User Configurable Spot Filters in DXSpider - A Primer

Compiled By W3BG – Jim Samuels

With Introduction by N3RD - Dave Hawes

10/19/2002

Table of Contents:

Introduction.

The PacketCluster software written in the mid-80s by Dick Newell, AK1A, has served us well. Dick has moved on, though, and has not supported the software with updates, etc. for the last 10 years. Numerous PacketCluster "clones" have come and gone over the years, however there is one, called DX Spider, which provides a very similar user interface to that of AK1A, allows internet connections of users and node-to-node links, is actively supported by the author, and best of all…is freeware. FRC has started to convert several nodes to Spider.

One of the strengths of DX Spider is its very powerful and flexible DX spot filtering routines. These filters are totally different from anything we learned how to do with PacketCluster, and along with their power and flexibility comes somewhat of a learning curve. Hence the need for this primer.

In the following sections, you will learn that you can filter DX spots by:

Frequency of the spot
Mode of the spot
Callsign of the spot (by state, country, zone, or specific callsign)
Callsign of the spotter (by state, country, zone, or specific callsign)
Callsign of the source node of the spot (by state, country, zone, or specific callsign)

With a few keystrokes, you can set up a filter for the CQ WW SSB contest, for example, that says that you only want to see SSB spots on the contesting bands. In the ARRL contest, it is simple to exclude spots for Ws and VEs. For example, the best all around one-line filter for users in the CQ WW SSB contest would be:

```
accept/spots on contesthf/ssb
```

This simply reads, "I want to get spots on the hf contesting bands on SSB only."

Jim Samuels, W3BG, has put together this primer which not only provides complete details on the format for all the available filter commands, but also provides useful examples that can be simply typed in, without the need to learn the specifics.

As always, your local sysop is available to help you out, if need be. Don't hesitate to contact him for assistance.

73 – Dave N3RD

Forward

While attempting to learn how DXSpider filters work, I found that had to glean bits and pieces of information from the DXSpider *User Manuals* and *Administrators Guide* as well as various posted messages, help files and the program and data-base files themselves. Therefore, this is by no means an original work. I have used and in some cases copied from some of these sources. What I have tried to accomplish is to gather this scattered information, put it in one spot (please pardon the pun) so others might benefit. I would advise those with interest to go back and read these other sources at their leisure.

Configuring Spot Filters

What is a spot filter?

A spot filter is one rule (a one line spot filter) or multiple rules (multiple line spot filters) which a user can setup with-in DXSpider to control which specific spot(s) are received at the shack console. These configurable filters/rules reside on the DXSpider node and are stored along with the user's other information. Filters can be likened to a car wash . . . . . like cars; information goes in one end dirty, gets washed and comes out the other end cleaned.

All spots received from other users on the cluster, or those received from other nodes, start out life destined for each and every connected user's console. If spot filtering has been configured, all spots headed for that user first go into the filter input, are processed and sent out the other end of these filters before being sent to the user's console. Like a car wash, each spot goes through one or many stages depending on whether the user wanted a simple or a super-duper filtering job. Along the way, the spot gets scrubbed, unwanted information removed or wanted information passed on and finally the wanted spots only are spit out the other end – nice and clean with all unwanted "stuff" sent down the drain to the infamous "bit-bucket." How can filters be used? For example, let's say our local user has never owned a microphone in this life and definitely doesn't want to see any of those useless SSB spots. Our user simply sets up a basic filter to reject any SSB spots before they reach the user's console. Similarly, it's now the ARRL CW DX contest weekend, so not only does our user not want to see SSB spots, but now doesn't want to see any UHF, VHF, DATA or any US/Canadian "DX" spots. Our user now only accepts HF CW CONTEST spots and in the same rule rejects spots for W and VE stations. In these and many more situations, "**filters are our friends**."

Types of spot filters used in DXSpider

Basic filter types are "**accept**", "**reject**", and "**clear**."

Reject filters – any spots that match will be dumped, all others passed on.

Accept filters – any spots that match are passed on, all others are dumped.

Clear filters – the filter slot(s) referenced will be cleared from the filter repository

For the most part we will use only reject and accept filters. These are the main filter types. Basically, *reject* means dump it and *accept* means take it and pass it on to the user. By nature, accept filters are more powerful than reject filters. A user can generally do with a one line accept rule what it could take many lines of reject rules to accomplish. However, the flip-side of this statement is that a series of reject filters are usually easier to administer and change.

