Custom_Items

Akkarinage edited this page on Nov 5, 2016 · 1 revision

title: Custom Items permalink: /Custom_Items/

Structure

First, let's take a look at the item_db in the db folder, and its structure:

```
ID,DBName,ScreenName,Type,Price,Sell,Weight,ATK,DEF,Range,Slot,Job,Class,Gender,Loc,wLV,eLV,  Refineable,View,{Script},
{OnEquip_Script},{OnUnequip_Script}
```

- **ID**: ID of the item.
- **Aegis Name**: This first name is the DB name. When you use @item and you know the name of the item, but not the ID, type this name instead. Try making the ScreenName the same as this so you wont get confused.
- **English Name**: This is the name the server shows.
- **Type**: Purpose of the item.

```
0 = Usable : healing  2 = Usable : other  3 = Misc  4 = Armor  5 = Weapon  6 = Card  7 = Pet Egg  8 = Pet Equipment
10 = Arrow/Ammunition  11 = Usable with delayed consumption (intended for 'itemskill').
     Items using the 'itemskill' script command are consumed after selecting a target.
     Any other command will NOT consume the item.  12 = Shadow Equipment
18 = Another delayed consume that requires user confirmation before using item.
```

- **Price**: Default NPC buying price in Zeny.
- **Sell**: NPC selling price in Zeny.
- **Weight**: Weight of the item * 10. Means, an item of 0.1 weight each, must be entered as 1.
- **ATK**: Base weapon attack, in case of a weapon.

    - In **RE** enabled servers this field have a optional delimiter **:** to define this item's weaponMATK bonus, for example, **30:50** would mean the item gives 30 atk and 50 weaponMATK
- **DEF**: Base defense for armor-type items.
- **Range**: Maximum range in map cells a weapon allows to be the player apart from it's target.
- **Slot**: Amount of card slots in weapon/armor-type items.
- **Job**: Which jobs this item is available for. Values below can be combined to achieve availability for multiple job classes, i. e. `0x2|0x4 -> 0x6` (Swordman+Mage)

```
(S.) Novice         (2^00): 0x00000001  Swordman          (2^01): 0x00000002  Mage              (2^02): 0x00000004
Archer              (2^03): 0x00000008  Acolyte           (2^04): 0x00000010  Merchant          (2^05): 0x00000020
Thief               (2^06): 0x00000040  Knight            (2^07): 0x00000080  Priest            (2^08): 0x00000100
Wizard              (2^09): 0x00000200  Blacksmith        (2^10): 0x00000400  Hunter            (2^11): 0x00000800
Assassin            (2^12): 0x00001000  Unused             (2^13): 0x00002000  Crusader          (2^14): 0x00004000
Monk                (2^15): 0x00008000  Sage              (2^16): 0x00010000  Rogue             (2^17): 0x00020000
Alchemist           (2^18): 0x00040000  Bard/Dancer       (2^19): 0x00080000  Unused             (2^20): 0x00100000
Taekwon             (2^21): 0x00200000  StarGladiator     (2^22): 0x00400000  Soul Linker       (2^23): 0x00800000
Gunslinger          (2^24): 0x01000000  Ninja             (2^25): 0x02000000  Gangsi            (2^26): 0x04000000
Death Knight        (2^27): 0x08000000  Dark Collector    (2^28): 0x10000000  Kagerou/Oboro     (2^29): 0x20000000
Rebellion           (2^30): 0x40000000  All Classes             : 0xFFFFFFFF  Every Job Except Novice   : 0xFFFFFFFE
```

- **Class**: Specifies whether the item can be used by normal, baby or reborn/rebirth classes (trans.). Values below can be combined, i. e. `1|4 -> 5`, Normal+Baby Classes (excl. Third Baby classes)
    - Note: For Pre-Renewal Setting `2` enables the item for rebirth/reborn (trans.) classes and 3rd classes, and 8 can be used for all kinds of Third Classes.

```
1  = Normal  2  = Reborn/Trans. Classes (High Wizards, Champions etc.) (excl. Trans-3rd classes)
4  = Baby Classes (excl. 3rd Baby Classes)  8  = 3rd Classes (excl. Trans-3rd classes and 3rd Baby classes)  16 = Trans-
3rd Classes  32 = Baby 3rd Classes
```

- **Gender**: Gender restriction for the item.

```
0 = Female  1 = Male  2 = No restriction (both)
```

