Loyal Patron API

Updated August 10th, 2020

Author: Justin Souter, Chief Software Architect
Email: justin@loyalpatron.com

Contents

Introduction

This document describes the Loyal Patron API. It is intended for software developers who are implementing list building, gift card and/or loyalty marketing features as strategic partners and/or merchants subscribed to a SaaS subscription. The interface can be used by point-of-sale systems, shopping cart software, mobile apps, web apps, and server software to enable gift card / loyalty (rewards) transaction processing and/or provide marketing automation experiences.

The API facilitates these functions for partner applications:

- Registration of new patrons (signup)
- Changing information for existing patrons
- Retrieving merchant registration configuration
- Accessing patron transaction history
- Authorizing a patron for app access
- Retrieving a list of merchants associated with a patron
- Retrieving memberships information
- Processing gift card transactions
- Processing loyalty (rewards) stored value transactions

Program Concepts

Stored Value Balance Types

The Loyal Patron gift and loyalty marketing system can track three separate stored value balances on a single account. A merchant can be configured to use any or all of these balance types:

Gift Balance: The current balance of gift value (in real dollars) that a patron has available to redeem toward their bill just as if they were spending cash. Gift value can be redeemed by patrons immediately after loading through the terminal.

Reward Balance: The current balance of rewards in cash value ($X.XX) available in the patron's account. By default, rewards can only be redeemed if the account has been activated. The concept of activation means the patron completed the signup process successfully (registration). This default setting can be turned OFF upon request which allows rewards redemption regardless of whether patrons complete registration.

Units/Comps Balance: The current balance of units (aka "comps") available in the patron's account. The Units/Comps balance is a prepaid value that tracks and redeems units of products or services in single integer format (10, 4, 1, etc.) as opposed to the currency $X.XX format used by the gift and rewards values. Examples of units are 'Games' (for the bowling or entertainment industry), 'Appetizers' (for restaurants), or 'Car Washes' (for auto detailing). The comp name is merchant configurable.

The use of units/comps is optional and may not be possible or necessary to implement across all terminal platforms based on unique system architecture.

Rewards Program Types

For merchants who desire running a daily rewards program that tracks spending on day to day sales there are two types of daily rewards structures available:

Instant Cash Back Rewards (Percent Method): This method enables merchants to reward patrons with a stated percentage (i.e., 10%) of the sales remittance amount on every sale. For example, assuming a 10% reward, a customer spending $55.00 on dinner would receive $5.50 back in rewards. A customer spending $1.00 would receive $0.10 added to their reward balance. The reward ratio is merchant configurable in the merchant back office.

Milestone Rewards (Threshold Method): The second type of reward program structure is a single level milestone program requiring the patron to spend X dollars before earning Y reward. For example, a patron may have to spend $100.00 in order to earn a $15.00 reward. Upon achieving the $100 spending threshold, the Amount to Reward counter resets to 0 and starts over counting to the spending threshold. Both the spending threshold and triggered reward is merchant configurable in the merchant back office.

NOTE: Loyal Patron is **not** a points-based loyalty system, where patrons earn one or more points for each purchase. All rewards are « cash-back » in Loyal Patron, where patrons always know the dollar value of rewards earned.

Transaction Disclosure to Patron

After completing a transaction, the patron is typically presented with a paper receipt that discloses their after-transaction account balances (as described above). The same patron also receives an automated thank you note sent by email or text message confirming latest reward balances. In addition, these values are also disclosed:

Transaction Amount: The Transaction Amount is the amount entered on the terminal for a given transaction type. This value could be the sale amount (if a sale transaction) or the load amount (for load gift or load reward transactions). For example, a $10 Remittance would have a transaction amount of $10. A $5 redemption would have a transaction amount of $5. A $3 Load Gift would have a transaction amount of $3. An $18 purchase which is paid for with $5 gift, $3 reward and $10 cash would be a Redeem Gift with a Transaction amount of $5, a Redeem Reward with Transaction Amount of $3 and a Remittance with a Transaction Amount of $10

Remitted Amount: The Remitted amount represents the cash that needs to actually be collected by the clerk or employee during the sales transaction when redemptions are processed. Since the redemption amount can be different (less then) then sales amount, the difference between the sales amount and the redemption amount is the remitted amount. For example, if a customer rings up a $20 sales ticket and wants to

use $5.00 reward available on their account to put toward the $20 purchase, then the remitted amount is $20.00 sale LESS $5.00 redemption = $15.00. The Remitted Amount field appears on transaction receipts whenever a redemption of value (gift and/or rewards) is processed by the clerk.

