By Bas v0.1.2

Oxide Rust Experimental Plugins Manual

Part 1 The Server

On how to set up and administer vanilla Rust Experimental servers
and to install Oxide, extensions and plugins

Installing steamcmd

As seen on https://developer.valvesoftware.com/wiki/SteamCMD

Create a folder somewhere with any name, lets call it yourSteamCmdLocation

Create a folder somewhere with any name, we call it yourRustServersLocation

Download steamcmd from http://media.steampowered.com/installer/steamcmd.zip

Extract steamcmd into yourSteamCmdLocation

Install / update Rust Experimental Server manually

Startbutton, cmd (to open a terminal)

```
c:\Users\Yourname>cd c:\yourSteamCmdFolder
c:\yourSteamCmdFolder>steamcmd.exe
Steam>login anonymous
Steam>force_install_dir c:\yourRustServersLocation
Steam>app_update 258550 -beta experimental
Steam>quit
```

Install / update Rust Experimental Server using a shortcut

You can automate the above by creating a shortcut somewhere and set the target property to

```
c:\yourSteamCmdLocation\steamcmd.exe +login anonymous +force_install_dir
c:\yourRustServersLocation +app_update "258550 -beta experimental" +quit
```

For example, mine is

```
c:\steamCmd\steamcmd.exe +login anonymous +force_install_dir
c:\rustServers +app_update "258550 -beta experimental" +quit
```

Load Vanilla Rust Experimental Servers using shortcuts

Create shortcut somewhere, set its target property to:

```
c:\yourRustServersLocation\RustDedicated.exe -batchmode -load
+server.hostname "My Server Name" +server.port 28015 +server.identity
"My_Server_Name" +server.seed 6738 +server.maxplayers 8 -server.worldsize
2000 -autoupdate
```

Set the shortcuts name to something useful, perhaps RustDedicatedPORTNR, that would make the name for the above RustDedicated28015

For example: my shortcuts

Named: *RustDedicated28047* Target:

```
C:\rustServers\RustDedicated.exe -batchmode -load +server.hostname "Bas
Rust 1" +server.port 28047 +server.identity "1" +server.seed 863029248
+server.maxplayers 8 -server.worldsize 4000 -autoupdate
```

Named: *RustDedicated28049* Target:

```
C:\rustServers\RustDedicated.exe -batchmode -load +server.hostname "Bas
Rust 2" +server.port 28049 +server.identity "2" +server.seed 4294967000
+server.maxplayers 8 -server.worldsize 4000 -autoupdate
```

Firewall may complain about the server: we know it is good, so allow.

When first starting the server: ignore the ***Couldn't load blahblah.sav file doesn't exist*** warning: its about loading the save file, which does not exist on first startup.

Server Options

hostname is name shown to users

load loads the game saved

port is the port used for this server. Open port on router to let people from outside your router in. port number +1 will also be used bij rcon by default.

identity is a foldername where all data about this particular server is kept, inside the yourRustServersLocation\server folder

seed is a random number that determines the way a map looks, should be between 1 and 4294967295. Due to a bug, high-number seeds close together will 'be the same'. (should be fixed soon)

maxplayers is the maximum number of players allowed in at the same time

worldsize is the length and with of the worldmap in 'meters' 2000 is minimum, 8000 maximum. memory used: at 2000: 350.000 kb+, at 4000: 926.000 kb+, at 8000: 2.270.000kb+ (4.400.000kb+)

autoupdate lets the server update automatically (dunno really what this does)

See http://rustdev.facepunchstudios.com/dedicated-server fore more commands like save interval, ticktime and secure.

Connect to your game

press **F1**,console system, to see the ingame console

If you are already connected to a server:

```
client.disconnect
```

To connect to one of your servers on the same machine or behind the same router:

```
client.connect YourLanIp:ServerPortNr
```

To get YourLanIp in windows 7

```
ipconfig
```

Searcht the output for:

```
Ipv4-adres.............: 192.168.2.200
```

Mine to get into Bas 1 with port set to 28047 is:

```
client.connect 192.168.2.200:28047
```

To connect from outside your router, with your routerport open pointing to your server machine:

```
client.connect YourWanIp:ServerPortNr
```

To get YourWanIp https://www.google.nl/#q=whats+my+ip

F1 to **close console** and return to game.

