



Training Manual

Chatbot, Greeter One, S-Bot

This is the Training Manual for the Chatbot, Greeter One, and S-Bot products from Metaverse Technology. These bots are used in Second Life to communicate with avatars autonomously, and more information can be found at www.metaversetech.com .

Congratulations on your choice of a bot from Metaverse Technology! If you have not done so yet, please refer to the Quick Start Guide to get your bot up and running quickly.

Each and every bot has enormous customization potential, and we will describe here how to train and customize your bot. Please refer to the User Guide to learn how to operate your bot.

TABLE OF CONTENTS

AIML	4
Metaverse Tech AIML.....	4
ALICE AIML.....	4
How to Write your Own AIML	5
Basic structure.....	5
Recursion	6
Wildcards * and <star/>	7
Random	9
That	10
Topic	12
Gaitobot.....	15
Keywords	20
LL:KEYWORDS	20
LLS:KEYWORDS.....	21
Pandorabots Resources	23
Bad Answer.....	23
Pandorabot Trainer.....	26
FAQ to AIML	28
Linked Messages.....	29

AIML

Metaverse Tech AIML

Metaverse Technology is happy to provide several specialized AIML files for use in Second Life. Please refer to the User Guide for instructions on installing these files.

secondlife.aiml	This file should be included in all bots operating within Second Life. Not only does it contain SL specific responses to questions about SL, it contains responses that utilize many of the first specialized keywords available.
star_FB.AIML	These three files are the first implementation of emotional reponse keywords for the FaceBot modules. They are only helpful for Greeter One bots that have the additional FaceBot body, or the S-Bot model which automatically includes FaceBot functionality.
under_FB.AIML	
W_FB.AIML	
X_MT.aiml	This file is very important to the S-Bot model. It includes the Google Define Integration that is required for the S-Bot to automatically query Google when it does not have an answer to a “What is ...” question.

They contain AIML that has been customized both for content, and for special features discussed in the Keyword section of this documents. Please feel free to modify these files once you have then installed in your bot. Information on how to write your own AIML is covered later in this document and would be useful in adapting these files.

ALICE AIML

The best part about creating these bots on Pandorabots.com is that a lot of work has been done already. Starting a bot with award winning knowledge is as simple as clicking on a couple of pages. We recommend using either Dr Wallace's A.L.I.C.E - March 2002, or Annotated A.L.I.C.E. AIML - September 2003 as your starting set. You can select the other options, but they will use an older set of knowledge. Two of them are for different languages.

The AIML files that come with the selections are based upon award winning bots, and you can copy the text and modify it if you are interested in adding addition information. There are also a number of AIML files available on

MetaverseTech.com that you can add by directly uploading them to your Pandorabots.com account. In the next section we will discuss customizing the files to suit your own specific needs.

How to Write your Own AIML

AIML is an XML type language. That means the information is surrounded by tags to tell the language interpreter how to process the text. Below is a description of commonly used tags for reference during this section. You can use a text editor or one of the other editors discussed later in this document.

Name	Opening tag	Closing Tag	Description
Category	<category>	</category>	These tags surround a complete Input and Response section
Pattern	<pattern>	</pattern>	These surround the Input to be matched
Template	<template>	</template>	These surround the Response section.
Think	<think>	</think>	These are used to surround a portion of the Response section to notify the interpreter that the enclosed information is processing data. It will not be shown to the user.
Srai (recursion)	<srai>	</srai>	This surrounds a portion of the response section, and is used to tell the interpreter that it should search for the enclosed somewhere else in its knowledge.
Random	<random>	</random>	Used with and to give a random response from a list of responses.
That	<that>	</that>	Used to match what the bot said previously.
Star	N/A	<star/>	This is used to substitute information in the Response that was picked up by the * in the input.
Person	N/A	<person/>	This is used to swap pronouns picked up by a * in the input and substitute them into the Response

Now that we have defined some of the many tags used in AIML, let's take a look at how to use them.

Basic structure

The most basic structure of a complete Input – Response AIML entry is a direct match. With this method, the Input is matched exactly, and the response is given exactly. For example:

Avatar: What can you do?

SBot: I can do anything since I have advanced intelligence.

Would be written in AIML as:

```
<category>
  <pattern>WHAT CAN YOU DO</pattern>
  <template> I can do anything since I have advanced
intelligence.</template>
</category>
```

The entire section is contained between the <category> and </category> tags. The text between <pattern> and </pattern> is what the interpreter will attempt to match when someone talks to you bot. In this case it must match exactly. It is important to note that there is no punctuation, and that it is entered in capital letters. The text between <template> and </template> is what the bot will say.

So why is there no punctuation, and why is it in capital letters? The interpreter strips all punctuation from an input, and looks for matches only on wording. It also does not care whether the input or the pattern is uppercase or lowercase, so “WhAt CaN yoU Do” would also be matched with this entry. One more thing to mention since we are on this topic, is that the interpreter will expand contractions so “I’m” is converted to “I AM” and “don’t” is converted to “do not” and so on.

Recursion

One very powerful thing in AIML is the ability to perform recursion. This can be used for several reasons, and is a way to return appropriate responses for more inputs. It can be used for many reasons, but here are a few common ones:

- Synonyms
- Spelling corrections
- Reducing complex sentences
- Getting responses to an input from multiple entries that occur elsewhere
- Much more...

So what if someone was to ask something similar to “What can you do?” but not exactly? For instance, we would like to return the same response for the following:

```
Avatar: Can you do anything?
SBot: I can do anything since I have advanced intelligence.
```

You could create a category to match this as described above, but let’s use recursion for this. It may seem unnecessary to use this for this purpose, but later we will see how powerful it can be.

```
<category>
  <pattern> Can you do anything </pattern>
  <template> <srai> what can you do</srai></template>
</category>
```

When the interpreter sees “Can you do anything?” it will search for the response associated with “what can you do” and will say the response associated with that entry. Note that the SRAI section is contained completely within the template. You can also put text outside the SRAI tags and it will be included in the response as in this example:

```
<category>
  <pattern> Can you do anything </pattern>
  <template> Of course, <srai> what can you
do</srai></template>
</category>
```

This would return “Of course, I can do anything since I have advanced intelligence.” The power of recursion is used very often with another powerful part of AIML; * and <star/>.

