

# Access-Listen

## Inhaltsverzeichnis

1	EIGENSCHAFTEN VON ACCESS-LISTEN.....	2
1.1	Eigenschaften .....	2
1.2	Arten von Access-Listen .....	3
1.2.1	Standard Access Listen.....	3
1.2.2	Extended Access Listen .....	3
1.2.3	Named Access List.....	3
1.3	Arbeitsweise von Access-Listen.....	3
1.4	Paketfilter.....	4
1.5	Wichtige Bemerkungen .....	5
1.6	Designregeln.....	5
1.7	Regeln zur Implementierung .....	6
2	KONFIGURATION VON ACCESS LISTEN.....	6
2.1	Allgemeine Konfiguration .....	6
2.2	Bedeutung der ACL-Number.....	7
2.3	Interface Konfiguration: Paketfilter .....	8
2.4	Bitmaske .....	8
2.5	Beispiele von Bitnetmasken.....	9
2.6	Standard IP Access Listen.....	9
2.7	Beispiel: Einzelner Host .....	9
2.8	Sonderfall: Kommunikation mit Router.....	10
2.9	Extended IP Access List: allgemein .....	11
2.10	Extended IP Access List: IP .....	12
2.11	Reservierte Worte .....	12
2.12	Extended IP Access List: TCP, UDP.....	13
2.13	Extended IP Access List: ICMP .....	13
2.14	Beispiel 1: DMZ.....	14
2.15	Beispiel 2: Host zu Host – Kommunikation .....	15

2.16	Named IP Access List.....	15
2.17	Named IP Access List: Besonderheiten .....	16
2.18	Beispiel Standard Access List: DMZ .....	16
2.19	Beispiel Extended: Host zu Host - Kommunikation .....	17
2.20	Show ip access-lists.....	17
2.21	Show ip interface .....	18
3	ALTERNATIVE NULL INTERFACE.....	18
3.1	Alternative zu Access Listen: Null Interface .....	18
3.2	Design mit Access Liste .....	19

## 1 EIGENSCHAFTEN VON ACCESS-LISTEN

### 1.1 Eigenschaften

- „access lists“ (ACLs) dienen der Identifikation von Kommunikationsströmen auf Layer 3- (IP-Adressen) und Layer 4-Ebene (Port-Adressen). Hauptzweck ist das Filtern durch
  - Identifizierung anhand der Quelladresse.
  - Identifizierung anhand von Quell- und Zieladresse
  - Identifizierung anhand von Layer 4 Protokoll, z. B. TCP, UDP etc.
  - Identifizierung anhand von Adresse + Service (z. B. Portnummer bei TCP/UDP)
- Access-Listen gibt es für alle Layer 3- und Layer 4-Protokolle
- Vielseitiges Hilfsmittel bei der Konfiguration von CISCO-Routern. Typische Einsatzgebiete sind:
  - Paketfilter auf Basis von Access-Listen
  - Hilfsmittel beim Einsatz der „debug“- Befehle
  - Management von Routing Tables (z. B. Route Redistribution)
  - Zuordnung zu Queues für QoS
  - Zugriffsbeschränkung auf den Router
  - Identifikation von „Interesting Traffic“ bei Dial-Up Verbindungen

- Verwendung in Route-Maps
- .....

## 1.2 Arten von Access-Listen

### 1.2.1 Standard Access Listen

- Einfachste Art der Access Listen
- Identifikation von Traffic ausschließlich anhand der **Source Adresse**

### 1.2.2 Extended Access Listen

- Identifikation von Traffic möglich anhand von
  - Quell-Adresse
  - Ziel-Adresse
  - Layer 4-Protokoll
  - Quell-Port
  - Ziel-Port

### 1.2.3 Named Access List

- Alternative Schreibweise von Access Listen
- „Standard Access Listen“ und „Extended Access Listen“ werden anhand einer Nummer identifiziert, „Named Access Listen“ anhand eines **frei wählbaren Namens**.
- Vereinfachter Umgang, da nachträgliches Löschen einzelner Zeilen möglich.

## 1.3 Arbeitsweise von Access-Listen

- Access Listen arbeiten nach dem „**first match**“ Prinzip, d. h. sobald ein Eintrag zutrifft, wird danach gehandelt, auch wenn es einen spezifischeren Eintrag gibt. Dabei ist die **Reihenfolge bei Access Listen entscheidend**.
- Trifft keine Bedingung der Access Liste auf ein Paket zu, so wird das Paket verworfen. „**Implicit deny any any**“ in der letzten Zeile. Dieser Eintrag wird nicht angezeigt. Es gilt allgemein: „Alles was nicht explizit erlaubt ist, ist verboten.“

