Apptrackr Application Programming Interface

Technical Specification 1.2  
July 2010  
5th Edition  
  
**Notational Conventions**

This document uses specific notational conventions. Information in *italics* will represent a field name of something. Information in **blue** will represent an object and action if a scope resolution operator delimits the appropriate information (for example, **Link::get()** represents the object **Link** and method/action **get**). Text in **UPPERCASE\_AND\_BOLD** represents a response code. Boolean values **true** and **false** are represented by green and red in bold text, respectively.   
  
When referencing blocks (a term used to reference an object OR an array, depending on the context) the name of the field and an empty set of brackets are displayed, for example auth{}.

**Introduction**

Apptrackr is a website dedicated to allowing users to share cracked iPhone, iPod Touch, and iPad applications so that people can trial them. This document outlines what can be achieved using the apptrackr Application Programming Interface (or API).   
  
The apptrackr API is an interface that allows developers to use the various functions of the apptrackr website for their own applications or websites.   
  
The functions available in the API include the following:

* Getting lists of applications sorted by certain filters (including search filters).
* Getting application information.
* Getting application links.
* Getting application screenshots.
* Submitting links for apps.
* Getting a list of the category.

**Interacting with the API**

**Making an API Request**

Requests to our API are sent to [http://api.apptrackr.org/.](http://api.apptrackr.org/) The request itself needs to be a properly-formatted JSON string sent via the POST variable request.   
  
The JSON string contains the following fields:

* *request*, a JSON encoded object which contains the request. Case sensitive. **Required.**
* *id*, a (maximum of 20 byte long) integer/string which will be returned along with the response. **Unnecessary and totally optional.** This is useful for asynchronous requests.

In PHP a full request to our API can be accomplished like so:

**<?php  
// This is the initial request.  
//  Object: App  
//  Action: getDetails  
//  Args:  
//    app\_id = 303612602  
//    fields = [ "appid" ]  
  
$request = array('object' => 'App',  
        'action' => 'getDetails',  
        'args' => array(  
                'app\_id' => 303612602,  
                'fields' => array("appid")  
        )  
);  
  
// Wrap the request and JSON encode it.  
  
$wrapper = array(  
        'request' => json\_encode($request)  
);  
  
// JSON and URLencode the wrapper.  
$wrapper = urlencode(json\_encode($wrapper));  
  
// Now, using the cURL extension, we perform the API request.  
$ch = curl\_init();  
curl\_setopt($ch, CURLOPT\_URL, 'http://api.apptrackr.org/');  
curl\_setopt($ch, CURLOPT\_POST, 1);  
curl\_setopt($ch, CURLOPT\_POSTFIELDS, "request=$wrapper");  
curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);  
$data = curl\_exec($ch);  
curl\_close($ch);  
  
// print the output  
print\_r($data);  
  
?>**

**Anatomy of a Request**

A request is also a JSON string which contains the following fields:

* *object*, the name of the object that you are calling. Case sensitive.
* *action*, the name of the action (method) that you are calling. Case sensitive as well.
* *auth*, an object that may be required for certain actions.
* *args*, an object which contains the arguments for the action being called. Different for every action (usually).

**Authentication**

Certain actions, including **Link::submit()**, require an authentication block to be passed within the request. Most of the time you will be sending a user *id* and the md5() of their password through the auth{} block in the *passhash* field.   
  
An auth{} block looks like this:

**{  
        'id':1,  
        'passhash':'5f4dcc3b5aa765d61d8327deb882cf99'  
}**

If you do not know the user's id, you have to call the API using the **User::checkAuth()** action. This allows you to send an auth{} block that looks like this:

**{  
        'username':'Kyek',  
        'passhash':'5f4dcc3b5aa765d61d8327deb882cf99'  
}**

The reason why we do this is so that whenever you contact us you can reference the user by their user id, which lets us save significant space in our caches and speeds up requests. Thus, we force our clients to use the id field in the auth{} block unless absolutely necessary.

**Handling the Response**

The response is a JSON object which may contain the following fields:

* *code*, an integer status code of the response.
* *data*, a JSON-encoded block which contains the response from the method, if there was one.
* *signature*, a base64-encoded signature which allows you to verify the data{} block from the response to make sure that someone hasn't intercepted the API call and forged a response. That would be nasty!

