# Release Bulletin Sample

(Suite Name) (Product Name) **- Automate TMPG Reporting**

|  |  |
| --- | --- |
| **Enhancement #** | XXXXXXX |
| **Business Objective** | (Suite Name) to automate the reporting of failed transactions as prescribed by Treasury Market Practice Group  (TMPG). |
| **Application Affected** | (Suite Name) (Product Name) |
| **Description** | (Suite Name) is enhanced as follows:  1. Identifies the failed transaction(s) based on the following condition:  clearance date > settlement date.  2. Derives the fail charges based on the following formula:  Fail Charges = (TMPG reference rate \* Failing Settlement Amount \* Number of days failing)/360   * TMPG reference rates are maintained in the application. * If a trade fails on day 1, then the **Fail Days** column is updated as ‘**1**’. For subsequent fails, the **Fail Days**   column is updated accordingly and fail charges is calculated.   * Even weekends and trading holidays are considered as active fail days.   3. Reports the failed transactions and their fail status on the **CTA** screen under the three columns **Fail**  **Charges**, **Fail Days**, and **Claim Status**.  You can add your reason(s) for fails under the **Comment** column.   * The manual actions **Waive off** and **Close** (entitlement controlled) are enabled on the **CTA** screen and **Multi Action** drop-down for these failed transactions.   4. Purges these failed transactions (which are either in **Settled** or **Waive off** status) from the application after a  number of days defined in the application.  These failed transaction(s) can be audited from the **View Details** window. |
| **Business Benefit** | (Suite Name) automatically calculates the fail charges and reports their transaction(s) on its UI, thus helping clients avoid calculate fail charges manually and view those transaction(s). |
| **Screen Modification** | The following screenshot shows the below listed items on the **CTA** screen:  l **Fail Days**, **Fail Charges**, and **Claim Status** fields containing values  l Manual Actions–**Close**, **Revise Counter Party**, and **Waive off**.  The following screenshot shows the manual actions **Waive off** and **Close** options on the **Choose Action** dropdown:  of the **Multi Actions** window: |
| **Additional Setup** | Not Applicable |
| **Limitations** | * Settlement of receipt/deliver fail charges is handled outside the application. * If failed transaction(s) gets either closed/waived off, then no updates are sent to the client source system. * For subsequent transaction(s) fails, (Suite Name) does not show the previous fail charges. * Only two sets of fail reference rates are available to maintain in the application at any given point in time. * For failed transactions on which cancel/amendment have been received, (Suite Name) does not calculate the fail charges. |
| **Related Enhancements** | Not Applicable |
| **More Details** | Not Applicable |

