Server Load Balancing with SAP and ACE

This guide provides configuration best practices for application optimization with SAP Business Suite and the Cisco data center solutions, including the Cisco Application Control Engine (ACE), Wide Area Application Services (WAAS), and the Cisco Application Analysis Solution (AAS). This guide describes how to accomplish the following:

- Deploy ACE into an existing server farm with minimal cost and disruption through the use of virtual contexts and role-based access control
- Optimize ACE server load balancing features for SAP including health monitoring and session persistence
- Achieve high availability with stateful failover on the ACE
- Analyze SAP transactions using the Cisco AAS
- Evaluate alternative deployment schemes for SAP with Cisco WAAS

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

SAP Overview

SAP Business Suite includes the following functional applications: Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Product Lifecycle Management (PLM), Supply Chain Management (SCM), and Supplier Relationship Management (SRM). It also includes the following middleware that provide infrastructure support to the functional applications:

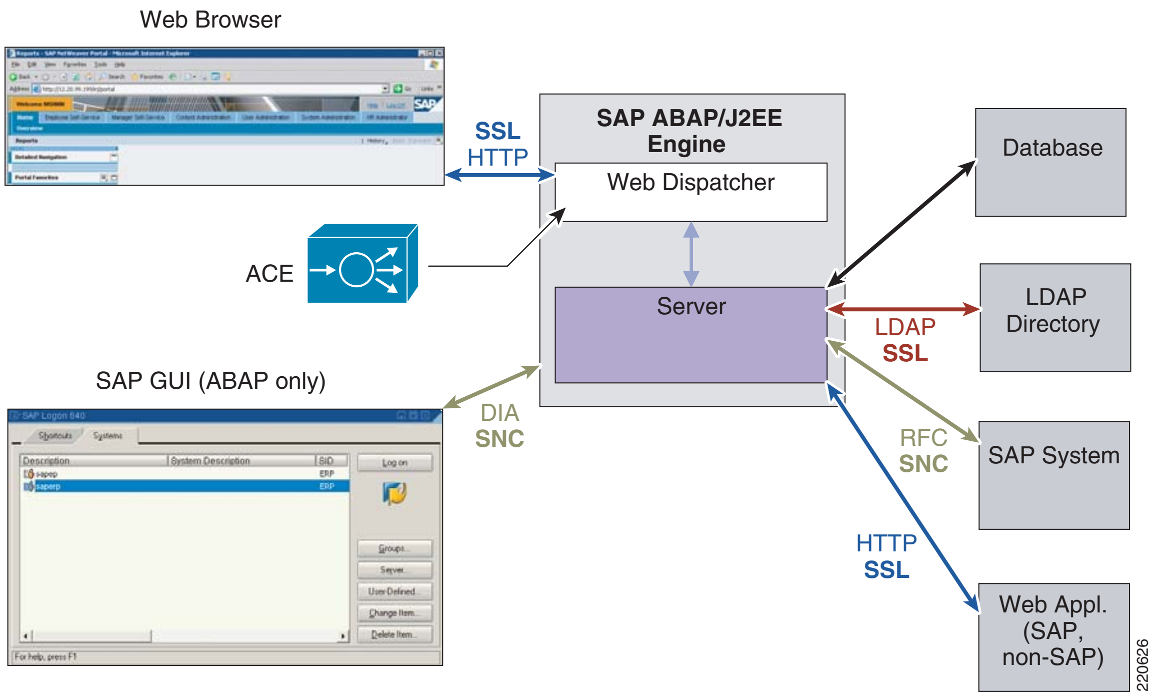
- Business Intelligence (BI, introduced 1998)—Provides enhanced graphical data analysis
- Enterprise Portal (EP, introduced 1999)—Provides a web-based portal
- Exchange Interconnect (XI, introduced 2002)—Provides an application gateway
- Master Data Management (MDM, introduced 2004)—Consolidates master data from multiple systems

These middleware modules along with other technologies form the foundation of the SAP Service Oriented Architecture, and today they are bundled together and marketed as NetWeaver 2004s. When combined with SAP Business Suite, they create a full enterprise deployment. Some of these middleware modules have also been included in earlier bundles from SAP such as MySAP (2003), Web Application Server (2004), and the original NetWeaver. The testing here focuses on the latest release of ERP (ECC 7.0) and NetWeaver 2004s (including BI, XI, MDM, and EP).