Wildcards * and <star/>

We have seen how to create responses by matching exactly what the user may enter when talking with our bot. Creating entries for every response a person could ask, would take a very long time if we were to use only that method. Even using recursion would not reduce the number of entries we would have to create, but luckily * and <star/> come our rescue.

* and <star/> are wildcards. * is used in the pattern to “pick up” a portion of the Input, and <star/> is used in the template to substitute that input into the Response portion of the template. This can be used to create a “copycat” entry.

```
Avater: Say I'm a little teapot
SBot: I am a little teapot.
```

(Note that the contraction is expanded from I’m to I am)

```
<category>
  <pattern> Say * </pattern>
  <template> <star/> </template>
</category>
```

The interpreter will match this response to any input that begins with “Say”. See how powerful this is? Then it will take everything in the Input after “Say” and put it in the response! When combined with the power of recursion, this can be used to reduce the inputs to catch more responses. Let’s say that you want your bot to have information about cars and boats. Most reduction methods that I’m about

to present are already written in the default code when you create your bot, but the following is a good example of using *, <star/>, and SRAI:

```
<category>
  <pattern> What is a * </pattern>
  <template> <srai><star/></srai> </template>
</category>
<category>
  <pattern> car </pattern>
  <template>
    A car is what humans use to drive around in.
    Robots drive cars too.  </template>
</category>
<category>
  <pattern> boat </pattern>
  <template>
    A boat is a waterborne vessel used to bear humans
    and their freight.
  </template>
</category>
```

In this example, a person could say “car” or “what is a car” and the same response would be returned. They could also say “boat” or “what is a boat”. With this example we have reduced what would normally be 4 entries to 3, and that is if your bot only knows about cars and boats, imagine how simple it would be to catch many entries for a bot that knows about a lot more!

Another wildcard is the underscore symbol “_”. This is used to capture input that comes before it. This next example shows how many inputs can be captured, and a somewhat meaningful response given with just 2 entries.

```
<category>
  <pattern>_ is nice </pattern>
  <template>
    I think <star/> is nice too.
  </template>
</category>
<category>
  <pattern> I don't like * </pattern>
  <template>
    Why don't you like <star/>.
  </template>
</category>
```

These two entries would capture a very large number of responses. “Mary is nice”, “Pie is nice”, and “Second Life is nice” would all get a response of “I think [Mary, pie, Second Life] is nice” depending on which one was said. Similarly, for the “I don’t like [subject]”.

Recursion can make expand this even more. Adding,

```
<category>
  <pattern>I think * </pattern>
  <template>
    <srai> <star/> </srai>
  </template>
</category>
```

would also yield a proper response to “I think [Mary, pie, Second Life] is nice”. In fact <srai> <star/> </srai> is used so often in AIML, it has a shortcut, <sr/> so the example above would be written:

```
<category>
  <pattern>I think * </pattern>
  <template>
    <sr/>
  </template>
</category>
```

In fact, you can use these methods to catch if an Input contains anything. See the following

```
<category>
  <pattern>_ tell me * </pattern>
  <template> <srai> <star index="2"> </srai></template>
</category>
```

This would catch, “SBot, tell me about a car.” Basically removing unnecessary parts from the Input. We will talk more about putting info inside the tags, later.

Random

Now that we have seen how to catch many responses, it might be useful if your bot did not say the same thing every time it matched an Input. This is where Random is helpful. Random responses can be very useful if you think the user might say the same thing, but you would like the bot to appear to have a bit more knowledge than just one line. For example, what if the person asks, “What do you know” or “What else do you know”? We would like the bot to say something different each time. So let’s see an example:

```

<category>
  <pattern>What * know</pattern>
  <template>I know about
    <random>
      <li>Linden Labs.</li>
      <li>prims.</li>
      <li>scripting.</li>
      <li>textures.</li>
    </random>
</category>

```

Now when someone asks the bot either of those questions, the bot will pick one of four answers:

- I know about Linden Labs.
- I know about prims.
- I know about scripting.
- I know about textures.

Then we can program the bot with information about each and if the person asks what else it knows about, it can bring up a new subject.

That

Now at this point, you might think, what if I want the bot to respond differently to the same Input? You also want a specific response, not just some random selection. The best example is what if your User says “Yes”? Well, if they are saying “Yes”, then your bot probably asked them a yes or no question. For the bot to know what they are saying “yes” to, then it needs to know what it said last.

Fortunately, there is a solution! Use <that> tags in your AIML. <that> tags are processed after the pattern, and so that is why it comes after the pattern tags. This is also why you should only use <that> if you have multiple entries for the same pattern. Let’s see an example. For simple coding, let’s assume that the bot has the code to ask the user two questions: “Are you an Aquarius?” or “Do you like pie?”. The interpreter automatically keeps track of the last response that it said, so the question entries could come from any of the code we’ve discussed before. The code to respond to these questions would be:

```

<category>
  <pattern>yes </pattern>
  <that>Do you like pie</that>
  <template>
    I like pie too.
  </template>
</category>
<category>
  <pattern> yes</pattern>
  <that>Are you an Aquarius</that>
  <template>
    I am also an Aquarius.
  </template>
</category>

```

Now when the user says “yes” to either of those questions, the bot knows how to answer. This can be useful if you want your bot to ask a series of question, or any other time when you want to have the bot respond differently to similar Inputs. However, you do not need to use <that> until you have entries with the same pattern.

Another good example is if our bot gave a random response. Let’s say that a person asked “What do you know”, and our bot gave one of its four responses. You can now write some useful entries if the person was to ask a general question “Tell me more”.

```

<category>
  <pattern>tell me more</pattern>
  <that> I know about Linden Labs</that>
  <template>
    Linden Labs are the creators of Second Life.
  </template>
</category>
<category>
  <pattern>tell me more</pattern>
  <that> I know about prims </that>
  <template>
    Prims are the base of all Second Life objects.
  </template>
</category>
<category>
  <pattern>tell me more</pattern>
  <that> I know about scripting </that>
  <template>
    Scripting defines the behavior of all objects in
    Second Life.
  </template>
</category>

```

```
<category>
  <pattern>tell me more</pattern>
  <that> I know about textures </that>
  <template>
    Textures make Second Life objects look good!
  </template>
</category>
```

When using <that>, you can also refer to an index, and you can also use *. This is a more advanced use, and is a bit beyond this documentation. You may need to go to www.alice.org to find documentation to help you with that.

