CHAPTER 11

Configuring Objects

Objects are reusable components for use in your configuration. They can be defined and used in adaptive security appliance configurations in the place of inline IP addresses. Objects make it easy to maintain your configurations because you can modify an object in one place and have it be reflected in all other places that are referencing it. Without objects you would have to modify the parameters for every feature when required, instead of just once. For example, if a network object defines an IP address and subnet mask, and you want to change the address, you only need to change it in the object definition, not in every feature that refers to that IP address.

This chapter describes how to configure objects, and it includes the following sections:

Configuring Objects and Groups

This section includes the following topics:

Information About Objects and Groups

The adaptive security appliance supports objects and object groups. You can attach or detach objects from one or more object groups when needed, ensuring that the objects are not duplicated but can be re-used wherever needed.

This section includes the following topics:

Information About Objects

Objects are created in and used by the adaptive security appliance in the place of an inline IP address in any given configuration. You can define an object with a particular IP address and netmask pair or a protocol (and, optionally, a port) and use this object in several configurations. The advantage is that whenever you want to modify the configurations created to this IP address or protocol, you do not need to modify all rules in the running configuration. You can modify the object, and then the change automatically applies to all rules that use the specified object. You can configure two types of objects: network objects and service objects. These objects can be used in Network Address Translation (NAT), access lists, and object groups.

Information About Object Groups

By grouping like objects together, you can use the object group in an ACE instead of having to enter an ACE for each object separately. You can create the following types of object groups:

- Protocol
- Network
- Service
- ICMP type

For example, consider the following three object groups:

- MyServices—Includes the TCP and UDP port numbers of the service requests that are allowed access to the internal network.
- TrustedHosts—Includes the host and network addresses allowed access to the greatest range of services and servers.
- PublicServers—Includes the host addresses of servers to which the greatest access is provided.

After creating these groups, you could use a single ACE to allow trusted hosts to make specific service requests to a group of public servers.

You can also nest object groups in other object groups.

Licensing Requirements for Objects and Groups

The following table shows the licensing requirements for this feature:

Model	License Requirement
All models	Base License.

Guidelines and Limitations for Objects and Groups

This section includes the guidelines and limitations for this feature.

Context Mode Guidelines

Supported in single and multiple context mode.

Firewall Mode Guidelines

Supported in routed and transparent firewall modes.

IPv6 Guidelines

Supports IPv6, with limitations. (See the Additional Guidelines and Limitations, page 11-3.)

Additional Guidelines and Limitations

The following guidelines and limitations apply to object groups:

- Objects and object groups share the same name space.
- Object groups must have unique names. While you might want to create a network object group named “Engineering” and a service object group named “Engineering,” you need to add an identifier (or “tag”) to the end of at least one object group name to make it unique. For example, you can use the names “Engineering_admins” and “Engineering_hosts” to make the object group names unique and to aid in identification.
- You cannot remove an object group or make an object group empty if it is used in a command.
- The adaptive security appliance does not support IPv6 nested object groups, so you cannot group an object with IPv6 entities under another IPv6 object group.

Configuring Objects

This section includes the following topics:

Configuring a Network Object

A network object contains a single IP address/mask pair. Network objects can be of three types: host, subnet, or range.

You can also configure auto NAT as part of the object definition; see Chapter 28, “Configuring Network Object NAT,” for more information.

Detailed Steps

	Command	Purpose
Step 1	**object network** *obj_name* **Example:** `hostname(config)# object-network OBJECT1`	Creates a new network object. The *obj_name* is a text string up to 64 characters in length and can be any combination of letters, digits, and the following characters: • underscore "_" • dash "-" • period "." The prompt changes to network object configuration mode.
Step 2	{**host** *ip_addr* | **subnet** *net_addr net_mask* | **range** *ip_addr_1 ip_addr_2*} **Example:** `hostname(config-network-object)# host 10.2.2.2`	Assigns the IP address to the named object. You can configure a host address, a subnet, or a range of addresses.
Step 3	**description** *text* **Example:** `hostname(config-network-object)# description Engineering Network`	Adds a description to the object.

Examples

To create a network object, enter the following commands:

```
hostname (config)# object network OBJECT1
hostname (config-network-object)# host 10.2.2.2
```

Configuring a Service Object

A service object contains a protocol and optional source and/or destination port.