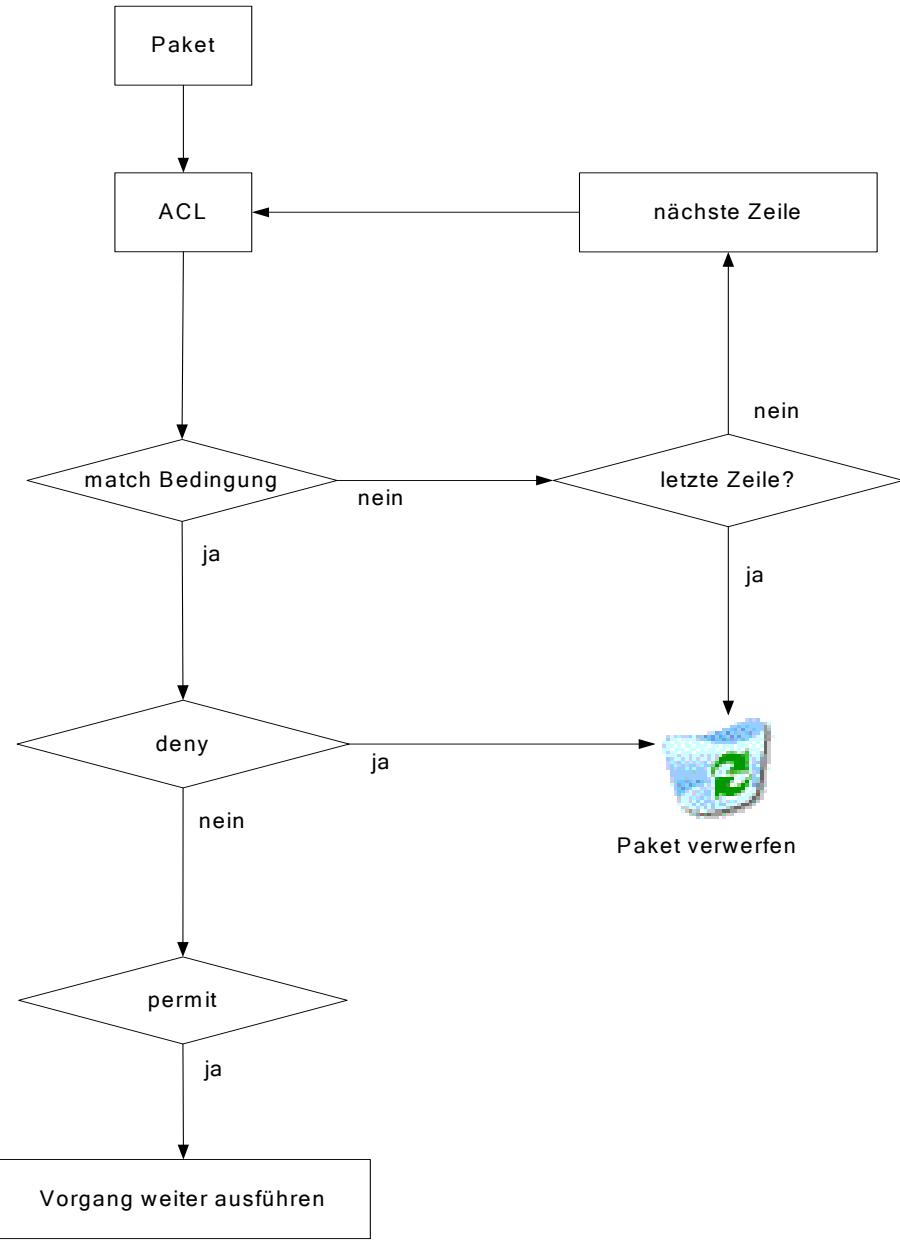


Bild: Arbeitsweise von Access-Listen

#### 1.4 Paketfilter

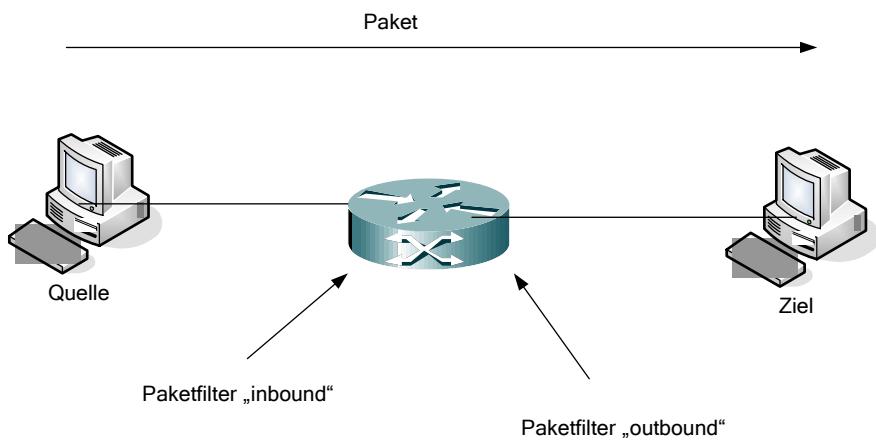


Bild: Platzierung von Paketfiltern

- **Access Listen** können als **Paketfilter** eingesetzt werden, d. h. es wird geprüft, ob ein Paket weitergeleitet oder verworfen werden soll (zuerst prüfen, dann ggf. weiterleiten).
- Inbound Access List
  - Vor Verarbeitung des Paketes durch den Routing Prozess wird geprüft, ob das Paket überhaupt weitergeleitet werden soll.
  - Da hier der Routingprozess u. U. gar nicht genutzt wird, ist eine „Inbound“-eingesetzte Access Liste effizienter als eine „Outbound Access List“.
- Outbound Access List
  - Nach Verarbeitung durch den Routingprozess und unmittelbar vor Versendung des Paketes aus dem entsprechenden Routerinterface wird geprüft, ob das Paket überhaupt weitergesendet werden soll (zuerst routen, dann prüfen, ob das Paket durch das Interface nach außen gelangen darf).
- Wird ein Paket verworfen, wird eine ICMP Message erzeugt und an den Sender zurückgeschickt.

## 1.5 Wichtige Bemerkungen

- Pro Protokoll und Interface nur eine Access Liste erlaubt
  - Einem Interface können mehrere Access Listen zugeordnet werden, jedoch nur für unterschiedliche Protokolle.
- Erst die Access Liste erstellen, dann dem Interface zuordnen:
  - „Implicit deny“ am Ende einer quasi leeren Access Liste hätte zur Folge, dass ansonsten jeglicher Verkehr über das Interface unmöglich ist.
- Access Listen filtern nur die Kommunikation über einen Router, nicht Pakete, die von dem Router selbst erzeugt werden, wie z. B. ICMP-Pakete, Routing Protokollupdate-Pakete.
- Eine Access Liste kann mehrfach verwendet werden.