Get your steamID

Once connected with a client, at the rust server:

To see a list of users: type

```
users
```

You will see a list of steamIDs, followed by the user names. Note your steamID.

A 16 digit number like 98005361191765611

Console commands at the server

To get a list of users that are connected

```
users
```

To change the time, can be anything from 0 to and including 24

```
env.time 8
```

To have an airdrop

```
event.run
```

To quit a server

```
quit
```

To say something to the users inside from the server console: type

```
say hello all!
```

To close server with 1 minute warning (every 5 seconds)

```
restart
```

To see all commands available (some will perhaps not work as suspected, it is work in progress)

```
find .
```

Make your steamID admin *or* moderator

At the server, you should make yourself owner *or* moderator (name and reason are optional)

ownerid (id) ("name") ("reason")

Normal players have auth level 0

Set auth level to 1, moderator:

```
moderatorid 98005361191765611 "Bas" "BossMan"
```

Reconnect with the client to the server, the server output for Bas is:

```
Bas has auth level 1
```

Or, set auth level to 2, owner:

```
ownerid 98005361191765611 "Bas" "BossMan"
```

Reconnect with the client to the server, the server output for Bas is:

```
Bas has auth level 2
```

Server Config Files

To write out server config files to C:\yourRustServersLocation\server\yourServersIdentity\cfg\:

```
server.writecfg
```

Now you can browse to C:\yourRustServersLocation\server\yourServersIdentity\cfg\ and inspect the configuration files.

'**users**.cfg' holds the authlevel (“ownerid” or “moderatorid”), the steamID, and the displayName of all admins/moderators.

```
ownerid 98005361191765611 "Bas" "BossMan"
```

'**bans**.cfg' holds by default two bans, each with the word “banid”, then the steamID, the username, then the reason:

```
banid 76561197965853940 "Spike Spiegel" "Trolling, spamming chat"
banid 76561197978026540 "Iron Pi" "Trolling, spamming chat"
```

Deleting a Server

Stop running the server by typing:

```
quit
```

Browse to C:\yourRustServersLocation\server\ and delete the folder with the name of the identity of the server that you want to delete

Remove the shortcut to start the server, or keep it, perhaps tinker with it, and run it again to create a fresh server in C:\yourRustServersLocation\server\

Install Oxide

The server untill now was 'vanilla', untouched, original fresh out of the package from steam itself.

To install Oxide, first close all vanilla servers:

```
quit
```

Go to http://oxidemod.org/

Use steam to create a login

Go to downloads, Oxide 2 for Rust Experimental, Download Now

Extract the zip file to yourRustServersLocation

Overwrite all, copy and replace all

Start server(s) with shortcut(s) like normal

You will see messages like

```
[Oxide] 10:11 PM [Info] Loading Oxide core
[Oxide] 10:11 PM [Info] Loading extensions...
[Oxide] 10:11 PM [Info] Loading plugins...
```

Install Rust:IO extension

Optional. Some things you cant do with a plugin. Like display a map. But you can do them in an extension. It is basically a dll file with extra functionality. You only have to copy the dll to one location, all servers will use the dll. Each server will have its own config file.

