

# Spongecell Developer API Primer

---

The Spongecell Developer API provides a universal method of accessing Spongecell calendar and user data. The API uses REST to provide a simple and easily accessible method of accessing the data over HTTP.

The focus of this document is to provide pointers on how to quickly get started using the Spongecell Developer API.

Further details on the API are available at <http://spongecell.com/api/help>.

## Introduction to REST

We highly recommend first reading about REST (Representational State Transfer) in Wikipedia at [http://en.wikipedia.org/wiki/Representational\\_State\\_Transfer](http://en.wikipedia.org/wiki/Representational_State_Transfer). It provides an excellent introduction to the concepts of REST and will ease adoption of the Spongecell Developer API.

One of the key concepts of REST is that regardless of the particular *resource*, nearly all of the API will have identical syntax. In addition, the HTTP method is used to distinguish between different actions: a GET will return an index, while a POST will create a new *resource*. The latter is of particular importance during development and testing to make sure that the intended action is being called.

## Accessing the Spongecell Developer API

### Summary

Although there are a large set of possible API calls and different usable scenarios in which they could be used, this section will go through a simple scenario to illustrate several important calls and concepts. Please refer back to the online API reference at <http://spongecell.com/api/help> for further details.

Each of the following steps is explained in detail in the sections which follow:

1. Authenticate to your Spongecell account.
2. View all the Calendars for a Spongecell account.
3. View the Events for a Calendar.
4. Create and modify Events.

### Authenticate to your *Spongecell* account

*Initial authentication* is similar to “logging in” to an account in that it only needs to be performed once per *session*. The Developer API supports several methods of authentication, but we will use the

common method using a user name and password. Please read more about authentication at <http://spongecell.com/api/help/authentication>.

Depending on the API method called with the authentication parameters, either a *session* or a temporary *authorization token* is created. Either will allow full access to the authenticated account for a limited amount of time, after which *initial authentication* will need to be performed again.

Examples of the calls follow. In both cases, a POST is required for the CREATE action.

[http://spongecell.com/api/authentication\\_tokens.xml?user\\_name=tester&password=test](http://spongecell.com/api/authentication_tokens.xml?user_name=tester&password=test) [POST]

```
<?xml version="1.0" encoding="UTF-8"?>
<authorization_token>
  <calendar_id type="integer"></calendar_id>
  <created_at type="datetime">2007-10-12T19:23:14Z</created_at>
  <expires_at type="datetime"></expires_at>
  <level>FULL</level>
  <owner_id type="integer">40</owner_id>
  <remaining_uses type="integer"></remaining_uses>
  <user_id type="integer">40</user_id>
  <value>oJv8IeIJE/bGSK6qlxBabjJnpnc%2BruqMQaliuuNZT9ANXdLGYSNEw1Dm%2B8Iq
8P5xBW3QJUQu93XXxhHzt65Uftuv2es6c9ErfJ/IhLKaB8FCuxZPKzhz7v//
cQysHySbAoAYsUBZxUna/uWlOhtveHUP9WTYiK0zg2EVRkNtdG0VPP529YcH
7j/HkD191T5m
</value>
  <widget_id type="integer"></widget_id>
</authorization_token>
```

The above call will return the actual token value in the *value* element, along with the id of the user to which it grants authorization in the *user\_id* element.

[http://spongecell.com/api/session.xml?user\\_name=tester@password=test](http://spongecell.com/api/session.xml?user_name=tester@password=test) [POST]

```
<?xml version="1.0" encoding="UTF-8"?>
<session>
  <first_name>Tester</first_name>
  <id type="integer">170</id>
  <last_name>User</last_name>
  <user_name>tester</user_name>
  <class>UserSpace::Promoter</class>
  <primary_email_address>tester@test.com</primary_email_address>
</session>
```

Creating a *session* will allow you access to that user's account without having to provide a token.

## View all the Calendars for a Spongecell account

Using the above *authorization token* or after creating a *session*, one can then view the calendars belonging to the corresponding Spongecell account. The example below uses the token; simply omit it if you created a session instead.

<http://spongecell.com/api/calendars.xml?token=oJv8IeIJE/bGSK6qlxBabpSA0SyG374oEYARxcYsWrmNHAwUN5EyIRrJyq87\n/JZ7vRkdmnSlqWzPnWD3egU7IIwWyClizi9H2KVhnQfJi8>

