SaurikIT™

Cydia™ Store

Application Programming Interface

Overview

The Cydia™ Store Application Programming Interface (API) allows product vendors and repository hosts to retrieve and provide information about the status of product purchasing and licensing transactions, and is provided and accessed over HTTP.

Request Protocol

Information about the state of product purchases is provided by

```
http://cydia.saurik.com/api/check
```

in response to HTTP requests. Requests are encoded in the query string of the request, and provide the following fields:

api (recommended)
The name and version of the API being used. For the API specified by this document, the value should be `store-0.9`, which is also the default value assumed by the server in the event that the *api* field is not provided.

device
The Unique Device Identifier (UDID) or custom identification hash — represented in lower-case hexadecimal — of the device for which purchase information is being requested.

mode
Either “local” or “recursive”. (See *Response Protocol*)

nonce
An arbitrary, one-time-use string. In most cases the current Unix time will suffice, but any random string is acceptable so long as the client never uses the same one twice.

product or **package**
The name of the package or product for which purchase information is being requested. Package names are defined by repository hosts, while product names can be set per-package in the Cydia™ Store database.

timestamp
The Unix time at which the request was generated. The timestamp must be within 5 minutes (300 seconds) of the server's internal Unix time when the request is processed.

vendor
The name of the party making the request.

version (conditionally recommended)
The version of the package being requested. If the *product* field was specified instead of the *package* field, the *version* field must not be included in the request; otherwise, it is recommended but not required.

host (optional)

The IP address of the device making the request of the vendor that in turn is making the current API request. Provided only when the API request is the immediate result of a user device request.

hash (optional)
The hashing method used to generate the device identifier used by the vendor. Currently the only supported hash function is “MD5”.

prefix (optional)
A string prepended to the UDID of a device before it is hashed.

signature
An HMAC-SHA1 of a data string (see *Signature Generation*) formed by the other fields and their values, signed with the vendor's secret key, or with the server's secret key if no vendor is specified.

If *hash* and *prefix* are specified, the identifier provided in the *device* field is assumed to be generated by applying the algorithm specified in the *hash* field to the concatenation of the *prefix* and the device's UDID.

Note that if the optional fields are specified, they too must be included in the data string used to calculate the *signature* for the request.

Signature Generation

The *signature* field is calculated as the HMAC-SHA1 of a string constructed from the other fields in the request and their values. The string is similar in specification to the query string used for the request, and indeed can be used to create the query string by appropriately appending the calculated *signature*. However, several restrictions apply to the data string that do not apply to the query string:

Both the query string and the data string conform to the specification for the `application/x-www-form-urlencoded` content type, but the data string must use **upper-case** hexadecimal percent encoded escapes and must use plus (+) characters instead of spaces. (The query string is free to use upper- or lower-case percent encoded escapes, and to use them instead of pluses to escape space characters.)

Additionally, the fields in the data string must appear in alphabetical order by field name, then by value if the field names are the same; but they may appear in any order in the query string. By the same token, the *signature* field may appear before, after or in between the other fields in the query string.

Example

To find out whether the user who owns the device with UDID `048108573c7ed8f52126a912d1517a6c40a48858` has purchased a license for the package com.widgco.wmark, we collect a data set similar to the following:

```
api=store-0.9&device=048108573c7ed8f52126a912d1517a6c40a
48858&host=32.174.245.141&mode=local&nonce=1234585489&pa
ckage=com.widgco.wmark&timestamp=1234585489&vendor=docho
st&version=0.9
```

Once we have constructed the data string, we take its HMAC-SHA1 using our secret key. (HMAC generation tools are available in the standard libraries of or in libraries for most popular web scripting languages.) Using the secret key `abcdef0123456789abcdef0123456789`, this produces the URL-safe base-64 encoded HMAC `F0AKwxM_oG5b9eExVprTWblO5V4`.

