Pink Visual API Documentation

Release

December 12, 2011

CONTENTS

The Pink Visual API is an interface for fetching data, images and videos from Pink Visual, a major adult entertainment company. The goal of the PV API is to empower others to create interesting adult web sites and apps using our content.

You can *make money* by linking to Pink Visual's tour pages and join forms, or you can run your own adult website using *Pluginfeeds*.

Check out the full website at http://api.pinkvisual.com

Contents:

CHAPTER

ONE

INTRODUCTION TO PV API

This guide will help you get started with the PV API.

1. **Sign Up**

- **Sign up** - Visit http://api.pinkvisual.com/signup/ to sign up for a PV API account.
- **Confirmation** - Confirm your e-mail address by clicking the link we mail to you when you sign up.
- **Settings** - Tweak your defaults on the settings page – http://api.pinkvisual.com/settings/ . You can determine which *data format* you prefer, and set up *default filters* so you only get straight or gay content, for example.

2. **Legal**

- **2257 link** - Since PV API allows you to display adult content, you should put a link on your web page (or app) that indicates compliance with the 2257 record-keeping laws. You should link to http://clients-support.com/2257.html for any images or videos you receive from the PV API.
- **Terms** - You must comply with our Terms of Service to use the PV API. We think they're fair.

3. **Implementation**

- **Build your site** - We can't help you with this part. You'll need to know some kind of web programming language. You'll also need a server, a domain name, and a good idea for a site. Or, you might be building an app. We don't really care what you make (so long as it follows our Terms of Service).
- **Getting data from the API** - The PV API provides all of its data via URL requests and JSON (or XML) data responses. Check out the *list of API requests* for details.

4. **Approval**

- **Getting approved** - While you're developing, your API key is limited to just a few items for any request. When you're ready to get approval, send us a link to your site, some details about it, and what your plans are. You should also let us know which API key you're using for it. We'll take a look at what you've got. Once we approve it, your API key will lose its limitations and you're good to go. You can submit for approval at http://api.pinkvisual.com/contact/ .

5. **Monetize**

- **Connect to Topbucks** - See *Monetize*.
- **Get full adult videos** - See *Pluginfeeds*.

CHAPTER

TWO

IMPLEMENTATION NOTES

2.1 Generic Handling

In order to facilitate generic handling of data from the PVAPI, all requested data is as consistent in its formatting as possible. No matter how you *request an episode*, you receive the same data for each *episode*. The request will always return the data inside an “episodes” array (for JSON) or “<episodes>” element (for XML) – even if your request was for just one episode.

This allows you to write very generic code to encapsulate and manage the requested data.

CHAPTER

THREE

PARAMETERS

This section contains the possible request parameters.

3.1 Version

http://api.pinkvisual.com/get/<latest>/...

Each request has a **version** parameter in its URL (emboldened as “latest” in the example above). PVAPI currently has only one version, the “latest” version. However, this parameter is provided in the event that some future change requires PVAPI to change a request, its returned data, or some other part of the system. In that event, you will be able to use the version parameter to specify an older version of the request.

You should always prefer to use the “latest” version. This is currently an alias for “1.0”, and you may choose to use “1.0” as your version if you want to ensure no request ever changes in a way that breaks your code.

In some cases, PVAPI may make upgrades or tweaks to the system that will not affect your code. For example, new data may be added, or the search algorithm altered. In cases like this, PVAPI will *not* create a new real version number. This function is reserved only for the rare changes that break backwards-compatibility.

3.2 Start

In order to facilitate pagination, you can pass the **start** parameter to most requests. This indicates where, out of the total data set, to begin fetching.

All requests have a default “start” of zero (0).

API keys that are in the “development” stage will have an enforced start value. You can remove this limit by getting your site or application approved.

3.2.1 Use

Add &start=X to any API request, where X is an integer and is the item to start on.

If you are showing 10 items per page, page 1 is start=0 and page 2 is start=10.

If you are showing 10 items per page, page 1 is start=0 and page 2 is start=10.

3.3 Limit

The **limit** parameter determines the number of items to fetch. When combined with “start”, this is the mechanism for pagination.

Most requests default to “no limit”.

Some requests have an enforced maximum limit. In this case, you can still pass any limit without receiving an error. You will only receive the maximum number allowed, and the “limit” entry in the returned data will be changed.