- **Loc**: Equipment location of armor and arrow-type items. Values below can be combined, i. e. 136 would indicate both accessory slots (typical value for accessories).

```
(2^0)        1 = Lower headgear  (2^1)        2 = Right hand  (2^2)        4 = Garment/Robe  (2^3)        8 = Accessory 1
(2^4)       16 = Armor  (2^5)       32 = Left hand  (2^6)       64 = Shoes  (2^7)      128 = Accessory 2
(2^8)      256 = Upper headgear  (2^9)      512 = Middle headgear  (2^10)    1024 = Costume Upper headgear
(2^11)    2048 = Costume Middle headgear  (2^12)    4096 = Costume Lower headgear  (2^13)    8192 = Costume Garment/Robe
(2^15)   32768 = Arrow (arrow-type items only)  (2^16)   65536 = Shadow Armor  (2^17)  131072 = Shadow Weapon
(2^18)  262144 = Shadow Shield  (2^18)  524288 = Shadow Shoes  (2^20) 1048576 = Shadow Accessory 2
(2^21) 2097152 = Shadow Accessory 1
```

- **wLV**: Weapon level of an item (1-4), other items 0.
- **eLV**: The minimum base level required for using/equipping the item.
- **Refineable**: Whether the item is available for refining (1) or not (0).
- **View**: Specifies the client-side look for the item.
  - Weapon-type items:
    a. Daggers
    b. One-Handed Swords
    c. Two-Handed Swords
    d. One-Handed Spears
    e. Two-Handed Spears
    f. One-Handed Axes
    g. Two-Handed Axes
    h. Maces
    i. *(not used)*
    j. Wand/Staff
    k. Bows/Crossbows
    l. Knuckle Weapons
    m. Musical Instruments
    n. Whips

    o. Books
    p. Katars
    q. Revolvers
    r. Rifles
    s. Shotguns
    t. Gatling guns
    u. Grenade launchers
    v. Fuuma shuriken
    w. Two-handed staves
    x. *Max Type*
    y. Dual-wield Daggers
    z. Dual-wield Swords
    aa. Dual-wield Axes
    ab. Dagger + Sword
    ac. Dagger + Axe
    ad. Sword + Axe
- Shield-type items:
    a. Guard, Novice Guard
    b. Buckler
    c. Shield, Holy Guard, Evangelist
    d. Mirror Shield
- Ammunition-type items:
    a. Arrows
    b. Throw-able daggers
    c. Bullets
    d. Shells
    e. Grenades
    f. Shuriken

    g. Kunai
    h. Cannonballs
    i. Throwable Items (Sling Item)
  - Headgear-type items: Please see the View IDs section of this guide.
- **{Script}**: This is where you put your item bonus. Whether it's an usable item or an equip item, it will take effect according to the item type.
- **{OnEquip_Script}**: This is where the bonus you want to be applied upon equipping goes. It will only execute as soon as you equip the item.
- **{OnUnequip_Script}**: This is where the bonus you want to be applied upon unequipping goes. It will only execute as soon as you unequip the item.

In the doc folder look for a text file called "item_bonus", it shows all the scripts items can have and how they work.

Structure (Database)

First, let's take a look at item_db table of your Ragnarok database:

```
+-----------------+----------------------+------+-----+---------+-------+
| Field           | Type                 | Null | Key | Default | Extra |
+-----------------+----------------------+------+-----+---------+-------+
| id              | smallint(5) unsigned | NO   | PRI | 0       |       |
| name_english    | varchar(50)          | NO   | UNI |         |       |
| name_japanese   | varchar(50)          | NO   |     |         |       |
| type            | tinyint(2) unsigned  | NO   |     | 0       |       |
| price_buy       | mediumint(8) unsigned | YES  |     | NULL    |       |
| price_sell      | mediumint(8) unsigned | YES  |     | NULL    |       |
| weight          | smallint(5) unsigned | NO   |     | 0       |       |
| attack          | smallint(5) unsigned | YES  |     | NULL    |       |
| defence         | smallint(5) unsigned | YES  |     | NULL    |       |
| range           | tinyint(2) unsigned  | YES  |     | NULL    |       |
| slots           | tinyint(2) unsigned  | YES  |     | NULL    |       |
| equip_jobs      | int(10) unsigned     | YES  |     | NULL    |       |
| equip_upper     | tinyint(2) unsigned  | YES  |     | NULL    |       |
| equip_genders   | tinyint(1) unsigned  | YES  |     | NULL    |       |
```

```
| equip_locations | mediumint(7) unsigned | YES  |     | NULL    |       |
| weapon_level    | tinyint(1) unsigned   | YES  |     | NULL    |       |
| equip_level     | tinyint(3) unsigned   | YES  |     | NULL    |       |
| refineable      | tinyint(1) unsigned   | YES  |     | NULL    |       |
| view            | smallint(5) unsigned  | YES  |     | NULL    |       |
| script          | text                  | YES  |     | NULL    |       |
| equip_script    | text                  | YES  |     | NULL    |       |
| unequip_script  | text                  | YES  |     | NULL    |       |
+-----------------+-----------------------+------+-----+---------+-------+
```

