Benepik

Benepik: Reward Credit API

Version: 1.0

Date: 18-12-2025

Prepared By: Benepik Technology Private Limited

Purpose: Technical Specification for API-Based Reward Disbursement

Confidential – For Authorized Use Only

1. Introduction

This document describes the complete technical specification for integrating with the **Benepik Reward Credit API**. It covers authentication, security controls, request/response structures, encryption and signing mechanisms, validation rules, error handling, and operational constraints.

The API enables clients to securely submit **single or bulk reward disbursement requests** via a POST endpoint.

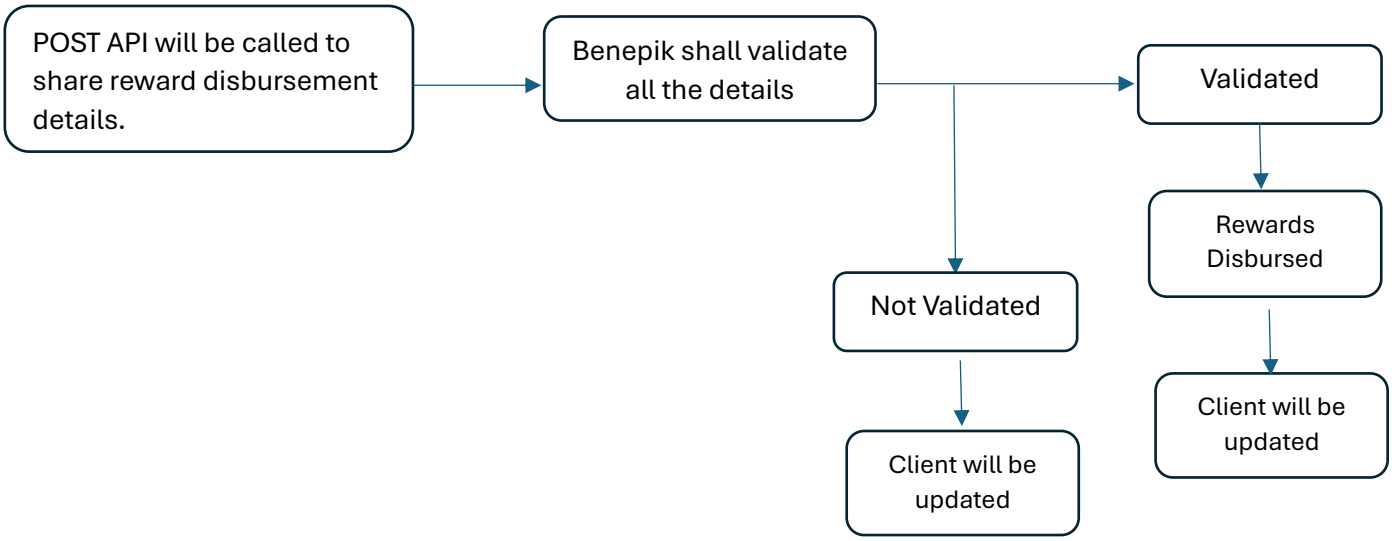
2. System Overview

The Reward Credit API supports **single and bulk reward submissions**. Each request passes through multiple validation layers:

i. IP Whitelisting
ii. JWT Token Validation
iii. HMAC Signature Validation
iv. Encrypted Checksum Decryption
v. Business Rule Validation
vi. Balance Availability Validation (Cost-Centre Level)

If all validations succeed, rewards are processed. If any validation fails, an appropriate error code and message are returned.

3. Workflow



i. Client triggers a **POST API** with encrypted and signed reward data.
ii. Benepik validates:

- Source IP
- JWT token
- HMAC signature

- Encrypted checksum payload
- Business rules and balance availability

iii. **If validated:** Rewards are disbursed successfully.
iv. **If not validated:** The client receives an error response.
v. If configured, the **final transaction status is sent via Webhook**.

4. Scope of Work

Sr. No.	Benepik Responsibilities	Client Responsibilities
i.	Provide authentication parameters (clientCode, authKey, secretKey)	Provide accurate reward recipient data.
ii.	Share Entity IDs, Mailer IDs, Certificate IDs	Maintain whitelisted IP addresses.
iii.	Validate and process reward submissions.	Ensure unique Transaction IDs.
iv.	Return structured success/error responses.	Maintain sufficient balance.

