



The FeedBlitz API

Version 1.0
26 October 2006

Easily enabling powerful integrated email subscription management services using XML, HTTPS and REST



The FeedBlitz API

V1.0 – October 26, 2006

The FeedBlitz API	i
Copyright	ii
Disclaimer	ii
About FeedBlitz	iii
What is the FeedBlitz API?.....	1
API Design and Usage Considerations	1
Privacy and Security	1
Accessing the API.....	2
Example usage:	3
Basic FeedBlitz API XML.....	3
API Methods	4
GET – Read Data	4
POST – Edit and Search	4
DELETE – Resource Removal	5
PUT – Add a New Resource	5
Sample API Interactions	6
Using FeedBlitz Functionality: Adding Email Services to Your Online Application.....	6
Adding Custom Validation to Email Signup	6
Testing the API	7
FeedBlitz API RESOURCE REFERENCE GUIDE	8
Subscribers (Subscriber Management)	8
GET – Fetching Subscriber Information.....	9
POST - Searching and sorting subscribers.....	9
POST – Editing Subscribers	10
DELETE – Deleting a subscriber’s subscription	11
PUT - Adding a new subscriber.....	12
Subscriptions (Managing the Updates you Receive).....	12
GET – Fetching Subscription Information	13
POST - Searching Subscriptions.....	14
POST – Editing Subscriptions	14
DELETE – Deleting a subscription	15
PUT - Adding a new subscription.....	15
Syndications.....	16
GET – Fetching Syndication Data	18
Summarizing Syndications	18
POST – Searching Syndications	19
POST – Editing Syndications	20
DELETE – Deleting Syndications	20
PUT – Creating a New Syndication.....	20
Captcha (Image Verification)	21
User (Account registration and Subscription Signup)	21
GET – About the Current User	21
POST – Updating the Current User	22

PUT – Registering a New User Account	23
PUT – A New Subscription to an Existing Syndication	23
DELETE – Deleting a User	24
Autoresponders	24
Time Zones	25
Dynamic Ad and Metric Insertion	25
Available Content Providers	25
A Syndication’s Dynamic Content Settings	26
GET – Fetching Dynacontent Information	26
API Support	27

Copyright

This information is copyright © FeedBlitz, LLC www.feedblitz.com All rights reserved.

Disclaimer

YOU EXPRESSLY AGREE THAT YOUR USE OF THIS INFORMATION AND THE FEEDBLITZ SOFTWARE SERVICES IS AT YOUR OWN RISK. NEITHER FEEDBLITZ, ITS AFFILIATES, NOR ANY OF THEIR RESPECTIVE OFFICERS, DIRECTORS, MEMBERS, SHAREHOLDERS, EMPLOYEES, AGENTS, THIRD PARTY CONTENT PROVIDERS, OR LICENSORS WARRANT THAT THE SERVICE, CONTENT OR FEEDBLITZ SOFTWARE WILL BE UNINTERRUPTED, TIMELY, SECURE OR ERROR-FREE; NOR DO THEY MAKE ANY WARRANTY AS TO THE RESULTS THAT MAY BE OBTAINED FROM USE OF THE SERVICE, CONTENT OR FEEDBLITZ SOFTWARE INCLUDING THEIR ACCURACY, RELIABILITY, QUALITY, ADEQUACY, TIMELINESS OR AUTHENTICITY. NOR DO THEY MAKE ANY WARRANTY AS TO THE ACCURACY, RELIABILITY, QUALITY, ADEQUACY TIMELINESS OR AUTHENTICITY OF ANY CONTENT, INFORMATION, SERVICE, PRODUCTS, MERCHANDISE OR OTHER MATERIAL PURCHASED PROVIDED BY OR THROUGH THE SERVICE. 2. THE SERVICE, FEEDBLITZ SOFTWARE AND CONTENT IS PROVIDED ON AN "AS IS" AND "AS AVAILABLE" BASIS WITHOUT WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. 3. THIS DISCLAIMER OF LIABILITY APPLIES TO ANY DAMAGES OR INJURY CAUSED BY ANY FAILURE OF PERFORMANCE, ERROR, OMISSION, INTERRUPTION, DELETION, DEFECT, DELAY IN OPERATION OR TRANSMISSION, COMPUTER VIRUS, COMMUNICATION LINE FAILURE, THEFT OR DESTRUCTION OR UNAUTHORIZED ACCESS TO, ALTERATION OF, OR USE OF RECORD, WHETHER FOR BREACH OF CONTRACT, TORTIOUS BEHAVIOR, NEGLIGENCE, OR UNDER ANY OTHER CAUSE OF ACTION REGARDLESS OF WHETHER FEEDBLITZ HAD NOTICE OF THE CAUSE OR SUCH CAUSE WAS FORESEEABLE. 4. IN NO EVENT WILL FEEDBLITZ, OR ANY PERSON OR ENTITY INVOLVED IN CREATING, PRODUCING OR DISTRIBUTING THE SERVICE, CONTENT OR THE FEEDBLITZ SOFTWARE, BE LIABLE TO YOU OR ANY OTHER PERSON OR ENTITY FOR ANY DAMAGES, INCLUDING (WITHOUT LIMITATION) DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, LOSS

OF PROFIT OR REVENUE OR PUNITIVE DAMAGES ARISING OUT OF THE USE OF OR INABILITY TO USE THE SERVICE, CONTENT OR FEEDBLITZ SOFTWARE. 5. THESE DISCLAIMERS AND LIMITATIONS SHALL APPLY EVEN IN THE EVENT OF A FUNDAMENTAL OR MATERIAL BREACH OR A BREACH OF THE FUNDAMENTAL OR MATERIAL TERMS OF THESE TERMS OF SERVICE. 6. Some jurisdictions do not allow the exclusion of certain warranties or the limitation or exclusion of liability for incidental or consequential damages. Accordingly, some of the above limitations may not apply to you.

About FeedBlitz

FeedBlitz is a service that monitors blogs, RSS feeds and Web URLs to provide greater reach for feed publishers. FeedBlitz takes all the headache out of converting feed and blog updates into email digests, delivered daily to subscribers' inboxes. FeedBlitz manages subscriptions, circulation tracking, testing, and is compatible with all major blogging platforms and services such as Blogger, TypePad and FeedBurner.

FeedBlitz enables end users to monitor any feed or blog, anonymously if they wish, regardless of whether the publisher of that feed is using FeedBlitz. FeedBlitz therefore provides a simple way for users to receive updates from their trusted sources using a familiar and ubiquitous medium - email.

Visit FeedBlitz online at www.feedblitz.com

What is the FeedBlitz API?

The FeedBlitz API enables users to access FeedBlitz features programmatically, allowing innovative new applications based on and including FeedBlitz data and functionality. The API enables comprehensive integration at the data and application layers using a consistent and easy to understand interface.