Detailed Steps

	Command	Purpose
Step 1	**object service** *obj_name* **Example:** `hostname(config)# object-service SERVOBJECT1`	Creates a new service object. The *obj_name* is a text string up to 64 characters in length and can be any combination of letters, digits, and the following characters: • underscore "_" • dash "-" • period "." The prompt changes to service object configuration mode.
Step 2	**service** {*protocol* | **icmp** *icmp-type* | **icmp6** *icmp6-type* | {**tcp** | **udp**} [**source** *operator port*] [**destination** *operator port*]} **Example:** `hostname(config-service-object)# service tcp source eq www destination eq ssh`	Creates a service object for the source mapped address. The *protocol* argument specifies an IP protocol name or number. The **icmp**, **tcp**, or **udp** keywords specify that this service object is for either the ICMP, TCP, or UDP protocol. The *icmp-type* argument names the ICMP type. The **icmp6** keyword specifies that the service type is for ICMP version 6 connections. The *icmp6-type* argument names the ICMP version 6 type. The **source** keyword specifies the source port. The **destination** keyword specifies the destination port. The *operator port* argument specifies a single port/code value that supports configuring the port for the protocol. You can specify "eq," "neq," "lt," "gt," and "range" when configuring a port for TCP or UDP. The "range" operator lists the beginning port and ending port.

Example

To create a service object, enter the following commands:

```
hostname (config)# object service SERVOBJECT1
hostname (config-service-object)# service tcp source eq www destination eq ssh
```

Configuring Object Groups

This section includes the following topics:

Adding a Protocol Object Group

To add or change a protocol object group, perform the steps in this section. After you add the group, you can add more objects as required by following this procedure again for the same group name and specifying additional objects. You do not need to reenter existing objects; the commands you already set remain in place unless you remove them with the **no** form of the command.

Detailed Steps

	Command	Purpose
Step 1	**object-group protocol** *obj_grp_id* **Example:** `hostname(config)# object-group protocol tcp_udp_icmp`	Adds a protocol group. The *obj_grp_id* is a text string up to 64 characters in length and can be any combination of letters, digits, and the following characters: • underscore "_" • dash "-" • period "." The prompt changes to protocol configuration mode.
Step 2	**description** *text* **Example:** `hostname(config-protocol)# description New Group`	(Optional) Adds a description. The description can be up to 200 characters.
Step 3	**protocol-object** *protocol* **Example:** `hostname(config-protocol)# protocol-object tcp`	Defines the protocols in the group. Enter the command for each protocol. The protocol is the numeric identifier of the specified IP protocol (1 to 254) or a keyword identifier (for example, **icmp**, **tcp**, or **udp**). To include all IP protocols, use the keyword **ip**. For a list of protocols that you can specify, see the "Protocols and Applications" section on page B-11.

Example

To create a protocol group for TCP, UDP, and ICMP, enter the following commands:

```
hostname (config)# object-group protocol tcp_udp_icmp
hostname (config-protocol)# protocol-object tcp
hostname (config-protocol)# protocol-object udp
```

```
hostname (config-protocol)# protocol-object icmp
```

Adding a Network Object Group

A network object group supports IPv4 and IPv6 addresses.

To add or change a network object group, perform the steps in this section. After you add the group, you can add more objects as required by following this procedure again for the same group name and specifying additional objects. You do not need to reenter existing objects; the commands you already set remain in place unless you remove them with the no form of the command.

Detailed Steps

	Command	Purpose
Step 1	**object-group network** *grp_id* **Example:** `hostname(config)# object-group network admins`	Adds a network group. The *grp_id* is a text string up to 64 characters in length and can be any combination of letters, digits, and the following characters: • underscore "_" • dash "-" • period "." The prompt changes to protocol configuration mode.
Step 2	**description** *text* **Example:** `hostname(config-network)# Administrator Addresses`	(Optional) Adds a description. The description can be up to 200 characters.
Step 3	**network-object** {**object** *name* | host ip_address | ip_address mask} **Example:** `hostname(config-network)# network-object host 10.2.2.4`	The **object** keyword adds an additional object to the network object group. Defines the networks in the group. Enter the command for each network or address.

Example

To create a network group that includes the IP addresses of three administrators, enter the following commands:

```
hostname (config)# object-group network admins
hostname (config-protocol)# description Administrator Addresses
hostname (config-protocol)# network-object host 10.2.2.4
hostname (config-protocol)# network-object host 10.2.2.78
hostname (config-protocol)# network-object host 10.2.2.34
```

Adding a Service Object Group

To add or change a service object group, perform the steps in this section. After you add the group, you can add more objects as required by following this procedure again for the same group name and specifying additional objects. You do not need to reenter existing objects; the commands you already set remain in place unless you remove them with the **no** form of the command.