Note : Rewards are only earned on the remitted amount (new money spent).

Amount to Reward (*Milestone Reward Programs Only*): The Amount to Reward field is only applicable for merchants who run a milestone rewards program (threshold method) where patrons receive a reward only when a specified spending milestone, or threshold, is attained by the customer. A customer earning a $10 reward for every $100 spent is an example of a milestone rewards configuration. The Amount to Reward field keeps a running tab on how much more money the customer must spend in order to receive their reward. Once the reward is attained, the Amount to Reward field resets back to 0 and starts counting backwards towards the next milestone.

Terminal Transaction Types

The stored-value transaction types supported via the API are:

- **Sale:** This function is the most commonly used transaction to record sales activity by your patrons to issue rewards. When a patron identifies their account during a sale transaction, you must record the exact amount of the sale transaction (with or without tax). We recommend post tax amount for clarity, transparency, and ease of reconciliation noting the reward ratio in the merchant back office can always be adjusted downward to reflect sales tax, if desired by the merchant.

  The redemption amount cannot be greater than the sale amount and of course the redemption amount cannot be greater than Gift Balance + Reward Balance combined. If a customer has value in BOTH their Gift and Rewards balances, the Gift Balance is withdrawn first followed by the Reward Balance (though this is configurable). If a redemption amount entered is greater than the sale amount, the host will return an error. The Redeem All option can be specified if the patron wishes to apply whatever gift and reward balance they have available to the sale.

- **Redeem Units/Comps:** Redemption of units/comps is allowed without a sale taking place. It is a separate operation, unlike the redemption of rewards or gift card balances. When units/comps are redeemed the units balance for the patron is decreased. A typical real world example would be running a Redeem Dessert transaction of 1 that reduces a customer's 'Dessert' Balance from 1 to 0. The 1 dessert could have been loaded by the system for the customer's birthday or anniversary, for example, which expires in 30 days if unredeemed.

  Units/comps redemption might not be honored if the merchant configuration has set a limit on the maximum redemption in a given time period (e.g., 1 comp

redemption allowed per week(. In this case an error message is returned in the response.

- **Load Units/Comps:** Used to load units/comps of products or services to a patron's account in integer format (1, 2, 3, 10 etc.), causing the account's units balance to increase. In most cases, comps are loaded automatically by the system based on an automated campaign such as a birthday, anniversary or holiday bonus. However, this transaction supports the manual loading of comps by clerks at the terminal.

- **Load Gifts:** This function is used to add gift value to a account when a customer provides money in advance (cash or credit card). This transaction is also known as a "prepaid load". The word "load" simply refers to adding the money to the account. Customers can add value to their account in advance or at any time even if a current balance still exists. Customers can then redeem this value over time to pay for products and services directly from their account rather than using another form of payment.

- **Load Rewards**: This transaction is used to manually load rewards to patron accounts outside of the normal channel of earning rewards from Sale transactions. This transaction type is ideal for adding dollar amounts for contests, resolving customer complaints, or any type of behavior where it does not make sense to ring up a sale. A negative amount may also be specified to make an adjustment to reduce the patron's reward balance. A negative load should NOT be used to void/credit a transaction. Use the provided credit function instead.

- **Membership**: Merchants can define membership classes to signify benefit levels for patrons. This function is typically used to upgrade a patron with special privileges not available to most loyalty members. An app may initiate, renew, and cancel these memberships using the membership transaction type. The membership_action request field should be set to "sale" or "cancel" with the sale action used for both new membership sales and renewals. The memb_class_name request field must contain the name of the membership class to assign to the patron, as configured in the merchant web application. All membership value and benefits are configured in the merchant back office.

- **Credit:** Any sale, gift card or loyalty transaction can be credited, using the "credit" type in the term_host service request. Transactions may be credited using a specific transaction id, or by default the most recent transaction will be credited if no transaction id is specified. An amount to be credited can be specified in the amount field, or left blank to credit (void) the entire transaction amount.

Fundraising Transactions (Optional)

Patron accounts can be optionally tied to fundraising organizations the merchant has agreed to donate to in their local community. If a transaction generates a donation, additional information will be returned in the response: the fundraiser field will be set to

the name of the fundraising organization, the donation amount will contain a non-zero amount, and the pre-formatted receipt will show a fundraising footer (with organization name and donation amount). Fundraising affiliation is performed inside the merchant back office and is not a terminal transaction type.