Close all servers:

```
quit
```

Go to http://oxidemod..org/

Go to extensions, Rust:IO, Download Now

Copy Oxide.Ext.RustIO.dll into C:\yourRustServersLocation\RustDedicated_Data\Managed\

Start server(s) with shortcut(s) like normal

```
[Oxide] 10:25 PM [Info] Rust:IO> Starting HTTP server on *:28047 ...
[Oxide] 10:25 PM [Info] Rust:IO> Installed.
```

You can now see a map when you set your browser to your serverip:portnr

Config files of RustIO

Browse in your explorer to C:\yourRustServersLocation\server\yourServerIdentity\oxide\config\

Open the RustIO config file:

```
{
  "broadcastChat": true,
  "broadcastDeaths": true,
  "broadcastSpawns": true,
  "defaultLanguage": "en",
  "displayBuildings": false,
  "displayMonuments": true,
  "displayMortality": true,
  "enableLocationSharing": true,
  "helpMessage": "To locate yourself on the LIVEMAP, browse to
<color=#CC3A2D>http://playrust.io</color> and type in this server's name!",
  "hideSpecificMonuments": [
    "cave",
    "mountain"
  ],
  "welcomeMessage": "To locate yourself on the LIVEMAP, browse to
<color=#CC3A2D>http://playrust.io</color> and type in this server's name!"
}
```

And change:

```
"welcomeMessage": "To locate yourself on the LIVEMAP, browse to
<color=#CC3A2D>http://playrust.io</color> and type in this server's name!"
```

perhaps to:

```
"welcomeMessage": ""
```

to not bother the players with rust:io feedback when they log in.

Type

```
oxide.reload RustIO
```

in the server console to load the new settings, and log in to the server with a client to check the message is gone.

Install plugins

Go to http://oxidemod..org/

Go to plugins, Oxide 2 for Rust Experimental, top-plugins to check the most popular plugins available.

For example: Give. Download now.

Its a .cs file, but it can also be a .py file, a .js file or a .lua file. It can even be a zip file.

Zip files are different. You need to check where to unzip them exactly into.

The single cs, py, js and lua files go into *each* server, at the folder: C:\yourRustServersLocation\server\yourServerIdentity\oxide\plugins\

As soon as you copy it into the server\yourServerIdentity\oxide\plugins\ map, Oxide will try to load it

```
[Oxide] 10:39 PM [Info] Loaded plugin Give v2.1.2 by Reneb
[Oxide] 10:39 PM [Info] Give: Creating a new config file
```

Plugin config files

Browse in your explorer to C:\yourRustServersLocation\server\yourServerIdentity\oxide\config\

Open the Give config file:

```
{
  "authLevel": {
    "give": 1,
    "giveall": 2,
    "givekit": 1
  },
  "Give": {
    "logAdmins": true,
    "overrightStackable": false
  },
  "Messages": {
    "itemNotFound": "This item doesn't exist: ",
    "multiplePlayersFound": "Multiple Players Found",
    "noAccess": "You are not allowed to use this command",
    "noPlayersFound": "No Players Found"
  }
}
```

You can set some flags under “Give”, tinker with the “Messages”, or set the “authLevel” of each function of Give (give, giveall and givekit). Give is the most important one.

If you only want the owner to be able to use that function: set to 2.

If you want both owners and moderators to be able to use that certain function, set to 1.

Reload the plugin with the new config file at the server console:

```
oxide.reload Give
```

Now any player that is logged in to the server with a high enough authLevel will be able to use *give* functions in the in game chat panel (use enter to open it)

```
/giveme wood 1000
```

or

```
/give SomeName wood 1000
```

Remove plugins

When you don't want a plugin to run in some server anymore:

Browse in your explorer to C:\yourRustServersLocation\server\yourServerIdentity\oxide\plugins\ and remove the plugin. Oxide will unload it at once. The plugin can also leave a config file in C:\yourRustServersLocation\server\yourServerIdentity\oxide\config\ perhaps you want to delete that as well.

Part 2 Writing Plugins

On how to write plugins for Rust Experimental - Oxide

in all four languages supported: C#, lua, javascript and python

Demonstrates simple tasks with example plugins

Choose programming language

There are four programming language file-types supported in the C:\yourRustServersLocation\server\yourServerIdentity\oxide\plugins folder: .cs, .lua, .js, and .py.

