OK this is a pretty simple enchanting script now that I know what I’m doing with it so hopefully I can it explain it a little easier than the trial and error took. First here are the commands that you will use to set it up:

setgoodenchantcommand <command> : sets what to do with the item when an enchant successfully lands

setbadenchantcommand <command> : sets what to do with the item when enchant is unsuccessful

setgoodfiltercommand <command> : sets what to do when specific filter is applied to enchanted item(needs set to use filteritems command)

setbadfiltercommand <command> : sets what to do when specific filter does not apply to enchanted item(needs set to use filteritems command)

filteritems <score> <keyword> : identifies all items currently in inventory with the keyword entered and filters them based on score.

NOTE::::::::::::::: when using first 4 commands remember to use %1 in the command part so it recognizes that you want it to do it to the item… I.E. setgoodenchantcommand put %1 bag…. if you want to have it put the successful enchants in your bag… or setbadenchantcommand drop %1 to drop an unsuccessful item to the ground(those are the 2 I use)

For the filteritems command you will need to know the score of the item before you start enchanting. To find this out use the appraise command or ‘compare <item> details’ command. From there when you filter you will have to enter the original score + the score from the max enchant. I.E. A level 1 open clan shop has the overall score of 10, hr/dr adds 5, all stats add 10 except int which adds 15) so after you solidify you will ‘filteritems 40(10 from original stats + 30 from max solidify) <item>’ get all the items out and resonate or illuminate them, get them out again and run ‘filteritems <70 for resonate, 80 for illuminate> <item> and it will get rid of the bad ones. Its not too hard once you get used to it, just remember solidify is always +30(IF YOU SET HR to equal 0 client side, filter will only put max dr in bag or whatever), resonate is +30 and illuminate is +40.

OK now for the actual commands to enchant, these are very simple: # = the amount of the item you purchase.

solidify # <item>

resonate # <item>

illuminate # <item>

Just make sure you buy the items first and it will automatically cast the spell on the number you tell it to, and yes it recasts when enchanters focus kicks in.

Here is the actual script. You will have to copy and paste everything below the dotted line and save it as a .xml file and than load it.

-------------------------------------------------------------------------------------------------------------------------

<muclient>  
<plugin  
 name="AardEnchanter"  
 author="Izocinoc"  
 id="c0cf2e77128fbe26baf59ca5"  
 language="Lua"  
 purpose="Aardwolf Enchanting Script"  
 save\_state="y"  
 date\_written="2010-06-15 12:20:00"  
 requires="4.72"  
 version="1.0"  
 >  
<description trim="y">  
<![CDATA[  
This script assists with bulk-enchanting items.  
Commands: setgoodenchantcommand, setbadenchantcommand, resonate, illuminate, solidify  
 setgoodfiltercommand, setbadfiltercommand, filteritems  
]]>  
</description>  
  
</plugin>  
  
  
<!-- Get our standard constants -->  
  
<include name="constants.lua"/>  
  
<!-- Triggers -->  
  
<triggers>  
 <!-- Triggers for detecting that illuminate, resonate, or solidify has been cast. -->  
 <trigger  
 enabled="n"  
 group="enchanting"  
 keep\_evaluating="y"  
 match="^.\* starts to glow brightly as you infuse it with holy magic\.$"  
 regexp="y"  
 script="detect\_enchant\_results"  
 sequence="1"  
 >  
 </trigger>  
 <trigger  
 enabled="n"  
 group="enchanting"  
 keep\_evaluating="y"  
 match="^.\* begins to hum softly\.$"  
 regexp="y"  
 script="detect\_enchant\_results"  
 sequence="1"  
 >  
 </trigger>  
 <trigger  
 enabled="n"  
 group="enchanting"  
 keep\_evaluating="y"  
 match="^You solidify .\* making it visible again\.$"  
 regexp="y"  
 script="detect\_enchant\_results"  
 sequence="1"  
 >  
 </trigger>  
  