Patron Account Activation

The concept of activation noted by the **Activated** field on the printed receipt (Yes or No) means a patron completed the signup process successfully from any list building source (Text to Join, Online via website, Kiosk, WiFi etc.) Loyal Patron supports. The activation process is designed to collect specific marketing data from patrons such as their name, email address, birthday and mobile phone number. When rewards or units/comps are preloaded or earned by patrons during sale transactions the system allows immediate redemption of the reward/comp as long as the patron's account has been activated. Without activation, rewards/comps can be accumulated but cannot be redeemed by default. This default setting can be turned OFF by merchants in their merchant back office, which would allow rewards and/or comps to be redeemable regardless of patron activation status.

Patron Account Identification

Patrons can be identified in the system by any of the following means with or without plastic cards:

- The patron may present their physical gift/loyalty card to be swiped.
- Clerks can hand-key the digits on the back of the physical card (which is necessary when the mag stripe encoding is damaged for some reason).
- Alternatively, and increasingly more commonplace, patrons can identify their account using their 10-digit mobile phone number. The API allows identification by a mobile number either with or without dashes ("-").

If more than one card is associated with a particular mobile phone number, the API will attempt to identify the account with the processing merchant. If the API can't do this then the transaction will not be processed and an error message is returned. Unregistered cards that have not been activated cannot be processed by mobile phone number since no phone number is on file.

Patron Account Login and Balance Checking

Merchants can elect to implement some basic HTML code on their site and enable patrons to access their website to login and check reward balances, review transaction history, and update their profile/privacy settings from the same login session. Loyal Patron provides a Web Tools Guide that is available to merchants inside their back office account. This instructional guide is typically performed by the merchant's webmaster. You can review an example of the Web Tools Guide at https://app.loyalpatron.com/acgpl/content/website_tools_guide.php?p=4&m=1960

Identifying the Processing Agent/Clerk/Employee

If desired and if supported by the terminal/app platform, transactions can be identified to the processing agent, or clerk. Clerks are set up in the merchant web application, by name and clerk id. A merchant can specify if a clerk id is required to process transactions, and if none is provided in the transaction request then the request will indicate an error. Otherwise the clerk id provided in the request is matched against the list of valid clerks. Of course, if clerks are not configured then this field in the request can be omitted.

For most POS system integrations, we turn on the Require Clerk ID checkbox during merchant configurations, which automatically pulls valid POS system IDs into the Loyal Patron system. Authorized merchant back office users can then edit the Clerk ID table in their back office and assign employee names to help identify which staff member processed which transaction for audit trail reporting and reconciliation.

Example Clerk ID Table in Merchant Back Office

Name↑	Clerk ID↑↓	Location↑↓
Adam Miller	PGQREPCRHY8QA	Newtown
Bill Prady	1069	Voorhees
Connie	5421	Newtown
Fan Test	7890	Voorhees
Maria	4587	Newtown

For additional program concepts and helpful HOW TO articles of various features, please browse our Knowledge Base at https://www.onboardingtutorials.com/

Accessing the API

The Loyal Patron API is accessed via HTTP endpoints using the POST request method. Parameters are sent as URL-encoded POST variables. The endpoint will respond in JSON format according to the request.

Endpoints

There are separate HTTPS endpoints for test and production. The test endpoint will return access errors and validation errors, whereas the production endpoint will fail silently on access errors, but still return validation errors.

Patron account information changes and transactions sent to the test endpoint will not affect production data. You will be provided with a test merchant account to facilitate development.

Test endpoint: https://app.loyalpatron.com/acgpl_test/api/

Production endpoint: https://app.loyalpatron.com/acgpl/api/

Append the API service name (like merchant_options or patron_profile) to the endpoint.

Parameters

Merchant id or Terminal id: Most of the API calls can be made using either a merchant id or a terminal id, for apps deployed in point-of-sale devices.

API Key: Your application/integration will be assigned an API key when you begin development. Use this key in all your API calls.

Account: Card id, cardless account id, or mobile phone number to identify account. Field with be stripped of punctuation and leading zeros (helpful for passing raw track data from swiped card at POS terminal).

Other parameters are show below along with each service.

