AFWSL Tags (syntax)

This is one of the most complex pieces of the game itself. The story line (sl) editor allows you to create a story line for the game. It allows you to use logic and the ability to do many different things in order to make the game more fun and interesting. You will need to have a basic idea on how logic (if then statements) work and a basic idea on how to program and script. The sl is written using the AFW scripting language (AFWSL). AFWSL is stored in an array that is 100x5 in size. It is then exported to an external file. The story lines are loaded by the maps. See the map howto for more info on this.

AFWSL contains around 63 different "tags" that will allow for various things. Below will describe the tags:

Logic Tags:

Overview:

All of the "chk" tags will return either true or false. After the value is returned, then you use the if_true and the if_false statements in order to complete the logic. The logic is a little reversed, but it will be explained later on. If you do not know what one of the syntax tags do, then look at the syntax tag section.

---

chk_for_item

This is the check for item tag. It allows you to check to see if you have >, <, or == that many items.
This is the syntax: chk_for_item, <item ID #>, <gt/lt/eq>, <quantity>
example: {chk_for_item, 1000, eq, 5,0} This will check to see if you have exactly 5 items with the ID # of 1000. if you want to check if they have more than 5, then use gt instead of eq. If you want to see if they have less than 5, then use lt, instead of eq.

---

chk_gold

This is the check gold tag. It allows you to check to see if you have >, <, or == that much gold.
This is the syntax: chk_gold, <gt/lt/eq>, <quantity>
example: {chk_gold, gt, 500,0,0} This will check to see if you have more than 500 gold.
*note – the extra zeros at the end are there to make sure you do not get garbage in the array, so put them there. However, they are not used.

---

chk_char_exists

This will check to see if a character exists or not.
This is the syntax: chk_char_exists, <who>
Example: {chk_char_exists, nick,0,0,0} this will check to see if nick is in the party or not.

---

chk_special

This will check to see if the airship, ship, or canoe exists.
This is the syntax: chk_special, <ship / boat/ canoo>
** note you must use the misspelled words or you must change the defines to the correct spelling.
Example: {chk_special, canoo,0,0,0} This will check to see whether you have the canoe or not.

---

chk_hp

This will check to see if a characters current hp is >,<, or = to a certain amount.
Syntax: chk_hp, <who>,<gt/lt/eq>, <amount>
Example: {chk_hp, toki, gt, 32000, 0} this will check to see if toki's hp is greater than 32000.

*note – the array is a signed int, which means that you cannot surpass the number 32767.

---

chk_max_hp

This will check to see if a characters max hp is >,<, or = to a certain amount.
Syntax: chk_max_hp, <who>,<gt/lt/eq>, <amount>
Example: {chk_max_hp, rexx, lt, 700, 0} this will check to see if rexx's max hp is less than 700

---

chk_mana

This will check to see if a characters current mana is >,<, or = to a certain amount.
Syntax: chk_mana, <who>,<gt/lt/eq>, <amount>
Example: {chk_mana, nick, eq, 250, 0} this will check to see if nick's mana is equal to 250

---

chk_max_mana

This will check to see if a characters max mana is >,<, or = to a certain amount.
Syntax: chk_max_mana, <who>,<gt/lt/eq>, <amount>
Example: {chk_max_mana, yoki, lt, 2500, 0} this will check to see if yoki's max mana is less than 2500

---

chk_str

This will check to see if a characters str is >,<, or = to a certain amount.
Syntax: chk_str, <who>,<gt/lt/eq>, <amount>
Example: {chk_str, nick, eq, 25, 0} this will check to see if nick's str is equal to 25

---

chk_dex

This will check to see if a characters dex is >,<, or = to a certain amount.
Syntax: chk_dex, <who>,<gt/lt/eq>, <amount>
Example: {chk_dex, nick, gt, 5, 0} this will check to see if nick's str is greater than 5