- **id**: ID of the item.
- **name_english**: This is the friendly name version of the item in the Database. When you use @item and you know the name of the item, but not the ID, type this name instead. Try making the ScreenName the same as this so you wont get confused. *Convert spaces to underscores (_)*.
- **name_japanese**: This is the in-game name of the item.
- **Type**: Purpose of the item.

```
0 = Usable : healing
2 = Usable : other
3 = Misc
4 = Armor
5 = Weapon
6 = Card
7 = Pet Egg
8 = Pet Equipment
10 = Arrow/Ammunition
11 = Usable with delayed consumption (intended for 'itemskill').
  Items using the 'itemskill' script command are consumed after selecting a target.
  Any other command will NOT consume the item.
12 = Shadow Equipment
18 = Another delayed consume that requires user confirmation before using item.
```

- **price_buy**: Default NPC buying price in Zeny.
- **pricesell**: NPC selling price in Zeny.
- **weight**: Weight of the item divided by 10. An item of 0.1 weight in-game, must be entered as 1 in Database. *(1 / 10 = 0.1)*.

- **attack**: Base weapon attack. Use **NULL** if non-weapon item.
  - In **RE** enabled servers this field have a optional delimiter **:** to define this item's weaponMATK bonus, for example, **30:50** would mean the item gives 30 atk and 50 weaponMATK
- **defence**: Base defense for armor-type items. Use **NULL** for weapon item and usually **0** for headgears, costumes, etc.
- **range**: Maximum range in map cells a weapon allows to be the player apart from it's target.
- **slots**: Amount of card slots in weapon/armor-type items.
- **equip_jobs**: Which jobs this item is available for. Values below can be combined to achieve availability for multiple job classes. **Note:** This is an **int** type column and does not accept **Hex** values. Either search for *Hex to decimal* on Google or use **CAST(0xFFFFFFFF AS INT)** as value when INSERTING or UPDATING. I only tested it on **MariaDB**. Change 0xFFFFFFFF to your needs.

```
(S.) Novice         (2^00): 0x00000001
Swordman            (2^01): 0x00000002
Mage                (2^02): 0x00000004
Archer              (2^03): 0x00000008
Acolyte             (2^04): 0x00000010
Merchant            (2^05): 0x00000020
Thief               (2^06): 0x00000040
Knight              (2^07): 0x00000080
Priest              (2^08): 0x00000100
Wizard              (2^09): 0x00000200
Blacksmith          (2^10): 0x00000400
Hunter              (2^11): 0x00000800
Assassin            (2^12): 0x00001000
''Unused''              (2^13): 0x00002000
Crusader            (2^14): 0x00004000
Monk                (2^15): 0x00008000
Sage                (2^16): 0x00010000
Rogue               (2^17): 0x00020000
Alchemist           (2^18): 0x00040000
Bard/Dancer         (2^19): 0x00080000
''Unused''              (2^20): 0x00100000
Taekwon             (2^21): 0x00200000
StarGladiator       (2^22): 0x00400000
Soul Linker         (2^23): 0x00800000
Gunslinger          (2^24): 0x01000000
Ninja               (2^25): 0x02000000
Gangsi              (2^26): 0x04000000
```

```
Death Knight        (2^27): 0x08000000
Dark Collector      (2^28): 0x10000000
Kagerou/Oboro       (2^29): 0x20000000
Rebellion           (2^30): 0x40000000
All Classes               : 0xFFFFFFFF
Every Job Except Novice   : 0xFFFFFFFE
```

- **equip_upper**: Specifies whether the item can be used by normal, baby or reborn/rebirth classes (trans.). Values below can be combined, i. e. `1|4 -> 5`, Normal+Baby Classes (excl. Third Baby classes)
  - Note: For Pre-Renewal Setting `2` enables the item for rebirth/reborn (trans.) classes and 3rd classes, and 8 can be used for all kinds of Third Classes.

```
1  = Normal
2  = Reborn/Trans. Classes (High Wizards, Champions etc.) (excl. Trans-3rd classes)
4  = Baby Classes (excl. 3rd Baby Classes)
8  = 3rd Classes (excl. Trans-3rd classes and 3rd Baby classes)
16 = Trans-3rd Classes
32 = Baby 3rd Classes
```

- **equip_genders**: Gender restriction for the item.

```
0 = Female
1 = Male
2 = No restriction (both)
```