Your app should implement timeout detection in case the network connection is lost. We recommend a 20 second timeout value.

Service: Merchant’s Configuration Options

Service Name: merchant_options

Obtains information useful to present the best experience to patrons. Merchants can configure required fields, custom field name, registration groups, and activation options from the LP merchant web app. This service retrieves that information.

If patrons will be registering physical cards (from card ranges configured in the Loyal Patron extranet), then initially only the card id field should be presented to the patron. Once that is collected the merchant_options call can be made to get the setup information for that particular card’s range.

Request

key - API key.

merchant / terminal - Id of requesting merchant or id of the requesting terminal.

card_id - Physical card id to retrieve card-specific information (optional).

Response

status - Set to "ok" or "error".

error - Error message (only present if status is "error").

hidden - Array of field names for hidden fields.

optional - Array of field names for optional fields.

required - Array of field names for required fields.

reg_group_name - Field title/label of the registration group field. If empty do not present reg group field.

reg_groups - Array of registration group names. If present in response show multi-select list of reg group names. If absent from response show checkbox or yes/no selection.

custom_field_name - Field title/label of the custom patron profile field.

bonuses - Array of bonuses and reward rates available for activation, birthday, purchases, and many other merchant-configuration.

units_name - Merchant’s configured units name.

use_rewards - Merchant uses rewards (1 or 0).

use_gifts - Merchant uses gifts (1 or 0).

use_units - Merchant uses units (1 or 0).

units_text - Name for units/comps field.

use_load_rewards - Merchant allows loading of rewards (1 or 0).

use_load_units - Merchant allows loading of units (1 or 0).

use_memb_class - Merchant uses membership classes (1 or 0).

use_milestone - Merchant uses milestone rewards type (1 or 0).

name - Merchant name.

short_name - Merchant short name.

Service: Patron Profile

Service Name: patron_profile

Verifies patron information and possibly registers patron. May send email or text message to patron to request activation of patron account.

Request

key - API key.

action - One of “register” (to register new card or account), “modify” (to change information for already registered account), or “get” (to get current profile information for account).

merchant / terminal - Id of requesting merchant or id of the requesting terminal.

validate_only - Set to 1 to NOT save the submitted data. Validation will be performed only and field messages will be returned. Valid for register or modify actions.

include_balances - Set to 1 to include balances and other account information for the specified merchant and account.

account - Card id, cardless account id, or mobile phone number to identify account. Only used for get action.

patron - Internal patron account id. Only used for modify action. Use the value of the id field received via the get action to identify the patron to modify.

The remaining request fields are used for register and modify actions:

firstname - First name.

lastname - Last name

password - Account password. Can be pre-verified against an identical entry.

password_v - If password is not pre-verified then it will be matched against this and an error return if not matched with password field.

card_id - Physical card id. If blank a cardless account will be created. Register action only.

email - Email address. Will be validated through external service.

address - Street address.

address2 - Continuation of street address, if needed.

city - City.

state - State or province.

zip - Zip code or postal code.

mobile - Mobile number.

gender - Gender (M, F, or blank).

custom_field - If merchant has configured a custom field. Labeled with custom_field_name from the merchant_options response.

birthday - Birthday, as MMDD or MM-DD.

birth_year - Four digit birth year.

annivdate - Anniversary, as MMDD or MM-DD.

in_reg_group - If registration groups are configured for merchant, this should be 1 or 0 depending on whether patron has selected the yes/no group setting.

activate_meth - Activation method (em = email, mt = mobile terminated, mo = mobile originated, ag = agent). For em and mt an activation message will be sent to the patron. For mo the account will be assumed activated.

activated - Account activation status (1 or 0).

Response

status - Set to "ok" or "error".

error - Error message (only present if status is "error").

messages - Keyed array of errors, one for each field that triggered a validation error. Omitted if request passed validation. Only will be returned for register or modify actions.

patron - Internal patron id. Only returned for get action.

profile - Patron profile data. Only returned for get action.

id - Patron id (use in modify call).
firstname - First name.
lastname - Last name

card_id - Physical card id.
email - Email address.
address - Street address.
address2 - Continuation of address.
city - City.
state - State or province.
zip - Zip code or postal code.
mobile - Mobile number.
gender - Gender (M, F, or blank).
custom_field - If merchant has configured a custom field. Labeled with custom_field_name from the patron_options response.
birthday - Birthday, as MMDD or MM-DD.
birth_year - Four digit birth year.
anniversary - Anniversary, as MMDD or MM-DD.
in_reg_group - If registration groups are not used, this should be 1 or 0 depending on whether patron has selected the yes/no group setting.
activate_meth - Activation method (em = email, mt = mobile terminated, mo = mobile originated, ag = agent).
activated - Account activation status (1 or 0).