# Release Fix Report (Enhancement) Sample

|  |  |
| --- | --- |
| Enhancement Number | XXXXXX |
| Severity | Medium |
| Summary | (Suite Name) receives the SWIFT pledge instruction message from the client source system and proceeds as follows:   * Populates the Bank Loan Account Number field with the default value of "000000002378" on the BPS-CAGE pledge inbound message. This field value is no longer mapped and populated from the 97A tag.   The default value can be configured from the DB script.   * Populates the Customer Account Number field on the BPS-CAGE pledge inbound message based on the value populated on the 97A tag as follows:   + If the 97A tag has the value populated with fewer than 20 characters such as "123456789A," then this field is padded with empty spaces to fill 20 characters and populated as " 123456789A."      * + If the 97A tag has the value populated with more than 20 characters such as 123456789ABCDEFGHIJKLMNO, then this field is populated by deleting the final 4 characters such as "123456789ABCDEFGHIJK." |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| Test Procedure | Test Step | Expected Result |
|  |  |  |
| 1. Validate the 'Bank Loan Account Number' value populated in Pledge inbound message for MT540 | | |
|  | 1. Process OCC pledge incoming SWIFT instruction– MT540  2. Login to the application and Navigate to CTA  3. Find the trade in CTA and right click it, click view details  4. Go to audit and click show message-->Raw Message for (Product Name) message  5. Validate the message for "Bank Loan Account Number" value to be defaulted to ‘000000002378’, at the position 77 to 88 (12 char fixed length) | 1. OCC pledge SWIFT instruction must be processed successfully to Clearance Pro  2. User must be logged in successfully and CTA page must be displayed  3. View details pop must be displayed  4. (Product Name) Raw Message must be displayed  5. ‘000000002378' 12 char fixed length number must be defaulted for Bank loan account number from 77 to 88 positions |
| 2. Validate the 'Bank Loan Account Number' value populated in Pledge inbound message for MT542 | | |
|  | 1. Process OCC pledge incoming SWIFT instruction– MT542  2. Login to the application and Navigate to CTA  3. Find the trade in CTA and right click it, click view details  4. Go to audit and click show message-->Raw Message for (Product Name) message  5. Validate the message for "Bank Loan Account Number" value to be defaulted to ‘000000002378’, at the position 77 to 88 (12 char fixed length) | 1. OCC pledge SWIFT instruction must be processed successfully to (Suite Name)  2. User must be logged in successfully and CTA page must be displayed  3. View details pop must be displayed  4. (Product Name) Raw Message must be displayed  5. ‘000000002378' 12 char fixed length number must be defaulted for Bank loan account number from 77 to 88 positions |
|  |  |  |

# Release Fix Report (Defect) Sample

Defect Number (XXXXXXX)

Details

|  |  |
| --- | --- |
| Application Affected | XXX |
| Severity | High |
| Summary | The PairOff Confirmation dialog box did not display the parent and resultant trades. |
| Details | After the Pairoff manual action on the CTA screen for a trade having an External Status as Pending, the PairOff Confirmation dialog box displayed just the Loading… text on it but displayed neither the resultant nor the parent trades even after a long wait.  Further, the parent trades were displayed in Pending status, but the resultant trade was not displayed on the CTA screen. |
| Solution | The parent trades and resultant trade display on the PairOff Confirmation dialog box and CTA screen. |

Test Procedure

|  |  |  |
| --- | --- | --- |
| Step No. | Description | Expected Result |
| 1. | Log on to ClearancePro application. | User must be logged in successfully. |
| 2. | Navigate to Views>Position View screen | User must be able to navigate to the Position View screen successfully. |
| 3. | Validate the Position View screen. | There must be numerous line items with different securities and their positions in different accounts. |
| 4. | Perform a manual action on the one of the line items (CA135087VH40). | All the below manual options must be displayed.   * Select All * Select None * Inter Account Movement * New South Bound Trade * New Trade * Position Change Details * Show All Outstanding Activity |
| 5. | Click the Inter Account Movement manual option. | Inter Account Movement screen must be successfully opened. |
| 6. | Validate the Acc Details. | Acc Details must be "SA000 and GA000." |

# User Guide (Sample)

Setup

The administrator has privileges to set up accounts in the application.

From the application menu, select Setup. The following menu items are displayed:

* Users
* Organizations
* Groups
* Entitlements
* SOD Cap Set up
* Users Login History
* Scheduler Notification
* Potential Matching
* Alert Notification
* Match Rules ICM
* Security Maturity Alerts
* Customer Mapping Configuration

**Organizations Screen**

As an administrator, you have the privilege to set up the organization details of the user in the application.

## Accessing the Organizations Screen

From the application menu, select Setup > Organizations. The Organizations screen displays. [See Organizations Screen: Fields and Descriptions on the next page.](XXX)

## Viewing the Organizations List

You can view all organizations available in the application.

To view organizations list:

1. Access the Organizations screen.

2. Click the icon.

3. Enter the search criteria, then click the icon. The Organizations screen displays all organizations of various users available in the application.

## Adding an Organization Information

You can add an organization information to the application.