Applications of the API vary in their potential complexity, and might include:

- Most recent subscriber displays on a web site.
- Integrating, embedding and hiding FeedBlitz signup with existing forms and web sites.
- Automatically creating FeedBlitz subscription *and* syndication accounts.
- Enabling email syndications and subscriptions to your content programmatically.
- Integrating FeedBlitz in Web 2.0 “mashups”.
- Creating RSS and OPML services based on your FeedBlitz subscriptions or syndications.
- Providing a branded or innovative management interfaces, e.g. for mobile users, IM clients, corporate web sites and intranets.

Some of these applications are expanded on in the sections below.

The API itself is a REST (representational state transfer) API, using XML and HTTP. The caller sends HTTP GET and POST requests as documented below to the API server, and they are returned an XML document containing the appropriate response.

API Design and Usage Considerations

The API is designed to minimize the number of queries the caller needs to make to generate a useful dataset. As such, it is a relatively rich interface giving the user access to many core FeedBlitz resources. In particular, some complexity has been added to “join” resources at runtime to return comprehensive information in a single interaction, thus avoiding the need to repetitively call the API for one resource and then for a series of related objects. It is designed, then, to be efficient in terms of minimizing the number of client – server round trip interactions.

As such, some API calls may be computationally and bandwidth extensive. *API users are strongly encouraged to ensure they call the API with the narrowest scope possible* required to fulfill their needs, and not to call it excessively. For example, the feeds a user is publishing will typically not change very often; there is therefore no need to call the API every minute to query the user’s syndications. An hourly or daily check is usually more than enough, using a cached copy for local serving for the user’s purposes.

Privacy and Security

The API may not be used to overcome FeedBlitz’s privacy and security policies. API users are bound by the same terms of service and privacy policies as the core FeedBlitz.com site (see <http://www.feedblitz.com/tos.asp>). For example:

- You may not use the API to determine the email of an anonymous subscriber.
- You may not use the API to discern information about resources unrelated to your account.
- You may not use the API to subscribe a user against their will.
- You may not use the API alter or change a different user's email, password or resources they control.

Some features are not available via the API by design for security or anti-abuse reasons. This includes subscriber import, which must be performed via the feedblitz.com user interface.

If FeedBlitz determines what appear to be attempts to abuse this API, such activity will result in the immediate termination of your FeedBlitz account, without refund (if applicable).

All API calls require authentication to the API server on each and every call. An authentication failure will return an error code to the caller. The results returned, like in the feedblitz.com GUI, are restricted in scope to the entities that the authenticated user has access to. Attempts to access entities outside of the caller's scope will return a failure code and / or empty resource XML.

The API is open to all users and publishers. In its initial release it will not be limited in terms of the number of calls allowed.

Accessing the API

The API is called by accessing a resource path (see the documentation below) as follows:

<https://api.feedblitz.com/f.api/resourcepath>

Access Notes:

- 128-bit SSL is required; plaintext HTTP requests will be rejected.
- The API requires basic authentication using your FeedBlitz login email address and password (i.e. the same credentials used at www.feedblitz.com)

The HTTP verbs allowed are:

- GET Return information about the resource
- POST Search, edit or update the resource
- PUT Create a new resource
- DELETE Delete the resource

No other verbs are supported. The extent to which each resource supports one or more of these verbs is documented in each resource's reference section below. Each call to the API is *stateless*; it is up to the caller to maintain any state variables necessary for their application. Authentication information is required on each and every call to the API.

Notes

- Any path entries beyond a valid path will simply be ignored.
- Anonymous users will not show their email addresses or IDs to the caller *unless* the API's caller is that user themselves, in which case the fields will be correctly populated.

Important:

- As FeedBlitz features are added other tags and attributes may appear in the returned XML. You should expect unknown or undocumented tags and handle them appropriately.
- A PUT is *not* the same as a POST if applied to an existing resource. A PUT to an existing resource will fail.
- Any changes made by a POST or DELETE are final and immediately committed. It may not be possible for you to undo any of these changes, even if you have a cached copy of the data you wish to restore. *The API assumes that the caller has made all the appropriate application checks prior to the API being called when modifying data.*

FeedBlitz, LLC is not liable for any data loss or damages you may incur by using this API. Use of the API implies no warranty or guarantees about reliability, performance or applicability, and is subject to the FeedBlitz standard Terms of Service at www.feedblitz.com/tos.asp

Example usage:

`https://api.feedblitz.com/feedblitz/subscriptions`

This call returns the XML containing all the subscriptions (if any) for the currently authenticated user.

Basic FeedBlitz API XML

XML sent or received is sent using Unicode UTF-8 encoding. All XML is wrapped within <feedblitz> API tags as follows

```
<feedblitzapi version="1.0" >
. resource specific XML
.</feedblitzapi >
```

The version number is required and MUST be 1.0.

API resources usually follow the following scheme:

```
/resource class/resource identifier/resource class/identifier
```

So this is a valid resource:

```
/syndications/57/subscribers/124
```

It represents subscriber number 124's subscription to syndication 57. Using different API methods, it is possible to access, alter and delete this resource (provided you have the privileges to do so), as well as add new resources.

API Methods

GET – Read Data

A GET for the above resource will return the XML for subscriber 124's subscription to syndication 57. This call will only be successful if the authenticated user is publishing syndication number 57, and if subscriber 124 is in fact subscribed to it. If the call is unsuccessful the returned XML will contain an empty <subscription /> tag (if the user does not own syndication 57) or a populated <subscription> tag with an empty <subscriber /> tag if subscriber 124 does not subscribe to feed 57.

It is possible to use shorter but complete paths as follows:

Path	Comment
/syndications	Returns all the user's published feeds
/syndications/57	Returns the user's published feed # 57
/syndications/57/subscribers	Returns all the subscribers to feed 57
/syndications/57/subscribers/124	Subscriber 124's subscription to feed 57

Note that /syndications/subscribers is an *invalid* resource. To fetch all their subscribers, a user should simply GET the /subscribers resource.

Also note that you should only use the paths documented here. Some resource paths you may expect to work are in fact invalid and not supported, such as /subscribers/124/subscriptions/57. It is in fact just the same as /syndications/57/subscribers/124 which can be used instead. /subscribers/124 is valid, returning all 124's subscriptions that this called syndicates.

POST – Edit and Search

If supported, you can POST edits at any point in the path structure. The results will depend on the path used and the data in the POST. Use the returned XML to judge the success or otherwise of your edit; do not rely on the HTTP return code (which will always be 200 OK in this release).

In this example, a POST at each point in the hierarchy behaves as follows:

Path	Comment
/syndications	Updates all the syndication entities included in the POST data
/syndications/57	Only updates feed #57. Fails if any other feed id is specified.
/syndications/57/subscribers	Edits the subscribers included in the POST data for feed 57
/syndications/57/subscribers/124	Updates subscriber 124's subscription to 57, if the IDs match.