- **equip_locations**: Equipment location of armor and arrow-type items. Values below can be combined, i. e. 136 would indicate both accessory slots (typical value for accessories).

```
(2^0)        1 = Lower headgear
(2^1)        2 = Right hand
(2^2)        4 = Garment/Robe
(2^3)        8 = Accessory 1
(2^4)       16 = Armor
(2^5)       32 = Left hand
```

```
(2^6)       64 = Shoes
(2^7)      128 = Accessory 2
(2^8)      256 = Upper headgear
(2^9)      512 = Middle headgear
(2^10)    1024 = Costume Upper headgear
(2^11)    2048 = Costume Middle headgear
(2^12)    4096 = Costume Lower headgear
(2^13)    8192 = Costume Garment/Robe
(2^15)   32768 = Arrow (arrow-type items only)
(2^16)   65536 = Shadow Armor
(2^17)  131072 = Shadow Weapon
(2^18)  262144 = Shadow Shield
(2^18)  524288 = Shadow Shoes
(2^20) 1048576 = Shadow Accessory 2
(2^21) 2097152 = Shadow Accessory 1
```

- **weapon_level**: Weapon level of an item (1-4), other items 0.
- **equip_level**: The minimum base level required for using/equipping the item.
- **refineable**: Whether the item is available for refining (1) or not (0).
- **view**: Specifies the client-side look for the item. *Tip:* Add your items first in your *accname* and *accessoryid*. The value you will put there is the same value you will put here.
    - Weapon-type items:
        a. Daggers
        b. One-Handed Swords
        c. Two-Handed Swords
        d. One-Handed Spears
        e. Two-Handed Spears
        f. One-Handed Axes
        g. Two-Handed Axes
        h. Maces
        i. *(not used)*
        j. Wand/Staff
        k. Bows/Crossbows

    l. Knuckle Weapons
    m. Musical Instruments
    n. Whips
    o. Books
    p. Katars
    q. Revolvers
    r. Rifles
    s. Shotguns
    t. Gatling guns
    u. Grenade launchers
    v. Fuuma shuriken
    w. Two-handed staves
    x. *Max Type*
    y. Dual-wield Daggers
    z. Dual-wield Swords
    aa. Dual-wield Axes
    ab. Dagger + Sword
    ac. Dagger + Axe
    ad. Sword + Axe
- Shield-type items:
    a. Guard, Novice Guard
    b. Buckler
    c. Shield, Holy Guard, Evangelist
    d. Mirror Shield
- Ammunition-type items:
    a. Arrows
    b. Throw-able daggers
    c. Bullets

    d. Shells
    e. Grenades
    f. Shuriken
    g. Kunai
    h. Cannonballs
    i. Throwable Items (Sling Item)
  - Headgear-type items: Please see the View IDs section of this guide.
- **script**: This is where you put your item bonus. Whether it's an usable item or an equip item, it will take effect according to the item type. Use **NULL** if none
- **equip_script**: This is where the bonus you want to be applied upon equipping goes. It will only execute as soon as you equip the item. Use **NULL** if none
- **unequip_script**: This is where the bonus you want to be applied upon unequipping goes. It will only execute as soon as you unequip the item. Use **NULL** if none

In the doc folder look for a text file called "item_bonus", it shows all the scripts items can have and how they work.

Defining Items clientside (Renewal Clients <

2012-04-10a & Main Clients <= 2012-07-10a)= After you make your item, put it in your item_db2.txt, so that you will not run into conflicts when updating the SVN later. Then go to your data folder and modify the following files, one after another.

idnum2itemdisplaynametable.txt

This file holds the item names, as displayed inside the client. Each item added as:

```
ItemID#ItemName#
```

ItemID is the number from your ID column inside the item db. If your item name contains spaces, replace those with _ (underscore).

idnum2itemdesctable.txt

This file contains the description of your item, when it is right-clicked. The syntax for an item is:

```
ItemID#  Item Description Line 1  Item Description Line 2  #
```

You can use any amount of lines for the description. If your description contains a #, make sure it is followed by a space character.

idnum2itemresnametable.txt

This file sets the inventory and drop-sprite for an the item. Item is defined as follows:

```
ItemID#SpriteName#
```

The SpriteName is the name of the files in collection and item folders inside texture and the ones inside sprite folder. If you want to use the look of an another item, search for it's id and copy the sprite name from there.

itemslotcounttable.txt

In this file you set the amount of visible slots for the item. You do not need to add 0 slot items or items, which do not have slots at all (i. e. usable items). Syntax:

```
ItemID#NumberOfSlots#
```

NumberOfSlots should be a number from 1-4, larger values might have undesired effects.

Files for non-identified items

The following files specify the name, description and look for items, which have not yet been identified. Their syntax follows the same rules as for the files with *idnum* prefix. Normally the same text is entered for non-equipment and generic description (which can be copied from official items) is used for equipment of all sorts (weapons, armor, headgears).