balances - Patron balances data. Only returned for get action when include_balances is set.

total_sales
reward_bal
reward_total
gift_bal
gift_total
units_bal
units_total
amount_to_milestone
trans_count
visit_count
memb_class_id
memb_class_name
memb_orig_date
memb_expire_date
memb_status
fundraiser_name
fundraiser_rate
fundraiser_total
units_name
last_visit
created
init_gift
init_reward
init_units

Service: Patron Account Authorization

Service Name: patron_auth

Provides account access authorization for patrons.

Request

key - API key.

account - Account number, card id, mobile number, or email of patron.

merchant / terminal - Id of requesting merchant or id of the requesting terminal.

password - MD5 encoded password.

Response

status - Set to "ok" or "error".

error - Error message (only present if status is "error").

accounts - Array of accounts, in case more than one match email/mobile. If this is present ask patron which account to use. If zero accounts returned then account/password set not matched.

Service: Merchants Associated with Account

Service Name: account_merchants

Retrieves information on all merchants associated with a patron account. Also used to create a new association between an existing patron account and a merchant. This feature enables the ability for the same patron to be affiliated with various merchant accounts without duplicating patron records.

Request

key - API key.

account - Account number.

action – “get” or “set” (assumes get if omitted)

merchant – Merchant to associate with account (set action only).

Response

status - Set to "ok" or "error".

error - Error message (only present if status is "error").

total_visits - Total number of visits for the account for all associated merchants.

total_merchants - Number of merchants associated with this account.

patron - Information identifying patron account:
name
email
mobile
register_date

merchants - Array of merchants objects:

id - Merchant’s id.
name - Merchant’s name.
short_name - Short version of merchant name.

address - Address of merchant location most commonly used by patron.
map_link - Map reference for merchant location (to open maps app).
phone
web

reward_balance - Patron’s reward balance with merchant.

gift_balance - Patron’s gift balance with merchant.
unit_balance - Patron’s unit balance with merchant.

use_rewards - Whether merchant uses rewards (1 or 0).
use_gifts - Whether merchant uses gifts (1 or 0).
use_units - Whether merchant uses units (1 or 0).
units_name - Merchant’s configured units name.

last_visit - Patron’s last visit to this merchant.
visit_count - Total number of visits this patron has made to merchant.

Service: Get Account Transactions

Service Name: transactions

Request

key - API key.

account - Account number.

start - Index of first transaction (1 based).

count - Count of transactions to return. Set to zero or omit to return all.

merchant / terminal - Id of requesting merchant or id of the requesting terminal.

Response

status - Set to "ok" or "error".

error - Error message (only present if status is "error").

total_transactions - Total number of transactions available. May be greater than the count of elements in transactions array depending on the count field in the request.

transactions - Array of transaction objects:

date_time
type
merchant_id
merchant_name
amount
reward
reward_balance
gift_balance
unit_balance
units_name
note

Service: Patron Account Transfer

Service Name: account_transfer

Transfers all gift card, comp and/or rewards value from one card account ID# to another. This feature is used for programs that distribute plastic mag stripe or bar coded cards to patrons.

Request

key - API key.

merchant / terminal - Id of requesting merchant or id of the requesting terminal.

account_from - Account to transfer from.

account_to - Account to transfer to.

profile - Which patron profile to keep (“from” or “to”).

Response

status - Set to "ok" or "error".

error - Error message (only present if status is "error").

log - Array of transfer processing log entries.

Service: Patron Memberships

Service Name: membership

Enables operators to execute a Membership Buy (or Cancel) transaction that classifies a patron into a membership class for tiered benefits processing or separate reporting purposes. For example, a clerk may upgrade a patron to GOLD status which increases patron's reward ratio from 5% to 10% and gives member $10 reward on their birthday instead of $5 (for non-GOLD members).

Request

key - API key.

action - Membership action. Currently only "list" is available.

merchant / terminal - Id of requesting merchant or id of the requesting terminal.

