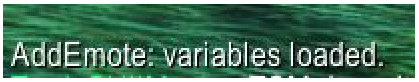
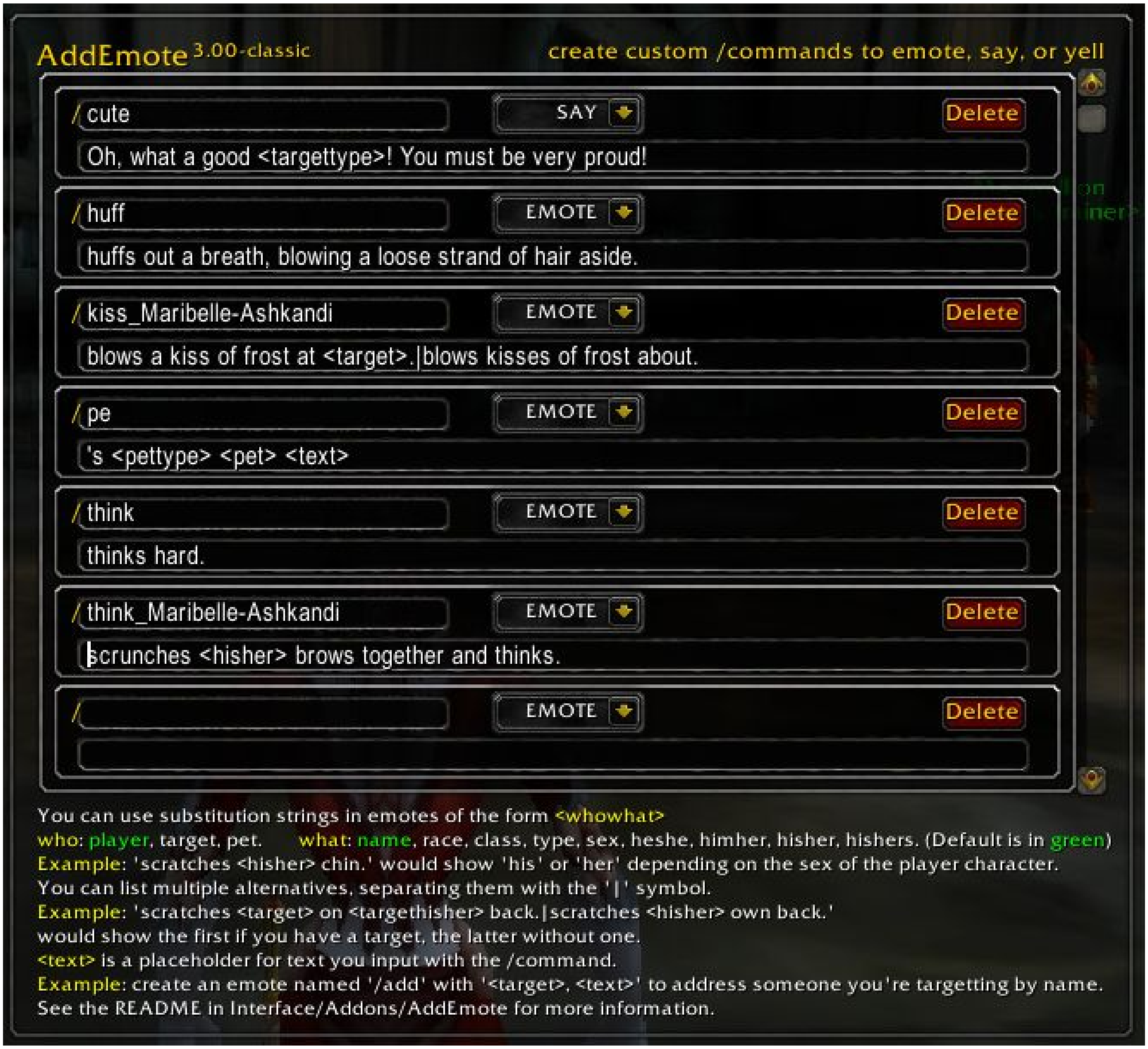
AddEmote is an addon that helps you to create additional slash-command emotes. These work like the ones that are built into World of Warcraft, and you can use them as shortcuts to make your character emote, say, and yell things.

If you've installed AddEmote properly, when you log in, you'll see a message like this in your chat box:



AddEmote: variables loaded.

You configure AddEmote via the WoW Interface Options panel. You can get to it quickly by using the command "/addemote", which will open Interface Options to the AddEmote control panel:



AddEmote 3.00-classic

create custom /commands to emote, say, or yell

/cute — SAY
Oh, what a good <targettype>! You must be very proud!

/huff — EMOTE
huffs out a breath, blowing a loose strand of hair aside.

/kiss_Maribelle-Ashkandi — EMOTE
blows a kiss of frost at <target>.|blows kisses of frost about.

/pe — EMOTE
's <pettype> <pet> <text>

/think — EMOTE
thinks hard.