Server Load Balancing and SAP

SAP uses a three-tiered architecture, as shown in Figure 1.

Figure 1 SAP Architecture



The presentation tier is either SAPGUI, a Windows application that runs on TCP/IP, or a web browser.

The SAP application tier is based on the SAP proprietary ABAP code as well as Java. It runs on almost any platform, from Windows to Linux servers to mainframes. SAP application servers scale by separating functions into a Central Instance (CI) and Dialog Instance (DI). The CI is the message server and is responsible for queuing and locking, while the DIs perform the actual processing of the application. SAP increases capacity by adding more DIs, thus the need for load balancing. There is a single CI, which achieves high availability through the clustering software of the platform on which it runs. As a result, messages to the CI cannot be load balanced. SAPGUI traffic is also ineligible for load balancing because it is sent only to the CI.

SAP can be deployed with numerous third-party databases such as Oracle, IBM, and Microsoft. Like the CI, there is a single instance of the database, and it achieves high availability through clustering, not load balancing.

Figure 1 shows Web Dispatcher (WD) software as the entity responsible for server load balancing and SSL offload. ACE provides an alternative to WD for environments requiring greater scalability and high availability. Following is a review of the services provided by ACE in a SAP environment.

- Server selection

  Both ACE and WD use methods to select servers as well as health checks to ensure their availability. WD excels in its knowledge of the server state. Because of its proprietary link to the SAP message server or CI, WD optimizes load distribution to the DIs, knowing precisely the performance load and availability characteristics of each server. Out of the box, ACE relies on more generic server selection processes such as weighted round robin and least connections. However, XML scripts can be customized to influence server selection based on other variables such as the number of dialog processes available in a DI.

- High performance SSL

WD offers SSL offload, but as a software-based solution, the performance achieved is gated by the platform that is running WD. ACE is designed specifically for high performance SSL and performs this function in hardware, providing up to 15,000 SSL connections per second. By terminating SSL on the ACE module, the traffic can be passed in the clear so that security mechanisms such as intrusion detection/prevention and web application security can be employed before delivery to the server. After these services have been applied, ACE can re-encrypt the traffic and forward it to the server. Alternatively, the SSL termination process can be entirely offloaded from the server for a significant performance boost of up to 10 times in some cases. When SSL session reuse is available on ACE, there will be a middle ground where traffic is still encrypted all the way to the server but at a lower processing cost.

- Content routing

  ACE can look deep into the HTTP header to make forwarding decisions based on other variables besides server load and availability. By directing requests to servers tuned for certain types of content, web browser or language support, the server environment can be further optimized.

- WAN acceleration

  In high latency WAN environments, response time for web transactions can be unacceptably slow. ACE provides a scalable way to transparently move server-destined transactions to acceleration devices such as Cisco Wide Area Application Services (WAAS) to optimize the session before forwarding traffic to the real servers. This results in better session response time for clients while providing high availability and scalability for the optimization engines themselves.

- Security

  When ACE is deployed inline, stateful access control lists (ACLs) limit the traffic flows allowed between clients and servers. Access control prevents worms and intruders from arbitrarily scanning and attacking vulnerable ports that may be open on server platforms but not necessary for the application itself. Locating access control in the network limits vulnerabilities, even when the servers are not completely hardened.

- End-to-end health monitoring

  Through the use of advanced probes, ACE can provide health checks that extend beyond the web/application servers themselves to also calculate the availability of all critical path resources before sending requests to a server. For example, if the database is down, ACE can forward the request to a backup server farm or sorry server rather than sending it to a web server.

- TCP reuse

  Instead of setting up a new TCP connection for every flow destined to a server, ACE can multiplex multiple flows across a single connection. This reduces the processing load on the server, allowing it to support more simultaneous connections.