 <!-- Trigger for detecting successful illuminate, resonate, or solidify. Illuminate and resonate share the same message. -->  
 <trigger  
 enabled="n"  
 group="detect\_enchant\_results"  
 keep\_evaluating="n"  
 match="^Magic pulses through .\*\, (enhancing its power|blessing it with fortune)\.$"  
 regexp="y"  
 script="handle\_good\_enchant"  
 sequence="1"  
 >  
 </trigger>  
   
 <!-- Trigger for detecting enchanter's focusing reversing an illuminate, resonate, or solidify. -->  
 <trigger  
 enabled="n"  
 group="detect\_enchant\_results"  
 keep\_evaluating="n"  
 match="^Unsatisfied with your focus\, you (darken the|silence the|return invisibility to) .\*\.$"  
 regexp="y"  
 script="recast\_enchant"  
 sequence="1"  
 >  
 </trigger>  
  
 <!-- Trigger for detecting an enchantment with no bonus. This will trigger off any line of text, with a lower priority than the triggers that detect success messages. -->   
 <trigger  
 enabled="n"  
 group="detect\_enchant\_results"  
 keep\_evaluating="y"  
 match="^.+$"  
 regexp="y"  
 script="handle\_bad\_enchant"  
 sequence="100"  
 >  
 </trigger>  
   
 <!-- Trigger for detecting item score in identify output -->  
 <trigger  
 enabled="n"  
 group="filtering"  
 keep\_evaluating="y"  
 match="^\| Score\s+\: (\d+)\s+\|\s\*$"  
 regexp="y"  
 script="handle\_score"  
 sequence="100"  
 >  
 </trigger>  
  
 <!-- Trigger for detecting when an item is not found -->  
 <trigger  
 enabled="n"  
 group="filtering"  
 keep\_evaluating="y"  
 match="^You do not have that item\.$"  
 regexp="y"  
 script="handle\_item\_not\_found"  
 sequence="100"  
 >  
 </trigger>  
  
 <!-- Trigger for detecting when a room is full -->  
 <trigger  
 enabled="n"  
 group="filtering"  
 keep\_evaluating="y"  
 match="^This room is already full\.$"  
 regexp="y"  
 script="handle\_room\_full"  
 sequence="100"  
 >  
 </trigger>  
  
   
</triggers>  
  
<!-- Aliases -->  
  
<aliases>  
 <alias  
 script="setgoodenchantcommand"  
 match="^setgoodenchantcommand\s\*(.\*)$"  
 regexp="y"  
 enabled="y"  
 sequence="100"  
 >  
 </alias>  
 <alias  
 script="setbadenchantcommand"  
 match="^setbadenchantcommand\s\*(.\*)$"  
 regexp="y"  
 enabled="y"  
 sequence="100"  
 >  
 </alias>  
 <alias  
 script="resonate"  
 match="^resonate\s+(\d+)\s+(.\*)$"  
 regexp="y"  
 enabled="y"  
 sequence="100"  
 >  
 </alias>  
 <alias  
 script="illuminate"  
 match="^illuminate\s+(\d+)\s+(.\*)$"  
 regexp="y"  
 enabled="y"  
 sequence="100"  
 >  
 </alias>  
 <alias  
 script="solidify"  
 match="^solidify\s+(\d+)\s+(.\*)$"  
 regexp="y"  
 enabled="y"  
 sequence="100"  
 >  
 </alias>  
 <alias  
 script="setgoodfiltercommand"  
 match="^setgoodfiltercommand\s\*(.\*)$"  
 regexp="y"  
 enabled="y"  
 sequence="100"  
 >  
 </alias>  
 <alias  
 script="setbadfiltercommand"  
 match="^setbadfiltercommand\s\*(.\*)$"  
 regexp="y"  
 enabled="y"  
 sequence="100"  
 >  
 </alias>  
 <alias  
 script="filteritems"  
 match="^filteritems\s+(\d+)\s+(.\*)$"  
 regexp="y"  
 enabled="y"  
 sequence="100"  
 >  
 </alias>  
</aliases>  
  