Note: Requests are accepted **only from whitelisted IP addresses.**

5. API Endpoint Details

- **HTTP Method:** POST
- **Endpoint URL:** {{bpcp_client}}api/sendRewards.
- **Protocol:** HTTPS (TLS ≥ 1.2 mandatory)

6. Client Authentication & Security

6.1. Required Headers

Header	Description
Authorization	Bearer <JWT_TOKEN>
REQUESTID	Client request identifier
X-TIMESTAMP	UNIX timestamp (seconds)
X-NONCE	Unique random value per request
X-SIGNATURE	HMAC-SHA256 signature
Content-Type	application/json

6.2. JWT Token Generation (Server-Side Only)

- **Algorithm:** HS256
- **Expiry:** 15 minutes
- **Used for:** Client authentication

Mandatory Claims

Claim	Description
iat	Issued at timestamp
exp	Expiry timestamp
iss	Token issuer
aud	Intended audience
jti	Unique token ID
clientId	Client identifier
adminId	Admin identifier

```
const jwt = require('jsonwebtoken');
const crypto = require('crypto');

function createBearerToken() {
 const authKey = "Kjs8df8tyTTJf92nq#3Jasf82^@2Lncs90dkfLcm03Fjs9";

const issuedAt = Math.floor(Date.now() / 1000);
 const expire = issuedAt + 900;

 const payload = {
  iat: issuedAt,
  exp: expire,
  iss: 'benepik-tech',
  aud: 'maytech-corp',
  jti: Buffer.from(crypto.randomBytes(16)).toString('base64'),
  clientId: 1200,
  event: 'reward',
  adminId: 23
 };
```

```
  return jwt.sign(payload, authKey, { algorithm: 'HS256' });
}
```

Note: authKey must never be exposed in frontend or browser code.

6.3. HMAC Signature Generation

- **Algorithm:** HMAC-SHA256
- **Secret Used:** secretKey
- **Signature String Format:** REQUESTID|X-TIMESTAMP|X-NONCE|checksum
- Base64-encode the final hash
- Send as `X-SIGNATURE` header

6.4. Encrypted Checksum

- **Algorithm:** AES-256-CBC
- **Key:** SHA-256(secretKey)
- **Payload:** Full request body (JSON)
- **Format:** Base64(IV + EncryptedPayload)

```
function generateChecksum(payload) {
  const secretKey = "UIw@8Jsk#28!dfjWm91zPqL7v6$Bnq02XakNfVp";
  const iv = crypto.randomBytes(16);
  const key = crypto.createHash('sha256').update(secretKey).digest();

  const cipher = crypto.createCipheriv('aes-256-cbc', key, iv);
  let encrypted = cipher.update(JSON.stringify(payload), 'utf8', 'base64');
  encrypted += cipher.final('base64');

  const combined = Buffer.concat([iv, Buffer.from(encrypted,
'base64')]);
  return {
    checksum: combined.toString('base64')
  };
}
```

7. Request Payload

Test Server URL:
Request URL: {{bpcp_client}}api/sendRewards

Request POST Parameters:


```
{
 "source": "0",
 "isSms": "1",
 "isWhatsApp": "1",
 "isEmail": "1",
 "data": [
  {
   "sno": "1",
   "userName": "Rahul",
   "emailAddress": "rahul.singh@gmail.com",
   "countryCode": "+91",
   "mobileNumber": "9999999999",
   "rewardAmount": "1",
   "personalMessage": "Impressive performance!",
   "messageFrom": "Akash",
   "ccEmailAddress": "",
   "bccEmailAddress": "",
   "reference": "",
   "mailer": "",
   "certificateId": "",
   "transactionId": "XYZA-851061147106-UYTRY-571789",
   "entityId": "1886",
   "column1": "",
   "column2": "",
   "column3": "",
   "column4": "",
   "column5": ""
  }
 ]
}
```


Limit: Maximum 500 users per API hit.

