This section describes how to configure server persistence using stickiness on the Cisco 4700 Series Application Control Engine (ACE) appliance.

Guide Contents
Overview
Setting Up an ACE Appliance
Creating a Virtual Context
Configuring Access Control Lists
Configuring Role-Based Access Control
Configuring Server Load Balancing
Configuring a Load-Balancing Predictor
Configuring Server Persistence Using Stickiness (this section)
Configuring SSL Security
Configuring Health Monitoring Using Health Probes

Contents

Overview

After reading this section, you should have a basic understanding of how the ACE appliance provides server persistence using stickiness, and how to configure HTTP cookie stickiness.

When customers visit an e-commerce site, they usually start by browsing the site. Depending on the application, the site may require that the client become persisted (stuck) to one server as soon as the initial connection is established, or the application may require this action only when the client starts to create a transaction, such as when building a shopping cart.

For example, after the client adds items to a shopping cart, it is important that all subsequent client requests are directed to the same real server so that all the items are contained in one shopping cart on one server. An instance of a customer?s shopping cart is typically local to a particular server rather than duplicated across

multiple servers.

E-commerce applications are not the only types of applications that require a sequence of client requests to be directed to the same real server. Any web applications that maintain client information may require stickiness, such as banking and online trading applications, or FTP and HTTP file transfers.

The ACE can be configured so that the same client can maintain multiple, simultaneous, or subsequent TCP or IP connections with the same real server for the duration of a session. This session persistence capability of the ACE is called stickiness. A session is defined as a series of transactions between a client and a server over some finite period of time (from several minutes to several hours).

Depending on the configured server load-balancing policy, the ACE sticks a client to an appropriate server after the ACE determines which load-balancing method to use. If the ACE determines that a client is already stuck to a particular server, then the ACE sends that client request to that server, regardless of the load-balancing criteria. If the ACE determines that the client is not stuck to a particular server, it applies the normal load-balancing rules to the request.

To determine how a particular client is stuck to a specific web server and how an application distinguishes each client or a group of clients, the ACE supports the following sticky methods:

- Source and/or destination IP address?For stickiness, you can use the source IP address, the destination IP address, or both to uniquely identify individual clients and their requests based on their IP net masks. However, if an enterprise or service provider uses a mega-proxy (a free, anonymous web proxy service) to establish client connections to the Internet, the source IP address is not a reliable indicator of the true source of the request. In this case, you can use another sticky method to ensure session persistence.

- Cookie?Client cookies uniquely identify clients to the ACE and to the servers that provide content. A cookie is a small data structure within the HTTP header that a server uses to deliver data to a web client, with the request that the client store the information. This information might include items that users have added to their shopping carts or travel dates that they have chosen. When the ACE examines a request for content and determines that the * content is sticky, it examines any cookie or URL present in the content request. The ACE uses the information in the cookie or URL to direct the content request to the appropriate server.

- Hypertext Transfer Protocol (HTTP) header?You can specify a header offset to provide stickiness based on a unique portion of the HTTP header.

The e-commerce application often dictates which of these methods is appropriate for a particular e-commerce application.

The ACE uses sticky groups for stickiness attributes. These attributes include the sticky method, timeout, replication, and attributes related to a particular sticky method.

To track sticky connections, the ACE uses a sticky table with information about sticky groups, sticky methods, sticky connections, and real servers. The ACE uses a configurable timeout mechanism to age out sticky table entries. When an entry times out, it becomes eligible for reuse. High connection rates may cause the premature aging out of sticky entries. In this case, the ACE reuses the entries that are closest to expiration first.

Entries in the sticky table can be either dynamic (generated by the ACE as needed) or static (configured). When you create a static sticky entry, the ACE places the entry in the sticky table immediately, and it remains in the sticky database until you remove it from the configuration.