## 1.6 Designregeln

- Allgemeine Designregeln
  - Beachte: „first match“- Prinzip

- Je spezifischer die Information, desto weiter oben (vorne) in der Liste sollte sie angesiedelt sein.
- Zeilen in der Liste werden in der Reihenfolge angeordnet, in der sie eingegeben werden.
- Nachträgliches Einfügen oder Löschen ist nicht möglich (Ausnahme: Löschen einzelner Zeilen in Named Access Lists)
  
- Implicit deny any
  - Nicht explizit erlaubte Kommunikation ist verboten.
  - Zu Erinnerung und für das „logging“ kann das „implicit deny“ explizit angegeben werden.
    - In diesem Fall wird es in den entsprechenden Show Kommandos mit angezeigt.
  - Je nach Design muss die Access List mit einem „permit any“ abgeschlossen werden.
    - In diesem Fall hat das „Implicit deny“ keine Bedeutung mehr.

## 1.7 Regeln zur Implementierung

- Standard Access Listen sollen zielnah implementiert werden, Extended Access Listen sollen quellnah platziert werden.
- Unnötiger Traffic sollte vor dem Backbone und vor langsamen Leitungen (WAN-Verbindungen, wie z. B. ISDN) herausgefiltert werden.
- Router mit hoher CPU Last nicht auch noch im Access Listen belasten.
- Genau prüfen, ob outbound oder inbound.
- Access Listen so kurz wie möglich halten.
- Am häufigsten zutreffende Statements möglichst weit oben in der Liste platzieren.
- Implementierte Access Listen gut dokumentieren

## 2 KONFIGURATION VON ACCESS LISTEN

### 2.1 Allgemeine Konfiguration

```
Router(config)#[no] access-list acl-number {permit | deny} // Test-Bedingung
```

**no** mittels des “no” Befehl kann eine ganze Access List gelöscht werden.

<i>acl-number</i>	Eindeutige Nummer der Access Liste
<i>permit</i>	Trifft die <i>Test-Bedingung</i> zu, wird das Paket zur Weiterverarbeitung an den nächsten Prozess übergeben
<i>deny</i>	Trifft die <i>Test-Bedingung</i> zu wird das Paket verworfen.
<i>Test-Bedingung</i>	Eintrag ist abhängig von der Art der Access Liste (Standard, Extended)

- Access Listen werden im globalen Konfigurationsmodus angelegt.
- Änderungen/Löschen von Access Listen wirken sofort nach deren Eingabe.
- Hinweis:  
Es ist „sicherer“ Access Listen in einem Editor zu schreiben und anschließend via TFTP oder „Copy and Paste“ auf den Router zu übertragen.

## 2.2 Bedeutung der ACL-Number

Protokoll	Access list Number
<b>Standard IP</b>	<b>1 bis 99</b>
	<b>1300 bis 1999</b>
<b>Extended IP</b>	<b>100 bis 199</b>
	<b>2000 bis 2699</b>
Ethernet type code	200 bis 299
Source-route bridging (protocol type)	200 bis 299
Transparent bridging (protocol type)	200 bis 299
DECnet and extended DECnet	300 bis 399
XNS	400 bis 499
Extended XNS	500 bis 599
Apple Talk	500 bis 599
Ethernet address	700 bis 799
Transparent bridging (vendor code)	700 bis 799
Source-route bridging (vendor code)	700 bis 799
IPX	800 bis 899
Extended IPX	900 bis 999
IPX SAP	1000 bis 1099
Extended transparent bridging	1100 bis 1199

Bild: Bedeutung der ACL-Nummern

- Die *acl number* ist keine Zeilenummer sondern die Nummer der Access Liste.
- Einträge mit der selben Nummer
  - gehören zusammen
  - werden in der eingegebenen Reihenfolge später abgearbeitet
- Die Nummer definiert die Art (Standard oder Extended) und das Protokoll der Access Liste

## 2.3 Interface Konfiguration: Paketfilter

```
Router(config-if)# [no] protocol access-group acl-number {in / out}
```

<i>no</i>	mittels des „no“ Befehls wird die Zuordnung gelöscht
<i>protocol</i>	Protokoll für das dem Interface eine Access Liste zugeordnet werden soll z. B. IP, IPX etc.
<i>acl-number</i>	Eindeutige Nummer der zuzuordnenden Access Liste
<i>in</i>	Die Access Liste wird <i>inbound</i> verarbeitet
<i>out</i>	Die Access Liste wird <i>outbound</i> verarbeitet

## 2.4 Bitmaske

Bit								Mask
128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1	3
0	0	0	0	0	1	1	1	7
0	0	0	0	1	1	1	1	15
0	0	0	1	1	1	1	1	31
0	0	1	1	1	1	1	1	63
0	1	1	1	1	1	1	1	127
1	1	1	1	1	1	1	1	225

Bild: Bitmasken

- Bitmasken von Access Listen werden quasi invers zu Bitmasken von Subnetzmasken gebildet.
- Eine „0“ in der Bitnetzmaske bedeutet „match“, eine „1“ in der Bitnetzmaske bedeutet „don’t care“.

## 2.5 Beispiele von Bitnetmasken

	Maske	Bedeutung	Alternative Schreibweise
10.0.0.0	0.255.255.255	Class A	
130.130.0.0	0.0.255.255	Class B	
200.200.200.0	0.0.0.255	Class C	
150.150.150.0	0.0.7.255	Netz 150.150.150.0 mit Subnetzmaske	
255.255.255.255	0.0.0.0	Lokaler Broadcast	
10.10.10.10	0.0.0.0	Einzelner Host	Host 10.10.10.10
0.0.0.0	255.255.255.255	Jede Adresse (alle)	any

Bild: Beispiele von Bitnetmasken

- Rechenregel:

„Klassische“ Subnetzmaske bilden, byteweise von „255.255.255.255“ subtrahieren

Beispiel:

Subnetzmaske 255.255.255.0

$$\begin{array}{r}
 \text{Bitmaske:} \quad 255 \ 255 \ 255 \ 255 \\
 - \underline{255 \ 255 \ 248 \ 0} \\
 \hline
 0 \quad 0 \quad 7 \quad 255
 \end{array}$$

## 2.6 Standard IP Access Listen

```
Router(config)# [no] access-list acl-number {permit | deny} source-adress [mask]
```

- no mittels des „no“ befehl kann eine ganze Access List gelöscht werden
- acl-number 1-99 oder 1300 bis 1999, eindeutige Nummer der Access Liste
- permit Trifft die source-address zu, wird das Paket zur Weiterverarbeitung an den nächsten Prozess übergeben
- deny Trifft die source-adress wird das Paket verworfen
- mask optionale Bitmaske. Defaultmäßig, wenn nicht angegeben, ist diese 0.0.0.0, also die Host Adresse selbst.