Topic

Now it may seem a stretch that you would have multiple entries with the same pattern and the same THAT, but it can happen. In fact, if done right, you can use topic to help get more categories without having to write quite as many entries. How is that possible? In conversations we often say similar things, and ask similar questions. Even knowing what your bot said last may not be enough information to lead your user properly. We will also look at changing and getting information inside of tags in this section.

This next example may seem simple, but it is meant to demonstrate why Topic is important.

Conversation one

```
Avatar: I like cars.
Bot: What do you like about them?
Avatar: I think they are fun.
Bot: Cars are fun, indeed.
```

Conversation two

```
Avatar: I like boats.
Bot: What do you like about them?
Avatar: I think they are fun.
Bot: Boats are fun, but can be expensive.
```

Note how the avatar and bot say the same thing in both cases “What do you like about them” and “I think they are fun”. Let’s see how we handle this. Pay special attention to the Topic tag.

Note the placement of the think and set name tags which are inside the template tags.

```

<category>
  <pattern>I like *</pattern>
  <template>
    What do you like about them.
    <think><set name = "Topic"> <star/> </set>
  </think>
</template>
</category>

<topic name = "cars">
<category>
  <pattern>I think they are fun</pattern>
  <that>What do you like about them </that>
  <template>
    Cars are fun, indeed.
  </template>
</category>
</topic>

<topic name = "boats">
<category>
  <pattern>I think they are fun</pattern>
  <that>What do you like about them </that>
  <template>
    Boats are fun, but can be expensive.
  </template>
</category>
</topic>

```

Now we have been able to handle a pretty complicated conversation with just a few entries. One more thing to note is that the entries inside of the cars and boats Topics are almost exactly the same. The only difference is the Response inside of the template. Using this method you could create quite a few entries that are very similar, but have different responses based on the topic. Using a copy-paste approach, this would not take very much work.

Now take a close look at the first entry. We set the topic with the line

```
<think><set name = "Topic"> <star/> </set> </think>
```

Think tells the interpreter that it is about to perform some special stuff that it isn't going to send out to the user. The Set tag sets a variable in the program. You can also set other variables such as

(Note: we left out the rest of the entry for simplicity)

```
<set name = "color"> blue </set>
```

This can be recalled at a later time by using

```
<get name = "color">
```

Which can be inserted into a template Response. This is useful if the bot should remember something for later. Some of these variables are set when the bot is loaded, and these are “properties” of the bot. The default values for these can be set by clicking on the properties link for your Pandorabots bot.

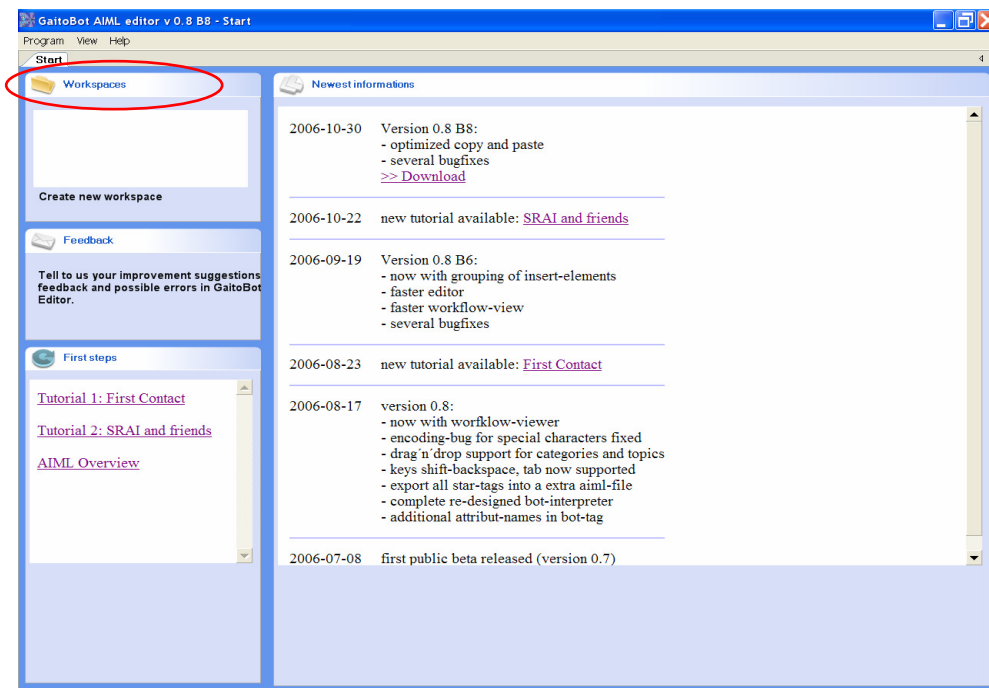
This is a basic description of a few of the things you can do with AIML. Much more information can be found on the web. www.alice.org and on Wikipedia you can find even more methods and examples. Pandorabots.com also has a list-serve that you can post questions to, and there are many forums and FAQs around the web.

Gaitobot

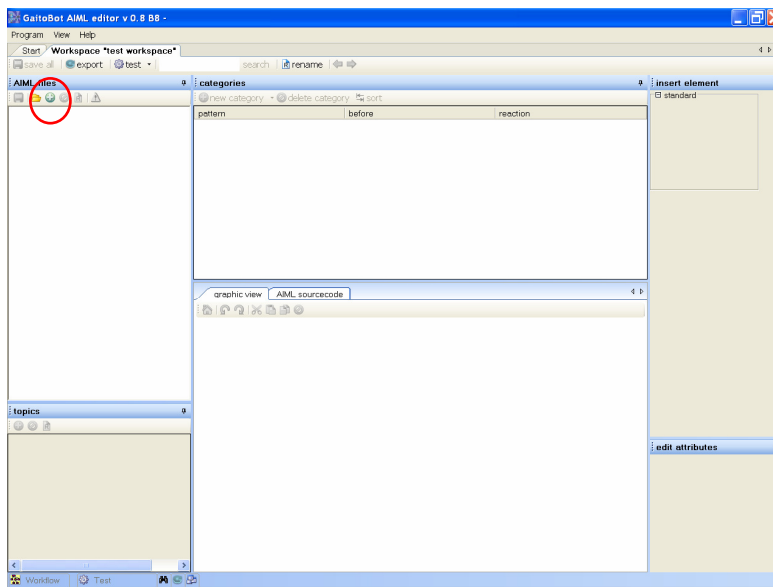
At the time of this writing, GaitoBot AIML editor is still in development. It is a very nice tool to help you create and organize you AIML categories with visual tags. It also has some nice layout features to look at how your categories forward to each other. This section will show how to use the GaitoBot editor, and a few important things you must do for it to work properly with Pandorabots.com. GaitoBot editor also has some video tutorials for using their program. See http://www.gaito.de/content/produkte_gaitobot/editor.asp?language=en for their downloads and tutorials.