For example, a response may look like this:

**{  
"code" : 200,  
"data" : "{\"links\":{\"1.3.1\":[{\"id\":511,\"appid\":\"303612602\",\"version\":\"1.3.1\",\"cracker\":\"w00tb00t\",\"url\":\"http:\\\/\\\/www.appscene.org\\\/download\\\/uqa52MtH0n82c1237e1c\",\"time\":1262968525,\"shorthand\":\"appscene.org\"}]}}",  
"signature" : "[...]" // for security reasons, we do not show the signature of documentation responses.  
}**

**Verifying the Response**

We suggest verifying the response whenever a data{} block is present. The following is an example in PHP which demonstrates verifying the response.

**<?php  
// $response is the response from the server which contains a code, data, and signature field.  
// get the public key (this is the same for all profiles)  
$apptrackr\_pub\_key = openssl\_pkey\_get\_public(<<<'EOF'  
-----BEGIN PUBLIC KEY-----  
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCxyZS+9iSODM7uiv4g1CNV36xg  
zHsEgZaFxcy88BibdUxAEFwr0CgCy1TrnTMe87PmAElCmatPpGUSYmFQtM7YEsPf  
UNfB/8q/dEeHXAH2I93PGN3wdLicY9K2SOz6GbkAkoEnpGSYwOKIBBsKi4/wZ33W  
UcFkpmqMMlaiSc0zjwIDAQAB  
-----END PUBLIC KEY-----  
EOF);  
// verify the response data with the signature provided  
if (!openssl\_verify($response->data, base64\_decode($response->signature),  
$apptrackr\_pub\_key)) {  
die("Invalid response!");  
}  
?>**

**Response Codes**

The following is a list of response codes that may be returned when manipulating the API. Each response code is an integer.

* **OK** (200), which means the request was successful.
* **CREATED** (201), which means that the request was successful and any content you submitted was immediately created and added to the database.
* **ACCEPTED** (202), which means that the request was successful. Information is accepted for review and will be handled as soon as possible.
* **PARTIALLY\_CREATED** (206), which means that only a subset of the submitted content was created and added to the database.
* **PARTIALLY\_ACCEPTED** (207), which means that only a subset of the submmited content was accepted for review.
* **BAD\_REQUEST** (400), which means that the request was badly formatted or improperly submitted.
* **UNAUTHORIZED** (401), which means that the action which was called requires authentication, yet valid authentication was not provided.
* **USER\_NOT\_PERMITTED** (402), which means that the action requires authentication but that the authentication provided represents a user which does not have permission to make the request.
* **FORBIDDEN** (403), which means that the API profile does not have permission to access the object and/or method.
* **OBJECT\_NOT\_FOUND** (404), which means that the reqest was made to an object which simply does not exist.
* **ACTION\_NOT\_FOUND** (405), which means that the request called an action which does not exist.
* **FORMAT\_NOT\_ACCEPTED** (406), which means that the format of the request did not suit the requirements of the action called.
* **UNACCEPTABLE\_DATA** (407), which means that the data provided to the action was invalid and unacceptable.
* **DEPRECATED** (408), which means that the request is making a call to an action in a way which is not supported anymore and must be updated.
* **RESOURCE\_NOT\_FOUND** (410), which means that the request referenced a resource which couldn't be found, identified or processed.
* **INTERNAL\_SERVER\_ERROR** (500), which means something went horrifically wrong -- but at least it wasn't your fault!
* **NOT\_IMPLEMENTED** (501), meaning you called something on our API which hasn't been implemented yet.
* **SERVICE\_UNAVAILABLE** (503), which means that the API is unavailable at the moment.
* **RESOURCE\_TIMEOUT** (504), which means that the request triggered a call to an external resource that has stopped responding.

This is a PHP class that stores all of the response codes.