- Standard IP Access Listen filtern nur Quelladressen

## 2.7 Beispiel: Einzelner Host

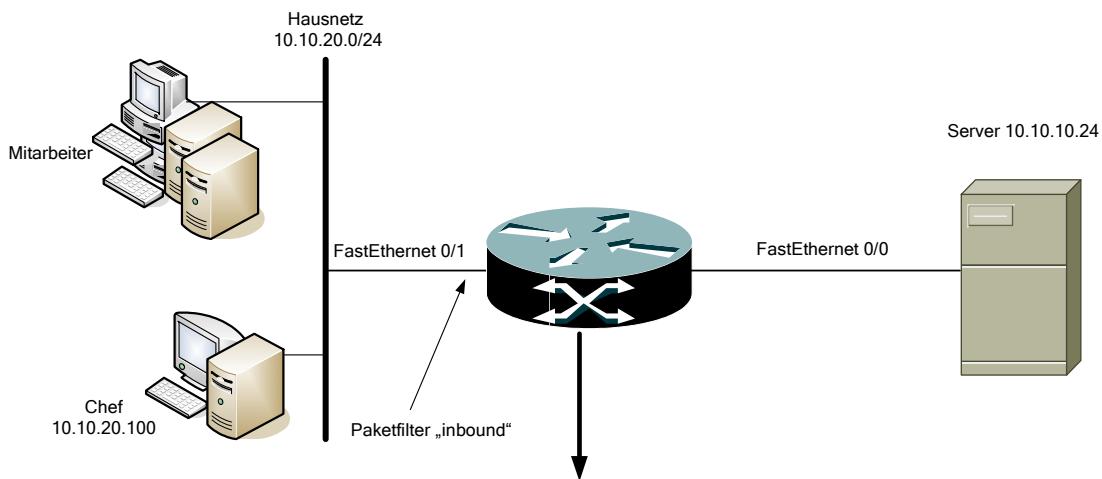
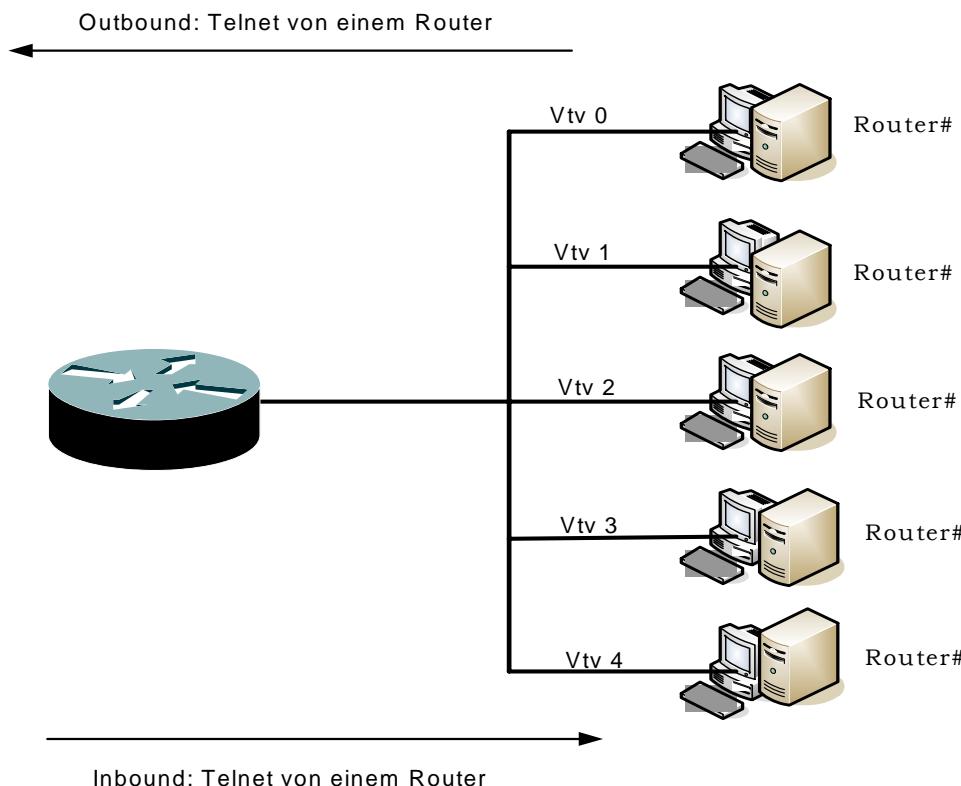


Bild: Beispiel einer Standard ACL

```
R4(config)# access-list 10 deny 10.10.20.100 0.0.0.0
R4(config)# access-list 10 permit 10.10.20.0 0.0.0.255
R4(config)# interface FastEthernet 0/1
R4(config-if)# ip access-group 10 in
```

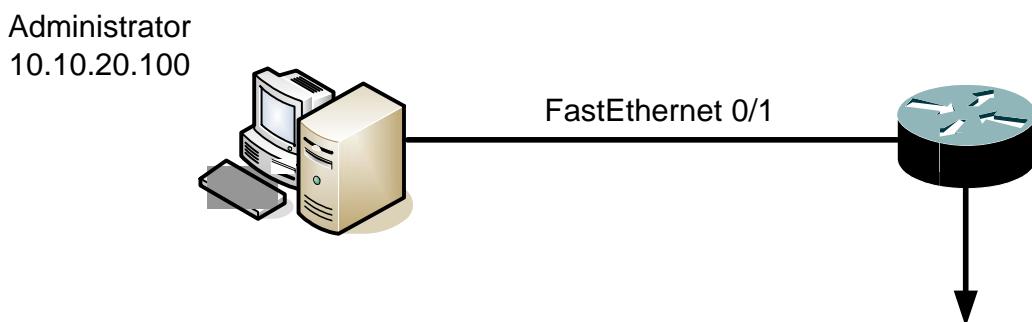
- Kommunikation des Hosts 10.10.20.100 mit dem Netz 10.10.10.0/24 wird verhindert (deny).
- Jegliche weitere Kommunikation zwischen Netz 10.10.10.0 und 10.10.20.0 ist erlaubt.