To add an organization information:

1. Access the [Organizations screen](XXXX).

2. Click the icon. The Add Organization Details window displays to enter the details.

3. Make the necessary changes.

4. Click Add. The organization with its details is added to the application.

## Modifying an Organization Information

You can modify an organization information in the application.

To modify an organization information:

1. Access the [Organizations screen](XXXX).

2. Click the icon. The Edit Organization Details window displays to edit the details.

3. Make the necessary changes.

4. Click Update. The edited details are updated in the application.

## Deleting an Organization Information

You can delete an organization information in the application.

To delete an organization information:

1. Access the [Organizations screen](XXXXX).

2. Click the icon. The delete confirmation window displays asking whether you want to delete the organization.

3. Click OK. The organization with its details is deleted from the application.

## Organizations Screen: Fields and Descriptions

|  |  |
| --- | --- |
| Field | Description |
| Icon |  |
| Icon-1 | Description |
| Icon-2 | Description |
| Icon-3 | Description |
| **Organizations** |  |
| Organization Name | Name of the organization |
| Address | Address of the organization |
| Layout Name | Layout name of the organization |
| Style Name | Style name of the organization |

# Code Documentation

The following code annotation sample is developed using the JSDoc markup language for JavaScript on Visual Studio Code editor:

## Using “Visual Studio Code” editor

Index.js file

// @ts-check

const { caltotal } = require("./petcounter");

/\*\*

 \* @file index.js is the homepage for this application

 \* @author Mithun

 \* @see <a href="https://en.wikipedia.org/wiki/India">India</a>

 \*/

/\*\*

 \* Pet name

 \* @type {string}

 \*/

const string = "Ruby";

/\*\*

 \* Pet number

 \* @type {number}

 \*/

const number = 100;

/\*\*

 \* My Array

 \* @type {Array<number>}

 \*/

const myArray\_number = [10, 132.12, 100];

// const myArray = [10, 132.12, 100, "Hi", true]

/\*\*

 \* Pet object

 \* @type {{id: number, name: string, age: number|string}}

 \*/

const object = {

  id: 1,

  name: "Coco",

  age: "2",

};

/\*\*

 \* Calculate pet age

 \* @param {number} current current year

 \* @param {number} yearOfBirth year of pet birth

 \* @returns {string} pet age

 \*/

const calculateAge = (current, yearOfBirth) => {

  return `${current - yearOfBirth}`;

};

console.log(calculateAge(2021, 2019));

//////////////////////////

/\*\*

 \*

 \* @typedef {Object} Dog

 \* @property {number} id

 \* @property {string} name

 \* @property {number|string} age

 \* @property {boolean} [isMale] gender {optional}

 \*/

/\*\*

 \* Custom Object

 \* @type {Dog}

 \*/

const custom\_object = {

  id: 1,

  name: "Bean",

  age: 2,

  // isMale: true,

};

/\*\*

 \* Class to create a new pet owner

 \*/

class Owner {

  /\*\*

   \* Pet owner detail

   \* @param {Object} ownerDetail

   \*/

  constructor(ownerDetail) {

    /\*\*

     \* @property {string} name pet owner name

     \*/

    this.name = ownerDetail.name;

    /\*\*

     \* @property {number} age pet owner age

     \*/

    this.age = ownerDetail.age;

  }

  /\*\*

   \* @property {Function} printOwner print out owner information

   \* @returns {void}

   \*/

  printOwner() {

    console.log(`Owner's name is ${this.name} and her age is ${this.age}`);