**/\*\*  
Code by Kyek. We love you, man.  
\*\*/  
Class APIStatusCode {  
        // Request was successful, nothing was created or accepted for review  
        const OK = 200;  
        // Request was successful and submission has been instantly posted  
        const CREATED = 201;  
        // Request was successful and submission has been accepted for review  
        const ACCEPTED = 202;  
        // Request was successful, but part or all of the submission already existed. Any unique content has been created.  
        const PARTIALLY\_CREATED = 206;  
        // Request was successful, but part or all of the submission already existed. Any unique content has been accepted.  
        const PARTIALLY\_ACCEPTED = 207;  
        // Request was badly formatted or improperly submitted  
        const BAD\_REQUEST = 400;  
        // Request requires a user's authentication, or the provided authentication was bad  
        const UNAUTHORIZED = 401;  
        // User authentication was accepted, but user does not have permission to perform the requested action  
        const USER\_NOT\_PERMITTED = 402;  
        // The API profile being used does not exist or is not activated.  
        const FORBIDDEN = 403;  
        // Request was made to an object that does not exist  
        const OBJECT\_NOT\_FOUND = 404;  
        // The API profile being used does not have permission to call this action  
        const ACTION\_NOT\_ALLOWED = 405;  
        // Request's format is either invalid or is not allowed for this type of request  
        const FORMAT\_NOT\_ACCEPTED = 406;  
        // Submitted data was in some way invalid, and no part of it could be accepted.  
        const UNACCEPTABLE\_DATA = 407;  
        // Request's format or requested action is deprecated  
        const DEPRECATED                = 408;  
        // Request referenced an external resource that could not be found  
        const RESOURCE\_NOT\_FOUND = 410;  
        // Request failed at no fault of the requesting party.  
        const INTERNAL\_SERVER\_ERROR = 500;  
        // Request called an action that either does not exist or is not yet available  
        const NOT\_IMPLEMENTED = 501;  
        // The API is temporarily unavailable  
        const SERVICE\_UNAVAILABLE = 503;  
        // The request triggered a call to an external resource that has stopped responding  
        const RESOURCE\_TIMEOUT = 504;  
}**

We recommend adding support for all response codes.

**Objects and Actions**

1. **User**

The User object holds methods which allow the client to manipulate our user database. To protect the privacy of our users, this object will not disclose any sensitive information about users. At this time, only a checkAuth action is available at this time.  
  
**checkAuth**

* + **What it does:**  
    This action simply requires an auth{} block that contains username and passhash fields. If the authentication details provided are valid, the userid will be returned. This method is essential for retreiving the userid of a user, because everywhere else in the API requires an auth{} block with a id field instead of a username field.
  + **Arguments:**  
    The checkAuth action does not need any arguments. This method only requires an auth{} block.
  + **Example:**  
    The following request will check Kyek's authentication.

**{  
        "object": "User",  
        "action": "checkAuth",  
        "auth": {  
                "username": "Kyek",  
                "passhash": "872c1c66c512dd160496c7496891230a"  
                }  
}**

If successful, the response data{} block will contain the following:

**{  
        "userid": 1  
}**

* + **Response Codes Related to this Action:**  
    This action will return an **OK** response code if everything went well. If not, an **UNAUTHORIZED** response code may be returned (which implies that the authentication failed).

1. **Category**

The Category object will provide API functionality with manipulating and getting information about the category details on Apptrackr. Keep in mind, all app information returned from **App::getDetails** and **App::getList** will reference or require a category ID, and without having a lookup table you will not know what that category ID represents. Use the **Category::getList** method to obtain a list of categories.  
  
**getList**

* + **What it does:**  
    This method returns a list of categories.
  + **Arguments:**  
    None required. No auth{} block is required either.
  + **Example:**  
    The following request will get a list of categories.