There are some 159 plugins available for Rust Experimental

89 Lua (1 is in a zip-file with a config file)

61 C#

6 java-script (1 is in a zip-file with some more folders)

3 python

Differences between languages

C# is closely knit with Oxide, as Oxide itself is written in C#. Lua, js and python all use the same library functions exposed by Oxide. This means there is sometimes a difference between C# on the one hand, and Lua, Py and Js on the other. Some examples are:

-C# uses System.Console.WriteLine to write plainly to the console, while all languages use their 'print' commands, which only writes to the console: through Oxide. Oxide will place *[oxide] the time [info]* before everything you print.

-All languages but C# use rust.BroadcastChat and rust.SendChatMessage , but C# uses PrintToChat and player.ChatMessage to do the same.

-Only C# has to have its class set to inherit from RustPlugin. All the others dont.

-You can give commands with the serverconsole in all languages but Python by just typing the command. I am not sure why, perhaps I am doing something wrong, but in python, you apparently have to type classname.command

Choose editor

Perhaps you already have a editor you like, that's fine.

For those looking for good free editors:

Visual Studio Community 2013: https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx Has the best auto-completion. Top notch c# support. Javascript and python are so-so, lua is not supported at all. When creating a new project, don't forget to set '**references**' to all dll's needed in C:\yourRustServersLocation\RustDedicated_Data\Managed

Or perhaps try Komodo Edit: http://komodoide.com/download/#edit for some lightweight quick editing of config files and such. For when you don't want to create a whole visual studio project to just edit a single file. Support for python and javascript is ok, lua and c# are barely supported.

HelloWorld, minimal plugin, print to console

See http://oxidemod.org/ docs for more information about plugins

Go to folder C:\yourRustServersLocation\server\yourServerIdentity\oxide\plugins

You don't need to have the Cs, Py, Js and Lua part at the begin of the filename. It is only to make them have different names, so they can run all together at once without creating name-issues.

Take note that the class name, if any, **should** be *exactly* the same as the filename without extension (the .cs, .py, .js or .lua part).

So the class-name for the c# example will be CsHelloWorld

This plugin shows how a **minimal plugin** looks like, and how to **print to the console**

LuaHelloWorld.lua

```
PLUGIN.Title = "LuaHelloWorld"
PLUGIN.Version = V(0, 1, 0)
PLUGIN.Author = "Bas"
function PLUGIN:Init()
    print("Hello World from Lua")
end
```

```
[Oxide] 1:20 AM [Info] Loaded plugin LuaHelloWorld v0.1.0 by Bas
[Oxide] 1:20 AM [Info] Hello World from Lua
```

CsHelloWorld.cs

```
namespace Oxide.Plugins
{
    [Info("HelloWorldCs", "Bas", "0.1.0")]
    class CsHelloWorld : RustPlugin
    {
        void Init()
        {
            System.Console.WriteLine("Hello World from Cs");
        }
    }
}
```

```
[Oxide] 12:29 AM [Info] Loaded plugin CsHelloWorld v0.1.0 by Bas
Hello World from Cs
```

JsHelloWorld.js

```
var JsHelloWorld = {
    Title : "JsHelloWorld",
    Version : V(0, 1, 0),
    Author : "Bas",
    Init: function() {
        print('Hello World from Js');
    }
}
```

```
[Oxide] 1:14 AM [Info] Loaded plugin JsHelloWorld v0.1.0 by Bas
[Oxide] 1:14 AM [Info] Hello World from Js
```

PyHelloWorld.py

```
class PyHelloWorld:
    def __init__(self):
        self.Title = "PyHelloWorld"
        self.Version = V(0, 1, 0)
        self.Author = "Bas"
    def Init(self):
        print 'Hello World from Py'
```

```
[Oxide] 11:59 PM [Info] Loaded plugin PyHelloWorld v0.1.0 by Bas
[Oxide] 11:59 PM [Info] Hello World from Py
```

ChatBack, chatcommand with arguments, chat to player

A plugin that has a chat command, that will generate a 'hello back' message.

