FlashChat Module API

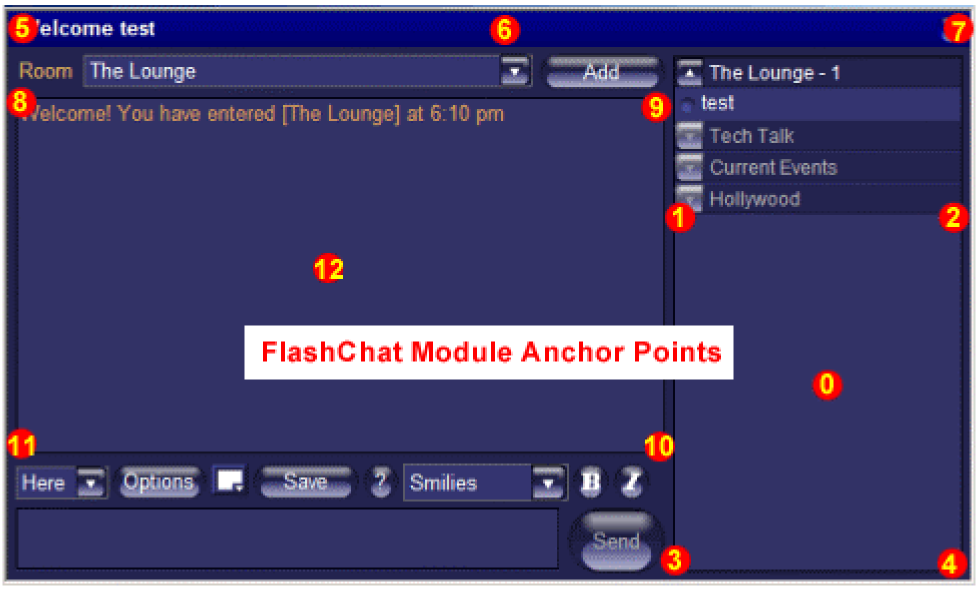
Beginning with FlashChat 4.0, externally loaded Flash files may communicate bi-directionally with the main chat. FlashChat can send event notifications to the module, and the module may in turn activate FlashChat's internal methods. This allows third-party developers to create powerful, dynamic add-ons to FlashChat without modifying FlashChat's internal code.

In FlashChat's config.php file (located in the "inc" folder), you will find the following array, although the actual values may vary depending on when you downloaded and installed FlashChat:

```
'module' => array(
      'achor'    => 0,
      'path'     => 'moduleText.swf',
      'stretch'  => false,
),
```

"moduleText.swf" is a sample module which has been included with FlashChat to help you get started. It may be the most useless and annoying module ever developed, but it does demonstrate all of the methods shown below.

The "stretch" option allows you to stretch the loaded SWF to the full anchor space available. To see this, try changing "path" to "banner.swf", which is included with the FlashChat distribution. There are 13 anchor points in FlashChat (0 - 12). These are locations in the FlashChat interface where modules can be positioned. Anchor points are only relevant for visible modules.



Communication from a module to FlashChat

change room

The module can forcibly move the current user to another room. This is useful for creating an "admin module" which can kick the user out of the room.

Usage
callChatFunc("changeRoom", args);

Sample

```
var args = new Object();
args['room'] = 'Hollywood';
callChatFunc("changeRoom", args);
```

logout

The module can forcibly logout the current user. For example, the module might be used as a timer, which could "close" the chat at a specified time. Also useful for development of an "admin module", which could logout the current user.

Usage
```
callChatFunc("logout", args);
```

Sample
```
var args = new Object();
callChatFunc("logout", args);
```

send popup message

The module can send user popup messages to FlashChat. For example, in a timer/logout module, the module may need to alert the user of the event, like "The chat will close in 5 minutes!"

Usage
```
callChatFunc("alert", args);
```

Sample
```
var args = new Object();
args['text'] = "This is alert test from module";
callChatFunc("alert", args);
```

send IRC command

The module can send any IRC command to FlashChat. For example /backtime, /me, etc. (see the FlashChat Wiki for a complete list of IRC commands).

Usage
```
callChatFunc("irc", args);
```

Sample
```
var args = new Object();
args['irc']  = "/me TEST module call to IRC";
args['args'] = "";
callChatFunc("irc", args);
```

Comunication from FlashChat to module

Note: all functions shown below must be redefined in the module root level.

change room

Any time that a user switches to a new room, the module is alerted of this event. Thus, the module always "knows" which room the user is in. This is useful for creation of an "intelligent banner ad" module, which can show a specific banner ad based on the room.

Usage
mOnRoomChanged(args);

Sample
```
function mOnRoomChanged(args)
{
    trace( "Room changed to " + args['room'] );
}
```

login

This is useful, for example, to create a "welcome" module - i.e., a module which sends a popup alert like "Welcome to the chat!" when the user logs in.

Usage
mOnUserLogin(args);

Sample
```
function mOnUserLogin(args)
{
    trace( "User login " + args['username'] );
}
```

change language

When the user changes his/her language, the module is alerted of this event... for example, in an "intelligent banner ad" module, it might be desirable to have a different banner ad based on the user's language.

Usage
mOnChangeLang(args);

Sample
```
function mOnChangeLang(args)
{
    trace("Change lang " + args['langname'] );
}
```

send IRC command

When an IRC command is sent by the user, the module is notified of this event. However, *parsing* the IRC command is the job of the module. Thus, if a module developer wishes to parse an IRC command so that the module can take some action, then the module developer will need to parse the IRC command line.

Usage
mOnIRC(args);

Sample
```
function mOnIRC(args)
{
    trace( "User send IRC command " + args['irc'] );
}
```

send message

Function called every time a user sends a simple text message.
Note: this function must return a changed message, or the original message.

Usage

```
mOnSendMsg(args);
```

Sample

```
function mOnSendMsg(args)
{
        _txt.text += "\nUser '" + args['username'] + "' send message '" + args['msg'] + "' from
        room '" + args['room'] + "'";

        return args['msg'] + “ * added from module *”;
}
```