**{  
        "object": "Category",  
        "action": "getList"  
}**

This is the response. *Note: some categories have been cut from this response to  
save space in this document.*

**{  
        "categories":[  
                {"id":1,"name":"Utilities","apps":788,"iconurl":"http:\/\/static.apptrackr.org\/caticons\/uti.jpg"},  
                {"id":2,"name":"Productivity","apps":435,"iconurl":"http:\/\/static.apptrackr.org\/caticons\/pro.jpg"},  
                {"id":3,"name":"Music","apps":426,"iconurl":"http:\/\/static.apptrackr.org\/caticons\/mus.jpg"},  
                {"id":4,"name":"Travel","apps":265,"iconurl":"http:\/\/static.apptrackr.org\/caticons\/tra.jpg"},  
                {"id":5,"name":"Navigation","apps":276,"iconurl":"http:\/\/static.apptrackr.org\/caticons\/nav.jpg"},  
                {"id":6,"name":"Books","apps":237,"iconurl":"http:\/\/static.apptrackr.org\/caticons\/boo.jpg"},  
        ]  
}**

In the response there will be a *categories* array within the unwrapped data{} block. This array will contain objects; each object is a category and has four fields: *id*, *name*, *apps*, and *iconurl*. The *iconurl* is a URL to a 57x57 image representing the category. *apps* is an integer which represents the amount of apps in that category. *name* is a string name of the category. *id* is the category ID.   
  
*Note: Category ID 0 (All) is not returned within this response. It needs to be manually augmented on the client-side. If it were to be in the result, it would look like this:*

**{  
        "id":0,  
        "name": "All",  
        "apps": 18000,  
        "iconurl": "http:\/\/static.apptrackr.org\/caticons\/all.jpg"  
}**

*Note: screenshots, icons, category icons, etc. on apptrackr have hotlinking protection; you need to download the images and store them, and you may not hotlink to them on a website.*

* + **Response Codes Related to this Action:**  
    **OK** will be returned on a successful category list (along with the category list in the data{} block). **RESOURCE\_NOT\_FOUND** will be returned if the category list could not be obtained.

1. **Screenshot**

The Screenshot object allows clients to access information and manipulate our screenshots database. Every application \_probably\_ has screenshots for you to access. Screenshots are typically either 480x320 (horizontal) or 320x480 (vertical). Horizontal images have been marked for you; every screenshot which is horizontal will have an *is\_horiz* field set to int(1).  
  
**get**

* + **What it does:**  
    The **Screenshot::get** action will return all of the screenshots for a given *app\_id*. Remember, app ids on apptrackr are the same as iTunes IDs on iTunes.
  + **Arguments:**  
    This action requires an args{} block with an *app\_id* field.
  + **Example:**  
    This example will get the screenshots for app id 304923687.

**{  
        "object": "Screenshot",  
        "action": "get",  
        "args": {  
                "app\_id": 304923687  
        }  
}**

Here's an example response:

**{  
        "links":[  
                {"is\_horiz":0,"url":"http:\/\/static.apptrackr.org\/itunes\/304923687-1263008213\/screenshot0\_1263008281\_1ecc50b19ac136dec237cef7d54cb7ff.jpg"},  
                {"is\_horiz":0,"url":"http:\/\/static.apptrackr.org\/itunes\/304923687-1263008213\/screenshot1\_1263008281\_67f828f6315a583dd6fe77128907d066.jpg"},  
                {"is\_horiz":0,"url":"http:\/\/static.apptrackr.org\/itunes\/304923687-1263008213\/screenshot2\_1263008281\_c88cbe17075baf6eb5ee4dbbfeab769f.jpg"},  
                {"is\_horiz":0,"url":"http:\/\/static.apptrackr.org\/itunes\/304923687-1263008213\/screenshot3\_1263008281\_d1ad2635891eb05bc785e9febaf56702.jpg"},  
                {"is\_horiz":0,"url":"http:\/\/static.apptrackr.org\/itunes\/304923687-1263008213\/screenshot4\_1263008281\_f0431faa9e8bd5da6d9ed3cd36666214.jpg"}  
        ],  
        "horizontal":0,  
        "vertical":5  
}**

In the response data{} block there are three fields: *horizontal*, *vertical*, and *links*. *horizontal* and *vertical* are integers that represent the number of each respective types of screenshots in the list. *links* is an array list of objects (screenshots). Each of these has a *is\_horiz*and *url* field. *is\_horiz* describes whether or not it's a horizontal image.  
  
*Note: screenshots, icons, category icons, etc. on apptrackr have hotlinking protection; you need to download the images and store them, and you may not hotlink to them on a website.*

* + **Response Codes Related to this Action**  
    **OK** will be returned on a successful request. **RESOURCE\_NOT\_FOUND** will be returned if screenshots could not be obtained for the given *app\_id*.

