

Spongecell Community Builder Primer

Spongecell provides capabilities for *Community Builders* to access important parts of the Spongecell functionality and easily offer it to their users through their own web-site.

The focus of this document is to provide pointers on how to quickly get started with the *Community Builder* functionality within the Spongecell Developer API. Please also reference the Spongecell Developer API Primer document.

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

Background

The *Community Builder* functionality is an optional set of functionality that can be enabled for Spongecell Promote accounts. This functionality allows a Promoter to create and manage a community of users which have access to a subset of the standard Spongecell functionality, but offer this functionality to their users through their own web-site.

A Spongecell Promote account with this functionality enabled will be referred to as a *Community Builder* account and is essentially a superset of a Promote account with the ability to perform extra functions relating to *communities*. Because it is a subset, when referring to a *Community Builder* account, it also refers to functionality that is available to Promoters or even standard Spongecell Calendar accounts.

Each *Community* will have a number of *Community User* accounts associated with it which are similar to a standard Spongecell Calendar user, except partitioned from any other *community users*, or any standard Spongecell Calendar users. This holds true even if the e-mail addresses are the same. The e-mail address of any community user **must** be unique **within** each community. Spongecell Calendar or Promote accounts can be considered as a different community, so the same e-mail address can be used for a Spongecell Promote (or Calendar) account, community X, and community Y. This also holds true for the user name.

Integration of *Community* web-site and *Spongecell* through the API

Summary

Most of the API integration involves the standard set of calls in the Spongecell Developer API. However, authentication needs to be handled differently in the *community* space, as it is necessary to be able to easily “act as” different *community users* and the initial authentication to the *community builder* account needs to be more secure than a standard log-on process.

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

1. Authenticate to your *Community Builder* account and receive a **temporary authorization token** for that account which will allow access to *Community Builder* functionality.
2. Provide a user of your community access to a Spongecell *community user* account.
 - a. Create the *Community User* account if it has not yet been created.
 - b. Authenticate to the user's *Community User* account and receive a **temporary authorization token** for that account which will allow access to *Community User* functionality.
3. Display a Spongecell *Widget* to your user, already authorized for the corresponding *community user* account.
4. Allow for the *community user* to interact with their account.

Authenticate to your *Community Builder* account

Initial authentication is similar to "logging in" to an account in that it only needs to be performed once *per session*. However, as just using a user-name and password to authenticate is not secure enough, the API provides an authentication mechanism using a SHA1 hash of a shared secret and timestamp. This *digest* is passed in the request along with the id of the *community* and a timestamp. 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. We recommend utilizing the *authorization token* in production.

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

http://spongecell.com/api/authentication_tokens.xml?community_id=40×tamp=20071012192313Z&digest=fe75076c375fb472086fbc9d374c4b5c20f1afda

```
<?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%2BruqMQaliuNZT9ANXdLGYSNEw1Dm%2B8Iq
8P5xBW3QJUQu93XXxhHzt65UFtuv2es6c9Er fJ/ IhLKaB8FCuxZPKzhz7v//
cQysHySbAoAYSUBZxUna/uWlhtveHUp9WTYiK0zg2EVRkNtdG0VPP529YcH
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. The token **must** be used in subsequent calls.

http://spongecell.com/api/session.xml?community_id=40×tamp=20071012192313Z&digest=fe75076c375fb472086fbc9d374c4b5c20f1afda

```
<?xml version="1.0" encoding="UTF-8"?>
<session>
  <first_name>Community</first_name>
  <id type="integer">170</id>
  <last_name>Builder</last_name>
  <user_name>community_builder</user_name>
  <class>UserSpace::Promoter</class>
  <primary_email_address>builder@communities.com</primary_email_address>
</session>
```

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

Create a *Community User* account

Using the above *authorization token* (for the *Community Builder* account), one can then create *Community User* accounts through the create user functionality in the API. The only required field is the email address of the *Community User*, the *authorization token*, and your community id; although the *first_name*, *last_name*, and *user_name* parameters can also be specified.

http://spongecell.com/api/users.xml?community_id=40&email_address=user@communities.com&token=oJv8IeIJE/bGSK6qlxBabpSA0SyG374oEYARxcYsWrmNHAwuN5EyIRrJyq87\n/JZ7vRkdmmSlqWzPnWD3egU7IIwWyClizi9H2KVhnQfJi8WPK5RVoForuxXZ\na8ruqii jvdb3/LkTmGezDd2QgddBxt/T8SnJGE2K1M4NEFraoSyjxk9h0vU\nTlCudUDX2Bbp\n