This section assumes that you know a bit about AIML. If you are not sure, or need a refresher, please see the AIML section for a more. You must also download and install the editor from their website. Please follow the link given above, unless you know German, because it may be difficult if you try to find the editor from their main page.

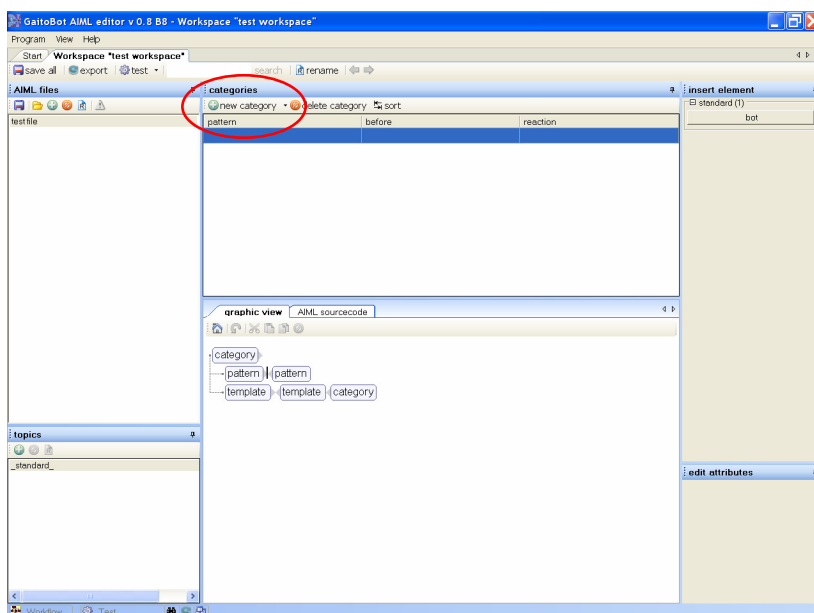
When you first load the editor you will see a start screen as shown below. This is the initial screen, and it contains some news and information from the Gaito website. To get started, Click “Create new workspace” and give your workspace a name like “test workspace”.



Now that we have a new workspace, you should see a screen as shown below:

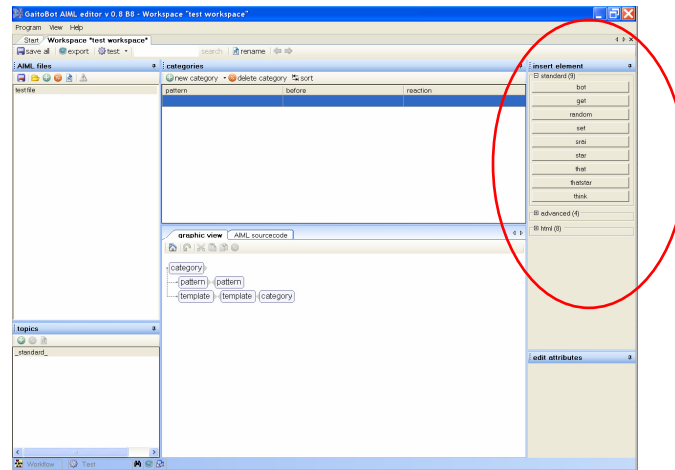


Let's create a new AIML file so click the green + in the upper left corner, and give it a name like "test file". This creates the file where all your categories will be stored, and shows a default topic of _standard_. Click on the green + near the upper middle of the screen next to "New Category" to create the first entry. You must have _standard_ (in the lower left) selected to do this. Now your screen should look like this:



Note that it automatically creates the beginning and ending tags for category, pattern, and template. This helps prevent you from forgetting a closing tag that could mess up your AIML file. If you click or use the arrow keys to reposition the cursor, the elements to the right (shown as bot in the Figure above) will change.

They change based on the location that they are allowed to be placed in. For example, if you put the cursor between the template tags you will see:



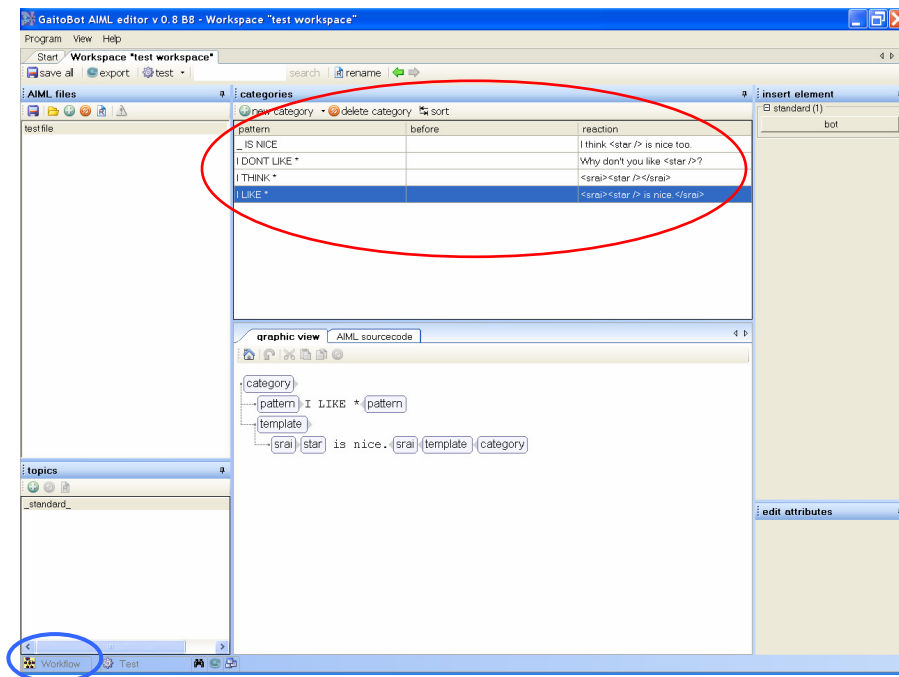
Clicking on these will automatically insert tags for those elements into that section. So let's create a couple of entries. Create entries for the categories below by entering text between the tags and clicking to add elements.

```
<category>
  <pattern>_ is nice </pattern>
  <template>
    I think <star/> is nice too.
  </template>
</category>
<category>
  <pattern> I don't like * </pattern>
  <template>
    Why don't you like <star/>.
  </template>
</category>

<category>
  <pattern>I think * </pattern>
  <template>
    <srai> <star/> </srai>
  </template>
</category>

<category>
  <pattern>I like * </pattern>
  <template>
    <srai> <star/> is nice.</srai>
  </template>
</category>
```

