



BGP AS-Path Filtering



Why use AS-Path Filtering?

- + Sometimes, the NLRI information is not known.
- + What IS known? The Autonomous System.
- + Scenarios for as-path filtering/matching
 - + Match on all prefixes from a specific neighboring AS.
 - + Match on all prefixes that originated in a specific AS.
 - + Match on all prefixes that passed through a specific AS.
 - + Many others.
- + AS-Path filters utilize regular expressions



Path Filters

- + Filtering based on AS-Path utilizes:
`(config)#ip as-path access-list <number> [permit | deny] regexp`
- + Applied against;
 - + A BGP neighbor using a “filter-list” (inbound or outbound)
 - + A Route-Map
 - `match as-path <number>`



AS-Path Interpretation

27	314	23	31	223
----	-----	----	----	-----

|27 314 23 31 223|
AS-path converted to string

ip as-path access-list 1 permit 31

String matched with regexp
Matching stops after first match



String Matching “Or”

- + The “pipe” symbol is a regular expression that means “either/or”

ip as-path access-list 1 permit 55|26

Which AS Paths does 55|26 match?

- 1 |613 817 2321 35|
- 2 |813 321 23**55** 41|
- 3 |4**26** 337 2316 92|
- 4 |523 547 2386 99|



String Matching: Delimiters

+ Example:

- + At the beginning and end of every AS-Path there are “invisible characters” you can’t see...but that you CAN match on.
 - ✓ This is what YOU see: 100 45 789 2000
 - ✓ This is what the ROUTER sees: **^100_45_789_2000\$**
- + **^100_** = Match on neighbor’s AS of “100”
- + **_2000\$** = Match on originating AS of “2000”



Path Filters

- + Example:

- + Deny all prefixes originated in AS 440.

- + Configuration:

```
router bgp 7
neighbor x.x.x.x remote-as 999
neighbor x.x.x.x filter-list 1 in
!
ip as-path access-list 1 deny _440$
ip as-path access-list 1 permit .*
```



String Matching: Ranges and Wildcards

- + A range of characters matches any single character in the range.

Examples: [5678] or [5-8,0]

- + A dot "." matches any single *character, space, or delimiting character*.

- + Example: 67.9 matches:

- ✓ 6709

- ✓ 6779

- ✓ 67 9



String Matching: Grouping

- + Parentheses can be used to group smaller regular expressions into larger groups of expressions.

Which AS Paths does `([1-4]17_121) | (6[7,8][0-9]_31)` match?

- 1 |213 317 1218 316 31|
- 2 |277 517 1218 316 31|
- 3 |667 317 3118 316 31|
- 4 |213 377 5679 316 31|



String Matching: Grouping

- + Parentheses can be used to group smaller regular expressions into larger groups of expressions.

Which AS Paths does `([1-4]17_121) | (6[7,8][0-9]_31)` match?

Answer:

- 1 |213 **317 1218** 316 31|
- 2 |277 517 1218 316 31|
- 3 |667 317 3118 316 31|
- 4 |213 377 **5679 316** 31|




String Matching: Repeating Characters

- + Repeating operators are placed to the right of characters/digits (called, “atoms”) you wish to match.

*	Matches zero-or-more atoms
?	Matches zero-or-one atom
+	Matches one-or-more atoms

An atom is a single character or a grouping.

Repeating Operators can be used to match atoms 

- The question mark (?) can be added to configuration lines by preceding it with Control-V (like you would if pasting something) followed by the question mark.

String Matching–Repeating Operators (2)

How do you match the following AS paths:

| 345 6 888 |

| 345 67 888 |

| 345 677 888 |

| 345 6777 888 |



- * = Zero-or-more

String Matching–Repeating Operators (2)

How do you match the following AS paths:

| 345 6 888 |

| 345 67 888 |

| 345 677 888 |

| 345 6777 888 |

Answer:

`^345_67*_888$`

7* = zero or more
instances of 7



- * = Zero-or-more

String Matching–Repeating Operators (3)

How do you match the following AS paths:

| 345 6 888 |

| 345 67 888 |

| 345 677 888 |

| 345 6777 888 |

Answer:

`^345_67*_888$`

7* = zero or more
instances of 7

What would be matched by the following?

`^345_(67)*_888$`



- * = Zero-or-more

String Matching–Repeating Operators (3)

How do you match the following AS paths:

| 345 6 888 |
| 345 67 888 |
| 345 677 888 |
| 345 6777 888 |

Answer:

$\wedge 345_67^*_888\$$

7* = zero or more
instances of 7

What would be matched by the following?

$\wedge 345_ (67)^*_ 888\$$

The above change would have matched...

| 345 888 |
| 345 67 888 |
| 345 6767 888 |
| 345 676767 888 |

(67)* = zero or more
instances of 67



- * = Zero-or-more

String Matching–Repeating Operators (4)

*	Matches zero-or-more atoms
?	Matches zero-or-one atom
+	Matches one-or-more atoms

How do you match the following AS paths:

| 345 888 |

| 345 67 888 |

But **not**...

| 345 6767 888 |

| 345 676767 888 |



- ? = Zero-or-one
- To enter a question mark, first type Ctrl-V (as if you were pasting something).