Numbering lines and slots

There are ten usable filter slots in DXSpider. Each slot holds one reject and one accept rule. Therefore, each type filter can have up to ten lines of rules, contained in these ten slots. The filter rules must be numbered sequentially, i.e., 0-9 lines of reject filter rules and 0-9 lines of accept filter rules to correspond to their respective slot position. If no number is used, every line is assumed to be in slot 1 and the addition of a second filter line of the same type without a number will just over-write the first that was previously written to slot 1. (Why not slot 0? I don't know. This is the way it works.) **Important:** The filter rules are applied in sequence, i.e., 0-9. If a line matches, action is taken on that line. The filter sequence acts on rules in the order listed. It acts on the reject filter in each slot before acting on the accept filter contained in that slot. If the slot is completely blank or if a reject or accept filter line missing in that slot it skips right over to the next filter rule in the sequence. A picture of a filter set might look like this:

Execution Sequence	**Slot Num ber**	**Filter Rule**
1	Slot0	*reject/spot 0 <pattern>*
2		*accept/spot 0 <pattern>*
3	Slot1	*reject/spot 1 <pattern>*
4		*accept/spot 1 <pattern>*
5	Slot2	*reject/spot 2 <pattern>*
6		*accept/spot 2 <pattern>*
.		.
19	Slot9	*reject/spot 9 <pattern>*
20		*accept/spot 9 <pattern*

Reject before accept.
This is not a good rule for life, but it makes sense for DXSpider filters. As a general rule, reject filter rules within a slot are always executed before accept filter rules. There is a very good reason for this. If a spot doesn't match a reject filter, the spot is passed to the next filter line in the set. However, if a spot matches an accept filter, it is sent immediately to the user.

Using Multiple Reject Filter Rules.
Another important concept to know is that you can do everything you want to do with multiple reject filters **AND NO ACCEPT FILTERS**. By default, if a spot doesn't match any of the reject filter definitions, then the system considers you want the spots and sends it to you. For example, the following two filters perform exactly the same thing:

```
accept/spots on contesthf
reject/spots not on contesthf
```

So, why would we choose one vs. the other? Using reject syntax allows you to add another filter line easily, without disturbing the first line. A real example will show us how this works. Let's say that there is a RTTY contest coming up and you don't wish to see the RTTY spots. Simply add another reject filter like this:

```
reject/spots 2 on hf/rtty
```

Note that we need to specify that this is second line of reject filter definitions. Also, the "RTTY" sub-band specification has to be associated with a range of bands; it can't be specified all by itself. So, we just add it behind the range of bands defined by "HF". So, in our example, if the user does a *show/filter*, he will be told by the Spider that his current filters are:

```
filter 1 reject not on contesthf
filter 2 reject on hf/rtty
```

With these filters set up, if a spot comes through on 14085 kHz, the filter works like this:

filter1: Is spot NOT on the HF contest bands? No. The spot doesn't match the filter definition, so pass it to next filter.
filter2: Is spot within the freq. Range defined for RTTY? Yes. Since the spot matches the filter definition, the spot is rejected, and the users never see it.

Had the frequency of the spot been 14025, then the spot would have not matched the filter2 definition either, would have passed through all the filters, and would have been sent to the user at the end of the filter set. Also, had the spot been on 10 MHz, it would have met the definition of filter1, been rejected immediately, and the filtering process would have stopped before processing filter2.

In addition, the filtering system has a rough time handling accept filters followed by reject filters and adds inefficiency to the processing. (Note: a reject as a "qualifier" to an accept rule in an accept filter line is okay as we will see below)

A very useful command

To see all active filters in use at any time, just type the following command:

```
show/filter
```

Case does not matter

In entering any filter – case does not matter. Upper, lower, or mixed case will not effect how filters work or perform

Qualifiers

Logical operands can be used in rule sets to combine multiple actions or qualify others. These are:

and	a *and* b= action
not	a *not* b= action
or	a *and not* (c *or* b)= action

Note: as a general rule when *or* is used you must also use parentheses (). We will see how these can be used in examples later.