## 2.8 Sonderfall: Kommunikation mit Router



## Bild: Kommunikation mit Router

- Vom Router selbst generierte Pakete können nicht gefiltert werden  
Folge: Outbound Access Listen sind nicht nutzbar.
- Es soll nur die Kommunikation mit dem Router nicht über ihn hinweg verhindert/eingeschränkt werden.  
Folge: Inbound Access Listen sind nicht nutzbar
- Lösung: „Virtuelle Lines“
  - Inbound: Aufbau einer Telnet-Session mit dem Router
  - Outbound: Aufbau einer Telnet-Session von einem Router aus



```
R4(config)# access-list 10 permit 10.10.20.100 0.0.0.0
```

```
R4(config)# line vty 0 4
```

```
R4(config-if)# access-class 10 in
```

- Folge: Nur Host 10.10.20.100 kann eine Telnet Verbindung zum Router aufbauen
- Befehl zum binden der Access-Liste an die Lines:  

```
R4(config-if)# access-class acl-number {in | out}
```

## 2.9 Extended IP Access List: allgemein

```
Router(config)#[no] access-list acl-number {permit | deny} protocol
          source-address [mask] [operator port]
          dest-address [mask] [operator port]
          [established] [log]
```

**acl-number** 100-199 oder 2000 bis 2699, eindeutige Nummer der Access Liste

**permit** Trifft die folgende Bedingung wird das Paket zur Weiterverarbeitung an den nächsten Prozess übergeben.

**deny** Trifft die folgende Bedingung zu, wird das Paket verworfen.

<i>protocol</i>	Layer 3 bzw. 4 Protokoll, für das die Liste angelegt ist. <b>Das Protokoll bestimmt die weitere Listenform.</b>
<i>source-address</i>	Quelladresse IP Pakets
<i>mask</i>	Bitmaske
<i>dest-address</i>	Zieladresse des IP Pakets
<i>mask</i>	Bitmaske
<i>operator-portport</i>	Angabe für UDP- bzw. TCP-Filter Operator kann sein: lt „less than“: weniger, kleiner, < gt „greater than“: mehr, größer, > eq „equal to“: gleich, = neq „not equal to“: ungleich, !=
<i>established</i>	optionale nur „inbound TCP“ Listen; lässt bereits etablierte Kommunikation durch (ACK Bit im TCP Header ist gesetzt)
<i>log</i>	erzeugt logging Messages, wenn der Eintrag einen Match erzeugt

## 2.10 Extended IP Access List: IP

```
Router(config)#access-list acl-number {permit | deny} ip
          source-address [mask]
          dest-address [mask] [log]
```

<i>source-address</i>	Quelladresse des IP Pakets
<i>mask</i>	Bitmaske
<i>dest-address</i>	Zieladresse des IP Pakets
<i>log</i>	erzeugt logging Messages, wenn der Eintrag einen Match erzeugt

- Beispiel:

```
Router(config)#access-list 100 deny ip 10.10.10.0 0.0.0.255 10.10.20.0 0.0.0.255
```

Alle Pakete, die von 10.10.10.0/24 nach 10.10.20.0/24 laufen, werden verworfen

## 2.11 Reservierte Worte

- Filtern eines einzelnen Hosts durch Angabe des Schlüsselwortes „**host**“ von der IP **Nummer**:

```
Router(config)#access-list 100 permit ip host 10.10.10.100 10.10.20.0 0.0.0.255
```

Wirkung: Host 10.10.10.100 darf mit Netz 10.10.20.0/24 eine Verbindung aufbauen.

- Verkürzte Schreibweise für „alle Netze“ ist „any“

```
Router(config)#access-list 100 permit ip any 10.10.20.0 0.0.0.255
```

Wirkung: Jedes Netz darf mit Netz 10.10.20.0 0.0.0.255 eine Verbindung aufnehmen.

## 2.12 Extended IP Access List: TCP, UDP

```
Router(config)#access-list acl-number {permit | deny} {tcp | udp}
    Source-address [mask][operator]
    dest-address [mask][operator port]
    [established][log]
```

*operator port* Portangabe für UDP, TCP Filter;

Operator kann sein:

lt „less than“ weniger, kleiner, <  
gt „greater than“ mehr, größer, >  
eq „equal to“ gleich, =  
neq “not equal to” ungleich, !=

Anstatt der Portnummer können häufig auch Portnamen eingegeben werden

*established* optionale nur in „inbound TCP“ Listen; lässt bereits etablierte Kommunikation durch (ACK Bit im TCP Header ist gesetzt);

*established* nur mit TCP möglich, da UDP kein ACK-Bit aufweist

*log* erzeugt logging Messages, wenn der Eintrag einen Match erzeugt

- Beispiel:

```
Router(config)#access-list 100 deny tcp 10.10.10.0 0.0.0.255
    10.10.20.0 0.0.0.255 eq 23
```

## 2.13 Extended IP Access List: ICMP

```
Router(config)#access-list acl-number {permit|deny} icmp
    Source-address [mask]
    dest-address [mask]
    [icmp-type [icmp-code]]| icmp-message] [log]
```

- icmp-type* Zahl zwischen 0 und 255 für ICMP Typ.
- icmp-code* Zahl zwischen 0 und 255 fpr ICMP Code (typspezifisch)
- icmp-message* Name für ICMP Message.  
Beispiele: echo, echo-reply, redirect, source-quench, time-exceeded

- Beispiel:

```
Router(config)#access-list 100 deny icmp 10.10.10.0 0.0.0.255 10.10.20.0 0.0.0.255 echo
```

Alle Ping-Pakete, die von 10.10.10.0/24 nach 10.10.20.0/24 laufen, werden verworfen.

