Item Transaction API

Extends API: Authenticated Monetization API
Up to date with: Authenticated Monetization API 0.2 and Monetization API 0.2.2

Current Version: 1.04

Summary

The **Item Transaction API** (abbreviated **ITA**) is intended to give a standard interface that allows services to credit and debit items for particular users in a particular game. The goal related to this API is for it to power:

- BAPI,
- GSAT,
- and monetization

That way, only one callback/api needs to be implemented by games (instead of 3 or 4). Another goal is for an implementation of this API to be built into each game-stack so that really this functionality will be automatically available for new games on one of the standard game-stacks.

The responder-end of the interface is intended to be implemented by games. The endpoint can be called by central-services, other games, or whoever else needs to transact on items for an external game. The endpoint that sends requests to the ITA will be called the **ITA Requester**.

- Blue items are standard parameters, and
- brown items are constants

Any field not marked (Optional) is required.

URL Convention

<protocol>://<host>/<optional-path>/itemTransaction/<version>

- <optional-path> - any path off the host can be used as long as the path ends with the correct conventions described above
- <version> - Value that only contains numbers and periods, corresponding to the version of the implementing spec it is built to (**not** the version of this API Interface).

Examples:

- http://mog-fb-qa.playdom.com/itemTransaction/1.0
- http://snf-qa01-external.vm.dfw.playdom.com/snc/public/labsgamethree/playdom/itemTransaction/2.1.3

Request

The HTTP method is POST and the request is UTF-8 encoded. All HTTP headers set by the request should be ignored.

Request Body

The request information is sent in the http request body. Note that the request body is **not** url-encoded and so there are no "get-" or "post-" parameters.

<securityHash>" "<jsonRequest>

where

- <securityHash> is a base-64-encoded SHA-1 HMAC hash of <jsonRequest> using <secret> as the hmac key. If the hash sent in the request doesn't match the hash calculated on the responder's end, the request should fail with <error-type> "unauthorized".
- <secret> is a shared secret unique between the "requester" and the endpoint url. The secret should be agreed upon between the team implementing the requester and the team implementing the responder (future versions of this API will hopefully obtain the secret using the DCOP auth service).** Monetization - when interacting with monetization, QA will have the secret "QA_secret_key" and production will have some other key created by monetization.
- " " is a space
- <jsonRequest> is a JSON string holding the request information.

<jsonRequest>

key	value
"system":	The name of the system, service, or program that is making the request. This can be created by whoever is writing the requester end. (When the auth service becomes available, this should probably be the same as the identifier used by the auth service, if the auth service uses something like that.)
"requester":	Who is making the request. E.g. "monetization" or "btetrud" . • If it is the service itself making the request, please use the same value as the is put for the "system" key. • If it is an individual, use a username of some kind - probably an ldap username.
"comment":	(Optional) Some description as to why the request is being made.
"info":	(Optional) JSON object that contains extra information about the request. This is the only place where custom parameters are added in the top-level request object; all undefined keys are reserved for future use.
"t":	Number of seconds from the epoch (unix timestamp in seconds) at the time this message was constructed.
"idOrigin":	A namespace for the "id" parameter. An example of this is the 3rd party that the "id" came from. • If the idOrigin is the system itself, use the same value as is put for the "system" key (Monetization API).
"id":	A unique identifier for this request. Used to dedupe requests if duplicates are sent out. Should be unique within the particular idOrigin given.
"network":	The namespace for the "user" parameter. This will be the BI 'network' tag (as opposed to the BI 'view_network').
"user":	The user ID of the user these item transactions will be made for. Plea se ensure that this parameter accepts both string and integer values.
"items":	A json-array where each element is a <itemObject>.

<itemObject>

<itemObject> describes one item to transact. It is an object with the following members:

key	value
"category":	The category of items this particular item falls in (the namespace for the item's "id" value).
"id":	The identifier of the item to transact (the id must only be unique inside the particular "category" value given).

"amount":	The number of items to transact. • Can be negative if the intention is to deduct that number of items
"info":	(Optional) A JSON object containing any additional parameters needed for an item. This is the only place where custom parameters are added inside the "items" object; all undefined keys in <itemObject> are reserved for future use.