  
<!-- Script -->  
  
  
<script>  
<![CDATA[  
  
enchantment = ""  
enchants\_left = 0  
item\_keywords = ""  
  
function is\_enchant\_initialized()  
 return GetVariable("goodenchantcommand") ~= nil and GetVariable("goodenchantcommand") ~= ""  
 and GetVariable("badenchantcommand") ~= nil and GetVariable("badenchantcommand") ~= ""  
end -- is\_enchant\_initialized  
  
function is\_filter\_initialized()  
 return GetVariable("goodfiltercommand") ~= nil and GetVariable("goodfiltercommand") ~= ""  
 and GetVariable("badfiltercommand") ~= nil and GetVariable("badfiltercommand") ~= ""  
end -- is\_filter\_initialized  
  
function setgoodenchantcommand(sName, sLine, wildcards)  
 if (wildcards[1] == nil or wildcards[1] == "") then  
 Note("Usage: setgoodenchantcommand <command>")  
 else  
 SetVariable("goodenchantcommand", tostring(wildcards[1]))  
 Note("When an enchantment succeeds, the command '" .. GetVariable("goodenchantcommand") .. "' will be executed.")  
 end  
end -- setgoodenchantcommand  
  
function setbadenchantcommand(sName, sLine, wildcards)  
 if (wildcards[1] == nil or wildcards[1] == "") then  
 Note("Usage: setbadenchantcommand <command>")  
 else  
 SetVariable("badenchantcommand", tostring(wildcards[1]))  
 Note("When an enchantment fails, the command '" .. GetVariable("badenchantcommand") .. "' will be executed.")  
 end  
end -- setbadenchantcommand  
  
function setgoodfiltercommand(sName, sLine, wildcards)  
 if (wildcards[1] == nil or wildcards[1] == "") then  
 Note("Usage: setgoodfiltercommand <command>")  
 else  
 SetVariable("goodfiltercommand", tostring(wildcards[1]))  
 Note("When filtering items, for each good item the command '" .. GetVariable("goodfiltercommand") .. "' will be executed.")  
 end  
end -- setgoodfiltercommand  
  
function setbadfiltercommand(sName, sLine, wildcards)  
 if (wildcards[1] == nil or wildcards[1] == "") then  
 Note("Usage: setbadfiltercommand <command>")  
 else  
 SetVariable("badfiltercommand", tostring(wildcards[1]))  
 Note("When filtering items, for each bad item the command '" .. GetVariable("badfiltercommand") .. "' will be executed.")  
 end  
end -- setbadfiltercommand  
  
function resonate(sName, sLine, wildcards)  
 enchantment = "resonate"  
 enchants\_left = tonumber(wildcards[1])  
 item\_keywords = wildcards[2]  
   
 if ((enchants\_left == nil) or (type(enchants\_left) ~= "number") or (item\_keywords == nil) or (item\_keywords == "")) then  
 Note("Usage: resonate <number> <keyword>")  
 elseif not is\_enchant\_initialized() then  
 Note("Error: script commands are not set. Please use 'setgoodenchantcommand' and 'setbadenchantcommand' before enchanting.")  
 else  
 perform\_enchant()  
 end  
end -- resonate  
  
function illuminate(sName, sLine, wildcards)  
 enchantment = "illuminate"  
 enchants\_left = tonumber(wildcards[1])  
 item\_keywords = wildcards[2]  
   
 if ((enchants\_left == nil) or (type(enchants\_left) ~= "number") or (item\_keywords == nil) or (item\_keywords == "")) then  
 Note("Usage: illuminate <number> <keyword>")  
 elseif not is\_enchant\_initialized() then  
 Note("Error: script commands are not set. Please use 'setgoodenchantcommand' and 'setbadenchantcommand' before enchanting.")  
 else  
 perform\_enchant()  
 end  
end -- illuminate  
  
function solidify(sName, sLine, wildcards)  
 enchantment = "solidify"  
 enchants\_left = tonumber(wildcards[1])  
 item\_keywords = wildcards[2]  
   
 if ((enchants\_left == nil) or (type(enchants\_left) ~= "number") or (item\_keywords == nil) or (item\_keywords == "")) then  
 Note("Usage: solidify <number> <keyword>")  
 elseif not is\_enchant\_initialized() then  
 Note("Error: script commands are not set. Please use 'setgoodenchantcommand' and 'setbadenchantcommand' before enchanting.")  
 else  
 perform\_enchant()  
 end  
end -- solidify  
  
function perform\_enchant()  
 if enchants\_left > 0 then  
 EnableTriggerGroup("enchanting", true)  
 enchants\_left = enchants\_left - 1  
 Send("cast '" .. enchantment .. "' '" .. item\_keywords .. "'")  
 else  
 Note("Enchanting complete.")  
 EnableTriggerGroup("enchanting", false)   
 end  
  
