Configuring Objects

This chapter describes how to configure reusable named objects and groups for use in your configuration, and it includes the following sections:

Information About Objects

Objects are reusable components for use in your configuration. They can be defined and used in ASA configurations in the place of inline IP addresses, services, names, and so on. 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.

Licensing Requirements for Objects

Model	License Requirement
All models	Base License.

Guidelines and Limitations

Context Mode Guidelines

Supported in single and multiple context mode.

Firewall Mode Guidelines

Supported in routed and transparent firewall mode.

IPv6 Guidelines

- Supports IPv6.
- The ASA does not support IPv6 nested network object groups, so you cannot group an object with IPv6 entries under another IPv6 object group.
- You can mix IPv4 and IPv6 entries in a network object group; you cannot use a mixed object group for NAT.

Additional Guidelines and Limitations

- Object 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.
- Objects and object groups share the same name space.
- You cannot remove an object or make an object empty if it is used in a command.

Configuring Objects

Configuring Network Objects and Groups

This section describes how to configure network objects and groups, and it includes the following topics:

Configuring a Network Object

A network object can contain a host, a network IP address, or a range of IP addresses, a fully qualified domain name (FQDN). You can also enable NAT rules on the object (excepting FQDN objects). (See Chapter 1, "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* | **fqdn** *fully_qualified_domain_name*} **Example:** `hostname(config-network-object)# host 10.2.2.2`	Assigns the IP address or FQDN to the named object. **Note** You cannot configure NAT for an FQDN object.
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 Network Object Group

Network object groups can contain multiple network objects as well as inline networks. Network object groups can support a mix of both IPv4 and IPv6 addresses.

Restrictions

You cannot use a mixed IPv4 and IPv6 object group for NAT, or object groups that include FQDN objects.

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	Add one or more of the following group members:	
	network-object object *name* **Example:** `hostname(config-network)# network-object host 10.2.2.4`	Adds an object to the network object group.
	network-object {**host** *ipv4_address* | *ipv4_address mask* | *ipv6-address/prefix-length*} **Example:** `hostname(config-network)# network-object host 10.2.2.4`	Adds a host or network inline, either IPv4 or IPv6.
	group-object *group_id* **Example:** `hostname(config-network)# group-object Engineering_groups`	Adds an existing object group under this object group. The nested group must be of the same type.

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
```

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
```

Configuring Service Objects and Service Groups

Service objects and groups identify protocols and ports. This section describes how to configure service objects, service groups, TCP and UDP port service groups, protocol groups, and ICMP groups, and it includes the following topics:

Configuring a Service Object

The service object can contain a protocol, ICMP, ICMPv6, TCP or UDP port or port ranges.

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* [*icmp_code*] | **icmp6** *icmp6-type* [*icmp_code*] | {**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 optional *icmp_code* specifies an ICMP code, between 1 and 255. 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 optional *icmp_code* specifies an ICMP code, between 1 and 255. For TCP or UDP, the **source** keyword specifies the source port. For TCP or UDP, 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 a Service Group

A service object group includes a mix of protocols, if desired, including optional source and destination ports for TCP or UDP.

Detailed Steps

	Command	Purpose
Step 1	**object-group service** *grp_id* **Example:** `hostname(config)# object-group service services1`	Adds a service 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 service configuration mode.
Step 2	Add one or more of the following group members:	
	service-object *protocol* **Example:** `hostname(config-service)# service-object ipsec`	Identifies the protocol name or number, between 0 and 255.
	service-object {**tcp** | **udp** | **tcp-udp**} [**source** *operator number*] [**destination** *operator number*] **Example:** `hostname(config-service)# port-object eq domain`	You can specify the source and/or destination ports, between 0 and 65535. For a list of supported names, see the CLI help. Valid operators include: • eq—Equals the port number. • gt—Greater than the port number. • lt—Less than the port number. • neq—Not equal to the port number. • range—A range of ports. Specify two numbers separated by a space, such as range 1024 4500.
	service-object {**icmp** [*icmp_type* [*icmp_code*]] | **icmp6** [*icmp6_type* [*icmp_code*]]} **Example:** `hostname(config-service)# port-object eq domain`	Specifies that the service type is for ICMP or ICMPv6 connections. You can optionally specify the ICMP type by name or number, between 0 and 255. The optional *icmp_code* specifies an ICMP code, between 1 and 255.
	service-object object *name* **Example:** `hostname(config-service)# port-object eq domain`	Specifies a service object name, created with the **object service** command.

	Command	Purpose
	group-object *group_id* **Example:** `hostname(config-network)# group-object Engineering_groups`	Adds an existing object group under this object group. The nested group must be of the same type.
Step 3	**description** *text* **Example:** `hostname(config-service)# description DNS Group`	(Optional) Adds a description. The description can be up to 200 characters.