Response

status - Set to "ok" or "error".

error - Error message (only present if status is "error").

memberships - Array of membership classes.

id
name
short_name
description
price
renew_price
expiration
expire_date
cart_sell
retail_value

Service: Transaction Processing

Service Name: term_host

This service accesses the high-performance transaction processing host, but through a convenient HTTP API.

Note: A direct, socket-based terminal host access option is available as well. Please contact Loyal Patron if you believe this option may be best for your implementation.

Request

key - Authentication key to validate client transaction. Will be provided by Loyal Patron for use in your application.

dry_run - Set to 1 to perform a "dry run" of the transaction, where no updates to the permanent data will take place. Set to 0 to actually record the transaction. This can be used to provide confirmation of the transaction to the user before finalizing the transaction.

type - The transaction type, either sale, redeem_units, load_units, load_gifts, load_rewards, detail_report, summary_report, membership, or credit.

account - Card number, account id, or phone number to identify patron.

terminal - The terminal number of the initiating terminal. Required.

clerk - The clerk number for this location. Optional. If present, it must have been previously configured via the merchant web application (unless the terminal type supports auto-generation of new clerk accounts when new clerk id encountered).

clerk_name - Clerk name. Optional. Only used if terminal supports auto-generation of clerks to set name associate with new clerk record.

trans_num - Terminal transaction number. Can be up to 24 characters. Typically in the format yymmddnnn, where yy is the two-digit year, mm is the two-digit month, dd is the two-digit day, and nnn is the three-digit transaction number for that day. Field is optional; a substitute will be created if none is supplied in the transaction.

amount - The amount of the transaction (dollars & cents, or units).

redeem - Amount of redemption (units for redeem_units type, or dollars & cents for sale).

redeem_all - Optional. Set to 1 to cause the maximum redemption to the transaction, ignoring the redeem amount if any is specified. Otherwise set to 0 or omit.

redeem_type - Optional. Controls the way value is redeemed for the transaction. Only valid for the sale type. Specify “gifts_first”, “rewards_first”, “gifts_only”, or “rewards_only.” If omitted the default is gifts first.

notes - Optional notes associated with transaction (ignored for credit or reports or membership).

transaction_id - Identifies a transaction to credit (along with account). If omitted or blank then the last transaction processed for the specified account will be credited.

membership_action - Identifies whether selling/renewing a membership (set to “sale”) or canceling a membership (set to “cancel”). Only used with the membership transaction type.

memb_class_name - Identifies the merchant’s membership class name to sell or renew for membership transactions.

days - Specifies the number of days of transaction history to include for summary and detail reports.

display -Set to 1 or 0 to control whether a pre-formatted display string should be returned. Optional.

receipt - Set to 1 or 0 to control whether a pre-formatted receipt string should be returned. Optional.

receipt_newline - Optional string to specify new line to terminal. Can include ANSI-C escape sequences. Defaults to hex A (linefeed). If “<fill>” is specified, no newline will be inserted. Instead, each line will be filled with spaces out to receipt_width.

receipt_newpage - Optional string to specify new page or form feed to terminal. Can include ANSI-C escape sequences. Defaults to hex C (formfeed).

receipt_width - Optional number of character columns for receipt return.

display_width - Optional number of character columns for display return.

display_newline - Optional string to specify new line to terminal. Can include ANSI-C escape sequences. Defaults to hex A (linefeed). If “<fill>” is specified, no newline will be inserted. Instead, each line will be filled with spaces out to display_width.

display_newpage - Optional string to specify new page or form feed to terminal. Can include ANSI-C escape sequences. Defaults to hex C (formfeed).

wide_print -Optional string to specify wide printing to terminal for receipt. Can include ANSI-C escape sequences. Defaults to unsupported.

normal_print - Optional string to specify normal printing to terminal for receipt. Can include ANSI-C escape sequences. Defaults to unsupported.