Detailed Steps

	Command	Purpose
Step 1	**object-group service** *grp_id* {**tcp** | **udp** | **tcp-udp**} **Example:** `hostname(config)# object-group service services1 tcp-udp`	Adds a service group. The **object** keyword adds an additional object to the service object group. The *grp_id* is a text string up to 64 characters in length and can be any combination of letters, digits, and the following characters: • underscore "_" • dash "-" • period "." Specify the protocol for the services (ports) you want to add with either the **tcp**, **udp**, or **tcp-udp** keywords. Enter the **tcp-udp** keyword if your service uses both TCP and UDP with the same port number, for example, DNS (port53). The prompt changes to service configuration mode.
Step 2	**description** *text* **Example:** `hostname(config-service)# description DNS Group`	(Optional) Adds a description. The description can be up to 200 characters.
Step 3	**port-object** {**eq** *port* | **range** *begin_port end_port*} **Example:** `hostname(config-service)# port-object eq domain`	Defines the ports in the group. Enter the command for each port or range of ports. For a list of permitted keywords and well-known port assignments, see the "Protocols and Applications" section on page B-11.

Example

To create service groups that include DNS (TCP/UDP), LDAP (TCP), and RADIUS (UDP), enter the following commands:

```
hostname (config)# object-group service services1 tcp-udp
hostname (config-service)# description DNS Group
hostname (config-service)# port-object eq domain

hostname (config)# object-group service services2 udp
hostname (config-service)# description RADIUS Group
hostname (config-service)# port-object eq radius
```

```
hostname (config-service)# port-object eq radius-acct

hostname (config)# object-group service services3 tcp
hostname (config-service)# description LDAP Group
hostname (config-service)# port-object eq ldap
```

Adding an ICMP Type Object Group

To add or change an ICMP type object group, perform the steps in this section. After you add the group, you can add more objects as required by following this procedure again for the same group name and specifying additional objects. You do not need to reenter existing objects; the commands you already set remain in place unless you remove them with the **no** form of the command.

Detailed Steps

	Command	Purpose
Step 1	**object-group icmp-type** *grp_id* **Example:** `hostname(config)# object-group icmp-type ping`	Adds an ICMP type object group. The *grp_id* is a text string up to 64 characters in length and can be any combination of letters, digits, and the following characters: • underscore "_" • dash "-" • period "." The prompt changes to ICMP type configuration mode.
Step 2	**description** *text* **Example:** `hostname(config-icmp-type)# description Ping Group`	(Optional) Adds a description. The description can be up to 200 characters.
Step 3	**icmp-object** *icmp-type* **Example:** `hostname(config-icmp-type)# icmp-object echo-reply`	Defines the ICMP types in the group. Enter the command for each type. For a list of ICMP types, see the"ICMP Types" section on page B-15.

Example

Create an ICMP type group that includes echo-reply and echo (for controlling ping) by entering the following commands:

```
hostname (config)# object-group icmp-type ping
hostname (config-service)# description Ping Group
hostname (config-service)# icmp-object echo
hostname (config-service)# icmp-object echo-reply
```

Nesting Object Groups

You can nest object groups hierarchically so that one object group can contain other object groups of the same type and you can mix and match nested group objects and regular objects within an object group. The security appliance does not support IPv6 nested object groups, however, so you cannot group an object with IPv6 entities under another IPv6 object-group.

To nest an object group within another object group of the same type, first create the group that you want to nest (see the "Configuring Object Groups" section on page 11-6), and then perform the steps in this section.

Detailed Steps

	Command	Purpose
Step 1	**object-group group** {{**protocol** | **network** | **icmp-type**} *grp_id* |**service** grp_id {**tcp** | **udp** | **tcp-udp**}} **Example:** `hostname(config)# object-group network Engineering_group`	Adds or edits the specified object group type under which you want to nest another object group. The **service**_*grp_id* is a text string up to 64 characters in length and can be any combination of letters, digits, and the following characters: • underscore "_" • dash "-" • period "."
Step 2	**group-object** *group_id* **Example:** `hostname(config-network)# group-object Engineering_groups`	Adds the specified group under the object group you specified in Step 1. The nested group must be of the same type. You can mix and match nested group objects and regular objects within an object group.