  }

}

/\*\*

 \* Link to Owner class

 \* See {@link Owner}

 \*/

const owner\_call = new Owner({

  name: "Kelly",

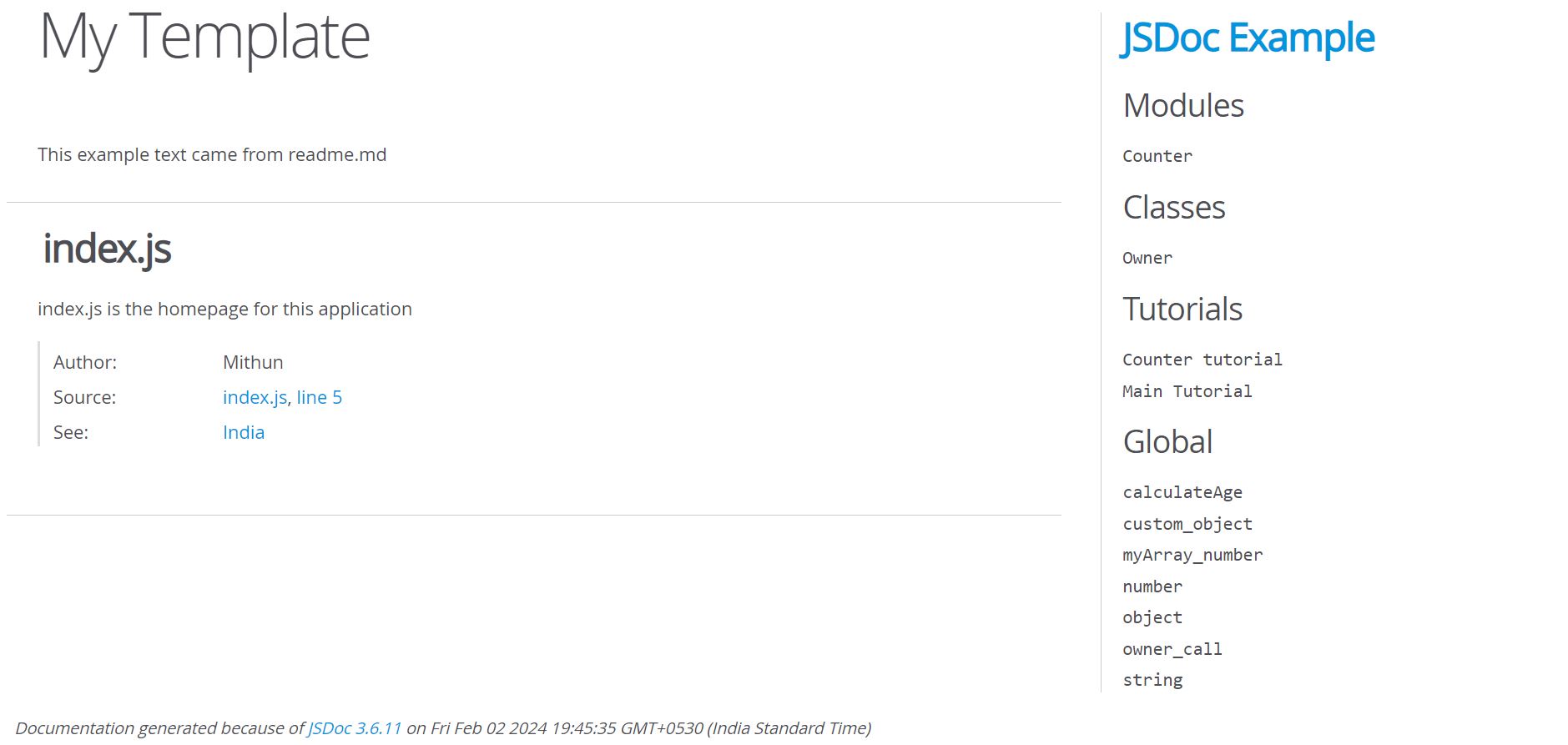
  age: 18,

});

owner\_call.printOwner();

console.log(caltotal(19, 5));

## Generated document using the automation generation tool (JSDoc)



# Documenting a JSON Request

The following JSON code sample represents a request to book a conference room:

{

"conference": {

"time": "2024-01-31 11:00",

"duration": 30,

"description": "2016 Project Planning Meeting",

"location": "Tower: 2, Building 18, Room 108",

"reminder": 18,

"invitees": ["Vinay@example.com", "Sangeetha@example.com",

"Siri@example.com", "Prathap@example.com"]