Example Request Body

<jsonRequest>: {

- "system": "monetization"
- "requester": "btetrud"
- "t":1344385436
- "idOrigin": "facebook"
- "id": 23489
- "network": "f"
- "user":"c28k3fjj9"
- "items":[
    - {
        - "category":"item"
        - "id":"12"
        - "amount":1
    - }
-]

}

stringified <jsonRequest>:

```
{"system":"monetization","requester":"btetrud","t":1344385436,"idOrigin":"
facebook","id":23489,"network":"f","user":"c28k3fjj9","items":[{"category"
:"item","id":"12","amount":1}]}
```

secret: dummySecret

securityHash: G7sSpScpOgVc/GnZqSohRzpIvu0=

full request body:

```
G7sSpScpOgVc/GnZqSohRzpIvu0=
{"system":"monetization","requester":"btetrud","t":1344385436,"idOrigin":"
facebook","id":23489,"network":"f","user":"c28k3fjj9","items":[{"category"
:"item","id":"12","amount":1}]}
```

Response

The response body is UTF-8 encoded. <jsonResponse>, is a JSON object with the following members when:

- The request was successfully completed
    - "result": "success"
- The request was not completed
    - "result": <failure-type>- is one of the following:
        - "temporaryFailure" indicates the transaction should be retried

        - "permenantFailure" indicates the transaction should not be retried
    - "message": <error-message> - a safe but informative error message indicating the problem(s) in the request. This is optional if a "type" is given.
    - "type": <error-type> - The name of the error. This is intended to be a short name, in camel-case, of an error that might need special handling. This is optional if a "message" is given. **The following cases have standard types:**
        - "missingParameter" - the request is missing a required parameter. **Justification: letting implementations send invalid requests can cause problems down the line.**
        - "badRequest" - the request has all the required parameters, but is invalid. Examples of this include invalid argument types, or unexpected values.
        - "unauthorized" - the request could not be authorized. <error-message> for this type of error are not recommended in normal cases.
        - "duplicate"- duplicate is detected (using the "idOrigin" and "id" members)
        - "alreadyFull" - the user already has the maximum amount of a particular item (like game energy)
        - "cannotDebit" - the request is to debit an item and the user doesn't have enough to debit
    - "info": (Optional) JSON object that contains extra information about the error.
    - "item": (Optional) The index (starting at 0) of the item which caused the failure. This parameter is included only if an error happened with a specific item.

Appropriate HTTP response codes can be returned in addition to (but not instead of) the request body, but the requester should determine success or failure from the "result" field. If the result is malformed, the result is treated like a "temporaryFailure" response.

Other Requirements

- The whole transaction succeeds or the whole transaction fails. If you can use some kind of transaction-rollback mechanism, please do.

Testing

Use the ITA testing tool to test an implementation. The tool also serves to showcase various examples.

Additional Examples

<securityHash>

Java:

```
// hmac based on RFC 2104 (unsure if this is true anymore)
    // returns base64 encoded string
    public static String sha1HMAC(String key, String message) throws
Throwable {
        SecretKeySpec spec = new SecretKeySpec(
            key.getBytes(),
            "HmacSHA1");

        Mac mac = Mac.getInstance("HmacSHA1");
        mac.init(spec);

        byte[] result = mac.doFinal(message.getBytes());
        byte[] empty = new byte[]{};  // don't do line separaters..
        Base64 encoder = new Base64(0,  empty);  // and don't chunk the
result furcrisake

        return encoder.encodeToString(result);
    }
```

Supplementary Info

Security Techniques used

- required id parameter makes every request single-use

Change Log

Note: all items in the change log are reflected in the canonical documentation above.

v1.04

- Updating to use Authenticated Monetization API 0.2 and Monetization API 0.2.2 which is minor clarification of HTTP response codes.

v1.03

- Merged the documentation that was previously referenced into this page.

v1.02

- Specified that the "user" parameter can be a string or integer value.

v1.01

- Changed "quantity" key to "amount"
- Extending Authenticated Monetization API and cleaning out duplication

v1.0

- Added summary
- Added "item" key for failure responses
- Added "network" key for requests

v0.4

- id is no longer optional
- idOrigin and requester should use the same value as is under the key "system" in certain cases
- always use POST requests
- change "t" parameter to be in seconds
- remove the "h" parameter, and instead encode the hash in the response body
- use HMAC instead of key+message hash

v0.3

- changed "user" parameter to "requester"
- changed convention "user":"app" to "requester":"system"
- mentioned major and minor versions refer to the version of this document
- change items.args to items.info
- change http status codes to custom error types

v0.2

- removed the ability to inform requester about deprecated token (this should probably be 100% handled by the auth service and be totally transparent to the API)
- adding ability to add custom information into an error response
- adding a convention for indicating that a request is a duplicate

See also

- Background