Comma Separation

Any command can have multiple pattern variables if commas separate them. For example:

reject/spot call_state nj,ny,pa,de,md

Reject filters

A reject filter line means that if spot matches send it to the trash, dump it, do not send it down the line to the next rule or to the user, but pass-on all other spots that do not match.

Syntax: *reject/spots [0-9] <pattern>*

Any of the following *patterns* may be used in this line:

freq <range>
on <range>
info <string>
call <prefixes>
call_dxcc <numbers>
call_itu <numbers>
call_zone <numbers>
call_state <state2-letter abbreviations>
by <prefixes>
by_dxcc <numbers>
by_itu <numbers>
by_zone <numbers>
by_state <state2-letter abbreviations>
origin <prefixes> Used primarily be SYSOPS, not by users and not discussed.
channel <prefixes> Used primarily be SYSOPS, not by users and not discussed.

Filters to reject spots based on frequency:

Syntax: *reject/spot [0-9] freq <range>* or;
reject/spot [0-9] on <range>

Important: both *freq* and *on* are exactly the same and can be used interchangeably – most persons use *on* (less typing.)

For *range*, you can specify a frequency like 7040, a range of frequencies like 0/30000 (the whole HF band) or use any of the “band” or “region” names defined in the *show/bands* command.

Bands Available:

73kHz:	71 -> 75
136kHz:	135 -> 138
160m:	1800 -> 2000
80m:	3500 -> 4000
60m:	5258 -> 5407
40m:	7000 -> 7400
30m:	10100 -> 10150
20m:	14000 -> 14350
17m:	18068 -> 18168
15m:	21000 -> 21450
12m:	24890 -> 24990
10m:	28000 -> 29700
military:	29700 -> 50000, 230000 -> 420000
band1:	47000 -> 49999, 52000 -> 68000
6m:	50000 -> 52000
pmrlow:	68000 -> 87500

4m:	70000 -> 70500
band2:	87500 -> 108000
aircraft:	108000 -> 137500
pmrmid:	138000 -> 165000
2m:	144000 -> 148000
pmrhigh:	165000 => 174000
band3:	176000 => 230000
220:	220000 => 222000
pmruhf:	425000 => 430000, 440000 => 471000
70cm:	430000 => 450000
band4:	471000 => 550000
band5:	550000 => 868000
23cm:	1240000 => 1325000
13cm:	2310000 => 2450000
9cm:	3400000 => 3475000
6cm:	5650000 => 5850000
3cm:	10000000 => 10500000
12mm:	24000000 => 24250000
6mm:	47000000 => 47200000

Regions Available:

all:	73khz 136khz 160m 80m 60m 40m 30m 20m 17m 15m 12m 10m 6m 4m 2m 220 70cm 23cm 9cm 6cm 3cm 12mm 6mm
vhfradio:	band1 band2
vhf:	6m 4m 2m 220
contesthf:	160m 80m 40m 20m 15m 10m
warc:	60m 30m 17m 12m
pmr:	pmrlow pmrmid pmrhigh pmruhf
spe:	10m 6m 4m 2m
shf:	23cm 13cm 9cm 6cm 3cm
vlf:	73khz 136khz
uhftv:	band4 band5
hf:	160m 80m 60m 40m 30m 20m 17m 15m 12m 10m
vhftv:	band1 band3
uhf:	70cm 23cm

Examples:

reject/spot 0 freq 7040	This will reject spots on 7,040 kHz and pass all others.
reject/spot 1 on 0/30000	This will reject spots from 0 to 30,000 kHz and pass on all others.
reject/spot 2 freq 2m	This will trash all spots in the frequency range 144000 -> 148000 kHz and pass on all others.

reject/spot 3 on vhf	This rule will reject all spots on 6m, 4m, 2m, and 220 and pass on all others.
reject/spot 4 freq hf and freq uhf	This rule will dump all spots on the 160m, 80m, 60m, 40m, 30m, 20m, 17m, 15m, 12m, 10m bands and all spots on 70cm and 23cm bands passing all other spots.
reject/spot on all	This is a special spot to be used only by members of the Yankee Clipper Contest Club during contest weekends. Hi!