}

}

The following sample is the documentation for the above JSON request:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Element** | | **Description** | **Type** | **Required** | **Notes** |
| Conference | | Top Level | Conference data object | Required |  |
|  | time | When the meeting begins | String | Required | Format:  YYYY-mm-dd HH:MM |
|  | duration | How long the conference lasts | String | Required |  |
|  | description | A description of the conference | String | Required |  |
|  | location | Location of the conference | String | Optional | Default is an empty string |
|  | reminder | How many minutes before the meeting a reminder event should take place | String | Optional | Default is 10 minutes. |
|  | invitees | A list of email addresses of people to invite to the meeting | String | Optional | The strings should be valid email addresses. Default is an empty array. |
|  |  |  |  |  |  |

# Documenting an XML Request

The following XML code sample represents a request for a daily data from an airport weather office:

<routineData>

<date>2024-31-01</date>

<hourlyData>

<time>10:00</time>

<device>

<id>28</id>

<temperature>40</temperature>

<humidity>11</humidity>

</device>

<device>

<id>29</id>

<temperature>41</temperature>

<humidity>9</humidity>

</device>

. ..

</hourlyData>

<hourlyData>

<time>11:00</time>

<device>

<id>28</id>

<temperature>38</temperature>

<humidity>10</humidity>

</device>

...

</hourlyData>

...

</routineData>

The following sample is the documentation for the above xml request:

Top Level

|  |  |  |
| --- | --- | --- |
| **Element** | **Description** | **Type** |
| routineData | Humidity and temperature data for one day | routineData element |

routineData: Represents humidity and temperature data for one day

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Description** | **Type** | **Notes** |
| date | The date when the humidity and temperature data was collected | String | Format: YYYY-mm-dd |
| hourlyData | Humidity and temperature data for one hour | hourlyData element |  |

hourlyData: Humidity and temperature data for one hour

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Description** | **Type** | **Notes** |
| time | Local time when the data was collected | String | Format: YYYY-mm-dd |
| device | One or more device objects having the data | device element |  |

device: contains temperature and humidity data collected from a device object

|  |  |  |
| --- | --- | --- |
| **Element** | **Description** | **Type** |
| Id | The device’s identifier | number |
| temperature | The measured temperature in degree celsius | number |
| humidity | The measured humidity in percentage | number |

# REST APIs Documentation

The following sample shows how to document the POST and GET calls:

## **Uploading a Sound File**

Uploads a sound file for a user’s profile.

### URL

POST <https://api.date.com/profile/sound>

### Headers

|  |  |  |  |
| --- | --- | --- | --- |
| **Header Name** | **Description** | **Required** | **Values** |
| Bearer | Access token | Required | See Authorization section. |
| Content-Type | The format of the sound file to upload | Optional | audio/mpeg or audio/x-wav.  Default is audio/mpeg. |
| Accept | The format of the returned data | Optional | application/xml or application/json. Default is application/json. |

### POST or PUT Body

The sound file.

**Note:** The sound file must be 3 minutes or shorter.

### Sample Request

POST <https://api.date.com/profile/sound>

Bearer: {access token}  
Content-Type: audio/mpeg  
Accept: application/json

{sound file}

### Response

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Description** | **Type** | **Notes** |
| Id | The ID of the new sound file | number |  |
| length | The length of the sound file | float | Length is in seconds |

### Sample Response

{

"id": 3543,

"length": 19.8

}

## **Retrieving a List of Sound Files**

Retrieves a list of sound file URLs and their lengths for the specified user.

### URL

GET https://api.date.com/user/{user id}/profile/sound

where {user id} is the ID of the user whose profile contains the sound files.