- num2itemdesctable.txt
- num2itemdisplaynametable.txt
- num2itemresnametable.txt

Defining Items Clientside (For New Clients)

For newer clients all the information which were earlier specified in txt files have been combined into a single file :

System/ItemInfo.lub

Since ItemInfo.lub is compiled, you need to decompile it to lua , add your customs and recompile back to lub file (Some have said that renaming a decompiled file to lub also works) Syntax:

```
[ ] = {     unidentifiedDisplayName =   ,     unidentifiedResourceName =   ,     unidentifiedDescriptionName = {   , ,   },
   identifiedDisplayName =   ,     identifiedResourceName =   ,     identifiedDescriptionName = {    },     slotCount =   ,
   ClassNum =  <View ID - yes the same one that was there item_db>    },
```

You can also put the different values in DescriptionName in separate lines.

Example: Lets say i want to add a weapon to item id 25000 - a One handed sword named Devastator with 3 slots & the image files & drop sprite files are all named Black_Sword.* . Also i want to display a Sword when not magnified.

```
[25000] = {     unidentifiedDisplayName = "Sword",     unidentifiedResourceName = "¼Òµå",
   unidentifiedDescriptionName = { "Unidentified item, can be identified with [Magnifier]." },
   identifiedDisplayName = "Devastator",     identifiedResourceName = "Black_Sword",     identifiedDescriptionName = {
                              "An Unholy Sword that was created with the sole purpose of destruction",
                              "Class :^777777 Sword^000000",                                  "Attack :^777777 325^000000",
                              "Weight :^777777 80^000000",                                 "Weapon Level :^777777 4^000000",
                              "Required Level :^777777 100^000000",
                              "Applicable Job :^777777 Novice, Swordsman Class, Merchant Class, Thief Class^000000"
                            },     slotCount = 3,     ClassNum = 2   },
```

Allocating Items on Client Side

Your custom item will have 4 (6 in case its a headgear or garment) files:

```
1) Drop sprite -> Helmet_drop.spr   2) Drop act file -> Helmet_drop.act  3) Item Inventory image -> Helmet_item.bmp
4) Item Collection image -> Helmet_collection.bmp (the one you see on its description window)  5) Headgear View sprite -
> Helmet.spr  6) Headgear View act file -> Helmet.act
```

Lets say i downloaded a custom item called Helmet which serves as a headgear and the above six files were in them.

Step 1:

We place the first four files based on what we specified in id2numresnametable.txt (IdentifiedResourceName in Iteminfo.lub). For our example lets say i used "Helmet" as the resource name :

```
i)   Copy Helmet_drop.spr to [RO Folder]\data\sprite\¾ÆÀÌÅÛ\Helmet.spr
ii)  Copy Helmet_drop.act to [RO Folder]\data\sprite\¾ÆÀÌÅÛ\Helmet.act
iii) Copy Helmet_item.bmp to [RO Folder]\data\texture\À¯ÀúÀÎÅÍÆäÀÌ½º\item\Helmet.bmp
iv)  Copy Helmet_collection.bmp to [RO Folder]\data\texture\À¯ÀúÀÎÅÍÆäÀÌ½º\collection\Helmet.bmp
```

Step 2 (only for Headgears):

For displaying headgear on the character there will be two additional files (sprite & act) or 4 if the sprite author intended for a separate set of files for male & female. In my example i have considered the first scenario.

The filename for headgear sprite are now specified in accname.lua file **(details of which is available in the View IDs Section)**

lets say i used

```
[ACCESSORY_IDs.HELMET] = "_Helmet",
```

then we need to:

```
i)   Copy Helmet.spr to [RO Folder]\data\sprite\¾Ç¸»ç¸®\¿©\¿©_Helmet.spr (Female)
ii)  Copy Helmet.act to [RO Folder]\data\sprite\¾Ç¸»ç¸®\¿©\¿©_Helmet.act
iii) Copy Helmet.spr to [RO Folder]\data\sprite\¾Ç¸»ç¸®\³²\³²_Helmet.spr (Male)
iv)  Copy Helmet.act to [RO Folder]\data\sprite\¾Ç¸»ç¸®\³²\³²_Helmet.act
```

Now it is ready to be used provided you have added entries properly to the lua files.

Specifications of the files

1. Drop sprites - ideally have 1 frame & its act file will be just an empty act (only header will be there inside)
2. Equip sprites - typically have 3+ frames to show images for various orientations & its act file will have the info on where to place each frame.
3. Item bmp - 24x24 size 256 bit bmp file.
4. Collection bmp - 75x100 size bmp file (usually 256 bit but there is no restriction)

How to differentiate between sprite files?