[WPK5RVoForuxXZ\na8rugijjvdb3/LLkTmGezDd2QgddBxt/T8SnJGE2K1M4NEFraoSyjxk9h0vU\nTlCudUDX2Bbp\n](http://WPK5RVoForuxXZ\na8rugijjvdb3/LLkTmGezDd2QgddBxt/T8SnJGE2K1M4NEFraoSyjxk9h0vU\nTlCudUDX2Bbp\n)

```
<?xml version="1.0" encoding="UTF-8"?>
<calendars>
  <calendar>
    <description></description>
    <id type="integer">257</id>
    <name>New Calendar 5338</name>
    <creator_name>Test User</creator_name>
    <rss_url>http://spongecell.com/api/rss/events/tester/New+Calendar+5338</rss_
url>
    <ics_url>http://spongecell.com/api/ics/calendar/tester/New+Calendar+5338</ics
_url>
  </calendar>
  <calendar>
    <description></description>
    <id type="integer">258</id>
    <name>Upcoming Events</name>
    <creator_name>Test User</creator_name>
    <rss_url>http://spongecell.com/api/rss/events/tester/Upcoming+Events</rss_url
>
    <ics_url>http://spongecell.com/api/ics/calendar/tester/Upcoming+Events</ics_u
rl>
  </calendar>
</calendars>
```

This call will return ids for each calendar which you will need to use in other calls.

## View the Events for a Calendar

There are two different ways of viewing events for a specified calendar. One is by viewing the individual calendar (which also shows the events for that calendar); the other is by viewing just the events for a specified calendar. The former will also show information for the calendar. Both take the same optional parameters, which allow for specifying a start time, end time, or maximum results.

The following links are “real” and can be used to see the full results. Further details can be found at <http://spongecell.com/api/help?resource=calendars> and <http://spongecell.com/api/help?resource=calendars>.

If you have previously authenticated, then all events will be shown. If not authenticated, then only public events will be shown.

[http://spongecell.com/api/calendars/62739?start\\_time=200705111700&max\\_results=20](http://spongecell.com/api/calendars/62739?start_time=200705111700&max_results=20)

```
<?xml version="1.0" encoding="UTF-8"?>
<calendar>
  <description></description>
  <id type="integer">62739</id>
  <name>Upcoming Events</name>
  <creator_name>Sponge Cell</creator_name>
  <rss_url>http://spongecell.com/api/rss/events/sample_user/Upcoming+Events</rs
s_url>
```

```

<ics_url>http://spongecell.com/api/ics/calendar/sample_user/Upcoming+Events</
ics_url>
  <events>
    <event>
      <created_on type="datetime">2007-05-22T19:57:47-07:00</created_on>
      <description>Our CTO will be one year older. Let's celebrate on the
Russian River!</description>
      <end_time type="datetime">2008-08-10T20:00:00-07:00</end_time>
      <id type="integer">392798</id>
      <image_url>https://spongecell.com/image/view/8404</image_url>
      <location>Guerneville, CA</location>
      <location_name>Somewhere on the Russian River</location_name>
      <start_time type="datetime">2008-08-10T20:00:00-07:00</start_time>
      <ticket_url></ticket_url>
      <title>CTO Birthday</title>
      <updated_on type="datetime">2007-09-16T20:48:59-07:00</updated_on>
      <promote type="boolean">>true</promote>
      <time_zone>Samoa</time_zone>
      <tag_list></tag_list>
      <attending></attending>
      <venue>
        <city></city>
        <country></country>
        <state></state>
        <street>Guerneville, CA</street>
        <zip></zip>
      </venue>
      <resources>
        <resource>
          <id>8404</id>
          <purpose>main</purpose>
          <url>https://spongecell.com/image/view/8404</url>
        </resource>
      </resources>
    <event_page_url>https://spongecell.com/event/view/392798</event_page_url>
  </event>
  <event>
    ...
  </event>
</events>
</calendar>

```

[http://spongecell.com/api/events.xml?calendar\\_id=62739](http://spongecell.com/api/events.xml?calendar_id=62739)

```

<?xml version="1.0" encoding="UTF-8"?>
<events>
  <event>
    ...
  </event>
  <event>
    ...
  </event>
</events>

```

## Create and modify Events

Once authenticated, new events can be created, existing events can be modified, and existing events can be deleted. Further details can be found at <https://spongecell.com/api/help?resource=events>.

<http://spongecell.com/api/events.xml?title=Sample+Event> [POST]

```
<?xml version="1.0" encoding="UTF-8"?>
<event>
  <created_on type="datetime">2007-10-12T12:30:07-07:00</created_on>
  <description></description>
  <end_time type="datetime">2007-10-12T12:30:07-07:00</end_time>
  <id type="integer">38</id>
  <image_url></image_url>
  <location></location>
  <location_name></location_name>
  <start_time type="datetime">2007-10-12T12:30:07-07:00</start_time>
  <ticket_url></ticket_url>
  <title>Sample Event</title>
  <updated_on type="datetime">2007-10-12T12:30:07-07:00</updated_on>
  <promote type="boolean">>false</promote>
  <time_zone>Los Angeles</time_zone>
  <tag_list></tag_list>
  <attending></attending>
  <resources>
  </resources>
</event>
```

A successful create call will return the newly created event, including the id, which can be used for subsequent operations on the event. Both creating and updating events take the same parameters.