 EnableTriggerGroup("detect\_enchant\_results", false)  
end -- perform\_enchant  
  
function detect\_enchant\_results(sName, sLine, wildcards)  
 EnableTriggerGroup("detect\_enchant\_results", true)  
end -- detect\_enchant\_results  
  
function handle\_good\_enchant(sName, sLine, wildcards)  
 command = string.gsub(GetVariable("goodenchantcommand"), "%%1", "'"..item\_keywords.."'")  
 Send(command)  
 perform\_enchant()  
end -- handle\_good\_enchant  
  
function handle\_bad\_enchant(sName, sLine, wildcards)  
 command = string.gsub(GetVariable("badenchantcommand"), "%%1", "'"..item\_keywords.."'")  
 Send(command)  
 perform\_enchant()  
end -- handle\_bad\_enchant  
  
function recast\_enchant(sName, sLine, wildcards)  
 enchants\_left = enchants\_left + 1  
 perform\_enchant()  
end -- handle\_good\_enchant  
  
function filteritems(sName, sLine, wildcards)  
 filter\_score = tonumber(wildcards[1])  
 item\_keywords = wildcards[2]  
   
 if ((filter\_score == nil) or (type(filter\_score) ~= "number") or (item\_keywords == nil) or (item\_keywords == "")) then  
 Note("Usage: filteritems <number> <keyword>")  
 elseif not is\_filter\_initialized() then  
 Note("Error: script commands are not set. Please use 'setgoodfiltercommand' and 'setbadfiltercommand' before running filteritems.")  
 else  
 EnableTriggerGroup("filtering", true)  
 Send("identify " .. item\_keywords)  
 end  
end -- filteritems  
  
function handle\_score(sName, sLine, wildcards)  
 score = tonumber(wildcards[1])  
   
 if score >= filter\_score then  
 command = string.gsub(GetVariable("goodfiltercommand"), "%%1", "'"..item\_keywords.."'")  
 else  
 command = string.gsub(GetVariable("badfiltercommand"), "%%1", "'"..item\_keywords.."'")  
 end  
   
 Send(command)  
 Send("identify " .. item\_keywords)  
end -- handle\_score  
  
function handle\_item\_not\_found(sName, sLine, wildcards)  
 EnableTriggerGroup("filtering", false)  
end -- handle\_item\_not\_found  
  
function handle\_room\_full(sName, sLine, wildcards)  
 Note("The room is full. Turning off filtering triggers.")  
 EnableTriggerGroup("filtering", false)  
end -- handle\_room\_full  
  
function OnPluginInstall()  
 Note("You have installed Izocinoc's enchanting script.")  
end -- OnPluginInstall  
  
function OnPluginConnect()  
end -- OnPluginConnect  
  
function OnPluginDisconnect()  
end -- OnPluginDisconnect  
  
function OnPluginClose()  
end -- OnPluginClose  
  
function OnPluginSaveState()  
end -- OnPluginSaveState  
  
function OnPluginEnable()  
end -- OnPluginEnable  
  
function OnPluginDisable()  
end -- OnPluginDisable  
  
function OnPluginCommand(sText)  
 -- I can use this function to parse user input from the command line  
 -- instead of using utils.msgbox for user input  
 return true  
end -- OnPluginCommand  
  
]]>  
</script>  
  
  
<!-- Plugin help -->  
  
<aliases>  
 <alias  
 script="OnHelp"  
 match="AardSpellup:help"  
 enabled="y"  
 >  
 </alias>  
</aliases>  
  
<script>  
<![CDATA[  
function OnHelp ()  
 world.Note (world.GetPluginInfo (world.GetPluginID (), 3))  
end  
]]>  
</script>   
  
</muclient>