API keys that are in the “development” stage will also have an enforced limit. You can remove this limit by getting your site or application approved.

3.3.1 Use

Add &limit=X to any API request, where X is an integer and the number of items to fetch.

If you’re showing 10 items per page, pass limit=10.

If you want *all* items, pass limit=-1 (negative 1) or omit this limit.

3.3.2 Tips

If you only need to know the total number of objects, you can make a request with limit=0. The returned data will still include the “total” entry as normal.

3.4 Sort

Many requests can be organized with the **sort** parameter. For example, if you want to show the hottest episodes, sort by rating.

3.4.1 Valid sort options

The following sorts are recognized by many API requests:

- name - Sort by episode/model/niche name.
- date - Sort by newest (episodes only).
- rating - Sort by rating (episodes only).
- relevance - Sort by relevance (search only).
- random - Random order for the results.

3.4.2 Use

Add &sort=X to any request, where X is the name of a sort option.

Add &sort=X to any request, where X is the name of a sort option.

3.5 Filters

You can restrict types of data by using the **filter** option. For example, if your site shows only gay porn, you should pass the “gay” filter.

Note: You can set default filters in your Settings page.

3.5.1 Valid filters

You can pass any of these filters:

- Orientation filters (mainly for Episodes):
- straight (s)
- gay (g)
- tranny (t)
- Gender filters (only for Actors):
- male (m)
- female (f)
- tranny (t)

3.5.2 Use

Filter your results by putting &filter=X in the API request URL, where X is either the full name (“gay”) or short name (“g”) of a filter, above.

You can use more than one filter by comma-separating them in the request: filter=straight,tranny (or filter=s,t)

If you pass no filters, you receive unfiltered content. If you pass at least 1 filter, you receive *only* the content requested by the filter.

If you pass no filters, you receive unfiltered content. If you pass at least 1 filter, you receive *only* the content requested by the filter.

3.6 Mode

The **mode** parameter specifies what format you’d like to receive data in. You can choose a default mode in your Settings page, but you can also override that with this parameter.

3.6.1 Valid Modes

- **json** - Receive data in the Javascript Object Notation format. This is a simple, human-readable format.
- **xml** - Recevie data in the Extensible Markup Language. This is more complicated than json, but we know some developers prefer XML.

3.6.2 Use

You can request a data mode by putting &mode=xml or &mode=json in your request URLs. If you omit this parameter, it will default to the one you chose on your Settings page.

CHAPTER

FOUR

DATA TYPES

The API returns the following data types:

4.1 Episode

An **episode** is one movie, including its *Niches*, meta data, and videos.

See also: *List of Episode Requests*

4.1.1 Data

Key

Episodes always come as an array inside the ‘episodes’ key.

Contents

An episode contains:

- *id* - The id is used to request the episode from any API function that takes an episode as a parameter.
- *name* - The name of the episode. For some *sources*, this is the name of the main actor. For others, it’s more fanciful.
- *date* - The episode’s release date, as a Unix timestamp.
- *desc* - A description of the episode.
- *votes* - The number of votes cast for this episode’s rating.
- *rating* - The episode’s rating as a decimal (e.g. “4.25”), on a scale of 1 to 5.
- *sources* - An array containing 1 *content source* entry – each episode has just one source. (The key is still “sources” to match other requests that return source(s).)
- *vidlen* - The length of video available, in minutes.
- *thumb* - A URL to the episode’s *thumbnail image* (160px by 120px).
- *hardcore_thumb* - A URL to the episode’s hardcore *thumbnail image* (160px by 120px).
- *stdimgs* - An array of URLs to the five *standard images*.
- *gallery* - An array of URLs to sixteen *gallery images*.

- *niches* - An array of *niches*.
- *trailers* - An array of *trailer videos* (see that article for formats).
- *actors* - An array of *actors* who appear in the episode.
- *joinlinks* - An array with "pc" and "mobile" links to join pages. If you are *monetizing* your API-based site, these links include your webmaster id.

4.2 Source

A **content source**, or **source**, is a grouping of *episodes* with a common theme. These relate to *Pink Visual* products.

See also: *List of Content Source Requests*

4.2.1 Data

Key

Content sources always come as an array inside the 'sources' key.

Some elements, like *episodes*, will come with a 'sources' key that contains one content source. The key name, 'sources', is still plural for consistency. Since each episode has only 1 source, its 'sources' key will contain 1 entry that is a content source.