- High availability

  Because ACE maintains the state of each connection and sticky entry in the standby context, failover is seamless and connections remain intact. This means that service is not impacted during maintenance or failures in the network.

- SLB consolidation

  While a pair of web dispatchers must be dedicated to each SAP application, all these functions can be combined onto a single pair of ACE modules or contexts.

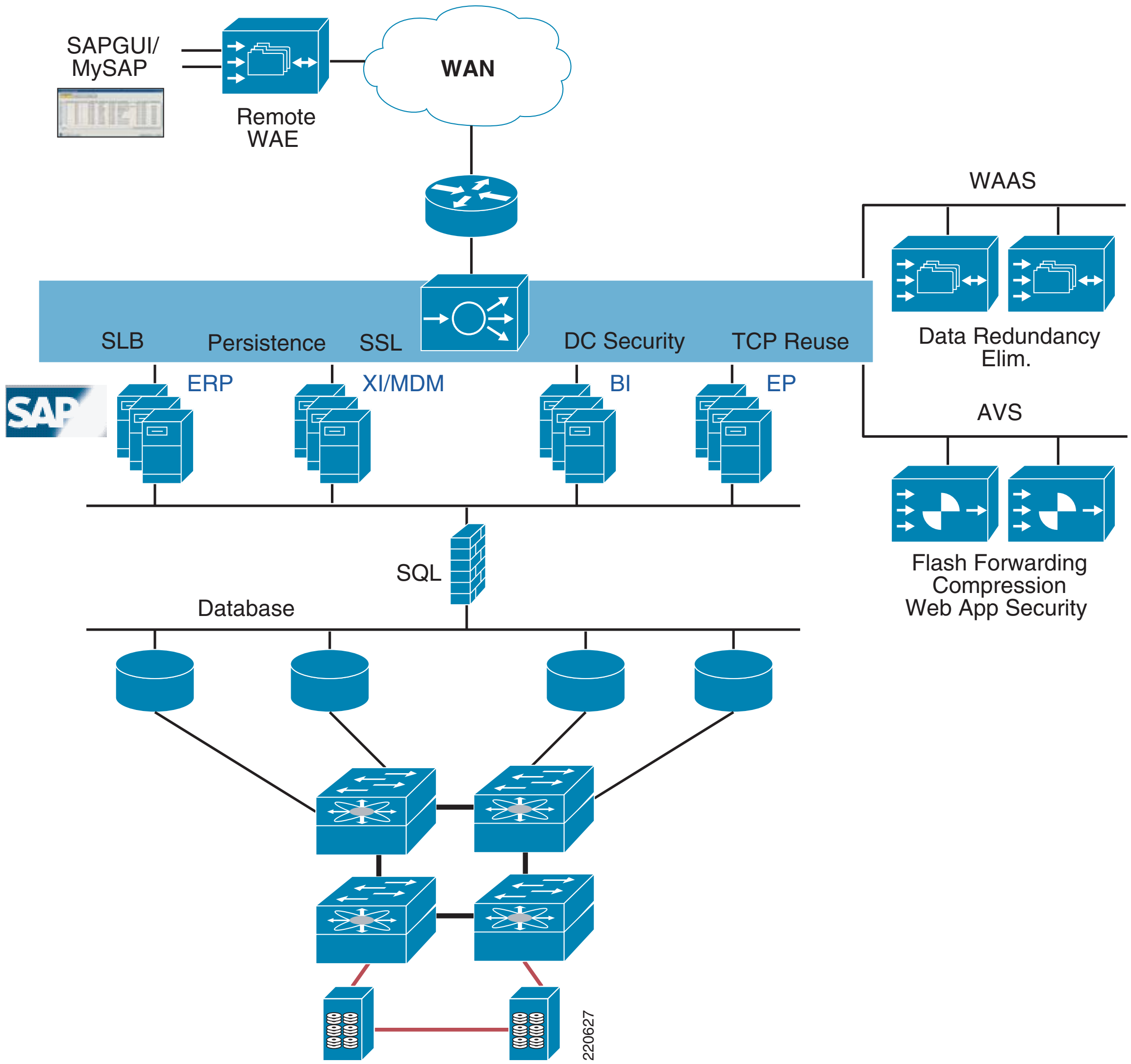
Network Design and Virtualization

The server farm consists of 18 servers. Each group has one CI and three DIs, except MDM, which has a single server running the CI/DI. There is also a separate database server for each application.