The results of a POST are typically (but not always – see the documentation) the same as a GET for the specified resource.

POST is also used for searching and sorting result sets. Where supported, replace the ID with the literal ‘search’:

Path	Comment
/syndications/search	Search and sort this user’s syndications
/syndications/57/subscribers/search	Search the subscribes to feed 57

Not all resources support searching. See the documentation below for resource-specific information.

DELETE – Resource Removal

DELETES may be issued to remove the associated resources from the system. Resources are typically not physically removed from FeedBlitz; rather they are marked as deleted. Deleted resources are retrieved, if present, by a GET statement, so be sure not to present them to end users if that would be confusing or inappropriate. The scope of a DELETE is the same as for a GET; it works at the level specified by the resource path. Use DELETE with care, and ensure that the user has confirmed their desire to make the relevant changes before you call the API. Use the returned XML, not the HTTP return code, to evaluate the success of the requested operation.

Path	Comment
/syndications	Deletes all published feeds, and all the subscriptions to them.
/syndications/57	Deletes feed # 57 and removes any subscriptions.
/syndications/57/subscribers	Deletes all the subscribers to feed 57, but leaves the syndication itself alone.
/syndications/57/subscribers/124	Deletes subscriber 124’s subscription to feed 57 only.

PUT – Add a New Resource

New resources are added by using a PUT at the relevant level in the path:

Path	Comment
/syndications	Add a new syndication
/syndications/57/subscribers	Add a new subscriber to feed 57

A PUT, like a POST, typically returns the full resource XML (as if a GET were performed on the newly created entity). Exceptions are made for privacy reasons in the /user resource, where only the email address of the user (if the user is non anonymous) is returned to the caller along with a status code indicating whether the operation was successful or not.

Important:

- A PUT on an existing resource is NOT treated by the FeedBlitz API as being the same as a POST to that resource. A PUT on an existing resource will not edit that resource's settings by design. You MUST use the POST method to alter an existing resource's variables.
- There are currently no DTDs or other formal online XML documentation for validating XML documents used by the API.
- Not all methods are supported by all resources. Don't assume.

Sample API Interactions

Using FeedBlitz Functionality: Adding Email Services to Your Online Application

Say you want to embed FeedBlitz capabilities into an existing form on your web site. You therefore need to present the FeedBlitz image verification test as part of your form, as the FeedBlitz API does not allow subscriptions or accounts to be created without passing the image verification test. This is to prevent automated submissions from spambots, and is the approach used on the feedblitz.com web site. Here's how you would do it (all these examples assume that you authenticate properly to the API server):

- 1) GET the /captcha resource to generate the challenge key and the path to the verification image.
- 2) Insert these values into your form and create a field in your form to capture the user's response.
- 3) POST the response and other captcha / subscription / registration data back to the appropriate API resource.
- 4) Check the return document for success, warning or failure codes.

If the image verification was passed and the API validated that the feed is available for subscriptions, then the user will have been added to the subscription and an appropriate activation email sent to them automatically by FeedBlitz. If not, the returned document will contain another captcha challenge for you to present to the user.

This basic technique, depending on the resource you PUT to and the data delivered, enables you to:

- Automatically start the subscription process for your site's registrants, or
- Automatically set up FeedBlitz email services for users of your application or service by creating a publication (syndication) account.

Adding Custom Validation to Email Signup

Say your site is required to validate the user's age, but the out of the box validation question FeedBlitz provides is not enough. (e.g. to satisfy the US COPPA regulations you might want to

verify a user's age by having a form where they enter their birth date, rather than simply asking them to confirm they are 13 or over).

In this case you would present your own signup form, along with the FeedBlitz verification image as pulled using the API. Once the user has passed your validation code from your custom form, you pass the email address and the response to the image verification challenge to the API to sign them up for your content.

Testing the API

The best way to get started is to access your user information. Simply type in the URL for your ID into your browser to GET the XML document:

<https://api.feedblitz.com/f.api/user>

Enter your FeedBlitz user name and password in your browser's security dialog, and the XML for your information will appear. How that information is presented to you depends on your browser; the FeedBlitz API does not provide an XML stylesheet as part of the response.

FeedBlitz API RESOURCE REFERENCE GUIDE

Subscribers (Subscriber Management)

Core resource path: `/subscribers`

`/subscribers` returns all the subscribers to all the user's feeds. Each subscriber's XML record includes basic information about the subscriber and the user's feeds to which they are subscribing. Here's an example reply, showing that there is one (`<count>`) subscriber in this set of resources. This subscriber has two subscriptions syndicated by this API user, one to feed 90291 and one to feed 96755. The subscriber may have subscriptions to other syndications; they are not shown to this caller.

```
<?xml version="1.0" encoding="utf-8" ?>
<feedblitzapi version="1.0" xmlns:link="http://www.w3.org/1999/link">
  <rsp stat="ok">
    <success code="0" msg="Authorized" />
  </rsp>
  <subscribers>
    <count>1</count>
    <subscriber>
      <id>21</id>
      <email>phil@hollows.com</email>
      <status>ok</status>
      <count>2</count>
      <subscription>
        <id>90291</id>
        <name>CNN.com</name>
        <description>CNN.com delivers up-to-the-minute news and
        information on the latest top stories, weather,
        entertainment, politics and more.</description>
        <link>http://rss.cnn.com/rss/cnn_topstories.rss</link>
        <created>2006-08-21 11:37:38 -0500</created>
        <updated>2006-08-22 09:56:51 -0500</updated>
        <status>unsubscribed</status>
        <liststatus>ok</liststatus>
        <referer />
        <ip />
      </subscription>
      <subscription>
        <id>96755</id>
        <name>CNNMoney.com</name>
        <description>From CNN and Money magazine, CNNMoney.com
        combines business news and in-depth market analysis with
        practical advice and answers to personal finance
        questions.</description>
        <link>http://rss.cnn.com/rss/money_topstories.rss</link>
        <created>2006-08-21 11:37:38 -0500</created>
        <updated>2006-09-06 16:52:28 -0500</updated>
        <status>ok</status>
        <liststatus>ok</liststatus>
        <referer>http://foo.com/f/f.fbz?NewList</referer>
        <ip>65.96.75.0</ip>
      </subscription>
    </subscriber>
  </subscribers>
</feedblitzapi>
```

Note that each resource class contains a `<count>` of the number of resources it contains; this is to simplify API uses that present summary information. In the above sample there are two `<count>` tags. The first indicates that the `<subscribers>` block contains 1 `<subscriber>`, and the second

that the <subscriber> block contains two <subscription>s. Note that there is a <subscribers> (plural) block containing one or more <subscriber> (singular) entries.

