Scripting Component functions

A list of most of the scripting component methods that are usable from scripts. The use of all of these things depends greatly will depend on knowing/understanding how things fit together as well as a solid understanding of a lot of underlying computer techniques.

public void startRepeater(int initialDelay, int delayAmount, int eventNumber)

Starts a repeating call to a script.
initialDelay – time until first execution
delayAmount – time until successive executions
eventNumber – the scripting event number to execute

public void stopRepeater()

Stops the above repeater.

public void simpleServerStart(int code, int errorCode, int mode, int port)

Starts a simple web service
code – the ICE code included with messages
errorCode – the ICE code included with error messages
mode – the mode for the web service 0 = pass data, 1 = active script, 77 – act directly
port – listen port on this computer

public void simpleServerStop()

Stops the simple web service

public void playSound(String clipFile, int responseCode, int localOrGlobal)

Plays a sound clip
clipFile – the file to play (webdav currently)
responseCode – the ICE code to include with the play finish message
localOrGlobal – indicates whether the file path includes the cell name

public void getMyAvatar()

Get the reference to the avatar for this client. Preparatory to moving avatar.

public void moveAvatarForward()
public void stopAvatarForward()
public void moveAvatarBack()
public void stopAvatarBack()
public void moveAvatarRight()
public void stopAvatarRight()
public void moveAvatarLeft()
public void stopAvatarLeft()

Move the client's avatar.

public void executeAction(parms various)

Execute a command inside a module with scripting command execution enabled.

public void setAnimationStartTranslate(float X, float Y, float Z)
public void setAnimationStartRotation(float X, float Y, float Z)
public void setAnimationTimeMultiplier(int thyme)
public void setAnimationStartKeyframe(int start)
public void setAnimationEndKeyframe(int end)
public void setAnimationIceCode(int code)
public void setAnimationSaveTransform(int save)
public void setAnimationPlayReverse(int reverse)

Commands used to set animation characteristics.

public void contentReadResource(String theScript)

Read a file from the ressources area of the module.
theScript – file name of the ressource file to read

public int contentReadFile(String thePath, int repository)

Read a webdav file.
thePath – the webdav based path to the file
repository – the area of webdav space to read – 0 = user, 1 = root, 2 = system.

public int contentWriteFile(String theDir, String theFile, String theData, int repository)

Write a file to webdav area.
theDir – path to the area to write
theFile – the file to write
theData – the data to write
repository – area of webdav to write

public int contentCreateDir(String theDir, int repository)

Create a webdav directory.
theDir – directory path to create
repository – area of webdav

public void sendChat(String msg, String from, String to)

Send a chat message.
msg – the text to send
from – name used as the message sender
to – name used as the destination

public void getChat()

Enable chat handling.

public void yat()

Function to query for avatars that are present and where they are.

public void unrollYatsForIncoming()

Take the results of the yat() call and send them one at a time to an established incoming socket. Pretty much a special purpose thingy.

public void getYat()

Enable script execution on presence changes.

public void establishSocket(int code, int errorCode, String ip, int port)

Establish an outgoing socket connection.
*c*ode – the ICE code sent with the incoming messages
errorCode – the ICE code sent with error messages
ip – ip of system to connect to
port – port of system

public void establishIncomingSocket(int code, int errorCode, int port)

Establish a socket that listens for incoming socket connections
code – incoming message ICE code
errorCode – error message ICE code
port – listen port

public void sendMessage(String buffer)

Send a message on an outgoing socket connections
buffer – the message

public void sendIncomingMessage(String buffer)

Send a message on an incoming socket connection
buffer – the message

public void watchMessage(float code)

Tell the scripting component to pass to the ICE script messages with this code.
code – an allowable code

public void dontWatchMessage(float code)

Stop watching for an ICE code.

public void clearWatchMessages()

Turn off any active watch messages.

public void establishProximity(float outer, float middle, float inner)

Turn on proximity event processing and create a three sphere proximity.
outer – outer sphere radius
middle – middle sphere radius
inner – inner sphere radius

public void postMessageEvent(String payload, int Code)

Send an ICE message.
payload – the message to send
Code – the ICE message code

public void postMessageEventToServer(String payload, float Code)

Send an ICE message to server to be relayed to all other clients.
payload – the message
Code – the ICE code

public void setStateString(String value, int which)
public String getStateString(int which)
public void setStateInt(int value, int which)
public int getStateInt(int which)
public void setStateFloat(float value, int which)
public float getStateFloat(int which)
public void setStateBoolean(boolean value, int which)
public boolean getStateBoolean(int which)

Handlers for dealing with the different types of state variables. Primarily for scripting languages that cannot set/get these variables directly.
value – the value to place into the state variable
which – the state variable array element

public void clearScriptMap()

Clear the map that holds the compiled scripts effectively requiring a recompile.

public void getInitialRotation()

Retrieve the current rotation.

public void getInitialPosition()

Retrieve the current position.

public void setTranslation(float x, float y, float z, int notify)

Set the translation of the cell(object).
- x – x translation
- y – y translation
- z – y translation
- notify – propagate this across server. 0 = propagate, 1 = no

public void setRotation(float x, float y, float z, float w, int notify)

Set the rotation of the cell.
- x – x rotation
- y – y rotation
- z – z rotation
- w – rotation axis
- notify – propagate across server

public void setScale(float x, float y, float z, int notify)

Set the object scale

public void moveObject(float x, float y, float z, int notify)
public void rotateObject(float x, float y, float z, float w, int notify)
public void scaleObject(float x, float y, float z, int notify)

Relative transform changes

public void mySleep(int milliseconds)

Pretty obvious

public void createCellInstance(String className, float x, float y, float z, String cellName)

Create a cell instance.
- className – the full class name
- x, y, z – location of cell
- cellName – the name to give this cell instance