Sub-bands as part of range
In conjunction with *range*, you can use the following sub-band names:

cw, rtty, data, ssb, and sstv

by using a forward-slash [(*band or region)/sub-band*] as part of the range definition.
For example:

reject/spot 0 freq hf/ssb This rule will reject all HF phone spots passing on all others

reject/spot 1 on hf/cw and not (on hf/data or on hf/rtty) This filter rule will reject all HF CW spots but will not reject DATA and RTTY spots in the CW range and will pass on all other spots.

Filters to reject spots based on the "info" data in the spot
Syntax: *reject/spot [0-9] info <string>*

This filter is used to key on information contained in the information section of the spot. One could use this to reject any spots containing IOTA, QSL OP or any other "key-word" used in the information string of the spot.
Examples:

reject/spot 0 info IOTA *This filter will reject spots containing IOTA information and pass on all others*

reject/spot 1 on hf/cw and not info iota This filter will reject all general CW spots on HF, but will still permit any HF CW spots that contain iota information in addition to passing all others.

reject/spot 2 info QSL	This filter will reject spots asking or containing QSL information and pass on all others

Note: The following series of filters are based on *call* and *by*. *Call* always references the call sign of the spotted DX station. *By* always references the call sign of the spotting station.

Filters to reject spots based on call**:**

Syntax: *reject/spot [0-9] call <prefixes>*

This filter is misleading in a way. It is strictly based on the spotted call sign letters or numbers entered and not based on countries or DXCC entities. One could filter on *JIMSAM62* if desired.

Examples:

reject/spot 0 call G	This filter will reject spots for G1AAA, GJ2BBB, and GW3CCC and will pass on spots for M0AAA.
reject/spot 1 call PA	This filter will reject spots for PA3AAA and pass on spots for PB4BBB
reject/spot 2 call K	This filter will reject spots for K1AA, KC4AAA, and KH6DDD and pass on spots for W3BG and N3RD

Filters to reject spots based on call_dxcc**:**

Syntax: *reject/spot [0-9] call_dxcc <numbers or prefixes>*

This filters is based on DXCC entities and uses either the country prefix or the DXCC entity number found by using the command *show/prefix <prefix>*

As in:

```
show/prefix w
W DXCC: 226 ITU: 7 CQ: 4 LL: 43 0 N 87 54 W (W, United-States-W)
```

show/prefix VE
VE DXCC: 197 ITU: 9 CQ: 5 LL: 45 18 N 66 6 W (VE, New-Brunswick-VE)
DXCC: 197 ITU: 9 CQ: 5 LL: 48 30 N 56 0 W (VE, Newfoundland-VE)
DXCC: 197 ITU: 9 CQ: 5 LL: 44 36 N 63 36 W (VE, Nova-Scotia-VE)
DXCC: 197 ITU: 4 CQ: 5 LL: 45 30 N 73 36 W (VE, Quebec-VE)
DXCC: 197 ITU: 4 CQ: 4 LL: 43 42 N 79 24 W (VE, Ontario-VE)
DXCC: 197 ITU: 3 CQ: 4 LL: 49 54 N 97 6 W (VE, Manitoba-VE)
DXCC: 197 ITU: 3 CQ: 4 LL: 50 30 N 104 36 W (VE, Saskatchewan-VE)
DXCC: 197 ITU: 2 CQ: 3 LL: 51 0 N 114 6 W (VE, Alberta-VE)
DXCC: 197 ITU: 2 CQ: 3 LL: 49 18 N 123 6 W (VE, British-Columbia-VE)
DXCC: 197 ITU: 75 CQ: 1 LL: 60 42 N 135 6 W (VE, Yukon-VE)

Example:

reject/spot 0 call_dxcc 226, 197 This spot filter will reject all spots for US and Canada stations and pass on all others.

reject/spot 1 call_dxcc w,ve not call w2wtc This spot filter will reject all spots for US and Canada stations and pass on all others including the special event station, W2WTC, who I want to work the next time he is on the air.

Filters to reject spots based on call_itu .

Similarly**, call_itu** and **call_zone** use ITU regions that can also be obtained using the *show/prefix <prefix>* command (see above.)

Syntax: *accept/spot [0-9] call_itu <numbers>*

Example:

reject/spot 0 call_itu 7 This spot filter will reject all spots for ITU region 7 and pass on all others.