Open both sprite files in Spr Conview. The number of frames in the files can be seen at bottom (Sprite: /). If it says Sprite: 1/1 then its a drop sprite and the other is equip sprite.

Sometimes what sprite authors do is they simply copy the original sprite itself to make a drop sprite. In this case you will see both of them as identical when opened in Spr Conview & we don't need to worry about which one is which (since they are the same). If this process confused you, here is an example folder, the sprite is done by drkangel. The folder is virus free.

View IDs, Having A Custom Headgear Without Xray

Look in your data folder and make sure there is a folder named "lua files", if there is not make it.Download the most recent lub package and extract it to your lua files folder in your data folder. Download the most recent accname.lua and accessoryid.lua, put it in your datainfo folder that is inside the lua files folder. Now download this lub compiling tool, and put it in the datainfo folder. Now, Open accessoryid.lua in the datainfo folder with notepad or a text editor of your choice, go all the way down until the names stop. Add your custom gears before the } symbol. You should use a high number just like with your custom item's database ID, or use unused IDs to be sure you won't have to redo them all in the future, or at least for a while. Example:

```
ACCESSORY_PINKBUNNY_HAIRBAND = 663,     ACCESSORY_GREENBUNNY_HAIRBAND = 664,     ACCESSORY_OLD_ELFEAR = 665,
ACCESSORY_THA_MAERO_MASK = 666,     ACCESSORY_THANATOS_MAI_MASK = 667,     --668 free     ACCESSORY_FISHPIN = 669,
ACCESSORY_CUSTOM_HAT = 900,  }
```

You must compile to lub if you leave space between IDs, if you don't plan on compiling to lub, make sure you do not skip any numbers.

Now open accname.lua, do basically the same thing but just in a different way, and for the sprite. Add your item to the bottom, have it look like the ones above it, except have your items name and the item's sprite name.

```
[ACCESSORY_IDs.ACCESSORY_PINKBUNNY_HAIRBAND] = "_ÇÎÅ©Åä³¢¸Ó¸®¶ì",
[ACCESSORY_IDs.ACCESSORY_GREENBUNNY_HAIRBAND] = "_±×¸°Åä³¢¸Ó¸®¶ì",     [ACCESSORY_IDs.ACCESSORY_OLD_ELFEAR] = "_°í´ë¿äÁ¤ÀÇ±Í",
[ACCESSORY_IDs.ACCESSORY_THA_MAERO_MASK] = "_Å¸³ªÅä½ºÀÇ½½ÇÄ°¡¸é",
[ACCESSORY_IDs.ACCESSORY_THANATOS_MAI_MASK] = "_Å¸³ªÅä½ºÀÇÁõ¿À°¡¸é",     [ACCESSORY_IDs.ACCESSORY_FISHPIN] = "_¹°°í±âÇÉ",
[ACCESSORY_IDs.ACCESSORY_CUSTOM_HAT] = "_Sprite_Name_Here",  }
```

Delete the accname.lub and accessoryid.lub. Now that you are done, make sure luac.exe is in your datainfo folder, right click the luac.exe, create a shortcut, right click the shortcut, and click properties, in the target area, add right after the quotes, with the quotes:

```
 -o "accessoryid.lub" "accessoryid.lua"
```

So that it looks something like this:

```
 "C:\Program Files (x86)\Gravity\RO\data\lua files\datainfo\luac5.0.2.exe" -o "accessoryid.lub" "accessoryid.lua"
```

Do this also to "accname.lub" except instead of accessoryid in the quotes, do accname. If there is any lua file leftover, you may want to move it somewhere else just for future changes, like My Documents. For future reference, this is also how you turn the other lub files to lua files.

Modifications

Sprite Replacement

To replace a headgear with your customized headgear, just delete one of the items you want, and replace your customised item ID with that id.

Lets say you have something like this:

```
15000,Angel_Wing,Angel Wing,4,,10,10,0,,1,0,0,3,0,0,0,0,0,{},{},{}
```

And there's an item you wont need, such as:

```
2220,Hat,Hat,5,1000,,200,,2,,0,10477567,2,256,,0,1,16,{},{},{}
```

Well, to replace it, just delete everything in the hat line but its ID, and paste there your custom item, all the line but the ID. Should be like this:

2220 ,Hat,Hat,5,1000,,200,,2,,0,10477567,2,256,,0,1,16,{},{},{} 15000, **Angel_Wing,Angel Wing,4,,10,10,0,,1,0,0,3,0,0,0,0,0,{},{},{}** vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv **2220,Angel_Wing,Angel Wing,4,,10,10,0,,1,0,0,3,0,0,0,0,0,{},{},{}**

All you need to do now is changing the View ID to the Hat, and change your file names to its name. You can find them on the idnum2resnametable.txt Search for 2220 and the gibberish besides it, w/o the # # is the file name, then rename it to your custom item name.