Examples

Create network object groups for privileged users from various departments by entering the following commands:

```
hostname (config)# object-group network eng
hostname (config-network)# network-object host 10.1.1.5
hostname (config-network)# network-object host 10.1.1.9
hostname (config-network)# network-object host 10.1.1.89

hostname (config)# object-group network hr
hostname (config-network)# network-object host 10.1.2.8
hostname (config-network)# network-object host 10.1.2.12

hostname (config)# object-group network finance
hostname (config-network)# network-object host 10.1.4.89
hostname (config-network)# network-object host 10.1.4.100
```

You then nest all three groups together as follows:

```
hostname (config)# object-group network admin
hostname (config-network)# group-object eng
hostname (config-network)# group-object hr
hostname (config-network)# group-object finance
```

You only need to specify the admin object group in your ACE as follows:

```
hostname (config)# access-list ACL_IN extended permit ip object-group admin host
209.165.201.29
```

Removing Object Groups

You can remove a specific object group or remove all object groups of a specified type; however, you cannot remove an object group or make an object group empty if it is used in an access list.

Detailed Step

Step 1 Do one of the following:

no object-group *grp_id* **Example:** `hostname(config)# no object-group Engineering_host`	Removes the specified object group. The *grp_id* is a text string up to 64 characters in length and can be any combination of letters, digits, and the following characters: • underscore "_" • dash "-" • period "."
clear object-group [**protocol** | **network** | **services** | **icmp-type**] **Example:** `hostname(config)# clear-object group network`	Removes all object groups of the specified type. **Note** If you do not enter a type, all object groups are removed.

Monitoring Objects and Groups

To monitor objects and groups, enter the following commands:

Command	Purpose
show access-list	Displays the access list entries that are expanded out into individual entries without their object groupings.
show running-config object-group	Displays all current object groups.
show running-config object-group *grp_id*	Displays the current object groups by their group ID.
show running-config object-group *grp_type*	Displays the current object groups by their group type.

Feature History for Objects and Groups

Table 1 lists the release history for this feature.

Table 1 Feature History for Object Groups

Feature Name	Releases	Feature Information
Object groups	7.0(1)	Object groups simplify access list creation and maintenance. The following commands were introduced or modified: **object-group** *protocol*, **object-group** *network*, **object-group** *service*, **object-group** *icmp_type*.
Objects	8.3(1)	Object support was introduced. The following commands were introduced or modified **object-network**, **object-service**, **object-group** *network*, **object-group** *service*, **network object**, **access-list extended**, **access-list webtype**, **access-list remark**

Configuring Regular Expressions

A regular expression matches text strings either literally as an exact string, or by using *metacharacters* so that you can match multiple variants of a text string. You can use a regular expression to match the content of certain application traffic; for example, you can match a URL string inside an HTTP packet. This section describes how to create a regular expression and includes the following topics:

- Creating a Regular Expression, page 11-12
- Creating a Regular Expression Class Map, page 11-15

Creating a Regular Expression

A regular expression matches text strings either literally as an exact string, or by using *metacharacters* so you can match multiple variants of a text string. You can use a regular expression to match the content of certain application traffic; for example, you can match a URL string inside an HTTP packet.

Guidelines

Use **Ctrl+V** to escape all of the special characters in the CLI, such as question mark (?) or a tab. For example, type **d[Ctrl+V]?g** to enter **d?g** in the configuration.

See the **regex** command in the *Cisco ASA 5500 Series Command Reference* for performance impact information when matching a regular expression to packets.

Note As an optimization, the adaptive security appliance searches on the deobfuscated URL. Deobfuscation compresses multiple forward slashes (/) into a single slash. For strings that commonly use double slashes, like “http://”, be sure to search for “http:/” instead.

Table 11-2 lists the metacharacters that have special meanings.