## 2.14 Beispiel 1: DMZ

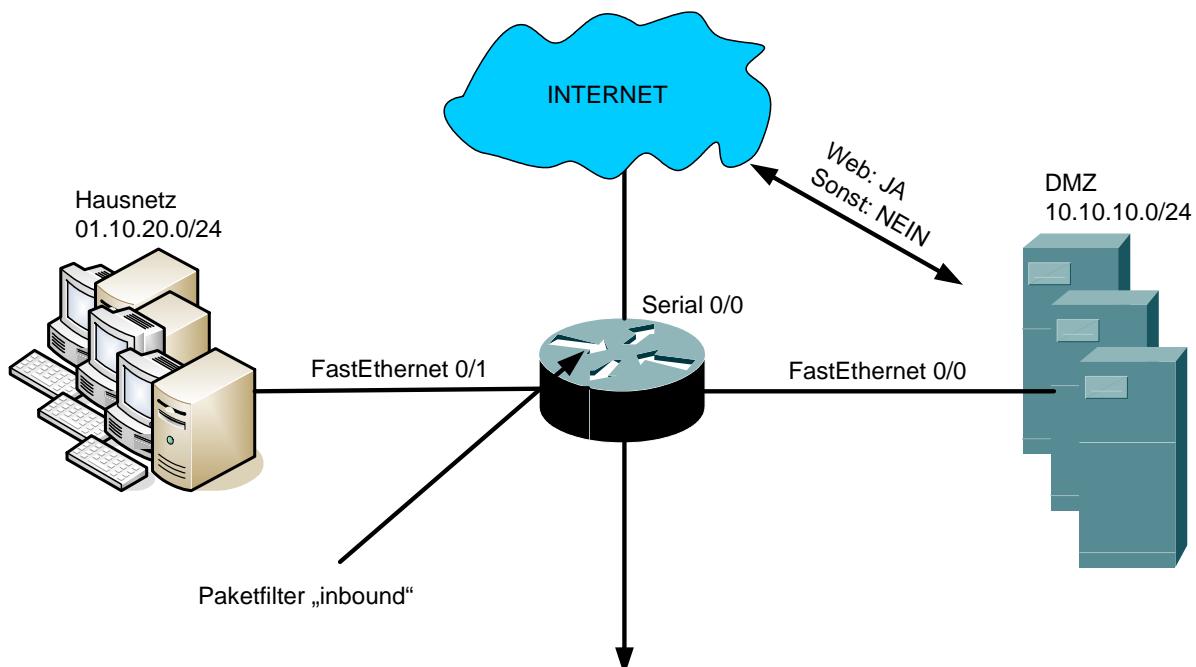


Bild: DMZ

```
R4(config)# access-list 100 permit tcp any 10.10.10.0 0.0.0.255 eq www
R4(config)# access-list 100 deny ip any any log
R4(config)# interface Serial 0/0
R4(config)# ip access-group 100 in
```

- Wirkung: Internet-User können auf port 80 (www) in der DMZ zugreifen, d. h. Web-zugriffe sind erlaubt
- Jegliche andere Kommunikation ist verboten und wird protokolliert.

## 2.15 Beispiel 2: Host zu Host – Kommunikation

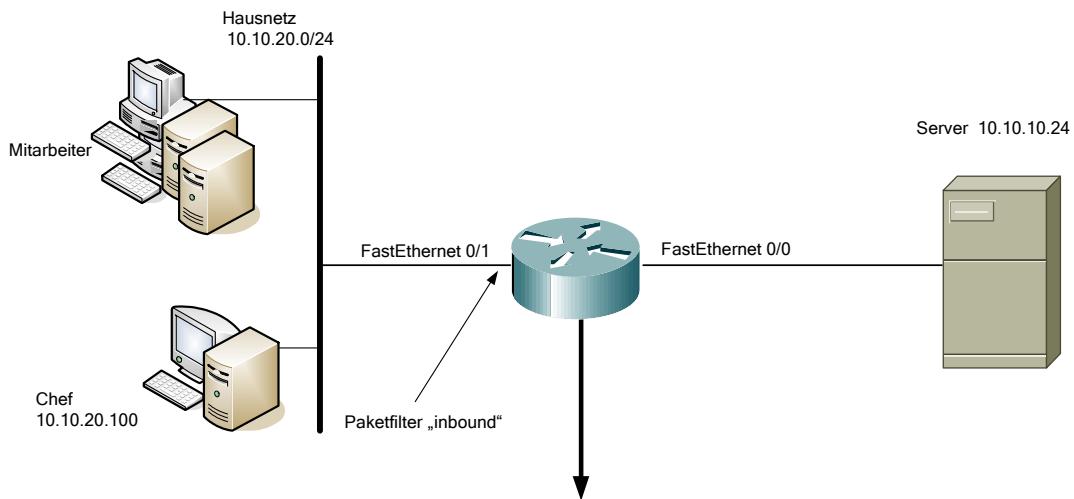


Bild: Host to Host - Kommunikation

```
R4(config)# access-list 101 deny ip host 10.10.20.100 host 10.10.10.24
R4(config)# access-list 101 permit ip any any
R4(config)# interface FastEthernet 0/1
R4(config-if)# if access-group 101 in
```

- Kommunikation des Hosts 10.10.20.100 mit dem Host 10.10.10.24 wird verhindert (deny).
- Jegliche weitere Kommunikation zwischen Netz 10.10.10.0 und 10.10.20.0 ist erlaubt

## 2.16 Named IP Access List

```
Router(config)#ip access-list {standard|extended} name
Router(config-std-nacl)# {permit|deny} {Test-Bedingung}
```