- Three ERP—saperp, saperp1, and saperp2
- Three EP—sapep, sapep1, and sapep2
- Three BI—sapbi, sapbi1, and sapbi2
- Three XI—sapxi, sapxi1, and sapxi2
- One MDM—sapmdm
- Five DB—saperpdb, sapepdb, sapbidb, sapxidb, and sapmdmdb

Figure 2 shows the logical design of application networking components for SAP. There is a single routed inline ACE context assigned for SAP. It has a separate virtual LAN (VLAN) interface for each of the SAP server groups (ERP, XI/MDM, BI, and EP) as well as a client VLAN interface to the Multilayer Switch Feature Card (MSFC). In this way, ACE stateful ACLs can be used to limit vulnerabilities and exploits between the servers. As shown, the ACE context plays a central role in this configuration. For the servers, it provides server load balancing, SSL offload, and TCP connection reduction as well as security. It also provides a way to scale the deployment of WAAS, which speeds performance on the WAN.

Figure 2 SAP Logical Design



This logical environment is implemented on a pair of Catalyst 6500s, each with a single Cisco Firewall Services Module (FWSM) and ACE module. Both ACE and FWSM are deployed in active-active mode, with the location of active/standby roles varying by context. This document does not discuss the overall configuration but does show the incremental tasks associated with deploying SAP into this existing environment.

When introducing a new application into the data center, the ideal scenario is to use the existing switching, routing, security, and load balancing resources already in place without having to order and configure additional racks of equipment. With the use of virtual contexts on ACE and FWSM, the SAP environment can be built this way and yet independently of the other server farms. The SAP contexts have their own configuration file and resources, and are not affected by changes in the other tiers.

To create the new SAP virtual context, assign the relevant VLANs to both the Catalyst 6500 and the ACE admin context. Setting a resource class for a context is typically optional, but it is required in the case of cookie-based persistence.

- Catalyst 6500

```
svclc vlan-group 1  952,956-958,962
```

- ACE admin context

```
resource-class sap
  limit-resource all minimum 0.00 maximum unlimited
  limit-resource sticky minimum 20.00 maximum equal-to-min

context sap
  allocate-interface vlan 952
  allocate-interface vlan 956-958
  allocate-interface vlan 962
  member sap
```

After the SAP context has been built and allocated, VLAN interfaces define the real servers and associate them with a server farm:

```
rserver host ep
  ip address 12.20.57.10
  inservice
rserver host ep1
  ip address 12.20.57.11
  inservice
rserver host ep2
  ip address 12.20.57.12
  inservice

serverfarm host ep
  probe ep
  rserver ep 51000
    inservice
  rserver ep1 51000
    inservice
  rserver ep2 51000
    inservice
```

Note that the SAP enterprise portal uses port 51000. ACE receives requests to the VIP on port 80 and translates them to port 51000 using the server farm configuration shown above.

Role-Based Access Control

As discussed above, an SAP environment may consist of many different modules. In this lab environment, there is a single functional application, SAP Business Suite ERP, which itself has database and application components. There is also the suite of Service-Oriented Architecture services from NetWeaver, including the Enterprise Portal and Business Intelligence, which each have their own application servers and databases. In other cases, SAP is much more broadly deployed, with additional functional applications such as Customer Relationship Management, Product Lifecycle Management, Supply Chain Management, and Supplier Relationship Management.