8. Response Structure

8.1 . Success Response

Success


```
{
  "code": 1000,
  "success": 1,
  "message": "Batches processed successfully",
  "batchResponse": [
    {
      "code": 1000,
      "success": 1,
      "message": "Reward processed successfully",
      "txns": [
        {
          "transactionId": "XYZA-851061147106-UYTRY-571789",
          "rewardAmount": "1"
        }
      ]
    }
  ]
}
```


8.2. Failure Response

Failure 1


```
{
  "success": 0,
  "error": {
    "1": [
      "Invalid Txn Id XYZA-85061147106-UYTRY-571788"
    ]
  },
  "code": 1010
}
```


Failure 2


```
{
  "success": 0,
  "error": {
    "1": [
      "Request repeated"
    ]
  },
  "code": 1022
}
```

9. Error Responses and Handling

Sr. No.	Error Code	HTTP Status	Description
1	**1000 – SUCCESS**	**200**	Request completed successfully.
2	**1001 – Unauthorized IP Address**	**401**	The request comes from an IP address that is not allowed.
3	**1002 – Invalid Client Code**	**401**	The client code (RequestID) is invalid or does not exist.
4	**1003 – Client Code Missing**	**401**	The request did not contain a required RequestID header.
5	**1004 – Missing/Invalid Bearer Token**	**401**	The Authorization: Bearer <token> header is missing or improperly formatted.
6	**1005 – Authentication Failed**	**401**	The JWT could not be verified (invalid signature, corrupted token, etc.).
7	**1006 – Token Expired**	**401**	The JWT has expired and must be regenerated.
8	**1007 – Checksum Required**	**401**	The request payload does not include the required encrypted checksum.
9	**1008 – Invalid Checksum**	**401**	Encrypted data could not be decrypted; checksum may be invalid or payload tampered.
10	**1009 – Required Parameter Missing**	**200**	One or more mandatory fields in the decrypted payload are missing.
11	**1010 – Input Error**	**200**	Payload fields are invalid, malformed, or in incorrect format.
12	**1011 – Unauthorized Access**	**401**	Access is denied due to insufficient permission or invalid credentials.
13	**1012 – Insufficient Balance**	**200**	Client does not have sufficient balance while processing the request. It can be cost center based
14	**1013 – No Rewards to process**	**200**	If the process has no rewards to process or rewards already processed.
15	**1050**	**200**	Pending/Request accept don't reinitiate same request
16	**1020**	**401**	HMAC_HEADER_MISSING
17	**1021**	**401**	REQUEST_EXPIRED
18	**1022**	**401**	REPLAY_REQUEST
19	**1023**	**401**	INVALID_SIGNATURE or RATE_LIMIT_EXCEEDED

20	**1024**	**401**	IP_BLOCKED
21		**429**	Too many request in a short time/Rate Limit Exceed (300 request per minute)
22		**500**	Internal Server Error
23		502	Bad Gateway
24		**503**	Service Unavailable
25		**504**	Request timeout

10.Important Points

- **Entity ID**, **Mailer ID**, and **Certificate ID** must match the shared lists. Updates will be provided for any new additions.
- **Source Field**:

- 0 → Mobile Number is the reward source.
- 1 → Email Address is the reward source.

- **Notification Flags**:

- isSMS, isWhatsApp, isEmail:
- 1 → Notification will be sent.
- 0 → Notification will not be sent.
- Fields marked with * are **mandatory**.
- If Source is 0, the mandatory fields are: Sno., User Name, Country Code, Mobile Number, Reward Amount, Entity ID, Transaction ID.
- If Source is 1, the mandatory fields are: Sno., User Name, Email Address, Mailer, Reward Amount, Entity ID, Transaction ID.
- The following fields are optional: Personal Message, Message From, CC Email Address, BCC Email Address, Reference, Certificate ID, Column1 to Column5.
- If the client requires custom columns, the custom column names must be defined. Once defined, these custom columns become mandatory in the data submission.
- Transaction Id is mandatory and it should be unique.
- If error code **1012** is returned, the transaction is accepted but rewards are **not distributed** due to insufficient balance. Once balance is maintained, rewards can be approved from the **Benepik Admin Panel**.

11. Important Integration Notes

- All requests are allowed only from whitelisted IP addresses.
- Requests originating from non-whitelisted IP addresses will be automatically blocked.
- Reward transactions must contain unique Transaction Id.
- Mailer ID and Entity ID must match Benepik-provided lists.
- Notification flags (SMS, WhatsApp, Email) operate independently.
- For API testing, UAT and Production keys remain separate but function identically.
- The authentication credentials (authKey and secretKey) will be rotated annually, and may be rotated earlier if required in the event of a security incident.
- The API for generating authKey and secretKey will be provided in later part.