/think_Maribelle-Ashkandi — EMOTE
scrunches <hisher> brows together and thinks.

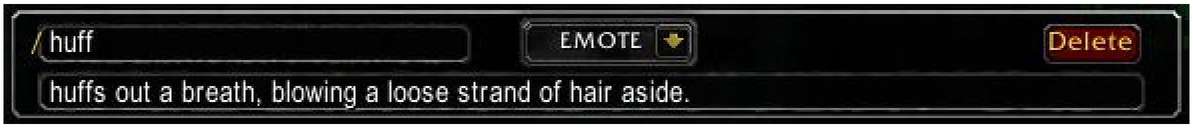
/ — EMOTE

You can use substitution strings in emotes of the form <whowhat>
who: player, target, pet. what: name, race, class, type, sex, heshe, himher, hisher, hishers. (Default is in green)
Example: 'scratches <hisher> chin.' would show 'his' or 'her' depending on the sex of the player character.
You can list multiple alternatives, separating them with the '|' symbol.
Example: 'scratches <target> on <targethisher> back.|scratches <hisher> own back.'
would show the first if you have a target, the latter without one.
<text> is a placeholder for text you input with the /command.
Example: create an emote named '/add' with '<target>, <text>' to address someone you're targetting by name.
See the README in Interface/Addons/AddEmote for more information.

This list is where you can make new slash-commands, update existing ones, and delete ones you don't want any more. Let's look at how you do these things in more detail.

Making New Commands

Let’s look at the box describing one of those commands:

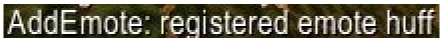


huff

huffs out a breath, blowing a loose strand of hair aside.

The box on the upper left is where you fill in the command you’ll use. The “/” is shown to remind you that you’ll type it when you issue the command -- you don’t type it when you’re creating commands here. The box below it is where you enter the text that will be used for the emote, say, or yell. It’s treated as it would be if you used the /me, /s, or /y command in WoW.

You can use “tab” to move between the boxes. “Enter” will move from one to the other as well, if they are not both filled in. If they are both filled in, hitting “Enter” in either box will register your new emote. When the emote is registered, you’ll get a message in the Chat Box, like this:



AddEmote: registered emote huff

The list of emotes will also rearrange itself, with the new emote being placed in alphabetical order with the others.

Updating Commands

If you decide you want to change what one of the commands does, you can simply edit the text that will be used for it, change the EMOTE/SAY/YELL option if you wish, and then press “Enter” from either text box, just as if you were making it for the first time. You’ll see the confirmation that the emote was registered in the chat box, just as for a new emote.

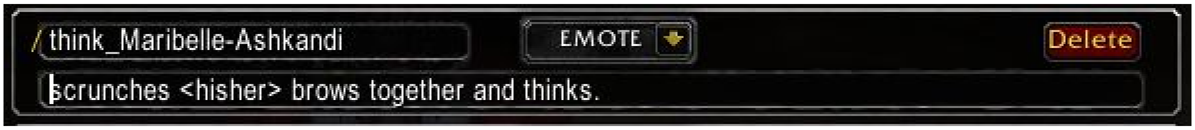
Deleting Commands

To delete a command, press the “Delete” button for that command. You’ll see a message in the chat box, either saying that the emote was unregistered, or that it was not unregistered because another emote with the same name still remains. Look at the **Character-Specific Emotes** section for how that can happen.

Advanced Usage

Character-Specific Emotes

In the example view, there was an emote shown:



think_Maribelle-Ashkandi

scrunches <hisher> brows together and thinks.

This is an example of a *character-specific* emote. If you create an emote with an underscore (_) in the slash command name, AddEmote will use the portion before the underscore as the command for the emote, and the part after as the character the emote applies to. So, for this one, if I do “/think” while playing as Maribelle-Ashkandi (WoW’s way of representing “Maribelle on the Ashkandi realm”), the given text will be used instead of the “/think” for other characters. You can also just give the name, instead of the name and realm, and it will work with any character with that name -- useful if you like to create the same character on multiple realms.

Substitution Strings

You may have noticed the **<hisher>** in the previous example. That's a *substitution string*. Those are provided by AddEmote (well, technically by ChatSubs, which is another addon I made before, and AddEmote includes a built-in copy). When these are used in chat, ChatSubs expands them for you, in this case changing "<hisher>" to "his" for a male character, and "her" for a female character.

