

LIVEPLAYTM

Live Play Service Player Rewards Technical Guide

Author: Fred Ring
fred.r@liveplaymobile.com

This document is intended for a technical audience looking to integrate with Live Play Services (“LPS”) player rewards.

Last Updated: 2026-02-16

Table of Contents

Table of Contents.....	1
Overview.....	2
LPS Rewards Pull.....	2
Pull Confirmations.....	3
Pull Frequency.....	3
Pull Integration Steps Summary.....	3
LPS Rewards Push.....	3
Your Endpoint API.....	4
Push Confirmations.....	4
Push Integration Steps Summary.....	4

Overview

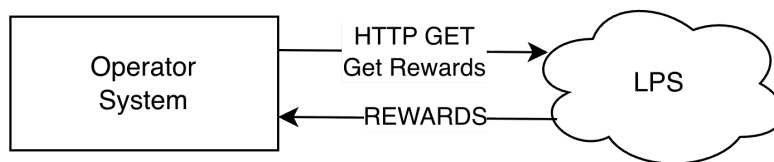
If you are running an LPS live-hosted game on your platform then you will want to give your players a chance to win the bonuses and leaderboard rewards that are given to players during the course of a LPS show. LPS provides two alternative methods to allow for this: a **pull** and a **push**.

You will implement one of the following two options, both of which are described in the guide below:

- **The pull** is simplest. We give you a URL and an API Key, and you can use any scripting language to get a list of pending rewards for processing to your players. See below.
- **The push** requires that you provide LPS an endpoint for the delivery of a list of player rewards to you for processing.

LPS Rewards Pull

This method allows the operator to pull rewards granted to their LPS players, using an HTTPS GET endpoint. The response contains a simple JSON list that contains all the pending rewards for your players.



Your system calls an HTTP GET endpoint with an API Key that LPS provides. The response contains a list of your player IDs, the amount of reward, currency of reward, the reward's date/time (sent as UTC), and a reason why the reward was given. Example of a response...

```
{ "rewards": [  
  { "operatorPlayerId": "<ID>", "amount": 20, "currency": "SC",  
    "rewardedOn": "2026-01-26 12:11:00", "reason": "Bonus Winner"},  
  { "operatorPlayerId": "<ID>", "amount": 100, "currency": "SC",  
    "rewardedOn": "2026-01-26 12:58:00", "reason": "leaderboard #1"},  
  { "operatorPlayerId": "<ID>", "amount": 50, "currency": "SC",  
    "rewardedOn": "2026-01-26 12:58:00", "reason": "leaderboard #2"},  
  { "operatorPlayerId": "<ID>", "amount": 25, "currency": "SC",  
    "rewardedOn": "2026-01-26 12:58:00", "reason": "leaderboard #3"}  
]}
```

The HTTP headers in your request must include **x-api-key** with your LPS-assigned API key. Requests without a valid key are rejected.

All pull requests must come from a known IP address. LPS needs to know one or more IP addresses where your requests originate from. Requests from unknown IP addresses are rejected.

The endpoint also requires that you pass an `operator_id`. LPS assigns all operators a UUID. As this is a GET request you include that in the URL...

```
https:<endpoint>?operator_id=<UUID>
```

Pull Confirmations

Each reward that is included in a pull response is marked as confirmed within LPS. Only unconfirmed rewards are included in responses so this ensures a reward is delivered only once. If you are simply testing the endpoint and want to get the same records with each request then add `test_response=1` to your request.

```
https:<endpoint>?operator_id=<UUID>&test_response=1
```

That returns a canned response that never changes. The rewards are dummy data and intended only for you to test the pull response.

Pull Frequency

For a good player experience, we recommend a pull every minute. This keeps players happy as their rewards arrive within a minute or so of the reward being given. The slowest we recommend is daily. LPS issues a large number of rewards every day and it is good for everyone to move rewards from *pending* to *confirmed* as quickly as is reasonable.

Avoiding Duplicates (idempotency)

You will receive each pulled reward only once. LPS marks each reward as delivery *confirmed* upon it being transmitted to your requesting system. This eliminates duplication of rewards using the pull API.

Currency

By default the currency is specified as "SC", but LPS can configure the currency sent with each reward.

Pull Integration Steps Summary

- Request an API key and URL from LPS
- Your team creates a script to pull the list of player rewards.. The LPS response is in a simple JSON list as outlined above.
- Send the LPS team one or more IP addresses that your pull requests will originate from.
- *Optional: LPS works with you to add additional player data to pass to your platform in the json*

LPS Rewards Push

Using this method, LPS pushes rewards to an endpoint that your team creates. Your endpoint puts the rewards into players' accounts. LPS personnel and processes use the endpoint to send rewards to your players.



Your Endpoint API

You are free to build anything that suits your business requirements. The method of delivery, formatting of data, and security mechanisms are up to you. LPS will adapt to your API.

We recommend a simple HTTPS REST API with a security mechanism, like IP restriction or an API Key or a hashed value based on a private key value. LPS will call your API programmatically so it should not require human interaction to operate.

JSON format for sending/receiving is generally best, but will adapt if you have other requirements.

Push Confirmations

How does LPS know that a reward needs to be sent to your players? Each reward starts life in a *pending* state. Those rewards await transmission to your endpoint. LPS uses your endpoint's response to determine what to do next.

- If your endpoint response indicates “success”, LPS marks the reward as confirmed and it will not be sent again.
- If your endpoint response indicates “failure”, LPS marks the reward as such and it will not be sent again. If you include a reason then LPS retains that reason permanently.
- If your endpoint response indicates “retry”, LPS will try again later. In this case we strongly recommend including a reason so LPS has a guide for retrying. LPS limits the number of retries. If the limit is exceeded then rewards are marked as “failure” and will not be sent again.

Avoiding Duplicates (idempotency)

LPS marks each successfully pushed reward as delivery *confirmed* upon it being successfully received by your endpoint. This eliminates duplication of rewards using the push API. LPS also records the date/time of each push attempt and any returned text so errors and confirmation codes are stored along with each reward. In the event of issues delivering a reward, it can be retried at a later time.

Message Formatting

LPS conforms to your endpoint's specifications for data format. You dictate the security, method of access, and the content of each endpoint call.

Push Integration Steps Summary

- Your team creates an endpoint that accepts LPS rewards, giving them to your players.

- You provide the endpoint, data formatting, and any authentication keys/processes to LPS.
- If required, LPS supplies you with IP addresses where our push calls will originate from.
- *Optional: LPS works with you and the game developer to add additional player data to pass to your platform when calling the endpoint.*