Table 11-2 regex Metacharacters

Character	Description	Notes
.	Dot	Matches any single character. For example, **d.g** matches dog, dag, dtg, and any word that contains those characters, such as doggonnit.
(*exp*)	Subexpression	A subexpression segregates characters from surrounding characters, so that you can use other metacharacters on the subexpression. For example, **d(o|a)g** matches dog and dag, but **do|ag** matches do and ag. A subexpression can also be used with repeat quantifiers to differentiate the characters meant for repetition. For example, **ab(xy){3}z** matches abxyxyxyz.
|	Alternation	Matches either expression it separates. For example, **dog|cat** matches dog or cat.
?	Question mark	A quantifier that indicates that there are 0 or 1 of the previous expression. For example, **lo?se** matches lse or lose. **Note** You must enter **Ctrl+V** and then the question mark or else the help function is invoked.
*	Asterisk	A quantifier that indicates that there are 0, 1 or any number of the previous expression. For example, **lo*se** matches lse, lose, loose, and so on.
+	Plus	A quantifier that indicates that there is at least 1 of the previous expression. For example, **lo+se** matches lose and loose, but not lse.
{*x*} or {*x*,}	Minimum repeat quantifier	Repeat at least *x* times. For example, **ab(xy){2,}z** matches abxyxyz, abxyxyxyz, and so on.
[*abc*]	Character class	Matches any character in the brackets. For example, **[abc]** matches a, b, or c.
[^*abc*]	Negated character class	Matches a single character that is not contained within the brackets. For example, **[^abc]** matches any character other than a, b, or c. **[^A-Z]** matches any single character that is not an uppercase letter.
[*a*-*c*]	Character range class	Matches any character in the range. **[a-z]** matches any lowercase letter. You can mix characters and ranges: **[abcq-z]** matches a, b, c, q, r, s, t, u, v, w, x, y, z, and so does [**a-cq-z]**. The dash (-) character is literal only if it is the last or the first character within the brackets: **[abc-]** or **[-abc]**.
“”	Quotation marks	Preserves trailing or leading spaces in the string. For example, **“test”** preserves the leading space when it looks for a match.
^	Caret	Specifies the beginning of a line.

Table 11-2 regex Metacharacters (continued)

Character	Description	Notes
\	Escape character	When used with a metacharacter, matches a literal character. For example, **\[** matches the left square bracket.
char	Character	When character is not a metacharacter, matches the literal character.
\r	Carriage return	Matches a carriage return 0x0d.
\n	Newline	Matches a new line 0x0a.
\t	Tab	Matches a tab 0x09.
\f	Formfeed	Matches a form feed 0x0c.
\x*NN*	Escaped hexadecimal number	Matches an ASCII character using hexadecimal (exactly two digits).
NNN	Escaped octal number	Matches an ASCII character as octal (exactly three digits). For example, the character 040 represents a space.

Detailed Steps

Step 1 To test a regular expression to make sure it matches what you think it will match, enter the following command:

```
hostname(config)# test regex input_text regular_expression
```

Where the *input_text* argument is a string you want to match using the regular expression, up to 201 characters in length.

The *regular_expression* argument can be up to 100 characters in length.

Use **Ctrl+V** to escape all of the special characters in the CLI. For example, to enter a tab in the input text in the **test regex** command, you must enter **test regex "test[Ctrl+V Tab]" "test\t"**.

If the regular expression matches the input text, you see the following message:

```
INFO: Regular expression match succeeded.
```

If the regular expression does not match the input text, you see the following message:

```
INFO: Regular expression match failed.
```

Step 2 To add a regular expression after you tested it, enter the following command:

```
hostname(config)# regex name regular_expression
```

Where the *name* argument can be up to 40 characters in length.

The *regular_expression* argument can be up to 100 characters in length.

Examples

The following example creates two regular expressions for use in an inspection policy map:

```
hostname(config)# regex url_example example\.com
```

```
hostname(config)# regex url_example2 example2\.com
```

Creating a Regular Expression Class Map

A regular expression class map identifies one or more regular expressions. You can use a regular expression class map to match the content of certain traffic; for example, you can match URL strings inside HTTP packets.

Detailed Steps

Step 1 Create one or more regular expressions according to the "Configuring Regular Expressions" section.

Step 2 Create a class map by entering the following command:

```
hostname(config)# class-map type regex match-any class_map_name
hostname(config-cmap)#
```

Where *class_map_name* is a string up to 40 characters in length. The name "class-default" is reserved. All types of class maps use the same name space, so you cannot reuse a name already used by another type of class map.

The **match-any** keyword specifies that the traffic matches the class map if it matches at least one of the regular expressions.

The CLI enters class-map configuration mode.

Step 3 (Optional) Add a description to the class map by entering the following command:

```
hostname(config-cmap)# description string
```

Step 4 Identify the regular expressions you want to include by entering the following command for each regular expression:

```
hostname(config-cmap)# match regex regex_name
```

Examples

The following example creates two regular expressions, and adds them to a regular expression class map. Traffic matches the class map if it includes the string "example.com" or "example2.com."

```
hostname(config)# regex url_example example\.com
hostname(config)# regex url_example2 example2\.com
hostname(config)# class-map type regex match-any URLs
hostname(config-cmap)# match regex url_example
hostname(config-cmap)# match regex url_example2
```

Scheduling Extended Access List Activation

This section includes the following topics:

Information About Scheduling Access List Activation

You can schedule each ACE in an access list to be activated at specific times of the day and week by applying a time range to the ACE.