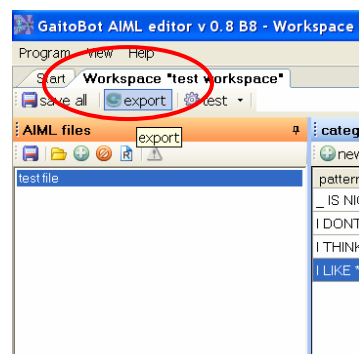
The screen should now look like:



One of the entries we created made a recursive call back to “_ is nice”. An interesting feature of the GaitoBot editor is Workflow. Select the entry “I like *” and then click Workflow in the lower left corner. This will show you that “I like *” forwards to “_ is nice”.

One thing to note at the time of writing this is, inserting “that” tags is a little tricky. Clicking between pattern and template does not work. To get to that point, you must click inside of pattern, and then use the right arrow key to move the cursor between pattern and template. At that point the “that” element will be shown in the right sidebar.

Now that we have a bit of AIML, let’s export it to be used with Pandorabots.com. Click on the AIML file in the left sidebar and click the Export button just above the sidebar.



This will bring a bar up from the bottom. You can change the directory that the export will send to by typing in the text field, or by clicking the browse button. To export the file, click the Start Export button. If the bar disappears while you are browsing for a directory, hold your mouse over the green circle near the bottom of the screen, this will bring the bar back.



Once the AIML is exported, you can upload the file to your bot at pandorabots.com. When it is uploaded, check the beginning part of the AIML file for a comment that says the version is 1.0.1. if you see something like this, change the value to 1.0 or pandorabots will not accept it when you publish your bot. Currently we are talking with the GaitoBot editor creators on creating an option to change this.

Also, if you decide to import code that you have written in notepad or some other text editor into the GaitoBot editor, there is one thing that you must be aware of. At the time of this writing, the editor does not handle extra whitespace (tabs and spaces) that is often put into AIML to make it more readable in standard text editors. If you want to import your file, you will need to remove all extra white space and line breaks. We are also talking with the creators of this program to see if this can be changed.

There are some other useful features of the editor that you can try. GaitoBot editor has a test function, so you can try out your newly created code, and it will show you what categories it picks up to get its response. There are also some “search” and “most used” recursion functions to help you edit you AIML once it gets big. Adding new topics is also easy since you can just click the button and give the topic a name.

Keywords

LL:KEYWORDS

These are the original keywords created in the first edition of the chatbots series. They are present in all three bots; Chatbot, Greeter One, S-Bot. The keywords are included into the AIML, but are parsed out by your bot. So in this way the bot can insert information that it gets directly from Second Life.

The following is a list of the keywords currently included in all releases.

LL:OWNER	A reference to the current owner of the bot.
LL:BOTNAME	The object name that you have given to your bot.
LL:TALKER	The current avatar your bot is conversing with at this time.
LL:LOCATION	The region that your bot is currently located.
LL:LANDOWNER	The current owner of the land that the bot is on. This updates upon rez, and after it is called.

These keywords can be put anywhere in the template section of the AIML. They will not be put out as chat by the bot, rather they will be replaced with the information that they represent.

The following is an example that is included in the secondlife.aiml file from MetaverseTech. Please refer to the user manual for directions on installing this AIML into you bot.

```
<category>
<pattern>WHERE ARE YOU</pattern>
<template>I am in LL:LANDOWNER's place in LL:LOCATION.
<think><set name="place"><set name="topic">Second
Life</set></set></think></template>
</category>
```

In this example, if the bot is on Bob's land in Big Mushamush, and you ask the bot, "Where are you?" It will respond with, "I am in Bob's place in Big Mushamush." Feel free to build your own AIML with these keywords.

LLS:KEYWORDS

We have expanded the functionality of the Greeter One and S-Bot series by adding user defined keywords! Now you have unlimited power to fully explore the benefits that Second Life has to offer AI development.

It is very simple to implement your infinite number of user defined keywords. Simply add LLS:YOURKEYWORD to the end of any AIML response while you are training your bot on Pandora.

NOTE: Be careful to note the difference between these keywords, which are preceded with LLS:, as opposed to the predefined keywords, which are preceded by LL:.

In Second Life our program will parse out LLS: and anything that comes after it in a response from the bot. This way the bot can continue a normal conversation without your keywords being sent in chat to the talker. At the same time our program will send a linked message out with a num = 2 and str = LLS:YOURKEYWORD.

Example: `llMessageLinked(LINK_SET, 2, "LLS:YOURKEYWORD", NULL_KEY);`

So then any module or script that you add to the greeter can perform functions and actions based on your keywords!

To illustrate the functionality of this new feature we have developed a series of LLS:KEYWORDS that express different capabilities. Except for the emotion keywords that work with the FaceBot add on to the Greeter One product, these examples are only programmed into the S-Bot. Feel free to create scripts for the Greeter One that take advantage of the LLS: keywords. The following is a list of examples.

LLS:IMAGE[<i>TEXTURE</i>]	For use with the display that comes with the S-Bot. Will display the <i>TEXTURE</i> if it is in the display inventory.
LLS:GIVE[<i>ITEM</i>]	Will give the <i>ITEM</i> to the current talker if the <i>ITEM</i> is in inventory.
LLS:URL[<i>http://www.yoursite.com</i>]	Will ask the current talker if they would like to be directed to the specified website.
LLS:MAP[<i>Region,x,y,z</i>]	For use with the display that comes with the S-Bot. Will make the display a teleport, where the talker can then right click and TP to specified location.
LLS:ANGRY, LLS:HAPPY, LLS:THINKING, LLS:SAD, LLS:SURPRISED	Emotional keywords that can be used in the S-Bot or Greeter One to display different facial expressions.