```
<?xml version="1.0" encoding="UTF-8"?>
<user>
  <first_name></first_name>
  <id type="integer">521</id>
  <last_name></last_name>
  <user_name>com_user_40_20071012193006Z</user_name>
  <class>UserSpace::CommunityUser</class>
  <calendar_id>203</calendar_id>
  <email_address>user@communities.com</email_address>
  <widget_id>21</widget_id>
</user>
```

This call will create a new *Community User* account and return several ids which should be stored and associated with that user account (on your own system). The *id* element shows the id of that created user, the *calendar_id* element shows the id of the calendar automatically created for the user, and the *widget_id* element shows the id of the (social network) widget automatically created for the user.

Please see <http://spongecell.com/api/help?resource=users> for further details on user creation.

Authenticate to a *Community User* account

Part of the additional functionality provided for *Community Builder* accounts is that authorization to the *Community Builder* account also provides full authorization to the *Community User* accounts belonging to that *Community*.

Thus, authentication is similar to the above but simpler in this situation as only the *Community User* account needs to be specified, although using a session is not really practical. The account is specified using the user id returned from the user creation, along with the token for the *Community Builder* account.

```
http://spongecell.com/api/authentication_tokens.xml?user_id=521&token=oJv8IeIJE/bGSK6qlxBabpSA0SyG374oEYARxcYsWrmNHAWuN5EyIRrJyq87\n/JZ7vRkdmnSlqWzPnWD3egU7IIwWyClizi9H2KVhnQfJi8WPK5RVoForuxXZ\na8ruqii jvdb3/LLkTmGezDd2QgddBxt/T8SnJGE2K1M4NEFraoSyjxk9h0vU\nTlCudUDX2Bbp\n (POST)
```

```
<?xml version="1.0" encoding="UTF-8"?>
<authorization_token>
  <calendar_id type="integer"></calendar_id>
  <created_at type="datetime">2007-10-12T19:30:06Z</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">521</user_id>
  <value>oJv8IeIJE/bGSK6qlxBabpSA0SyG374oEYARxcYsWrmNHAWuN5EyIRrJyq87
/JZ7vRkdmnSlqWzPnWD3egU7IM99mZoQItcuIQ98LmF%2BD%2BIyfv1ktVmOqXGd
GYa8VpewzoUtbqIIwSihObmBWNlhJ4jfoQszu7euJMYtxO%2BF50IIBMhI6uMn
LgGlEblTabRp
</value>
  <widget_id type="integer"></widget_id>
</authorization_token>
```

The token returned with this call will allow access to the *Community User* account.

Account Interaction

Interaction with the *Community Builder* and *Community User* accounts (and calendars) can all be performed using the corresponding *authorization token*. Simply pass **token=XXXXX** as an additional parameter on any call and it will result in the call being performed on that particular account. This allows the *Community Builder* to in effect “act as” either the *Builder* or the *User* depending on the situation simply by passing the appropriate token.

As each token is temporary and the *Community User* token is distinct from the *Community Builder* token, it is completely safe to embed the *Community User* token in web pages shown to that user.

The below section describes how this can be used with the existing Spongecell Flash-based widgets, but this token can be utilized with any custom code to allow for transparent access to the *Community User* account without requiring the user to log on manually.

Display a Spongecell Widget

The existing Flash-based widgets (Myspace Event List and Calendar List View) support using the *authorization token* to perform automatic “logging in” of the user through the widget. This allows the user to interact with their calendar through the widget and perform actions such as viewing their calendar, adding events, setting reminders, etc.

Using Spongecell Promote, you can create and customize these widgets for your *Community Builder* account, as well as view the embeddable HTML for each widget. As the widget settings are stored within Spongecell, only the id of the widget and its dimensions are distinct relative to different widgets.

However, in order for the widget to perform the automatic “logging in” of a specific user, the *authorization token* also needs to be passed to the widget at display time. This can be done by specifying the token in the flash source url.

```
<embed  
src='http://spongecell.com/flash/download/1/event_list.swf?token=oJv8IeIJE/bG  
SK6qlxBabpSA0SyG374oEYARxcYsWrmNHawuN5EyIRrJyq87\n/JZ7vRkdmnSlqWzPnWD3egU7IM9  
9mZoQItcuIQ98LmF%2BD%2BIyfv1ktVmOqXGd\nGYa8VpewzoUtbqIIwSihObmBWNlhJ4jfoQSzu7  
euJMYtxO%2BF50IIBMhI6uMn\nLgG1EblTabRp&widgetId=16&apiURL=http://Spongecell.c  
om/api/&imgHost=http://s3.amazonaws.com/spongecell/crossdomain.xml'  
allowScriptAccess='always' allowNetworking='all' type='application/x-  
shockwave-flash' width='500' height='450' />
```

Once embedded in the above fashion, the widget will behave as if the user had logged in manually, thus allowing them to create new events, edit and delete existing events, send reminders and invitations.