---

chk_int

This will check to see if a characters int is >,<, or = to a certain amount.
Syntax: chk_int, <who>,<gt/lt/eq>, <amount>
Example: {chk_int, nick, eq, 5, 0} this will check to see if nick's int is equal to 5

---

chk_def

This will check to see if a characters def is >,<, or = to a certain amount.
Syntax: chk_def, <who>,<gt/lt/eq>, <amount>
Example: {chk_def, nick, gt, 25, 0} this will check to see if nick's def is greater than 25

---

chk_eq

This will check to see if a characters is equipped with a certain item. Works for weapons and armor.
Syntax: chk_eq, <who>,<item ID #>
Example: {chk_eq, nick, 2000, 0, 0} this will check to see if nick is wearing the item with the ID number of 2000.

---

chk_xy

This will check if you are at a certain location on the map, while facing a certain direction.
Syntax: chk_xy, <xpos>, <ypos>, <direction>
example: {chk_xy, 3,5,up,0} check to see if you are at coordinates (3,5) while facing up.

---

asker

This tag will display a menu box that will display a “yes” or “no” and the person gets to select which one they want. If yes, then true is returned. If no or the shift key is pressed, then false is returned. You should display a message first, which is asking a question. Then use the asker.
Syntax: asker
example: {asker,0,0,0,0}

---

chk_story_marker

In order to keep track whether special events happened or not, I initiated 42 different “markers” in the save file. Each marker can be for different events. The main concept is that if the marker = 0, then the event did not happen. If the marker =1, then the event happened. You can put any number into this you want. You can use compound events. For example: you need to complete a series of events in order to attain something. After completing the first event, the marker becomes 1. now you use the chk_story_marker tag to check to see if the marker is 1. When it is, then you execute the 2nd quest. Once that quest is complete, then change the marker to 2. And repeat as necessary, until you completed all quests in order.

This tag will check to see if an inventory marker is equal to a certain number.
Syntax: chk_story_marker, <inventory marker # (2-42)>, <number>

example: { chk_story_marker, 1, 2,0,0} this will check if the inventory marker # 1 is equal to the number 2.

** Note: inventory marker #1 is reserved for shops, so you need to start at 2.

---

chk_treasure

This, like the chk_story_marker, also has a set of markers to let you know whether you have obtained a treasure chest or not. You can have up to 100 different treasure chests in the game; therefore, there are 100 different markers. This tag will check to see if the marker = 0 or the marker =1. if 0, then it will return false, ie the treasure has not be retrieved yet. If the marker =1, then it will return true that the treasure has already been retrieved.
Syntax: chk_treasure, <treasure marker (1-100)>
example: {chk_treasure, 3,0,0,0} check treasure marker 3 and see if the treasure exists or not.

---

if_true
end_if_true

The if_true statement is used to catch the return value of the "chk" routines. If the "chk" routines return true, then it will search for the first if_true tag. When it finds it, it will execute the code between the if_true and the end_if_true statement. You can have imbedded or "nested" if statements if you wish; however, it is limited to 15 deep.
Syntax: if_true … <if ID>
syntax: end_if_true
example:
{chk_treasure, 3,0,0,1}
{if_true,0,0,0,1}
    execute code here
…
{end_if_true,0,0,0,1}
{if_false,0,0,0,1} these are needed because if it is found false, then it will search for the first false
{end_if_false,0,0,0,1} statement and then execute the code inside of it. If it cannot find the if_false statement, then it will end the storyline. This is also true for the if_true statement. If it cannot find an if_true statement when the "chk" statement returns true, then it will end the storyline. You will need to make sure that you use the end_if's, because if you do not, then it will not know when to stop executing. This will cause you problems. It works just like normal logic.

The if tags have an id number at column number 4. these id numbers are related to the chk statements. See the programming in AFWSL howto for more info.

---

if_false
end_if_false

This operates the same way as the if_true statement, but it catches the false side of things instead.
Syntax: if_false
syntax: end_if_false
look at if_true for an example.