<i>name</i>	Frei wählbarer Name der Access Liste
<i>standard  extended</i>	Art der Access Liste, äquivalent zur Nummerierung
<i>permit</i>	Trifft die Test-Bedingung zu, wird das Paket zur Weiterverarbeitung an den nächsten Prozess übergeben
<i>deny</i>	Trifft die Test-Bedingung zu, wird das Paket verworfen
<i>Test-Bedingung</i>	Form der Test-Bedingung ist von der Art der Access Liste abhängig

- Ab IOS 11.2 verfügbar
- Named Access Listen können genutzt werden:

- Wenn die Anzahl der Access Listen die Anzahl der Access Listen Nummern überschreitet.
- Wenn „sprechende“ Namen gewünscht sind

## 2.17 Named IP Access List: Besonderheiten

- Nach Anlage der Access Liste wechselt man in den Named Access List Konfigurationsmodus (nacl)
- Je nach Art der Access Liste ändert sich das Router-Prompt:  
Standard Access Liste

```
Router(config-std-nacl)#[/pre]

```

Named Access Liste

```
Router(config-ext-nacl)#[/pre]

```

- Im Konfigurationsmodus einer Access Liste können einzelne Zeilen gelöscht werden.
- Neue Zeilen werden immer an das Ende einer Access Liste angefügt

## 2.18 Beispiel Standard Access List: DMZ

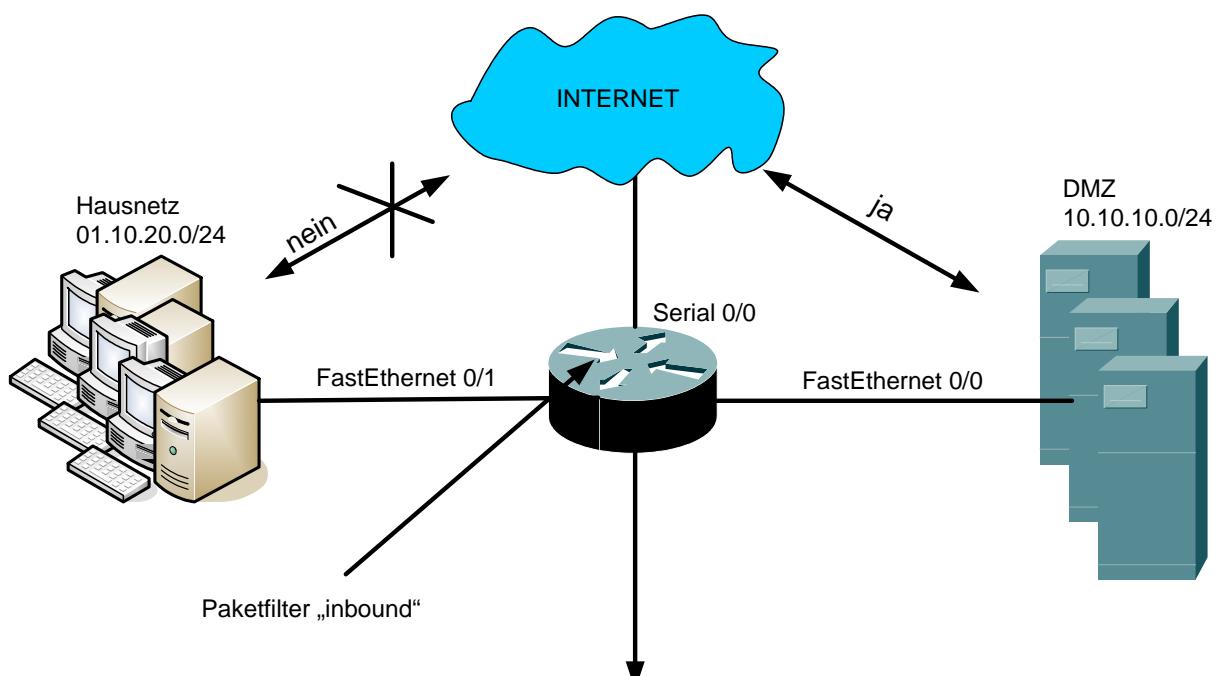


Bild: Standard Access List

```
R4(config)#ip access-list standard Std-Liste
R4(config-std-nacl)#permit 10.10.10.0 0.0.0.255
[/pre]

```

```
R4(config-std-nacl)#exit
```

```
R4(config)#interface FastEthernet 0/1  
R4(config-if)#ip access-group Std-Liste out
```

