D2R Loot Filter

This document will explain how to use the loot filter, some of it's requirements and what kind of options it currently supports. This filter is likely to change much in time, and will have versions listed in the script.

Requirements:

- Must be using the MP-Enabled client files (D2RLAN Recommended)
- Must be using D2RHUD-MP version 1.0.6 or later
- Must have loot filter files: lootfilter.lua, lootfilter_config.lua

Capabilities/Usage:

This tool can be used to both hide unwanted items, as well as change their names.
It is an active filter, meaning it can dynamically perform these actions, based on conditions/rules that you have determined in your config file. This includes special actions such as a drop notification when filtered items appear, etc.

It also has advanced "search" capabilities that will allow you to make rules based on the item's given stats, it's quality or rarity, how many sockets it has, it's ethereal status, where it dropped, and truly, much much more. Use it to make "screen clutter" an issue of the past or make certain items stand out visually based on your needs.

How it works:

For transparency, I will explain how this process works, and what you will need to do as a player to take full advantage of it's benefits. Keep in mind, all of this work is done by volunteers, in their spare time.

D2RHUD-MP: This acts as the muscle behind the operation. It interacts with the game directly, and uses the game's internal command logic to perform special actions.
LootFilter.LUA: This acts as the brains behind the configuration file (below), converting the user-friendly config file format into logic that D2RHUD is able to process.
LootFilter_Config.LUA: This is the end-user file, the one you will edit to customize your loot filter. It uses no carding jargon, and will also have user-made templates easily loadable.

Loot Filter Configuration:

This tool provides a wide array of different conditions you can apply, as well as data you can retrieve for various output purposes. Below, I will list all of the available commands, as well as an example rule we can use to get you started. These commands will be listed by their type.
Any command omitted from the rule, is assumed to be ignored/false, and is not required.
For many of these commands, you will need to reference the .txt files directly. Many mod authors should include things like itemcodes, level names, etc in their config files.

Basic Item Conditions:

These are conditions you will use very frequently for most of your rules

code or codes: This uses a 3-character itemcode to determine which items should be affected. If no other conditions are set, it will match all items of that code or codes.
A special case exists if using codes: "allitems" to match all items.
Example Usage: code = "abc" or codes = { "abc", "def", "ghi" }
Value Output: {code}

index: This uses an integer to determine the unique identifier for set/unique items.
For example, to determine if an unidentified ring is the Stone of Jordan or not, you would check index 122, the unique identifier for that unique.
Example Usage: index = 122
Value Output: {index}

quality: This uses an integer to determine the quality of the item being filtered, such as magic, rare, unique, etc. Values 1-9 are accepted (table shown in lootfilter.lua), and ranges can be used for greater than less than, etc.
Example Usage: quality = "7" or quality = "6+" or quality = "4-"
Value Output: {quality}

rarity: This uses an integer to determine the rarity of the item being filtered; Normal, Exceptional or Elite. Values 0-2 are accepted (table shown in lootfilter.lua), and ranges can be used for greater than less than, etc.
Example Usage: rarity = "2" or rarity = "1+" or rarity = "2-"
Value Output: {rarity}

Advanced Item Conditions:

These are slightly more advanced conditions you will find helpful for customization

sockets: This uses an integer to determine the socket count of the item being filtered.
Values 0-6 are accepted and ranges can be used for greater than less than, etc.
Example Usage: sockets = "2" or sockets = "3+" or sockets = "5-" or sockets = "1, 2"
Value Output: {sockets}

ethereal: This uses a boolean flag (True or False) to determine filtering for ethereal items.
Example Usage: ethereal = true
Value Output: {ethereal}

stat: This uses a small groupset to lookup specific item stats for filtering purposes.
It can take 1 or 2 arguments for the stat, if the stat also uses a parameter.
It can also compare ranges of stats, to match target values in between.
The index is the stat# in question, so if we check stat 3, we are checking vitality.
If using a parameter, we can change index to 83 to check for +2 paladin skills using param 3
Example Usage (No Param): stat = { index = 3, op = ">=", value = 100 }
Example Usage (Param): stat = { index = 83, op = "==", value = 2, param = 3 }
Value Output: {stat=(xx)}

