Script System Documentation

Last Updated: Wednesday, April 22, 2009

Table of Contents

1. Standards
2. Script Variables
3. Database and Database Object Implementation
4. Running a script
5. In closing

Standards

The first thing of importance when using the script system are standards that should be followed to allow for safe and problem free between the game and scripts, as well as maintaining a consistency that will make script author's lives easier. I have been using these standards thus far, and would appreciate it anyone writing API would use them as well.

API Standards

* Method names should be camel style, as is standard to Java. IE: getPlayerName();.
* All API methods should be named the type of object + ScriptAPI.

EX:  
 ItemScriptAPI

ArmorScriptAPI

* Absolutely under no circumstances are any methods to receive or give scripts access to internal methods. Instead, all methods should be interface methods.

IE:

Instead of creating a method such as the following:

public void addWaypoint(Player targetPlayer, Waypoint w) {

//add waypoint

targetPlayer.addWaypoint(w, true);  
 }

Create the following:

public void addWaypoint(long targetPlayerID, int planetID, int x, int y, int z) {

//Get the player in question.

Player targetPlayer = server.getPlayer(targetPlayerID);

//Ensure the player is valid.

if(targetPlayer != null) {

//Create the waypoint. And yes, I know this isn't a valid //constructor, it's just for demonstration's sake.

Waypoint w = new Waypoint(planetID, x, y, z);

//Add the waypoint.

targetPlayer.addWaypoint(w);

} else {  
 //throw an error exception  
 }

}

Yes, I know it's more code, but this way the scripts won't have direct access to the Player object, meaning there is absolutely no way there can be any malicious scripts. Plus, we can keep the contents of the Player object a secret, while all Script API methods will need to be JavaDoc'd. This'll come in handy until open source.

Notice the argument for the player ID. This will be explained in the next section of this document.

* All API should extend ScriptAPI. This is explained further down in this document.

Object Responsibility Standards

Each api object handle/implements methods form a specific object. Note that because every api object extends ScriptAPI, they receive the ability to manipulate the same objects as ScriptAPI. If you implement a new method from a target class (IE: Player) please implement it in the proper API class (IE: ScriptAPI).

ScriptAPI: SOEObject, Player

ItemScriptAPI: TangibleItem

Script Variables

As mentioned above, the only "player" scripts are given are an ID. On script runtime, a few variables are inserted depending on the type of object. Note that because all Script API objects extend ScriptAPI and inherit all ScriptAPI methods, the variables for player and object type are inserted the same in all cases.

System Scripts (will handle more script types):

api (variable to access ScriptAPI object).

targetPlayerID (variable for ID of the player interacting with the object, I do intend to implement the ability to get a list of all IDs around the object).

Item scripts:

api (variable to access ScriptAPI object).

targetPlayerID (variable for ID of the player interacting with the object, I do intend to implement the ability to get a list of all IDs around the object).

itemTemplateID (variable for the ID of the in game item that is running this script).

Database and Database Object Implementation

I'm not going to go into how the ScriptManager parses the scripts, just add a column into whatever object you're adding for script file names (IE: ScriptFileName.js), and I'll update the parse methods. It's worth nothing that whatever the container object is you're using to store the database data for the specified type of object (IE: ItemTemplate) needs to have methods to set and get the script file name.

Running a Script

Running a script is simple, simply get the ScriptManager from the ZoneServer, and use one of the script methods:

runSystemScript() (For system functions, like Tutorials, logging in, etc)

runScript() (For Items, and generic SOEObjects)

runScriptGenericMode()

runScriptGenericMode() requires an explination:

It allows you to run a script for an object or item it was not necessarily designed to run for. This method was designed for the in game command @runScript (@runscript [Script File Name] [Script Type on Run] [Target Object ID on Run] <Target Player(First Name) on Run>). You can also use this method to run a script for the type of object it WAS designed for.

In other words, a script developed for item template ID #102 can be run for world object #203. Note that using a script for an object it was not designed for WILL cause problems (IE: This example).

In Closing

Yes, right now the script system is a bit complicated and almost too robust, but I intend to fix that when my summer break starts (April 30th, 2009). If anyone adding API needs assistance, or something changed, let me know. Were I to explain the manager this document would be 2, 3 pages longer.