Once we have calculated the HMAC for our request data, we use it as the value for the *signature* field in our final query string resulting in an example request URL similar to:

```
http://cydia.saurik.com/api/check?nonce=1234585489&vendo
r=dochost&mode=local&package=com.widgco.wmark&host=32.17
4.245.141&api=store-0.9&version=0.9&device=048108573c7ed
8f52126a912d1517a6c40a48858&timestamp=1234585489&signatu
re=F0AKwxM_oG5b9eExVprTWblO5V4
```

Response Protocol

When the server receives a request as described in *Request Protocol*, it provides an `application/x-www-form-urlencoded` response. If the request was correctly-formed, the response contains fields representing the available information about the requested purchase records:

message (optional)
A descriptive message provided by the payment service.

nonce
The same as the value of the nonce provided in the request.

payment
An identifier of the payment made by the purchaser.

provider
The payment service used for the transaction, *e.g.* Amazon, PayPal.

state
A normalized indication of the status of the payment. See *State Codes* for details.

status
The payment status as provided by the payment service.

signature
An HMAC-SHA1 of a data string formed by the other fields and their values, signed with the vendor's secret key, or with the server's secret key if no vendor was specified in the request. This signature is calculated — and can be verified — by the same method used to calculate the signature for a request (see *Request Protocol*).

If the *mode* field of the request was set to “recursive” the server requests information on the requested purchase record from the server of the vendor of the specified product, and returns it if any is available. Such server-to-server requests do not include a *vendor* field and always specify a “local” *mode*; that is, the recursion is at most one level deep. If no information is available from the vendor's server, or if the *mode* field in the request was set to “local”, the server returns whatever information on the requested purchase record it has available locally.

If no information is available on the specified purchase record via the specified mode, only the *nonce* and *signature* fields are returned.

State Codes

Normally, the *state* field of a response will have one of the following values:

error
The status of the transaction could not be obtained because of an error communicating with the payment service.

pending
The transaction is still pending, *e.g.* due to a delay in communication between the payment service and the purchaser's credit provider.

failed
The transaction was not completed successfully, *e.g.* due to the purchaser lacking sufficient credit for payment.

completed
The payment completed successfully.

reversed
The payment transaction with the user has been reversed, *e.g.* via the payment provider due to a return request.

However, if the payment service returned a status that is unfamiliar to the server, no *state* field or value is returned.

Error Messages

If the server receives a request that cannot be fulfilled because it is not formed as specified by the *Request Protocol*, it returns an explanatory field in the response body instead of the requested information. The content of the explanatory fields is not part of the specification of the Cydia™ Store API, but some reasonable examples are provided below.

If the server could not determine the vendor making the request from the provided query string, a lone *message* field is returned:

missing vendor
The request omitted the mandatory *vendor* field.

unknown vendor
The *vendor* field was present, but specified a vendor not authorized with the Cydia™ Store database.

unsupported api
The *api* field specified an identifier for an API or API version not supported by the server.

If the vendor was correctly specified, an *error* field is returned, along with a *signature* field whose value is the HMAC-SHA1 of the URL-encoded *error* field, signed with the secret key of the vendor. The *error* field may have a value similar to one of the following:

missing nonce
The request omitted the *nonce* field.

missing timestamp
The request omitted the *timestamp* field.

missing product or package
The request omitted both the *product* field and the *package* field.

invalid product
The requested product or package value is not recorded in the Cydia™ Store database.

missing device
The request omitted the *device* field.

invalid mode
The request omitted the mode field, or included the mode field with a value other than “local” or “recursive”.

missing signature
The request omitted the *signature* field.

invalid signature
The *signature* provided in the request was incorrect.

Example

In response to the request represented by the URL in the *Request Protocol* example, the server would — if the private key, vendor, package and timestamp were valid — provide an application/x-www-form-urlencoded response similar to the following:

```
nonce=1234585489&status=Success&provider=Amazon&state=co
mpleted&signature=ZZCicZZZd61fKzh5y7n_FksRv68&payment=11
```

However, if the provided UDID was not associated with a purchase record for the package, no information would be available and the server would return only the *nonce* and a *signature*:

```
nonce=1234585489&signature=gGcuiEC2qsyD0cIn-iu7fCHNdjU
```

Similarly, if the request were malformed in some way, instead of information on the requested purchase record the server would return an error message similar to the following:

```
signature=RFtOfaT7KNwDyolZq151ESh8zKg&error=invalid+sign
ature
```