Modifier tags:

walk

This should move the main character in one general direction. However, I have not tested it, so it may not work.
Syntax: walk, <direction>
example: {walk, up,0,0,0} walk north 1 square.

move

In theory, this should move one of the characters you placed into the story line in one general direction. However, this, too, is untested. It may or may not work. Try it and see what happens.
Syntax: move, <direction>, <char>, <sprite_location ID #>
example: {move, down, sage,5,0} This will move the the sage with the ID # of 5, south one sprite.

add_sprite

This will add a towns person to the map. Towns people are those people who walk around that you can talk to. Keep in mind that these people do walk around. If you want them stationary, you will have to use add_nomove_sprite.
Syntax: add_sprite,<xpos>, <ypos>, <char>, <sprite_location ID #>
example: {add_sprite, 3,5,sage, 2} This will add a sage, with the ID # of 2 to the map at coordinates (3,5).

remove_sprite

This will remove a towns person from the map.
Syntax: remove_sprite, <sprite_location ID #>
example: {remove_sprite, 6,0,0,0} this will remove the towns person with the ID # of 6.

transfer_sprite

This will move a towns person from its current position to another position you specify. Like if they have a teleport spell or something. Use your imagination for this.
Syntax: transfer_sprite, <xpos>, <ypos>, <char>, <sprite_location ID #>
example: {transfer_sprite, 43,12, sage, 4} this will move the sage with the ID # of 4 from his current position to coordinates (43,12).

set_map

This will move your main character to a new map to the coordinates you decide. The new map will be displayed after this is used.
Syntax: set_map, <xpos>, <ypos>, <char>, <map ID #>
example: {set_map, 15,21, char_nick, 10} this will put nick at coordinates (15,21) on map 10.

warp

This tag allows you to add extra warp points to your map. Bascially if you end up at coordinates (x,y) then it will "warp" you to a new set of coordinates instantly. This is used for stairs and special transports and such. However, only use it in the event the number of warp points exceed 18 for any given map.

Syntax: warp, <xpos>, <ypos>, <new xpos>, <new ypos>
example: {warp, 3,5,12,15} if you enter coordinates (3,5) then you will be instantly relocated to coordinates (12,15)

---

msg

This tag will display a message. In order to display the message, you must first create a message file with messages in it. You must give the message file a number and then know which row the message is stored at in order to call the message.
Syntax: msg, <message file #>, <message ID #>, <TRUE/FALSE>
Example: {msg, 2, 64,TRUE,0} this means that you are loading the file "mess2" and then loading it. Then you are going to row 64 of the array and displaying the message that lies there. The TRUE/FALSE tells whether you should pause the message (TRUE) or not pause it (FALSE). General rule: always pause the message, not unless you are asking a question and you need to use the "asker" tag. This example shows a simple message that is said. Below is an example on how to use FALSE.

Example 2:
{msg, 2, 64,FALSE,0}
{asker,0,0,0,1}
{if_true,0,0,0,1}
…
{end_if_true,0,0,0,1}
{if_false,0,0,0,1}
…
{end_if_false,0,0,0,1}

Example 2 is showing the proper way to execute a message that would ask a question and require a response. Since the FALSE is activated, it continues onto the next line of the story line, instead of pausing. The asker will pause and when the user decides on what to do, the below if statements will be executed or bypassed.

Gotoa

If you are old school, then you may hate the goto statement; however, I love the goto statement, so I added it into my logic. Basically, you should know your row numbers, because you have to hard code them in. Since this should be the case, gotoa simply jumps the the row number you specify.

Syntax: gotoa, <row number>
example: {gotoa, 5,0,0,0} this will reset your current row number to 5 and start executing code from row number 5. Be careful, because you could create an infinite loop that you cant even get out of.

---

load_story

Because story lines are small in size, I enabled an expander to allow for a longer story line. This expander is called load_story. Basically, it load another story line file into place of the current one and then set you at whatever row you decide. It is like a loader and the gotoa statements combined.
Syntax: load_story, <story line ID #>, <row #>
example: {load_story, 4, 0, 0,0} this will load the story line file 4 and put you at row 0, which is the first line in the new story line. Keep in mind that the old story line file will be erased and only the new one will be active.