It demonstrates to **create a chat command** and link it to one of your own functions, and how to extract data like displayName and userID of the player giving the chat command, and any or none **arguments**.

And it shows you how to let the plugin **chat to a player**

LuaChatBack.lua

```
PLUGIN.Title = "LuaChatBack"
PLUGIN.Author = "Bas"
PLUGIN.Version = V(0, 1, 0)
function PLUGIN:Init()
    command.AddChatCommand("hellolua", self.Object, "chat_hello")
end
function PLUGIN:chat_hello(player, cmd, args)
    rust.SendChatMessage(player, "Hello back from Lua", null, "0")
    --just some additional information you can use when handling the chat
command
    print('player.displayName='..player.displayName)
    rust.UserIDFromPlayer(player)
    print('player.userID='..rust.UserIDFromPlayer(player))--lua does not know
uint player.userID, it turns up as a float
    print('cmd='..cmd)
    msg='Arguments:'
    if args.Length == 0 then
        msg=msg..' None'
    else
        for i=0, args.Length-1, 1 do
          msg=msg.." "..args[i]
        end
    end
    print (msg)
end
```

Type in chat:

```
/hellolua test 1 23 456
```

Get typed back:

```
Hello back from Lua
```

Server Console gives additional information:

```
[Oxide] 3:18 AM [Info] player.displayName=Bas
[Oxide] 3:18 AM [Info] player.userID=98005361191765611
[Oxide] 3:18 AM [Info] cmd=hellolua
[Oxide] 3:18 AM [Info] Arguments: test 1 23 456
```

CsChatBack.cs

```
namespace Oxide.Plugins
{
    [Info("CsChatBack", "Bas", "0.1.0")]
    class CsChatBack : RustPlugin
    {
        [ChatCommand("hellocs")]
        void chat_hello(BasePlayer player, string cmd, string[] args)
        {
            player.ChatMessage("Hello back from Cs");
            //just some additional information you can use when handling the
chat command
            System.Console.WriteLine("player.displayName="+player.displayName);
            System.Console.WriteLine("player.userID="+player.userID);
            System.Console.WriteLine("cmd="+cmd);
            if (args.Length==0){
                System.Console.WriteLine("Arguments: None");
            }else{
                string msg="Arguments:";
                foreach (string arg in args){
                    msg=msg+" "+arg;
                }
                System.Console.WriteLine(msg);
            }
            System.Console.WriteLine("");//flush console to prevent last line
not visible
        }
    }
}
```

Type in chat:

```
/hellocs 1 2 3
```

Get typed back:

```
Hello back from Cs
```

Server Console gives additional information:

```
[Oxide] 3:08 AM [Info] player.displayName=Bas
[Oxide] 3:08 AM [Info] player.userID=98005361191765611
[Oxide] 3:08 AM [Info] cmd=hellocs
[Oxide] 3:08 AM [Info] Arguments: 1 2 3
```

JsChatBack.js

```
var JsChatBack = {
    Title : "JsChatBack",
    Author : "Bas",
    Version : V(0, 1, 0),
    Init: function() {
        command.AddChatCommand("hellojs", this.Plugin, "chat_hello");
    },
    chat_hello: function(player, cmd, args){
        rust.SendChatMessage(player, "Hello back from Js", null, "0");
        /*just some additional information you can use when handling the chat
command*/
        print ("player.displayName="+player.displayName);
        print ("player.userID="+player.userID);
        print ("cmd="+cmd);
        if (args.length==0) {
            print ("Arguments: None");
        }else{
            msg="Arguments:";
            for (var i = 0; i < args.length; i++) {
                msg=msg+" "+args[i];
            }
            print (msg);
        }
    }
}
```

Type in chat:

```
/hellojs a b c
```

Get typed back:

```
Hello back from Js
```

Server Console gives additional information:

```
[Oxide] 3:00 AM [Info] player.displayName=Bas
[Oxide] 3:00 AM [Info] player.userID=98005361191765611
[Oxide] 3:00 AM [Info] cmd=hellojs
[Oxide] 3:00 AM [Info] Arguments: a b c
```

