SmartFoxServer Lite - quick reference

In this document you will find a synthetic explanation of the properties, methods and events of the SmartFoxClient Flash API.

It is very important that you have already read the introductory articles in the gotoAndPlay() **Multiplayer Central** (http://www.gotoandplay.it/_articles/multiplayerCentral/) before you go on with this document.

Also in the same site section you will find the step-by-step tutorials for the examples bundled with the SmartFoxServer Lite package.

This document is still under development and we'll update it with the next server versions.

Properties

The following properties are available in the instance of the SmartFoxClient object.
For example:

```
server = new  SmartFoxClient()
...
...
// code here for loggin in a room
..
..
trace("You are logged in room: " + server.activeRoomId)
```

activeRoomId
The id of the room where you are currently logged in

myUserId
The user id assigned by the server to you when you logged in

myUserName
The user name you're currently using

playerId
The playerId assigned by the player to you when you enter a game room.

Methods

NOTE: with most of the methods you will find listed here, the last argument (roomId) is optional.

You don't have to pass this parameter unless you're sending the message/command from a room that is different to the one you're currently logged in. This can happen only if your application allows a user to be present in more than one room at a time. (for example in a chat room and a game room, simultaneously)

The SmartFoxClient **activeRoomId** property is always set to the last joined room, so you can always know the id of the room you're currently in.

If your application/game needs to have users present in more than one room at a time then you should keep a list of the rooms in which your client is logged in and always specify the roomId param when calling a method that requires it.

autoJoin()

Automatically join the user in the default room, if it exist.
You can specify a default room in the "room definition" section of the config.xml file provided with SmartFoxServer Lite.

When a room is marked as autoJoin (adding the autoJoin="true" attribute in the <Room></Room> tag) it becomes the default room where all clients are initially joined when they login successfully.

Example:

```
serverInstance.autoJoin()
```

Event fired: **onJoinRoom / onJoinRoomError**

connect(ip, port)

Attempts to connect to the server.

ip = a string with the ip address
port = a port number (default = 9339)

The request will fire an onConnection event

Example:

```
serverInstance.connect("127.0.0.1", 9339)
```

Event fired: **onConnection**

createRoom(roomObj)

Creates a new room in the current zone

roomObj = a room Object

Example:

```
room = new Object()
room.name = "My Brand New Room"
room.maxUsers = 10


serverInstance.createRoom(room)
```

Event fired: **onRoomAdded / onCreateRoomError**

disconnect()

Closes the connection between client and server.

This will in turn fire a onConnectionLost event

Example:

```
serverInstance.disconnect()
```

Event fired: **onConnectionLost**

getActiveRoom()

Returns the unique id of the room you're currently in.

Example:
var **currRoom** = serverInstance.getActiveRoom()

getRoom(roomId)

Returns a room object from a room id. The id can be either the room unique id or its name.

roomId = the id of the room (numeric id, or room name)

Example:

```
var room = serverInstance.getRoom("The Hall")
```

If a room called "The Hall" exist in the current zone then the method will return its object

getRoomList()

Retrieves from the server the current list of rooms in the zone.

Example:

```
serverInstance.getRoomList()
```

Event fired: **onRoomListUpdate**

login(zone, nick)

Logs a user in a specific zone.

The standard SmartFoxServer Lite login method accepts guest users. Duplicate nicknames are not allowed. If a user logs in with an empty nickname the server automatically creates a name for the client using this format: "guest_n" where n is a progressive number.

If you need to implement your own login procedure, for example to check nicknames against a database, you can add it to your code BEFORE the SmartFox login code. This way, once the client is validated, you can just use the stadard login procedure.

zone = the name of the zone (a string)
nick = the user's nickname

Example:

```
serverInstance.login("testZone", "Jim")
```

Event fired: **onLogin**

joinRoom(newRoom, pword, dontLeave, oldRoom)

This command join the user in a new room. It also allow to choose what to do if the user was already present in another room.

newRoom = Id of the new room
pword = password for the room (if needed)
dontLeave = (OPTIONAL) boolean flag. If true the user will not leave the room he/she was previously in
oldRoom = (OPTIONAL) Id of the room to leave.

Example 1:

```
serverInstance.joinRoom(10)
```

The user joins the room with id=10 and he will leave the previous room

Example 2:

```
serverInstance.joinRoom(12, "mypassword")
```

The user joins the room with id=12 and he will leave the previous room.
The room is private so we provide a password.

Example 3:

```
serverInstance.joinRoom(15, "", true)
```

The user joins the room with id=15 and requests not to leave the current room he was in.

Event fired: **onJoinRoom / onJoinRoomError**

roundTripBench()

Starts measuring the time it takes for a very short message to go from the client to the server

and back to the client.
The result is given by the related onRoundTripResponse event.