Tag	Comment
<id>	The subscriber's unique identifier in FeedBlitz. If the subscriber has elected to be anonymous, this will be an empty <id /> tag.
<email>	The subscriber's current email address. If the subscriber has elected to be anonymous, this value will be "anonymous" and not a valid email address.
<status>	The subscriber's current system status.

<subscription> resources are fully described in the subscription section below.

If a subscriber is listed as anonymous their email address will appear as "anonymous"

GET – Fetching Subscriber Information

Valid resource paths that return subscriber XML are:

Path	Comment
/subscribers	Returns all the user's subscribers and subscriptions
/subscribers/124	All subscriber # 124 subscriptions
/syndications/57/subscribers	Returns all the subscribers to feed 57
/syndications/57/subscribers/124	Subscriber 124's subscription to feed 57

If you request a resource that does not exist (say subscriber 124 does not actually subscribe to your syndication 57) you will be returned an empty <subscribers /> tag.

POST - Searching and sorting subscribers

You may POST to a subscriber URL replacing the subscriber ID with the word "search" to scan the user's subscriber database, as follows:

Path	Comment
/subscribers/search	Search all subscribers
/syndications/57/subscribers/search	Search subscribers to feed 57

XML is posted to the resource to indicate the search and sort parameters, as follows:

```
<feedblitzapi version="1.0" xmlns:xml="http://www.w3.org/1999/xml">
  <searchpattern>foo</searchpattern>
  <searchstatus>undeclared</searchstatus>
  <sort order="desc">email</sort>
  <limitstart>250</limitstart>
  <limitcount>500</limitcount>
</feedblitzapi>
```

Tag	Comment
<searchpattern>	Performs a literal (i.e. not keyword) search for the specified text in the subscriber's email address.

<searchstatus>	Returns subscribers with that given status.
<sort>	<p>Sorts by the specified field. Specify “desc” as the order attribute to sort backwards; otherwise sorts will be returned in ascending order regardless. The default sort is by subscriber id. Valid sort values are:</p> <ul style="list-style-type: none"> • id sort by subscriber id • email sort by email address • status sort by subscriber’s system status • created sort by subscription create date • updated sort by subscription updated date
<limitstart>	Only return results from this count forward. If omitted zero (the beginning) is assumed. The first row as a row number of zero.
<limitcount>	Restrict the number of returned objects this number. So the example above will return rows 250 to 749 inclusive, if they exist. Note that the results returned may not return <limitcount> individual subscribers; rather <limitcount> subscriptions are returned.

Note that when sorting by subscriber fields, all the appropriate subscription data is placed in a single subscriber record – there will be exactly one subscriber record in the returned result set.

If, however, you sort by a subscription date field (created / updated) then the subscribers are presented in subscription date order. There therefore may be multiple <subscriber> blocks for the same subscriber in the returned XML (depending on the search parameters). It is also possible, depending on the date values, for any one <subscriber> record to have multiple <subscription> records if that subscriber happens to have consecutive subscriptions in the date sorted result set.

If both <searchpattern> and <searchstatus> is specified both conditions must be satisfied (i.e. the search is <searchpattern> AND <searchstatus>). You may omit or specify empty tags for search parameters you do not care about, so the following is a valid search for all subscriber emails containing “noemail.org”:

```
<feedblitzapi version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink">
  <searchpattern>noemail.org</searchpattern>
</feedblitzapi >
```

The following search returns confirmed subscribers in reverse chronological order, and so could be used to provide “welcome to our newest subscriber” data on a web site.

```
<feedblitzapi version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink">
  <searchstatus>ok</searchstatus>
  <sort order="desc">updated</sort>
</feedblitzapi >
```

POST – Editing Subscribers

You may POST to change a subscription’s state, with the following restriction: you may not change a subscriber with a deleted, unsubscribed or ignored status. If you post a field other than status it will be ignored.

The following XML sets subscriber 17's state to "deleted"

```
<feedblitzapi version="1.0">
  <subscribers>
    <subscriber>
      <id>17</id>
      <status>deleted</deleted>
    </subscriber>
  </subscribers>
</feedblitzapi >
```

You can use this XML in two ways, removing a subscriber from one or multiple lists in a single operation:

- To remove subscriber 17 from ALL your syndications, post the above XML to the `/subscribers` resource path
- To remove subscriber 17 from syndication 57, post the above XML to `/syndications/57`

If you wish, you may omit the subscriber number from the XML and post it to the subscriber-specific path as follows (since the subscriber ID is determined by the path):

```
<feedblitzapi version="1.0">
  <subscribers>
    <subscriber>
      <status>deleted</deleted>
    </subscriber>
  </subscribers>
</feedblitzapi >
```

- To remove subscriber 17 from ALL your subscriptions, post the above XML to the `/subscribers/17` resource path
- To remove subscriber 17 from syndication 57, post the above XML to `/syndications/57/subscribers/17`

If an ID is specified in the XML and in the PATH, the ID specified in the PATH will be used. No warning will be given in the response.

After a POST has been made to edit subscribers, you will be returned the appropriate subscriber XML for resource specified in the path.

DELETE – Deleting a subscriber's subscription

A DELETE is analogous to a POST setting the status to "deleted" (indicating that you the owner of the list removed the subscriber, as opposed to the subscriber unsubscribing themselves). You may DELETE in the following resources:

Path	Comment
<code>/subscribers</code>	Deletes all the subscribers to all your syndications
<code>/subscribers/17</code>	Deletes subscriber 17 from all your syndications
<code>/syndications/57/subscribers</code>	Deletes all the subscribers to syndication 57, but leaves the syndication in place.

/syndications/57/subscribers/17	Deletes subscriber 17 from syndication 57
---------------------------------	---

After a delete, you will be returned the XML for the relevant resource.

PUT - Adding a new subscriber

You may not PUT to a subscriber resource. To create a new subscription you must PUT to the /user resource instead. See the documentation for the /user resource for important information about starting the subscription process for a syndication that you manage using the API.