Analog zum Beispiel 1 der Standard Access Liste

## 2.19 Beispiel Extended: Host zu Host - Kommunikation

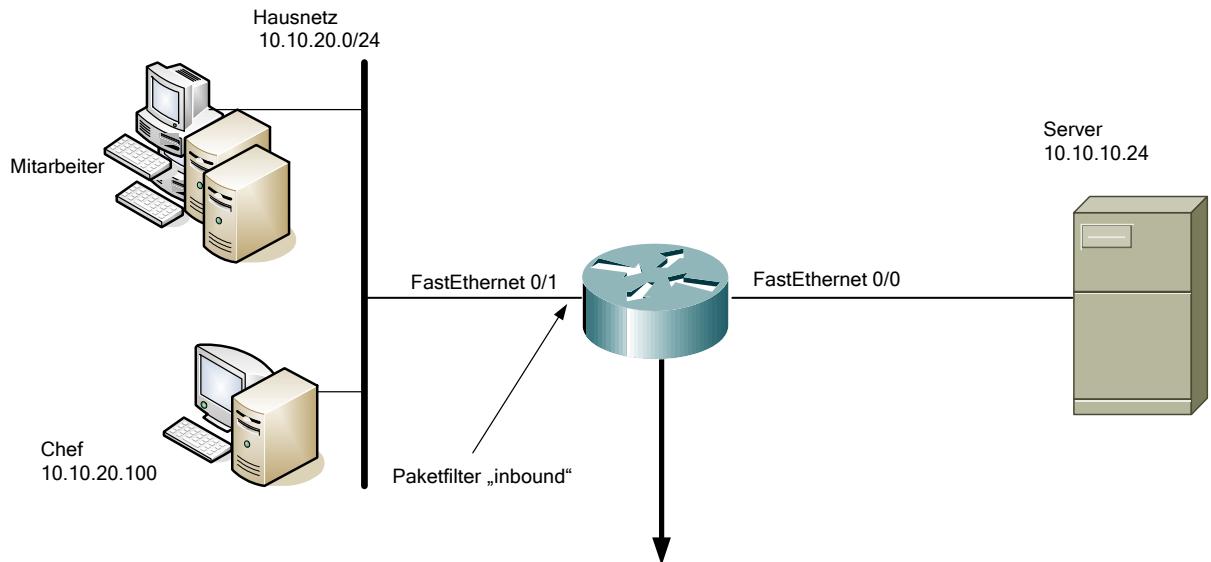


Bild: Extended Access List

```
R4(config)#ip access-list extended NoChef  
R4(config-ext-nacl)#deny ip host 10.10.20.100 host 10.10.10.24  
R4(config-ext-nacl)#permit ip any any  
R4(config-ext-nacl)#exit
```

```
R4(config)#interface FastEthernet 0/1  
R4(config-if)#ip access-group NoChef in
```

Analog zum Beispiel 2 Extended Access List

## 2.20 Show ip access-lists

```
R4#sh ip access-lists  
Standard IP access list 1  
    permit 10.10.10.0, wildcard bits 0.0.0.255  
Extended IP access list 100  
    deny icmp any any log
```

```
    permit ip any any
Extended IP access lost NoPing
    deny icmp any any echo log (5 matches)
    permit ip any any
```

- Gibt alle Access Listen wieder
- „(5 matches)“ zeigt an, wie oft ein Eintrag zum Tragen gekommen ist
- Implicit deny wird nicht angezeigt
- Optional kann am Ende des Befehls noch die Liste angegeben werden
  - Name
  - Nummer

## 2.21 Show ip interface

```
R4#sh ip int f 0/1
FastEthernet0/1 is up, line protocol is up
    Internet address is 172.114.1.4/24
    Broadcast address is 255.255.255.255
    Address determined by non-volatile memory
    MTU is 1500 bytes
Helper address is not set
    Directed broadcast forwarding is disabled
    Outgoing access list is 1
    Inbound access list is not set
[...]
```

- Zeigt die einem Interface zugeordnete Access Listen Nummern/Namen an.
- Unterscheidet „Inbound“ (Ingoing) und „Outbound“ (Outgoing)

## 3 ALTERNATIVE NULL INTERFACE

### 3.1 Alternative zu Access Listen: Null Interface

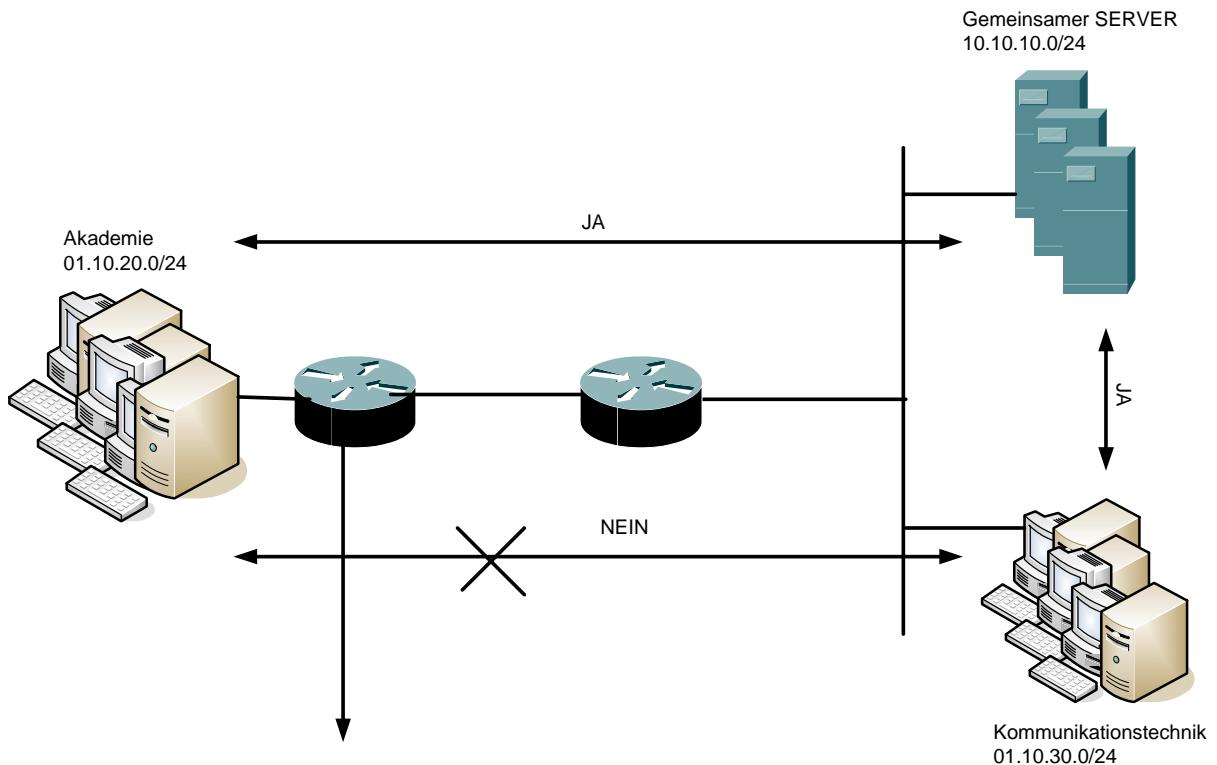


Bild: Null Interface

```
R4(config)#ip route 10.10.30.0 255.255.255.0 null 0
```

- Das „Null Interface“ ist ein virtuelles Interface.
- Pakete, die in dieses Interface geroutet werden, werden verworfen („discarded“).
- Es wird keine ICMP Meldung erzeugt.
- Vorteil: geringer CPU Overhead.

### 3.2 Design mit Access Liste