The FaceBot has a very active use of the LLS:KEYWORDS. There should be an open permission script in your bot called “Face.” This script takes advantage of the emotion LLS: indicators, and displays different facial expression textures on your bot. You can get AIML where we have started embedding these keywords at MetaverseTech.com. We offer additional texture options, such as a Male version, for free from SLXCHANGE or at our store in Big Mushamush.

As an example, if you include the following code in the linked message receiver of a script, it will attempt to show the “F Happy” texture.

```
if(llSubStringIndex(str,"LLS:HAPPY") > -1)
{
    llSetTexture("F Happy", 2);
    llSetTexture("F Happy", 4);
}
```

Pandorabots Resources

Bad Answer

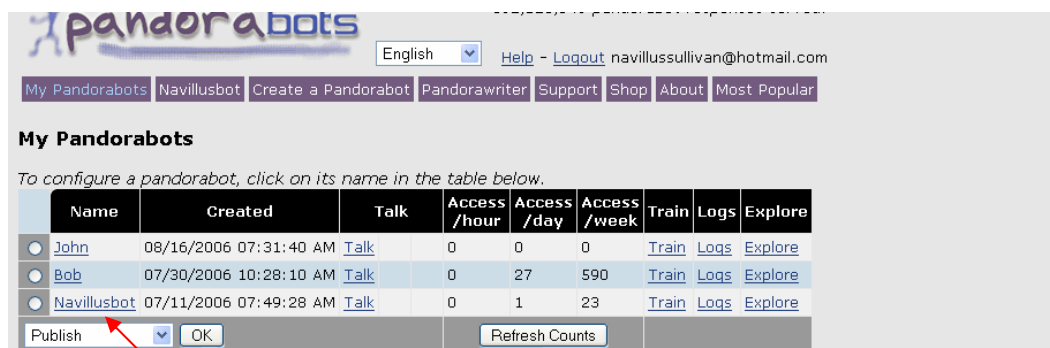
The greatest thing about the AIML based chatbot is the ease at which it can be specialized. Our implementation supports direct training through the bad answer protocol. This works from within Second Life, if you add the bad answer aiml to your bot personality on Pandorabots.com.

Download the bad answer protocol from

<http://www.alicebot.org/aiml/aaa/Badanswer.aiml>.

Right click and select Save As ... and save it as “badanswer.aiml”.

Next, upload the file to the Pandorabots site. To do this, sign into Pandorabots.com with the account you created in the Quick Start Guide.



The screenshot shows the 'My Pandorabots' section of the Pandorabots website. It features a table with columns: Name, Created, Talk, Access /hour, Access /day, Access /week, Train, Logs, and Explore. Three bots are listed: John, Bob, and Navillusbot. Below the table is a 'Publish' button with a dropdown arrow, an 'OK' button, and a 'Refresh Counts' button. A red arrow points to the 'Publish' button.

Name	Created	Talk	Access /hour	Access /day	Access /week	Train	Logs	Explore
John	08/16/2006 07:31:40 AM	Talk	0	0	0	Train	Logs	Explore
Bob	07/30/2006 10:28:10 AM	Talk	0	27	590	Train	Logs	Explore
Navillusbot	07/11/2006 07:49:28 AM	Talk	0	1	23	Train	Logs	Explore

Select the name of your bot, or at least the one you want to work with now.



The screenshot shows the 'Navillusbot' page on Pandorabots.com. The navigation menu includes links for Train, Properties, Configure, AIML, Custom HTML, AOL IM, Oddcast VHost, HostABot, Logs, Explore, and Subscribers. A red arrow points to the 'AIML' link.

Navillusbot

This pandorobot is published at:

- <http://www.pandorabots.com/pandora/talk?botid=c7e009881e366da8>

Want to make your pandorobot stand out from the crowd? Try one of the techniques below:

- Add an animated speaking character with an [Oddcast VHost](#) face.
- Or try out an alternative Flash-based character from [HostABot](#).

Select the AIML link.

Depending on which configuration you chose in the Quick Start Guide, you should now see page filled with a list of AIML files. At the bottom of the page you can upload files.

star.aiml	0	99,905	10/04/2006 07:17:59 PM	Browse	Download	(local)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
under.aiml	0	20,346	10/04/2006 07:17:59 PM	Browse	Download	(local)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
w.aiml	0	1,431,188	10/04/2006 07:17:59 PM	Browse	Download	(local)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
update.aiml	0	2,708	10/04/2006 07:32:57 PM	Browse	Download	(local)	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Total number of files: 41

[Submit Changes](#)

To add a new AIML file to your Pandorobot, you can either [create a new AIML file](#) from scratch using your browser as the editor, or you can create a file on your local machine and use the form below to upload it to Pandorabots.

Enter the names of the AIML files you wish to upload in the form below. You can use the **Browse...** button to locate the files on your local machine. If you don't see a **Browse..** button your browser may not support file uploads. Each file must either be an AIML file or a free-format text dialog file to be converted to AIML using [Pandorawriter](#). To convert a file using Pandorawriter you should check the appropriate **convert text dialog to AIML** check box.

File 1: [Browse...](#) ☐ convert text dialog to AIML

File 2: [Browse...](#) ☐ convert text dialog to AIML

File 3: [Browse...](#) ☐ convert text dialog to AIML

File 4: [Browse...](#) ☐ convert text dialog to AIML

File 5: [Browse...](#) ☐ convert text dialog to AIML

[Upload AIML File\(s\)](#)

[Home](#) - [Pandorabots Mailing Lists](#) - [Pandorabots Documentation](#) - [Pandorabots FAQ](#)
[The A.L.I.C.E. AI Foundation](#) - [A.L.I.C.E. Mailing Lists](#) - [AIML Documentation](#) - [AIML at a Glance](#) - [AIML 1.0.1 Draft Specification](#)
[The AIML Scripting Resource](#) - [The Chatterbot Collection](#) - [AI Nexus Forum](#) - [Botworld](#) - [DigitalGirl](#) - [Botmasters' Frappi!](#) - [Virtual Humans Forum](#)
[Talk to A.L.I.C.E.](#) - [Talk to Lauren](#) - [Oddcast SitePal](#)

Use the Browse buttons to find the file that you have downloaded to your computer.

Once you have them selected, hit the Upload AIML File.

If your file was uploaded, it will now appear in lower case at the bottom of the list of AIML files.

Once you have installed this AIML you can train the bot from Second Life, by responding “Bad Answer” or “wrong” whenever you are unhappy with what the bot has said. You must be careful to format your responses carefully as shown next or you may not get the change you want.