### Query Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Description** | **Type** | **Required** | **Notes** |
| sortOrder | The order to return the sound file information. | String | Optional | Valid values: mostRecent, earliest, shortest, longest.  Default is mostRecent. |

**Note:**

* **mostRecent** returns the most recent sound files to the earliest.
* **earliest** returns the earliest sound files to the most recent.
* **shortest** returns the shortest sound files to longest.
* **longest** returns the longest sound files to the shortest

### Headers

|  |  |  |  |
| --- | --- | --- | --- |
| **Header Name** | **Description** | **Required** | **Values** |
| Bearer | Access Token | Required | See Authorization section |
| Accept | The format of the returned data | Optional | application/xml or application/json. Default is application/json |

### Sample Request

GET <https://api.sounddate.com/user/345354/profile/sound?sortOrder=shortest>

Bearer: {access token}  
Accept: application/json

### Response

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Description** | **Type** | **Notes** |
| Soundfiles | List of sound file information | Array |  |
| id | The ID of the sound file | integer |  |
| url | The URL of the sound file | string |  |
| length | The length of the sound file | Float | The length in seconds. |

Sample Response

{

"soundFiles": [

{

"id": 23456,

"url": "https://api.sounddate.com/profile/sound/23456.mp3",

"length": 11.2

},

{

"id": 24559,

"url": "https://api.sounddate.com/profile/sound/24559.mp3",

"length": 19.8

}

]

}

## Status Codes and Errors

The following table lists the returned HTTP status codes.

|  |  |  |
| --- | --- | --- |
| **Code** | **Description** | **Notes** |
| 200 | OK | Sound file was successfully added. |
| 401 | Unauthorized | Access token is invalid |
| 413 | Request Entity Too Large | Uploaded sound file is longer than 5 minutes. |

# Open API Specification Documentation

openapi: 3.0.1

info:

title: Strope Usage Records

description: "Usage records allow you to report customer usage and metrics to Stripe\

\ for [metered billing]( https://stripe.com/docs/billing/subscriptions/metered-billing)\

\ of subscription plans."

version: 2019-02-19

servers:

- url: https://api.stripe.com/v1

security:

- basicAuth: []

paths:

/subscription\_items/{subscription\_item}/usage\_records:

post:

description: |

Creates a usage record for a specified subscription item and date, and fills it with a quantity.

Usage records provide `quantity` information that Stripe uses to track how much a customer is using your service. With usage information and the pricing model set up by the [metered billing](https://stripe.com/docs/billing/subscriptions/metered-billing) plan, Stripe helps you send accurate invoices to your customers.

The default calculation for usage is to add up all the `quantity` values of the usage records within a billing period. You can change this default behavior with the billing planâ€™s `aggregate\_usage` [parameter](https://stripe.com/docs/api/plans/create#create\_plan-aggregate\_usage). When there is more than one usage record with the same timestamp, Stripe adds the quantity values together. In most cases, this is the desired resolution, however, you can change this behavior with the `action` parameter.

The default pricing model for metered billing is [per-unit](https://stripe.com/docs/api/plans/object#plan\_object-billing\_scheme) pricing. For finer granularity, you can configure metered billing to have a [tiered](https://stripe.com/docs/billing/subscriptions/tiers) pricing model.

parameters:

- name: subscription\_item

in: path

description: The ID of the subscription item for this usage record.

required: true

schema:

type: string

requestBody:

content:

application/x-www-form-urlencoded:

schema:

required:

- quantity

type: object

properties:

quantity:

type: string

description: The usage quantity for the specified timestamp.

action:

type: string

description: |

Valid values are `increment` (default) or `set`. When using `increment` the specified quantity will be added to the usage at the specified timestamp. The `set` action will overwrite the usage quantity at that timestamp. If the subscription has [billing thresholds](https://stripe.com/docs/api/subscriptions/object#subscription\_object-billing\_thresholds), increment is the only allowed value.

enum:

- increment

- set

timestamp:

type: string

description: "The timestamp for the usage event. This timestamp\

\ must be within the current billing period of the subscription\

\ of the provided `subscription\_item`, and must not be in the\

\ future. When passing `\"now\"`, Stripe records usage for the\

\ current time. Default is `\"now\"` if a value is not provided."

required: true

responses:

"200":

description: Successful response

content:

application/json:

schema:

$ref: '#/components/schemas/usageRecord'

/subscription\_items/{subscription\_item}/usage\_record\_summaries:

get:

description: |

For the specified subscription item, returns a list of summary objects. Each object in the list provides usage information thatâ€™s been summarized from multiple usage records and over a subscription billing period (e.g., 15 usage records in the billing planâ€™s month of September).