Subscriptions (Managing the Updates you Receive)

Core resource path: [/subscriptions](#)

Returns data about a user's subscriptions (i.e. the feeds to which they themselves are subscribing), or an empty <subscriptions /> tag. Information is provided about the subscription itself, but no publisher information is provided in order to protect the privacy of the publisher. Sample XML is below.

```
<?xml version="1.0" encoding="utf-8" ?>
<feedblitzapi version="1.0" xmlns:link="http://www.w3.org/1999/link">
  <rsp stat="ok">
    <success code="0" msg="Authorized" />
  </rsp>
  <subscriptions>
    <count>1</count>
    <subscription>
      <id>90291</id>
      <name>CNN.com</name>
      <description>CNN.com delivers up-to-the-minute news and
information on the latest top stories, weather, entertainment,
politics and more.</description>
      <link>http://rss.cnn.com/rss/cnn_topstories.rss</link>
      <created>2006-08-21 11:37:38 -0500</created>
      <updated>2006-08-22 09:56:51 -0500</updated>
      <status>unsubscribed</status>
      <liststatus>ok</liststatus>
      <referer>http://www.cnn.com/</referer>
      <ip>1.2.3.4</ip>
    </subscription>
  </subscriptions>
</feedblitzapi>
```

Tag	Comment
<count>	Number of <subscription> objects in a <subscriptions> block
<id>	The subscription's unique FeedBlitz ID
<name>	The name of the feed as defined to FeedBlitz. The feed's polled XML may be different.
<description>	The description of the content as defined to FeedBlitz. The polled XML may differ.
<link>	The URL polled by FeedBlitz
<created>	Timestamp when the feed was added to FeedBlitz
<updated>	Timestamp of the last status change (e.g. from pending to ok)

<status>	The feed's current status (pending / ok / paused / deleted / finished / ignored) as set by the subscriber
<liststatus>	The subscription's current system status as of the last poll (e.g. ok / fail)
<referrer>	<p>The source of the subscription. This may be:</p> <ul style="list-style-type: none"> • Blank – no referrer was found, or the subscription was added prior to FeedBlitz tracking referrer URLs • URL – the referring URL as reported by the user's client • Import: <text> - the subscriber was imported. The text describes the IP and the FeedBlitz user ID used to perform the import. • Autosyndication: <URL> The subscription was added automatically because the user is subscribed to a syndicated OPML (reading list / blogroll) file • Autosubscription: <URL> The subscription was added automatically because the user had subscribed themselves to an OPML (reading list / blogroll) file • Autoresponder: feedid The subscription is an autoresponder, associated with the specified feed. <p>Note: We retain HTTP's mis-spelling of the word referrer here, using a single "r"</p>
<ip>	The IP octet of the host creating (not updating) this subscription. May be blank if the subscription was created prior to FeedBlitz tracking subscription IPs.
<uid>	User ID required to access the URL. Only returned if you are the owner of the subscription.
<pwd>	Password required to authenticate to the remote URL. Again, only displayed if you are the owner of the subscription.
<includetags>	Comma separated list of tags to include in the email. These tags are applied only to posts that pass the publisher's tags (if any)
<excludetags>	Comma separated list of tags to exclude from the email. These tags are applied only to posts that pass the publisher's tags (if any)
<excludeifboth>	By default, if an include and exclude tag are present, the post is included. This is "1" if the post should be excluded. Applies to the subscriber's tags only, and only to posts filtered in by the publisher's tags.

It is easy to convert this output using a script or XSL into an RSS feed, HTML page or OPML blogroll for a user based on this output. For most users there should be no reason to check this content more than daily, unless the caller knows that a subscription change has just been made.

Important

- Premium subscription information is not currently provided.

GET – Fetching Subscription Information

Valid paths returning subscription XML are:

Path	Comment
/subscriptions	Returns all the user's subscriptions
/subscriptions/57	Returns the user's subscription to feed # 57

POST - Searching Subscriptions

Like some other resources, Subscriptions support searching via POST of an XML document to the resource URL:

```
<feedblitzapi version="1.0" xmlns:link="http://www.w3.org/1999/link">
  <searchpattern>foo</searchpattern>
  <searchstatus>fail</searchstatus>
  <searchlink>http://foo.com</searchlink>
  <sort order="desc">updated</sort>
</feedblitzapi >
```

Tag	Comment
<searchpattern>	Performs a literal (i.e. not keyword) search for the specified text in the subscription's name, description or URL.
<searchstatus>	Returns subscriptions with the given status. Valid feed status values are: <ul style="list-style-type: none"> • ok • fail • paused • deleted
<searchlink>	Searches for an exact match for the URL. Since URL's are unique to any one user within FeedBlitz, it makes no sense to specify <searchlink> and <searchpattern>.
<sort>	Sorts by the specified field. Specify "desc" as the order attribute to sort backwards; otherwise sorts will be returned in ascending order regardless. The default sort is by list id. Valid sort values are: <ul style="list-style-type: none"> • id sort by subscriber id • status sort by subscriber's system status • created sort by subscription create date • updated sort by subscription updated date • link sort by subscription URL • description sort by the feed's description

Specifying more than one search criteria (e.g. <searchpattern> and <searchstatus>) requires that all conditions to be satisfied for a subscription to be returned, i.e. the search is an AND operation.

POST – Editing Subscriptions

You may POST to change a subscription's state. The behavior and values you may change with a POST depend on the nature of the subscription you are editing.

- If you are subscribing to a different user’s syndication, only the status will be changed.
- If you are subscribing to your own syndication, you may change the name, link, uid, pwd and status values. Your syndication will be updated with these values.
- If your subscription is not a FeedBlitz syndication, you may change the name, link, uid, pwd and status values.

The XML you post may have multiple <subscription> tags within the single <subscriptions> collection. In addition, you may operate on the following resources:

Path	Comment
/subscriptions	Changes all your subscriptions as specified in the XML
/subscriptions/123	Changes subscription # 123.

If an ID is specified in the XML and in the PATH, the ID specified in the PATH will be used. No warning will be given in the response.

After a POST has been made to edit subscribers, you will be returned the appropriate subscriber XML for resource specified in the path.

To unsubscribe from a feed, set its status to “unsubscribed.”

DELETE – Deleting a subscription

A DELETE is analogous to a POST setting the status to “deleted” If you wish to stop receiving updates from a subscription (including one that you are syndicating), you may delete it or POST an “unsubscribed” status to the resource.

You may DELETE in the following resources:

Path	Comment
/subscriptions	Deletes all your subscriptions
/subscriptions/17	Deletes subscription 17

After a delete, you will be returned the XML for the relevant resource.

PUT - Adding a new subscription

Using PUT creates a new, independent subscription in FeedBlitz for the caller alone. *This method should only be used where there is no existing FeedBlitz syndication for the user to subscribe to.* This is the method used in FeedBlitz.com when a user sets up a new subscription from the FeedBlitz dashboard itself. This is NOT the method used when the user enters their email address in your signup dialog on your website (that’s the PUT to the \user resource).

So, if you expect more than one user to subscribe to any given feed in FeedBlitz, or you want to syndicate this feed to others, do not use this method.

Instead, create a new Syndication resource (if necessary) and then PUT to the appropriate /user resource to subscribe the user to it. This latter approach greatly simplifies management tasks, since updating the subscribers is then simply a matter of changing the relevant /syndication resource, as opposed to having to modify multiple individual distinct subscriptions. Using /syndications also makes all the subscribers to a feed appear in a single location in the FeedBlitz GUI.

If you do need to create a standalone subscription, PUT to this resource. If you are calling on behalf of a third party client, you should specify the (client) IP and referrer fields if applicable; otherwise the system will attempt to derive them from the runtime environment.

You may PUT any field to the API. You may omit fields that do not apply, but <name> and <link> are required. Date fields will be ignored and the actual times inserted.