pstat: This uses the same logic as the "stat" command, but applies to the player instead.
Example Usage (No Param): pstat = { index = 3, op = ">=", value = 100 }
Example Usage (Param): pstat = { index = 83, op = "==", value = 2, param = 3 }
Value Output: {pstat=(xx)}

location: This uses either "onground", "onplayer" or "equipped" as conditions for filtering.
It can be used to provide varying outputs depending on location, or for restriction purposes.
The default behavior when no location is specified is "onground".
Example Usage: location = "equipped"

difficulty: This uses either "Normal", "Nightmare" or "Hell" as conditions for filtering.
It can be used to provide varying outputs depending on difficulty, or for restriction purposes.
The default behavior when no difficulty is specified, is all difficulties.
Example Usage: difficulty = "Hell" or difficulty = "Nightmare-" or difficulty ="Normal, Hell"

area: This uses the level name from the mod to conditionally filter items by area.
You may have multiple rules for the same items, depending on the area they drop in.
Example Usage: area = “Blood Moor” or area = “The Chaos Sanctuary”

Output Control Actions:

These are actions you can perform to hide, alter or notify you of filtered items.
All text strings support color commands such as {orange} or {red} , etc

hide: This uses a boolean flag (True or False) to determine hide status.
This command is optional if the desired behavior is hide = false.
Example Usage: hide = true

default_hide: This uses a boolean flag (True or False) to determine hide status.
Similar to the “hide” command, but is used for hiding all items that do not meet the conditions
Example Usage: default_hide = true

name_override: This uses a text string to override the default item name.
Can also use newline characters (\n) to add more ‘depth’ to your name.
Example Usage: name_override = “LootFiltered 5000”

prefix: This uses a text string to insert text before the item name (altered or original).
Also useful for including item indicators or reporting item/player stats.
Example Usage: prefix = “[III]” or prefix = “STR = {stat=(0)}”

prefix: This uses a text string to insert text after the item name (altered or original).
Also useful for including item indicators or reporting item/player stats.
Example Usage: suffix = “[III]” or suffix = “STR = {stat=(0)}”

notify: This uses a boolean flag (True or False) to display drop notifications.
It will display a chat text message that your filtered item has dropped.
Example Usage: notify = true

notify_message: This uses a text string to display a custom message instead of the default drop notification message. The item name will still be shown after the custom message.
It requires the command notify = true to be set.
Example Usage: notify_message = "Let's Gooooooooo: "

Example Rules:

For those of you thinking, "So it seems nice and all, but how do I use it?", let's discuss.
Here are some example rules that can be applied to achieve varying results.
Keep in mind that these are highly customizable and there may also be user-made templates made available to you already.

Example #1: Hiding all 1 socket non-magical or higher items in the game

```
{
    codes = "allitems",  – Applies to all items instead of listing item codes
    quality = "3-",  – Applies to superior and lower quality items
    sockets = "1"  – Applies to 1 socket items
    hide = true  – Hide the items that match these conditions ^^
}
```

Example #2: Displaying drop notifications for perfect Lightning Rainbow Facets only

```
{
    code = "jew",  – Applies to jewels only
    quality = "7",  – Applies to unique quality only
    index = 392,  – Unique ID for Lightning Death Facet
    stat = { index = 330,  op = "==", value = 5 }, – Checks 'passive_ltng_mastery' is 5
    stat = { index = 334,  op = "==", value = 5 }, – Checks 'passive_ltng_pierce' is 5
    notify = true,  – Show me a chat message please
    Notify_message = "{yellow}Perfect RBF Dropped: "
    – Change the chat message to include the color of the element
}
```

Example #3-1: Hide lower potions (based on level)

```
{
    codes = { "hp1", "mp1" },  – Applies to minor healing/mana potions
    pstat = { index = 12,  op = ">=", value = 25 }, – Char Level is >= 25
    hide = true  – Hide the items that match these conditions ^^
}
```

Example #3-2: Hide lower potions (based on level)

```
{
        codes = { "hp2", "mp2" },  –  Applies to minor healing/mana potions
        pstat = { index = 12,  op = ">=", value = 40 }, –  Char Level is >= 40
        hide = true  –  Hide the items that match these conditions ^^
}
```