For example:

YOU: Hello bot.

BOT: Hello, nice to meet you.

YOU: What is SL?

BOT: I don't know, maybe I should Google it.

YOU: Bad Answer.

BOT: Would you like to teach me a new answer to " What is SL?"? Yes or No?

YOU: yes.

BOT: OK, what should I have said?

YOU: SL is the coolest place in the Metaverse.

BOT: " SL is the coolest place in the Metaverse"? Does this depend on me having just said, "Hello, nice to meet you"?

YOU: no

BOT: I'll try to remember to respond, "SL is the coolest place in the Metaverse." when you say, "What is SL?"

So next time ...

YOU: What is SL?

BOT: SL is the coolest place in the Metaverse

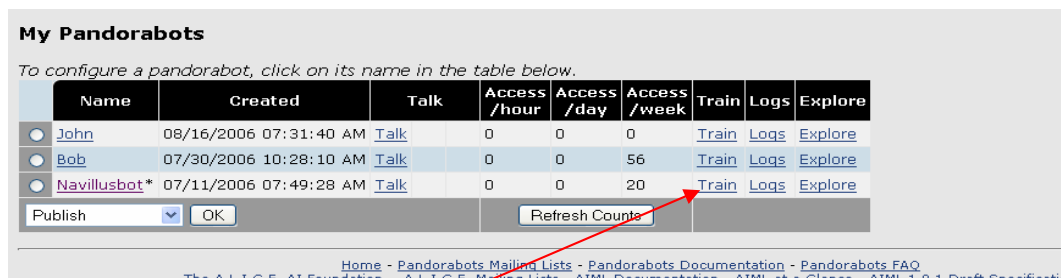
You must be in a current conversation with the bot, for the bot to respond. It is important to note that anyone can use this feature regardless of ownership. So make sure you consider that anyone can change your bot's knowledge if you implement this.

More information on the Bad Answer AIML can be found at <http://www.alicebot.org/aiml/aaa/Badanswer.txt>.

Pandorobot Trainer

Training directly from the Pandorabots website has many powerful advantages. There are two primary methods for accessing this process, through the Train Interface, and through the Logs.

First log into your Pandorabots.com account.



My Pandorabots

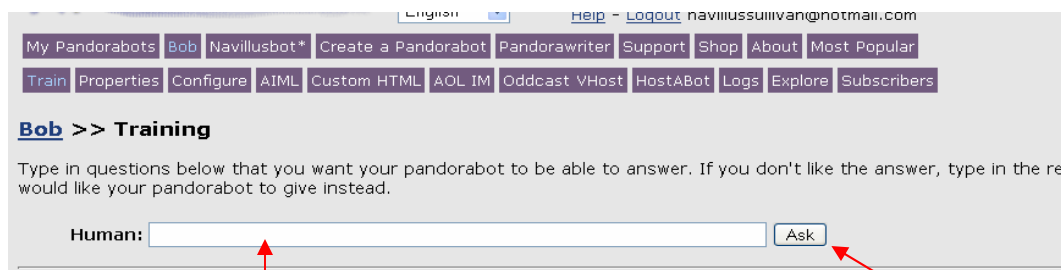
To configure a pandorobot, click on its name in the table below.

Name	Created	Talk	Access /hour	Access /day	Access /week	Train	Logs	Explore
John	08/16/2006 07:31:40 AM	Talk	0	0	0	Train	Logs	Explore
Bob	07/30/2006 10:28:10 AM	Talk	0	0	56	Train	Logs	Explore
Navillusbot*	07/11/2006 07:49:28 AM	Talk	0	0	20	Train	Logs	Explore

Publish

[Home](#) - [Pandorabots Mailing Lists](#) - [Pandorabots Documentation](#) - [Pandorabots FAQ](#)
The A.I.L.C.E. AI Foundation - A.I.L.C.E. Mailing Lists - AIML Documentation - AIML at a Glance - AIML 1.0.1 Draft Specification

Select the “Train” option on the row of the bot that you want to work with at this time.



My Pandorabots [Bob](#) [Navillusbot*](#) [Create a Pandorobot](#) [Pandorawriter](#) [Support](#) [Shop](#) [About](#) [Most Popular](#)

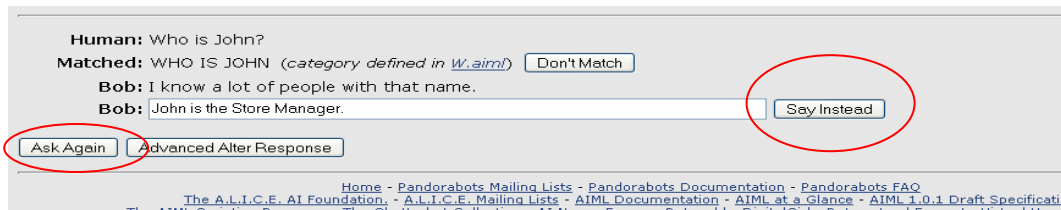
[Train](#) [Properties](#) [Configure](#) [AIML](#) [Custom HTML](#) [AOL IM](#) [Oddcast VHost](#) [HostABot](#) [Logs](#) [Explore](#) [Subscribers](#)

Bob >> Training

Type in questions below that you want your pandorobot to be able to answer. If you don't like the answer, type in the response you would like your pandorobot to give instead.

Human:

Enter your statement or question in the text box, and select “Ask”.



Human: Who is John?

Matched: WHO IS JOHN (category defined in [W.aiml](#))

Bob: I know a lot of people with that name.

Bob: John is the Store Manager.

[Home](#) - [Pandorabots Mailing Lists](#) - [Pandorabots Documentation](#) - [Pandorabots FAQ](#)
The A.I.L.C.E. AI Foundation - A.I.L.C.E. Mailing Lists - AIML Documentation - AIML at a Glance - AIML 1.0.1 Draft Specification
The AIML Scripting Resource - The Chatterbot Collection - AI Navis Forum - Botworld - DigitalGirl - Botmasters' Forum - Virtual Humans

In this case “Who is John?” was the question asked, and “John is the store Manager” is the response we want to train. Click on “Say Instead” to complete the training. Now you can “Ask Again” and your bot will have learned the new response.

The other method is to utilize the chat Logs as a starting point for training.

My Pandorabots

To configure a pandorobot, click on its name in the table below.