The in-game interface gives a little information about the available substitution strings. Here's a more full tutorial on what's available:

A full substitution string is of the form **<whowhat>**. The **who** portion indicates a **unit**. The units you can use are

player, **pet**, **target**

and, in retail only right now, **focus**. The **what** portion indicates what information you want to display about the unit in question. This can be:

name, **heshe**, **himher**, **hisher**, **hishers**, **sex**, **type**
boygirl, **manwoman**, **ladlass**, **mastermistress**, **lordlady**, **sirmadam**

Here's what each does:

name: Gives the unit's name, as reported by the game.

heshe, himher, hisher, hishers: Gives the appropriate option, depending on the sex the game reports for the unit. When the game reports no sex for the unit, gives "it" or "its".

sex: Shows as "male", "female", or "person", as the game indicates. Note that, according to the game, pets are normally the same sex as their owner.

type: Gives the type of a creature. For warlock and hunter pets, this gets fairly detailed, and it's most useful for **<pettype>**, giving reasonable outputs like "bear", "cat", and "imp" with them. For creatures that are not pets, it will usually be one of the broad types the game recognizes, such as "aberration", "dragonkin", and "humanoid".

The second line gives the appropriate alternative depending on the character's sex, as reported by the game. Reasonable defaults are given when sex is not reported. (To be specific, "child", "person", "boss",

The above are all fairly reliable. The following are available, but are generally only useful for other players:

class, **race**

class: Gives the unit's class. Note that WoW is very inconsistent about how it assigns classes to NPCs, so it may not be what you expect if used on an NPC. It may say "unknown class" when used on some NPCs and creatures.

race: Gives the race for players. Unfortunately, WoW doesn't give any way to get the "real" race of NPCs, so for them, it falls back to the type, and most NPCs who you would expect to be one of the player races will be "humanoid".

The default for **who** is "player", so <name> will give your character's name, <heshe> will be based on your character's sex, and so on. The default for **what** is "name", so <target> will give your target's name, and <pet> your pet's name.

These substitution strings are available for all chat, so you can even use them with normal /s, /me, and so on.

Capitalization

If you capitalize the first letter of a substitution string, then the output will also be capitalized. Names are always capitalized by default.

The <text> Substitution String

Lastly, there's a special substitution string, **<text>**. It gets replaced by any text you type after the /command. This allows you to create commands you can add "extra flavor" into on the fly, like so:

waves <text>.

If this were assigned to "/wave", and you did "/wave a red flag", then others would see something like:

Maribelle waves a red flag.

You can combine this with other substitutions, though. For example:

waves <text> at <target>.

so "/wave happily" when Maribelle has Finch targeted would send out "Maribelle waves happily at Finch." If you leave off any text, the space before a <text> string will also be omitted, so the first without any additional text given will do "waves.", and the second would do "waves at Name."

Alternatives

Some of the built-in emotes in WoW do different things, depending on whether you have a target when you use them or you don't. For example, the default /kiss emote sends out either "You blow a kiss to <target>." or "You blow a kiss into the wind."

You can make your AddEmote commands do this as well, by giving alternatives. You separate each alternative with the pipe symbol (|, on the same key as \ on US keyboards). So, in AddEmote's system, the default /kiss emote would be:

blows a kiss into the wind.|blows a kiss to <target>.

When you use substitution strings with alternatives, AddEmote checks which units each alternative refers to. If a unit isn't available (e.g, it has <target>, but you don't have a target) that alternative is eliminated. If all the units are available, then that alternative gets a score equal to the number of units it uses. When it's done, the alternative with the highest score "wins". So in this example, if you have a target, "blows a kiss to <target>." will win, and if you do not, it will be eliminated, leaving "blows a kiss into the wind." as the only remaining alternative.

If there are multiple alternatives left with the same score, AddEmote will choose one of them at random. So, you can create emotes that randomly cycle among alternatives. Note that this cycling is fully random -- it has no preference to avoid an emote it just did. Thus, spamming the same emote likely won't produce desirable results.

If there are *no* alternatives left, then AddEmote will still try to output something. In that case, it will add all the alternatives back into the pool, randomly select one, and hope for the best. This is unlikely to give good output, so I highly recommend giving each command some reasonable fallback that doesn't use any substitution strings except ones for your character.

Upgrading from Previous Versions

If you have an earlier version of AddEmote installed, your global emotes will automatically be migrated to the new format. Because of the way World of Warcraft stores character-specific data, however, your character-specific emotes will be added when you log in to the character they belong to. There is no way for AddEmote to access them if you are not logged into that character.

They'll be added with "_charactername-realm" appended, as described above. If you have multiple characters with the same name on different realms, and were creating the same character-specific emotes for them, you might want to edit one to have just the character name, and then remove the multiple versions. I hope you'll find the new system to be easier to use!

Transferring Emotes

To copy your created emotes to a different computer or installation (say, from retail to classic), do the following:

1. Navigate to the World of Warcraft folder. On a Windows system, this will normally be C:\Program Files(x86)\World of Warcraft\
2. Go into the _retail_ or _classic_ folder, whichever has the already-created emotes.
3. Go into the WTF folder, then continue down into Account, the name of the account, and SavedVariables.
4. Get the AddEmote.lua file in that location.

You'll copy that to the equivalent location under the other installation or on the other computer. Then, when you log into WoW on it, you'll have your saved emotes from the other installation/system!