Remember to add the files to data\sprite\¾ÆÀÌÅÛ (Icon sprite) and both data\sprite\¾Ç¼º»ç¸®\[32] (Male Item Sprite) and data\sprite\¾Ç¼º»ç¸®\¿© (Female Item Sprite). Remember, these 2 last item names, must have either a [32]_ for male or ¿©_ for female, at the beginning of the name, both .spr and .act. Also, don't forget to change the other files:

idnum2itemdisplaynametable.txt , idnum2itemdesctable.txt , idnum2itemdesctable.txt , num2itemdesctable.txt , num2itemdisplaynametable.txt , num2itemresnametable.txt and itemslotcounttable.txt

Use these files if you want to replace sprites, they will be useful.

- **actOR:** with the actor you modify the act file, which is the one who tells the client where does the weapon go and what does it have to do and at a certain moment.
- **SPR Conview:** you can use these to view which sprite is which, and also see sequence per sequence.

Item Restrictions

Look on your db folder for a file called item_trade.txt and open it

Now, the pattern for a flag is:

```
Item ID, TradeMask, GM-Level Override
```

- **Item ID**: the ID of your item.
- **TradeMask**: Testrictions the item will have, such as being dropped, stored or traded. These values can be combined to achieve multiple effects.

```
1:Item can't be dropped  2:Item can't be traded (nor vended)  4:Item can only be traded with wedded partner
8:Item can't be sold to NPCs  16:Item can't be placed in the cart  32:Item can't be placed in the storage
64:Item can't be placed in the guild storage
```

- **GM-Level Override**: This is the minimum GM level a player must have to avoid these restrictions.

Item Script

Okay, I'll update this as I go; otherwise, I wont have anything new to show. First with usable items.

Well, there are 3 types of usables:

1.- Healing items, they have in the type field a 0.

2.- Other uses items, such as fly wings and so, which doesn't heal, but does have an instant effect, they have in the type field a 2.

3.- Usable items, which are skills that requires a target selection, therefore they have a delayed use, and in newer SVNs the item wont be gone until the target is chosen and the cast has finished. Means, you wont lose the item if the skill hasn't been done. They have in the type field an 11.

Now, for an item skill, the pattern for a skill is:

```
itemskill  , ;
```

Without <> of course.

Now, we go to our db\skill_db.txt and search for the blessing ID, searching for AL_ or blessing, but first, the structure of a skill:

```
id,range,hit,inf,element,nk,splash,max,list_num,castcancel,cast_defence_rate,inf2,maxcount,skill_type,blow_count,name,description
```

Now, the AL_BLESSING or Blessing Skill:

```
34,9,6,16,0,0x1,0,10,1,yes,0,0,0,magic,0,  AL_BLESSING,Blessing
```

We grab the ID and put it on our code:

```
itemskill 34, ;
```

Don't mind the actual level of the skill_db.txt for the skill level part. You can actually put the level you want here below 99, so lets pick 4:

```
itemskill 34,4;
```

And finally, insert it on the code.

```
  ,{ itemskill 34,4; },{},{}
```

There, we have a Lvl 4 Blessing item skill.

Remember, one space before and one after the script, NOT TAB!

And so on with other skills. Just remember these steps and they should be fine. And remember, itemskill is NOT the same as skill. itemskill will make you CAST that skill, while skill will GIVE you the skill. If you give an item, whether delayed use or normal usable two itemskill commands, only the latter will come into effect, as you cannot cast two skills at the same time.

Now, for Potions.

Instead of using itemskill, we will use itemheal:

```
,{ itemheal 100,0; },{},{}
```

The green is the HP, the blue is the SP. To add that random heal amount effect, just pick a set amount, let's say 120, then pick 1 or 2 numbers, up to you. Then, add the rand() command, like this:

```
,{ itemheal rand(120,9,49),0; },{},{}
```

Explanation: This item will heal about 49~120 hp and 0 SP. To heal SP you change the 0 to a number. Like this:

```
,{ itemheal rand(120,9,49),50; },{},{}
```

It will now heal 50 SP. Now, for skilleffect and sc_start.