Response

host_version - Software version of the terminal host.

patron - Name of the patron.

type - Transaction type echoed from the request.

trans_desc - Transaction description for display to user on receipt and/or terminal screen. Empty for sale transaction.

trans_amount - Value amount for the transaction; reflects the amount field in the request. Same as sale_amount for sale transaction type.

redeem_amount - Total amount that was redeemed. May be different from the amount in the redeem field in the request, such as when redeem all is specified. Only useful for sale transaction type.

sale_amount - Total amount of sale. Only useful for sale transaction type.

remitted_amount - The amount patron remitted for this transaction, which is the sale amount less any redemptions. Used for sale transaction type to show what portion of the sale must be remitted. Used for credit transaction type to show what portion of the sale must be refunded (will be a negative number).

amt_is_units - Will be 1 if sale or redeemed amounts are integer (units). Otherwise 0.

redeemed_gifts - The amount of the patron's gift balance that was redeemed.

redeemed_rewards - The amount of the patron's reward balance that was redeemed. Empty if nothing was redeemed.

earned_rewards - Amount of rewards earned for this transaction. Empty if nothing was earned.

to_milestone - If merchant is using milestone rewards, this is the amount the patron must remit to earn the milestone reward.

milestone_reward - The reward amount that will be earned at the next milestone. If field is missing or evaluates to integer zero then do not display to_milestone or milestone_reward fields.

initial_gift, initial_reward, initial_units - Initial grants if this is the first time the account is used and initial grants are available.

merchant_id - The merchant id of the processing merchant.

merchant_name - The name of the processing merchant.

fundraiser - Will be set to the name of the fundraising organization if the transaction is related to one. Otherwise empty.

donation - Donation amount made to fundraiser for this transaction. Do not display if fundraiser field is blank.

trans_datetime - The date and time of the transaction, in m/d/y h:m:s p format, where y is the four-digit year and time is in 12-hour formation, and p is the am/pm indicator.

account - Patron's account id, reflected from request, or looked up from phone number if supplied in request.

activated - Will be 1 if account is activated, otherwise 0.

gift_balance, reward_balance, unit_balance - Balances for gifts, rewards, and units after the transaction is complete (or as if the transaction would have completed in the case of a dry run).

using_gifts, using_rewards, using_units - Indicates if the merchant is using gifts, rewards, and units. Can be used to hide or show balances or other information related to gifts, rewards, or units.

footer1, footer2 - Receipt footers specified by merchant or location.

program_domain - Private label domain name.

transaction_id - Transaction identifier for this transaction, identical to term_trans_num in the request, if provided. Use this to display identifying information on patron receipt, and to identify transaction for crediting.

units_name - Term used by this merchant to label units/comps, such as "Games."

display - Formatted response for display.

receipt - Formatted response for receipt. Maximum size of receipt data is 2k bytes.

error - Error message if transaction failed. Not present if transaction was successful.

memb_class - Name of membership class, if any.

memb_status - Status of membership, if any.

Sample Terminal Receipt

The example printed receipt below displays a fully implemented receipt with all three (3) stored value balances: (Gift Balance, Reward Balance and Comp/Unit Balance aka ‘Small Yogurt’). In addition, this receipt displays Membership status if the optional Memberships module is enabled. Furthermore a fundraising donation accrual message is added as a footer if the merchant has the optional Fundraising module enabled.

Receipt Layout from Transaction Response Fields

Field Name	Description	Sample Data	Format	Example		
patron	the patron name	John Buck	center bold	**John Buck**		
trans_desc	description of transaction; omit if empty (sale transaction)	Got $40.00 gift	center bold	**Got $40.00 gift.**		
earned_rewards	if not blank or “0” / “0.00” show as "Earned $<earned_rewards>"	10	center bold	**EARNED $2.00!**		
fundraiser	if not blank show as "Donated $<donation> to <fundraiser>"	Malibu School	center regular	Donated $2.00 to Malibu School		
to_milestone	if float value > 0 show as "<to_milestone> from <milestone_reward> reward”, otherwise omit this line	33.94	center regular	$42.00 from $10.00 reward		
account	patron loyalty account as “Loyalty Account <account>”	618838594	center regular	Loyalty Account 618838594		
transaction_id	show as "Loyalty Trans Id <transaction_id>	3141566713593	Spread	Loyalty Trans ID		3141566713593
activated	if 1 show “Yes”, if 0 show “NO”	1	Spread	Activated		Yes
redeem_amount	show as "Redeemed $<redeem_amount>"	0	Spread	Redeemed		$7.00
memb_class	if not blank show as "<memb_class> Member”, otherwise omit line		Spread	Membership		Beer Club

memb_status	if not blank show as "Member Status <memb_status>", otherwise omit line		Spread	Membership Status		Expires 6/20/2019
gift_balance	gift balance of account	0	Spread	Gift Bal		$234.50
reward_balance	reward balance of account	851.39	Spread	Rewards Bal		$22.00
unit_balance	unit balance of account	0	Spread	Members Choice Bal		23
footer1	use if not blank at bottom of receipt	Join us for Triple Rewards Mondays	center bold	**Join us for Triple Rewards Mondays**		
footer2	use if not blank at bottom of receipt		center regular	Visit Now!		
milestone_reward	see to_milestone	10				
donation	see fundraiser	4				
using_rewards	if 1 show reward balance	1	n/a			
using_gifts	if 1 show gift balance	1	n/a			
using_units	if 1 show units balance	1	n/a			
units_name	use behind units balance print; see unit_balance	Members Choice				
amt_is_units	if 1 show redeem amount as "Redeemed <redeem_amount> <units_name>"	0				