---

set_story_marker

This will change the inventory marker to a different number. See the chk_story_marker for more information about this.
Syntax: set_story_marker, <inventory marker # (2-42)>, <number>
example: {set_story_marker, 2,1,0,0} this will set inventory marker #2 to the value of 1.

---

inc_exp

This will give a character exp.
Syntax: inc_exp, <who>, <amount>
example: {inc_exp, nick, 2000,0,0} this will give nick 2000 exp.

---

line

This uses the line routine to draw a line from (x1,y1) to (x2,y2). I thought this could be useful for if you wanted to draw some gfx.
Syntax: line, <x1>, <y1>,<x2>,<y2>
example: {line, 2,3,2,23} draw a line form (2,3) to (2,23) (this would be a vertical line)

---

set_message

This will give a towns person something to say. See Programming in AFWSL for more info.
Syntax: set_message, <sprite_location ID #>, <message file #>, <message ID #>
example: {set_message, 4, 2, 54, 0} make the towns person with the ID #4 say row 54 from the mess2 file.

---

add_nomove_sprite

This will add a towns person that does not move around.
Syntax: add_nomove_sprite, <xpos>, <ypos>, <char>, <sprite_location ID #>
example: {add_nomove_sprite, 3,5,sage, 5} add a non movable sage, with the ID #5 to the coordinates (3,5)

---

add_char

This will add a character to the party.
Syntax: add_char, <who>, <average stats? T/F>
example: {add_char, toki, TRUE, 0,0} this will add toki to the party and it will average her stats, so you don't have to create them. Averaging the stats is a helpful thing. Basically, it will take all the stats of all the party members, add them together and then average out each stat. That average will become the added character's (in this case, toki) stats. If you wish to set the stats yourself, then use FALSE instead of TRUE.

---

remove_char

This will remove a character from the party.
Syntax: remove_char, <who>
example: {remove_char, nick,0,0,0} this will remove nick from the party.

---

add_item

This will add a certain number of items to the inventory. Remember the max amount of items is 500.
Syntax: add_item, <item ID #>, <amount>
example: {add_item, 1023, 3, 0,0} this will add 3 items with the ID # of 1023

---

remove_item

This will remove a certain number of items from the inventory. If you want to remove all, then use a number over 500. If you use a negative number, then it will add to the items.
Syntax: remove_item, <item ID #>, <amount>
example: {remove_item, 1023, 1000, 0,0} this will attempt to remove 1000 of the items with the ID# of 1023. Basically, all of the items will be removed, because there is a max of only 500.

---

add_gold

This will add a certain amount of gold to your party. The max you can add is about 32000, so if you want to add more than that, then you will have to repeat this command multiple times until you attain the desired amount that you wanted.
Syntax: add_gold, <quantity>
example: {add_gold, 1500,0,0,0} this will add 1500 gold to your party.

---

remove_gold

This will remove gold from your party.
Syntax: remove_gold, <quantity>
example:
{remove_gold, 32000,0,0,0}
{remove_gold, 32000,0,0,0}
Since I did it twice, it will remove 64000 gold instead of 32000 gold.

---

add_special

This will add the canoe, ship, or airship to your inventory. However, you will need another command to set the map and X/Y coordinates.
Syntax: add_special, <ship / boat/ canoo>
example: {add_special, ship,0,0,0} this will add the airship.

---

reduce_hp

This will basically “deal damage” to a character. It simply reduces their current hp by a certain amount. Note: they can die from this.
Syntax: reduce_hp, <who>, <amount>
example: {reduce_hp, rexx, 1200,0,0} this will reduce rexx's current hp by 1200.

---

reduce_mana

This will cause the person to “use” their mana by a certain amount. It reduces a character's current mana.
Syntax: reduce_mana, <who>, <amount>
example: {reduce_mana, rexx, 200,0,0} this will reduce rexx's current mana by 200.