Filters to reject spots based on call_zone**:**

Syntax: *reject/spot [0-9] call_zone <numbers>*

This filters is based on CQ zones and uses the CQ zone number found by using the command *show/prefix <prefix>* (see above.)

Example:

reject/spot 0 call_zone 5 This spot filter will reject all spots for CQ zone 5 and pass on all others.

Filters to reject spots based on call_state**:**

Syntax: *reject/spot [0-9] call_state <state2-letter abbreviations>*

This filters is based on state of the call spotted for those call signs contained in the usdb database. Type the command: show/usdb <call sign> to see an example of a listing in the database.

```
show/usdb k3ww
K3WW    -> Perkasie, PA
```

Example:

reject/spot call_state nj,ny,pa,de,md This spot filter will reject all spots for stations in the Mid-Atlantic states and pass on all others.

Filters to reject spots based on by**:**

by filters are similar to and function exactly as ***call*** filters except that they act on the **spotting station call sign** and **not** the **spotted call sign**

So . . . . .

Syntax: *reject/spot [0-9] by <prefixes>* This filter is similar to and functions like the *call <prefixes>* (See above) except that it rejects spots generated by the **spotting call sign** and passes all other spots.

Syntax: *reject/spot [0-9] by_dxcc <numbers>* This filters is based on DXCC entities and uses the DXCC entity number found by using the command *show/prefix <prefix>* and it rejects spots generated within the **spotting DXCC entity** and passes all other spots.

Syntax: *reject/spot [0-9] by_itu <numbers>* This filter is based on ITU regions and uses the ITU region number found by using the command *show/prefix <prefix>* (see above) except that it rejects spots generated by a **spotting call sign** within the ITU region and passes all other spots.

Syntax: *reject/spot [0-9] by_zone <numbers>* This filter is based on CQ zones and uses the CQ zone number found by using the command *show/prefix <prefix>* (see above) except that it rejects spots generated by a **spotting call sign** within the CQ zone and passes all other spots.

Syntax: *reject/spot [0-9] by_state <state2-letter postal codes>* This filter is based on the state of the spotting station found by using the command *show/usdb <call sign>* and passes all other spots.

Accept filters
A accept filter line means that if spot matches pass it on to the user, send it down the line to the next rule or to the user, and trash, dump, all other spots that do not match to the next filter line.

Syntax: *accept/spots [0-9] <pattern>*

freq <range>
on <range>
info <string>
call <prefixes>
call_dxcc <numbers>
call_itu <numbers>
call_zone <numbers>
call_state <state2-letter abbreviations>
by <prefixes>
by_dxcc <numbers>
by_itu <numbers>
by_zone <numbers>
by_state <state2-letter abbreviations>
origin <prefixes> Used primarily be SYSOPS, not by users and not discussed.
channel <prefixes> Used primarily be SYSOPS, not by users and not discussed.

Using these patterns, we can accept spots based upon:
Frequency of the spot
Call sign of the spot (country or zone)
Call sign of the spotter (country or zone)
Contents of the “information field” which comes with the spot

Filters to accept spots based on frequency**:**

Syntax: *accept/spot [0-9] freq <range>* or;
accept/spot [0-9] on <range>

Important: as noted before both *freq* and *on* are exactly the same and can be used interchangeably.

For *range*, you can specify a frequency like 7040, a range of frequencies like 0/30000 (the whole HF spectrum) or use any of the band/regions names defined in the *SHOW/BANDS* command (see above.).

Examples:

accept/spot 1 on 0/30000 This will pass on a HF spots only from 0 to 30,000 kHz and dump all others.

accept/spot 2 freq 2m This passes on all spots in the frequency range 144000 -> 148000 kHz and trash all others.

accept/spot 3 on vhf This rule will only pass on spots on 6m, 4m, 2m, and 220 and reject all others.

accept/spot 4 freq hf and freq uhf This rule will pass on all spots on the 160m, 80m, 60m, 40m, 30m, 20m, 17m, 15m, 12m, 10m bands and all spots on 70cm and 23cm bands only. All other spots are trashed.

Sub-bands as part of range

In conjunction with *range*, you can use the following sub-band names: CW, RTTY, DATA, SSB, and SSTV by using a back-slash [(*band or region)/sub-band*] as part of the range definition.