Terminal Implementation Checklist

1. Review program capabilities and decide on desired terminal/app features: gift value, reward value, units/comps value, milestone rewards, fundraising, memberships etc.

2. Decide if you will use the pre-formatted receipt, or if you are creating your own receipt from the response data fields.

3. Log into your Loyal Patron merchant back office (test branch) and create one or more patron accounts for testing.

4. Log into the Loyal Patron merchant back office (test branch) and click on Virtual Terminal. Walk through the terminal operations for the program features you will be implementing. Consider how these operations will be implemented on your terminal/app.

5. Verify basic communication using the ping request.

6. If your merchants will be using different features (one using gift value, another using memberships, etc), use conditional display by checking for valid field values or by checking response fields use_gifts, use_units, use_rewards.

7. Provide rewards redemption options during the terminal sale/remittance/tender phase. Usually a specific redeem amount may be entered, or a “redeem all” option selected. Use the later to set redeem_all in the transaction request.

Merchant Options Considerations:

Using gifts
- Provide load gift feature
- Show gift balance on receipt

Using rewards
- Provide adjust rewards feature
- Show rewards balance on receipt

Using milestone rewards
- Show “$X away from $Y reward” on receipt

Using fundraising
- Show donation and fundraiser on receipt.

Using memberships
- Show membership name and status on receipt.

Sample Transaction Response


```
{
        "host_version" : "33",
        "patron" : "Justin Souter”,
        "trans_amount" : "2.00",
        "type" : “load_units”,
        “trans_desc” : “Got 2 Games”,
        "units_name" : "Games",
        "redeem_amount" : "0.00",
        "sale_amount" : "2.00",
        "remitted_amount" : "0.00",
        "redeemed_gifts" : "0.00",
        "redeemed_rewards" : "0.00",
        "earned_rewards" : "0.00",
        "to_milestone" : "0.00",
        "milestone_reward" : "0",
        "initial_gift" : "0.00",
        "initial_reward" : "0.00",
        "initial_units" : "0",
        "merchant_id" : "997",
        "merchant_name" : "FK Test Merchant",
        "fundraiser" : "",
        "donation" : "",
        "trans_datetime" : "09/30/2010 06:44:57 PM",
        "account" : "401000010",
        "activated" : "0",
        "gift_balance" : "44.00",
        "reward_balance" : "10.00",
        "unit_balance" : "6",
        "using_rewards" : "1",
        "using_gifts" : "1",
        "using_units" : "1",
        "footer1" : "Thank you for visiting FK Test Merchant",
        "footer2" : "Check balances at www.finderskeeperscard.com",
        "program_domain" : "finderskeeperscard.com",
        "transaction_id" : "100930000",
        "amt_is_units" : "1"
}
```

Sample Code for API Access (C#)

```
namespace LoyalPatronPatronAPISample
{
   class Program
   {
      static void Main(string[] args)
      {
         Task.Run(() => MainAsync());
         Console.ReadLine();
      }

      const string ApiKey = “<your_api_key>”;
      const string Hostname = "https://app.loyalpatron.com/";
      const string ApiPath = “acgpl_test/api/";

      static async Task MainAsync()
      {
         const string service = "patron_profile";
         const string account = "618838459";

         using (var client = new HttpClient())
         {
            client.BaseAddress = new Uri(Hostname);
            var content = new FormUrlEncodedContent(new[]
            {
               new KeyValuePair<string, string>("key", ApiKey),
               new KeyValuePair<string, string>("account", account),
               new KeyValuePair<string, string>("action", "get")
            });

            var result = await client.PostAsync(ApiPath + service,
            content);
            string resultContent = await
            result.Content.ReadAsStringAsync();

            Console.WriteLine(resultContent);
         }
      }
   }
}
```