---

inc_hp

This will basically “heal” a character by a certain amount.
Syntax: inc_hp, <who>, <amount>
example: {inc_hp, rexx, 200,0,0} this will heal rexx by 200 hp.

---

inc_mana

This will “heal” a character's mana.
Syntax: inc_mana, <who>, <amount>
example: {inc_mana, rexx, 200,0,0} this will heal rexx's mana by 200.

---

heal_all

This will heal all players a certain amount of hp and a certain amount of mana
Syntax: heal_all, <HP amount>, <Mana amount>
example: {heal_all, 500, 200,0,0} this will restore 500 hp and 200 mana to all characters

---

full_heal

This will restore about 1 million hp and mana to all characters.
Syntax: full_heal
example: {full_heal,0,0,0,0}

---

inc_max_hp

This will increase a character's max hp by a certain amount.
Syntax: inc_max_hp, <who>, <amount>
example: {inc_max_hp, nick, 500,0,0} this will increase nick's max hp by 500.

---

inc_max_mana

This will increase a character's max mana by a certain amount.
Syntax: inc_max_mana, <who>, <amount>
example: {inc_max_mana, nick, 500,0,0} this will increase nick's max mana by 500.

---

inc_str

This will increase a character's max str by a certain amount.
Syntax: inc_str, <who>, <amount>
example: {inc_str, nick, 5,0,0} this will increase nick's max str by 5.

---

inc_dex

This will increase a character's max dexr by a certain amount.
Syntax: inc_dex, <who>, <amount>
example: {inc_dex, nick, 5,0,0} this will increase nick's max dex by 5.

---

inc_int

This will increase a character's max int by a certain amount.
Syntax: inc_int, <who>, <amount>
example: {inc_int, nick, 5,0,0} this will increase nick's max int by 5.

---

inc_def

This will increase a character's max def by a certain amount.
Syntax: inc_def, <who>, <amount>
example: {inc_def, nick, 5,0,0} this will increase nick's max def by 5.

---

inc_skill

This will increase a specific skill or spell on a character by a certain amount.
Syntax: inc_skill, <who>, <skill ID #>, <skill %>
example: {inc_skill, nick, 5000,40,0} this will increase nick's H2H skill by 40%

---

remove_special

This will remove either the canoe, airship, or ship from the inventory.
Syntax: remove_special, <ship / boat/ canoo>
example: {remove_special, canoo,0,0,0} remove the canoe from the inventory.

---

move_special

After you add a canoe, airship, or ship, you will have to place it on a map. This is what this command does.
Syntax: move_speical, <ship / boat/ canoo>, <xpos>, <ypos>, <map ID #>
example: {move_special, boat, 17,23,3} this will place the ship on map 3 at coordinates (17,23)

---

clear

This will call the clrscr() function.
Syntax: clear
example {clear,0,0,0,0}

---

add_ability

This will add either a skill or spell to a character; however, it will not give them a skill%. You will need to use inc_skill to increase or set the percentage on the skill or spell.
Syntax: add_ability, <who>, <skill / spell>
example: {add_ability,nick,h2hc,0,0} this will add the h2hc combat skill to nick.

---

set_level

This will set the level of one character. It will not increase the level, but make the character's level = <number>
Syntax: set_level, <who>, <number>,<cheat override T/F>
the <cheat override T/F> is either TRUE or FALSE. If true, then if the current level of the character is higher than the one you are trying to set the level to, then the level will not be reset. If false then it will force the level to a certain number without reguard to whether it is higher or lower.
example: {set_level,nick,100,FALSE,0} set nick's level to 100, but do not allow the cheat to override this.

---

life

This will bring a character back to life with a certain amount of hp. You will need this if you add a character for the first time. Life is the only way to increase a character's hp from 0. healing only increases in the event the hp is >=1.
Syntax: life, <who>, <amount>
example: {life,nick,30,0,0} bring nick back to life with 30 hp.

