Subject: Re: [SSGM Plugin] CheckPlayerName
Posted by iRANian on Fri, 15 Aug 2014 09:17:34 GMT
View Forum Message <> Reply to Message

In CheckName() instead of looping over every player to see if the name you want to set a player's name to is already used you can just use the function to get a cPlayer*, GameObject* or playerID by player name. If any of those functions don't return anything valid the name isn't in use.

There's also another thing you can speed up in the CheckName() function, isntead of doing:

```
  if(stristr(StringClass::getFormattedString("%ls",name),bad_words[i]))
```

You should store the WideStringClass of 'name' before entering the loop, as you're executing getFormattedString to create a StringClass from the WideStringClass everytime when entering the loop.

While reading in the bad words list instead of using WordCount= to find out how many bad words there are in the list you can use the following two INIClass functions:

```
 int Entry_Count(char const *section) const;
 const char *Get_Entry(char const *section,int index) const;
```

Is this function call in OnLoadGlobalINISettings() needed?:

```
 word_list.Delete_All();
```

If you want to make the 'check chars' functionality quicker you can create a bool array of 256, and setting the value of the array to 0 ('false') in the global INI, using memcpy(). Then read in the InvalidChars string from the INI and loop over every character of the InvalidChars string, convert each character to int and use this int as index in the bool array, to set the index to 'true'. Then in the CheckChars() function simply loop over every character of the player name, convert the character to int and use the int as index into the global bool array, if the bool at that index is set to 'true' then the player name contains an invalid char.

Something like this:

```
bool CharacterInvalidArray[256];

void CheckPlayerName::OnLoadGlobalINISettings(INIClass *SSGMIni)
{
 memcpy((char*)&CharacterInvalidArray, 0x0, 256 * sizeof(bool))

 StringClass InvalidCharsString;
 SSGMIni->Get_String(InvalidCharsString, "InvalidChars", "Invalid", "");
 for (int i = 0; i < InvalidCharsString.Get_Length(); i++)
 {
  int Index = (int)InvalidCharsString[i];
  CharacterInvalidArray[Index] = true;
 }
}

bool checkChars(WideStringClass name, int ID)
{
 StringClass SimpleName = StringClass(name);
 bool change_name = false;
 for (int i = 0; i < Simplename.Get_Length(); i++)
 {
  int Index = (int)SimpleName[i];
  if ( CharacterInvalidArray[Index ]== true )
  {
   SimpleName[i] = '_';
   change_name = true;
  }
 }

 if (change_name)
 {
  Find_Player(ID)->Set_Name(SimpleName);
 }

 return(change_name);

}
```

Note that you can justserveers with Chinese characters (hence why PlayerName is WideStringClass instead of just StringClass) so you should also check if the playername only contains ANSI characters when joining.

You can get the WideStringClass player name on join and use WideStringClass::Is_ANSI() to check if there are non-ANSI/ASCII characters. The definition for Is_ANSI() is

```
bool WideStringClass::Is_ANSI()
{
 if (m_Buffer)
 {
  for (int i = 0;m_Buffer[i] != 0;i++)
  {
   unsigned short value = m_Buffer[i];
   if (value > 255)
   {
    return false;
   }
  }
 }
 return true;
}
```

You can create your own function which will replace non-ANSI characters, variable 'value' > 255 in the above function, with underscores or something.

What Dragonade does (in the connection request event)is:

```
 for (int i = 0;i < Request.clientName.Get_Length();i++) {
  if (Request.clientName[i] < 33 || Request.clientName[i] > 126) {
   Request.clientName.RemoveSubstring(i,1);
   i--;
  }
```

Where Request.clientName is WideStringClass.