Okay, let's see. Here's a example of an item casting a skill and simulating a skill.

```
,{ itemskill 34,10; skilleffect 29,0; sc_start 12,140000,5; },{},{}
```

skilleffect will display on your client a skill visual effect. The pattern is:

```
skilleffect  , ;
```

skill ID Self-explanatory.

number Have you seen those skills that shows up numbers upon casting? Such as heal? Well, putting a number there will display it. But only works on skills with numbers floating.

sc_start has the following pattern:

```
sc_start  , , [, ];
```

status id Refer to const.txt in your db folder for all effects. Open it with a text editor then search by either clicking edit, then find, or press and hold the ctrl key, and while still holding it press the f key, then release, and search for "SC_ALL -1". Everything below this is an effect. The effects stop at "e_gasp 0" which are emoticons.

duration Is the amount time in milliseconds (1000 msec = 1 sec) that the status will be active.

val1 Typically it's the level of the skill, associated with this status.

unit id Allows to start the status on an other unit, than the attached player. For players this is account id. This parameter is optional, and can be omitted.

Now, for the item I said, it will show you the visual effect of Blessing and cast on you a lvl 5 Agi-Up! skill for 140 seconds.

For more item scripts see script_commands.txt in your doc folder, as well as item_bonus.txt in your doc folder.

==Weapon Sprite Solution (Renewal Clients <= 2012-04-10a & Main Clients <= 2012-07-10a)== For these clients, Weapons are limited to use a range of Item IDs hardcoded in the clientn for each type. For e.g.

```
1265,Bloody_Roar,Bloody Roar,4,,10,1000,120,,1,0,4096,7,2,34,4,75,1,16,
{ bonus bIgnoreDefRace,RC_DemiHuman; bonus bFlee,-160; bonus bFlee2,-160; bonus bNoRegen,1; bonus bNoRegen,2; },{},{}
1266,Test,Test,4,4000,2000,10,165,,1,0,4096,7,2,34,3,33,1,16,{}

// 1-Handed Axes  1301,Axe,Axe,4,500,,800,38,,1,3,8803555,7,2,2,1,3,1,6,{}
```

Here the item 1266 is a custom katar and it does show up as a katar (if you have the proper sprite files ofcourse). but if i use some id like say 22000, client wont display it. So what is the range of item ids you can use? Look below:

- One handed Swords = 1100-1149, 13400-13499
- Two handed Swords = 1150-1199, 21000-21999
- Knives, Daggers etc = 1200-1249, 13000-13099
- Katars = 1250-1299 ; Has 35 free IDs
- One handed Axes = 1300-1349; Has 43 free IDs
- Two handed Axes = 1350-1399; Has 32 free IDs
- One handed Spears = 1400-1449; Has 34 free IDs
- Two Handed Spears = 1450-1471, 1474-1499
- Maces = 1500-1549, 16000-16999
- Books = 1550-1599 ; Has only 2 IDs.
- Knuckles = 1800-1899 ; Has 95 free IDs
- One Handed Staves/Rods = 1600-1699; Has 79 free IDs
- Two Handed Staves/Rods = 1472,1473,2000-2099
- Bows = 1700-1749, 18100-18499

- Guitars = 1900-1949 ; Has 32 free IDs
- Whips = 1950-1999 ; Has 130 free IDs

- Handguns = 13100-13149
- Other guns = 13150-13199

- Ninja weapons = 13300-13399

The number of unused Item IDs left known for a range has also been mentioned above. Best practice to follow check in your range in official db before adding custom weapon.

Weapon Sprite Solution (For New Clients)

For new clients the view id system is also applicable to client. To add a custom weapon you need to first edit a file called weapontable.lub in your data folder

```
data/luafiles514/lua files/datainfo/weapontable.lub
```

I will be adding **Oriental_Sword** which will be a 1-Handed sword. Open weapontable.lub. First you will see a table called **Weapon_IDs**. Take note of the first 30 values in this table - these are the only available Weapon types in the client right now.

Anyways go to the last entry which should be for Wizardy Staff = 97. You can use a view id after that like shown below

```
 WEAPONTYPE_Oriental_Sword = 98,
```

Come down and you see the next table called **WeaponNameTable**. Here is where you add your sprite name suffix. What do i mean by that? Its the last part in your weapon sprite & act file. Before it used to be _.

```
data/sprite/ / _ .spr
```

OK Back to topic. so I add my entry like shown below.

```
[Weapon_IDs.WEAPONTYPE_Oriental_Sword] = "_Oriental"
```

Lastly come down further in weapontable.lub and you see the last table called **Expansion_Weapon_IDs**. Remember the 30 types i told you to take note of ? here we assign one of those to our custom (like a mapping or connection). Since mine is a 1-Handed Sword I specify it like below.

```
[Weapon_IDs.WEAPONTYPE_Oriental_Sword] = Weapon_IDs.WPCLASS_WEAPONTYPE_SWORD
```

Now for the most important part. For our client to actually pick up all these details we need to provide the view id which we used in Weapon_IDs table as the ClassNum value in ItemInfo.lua. Check the ItemInfo.lub format shown above for details.

With this your weapon sprite will become visible while attacking.