Contents

Content sources contain:

- *id* - The id is used to request the content source, or used in requests like `episodes for source` to request a group of episodes. This key is a short lowercase string.
- *name* - The name of the content source.
- *gay* - A boolean (true/false) that indicates whether the content from this source is "gay". In the scope of this system, "gay" refers to content that involves one or more men and no women. Lesbian content is categorized as straight for that reason.

4.3 Niche

A **niche** is a tag, usually attached to an *Episode*, which describes what's going on in that episode.

See also: *List of Niche Requests*

4.3.1 Data

Key

Niches always come as an array inside the 'niches' key.

Contents

Niches contain:

- *id* - The id is used to request the niche, and can also be submitted in an *episode search* to retrieve episodes with this niche.
- *name* - The display name of the niche.

4.3.2 Requests

The following requests return niches:

- `niches` - Fetches a list of niches.

4.3.3 Notes

Some niches have multiple-word ids (they contain a space).

The "18+ Teen" niche is named to be more search engine friendly. Some search engines prefer that the word "teen" be qualified to prevent legal ambiguity.

4.4 Actor

An **actor** is one performer who appears in an *episode*.

See also: *List of Actor Requests*

4.4.1 Data

Key

Actors always come as an array inside the 'actors' key.

Contents

An actor contains:

- *id* - The id used to request this actor (or data related to the actor) from any API request. Actor ids are strings.
- *name* - The actor's name. If the actor uses a *nom de porn* (stage name), that is the only name provided. You cannot request an actor's real name.
- *sex* - This will be one of "male", "female" or "tranny".
- *thumb* - A URL to the actor's 160x120 thumbnail image.
- *stdimgs* - An array of the *standard images* available for this actor.

4.5 Videos

Each *Episode* has various **videos**, in two access levels:

Trailers are short videos that show off parts of the episode. These videos are provided for most episodes.

Paid content videos are full scenes (and sub-clips that have been cut from them, for slower connections) for each episode. These come from a separate API request and are only available to API users who have linked a *Pluginfeeds* account to their API key.

4.5.1 Format

All videos are in the MPEG format.

4.5.2 Requests

Trailers come in an array under the "trailers" key for every *Episode*. This array will have the various quality sub-arrays inside it.

Videos come only in the `paid content` request, in the "videos" key.

4.5.3 Quality

The "trailers" and "videos" keys are sub-divided by quality. All videos come in four different qualities:

- **min** - "Minimum" quality. These videos have a low framerate and resolution, but are good for very slow connections like some mobile networks.
- **low** - Low quality. This quality is a good choice for most mobile phones, as it will play quickly on many networks, and its resolution is enough for those phones.
- **med** - Medium quality. This is a good quality to show to PCs with slow to moderate connections. This quality is higher than necessary for most phones, which don't show a big difference between *low* and *med*.
- **high** - High quality. This quality will be slow to load on some PC internet connections, but looks good on most PC screens.

Note: Not all qualities are available for all episodes. If an episode doesn't have a certain quality of trailer or video, the array will still have that quality key, but the value will be null.

4.5.4 Trailers

Even though there is always just 1 trailer video in each quality level for an episode, the quality arrays contain a "clip" entry with the URL to the trailer. This is to accommodate the XML DTD. However, it is true even for JSON requests.

4.5.5 Paid Videos

The quality arrays for the "videos" key in the `paid content` request contain multiple video URLs. These are keyed by the sub-clip number (in JSON) or the word "clip" (in XML) for sub-clips, or "scene" for the full scene. The sub-clips contain the same video as the full scene, but are split to reduce load times on slower connections.

Note: URLs you receive for full videos are access-controlled and will expire. It's important that you do not re-use these URLs. When you are going to provide them to a visitor on a web page, request them fresh from the PV API.

4.6 Images

Several PVAPI requests return image URLs. Because PVAPI is an Adult service, these images often graphically depict sexual acts.

Pink Visual is the 2257 records holder for all images provided. You should see the `Getting Started` document for information on 2257 data and how you should link to our 2257 records in order to comply with the law.

4.6.1 Thumbnails

Episodes and *actors* both have thumbnail images. These images are 160px by 120px and usually depict the face of an actor. However, they may depict sexual acts as well.