Examples:

accept/spot 0 freq hf/ssb This rule will only accept and pass on HF phone spots rejecting all others

accept/spot 1 on hf/cw and not (on hf/data or on hf/rtty This filter rule will accept all HF CW spots but will not include DATA and RTTY spots in the CW range. In addition all other spots will be dumped.

Filters to accept spots based on info**:**

Syntax: *accept/spot [0-9] info <string>*

This filter is used to key on information contained in the information section of the spot. One could use this to accept any spots containing IOTA, QSL OP or any other "key-word" used in the information string of the spot.

Examples:

accept/spot 0 info IOTA This filter will accept spots containing IOTA information only and reject all others

accept/spot 1 on 10m/ssb and info iota This filter will accept only 10m SSB spots, but will still permit any spots that contain iota information in addition - rejecting all other spots.

accept/spot 2 info QSL This filter will accept spots asking or containing QSL information and dump all other spots

Note: The following series of filters are based on *call* and *by*. *Call* always references the call sign of the spotted DX station. *By* always references the call sign of the spotting station.

Filters to accept spots based on call**:**

Syntax: *accept/spot [0-9] call <prefixes>*

This filter is misleading in a way. It is strictly based on the spotted call sign letters or numbers entered and not based on countries or DXCC entities.

Examples:

accept/spot 0 call G This filter will accept spots for G1AAA, GJ2BBB, and GW3CCC and reject all others, including M0AAA.

accept/spot 1 call PA This filter will accept spots for PA3AAA and reject spots for PB4BBB as well as all others.

accept/spot 2 call K This filter will accept spots for call signs beginning with “K”, i.e., K1AA, KC4AAA, KH6DDD and reject spots for W3BG and N3RD as well as all other spots.Filters to accept spots based on **call_dxcc**:

Filters to accept spots based on call_dxcc**:**

Syntax: *accept/spot [0-9] call_dxcc <numbers or prefixes>*

This filters is based on DXCC entities and uses either the country prefixes or the DXCC entity number found by using the command *show/prefix <prefix>* See example of *show/prefix <prefix>* above.

Examples:

accept/spot 0 call_dxcc 226,197 or
accept/spot 0 call_dxcc ve,w
(Both will work) These spot filters will accept all spots for US and Canada stations and trash all others.

accept/spot 1 call_dxcc w not call w3fm This spot filter will accept all spots for US stations and yet reject any spots for W3FM who is always being spotted by Europeans and filling up my screen.

Filters to accept spots based on call_itu .

Similarly**, call_itu** and **call_zone** use ITU regions that can also be obtained using the ***show/prefix <prefix>*** command (see above.)

Syntax: *accept/spot [0-9] call_itu <numbers>*

Example:

accept/spot 0 call_itu 7 This spot filter will accept all spots for ITU region 7 and reject all others.

Filters to accept spots based on call_zone.

Syntax: *accept/spot [0-9] call_zone <numbers>*

This filter is based on CQ zones and uses the CQ zone number found by using the command *show/prefix <prefix>* (see above.)

Example:

accept/spot 0 call_zone 5 This spot filter will accept all spots for CQ zone 5 and reject all others.

Filters to accept spots based on call_state**:**

Syntax: *accept/spot [0-9] call_state <state2-letter postal codes>*

This filters is based on state of the call spotted for those call signs contained in the usdb database.

Example:

accept/spot 0 call_state pa This spot filter will accept all spots of stations located in the Commonwealth of Pennsylvania and reject all others. It's the PA QSO Party Weekend.

Filters to accept spots based on by**:**

by filters are similar to and function exactly as ***call*** filters except that they act on the **spotting station call sign** and **not** the **spotted call sign**

So . . . . .

Syntax: *accept/spot [0-9] by <prefixes>* This filter is similar to and functions like the *call <prefixes>* (See above) except that it accepts spots generated by the **spotting call sign** and dumps all other spots.

Syntax: *accept/spot [0-9] by_dxcc <numbers>* This filters is based on DXCC entities and uses the DXCC entity number found by using the command *show/prefix <prefix>* and it accepts spots generated within the **spotting DXCC entity** and rejects other spots.