---

set_tnl

This will set the tnl of a character to a certain number.
Syntax: set_tnl, <who>,<amount>
example: {set_tnl,nick,35,0,0} set nick's tnl to 35 exp.

---

add_rand_battles

This is a special tag that will let you set a battle on a map that normally does not have battles. For example, lets say you create a town. Then an event happens in the story where the town becomes a war zone. This is where add_rand_battles comes into play. See the map howto for more info on adding battles.
Syntax: add_rand_battles, <mob file id #>, <low mob>. <high mob>
example: {add_rand_battles, 2, 1,8,0} cause the mobs2 file to be used, but use only the mobs from 1 to 8.

---

add_rand_battles_ocean

Same concept as add_rand_battles, but it does it for the ocean.
Syntax: add_rand_battles_ocean, <mob file id #>, <low mob>. <high mob>
example: {add_rand_battles_ocean, 3, 1,8,0} cause the mobs3 file to be used, but use only the mobs from 1 to 8.

---

add_rand_battles_river

Same concept as add_rand_battles, but it does it for the rivers.
Syntax: add_rand_battles_river, <mob file id #>, <low mob>. <high mob>
example: {add_rand_battles_ocean, 4, 1,6,0} cause the mobs4 file to be used, but use only the mobs from 1 to 6.

---

set_world

This is a special command that I created to do modifications on the world map. It will change one sprite at a certain coordinate to another sprite. The main idea behind this was the “locked and Unlocked” door idea.. a locked door cannot be walked upon; however, if a key were used, then you “unlock” the door and open it. This posed a problem, because maps do not change. This command; however, makes it so that you can change the look of a map, such as switching out a closed door with one that is opened that you can walk though.

Syntax: set_world, <x row>, <y row>, <sprite>
** Note that <x row> and <xpos> are 2 different things. The <xpos> is the position in which a character is at. The <x row> is the row number for the map array. See the map howto for more info on this.
Example: {set_world, 4,2,123,0} redo the sprite at coordinates (4,2) to sprite # 123.

---

free_stuff

This activates the free stuff cheat. This cheat makes all items free if activated. You must have the gold to buy the items, but when you buy the items, you don't lose gold when buying stuff. This does not work for training stats.

Syntax: free_stuff, <TRUE/FALSE>
Example: {free_stuff, TRUE,0,0,0} this will activate the cheat. FALSE will turn the cheat off.

Character Defines:

this is a list of different mob sprites that have been defined that you can you. Note: you can add more to this.

prince, sage, robes, lady, woman, old_woman, guard, bmage, wmage, warrior, scholor, ninja, guy, fighter, char_rexx, char_yoki, char_toki, char_nick

syntax tags:

<direction> can be 1 of 4 directions: up, down, left, or right.
<xpos> and <xpos> are the X/Y coordinates you character will be standing at on the map.
<map ID #> - this is the map id number of any given map you created.
<who> can be 1 of 4 names: nick, toki, yoki, or rexx.
<gt/lt/eq> stands for greater than (gt), less than (lt), and equal to (eq).
<inventory marker # (2-42)> this is a tag, which is used for the inventory markers. There are 42 markers, so you must specify which marker to use. The marker numbers range from 1 to 42. ** Note: inventory marker #1 is reserved for shops, so you need to start at 2.
<treasure marker (1-100)> this tag is like the inventory marker. There are 100 markers to mark whether the treasure has been gotten or not. The range is from 1 to 100.
<char> - this tag needs a character number, so it can display their bitmap. Look that the character defines section to see a list of characters you can put into it.
<sprite_location ID #> - this tag represents the ID number of the sprite you added to the map. This id number keeps track of the sprite and is the linker that is used to link messages that random towns people can say. Since there can be multiple towns people that look the same, the ID is used to keep track of all the different people. The ID number corresponds to the row number an array that contains all of the towns people. There can be up to 60 towns people per map, meaning that your ID numbers range from 0 → 59.