Episodes also have hardcore thumbnails, which are also 160x120 but generally depict "hardcore" situations.

4.6.2 Standard Images

Episodes come with 5 standard images:

1. 312x416 - This is a tall image that is usually a good representation of the episode. Depending upon the *content source*, it will usually depict the main actor engaging in a sex act. However, some sources have a different focus.
2. 212x159 - This image usually comes from the beginning of the episode, and usually shows the actors still clothed. However, some episodes will show sex acts in this image as well.
3. 212x159 - This image is taken from the middle of the episode and is a preview of its contents.
4. 212x159 - This image is taken from the middle of the episode and is a preview of its contents.
5. 212x159 - This image is usually from the end of the episode, and may show a "facial" or other completion act.

Actors has, at this time, just one standard image.

1. 282x159 - This is generally a larger version of the actor's thumbnail image.

4.6.3 Gallery

Episode data includes a set of gallery images. These are are selection of thumbnail images from the episode's full gallery.

Episodes include 16 gallery images, taken from roughly even points in the episode to provide an idea of the contents of the episode from beginning to end.

The gallery images are usually 245x200 pixels. However, older episodes frequently have smaller gallery images. PVAPI recommends that if you use these images, you put them in an element that resizes itself as-needed.

CHAPTER

FIVE

API REQUESTS

This is a list of the valid API requests.

5.1 Episode Requests

This category contains requests that return an *episode*.

- `one episode`
- `all episodes`
- `episodes from content source`
- `episodes with a niche`
- `episodes with an actor`
- `search episodes`
- `episodes beginning with a letter`
- `paid content`

Episode requests have a fixed maximum value for *limit*. If you request a higher limit, or omit the limit parameter, the request will be processed but you will receive at most the maximum allowed limit.

5.2 Content Source Requests

This is a list of the requests that return a *content source*.

Link these:

- `one source`
- `all sources`

5.3 Niche Requests

This is a list of the requests that return a *niche*.

Link these:

- `one niche`

- `all niches`
- `niches for an episode`
- `niches beginning with a letter`

5.4 Actor Requests

This is a list of the requests that return an *actor*.

Link these:

- `one actor`
- `all actors`
- `search actors`
- `actors in an episode`
- `actors for a niche`
- `actors beginning with a letter`

CHAPTER

SIX

PINK VISUAL

Pink Visual, the innovative adult company that brings you the PV API, is a reality and gonzo pornography company based in Van Nuys, California.

PV API combines several Pink Visual programs and services, including our affiliate program Topbucks and our video feeds program Pluginfeeds.

At Pink Visual, the motto is "We Innovate, You Masturbate". Now we've decided to turn it around, to encourage you to be the innovator. Use the PV API to bring our quality adult content to the world in ways we've never imagined!

CHAPTER

SEVEN

MONETIZE

There are a lot of reasons to use the PV API, and one of them is to **make money**. The way to do this is with the Topbucks affiliate program.

Once you have both a PV API account and a Topbucks account, follow these instructions to link the two:

1. Log into PV API.
2. Go to your settings page.
3. Enter your Topbucks webmaster id in the "revid" field and click "submit".

Now, whenever you receive *episode data* from the PV API, the "joinlinks" data for that episode will contain your webmaster id. Anybody who signs up for a Pink Visual website from one of these links will count as an affiliate sale for you.

Check out Topbucks.com for more information on the affiliate program.

Note: Your webmaster id will only be attached to links generated for an "Approved" api key. If you're still using a "Development" level key, your revid will not appear.

CHAPTER

EIGHT

PLUGINFEEDS

If you're building a complete adult site using the PV API, you might want some real adult content to fill it. The images you get with *episode requests* can be explicit, but you can also get *full videos*.

To get full videos, you must first link a Pluginfeeds account to your PV API account. Once you have a Pluginfeeds account linked up, you can use the `paid content request` to get URLs to full videos.

How to link your Pluginfeeds account:

1. Find your customer id and access code here: http://customer.pluginfeeds.com/.
2. You will also need to add your API-based domain to the "allowed referrers" list on that page.
3. Log into PV API.
4. Go to your settings page.
5. Enter your Pluginfeeds customer id and access code.

Note: Any bandwidth used by people viewing videos from the `paid content` URLs will be charged to your Pluginfeeds account.

Check out Pluginfeeds.com for more information on adult video feeds.