You can configure stickiness by following these steps:

1. Ensure that resources are allocated for stickiness.

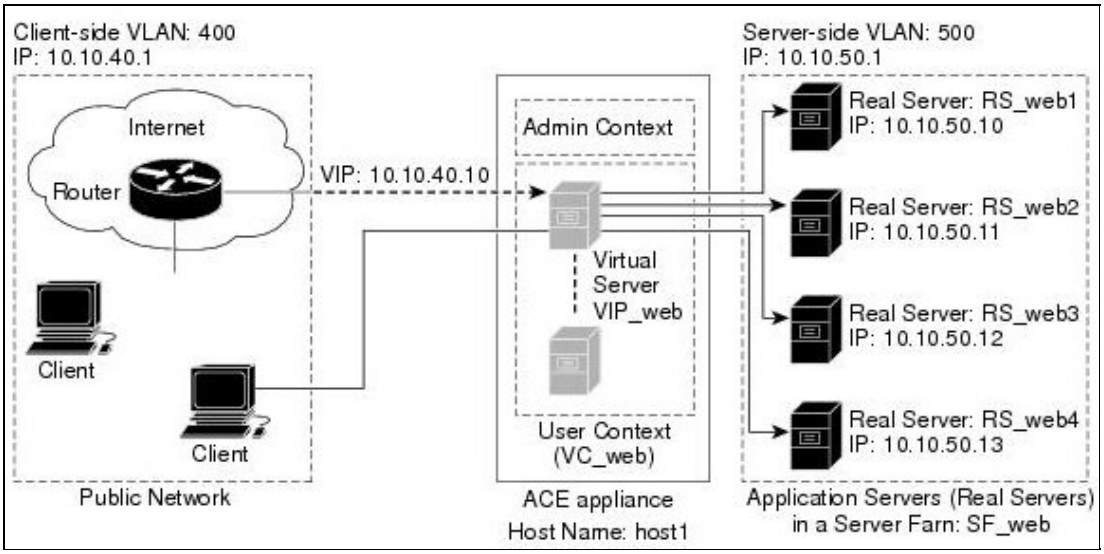
2. Create a sticky group.

3. Associate the sticky group with a Layer 7 server load-balancing action of a virtual server.

4. Deploy the configuration.

Figure 1 illustrates that in a server load-balancing environment, requests from a client are stuck to real server RS_web4 in a session.

Figure 1 Client Requests Stuck to a Server



This section describes how to configure stickiness using the HTTP cookie sticky method. For information on how to configure stickiness using the IP address and HTTP header methods, see:

Cisco ACE 4700 Series Appliance Server Load-Balancing Configuration Guide.