Example:

```
serverInstance.roundTripBench()
```

Event fired: **onRoundTripResponse**

sendPublicMessage(msg, roomId)

Sends a public chat message.

msg = a message string
roomId = (OPTIONAL) the room id of the current room

Example:

```
serverInstance.sendPublicMessage("Hello guys!!")
```

Event fired: **onPublicMessage**

sendPrivateMessage(msg, userId, roomId)

Sends a private chat message.

msg = a message string
userId = the id of the private message recipient
roomId = (OPTIONAL) the room id of the current room

Example:

```
serverInstance.sendPrivateMessage("Hi John, how are you")
```

Event fired: **onPrivateMessage**

sendObject(obj, roomId)

Sends an actionscript object to the other users in the room.

This is usefull for sending complex data to clients like a game move.

Variable types allowed are: Number, String, Boolean, Object, Array, null

obj = the object to send
roomId = (OPTIONAL) the room id of the current room

Example:

```
move = {}
move.x = 150
move.y = 250
move.speed = 8

serverInstance.sendObject(move)
```

Event fired: **onObjectReceived**

setRoomVariables(varObj, roomId)

Set/Change a variable in the current room.

varObj = an object containing the variables to set/update
roomId = (OPTIONAL) the room id of the current room

Example:

```
obj = {}
obj.gameName = "Tetris"
obj.bestScore = 250000

serverInstance.setRoomVariables(obj)
```

Event fired: **onRoomVariablesUpdate**

setUserVariables(varObj, roomId)

Set/Change the value of one or more user variables.

varObj = an object containing the variables to set/update
roomId = (OPTIONAL) the room id of the current room

Example:

```
obj = {}
obj.name = "Paul"
obj.age = 20

serverInstance.setUserVariables(obj)
```

Event fired: **onUserVariablesUpdate**

Events

Here follows a list of the events fired by the SmartFoxServer Lite during runtime.

onCreateRoomError(errorMsg)

An error occurred while creating a new Room.

errorMsg = a string with the server message

onConnection()

Event is fired when the connection between client and server is established.

onConnectionLost()

The event fires when the connection between client and server was lost.

onJoinRoom(roomObj)

Event is fired when a room is joined successfully.

roomObj = the Room object representing the room just joined.

onJoinRoomError(error)

When a room join fails, this event is fired.

error **=** a string with the error message from the server

onLogin(resObj)

This is the message sent by the server in response to a client login request.

The resObj.success property is TRUE if the login was successfull or FALSE if an error occurred.

If **resObj.success** == true, you will receive:

resObj.name = the nickname for your client*

else if **resObj.success** == false, you will receive

resObj.error = a string describing the server error

***NOTE:** Why does the server sends the nickname back?

Actually the nickname sent by the clients to the server may be processed if it contains more than 50 characters (maximum allowed) or if it contains non allowed characters. (Permitted characters are all numeric and alphanumeric chars plus some ascii symbols like _ () [] {} etc..)

onObjectReceived(asObj, user)

Event is fired upon reception of an Actionscript object.

asObj = the object
user = User object of the sender

onPublicMessage(message, user)

Event is fired upon reception of a public chat message.

message = message string
user = User object of the sender

onPrivateMessage(message, user)

Event is fired upon reception of a private chat message.

message = message string
user = User object of the sender

onRoomAdded(newRoom)

A new room was added/created in the current Zone.
newRoom = the Room object representing the new room

onRoomDeleted(roomDeleted)

A room was deleted in the current Zone.

roomDeleted = the deleted Room object

onRoundTripResponse(trip)

This event is fired when the server responds to a roundTripBench() request.

The roundtrip time represents the amount of milliseconds that it takes a message to go from the client to the server and get back to the client.

trip = time expressed in ms.

You can calculate the average ping time from client to server by dividing trip by two.

Example:

```
pingTime = int(trip / 2)
```

onRoomListUpdate(roomList)

Event is fired when the list of rooms is updated, i.e. when you login in a new zone.

roomList is a list of Room objects.

onRoomVariablesUpdate(room)

Event is fired when one of the Room Variables is modified.

room = the Room object where the variables changed.

You can inspect Room variables by either reading the **variables** property or by calling the calling getVariable(varname) method on the Room object.

onUserCountChange(room)

The userCount property represents the number of users present in each room of the current zone. When the number of users in one room changes this event is fired.
You will receive these events from all the other rooms in the same zone as the room you're currently in.

room = the Room object where the user count changed.

To obtain the new number of users you can call the **room.getUserCount()** method

onUserEnterRoom(fromRoom, user)

A new user has entered the room you're currently in.

fromRoom = the current roomId
user = the User object representing the new client

onUserLeaveRoom(fromRoom, usrId, usrName)

A user left the current room.

fromRoom = current room Id
usrId = the id of the user
usrName = the name of the user*

*__note__: you get the id and name of the user instead of its User object because the API has already removed it from its userList

onUserVariablesUpdate(user)

A user in the current room has updated his/her user variables.

user = User who changed the vars

You can inspect User variables by either reading the **variables** property or by calling the calling getVariable(varname) method on the User object.