Licensing Requirements for Scheduling Access List Activation

The following table shows the licensing requirements for this feature:

Model	License Requirement
All models	Base License.

Guidelines and Limitations for Scheduling Access List Activation

This section includes the guidelines and limitations for this feature.

Context Mode Guidelines

Supported in single and multiple context mode.

Firewall Mode Guidelines

Supported in routed and transparent firewall modes.

IPv6 Guidelines

Supports IPv6.

Additional Guidelines and Limitations

The following guidelines and limitations apply to using object groups with access lists:

- Users could experience a delay of approximately 80 to 100 seconds after the specified end time for the ACL to become inactive. For example, if the specified end time is 3:50, because the end time is inclusive, the command is picked up anywhere between 3:51:00 and 3:51:59. After the command is picked up, the adaptive security appliance finishes any currently running task and then services the command to deactivate the ACL.
- Multiple periodic entries are allowed per **time-range** command. If a **time-range** command has both **absolute** and **periodic** values specified, then the **periodic** commands are evaluated only after the **absolute** start time is reached, and they are not further evaluated after the **absolute** end time is reached.

Configuring and Applying Time Ranges

You can add a time range to implement a time-based access list. To identify the time range, perform the steps in this section.

Detailed Steps

	Command	Purpose
Step 1	**time-range** *name* **Example:** `hostname(config)# time range Sales`	Identifies the time-range name.
Step 2	Do one of the following:	
	periodic *days-of-the-week time* **to** [*days-of-the-week*] *time* **Example:** `hostname(config-time-range)# periodic monday 7:59 to friday 17:01`	Specifies a recurring time range. You can specify the following values for *days-of-the-week*: • **monday**, **tuesday**, **wednesday**, **thursday**, **friday**, **saturday**, or **sunday**. • **daily** • **weekdays** • **weekend** The *time* is in the format *hh*:*mm*. For example, 8:00 is 8:00 a.m. and 20:00 is 8:00 p.m.

	Command	Purpose
	`absolute` **`start`** *`time date`* [**`end`** *`time date`*] **Example:** `hostname(config-time-range)# absolute start 7:59 2 january 2009`	Specifies an absolute time range. The *time* is in the format *hh*:*mm*. For example, 8:00 is 8:00 a.m. and 20:00 is 8:00 p.m. The *date* is in the format *day month year*; for example, **1 january 2006**.
Step 3	**`access-list`** *`access_list_name`* [**`extended`**] {**`deny`** | **`permit`**}`...`[**`time-range`** *`name`*] **Example:** `hostname(config)# access list Marketing extended deny tcp host 209.165.200.225 host 209.165 201.1 time-range Pacific_Coast`	Applies the time range to an ACE. **Note** If you also enable logging for the ACE, use the **log** keyword before the **time-range** keyword. If you disable the ACE using the **inactive** keyword, use the **inactive** keyword as the last keyword. See Chapter 13, "Adding an Extended Access List," for complete **access-list** command syntax.

Example

The following example binds an access list named "Sales" to a time range named "New_York_Minute":

```
hostname(config)# access-list Sales line 1 extended deny tcp host 209.165.200.225 host
209.165.201.1 time-range New_York_Minute
```

Configuration Examples for Scheduling Access List Activation

The following is an example of an absolute time range beginning at 8:00 a.m. on January 1, 2006. Because no end time and date are specified, the time range is in effect indefinitely.

```
hostname(config)# time-range for2006
hostname(config-time-range)# absolute start 8:00 1 january 2006
```

The following is an example of a weekly periodic time range from 8:00 a.m. to 6:00 p.m on weekdays:

```
hostname(config)# time-range workinghours
hostname(config-time-range)# periodic weekdays 8:00 to 18:00
```

Feature History for Scheduling Access List Activation

Table 3 lists the release history for this feature.

Table 3 Feature History for Scheduling Access List Activation

Feature Name	Releases	Feature Information
Scheduling access list activation	7.0	You can schedule each ACE in an access list to be activated at specific times of the day and week. The following commands were introduced or modified: **object-group** *protocol*, **object-group** *network*, **object-group** *service*, **object-group** *icmp_type*.