<http://spongecell.com/api/events/38.xml?title=Different+Event> [PUT]

```
<?xml version="1.0" encoding="UTF-8"?>
<event>
  <created_on type="datetime">2007-10-12T12:30:07-07:00</created_on>
  <description></description>
  <end_time type="datetime">2007-10-12T12:30:07-07:00</end_time>
  <id type="integer">38</id>
  <image_url></image_url>
  <location></location>
  <location_name></location_name>
  <start_time type="datetime">2007-10-12T12:30:07-07:00</start_time>
  <ticket_url></ticket_url>
  <title>Different Event</title>
  <updated_on type="datetime">2007-10-12T12:30:07-07:00</updated_on>
  <promote type="boolean">>false</promote>
  <time_zone>Los Angeles</time_zone>
  <tag_list></tag_list>
  <attending></attending>
  <resources>
  </resources>
</event>
```

A successful update call will also return the updated event. A delete is simply performed by using the same address as above (without parameters) but using the DELETE method.

<http://spongecell.com/api/events/38.xml> [DELETE]

## Accessing the API through Flash

Spongecell provides an example set of ActionScript 3 code which the developer can use as a base for building Flash applications which access the API. Although it is not necessary to use this code, it does provide a simple way of accessing all the current API methods from within Flash, and thus an excellent starting point for the Flash developer.

## RestClient

The main entry point for these helpers is the `com.spongecell.api.RestClient` class. The `RestClient` handles all the communication with Spongecell, parses the returned XML (or error message) and then fires a specific event with the necessary information.

Each API controller (or **resource**) has a corresponding set of methods in `RestClient`, with each controller action corresponding to a single method. Each method also has a corresponding constant string in `RestClient` which can be used to listen for the completion of that event.

## Events

Each **resource** has a corresponding event (or two) which includes parsed data from the API response. These events are in the `com.spongecell.events` package.

Each of these events is a subclass of the `APIResponseEvent`, which includes a field indicating whether the call was a success (or a failure), an error message (in case of failure), and the raw returned XML.

```
public class APIResponseEvent extends flash.events.Event
{
    public var success:Boolean;
    public var error:String;
    public var xml:XML;

    public function APIResponseEvent(type:String, e:String=null,x:XML=null)

...
}
```

## Models

There are also ActionScript classes encapsulating the more complex return types from the API. These are found in the `com.spongecell.model` package.

## Examples

### Get events from a calendar

The `getCalendar` method takes a calendar ID and optional arguments for start-date and max-results. Details on the parameters are available at <http://spongecell.com/api/help?resource=calendars>.

```
public static const EVENT_GET_CALENDAR = "getCalendar";

public function getCalendar(calendarId:String, startDate:String=null,
maxResults:String=null):void
{
    ...
}
```

When the method call has completed (either a success or a failure), a `APICalendarResponseEvent` is dispatched, and can be listened for using the `EVENT_GET_CALENDAR` constant. This event includes a parameter indicating whether the calendar is read-only (or writeable due to the current user having authenticated to be able to write to this calendar) as well as an Array of the returned events.

```
public class APICalendarResponseEvent extends APIResponseEvent
{
    public var events:Array;
    public var readOnly:Boolean;

    public function APICalendarResponseEvent(type:String, ev:Array,
r:Boolean, e:String=null)
    {
        ..
    }
}
```

Each of the returned events (in the events Array parameter) are actually `EventModel` objects which have been parsed from the returned XML.

```
public class EventModel extends Object
{
    public var id:Number;
    public var title:String;
    public var description:String;
    public var promote:Boolean;

    public var venue:VenueModel;

    public var startTime>Date;
    public var endTime>Date;
    public var timeZone:String;

    ...
}
```

## Advanced Usage

Although not necessary, there are other classes available for customizing or extending the current wrapper functionality.

### APIRequest

The `com.spongecell.api.APIRequest` class wraps a standard `URLRequest` with logic to handle calling out to the Spongecell API. Includes static constructor methods to create `APIRequests` for the various REST protocols (get, put, post, delete).

### XMLLoader

The `com.spongecell.net.XMLLoader` class wraps a standard `URLLoader` with logic to easily handle the common error conditions. The `RestClient` class uses this as the primary method of communicating with the Spongecell API. At completion of the load, a `com.spongecell.net.XMLLoaderEvent` is dispatched and carries with it the necessary information to handle the success or failure of the call.

## Accessing the API through Javascript

As the API is a RESTful API, it can easily be accessed from within Javascript even though there are not yet any Javascript wrapper code available to aid in the integration. The API also currently only returns XML, but there are plans for the code to return JSON as well to facilitate integration with Javascript.

## Accessing the API through Other Means

As the API is a RESTful API, it can easily be accessed using any other means, such as Java. Although we have no plans to develop wrapper code for other languages, the standard HTTP request and response methods make it a simple integration process.