Syndications

Core resource path: [/syndications](#)

Returns data about the user's current published syndications, or an empty <syndications /> tag. If a syndication is found, comprehensive information is provided (including summary subscriber metrics) to avoid further API queries. A summary version is also available (see the "summarizing syndications" section below).

```
<?xml version="1.0" encoding="utf-8" ?>
<feedblitzapi version="1.0" xmlns:link="http://www.w3.org/1999/link">
  <rsp stat="ok">
    <success code="0" msg="Authorized" />
  </rsp>
  <syndications>
    <count>1</count>
    <syndication>
      <id>33369</id>
      <publisherid>21</publisherid>
      <publisheremail>phil@hollows.com</publisheremail>
      <name>FeedBlitz Widgets</name>
      <description>FeedBlitz Widgets a test feed</description>
      <link>http://foo.com/foo/atom.xml</link>
      <created>2006-02-24 23:03:11 -0500</created>
      <status>ok</status>
      <liststatus>ok</liststatus>
      <turbo />
      <uid />
      <pwd />
      <notifyowner>1</notifyowner>
      <tease>Read the whole entry</tease>
      <entrylength />
      <keepprivate>1</keepprivate>
      <type>Atom 1.0</type>
      <linktrack>0</linktrack>
      <opentrack>0</opentrack>
      <isauto>0</isauto>
      <autoresponderid>0</autoresponderid>
      <validationquestion />
      <validationfail />
      <includetags />
    </syndication>
  </syndications>
</feedblitzapi>
```

```

<excl udetags />
<excl udei fboth />
<addcomments>0</addcomments>
<forwardemail >1</forwardemail >
<timezoneid>35</timezoneid >
<dynacontent>
  <id>103</id >
  <type>0</type>
  <providerid>2</providerid >
  <value>pub-xxx</value>
</dynacontent>
<dynacontent>
  <id>140</id >
  <type>1</type>
  <providerid>4</providerid >
  <value>UA-xxx-1</value>
</dynacontent>
<subscribersummary>
  <subscribers status="deleted">2</subscribers>
  <subscribers status="ignored">1</subscribers>
  <subscribers status="ok">3</subscribers>
</subscribersummary>
</syndication>
</syndications>
</feedblitzapi >

```

Entries are:

Tag	Comment
<count>	Number of <syndication> objects in a <syndications> block
<id>	The syndication's unique FeedBlitz ID
<publisher>	The subscriber ID of the publisher (blank if anonymous)
<publisheremail>	The publisher's address (or "anonymous")
<name>	The name of the feed as defined to FeedBlitz. The feed's polled XML may be different.
<description>	The description of the content as defined to FeedBlitz. The polled XML may differ.
<link>	The URL polled by FeedBlitz
<created>	Timestamp when the feed was added to FeedBlitz
<status>	The feed's current status (ok / paused / deleted) as set by the owner
<liststatus>	The feed's current system status as of the last poll (e.g. ok / fail)
<turbo>	The syndication's turbo frequency, if any. -2 indicates a standard (free service), as does zero or empty. 1440 indicates an active time zone override. -3 indicates on-demand, -5 Express. Other positive values indicate the number of minutes between each poll cycle.
<uid>	User id if required by the poller for feeds needing authentication
<pwd>	Password if required to access an feed needing authentication
<notifyowner>	Whether the user wishes to receive (un)subscribe email notifications
<tease>	Call to action text if the user is truncating feed content
<entrylength>	Maximum # of characters of an entry. If 0, only headlines are sent.
<keepprivate>	Do not show in FeedAdvisor results or other public places
<type>	Type of feed at create time. Not updated by the system and not to be relied upon.
<linktrack>	1 if the user wishes to track click-throughs
<opentrack>	1 if the user wishes to track email opens

<isauto>	1 if this syndication may be used as an autoresponder
<autoresponderid>	The ID of the feed this feed uses as an autoresponder
<validationquestion>	User must confirm this text for a subscription to succeed
<validationfail>	Text to present if the user does not confirm the verification
<subscriberssummary>	Block of tags summarizing subscriber info for this feed
<includetags>	Comma separated list of tags to include in the email
<excludetags>	Comma separated list of tags which should not be in the mail
<excludeifboth>	By default, if an include and exclude tag are present, the post is included. This is "1" if the post should be excluded.
<addcomments>	1 if the user has comments links added to their outbound emails.
<forwardemail>	1 if the user has "email to a friend" enabled in the outbound emails.
<timezoneid>	The time zone ID of the feed; used to determine the time and scope of a "nightly" run.
<dynacontent>	Automatically inserted content for ads and site metrics.
<subscribers status="">	Count of subscribers with this status.

The <subscribers> summary tag only shows the # of subscribers present. So if there are no pending registrations there will be no <subscribers status=""> entry (as is the case in the example record above). Note that the <subscribers> tag here is not the same as the <subscribers> resource documented elsewhere in this document. If the called requires complete subscriber information they can access the appropriate /subscribers resource for the relevant syndication(s).

It is easy to convert this output using a script or XSL into an RSS feed, HTML page or OPML blogroll for a user based on this output. For most users there should be no reason to check this content more than daily, unless the caller knows that a subscription change has just been made.

Important

- Premium feed information is not currently provided.

GET – Fetching Syndication Data

Valid paths returning syndication XML are:

Path	Comment
/syndications	Returns all the user's published feeds
/syndications/57	Returns the user's published feed # 57

See the subscribers section for additional resource URLs that are syndication-specific.

Summarizing Syndications

For some needs, such as presenting a GUI with just the name, description and URL of the syndication, the XML returned by this call is unnecessarily verbose. The caller may reduce the

amount of information returned by specifying the “summary=1” in the resource URL that they GET or POST to. An example is:

<https://api.feedblitz.com/f.api/syndications?summary=1>

When the API sees a summary flag associated with a request for a syndication resource, the returned XML is reduced to the following:

```
<?xml version="1.0" encoding="utf-8" ?>
  <feedblitzapi version="1.0" xmlns:link="http://www.w3.org/1999/link">
    <rsp stat="ok">
      <success code="0" msg="Authorized" />
    </rsp>
    <syndications>
      <count>1</count>
      <syndication>
        <id>33369</id>
        <name>FeedBlitz Widgets</name>
        <description>FeedBlitz Widgets a test feed</description>
        <link>http://foo.typepad.com/foo/atom.xml</link>
        <status>ok</status>
        <liststatus>ok</liststatus>
        <turbo>1440</turbo>
        <subscriberssummary>
          <subscribers status="deleted">2</subscribers>
          <subscribers status="ignored">1</subscribers>
          <subscribers status="ok">3</subscribers>
        </subscriberssummary>
      </syndication>
    </syndications>
  </feedblitzapi>
```

This form is simpler and easy to convert to RSS, HTML and OPML. There is no analogous summary operation for subscription resources.