Syntax: *accept/spot [0-9] call_itu <numbers>* This filters is based on ITU regions and uses the ITU region number found by using the command *show/prefix <prefix>* (see above) except that it accepts spots generated by a **spotting call sign** within the ITU region and rejects all other spots.

Syntax: *accept/spot [0-9] call_zone <numbers>* This filters is based on CQ zones and uses the CQ zone number found by using the command *show/prefix <prefix>* (see above) except that it accepts spots generated by a **spotting call sign** within the CQ zone and rejects all other spots

Syntax: *accept/spot [0-9] by_state <state2-letter postal codes>* This filters is based on the state location of the spotting station found by using the command *show/usdb <call sign*> and accepts only those spots generated by stations from the states(s) specified rejecting all other spots.

Clear filters

A clear filter line will delete the slot number specified or all slots and consequently all filters that have been created by a user..

Syntax: *cleart/spots [0-9]* or
clear/spots all

Example:

clear/spots 2	This will clear any or both accept and reject spot filters in slot 2.
clear/spots all	This will clear each and every user spot filter - it will clear out all filters in all slots.

Note – if you just want to replace a spot filter, enter the rule again (with a line number) and it will overwrite the previous filter in that slot. If you forget the line number, it will overwrite the filter in slot 1 by default.

Some Practice Examples

The proceeding sections have discussed the basics of DXSpider filters. The following are some examples utilizing basic filters and some not so basic combination filters:

Lets say you only want to see any of those 6m, 2m, or 220 spots:

reject/spot 0 on uhf

As a good stand alone contest filter:

accept/spot on contesthf/<mode> where mode is either CW, SSB, or RTTY

Note: since a slot number is not included slot 1 is assumed.

It's a CW contest weekend so you don't want to see any WARC band or SSB spots:

accept/spots0 on contesthf/cw

It’s the same weekend , but you also don’t want to see any US or Canadian spots, or any rtty and data spots that are included in the CW portion of the bands. Any of the following will accomplish the same result:

```
reject/spot 0  not on contesthf/cw
reject/spot 1 on contesthf/data
reject/spot 2 call_dxcc w,ve
```

or;

```
accept/spot 0 on contesthf/cw and not (call_dxcc 226,197 or on contesthf/data)
```

or;

```
accept/spot 0 on contesthf/cw and not (call_dxcc w,ve or on contesthf/data)
```

The following two discussions are from the Administrator Manual and are a good “textbook” examples:

```
rej/spot on hf/cw
acc/spot on 0/30000
acc/spot 2 on 50000/1400000 and (by_zone 14,15,16 or call_zone 14,15,16)
```

Note that accept and reject can be abbreviated. Also, the first filter has not been specified with a number. This will automatically be assumed to be number 1. In this case, we have said to reject all HF spots in the CW section of the bands but accept all others at HF. Also accept anything in VHF and above that is spotted in or by operators in the zones 14, 15 and 16. Each filter slot actually has a 'reject' rule slot and an 'accept' rule slot. The reject rule slot is executed BEFORE the accept rule slot.

It was mentioned earlier that after a reject test that doesn't match, the default for following tests is 'accept', the reverse is true for 'accept'. In the example what happens is that the reject is executed first, any non hf/cw spot is passed to the accept line, which lets through everything else on HF. The next filter line lets through just VHF/UHF spots from EU.

If you set a reject filter like this ...

```
reject/spots on hf/cw
```

Then you will get everything *except* HF CW spots. You could make this single filter even more flexible. For example, if you are interested in IOTA and will work it on CW even though normally you are not interested in CW, then you could say ...

```
reject/spots on hf/cw and not info iota
```

But in that case you might only be interested in iota and say …

```
accept/spots not on hf/cw or info iota
```

which achieves exactly the same thing. Note that since slot numbers were not used slot1 is assumed.

Contacts

This Primer is a work in progress. Additional features and filter s are added from time to time by Dirk Koopman, G1THL, the developer behind DXCluster. So periodic revisions will be made to this document. If you have any questions, comments, or suggestions relative to this primer on spot filtering, please contact:

Jim Samuels, W3BG jimsam@comcast.net; or
Dave Hawes, N3RD (W3FRC Cluster SYSOP) dave.n3rd@comcast.net