Access Control List

NRP includes a complete Access Control List (ACL) that allows you to secure and limit access to the server, its functions, and resources in any number of ways.

The key concept of NRP's ACL (and ACLs in general) is that you're giving a specific object a set of rights. In NRP's case objects is one of two things - resources or users. There are numerous *rights* available in NRP - these mainly focus on server-side scripting functions.

What this essentially means is that the ACL allows you to choose exactly what functions a resource or user can perform. This can obviously be invaluable - for example preventing all your server's players from being able to ban each other, or preventing your new untested resources from doing the same.

Of course, *with great power comes great responsibility* and it is very easily possible to completely break resources - for example, disabling spawnPlayer for all resources would be a Bad Thing. Of course, there are situations when you might want to do this - if you want to force all your resources to use a spawn manager resource for example, but even this is somewhat draconian.

Contents

Understanding the ACL

There are two major components to the ACL: groups and ACL lists. They appear as **<group name="">** nodes and **<acl name="" />** nodes. Their purpose is to:
1. Grant users permission to control the server and use resource commands. Examples:

- Allowing only admins to use the giveweapon function of freeroam
- Allowing all users to start a resource

2. Grant resources permission to use script functions and functions of other resources. Examples:

- Allowing a resource to use the restartResource function
- Allowing a resource to use the call function to use exported functions from another script

Default Groups

NRP has provided some default groups with increasing permissions. These groups are:

- **Everyone**
- **Moderator**
- **SuperModerator**
- **Admin**
- **Console** - This controls permissions of people who are using the console through **<object name="user.Console" />**
- **RPC** - Remote Procedure Call. Specifically grants access to callRemote only and disables commands of default resources. Check the function for details.

To explain further, I will use the Everyone group as an example. By default it looks like this:

```
<group name="Everyone">
    <acl name="Default" />
    <object name="user.*" />
    <object name="resource.*" />
</group>
```

You will first notice the acl name inside the group. It defines what permissions the group has. Users and resources in this group will have the permissions specified on the "Default" acl name list. *Note: You will notice this group is special, in that it includes every user and resource by using a **wildcard (*)** where the user or resource name would be.*

Now, scroll further down the ACL and you will see the **<acl name="Default" />** listing. Note I have trimmed this list

dramatically due to its length.

```
<acl name="Default">
    <right name="command.start" access="false" />
    <right name="command.stop" access="false" />
    <right name="command.stopall" access="false" />
    ...etc etc...
    <right name="function.executeCommandHandler" access="false" />
    <right name="function.setPlayerMuted" access="false" />
    <right name="function.restartResource" access="false" />
    ...etc etc...
    <right name="general.adminpanel" access="false" />
    <right name="general.tab_players" access="false" />
    <right name="general.tab_resources" access="false" />
    ...etc etc...
    <right name="command.freeze" access="false" />
    <right name="command.shout" access="false" />
    <right name="command.spectate" access="false" />
    ...etc etc...
</acl>
```

- **Function** entries are NRP scripting functions. For example, if a resource needed to use restartResource and was only in the 'Everyone' group (with the 'Default' list), it would be denied access to restartResource and fail to work correctly.
- **Commands** are created when a resource uses addCommandHandler. An example would be typing **/createvehicle [vehicle]** in the chatbox for the freeroam resource. This controls whether users in the group using this ACL can use the command. *Note: commands have no effect on resources within the group. Commands are only related to users.*
    - *General is a custom right name group created by the admin resource but it works on the same principles. The script works with them by using hasObjectPermissionTo*

You will notice some groups such as admin have multiple **<acl name="" />** nodes. An example is the admin group:

```
<group name="Admin">
    <acl name="Moderator" />
    <acl name="SuperModerator" />
    <acl name="Admin" />
    <acl name="RPC" />
    <object name="resource.admin" />
    <object name="resource.webadmin" />
    <object name="user.Random" />
</group>
```

This gives all the permissions defined in each **<acl name="" />** node in order of listing. So for example, the admin group makes sure all the permissions are given to admins by using all the lists. If there are any conflicts, the lowest entry wins. For example, pretend these 2 acls were in a group in the following order:

1. <acl name="Default"> sets <right name="general.ModifyOtherObjects" access="false" />
2. <acl name="Admin"> sets <right name="general.ModifyOtherObjects" access="true" />
3. For all users and resources in group admin: <right name="general.ModifyOtherObjects" access="true" />

Resource Groups and ACLs

You will notice there are some other groups that came with NRP. These were defined by resources that came with NRP. If a resource wants to designate specific ACL rights not provided by the default NRP groups, it can create its own ACL name and a group to use it. I will show AMX's entry as an example. AMX is designed to emulate SA-MP scripts and it needs a certain set of permissions that don't fit the default groups well. It is shown below:

```
<group name="AMX">
    <acl name="AMX" />
    <object name="resource.amx" />
</group>

<acl name="AMX">
    <right name="general.ModifyOtherObjects" access="true" />
    <right name="function.startResource" access="true" />
    <right name="function.stopResource" access="true" />
    <right name="general.adminpanel" access="false" />
    ...etc etc...
    <right name="command.kick" access="false" />
    <right name="command.freeze" access="false" />
    <right name="command.mute" access="false" />
    ...etc etc...
</acl>
```

Grant resources access to other resources

To allow resource access to any other resource, you have to allow it explicitly in your ACL file. The rule `general.ModifyOtherObjects` grants full access to any resource on the server. See the example in the last section of what it should look like.

If you don't want to grant a resource complete access to every resource on the server, then you can append the modifiable resource's name preceded by a dot (.) to the `general.ModifyOtherObjects` rule:

```
<acl name="Custom">
    <right name="general.ModifyOtherObjects.resourceName" access="true" />
    <right name="general.ModifyOtherObjects.admin" access="true" />
    <right name="general.ModifyOtherObjects.runcode" access="true" />
    <right name="general.ModifyOtherObjects.mapmanager" access="true" />
</acl>
```

The ACL definitions above will grant any resource with the ACL **Custom** access to the resources *resourceName*, *admin*, *runcode* and *mapmanager*.

Modifying the ACL

There are three ways you can modify the ACL - how you do it depends on who you are.

HTTP Interface

You can use the webadmin HTTP interface to modify the ACL in your web browser. This is by far the easiest way to do so. Just make sure the *webadmin* resource is started on your server and visit *http://ServerIP:HttpPort/*. You can then use the two sections - ACLs and Groups. ACLs allow you to create your Access Control Lists - lists of rights. Groups allow you to group together collections of users and assign ACLs to them. For example, the ACLs section allows you to specify that the Admin ACL has access to the *start* console command. You can go to the Groups section and create an Admin *group* that has access to your Admin *acl*. You can then add users to your Admin *group*.

XML file

You can modify the ACL.xml file manually. This has a fairly straightforward syntax, but it can get a bit confusing at times. If you do it while the server is running, don't forget to call the scripting function aclReload ("start runcode", "run aclReload()") so the new ACL is loaded, or otherwise stop your server before modifying. This also prevents your changes from being accidentally overwritten by the server.

Scripting functions

You can use a large number of ACL scripting functions to modify the ACL on the fly. Of course, you can (and really should!) limit access to the ACL functions with the ACL. Note that the **admin resource** that comes with NRP can be used to manage the ACL from the resources tab. You must be set up as admin to use the admin panel. Click here for admin setup instructions.