POST – Searching Syndications

Like some other resources, Syndications support searching via POST of an XML document to the resource URL:

```
<feedblitzapi version="1.0" xmlns:link="http://www.w3.org/1999/link">
  <searchpattern>foo</searchpattern>
  <searchstatus>fail</searchstatus>
  <searchlink>http://foo.com</searchlink>
  <sort order="desc">updated</sort>
</feedblitzapi>
```

Tag	Comment
<searchpattern>	Performs a literal (i.e. not keyword) search for the specified text in the syndication’s name, description or URL. It is not possible to restrict the search to just the URL, say.
<searchstatus>	Returns syndications with that given status. Valid feed status values are: <ul style="list-style-type: none"> • ok • fail • paused • deleted
<searchlink>	Searches for an exact match for the URL. Since URL’s are unique to any one

	user within FeedBlitz, it makes no sense to specify <searchlink> and <searchpattern>.
<sort>	Sort the results, if omitted the default sort of list id will be used. You may sort on any tag in the syndication record that does not have child tags, e.g. you may sort on the “created” field but not the “subscribersummary” tag. You may sort on these fields even if you have requested summary results only, when the field itself is not returned in the response XML.

Specifying multiple search parameters (e.g. both <searchpattern> and <searchstatus>) requires all conditions to be satisfied for a syndication to be returned. You may post a search to a syndication URL with the summary flag enabled; this will return the summary form of the syndication XML resource.

POST – Editing Syndications

You may POST syndication information to the relevant resource URL. Dates are stripped out, as are any <subscribersummary> values. Note that changing the information in a syndication changes it for all subscribers, and may also change the appearance of the feed in FeedBlitz sites, including FeedAdvisor.com.

Setting a syndication’s status to “deleted” will stop circulation of email updates and remove the circulation from subscriber’s dashboards. Marking it as “paused” stops circulation, but leaves the feed in subscriber’s dashboards (in the paused state).

A POST will return the appropriate XML for the selected resource.

DELETE – Deleting Syndications

Deletes the resource, returning the appropriate XML on completion.

PUT – Creating a New Syndication

PUT the appropriate XML data to the /syndications resource. The feed will become immediately available for you to manage via the API or the FeedBlitz.com GUI. The caller will also be added automatically as a new subscriber.

IMPORTANT: No autodiscovery or validation is performed on the URL passed to the API. If you wish to perform autodiscovery you should perform that yourself in advance, determine the URL to be added as the link, and then POST it to the API.

If successful, you will be returned the full XML for the new resource. The core element in the returned XML is the ID. The ID is the same value as used in the HTML “FEEDID” input parameter for FeedBlitz signup dialogs and links. You may therefore use this information to dynamically construct a signup dialog yourself without having to use the feedblitz.com GUI.

Captcha (Image Verification)

Resource path: `/captcha`

Returns a path to a verification image and a key to go with it. Users type the key embedded in the image, which is then sent back to the API server for verification. The captcha image may be used standalone, and is also embedded with the creation of a new user or a new subscription resource.

```
<?xml version="1.0" encoding="utf-8" ?>
  <feedblitzapi version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rsp stat="ok">
      <success code="0" msg="Authorized" />
    </rsp>
    <captcha>
      <random>123</random>
      <img>http://feedblitz.com/an_image_file</img>
      <response>what_the_user_typed</response>
    </captcha>
  </feedblitzapi >
```

When used with a GET the `<response>` tag is not returned. To validate a user's response, post the `<random>` and the `<response>` tags to the resource (the `` tag is not required). If it validates, the `<rsp>` tag from the POST will be OK. If not, a new `<captcha>` will be served back to you to be re-presented to the user.

You may use the `/captcha` resource standalone to validate a user is not a machine, as well as using it embedded in the `/user` resources as described below.

User (Account registration and Subscription Signup)

Resource path: `/user`

The `/user` resource has three distinct uses:

- 1) To retrieve information about the authenticated user
- 2) To register a new account at FeedBlitz
- 3) To start the new subscription signup process

GET – About the Current User

Returns information about the current logged on user, including their password. The API requires SSL, so the integrity of the password information is protected in transit.

```
<?xml version="1.0" encoding="utf-8" ?>
  <feedblitzapi version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink">
    <rsp stat="ok">
      <success code="0" msg="Authorized" />
    </rsp>
  </feedblitzapi >
```

```

</rsp>
<user>
  <i d>123</i d>
  <email>phil@feedblitz.com</email>
  <password>sekret</password>
  <mailformat>HTML</mailformat>
  <keepprivate>0</keepprivate>
  <status>ok</status>
  <created>2005-08-01 22:23:00 -0500</created>
  <ip>1.2.3.4</ip>
  <failcount>0</failcount>
  <bounces />
</user>
</feedblitzapi>

```

Tag	Comment
<id>	The user's unique identifier within FeedBlitz
<email>	The user's email address
<password>	The user's FeedBlitz password
<keepprivate>	1 if the user is set to be reported to publishers as anonymous, 0 otherwise
<status>	ok / pending / undeliverable / deleteme / deleted / unsubscribed
<created>	Date the ID was created
<ip>	IP from which the user was created. May be an octet or "Imported by:" and text describing the source of the address.
<mailformat>	The literal: HTML or TEXT
<failcount>	Number of bounces received. Used for soft bounce processing (hard bounces send the user to the "deleteme" state, from which they are set to "deleted" nightly).
<bounces>	Number of consecutive delivery failures as recorded by the FeedBlitz servers. A successful delivery resets this value to zero (i.e. an empty tag).

There is exactly one user if the caller authenticates: the caller themselves.

POST – Updating the Current User

The current user may change any of their settings (except their ID) by posting the XML to the /user resource.

Important: If you set the current user's state to "deleted" they will no longer receive updates, but their syndications will remain intact (so that their subscribers will continue to receive updates). This is by design: should the publisher of an email become undeliverable and then removed by the system (asset to "deleted") then it is unreasonable to terminate their subscribers as well.

A user who successfully logs in to either the GUI or the API will have their status restored.

To remove a user AND their syndications AND their subscribers, you need to DELETE the user using the DELETE method (see below).