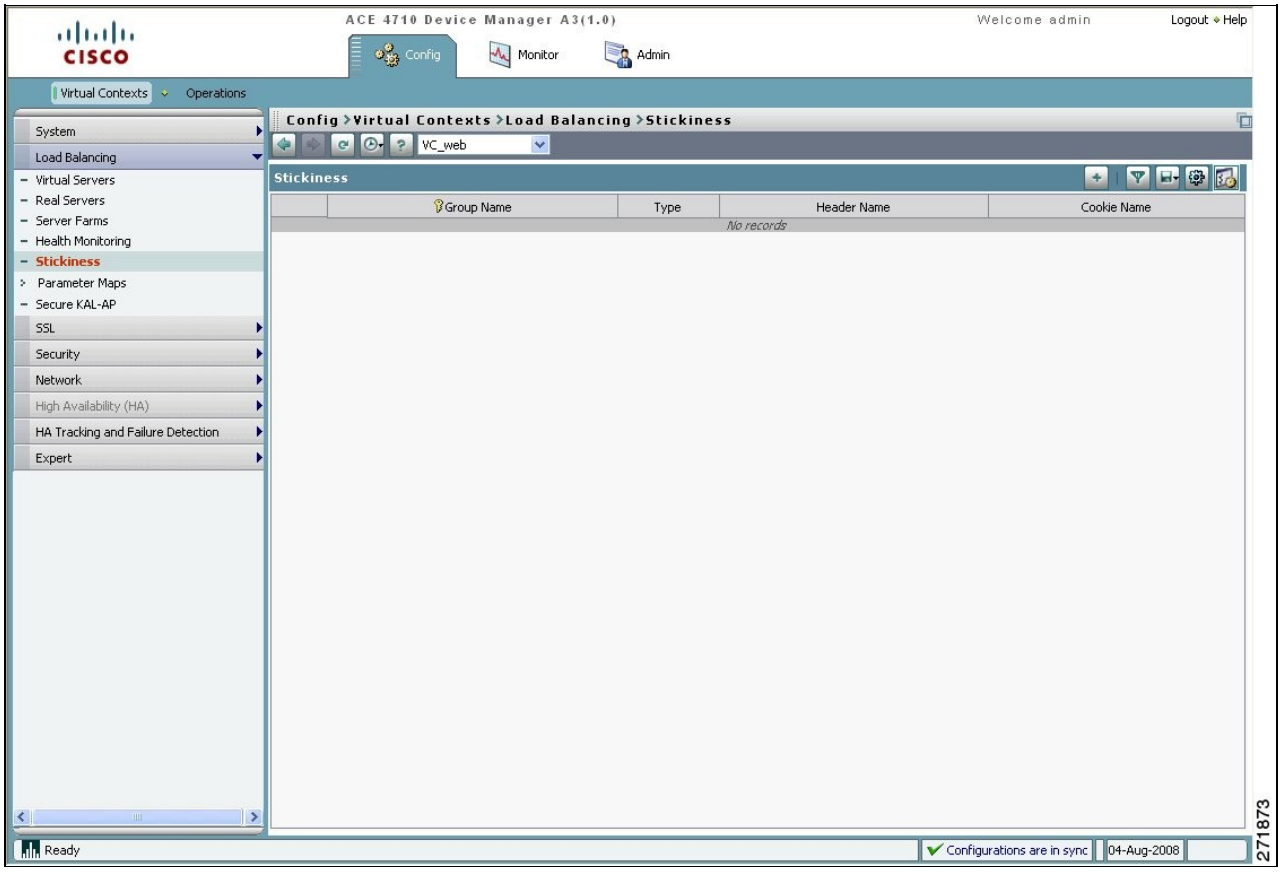
Configuring HTTP Cookie Stickiness Using the Device Manager GUI

You can configure HTTP cookie stickiness using the GUI by following these steps:

1. Make sure that the context in which you are configuring the sticky group is associated with a resource class that allocates resources to stickiness. See the ?Creating a Resource Class? procedure in the Creating a Virtual Context section.