The list is sorted in reverse-chronological order (newest first). The first list item represents the most current usage period that hasnâ€™t ended yet. Since new usage records can still be added, the returned summary information for the subscription itemâ€™s ID should be seen as unstable until the subscription billing period ends.

parameters:

- name: subscription\_item

in: path

description: Only summary items for the given subscription item.

required: true

schema:

type: string

- name: ending\_before

in: query

description: "A cursor for use in pagination. `ending\_before` is an object\

\ ID that defines your place in the list. For instance, if you make a list\

\ request and receive 100 objects, starting with `obj\_bar`, your subsequent\

\ call can include `ending\_before=obj\_bar` in order to fetch the previous\

\ page of the list."

schema:

type: string

- name: limit

in: query

description: "A limit on the number of objects to be returned. Limit can range\

\ between 1 and 100, and the default is 10."

schema:

maximum: 100

minimum: 1

type: integer

- name: starting\_after

in: query

description: "A cursor for use in pagination. `starting\_after` is an object\

\ ID that defines your place in the list. For instance, if you make a list\

\ request and receive 100 objects, ending with `obj\_foo`, your subsequent\

\ call can include `starting\_after=obj\_foo` in order to fetch the next page\

\ of the list"

schema:

type: string

responses:

"200":

description: Successful response

content:

application/json:

schema:

$ref: '#/components/schemas/usageRecordSummary'

components:

schemas:

usageRecord:

type: object

properties:

id:

type: string

description: Unique identifier for the object.

object:

type: string

description: String representing the objectâ€™s type. Objects of the same

type share the same value.

enum:

- usage\_record

livemode:

type: boolean

description: Has the value `true` if the object exists in live mode or the

value `false` if the object exists in test mode.

quantity:

minimum: 0

type: integer

description: The usage quantity for the specified date.

subscription\_item:

type: string

description: The ID of the subscription item this usage record contains

data for.

timestamp:

type: integer

description: The timestamp when this usage occurred.

usageRecordSummary:

type: object

properties:

object:

type: string

description: String representing the objectâ€™s type. Objects of the same

type share the same value.

enum:

- list

url:

type: string

description: URL of the subscription item summaries.

has\_more:

type: boolean

description: "Whether or not there are more elements available after this\

\ set. If `false`, this set comprises the end of the list."

data:

type: array

items:

$ref: '#/components/schemas/usageRecordData'

description: "A dictionary with a `data` property that contains an array of\

\ up to `limit` summaries, starting after summary `starting\_after`. Each entry\

\ in the array is a separate summary object. If no more summaries are available,\

\ the resulting array is empty."

usageRecordData:

type: object

properties:

id:

type: string

description: ID of the usage record.

object:

type: string

description: String representing the objectâ€™s type. Objects of the same

type share the same value.

enum:

- usage\_record\_summary

invoice:

type: string

description: ID of the invoice.

livemode:

type: boolean

description: Has the value `true` if the object exists in live mode or the

value `false` if the object exists in test mode.

period:

type: object

properties:

end:

type: integer

description: Ending timestamp.

start:

type: integer

description: Starting timestamp.

description: Time period.

subscription\_item:

type: string

description: ID of the subscription item.

total\_usage:

type: integer

description: The total number of usage records in the billing plan.

securitySchemes:

basicAuth:

type: http

scheme: basic

x-original-swagger-version: "2.0"