1. **Link**

The link object allows you to manipulate links on apptrackr, which includes submission and listing links for a particular application.  
  
**get**

* + **What it does:**  
    The **Link::get** action will return a list of links for a given application. We do not return the full list unless an *all\_versions* argument has been passed.
  + **Arguments:**  
    This action requires an *app\_id* field in the args{} block which must be an integer application ID.
  + **Example:**  
    This is a request which gets the latest version's links for the application ID 310749044 (Trivial Pursuit®).

**{  
        "object": "Link",  
        "action": "get",  
        "args": {  
                "app\_id": 310749044  
        }  
}**

Here is an example response:

**{  
        "links":{  
                "1.1.20":[  
                        {  
                                "id":36372,  
                                "appid":"310749044",  
                                "version":"1.1.20",  
                                "cracker":"Matsuya",  
                                "url":"http:\/\/www.appscene.org\/download\/qMvRb5280I33c438f7c1",  
                                "time":1265135049,  
                                "shorthand":"appscene.org",  
                                "packagetype":"ipa",  
                                "active":1  
                        },  
                        {  
                                "id":33072,  
                                "appid":"310749044",  
                                "version":"1.1.20",  
                                "cracker":"YukaChan",  
                                "url":"http:\/\/www.appscene.org\/download\/BF5UGL9dNs77529c7af2",  
                                "time":1264246263,  
                                "shorthand":"appscene.org",  
                                "packagetype":"ipa",  
                                "active":1  
                        }  
                ]  
        }  
}**

Inside of the response is a links{} block. The keys are version numbers and their values are arrays which contain link objects. Every link has the following fields: *id*, *appid*, *version*, *cracker*, *url*, *time*, *shorthand*, *packagetype*, and *active*. *id* is the ID of the link. *appid* is the ID of the application. *version* is the version number. *cracker* is a string cracker name (if it's empty, please default to "Unknown"). *url* is the url to the link. *time* is a unix timestamp that represents when the link was submitted. *shorthand* is a shorthand representation of the URL. *packagetype* is either going to be "ipa" or "torrent". *active* will typically be 1.

* + **Response Codes Related to this Action:**  
    On a successful request, **OK** is returned. In the event an auth{} block is required and the one supplied is invalid, **FORBIDDEN** will be returned. If links for the application ID provided cannot be found, **RESOURCE\_NOT\_FOUND** will be returned.

**submit**

* + **What it does:**  
    This action will allow the client to submit up to five links for a particular application. These links must adhere to the same whitelist that the main website does. Submissions using the API will be entered into the same submission queue that submissions made on the website are entered into.   
      
    This action **requires** authentication using the auth{} block.
  + **Arguments:**  
    This action requires an iTunes URL to be based within the args{} block as the *itunes\_url* field. Other mandatory arguments include a *version* argument (string version name), *cracker* (cracker name), and the *links* argument (an array of strings, which are all URLs).
  + **Example:**

**{  
        "object": "Link",  
        "action": "submit",  
        "auth": {  
                "id": 34759,  
                "passhash": "df0fbf14ca64335bcfc046761fe13a0e"  
                },  
        "args": {  
                "itunes\_url": "http://itunes.apple.com/us/app/voodoofun/id295459100?mt=8",  
                "version": "1.0",  
                "cracker": "Unknown",  
                "links": [  
                        "http://www.appscene.org/download/FBcN35jZvc390e32d24f"  
                ]  
        }  
}**

There is no response in the data field of this action.

* + **Response Codes Related to this Action:**  
    This action will return **ACCEPTED** or **PARTIALLY\_ACCEPTED**. **ACCEPTED** implies that all of the contents of the submission were immediately entered into the submission queue, while **PARTIALLY\_ACCEPTED** implies that only some of the submission was entered into the queue (the reasons being that at least one of the links did not work). If the entire submission failed and no links were submitted, **UNACCEPTABLE\_DATA** is returned.

1. **App**

The App object will allow the client to get details and lists of applications that apptrackr indexes. Application IDs are, as always, exactly the same as the IDs on iTunes. Search can be achieved using the search field in **App::getList**.  
  