PUT – Registering a New User Account

To register a new user and start the authentication process via the email activation link, PUT the XML document to the /user resource (API callers must authenticate with a different ID). A valid captcha entry MUST be part of the XML document sent to the resource, as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
  <feedblitzapi version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink">
    <user>
      <email>phil@feedblitz.com</email>
      <password>sekret</password>
      <keepprivate>0</keepprivate>
      <mailformat>TEXT</mailformat>
      <ip>1.2.3.4</ip>
      <captcha>
        <random>123</random>
        <response>what_the_user_typed</response>
      </captcha>
    </user>
  </feedblitzapi >
```

If the image verification fails (i.e. the <response> is incorrect) you will be returned a fresh captcha that must be used in the next attempt.

If the registration fails (because the user already has an account) then a warning or error status will be returned indicating the nature of the problem.

Regardless, the returned document will only contain the email you attempted to register; it is NOT the same as a GET for that user. This preserves privacy and security settings that the extant user may have in place. Also note that a PUT to an existing user is NOT the same as a POST – you may only use PUT to create a new user. Callers must POST to update an account's settings.

If the PUT is successful the user will be sent an activation email. You may neither use the API nor the FeedBlitz GUI using the new account's credentials until the user has activated their account using the link in the confirmation email. If the link must be resent the user must use the lost password facility at feedblitz.com

PUT – A New Subscription to an Existing Syndication

This operation mimics the effect of the user adding their email to your FeedBlitz signup form. The process is similar to creating a new account, except in this case there is a subscription entity in the user XML as well as the captcha.

You do not need to know the user's FeedBlitz ID; the email address will be used. The subscription's ID is required however, as in the following XML:

```
<?xml version="1.0" encoding="utf-8" ?>
  <feedblitzapi version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink">
    <user>
      <email>phil@feedblitz.com</email>
      <captcha>
        <random>123</random>
        <response>what_the_user_typed</response>
      </captcha>
    </user>
  </feedblitzapi >
```

```
</captcha>
<subscription>
  <i d>1234</i d>
  <referrer>http://foo.com/signup.html </referrer>
  <i p>1. 2. 3. 4</i p>
</subscription>
</user>
</feedblitzapi>
```

You may have multiple <subscription> blocks in the XML to sign a user up for multiple options (e.g. a daily and a weekly summary). Review the returned XML carefully for errors.

If the email address exists in FeedBlitz already, an email will be sent to the user inviting them to confirm their subscription. If not, an account will be created and then the activation email sent. Like the PUT above, the only returned user information will be the email address supplied by the caller, and a PUT is not the same as a POST – a PUT on an existing user will not update that user’s information (such as their password).

DELETE – Deleting a User

The delete method not only marks the user as deleted, it also suspends all the user’s syndications, subscriptions, and the subscriptions of any users subscribed to that user’s syndications.

Autoresponders

Resource URL: [/autoresponders](#)

FeedBlitz offers an autoresponder capability based on source RSS blogs / feeds. This resource returns all the autoresponders published by the user. If none are defined an empty <autoresponder /> tag is returned.

```
<?xml version="1.0" encoding="utf-8" ?>
<feedblitzapi version="1.0" xmlns:link="http://www.w3.org/1999/link">
  <rsp stat="ok">
    <success code="0" msg="Authorized" />
  </rsp>
  <autoresponders>
    <count>1</count>
    <autoresponder>
      <i d>123</i d>
      <publisherid>57</publisherid>
      <name>Thank You</name>
      <description>Send a free daily email course for 7
      days</description>
      <link>http://mysite.com/courseware/email_free.rss</link>
    </autoresponder>
  </autoresponders>
</feedblitzapi>
```

Tag	Comment
<i d>	The ID of the feed. Use with the /subscriptions resource to find more information on the feed.
<publisherid>	The ID of the publisher
<name>	The name of the autoresponder as defined by the user to FeedBlitz

<description>	The feed's description, as entered by the user to FeedBlitz.
<link>	The source URL that is polled to feed the autoresponder.

This resource only supports the GET method. To change an autoresponder you need to POST to the appropriate /syndication resource.

For information on setting up and using autoresponders in FeedBlitz, see <http://feedblitz.blogspot.com/2006/05/feedblitz-announces-rss-and-blog.html>

Time Zones

Resource path: **/timezones**

In FeedBlitz 2 it is possible to specify the time zone of a syndication. In the standard service, FeedBlitz polls each syndication daily, starting shortly after 1am local time (before version 2, polling started shortly after 1am US eastern time). FeedBlitz adjusts its poll time for daylight saving changes in both the selected time zone and the server's own time zone.

The /timezone resource gives information about all the potential time zones available to FeedBlitz. A syndication's time zone ID is a reference to the relevant entity from this call.

Tag	Comment
<id>	The time zone ID
<name>	The common name for the time zone
<display>	The display name for the zone. Includes the offset from GMT as text, and one or more city or country names as appropriate.
<offset>	The <i>current</i> offset from GMT in text, e.g. -05:00
<offsetminutes>	The <i>current</i> offset from GMT in minutes

This resource only supports the GET method.

Dynamic Ad and Metric Insertion

FeedBlitz 2 adds the ability for users to add site tracking and advertising code to their FeedBlitz-delivered subscriber landing pages. In the API these entities are /dynacontent (for the subscription's settings) and /dynaproviders – they are returned as part of the /syndication data.

Available Content Providers

Resource path: **/dynaproviders**

Lists the available providers and their IDs, used for correlating with the /dynacontent objects returned by the syndication resources. This is a read only resource (i.e. only GET is supported).

Tag	Comment
<id>	The provider ID

<name>	The common name for the provider
<type>	0 for an ad network. 1 for a visitor analysis service.

A Syndication's Dynamic Content Settings

Resource path: `/dynacontent`

This resource describes the syndication-specific settings (such as your unique ID) for the associated provider.

```
<?xml version="1.0" encoding="utf-8" ?>
<feedblitzapi version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink">
  <rsp stat="ok">
    <success code="0" msg="Authorized" />
  </rsp>
  <dynacontent>
    <count>1</count>
    <dynacontent>
      <id>167</id>
      <providerid>2</providerid>
      <listid>112441</listid>
      <type>0</type>
      <value>pub-4583890834284459</value>
    </dynacontent>
  </dynacontent>
</feedblitzapi>
```

Tag	Comment
<id>	The unique identifier in FeedBlitz for this dynacontent.
<providerid>	The provider this dynacontent uses; corresponds to the ids from the \dynaprovider resource.
<listid>	The syndication to which this dynacontent belongs
<value>	The provider-specific variable that determines the content presented to the user. Usage and value is defined by the provider.

Note that there is no status field; if defined the relevant /dynacontent resource is active. Also note that there is a one to one match between a /dynacontent resource and a /syndication resource. To specify the same /dynacontent values for multiple /syndications a new /dynacontent entity must be created using a PUT for the relevant <listid>

GET – Fetching Dynacontent Information

It is expected that, most of the time, the dynacontent will be retrieved as part of the relevant \syndication resource (in which case the <listid> is implicit in the parent \syndication). A GET method is provided here for completeness.

Path	Comment
/dynacontent	Returns all the user's dynacontent settings for all the syndications for that user.
/dynacontent/124	The dynacontent for ID # 124 (NOT syndication # 124)

The following verbs are also supported: PUT, POST and DELETE.

API Support

For support, please write to the standard FeedBlitz support address with your case. The standard knowledge base will be updated as necessary.