Examples

The following example shows how to add both TCP and UDP services to a service object group:

```
hostname(config)# object-group service CommonApps
hostname(config-service-object-group)# service-object destination tcp eq ftp
hostname(config-service-object-group)# service-object destination tcp-udp eq www
hostname(config-service-object-group)# service-object destination tcp eq h323
hostname(config-service-object-group)# service-object destination tcp eq https
hostname(config-service-object-group)# service-object destination udp eq ntp
```

The following example shows how to add multiple service objects to a service object group:

```
hostname(config)# service object SSH
hostname(config-service-object)# service tcp destination eq ssh
hostname(config)# service object EIGRP
hostname(config-service-object)# service eigrp
hostname(config)# service object HTTPS
hostname(config-service-object)# service tcp source range 0 1024 destination eq https
hostname(config)# object-group service Group1
hostname(config-service-object-group)# service-object object SSH
hostname(config-service-object-group)# service-object object EIGRP
hostname(config-service-object-group)# service-object object HTTPS
```

Configuring a TCP or UDP Port Service Group

A TCP or UDP service group includes a group of ports for a specific protocol (TCP, UDP, or TCP-UDP).

	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 "." Specifies 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	Add one or more of the following group members:	
	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 1-11.
	group-object *group_id* **Example:** `hostname(config-network)# group-object Engineering_groups`	Adds an existing object group under this object group. The nested group must be of the same type.
Step 3	**description** *text* **Example:** `hostname(config-service)# description DNS Group`	(Optional) Adds a description. The description can be up to 200 characters.

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
```

Configuring an ICMP Group

An ICMP group includes multiple ICMP types.

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	Add one or more of the following group members:	
	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 1-15.
	group-object *group_id* **Example:** `hostname(config-network)# group-object Engineering_groups`	Adds an existing object group under this object group. The nested group must be of the same type.
Step 3	**description** *text* **Example:** `hostname(config-icmp-type)# description Ping Group`	(Optional) Adds a description. The description can be up to 200 characters.

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
```

Configuring a Protocol Group

A protocol group contains IP protocol types.

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	Add one or more of the following group members:	
	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 1-11.
	group-object *group_id* **Example:** `hostname(config-network)# group-object Engineering_groups`	Adds an existing object group under this object group. The nested group must be of the same type.
Step 3	**description** *text* **Example:** `hostname(config-protocol)# description New Group`	(Optional) Adds a description. The description can be up to 200 characters.

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
```

Configuring Local User Groups

You can create local user groups for use in features that support the identity firewall (IDFW) by including the group in an extended ACL, which in turn can be used in an access rule, for example.

The ASA sends an LDAP query to the Active Directory server for user groups globally defined in the Active Directory domain controller. The ASA imports these groups for identity-based rules. However, the ASA might have localized network resources that are not defined globally that require local user groups with localized security policies. Local user groups can contain nested groups and user groups that are imported from Active Directory. The ASA consolidates local and Active Directory groups.

A user can belong to local user groups and user groups imported from Active Directory.

Prerequisites

See Chapter 1, "Configuring the Identity Firewall," to enable IDFW.

Detailed Steps

	Command	Purpose
Step 1	**object-group user** *user_group_name* **Example:** `hostname(config)# object-group user users1`	Defines object groups that you can use to control access with the Identity Firewall.
Step 2	Add one or more of the following group members:	
	user *domain_NetBIOS_name\user_name* **Example:** `hostname(config-user-object-group)# user SAMPLE\users1`	Specifies the user to add to the access rule. The *user_name* can contain any character including [a-z], [A-Z], [0-9], [!@#$%^&()-_{}.]. If *domain_NetBIOS_name\user_name* contains a space, you must enclose the domain name and user name in quotation marks. The *user_name* can be part of the LOCAL domain or a user imported by the ASA from Active Directory domain. If the *domain_NetBIOS_name* is associated with a AAA server, the *user_name* must be the Active Directory sAMAccountName, which is unique, instead of the common name (cn), which might not be unique. The *domain_NetBIOS_name* can be LOCAL or the actual domain name as specified in **user-identity domain** *domain_NetBIOS_name* **aaa-server** *aaa_server_group_tag* command.
	group-object *group_id* **Example:** `hostname(config-network)# group-object Engineering_groups`	Adds an existing object group under this object group. The nested group must be of the same type.
Step 3	**description** *text* **Example:** `hostname(config-protocol)# description New Group`	(Optional) Adds a description. The description can be up to 200 characters.

Configuring Security Group Object Groups

You can create security group object groups for use in features that support Cisco TrustSec by including the group in an extended ACL, which in turn can be used in an access rule, for example.

When integrated with Cisco TrustSec, the ASA downloads security group information from the ISE. The ISE acts as an identity repository, by providing Cisco TrustSec tag to user identity mapping and Cisco TrustSec tag to server resource mapping. You provision and manage security group access lists centrally on the ISE.

However, the ASA might have localized network resources that are not defined globally that require local security groups with localized security policies. Local security groups can contain nested security groups that are downloaded from the ISE. The ASA consolidates local and central security groups.

To create local security groups on the ASA, you create a local security object group. A local security object group can contain one or more nested security object groups or Security IDs or security group names. User can also create a new Security ID or security group name that does not exist on the ASA.

You can use the security object groups you create on the ASA to control access to network resources. You can use the security object group as part of an access group or service policy.

Prerequisites

See Chapter 1, "Configuring the ASA to Integrate with Cisco TrustSec," to enable TrustSec.

Detailed Steps

	Command	Purpose
Step 1	**object-group security** *objgrp_name* **Example:** `hostname(config)# object-group security mktg-sg`	Creates a security group object. Where *objgrp_name* is the name for the group entered as a 32-byte case sensitive string. The *objgrp_name* can contain any character including [a-z], [A-Z], [0-9], [!@#$%^&()-_{}.].
Step 2	Add one or more of the following group members:	
	security-group {**tag** *sgt#* | **name** *sg_name*} **Example:** `hostname(config)# security-group name mktg`	Specifies the type of security group object as either an inline tag or a named object. • **tag** *sgt#*—Enter a number from 1 to 65533 for a Tag security type. • **name** *sg_name*—Enter a 32-byte case-sensitive string for a Name security type. The *sg_name* can contain any character including [a-z], [A-Z], [0-9], [!@#$%^&()-_{}.]. An SGT is assigned to a device through IEEE 802.1X authentication, web authentication, or MAC authentication bypass (MAB) by the ISE. Security group names are created on the ISE and provide user-friendly names for security groups. The security group table maps SGTs to security group names.

	Command	Purpose
	group-object *group_id* **Example:** `hostname(config-network)# group-object Engineering_groups`	Adds an existing object group under this object group. The nested group must be of the same type.
Step 3	**description** *text* **Example:** `hostname(config-protocol)# description New Group`	(Optional) Adds a description. The description can be up to 200 characters.

Examples

The following example shows how to configure a security group object:

```
hostname(config)# object-group security mktg-sg
hostname(config)# security-group name mktg
hostname(config)# security-group tag 1
```

The following example shows how to configure a security group object:

```
hostname(config)# object-group security mktg-sg-all
hostname(config)# security-group name mktg-managers
hostname(config)# group-object mktg-sg // nested object-group
```

Configuring Regular Expressions

Creating a Regular Expression

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.

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 command reference for performance impact information when matching a regular expression to packets.

Note As an optimization, the ASA 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 1-1 lists the metacharacters that have special meanings.

Table 1-1 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 1-1 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.

Prerequisites

Create one or more regular expressions according to the "Creating a Regular Expression" section on page 1-14.

Detailed Steps

Step 1 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 2 (Optional) Add a description to the class map by entering the following command:

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

Step 3 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
```

Configuring Time Ranges

Create a reusable component that defines starting and ending times that can be applied to various security features. Once you have defined a time range, you can select the time range and apply it to different options that require scheduling.

The time range feature lets you define a time range that you can attach to traffic rules, or an action. For example, you can attach an access list to a time range to restrict access to the ASA.

A time range consists of a start time, an end time, and optional recurring entries.

Guidelines

- Multiple periodic entries are allowed per time range. If a time range has both absolute and periodic values specified, then the periodic values are evaluated only after the absolute start time is reached, and they are not further evaluated after the absolute end time is reached.
- Creating a time range does not restrict access to the device. This procedure defines the time range only.

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.
	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**.

Examples

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
```

Monitoring Objects

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

Table 1-2 lists each feature change and the platform release in which it was implemented.

Table 1-2 Feature History for Object Groups

Feature Name	Platform Releases	Feature Information
Object groups	7.0(1)	Object groups simplify access list creation and maintenance. We introduced or modified the following commands: **object-group** *protocol*, **object-group** *network*, **object-group** *service*, **object-group** *icmp_type*.
Objects	8.3(1)	Object support was introduced. We introduced or modified the following commands: **object-network**, **object-service**, **object-group** *network*, **object-group** *service*, **network object**, **access-list extended**, **access-list webtype**, **access-list remark**.
User Object Groups for Identity Firewall	8.4(2)	User object groups for identity firewall were introduced. We introduced the following commands: **object-network user**, **user**.

Table 1-2 Feature History for Object Groups (continued)

Feature Name	Platform Releases	Feature Information
Mixed IPv4 and IPv6 network object groups	9.0(1)	Previously, network object groups could only contain all IPv4 addresses or all IPv6 addresses. Now network object groups can support a mix of both IPv4 and IPv6 addresses. **Note** You cannot use a mixed object group for NAT. We modified the following commands: **object-group network**.
Security Group Object Groups for Cisco TrustSec	8.4(2)	Security group object groups for TrustSec were introduced. We introduced the following commands: **object-network security**, **security**.
Extended ACLand object enhancement to filter ICMP traffic by ICMP code	9.0(1)	ICMP traffic can now be permitted/denied based on ICMP code. We introduced or modified the following commands: **access-list extended**, **service-object**, **service**.