	Name	Created	Talk	Access /hour	Access /day	Access /week	Train	Logs	Explore
<input type="radio"/>	John	08/16/2006 07:31:40 AM	Talk	0	0	0	Train	Logs	Explore
<input type="radio"/>	Bob	07/30/2006 10:28:10 AM	Talk	0	0	56	Train	Logs	Explore
<input type="radio"/>	Navillusbot*	07/11/2006 07:49:28 AM	Talk	0	0	20	Train	Logs	Explore

Publish

[Home](#) - [Pandorabots Mailing Lists](#) - [Pandorabots Documentation](#) - [Pandorabots FAQ](#)
 The A.I.I.C.F. AI Foundation - [A.I.I.C.F. Mailing Lists](#) - [AIML Documentation](#) - [AIML at a Glance](#) - [AIML 1.0.1 Draft Specification](#)

Select the “Logs” link on the row of the bot that you want to work with right now, from your “My Pandorabots” page.

Bob >> Conversation Logs

Show conversations (per page) for the last days.

<input type="checkbox"/>	Start	End	Replies	With
<input type="checkbox"/>	11/19/2006 08:20:18 PM	11/19/2006 08:20:18 PM	1 / 1	Botmaster
<input type="checkbox"/>	11/14/2006 08:33:35 PM	11/14/2006 08:33:35 PM	1 / 1	40216779
<input type="checkbox"/>	11/12/2006 11:33:24 PM	11/13/2006 07:58:23 PM	37 / 37	47272497
<input type="checkbox"/>	11/13/2006 12:42:02 PM	11/13/2006 12:46:03 PM	14 / 14	232652689
<input type="checkbox"/>	11/13/2006 04:27:42 PM	11/13/2006 04:29:14 PM	4 / 4	251021331

Mark as Unread

Conversations 1-5 of 5 | [First](#) | [Previous](#) | [Next](#) | [Last](#)

Select any of the conversations that users have had with your bot by selecting any of the numbers under the “Replies” column.

[11/10/2006 11:32:22 PM - [72.5.12.130](#)]
Human: Can you represent an advertising agency?
Bob: How old are you? Oh, you are a poet.

[11/10/2006 11:32:36 PM - [72.5.12.130](#)]
Human: Wow, there's a nonsequitur
Bob: I'm glad you're impressed. Tell me more about it.

[11/10/2006 11:32:41 PM - [72.5.12.130](#)]
Human: You're funny
Bob: Thanks you make me laugh too.

[11/10/2006 11:33:03 PM - [72.5.12.130](#)]
Human: I figured.
Bob: You are very clever.

[11/10/2006 11:33:03 PM - [72.5.12.130](#)]
Human: Greetings to our robotic overlords.
Bob: Hi there!

[11/10/2006 11:33:19 PM - [72.5.12.130](#)]
Human: Gye
Bob: That remark was either too complex or too simple for me. We were talking about A nonsequitur.

Search for a statement or question where you would like to alter your bots response, and click on the “Train” button to the left.

You will be taken back to the training interface that we just covered, and you can alter the bots response in any way you wish.

FAQ to AIML

Pandorabots offers a very useful feature called Pandorawriter (<http://www.pandorabots.com/botmaster/en/aiml-converter.html>). It is a “Dialog to AIML Parser”. We have found that it is very helpful if you happen to already have a FAQ. It will translate you FAQ or Dialog directly AIML. It works best if the answers and questions are single sentences each.

Copy and Paste your material into the first text box or select a text file on your computer for conversion.

The screenshot shows the Pandorabots AIML Converter interface. A red arrow points to the first text box containing the dialog: "What is Second Life?" and "Second Life is cool!". Another red arrow points to the "Convert" button. A third red arrow points to the "Browse..." button in the "Convert a text file dialog to AIML" section. A fourth red arrow points to the "Download AIML" button at the bottom. The generated AIML code is displayed in a text box below the "Convert" button.

Convert a dialog to AIML

What is Second Life?
Second Life is cool!

Convert a text file dialog to AIML

Generated AIML

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
  <category>
    <pattern> What is Second Life </pattern>
    <template> Second Life is cool! </template>
  </category>
</aiml>
```

Click on “Convert”, then select “Download Aiml”, and save the file to disk. If you have done any other training you will already have an update.aiml file in you Pandorabots.com bot personality. We would suggest naming this file something other then update.aiml before uploading.

Linked Messages

We have added twelve linked messages to the Greeter One and S-Bot line to indicate the current state of the bots. The messages are automatically sent out of the main scripts to all other scripts in any linked object. They are sent out on Channel 2. A standard linked message is used.

Example: `llMessageLinked(LINK_SET, 0, "INIT:OUT", NULL_KEY);`

The following is a list of these messages.

INIT:IN	Indicates that the bot is initializing all of its parameters.
INIT:OUT	Initializing parameters is complete.
TOUCHED	We put this in for simplicity, notification of prim being touched
HELLO	Indicates start of a conversation with a unique talker.
GOODBYE	Conversation has ended, including if the bot time outs.
BOTCALL:OUT	Script is sending out a HTTP request to the Pandora site. Does not reflect the initial call made by the Hello function.
BOTCALL:IN	HTTP response has been received.
SLEEP:IN	Bot is in sleep mode, will not accept chat calls.
SLEEP:OUT	Bot has left sleep state.
SENSORMODE:IN	Bot is in Greet All mode, and proximity sensors are active
SENSORMODE:OUT	Bot is in Specific Talker mode, and will wait for someone to initiate conversation.

Your bot should have an open permission “Control” script in it that has sample code of using these linked messages. The linked messages will come through exactly as shown above, and can be utilized by placing code similar to the following example in the link message receiver of any script you add to your bot.

```
if(str == "INIT:IN")    { **** Insert code here that you want activated by the init call *** }
```

The sensor also records the position of the current avatar that the bot is talking with, and will send this information out as a linked message with num equal to 1.

```
llMessageLinked(LINK_SET, 1, (string)llDetectedPos(i), NULL_KEY);
```

Any new scripts can take advantage of this information by placing the follow line of code in the linked message receiver.

```
if(num == 1) { avatarPos = (vector)str; }
```

The following is a list of chat channels and linked message nums that should be avoided by any other scripts. Operating on these channels can serious impair the functionality of your bot.

//Chat 12

//Chat -42

//Chat -1945489725

//LM 0

//LM 2

//LM 41

//LM 42

//LM 43

//LM 95

//LM 96

//LM 97

//LM 98

//LM 99

//LM 111

//LM 756