String Matching–Repeating Operators (4)

*	Matches zero-or-more atoms
?	Matches zero-or-one atom
+	Matches one-or-more atoms

How do you match the following AS paths:

| 345 888 |

| 345 67 888 |

But **not**...

| 345 6767 888 |

| 345 676767 888 |

Answer:

`^345_(67)?_888$`

`(67)?` = zero or one instances of 67



- ? = Zero-or-one
- To enter a question mark, first type Ctrl-V (as if you were pasting something).

String Matching–Repeating Operators (5)

*	Matches zero-or-more atoms
?	Matches zero-or-one atom
+	Matches one-or-more atoms

How do you match the following AS paths:

| 345 67 888 |

| 345 6767 888 |

But **not**...

| 345 888 |



- + = One or more

String Matching–Repeating Operators (5)

*	Matches zero-or-more atoms
?	Matches zero-or-one atom
+	Matches one-or-more atoms

How do you match the following AS paths:

| 345 67 888 |

| 345 6767 888 |

But **not**...

| 345 888 |

Answer:

`^345_(67)+_888$`

`(67)+` = one or more instances of 67



- + = One or more

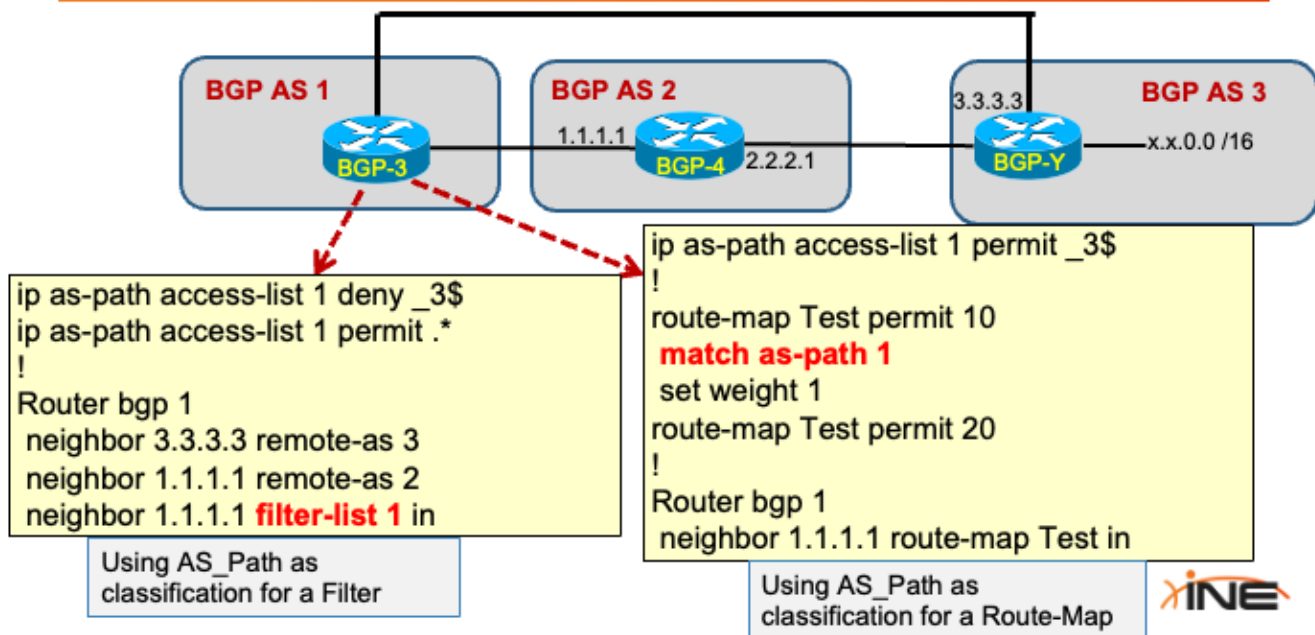
String Matching: Most Common

- + $\wedge 200_$ = Any route that was advertised from neighbor's AS of "200"
- + $_200\$$ = Any route that originated in AS "200"
- + $_200_$ = Any route that passed through AS "200"
- + $\wedge \$$ = Any route that originated in our own, local AS.
- + $\.*$ = Match everything



- The **[0-9]** pattern matches any digit.
- The **[0-9]+** pattern matches a sequence of digits.
- The **_[0-9]+_** pattern matches a complete number (the **_** characters match separators, including beginning or end of string).
- The **_([0-9]+)_** pattern matches a complete number and saves it for further reference.
- The **\1** pattern matches a previously saved match.
- The regular expression **_([0-9]+)_1_1_1_1_1_** therefore matches any AS path where a single AS number appears five or more times in a sequence.

Applying AS-Path Filters



- For inbound updates, Route-Maps are processed before Filter-Lists. So it is more scalable to enforce AS-Path filtering via a route-map than a Filter-List.

Editing AS-Path Access-Lists

- + Cisco IOS does not apply sequence values to as-path access-list statements.
- + New lines added to existing as-path acl's are added to the end of the existing acl.
- + There is no way to insert a new line in the middle of an existing as-path acl.
- + One can remove/delete individual lines of an as-path acl.



Monitoring AS-Path Filters

router#

```
show ip as-path-access-list [filter list]
```

- Displays one or all filter-lists

router#

```
show ip bgp regexp regular-expression
```

- Displays all routes in the BGP table permitted by the specified AS-path access-list

router#

```
show ip bgp filter-list access-list-number
```

- Displays all routes in the BGP table matching regular-expression in one or all filter-lists