PyChatBack.py

```
class PyChatBack:
    def __init__(self):
        self.Title = 'PyChatBack'
        self.Author = 'Bas'
        self.Version = V(0, 0, 1)
    def Init(self):
        command.AddChatCommand('hellopy', self.Plugin, 'chat_hello')
    def chat_hello(self, player, cmd, args):
        rust.SendChatMessage(player, "Hello back from Py", None, "0")
        #just some additional information you can use when handling the chat
command
        print 'player.displayName='+player.displayName
        print 'player.userID='+str(player.userID)
        print 'cmd='+cmd
        if args:
            print('Arguments: '+' '.join(args))
        else:
            print('Arguments: None')
```

Type in chat:

```
/hellopy test
```

Get typed back:

```
Hello back from Py
```

Server Console gives additional information:

```
[Oxide] 2:34 AM [Info] player.displayName=Bas
[Oxide] 2:34 AM [Info] player.userID=98005361191765611
[Oxide] 2:34 AM [Info] cmd=hellopy
[Oxide] 2:34 AM [Info] Arguments: test
```

AdminBroadcast, check authLevel, chat to all at once

Shows how to **broadcast a chat to all players** at once

Also demonstrates how to **check the authLevel** of the player that is using the chatcommand

AuthLevels are: 2 for owner, 1 for moderator, 0 for normal users

LuaAdminBroadcast.lua

```
PLUGIN.Title = "LuaAdminBroadcast"
PLUGIN.Author = "Bas"
PLUGIN.Version = V(0, 1, 0)
function PLUGIN:Init()
    command.AddChatCommand("broadcastlua", self.Object, "chat_broadcast")
end
function PLUGIN:chat_broadcast(player, cmd, args)
    if player.net.connection.authLevel > 0 then
        msg="Broadcast Message"
        if args.Length > 0 then
            for i=0, args.Length-1, 1 do
                msg=msg.." "..args[i]
            end
        end
        rust.BroadcastChat("LuaAdminBroadcast",msg)
    end
end
```

CsAdminBroadcast.cs

```
namespace Oxide.Plugins
{
    [Info("CsAdminBroadcast", "Bas", "0.1.0")]
    class CsAdminBroadcast : RustPlugin
    {
        [ChatCommand("broadcastcs")]
        void chat_broadcast(BasePlayer player, string cmd, string[] args) {
            if (player.net.connection.authLevel > 0) {
                string name = "CsAdminBroadcast";
                string msg="Broadcast Message";
                if (args.Length>0) {
                    foreach (string arg in args) {
                        msg=msg+" "+arg;
                    }
                }
                string userID = "0";
                PrintToChat(name+" "+msg, userID);
            }
        }
    }
}
```

JsAdminBroadcast.js

```
var JsAdminBroadcast = {
    Title : "JsAdminBroadcast",
    Version : V(0, 1, 0),
    Author : "Bas",
    Init: function() {
        command.AddChatCommand("broadcastjs", this.Plugin, "chat_broadcast");
    },
    chat_broadcast: function(player, cmd, args)
    {
        if (player.net.connection.authLevel > 0)
        {
            msg="Broadcast Message";
            if ( args.length > 0)
            {
                for (var i = 0; i < args.length; i++)
                {
                    msg=msg+" "+args[i];
                }
            }
            rust.BroadcastChat("JsAdminBroadcast",msg);
        }
    }
}
```

PyAdminBroadcast.py

```
class PyAdminBroadcast:
    def __init__(self):
        self.Title = 'PyAdminBroadcast'
        self.Version = V(0, 0, 1)
        self.Author = 'Bas'
    def Init(self):
        command.AddChatCommand('broadcastpy', self.Plugin, 'chat_broadcast')
    def chat_broadcast(self, player, cmd, args):
        msg="Broadcast Message "
        if args:
            msg=msg+' '.join(args)
        rust.BroadcastChat("PyAdminBroadcast",msg)
```