2. Choose **Load Balancing > Stickiness**. The Stickiness pane appears (Figure 2).

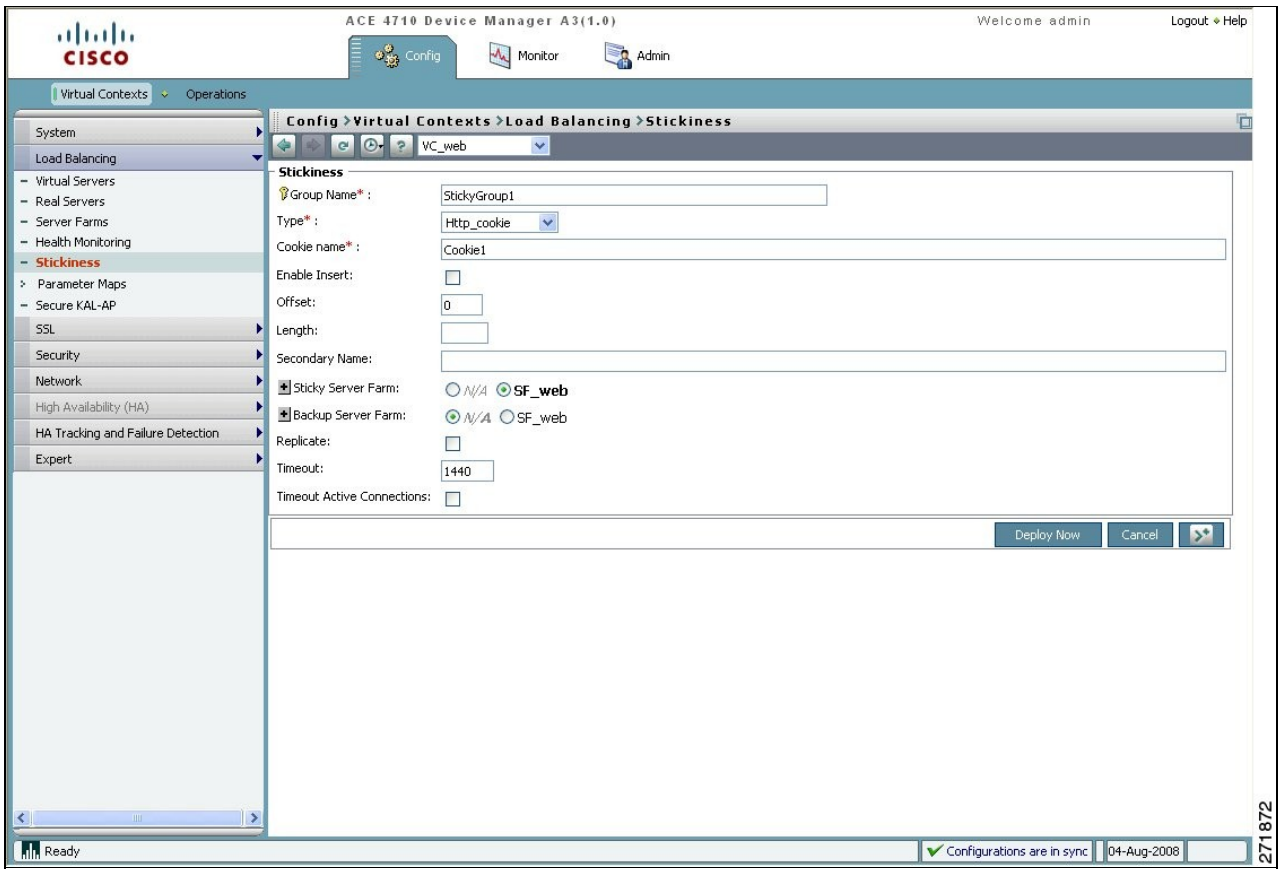
Figure 2 Stickiness Pane



3. Choose the **VC_web** context.

4. Add a new sticky group by clicking **Add**. The Stickiness configuration window appears (Figure 3).

Figure 3 Stickiness Configuration Window



5. Enter the following attributes for the new sticky group. Leave the remaining attributes blank or with their default values.

◊ Group Name: StickyGroup1

◊ Type: Http_cookie

◊ Cookie name: Cookie1

◊ Sticky Server Farm: SF_web

6. Add the new sticky group to the Stickiness pane by clicking **Deploy Now.**

Configuring HTTP Cookie Stickiness Using the CLI

You can configure HTTP cookie stickiness using the CLI by following these steps:

1. Verify that you are operating in the desired context by checking the CLI prompt. If necessary, change to the correct context.

```
host1/Admin# changeto VC_web

host1/VC_web#
```

2. Enter configuration mode.

```
host1/VC_web# config

host1/VC_web(config)#
```

3. Create an HTTP-cookie-type sticky group and enter the cookie configuration mode.

```
host1/VC_web(config)# sticky http-cookie Cookie1 StickyGroup1

host1/VC_web(config-sticky-cookie)#
```

4. Configure a timeout for HTTP cookie stickiness.

```
host1/VC_web(config-sticky-cookie)# timeout 1440
```

5. Associate a server farm with the sticky group and exit configuration mode.

```
host1/VC_web(config-sticky-cookie)# serverfarm SF_web

host1/VC_web(config-sticky-cookie)# exit

host1/VC_web(config)# exit

host1/VC_web#
```

6. Display the HTTP cookie configuration.

```
host1/VC_web# show running-config sticky
```

In this section, you have configured a sticky group using the HTTP-cookie method. In the next section, you will configure SSL security.