
{
  "response_payload": {
    "retailer_id": "string",
    "user_id": "string",
    "summary": {
      "total_points": 0,
      "life_time_points": 0
    },
    "details": [
      {
        "account_name": "string",
```

```
        "user_point_account_id": "string",
        "point_account_id": "string",
        "grouping_label": "string",
        "available_balance": 0,
        "life_time_value": 0
      }
    ]
  },
  "status": 0,
  "error_response": {
    "code": "string",
    "message": "string",
    "raw_message": "string",
    "stack_trace": "string"
  },
  "logs": [
    {
      "log_level": "Trace",
      "message": "string",
      "stack_trace": "string",
      "time_occurred": "2018-12-03T15:21:11.371Z",
      "duration": 0,
      "scope_observer_id": "00000000-0000-0000-0000-000000000000",
      "entity_id": "00000000-0000-0000-0000-000000000000"
    }
  ],
  "message": "string",
  "scope": {
    "retailer_id": "string",
    "event": "string",
    "extended_scope": {}
  },
  "request_payload": {},
  "authorized_integrator_id": "00000000-0000-0000-0000-000000000000",
  "started": "2018-12-03T15:21:11.371Z",
  "completed": "2018-12-03T15:21:11.371Z"
}
```

Fetching a Balance for All Point Accounts

Alternatively, you can use a different endpoint to fetch a balance that reflects all of the customer's point accounts. If your loyalty program requires multiple point accounts, then the API returns the current balance and lifetime historical balance for all active point accounts.

```
POST /api/1.0/user_points/all_balances
```

This endpoint passes in a request object that contains no point account identifiers.

Request

```
{
  "retailer_id": "string",
  "user_id": "string"
}
```

The request object contains only IDs for the retailer and the customer.

The platform returns a response object:

Response

```
{
  "response_payload": {
    "retailer_id": "string",
    "user_id": "string",
    "summary": {
      "total_points": 0,
      "life_time_points": 0
    },
    "details": [
      {
        "account_name": "string",
        "user_point_account_id": "string",
        "point_account_id": "string",
        "grouping_label": "string",
        "available_balance": 0,
        "life_time_value": 0
      }
    ]
  },
  "status": 0,
  "error_response": {
    "code": "string",
    "message": "string",
```

```
    "raw_message": "string",
    "stack_trace": "string"
  },
  "logs": [
    {
      "log_level": "Trace",
      "message": "string",
      "stack_trace": "string",
      "time_occurred": "2018-12-03T15:21:11.389Z",
      "duration": 0,
      "scope_observer_id": "00000000-0000-0000-0000-000000000000",
      "entity_id": "00000000-0000-0000-0000-000000000000"
    }
  ],
  "message": "string",
  "scope": {
    "retailer_id": "string",
    "event": "string",
    "extended_scope": {}
  },
  "request_payload": {},
  "authorized_integrator_id": "00000000-0000-0000-0000-000000000000",
  "started": "2018-12-03T15:21:11.389Z",
  "completed": "2018-12-03T15:21:11.389Z"
}
```

Step 2: Increase or Decrease Balance of Customer Point Accounts

Now, since the balance is known, you can increase (deposit) or decrease (spend) the account(s).

Increasing a Point Balance

This endpoint allows you to deposit points into a customer's account.

```
POST /api/1.0/user_points/deposit
```

The endpoint passes in a request object that specifies the points being deposited. It requires both a point account and a point source:

Request

```
{
  "retailer_id": "string",
```

```
  "user_id": "string",
  "deposit_details": [
    {
      "point_source_id": "00000000-0000-0000-0000-000000000000",
      "amount": 0,
      "user_point_account_id": "00000000-0000-0000-0000-000000000000",
      "point_account_id": "00000000-0000-0000-0000-000000000000",
      "reference_id": "string",
      "reference_type": "string",
      "transaction_id": "00000000-0000-0000-0000-000000000000"
    }
  ]
}
```

The *point_source_id* attribute identifies the source, or origination point pool, for the points. For example, clients may set up multiple point sources to track distinct liabilities for different categories of points, such as base points, bonus points, and store-branded credit card p[oints.

The *point_account_id* attribute is required to specify into what account the points are to be deposited and to increase the account's existing point balance. Furthermore, the points inherit the expiration policy associated with this point account.

The *reference_id* and *reference_type* attributes are optional but can be very useful when trying to reconcile the point audit log to activity originating in a third-party system. For example, if the API were to be invoked by a POS system, it would be useful to define the reference type as POS and the reference ID as some transaction ID from the POS.

The *transaction_id* attribute is also optional; it too can be useful for reconciliation purposes. The *user_point_account_id* attribute is optional and rarely required.

After the endpoint makes the request for the deposit, the platform returns a response object, which is shown below:

Response

```
{
  "response_payload": {
    "retailer_id": "string",
    "user_id": "string",
    "tracking_id": "string",
    "details": [
      {
        "account_name": "string",
        "user_point_account_id": "string",
        "point_account_id": "string",
        "deposit_amount": 0,
```