**getDetails**

* + **What it does:**  
    This action will allow the client to get details for a given *app\_id*. The client must specify what fields it wants returned.
  + **Arguments:**  
    This action requires the *app\_id* and *fields* arguments be sent in the args{} block of the request. *app\_id* is an integer ID of the application, and *fields* is an array of the fields that the client wants to return.
  + **Example:**  
    This request will get the *app\_id* of application ID 291443933. Sounds odd, but hey it's an example nonetheless.

**{  
        "object": "App",  
        "action": "getDetails",  
        "args": {  
                "app\_id": 291443933,  
                "fields": ["appid"]  
        }  
}**

The response:

**{  
        "app": {  
                "appid": 291443933  
        }  
}**

* + **Response Codes Related to this Action:**  
    This action will return **OK** on a successful request. If the *app\_id* requested does not exist, **RESOURCE\_NOT\_FOUND** will be returned. If the client makes a request for a field that cannot be requested, **ACTION\_NOT\_ALLOWED** will be returned.
  + **Fields that can be Requested:**  
    The following fields can be requested:  
    - appid
    - scrapeid
    - add\_date
    - last\_modification
    - itunesurl
    - name
    - seller
    - category
    - storeid
    - size
    - latest\_version
    - release\_date
    - copyright
    - price
    - rating
    - icon57
    - icon75
    - icon100
    - icon175
    - description
    - whatsnew
    - requirements
    - languages
    - universal

**getList**

* + **What it does:**  
    This action allows the client to retrieve a list of applications from apptrackr which are sorted and filtered by certain information. The pagination functionality on the main website (15, 30, 60 apps per page) is also handled by the API.
  + **Arguments:**  
    - **appsPerPage** is a **required** argument that can be an integer 15, 30 or 60. This defines how many apps will be returned per page max.
    - **sort** is a **required** argument that can be an integer 1, 2, 3, or 4, which defines the sorting options of the request. 1 defines newest apps, 2 defines newest apps and updates, 3 defines alphabetical order, and 4 defines search relevance (this will only work on searches).
    - **page** is a **required** argument that is an integer defining what page needs to be retrieved. Pages start at number 1.
    - **search** is an optional argument that is a string which filters apps by certain search keywords.
    - **category** is an optional argument that is an integer which defines what category of apps we are returning. 0 means all categories, and is the default.
    - **showPageCount** is an optional argument that is a boolean argument defining whether or not we need to retrieve the number of pages for the parameters we have provided. Please do this once, since it forces us to make another request to the backend.
    - **deviceid** is an optional argument that is an integer which defines what device category of applications should be returned. 0 means All, 1 means iPhone and iPod Touch compatible applications, and 2 means iP**a**d compatible applications. By default this is 0.
  + **Example:**

**{  
        "object": "App",  
        "action": "getList",  
        "args": {  
                "sort": 1,  
                "appsPerPage": 15,  
                "page": 1  
        }  
}**

The response is pretty huge, obviously. Essentially, there will exist an apps{} block inside the unwrapped *data* field from the response. This apps{} block will be an array that contains objects: each object is an app returned. The fields returned per app are *id*, *add\_date*,*last\_modification*, *name*, *seller*, *category*, *latest\_version*, *icon57*, *icon75*, *icon100*, *deviceid*, and *universal*.

**scrape**

* + **What it does:**  
    This action will use the Apptrackr scraping object to perform a scrape operation on a given iTunes URL. This action is useful for returning what versions we know about and other miscellaneous details about applications from iTunes.
  + **Arguments:**  
    This action requires an iTunes URL to be based within the args{} block as the *itunes\_url* field.
  + **Example:**

**{  
        "object": "App",  
        "action": "scrape",  
        "args": {  
                "itunes\_url": "http://itunes.apple.com/us/app/keynote/id361285480?mt=8"  
        }  
}**

This request would return the appropriate fields of scrape information, which include:

* + - **id**: a string which is the *appid* and *scrapedate* delimited by a '-'.
    - **md5**: an md5 checksum of the scrape XML data as returned by iTunes when the scrape occurred
    - **scrapedate**: a timestamp of when the scrape was performed and added to the database
    - **itunesurl**: the iTunes URL for the scrape
    - **appid**: the application ID from iTunes
    - **allversions**: a comma-delimited list of version numbers that we know about
    - **name**: the name of the application
    - **seller**: the seller of the application
    - **category**: the string category name
    - **storeid**: the store ID (needed during the scrape process, we retain it for efficiency)
    - **size**: the size of the application as identified by iTunes
    - **latest\_version**: the latest version of the application
    - **release\_date**: the release date as denoted by iTunes
    - **copyright**: the copyright data
    - **price**: the price of the application as described by iTunes
    - **rating**: the rating of the application as described by iTunes (age-rating)
    - **icon-100**: the location of the 100x100 icon for the app on Apple's servers
    - **icon-75**: the location of the 75x75 icon for the app on Apple's servers
    - **icon-57**: the location of the 57x57 icon for the app on Apple's servers
    - **icon-175**: the location of the 175x175 icon for the app on Apple's servers
    - **description**: the description of the application as provided by iTunes
    - **whatsnew**: the description of the latest version of the application as provided by iTunes
    - **requirements**: the requirements the application needs (devices it supports)
    - **languages**: the languages the application supports
    - **screenshots**: an array of screenshot URLs on Apple's servers
    - **universal**: an integer 1 or 0 reflecting whether the application is a fat binary (universal application) and can be ran on both iPad and iPhone/iPod natively.

**checkUpdates**

* + **What it does:**  
    This action will check the latest versions for multiple application IDs provided, useful for checking for updates to many applications on a device.
  + **Arguments:**  
    This action requires an array of integer app IDs be provided within the args{} block as the field name *appids*.   
      
    A maximum of 100 requests can be made in a single call to this object.
  + **Example:**

**{  
        "object": "App",  
        "action": "checkUpdates",  
        "args": {  
                "appids": [  
                        328526712,  
                        367787154,  
                        360529272,  
                        327380146  
                ]  
        }  
}**

This action will respond with an object in the data{} block containing associative *latest\_version* numbers for each *appid*. If information could not be identified for an *appid*, it will return boolean **false** instead of the version string.

* + **Response Codes Related to this Action:**  
    This action will return **OK** in most cases.

1. **Bundle**

This object takes advantage of an internal storage on apptrackr for bundle ID resolution. In essence, the bundle service lets you retrieve what iTunes IDs associate with any given amount of bundle IDs. Apptrackr stores most of the App Store's application information in this database.   
  
Profiles which use this API Object must cache as much data locally as possible to avoid calling the actions unnecessarily.  
  
**getItunesIDs**

* + **What it does:**  
    This action will return a list of known iTunes IDs for each given bundle ID. You may only request 100 bundle IDs at a time.
  + **Arguments:**  
    This action accepts a bundleList field in the args{} block. The field is an array of string bundle identifiers which the client wishes to look up.
  + **Example:**

**{  
        "object": "Bundle",  
        "action": "getItunesIDs",  
        "args": {  
                "bundleList": [  
                        "com.turtle.face",  
                        "net.taco.meow",  
                        "com.frost.tom",  
                        "what.the.hell"  
                ]  
        }  
}**

The response is an object containing the lookup values. Failed lookups may be boolean false instead of an integer iTunes ID.

* + **Response Codes Related to this Action:**  
    This action will return **OK** in most cases.

**getBundleIDs**

* + **What it does:**  
    This action will return a list of known bundle IDs for each given iTunes ID. You may only request 100 iTunes IDs at a time.
  + **Arguments:**  
    This action accepts a iTunesList field in the args{} block. The field is an array of integer iTunes IDs which the client wishes to look up.
  + **Example:**

**{  
        "object": "Bundle",  
        "action": "getBundleIDs",  
        "args": {  
                "iTunesList": [  
                        371250129,  
                        338343971,  
                        301517052  
                ]  
        }  
}**

The response is an object containing the lookup values. Failed lookups may be boolean false instead of a string bundle identifier.

* + **Response Codes Related to this Action:**  
    This action will return **OK** in most cases.