```
      "available_balance": 0,
      "life_time_value": 0
    }
  ],
  "audits": [
    {
      "id": "string",
      "retailer_id": "string",
      "user_id": "string",
      "account_name": "string",
      "point_account_id": "string",
      "user_point_account_id": "string",
      "modification": 0,
      "amount_spent": 0,
      "amount_expired": 0,
      "audit_type": "Other",
      "modification_type": "string",
      "modification_entity_id": "string",
      "spend_weight": 0,
      "point_source_id": "string",
      "point_source_name": "string",
      "time_of_occurrence": "2018-12-03T15:21:11.438Z",
      "request_id": "string",
      "transaction_id": "string"
    }
  ]
},
"status": 0,
"error_response": {
  "code": "string",
  "message": "string",
  "raw_message": "string",
  "stack_trace": "string"
},
"logs": [
  {
    "log_level": "Trace",
    "message": "string",
    "stack_trace": "string",
    "time_occurred": "2018-12-03T15:21:11.438Z",
    "duration": 0,
    "scope_observer_id": "00000000-0000-0000-0000-000000000000",
    "entity_id": "00000000-0000-0000-0000-000000000000"
  }
],
"message": "string",
"scope": {
  "retailer_id": "string",
  "event": "string",
  "extended_scope": {}
},
"request_payload": {},
"authorized_integrator_id": "00000000-0000-0000-0000-000000000000",
```

```
  "started": "2018-12-03T15:21:11.438Z",
  "completed": "2018-12-03T15:21:11.438Z"
}
```

Decreasing a Point Balance

This endpoint allows you to spend points, reducing points in a customer's account.

```
POST /api/1.0/user_points/spend
```

You can can choose to include or omit a specific point account in the request object. If the object does not include a point account, the system uses the default point account set up in the platform's Point Management Module. Currently, the platform only supports a FIFO (first in first out) spend policy. In other words, the oldest points will be spend first.

The endpoint passes in a request object that specifies the points being spent:

Request

```
{
  "retailer_id": "string",
  "user_id": "string",
  "amount": 0,
  "point_account_ids": [
    "00000000-0000-0000-0000-000000000000"
  ],
  "reference_id": "string",
  "reference_type": "string",
  "transaction_id": "00000000-0000-0000-0000-000000000000",
  "force_spend": true
}
```

The *reference_id* and *reference_type* attributes are optional but can be very useful when trying to reconcile the point audit log to activity originating in a third-party system. For example, if the API were to be invoked by a POS system, it would be useful to define the reference type as POS and the reference ID as some transaction ID from the POS.

The *transaction_id* attribute is also optional; it too can be useful for reconciliation purposes.

The *force_spend* attribute is typically set to "false." When set to "false," the system follows all parameters defined in the point account definition. For example, the system won't allow the customer to spend points they do not have. If *force_spend* is "true," it will spend the points

regardless of whether or not the customer has sufficient balance. Depending on the point account configuration, this event might result in the customer having a negative balance.

After the endpoint makes the request for the spend, the platform returns a response object, which is shown below:

Response


```
{
  "response_payload": {
    "retailer_id": "string",
    "user_id": "string",
    "tracking_id": "string",
    "summary": {
      "amount_spent": 0,
      "remaining_balance": 0
    },
    "details": [
      {
        "account_name": "string",
        "user_point_account_id": "string",
        "point_account_id": "string",
        "amount_spent": 0,
        "remaining_balance": 0
      }
    ],
    "audits": [
      {
        "id": "string",
        "retailer_id": "string",
        "user_id": "string",
        "account_name": "string",
        "point_account_id": "string",
        "user_point_account_id": "string",
        "modification": 0,
        "amount_spent": 0,
        "amount_expired": 0,
        "audit_type": "Other",
        "modification_type": "string",
        "modification_entity_id": "string",
        "spend_weight": 0,
        "point_source_id": "string",
        "point_source_name": "string",
        "time_of_occurrence": "2018-12-03T15:21:11.411Z",
        "request_id": "string",
        "transaction_id": "string"
      }
    ]
  },
  "status": 0,
  "error_response": {
```

```
    "code": "string",
    "message": "string",
    "raw_message": "string",
    "stack_trace": "string"
  },
  "logs": [
    {
      "log_level": "Trace",
      "message": "string",
      "stack_trace": "string",
      "time_occurred": "2018-12-03T15:21:11.411Z",
      "duration": 0,
      "scope_observer_id": "00000000-0000-0000-0000-000000000000",
      "entity_id": "00000000-0000-0000-0000-000000000000"
    }
  ],
  "message": "string",
  "scope": {
    "retailer_id": "string",
    "event": "string",
    "extended_scope": {}
  },
  "request_payload": {},
  "authorized_integrator_id": "00000000-0000-0000-0000-000000000000",
  "started": "2018-12-03T15:21:11.411Z",
  "completed": "2018-12-03T15:21:11.411Z"
}
```