Each of these software modules may have its own team of server administrators, and responsibilities for the application delivery might be divided between applications, database, security, management layer, and so on. ACE provides a mechanism to customize the scope of control available to these various administrators using a capability called roles-based access control (RBAC). This constrains the commands and actions of an administrator to the role they have been assigned. ACE comes pre-packaged with a number of predefined roles, as shown below. Roles can also be customized as needed.

```
ACE-1/sap# sh role

Role: Admin (System-defined)
 Description: Administrator
```

```
Number of rules: 2
 ---------------------------------------------
 Rule    Type    Permission      Feature
 ---------------------------------------------
  1.   Permit    Create                 all
  2.   Permit    Create         user access

Role: Network-Admin (System-defined)
Description: Admin for L3 (IP and Routes) and L4 VIPs
Number of rules: 6
 ---------------------------------------------
 Rule    Type    Permission      Feature
 ---------------------------------------------
  1.   Permit    Create           interface
  2.   Permit    Create             routing
  3.   Permit    Create          connection
  4.   Permit    Create                 nat
  5.   Permit    Create                 vip
  6.   Permit    Create         config_copy

Role: Server-Maintenance (System-defined)
Description: Server maintenance, monitoring and debugging
Number of rules: 5
 ---------------------------------------------
 Rule    Type    Permission      Feature
 ---------------------------------------------
  1.   Permit    Modify                real
  2.   Permit     Debug          serverfarm
  3.   Permit     Debug                 vip
  4.   Permit     Debug               probe
  5.   Permit     Debug         loadbalance

Role: Server-Appln-Maintenance (System-defined)
Description: Server maintenance and L7 policy application
Number of rules: 4
 ---------------------------------------------
 Rule    Type    Permission      Feature
 ---------------------------------------------
  1.   Permit    Create                real
  2.   Permit    Create          serverfarm
  3.   Permit    Create         loadbalance
  4.   Permit    Create         config_copy

Role: SLB-Admin (System-defined)
Description: Administrator for all load-balancing features
Number of rules: 8
 ---------------------------------------------
 Rule    Type    Permission      Feature
 ---------------------------------------------
  1.   Permit    Create                real
  2.   Permit    Create          serverfarm
  3.   Permit    Create                 vip
  4.   Permit    Create               probe
  5.   Permit    Create         loadbalance
  6.   Permit    Create                 nat
  7.   Permit    Modify           interface
  8.   Permit    Create         config_copy

Role: Security-Admin (System-defined)
Description: Administrator for all security features
Number of rules: 7
 ---------------------------------------------
 Rule    Type    Permission      Feature
 ---------------------------------------------
```

```
  1.   Permit    Create         access-list
  2.   Permit    Create             inspect
  3.   Permit    Create          connection
  4.   Permit    Modify           interface
  5.   Permit    Create                 aaa
  6.   Permit    Create                 nat
  7.   Permit    Create         config_copy

Role: SSL-Admin (System-defined)
Description: Administrator for all SSL features
Number of rules: 4
 ---------------------------------------------
 Rule    Type    Permission      Feature
 ---------------------------------------------
  1.   Permit    Create                 ssl
  2.   Permit    Create                 pki
  3.   Permit    Modify           interface
  4.   Permit    Create         config_copy

Role: Network-Monitor (System-defined)
Description: Monitoring for all features
Number of rules: 1
 ---------------------------------------------
 Rule    Type    Permission      Feature
 ---------------------------------------------
  1.   Permit   Monitor                 all
```

Two of these roles, security and SLB-admin, are leveraged here for the SAP environment. In this way, security personnel can manage access control without the possibility of mistakenly disrupting the server load balancing configuration, and vice versa. Applying these pre-defined roles is simple; in the following example, "Mark" is assigned the role of security admin, and "Tom" is assigned as the SLB admin.

```
user Mark pass cisco123 role Security-Admin
user Tom pass cisco123 role SLB-Admin
```

When Mark logs in and tries to configure an rserver, he is blocked:

```
ACE-1/sap# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
ACE-1/sap(config)# rserver xyz
                   ^
% invalid command detected at '^' marker.
```

Only commands permitted by the security role can be seen by Mark:

```
ACE-1/sap(config)# ?
Configure commands:
  aaa              Configure aaa functions
  access-group     Activate context global access-list
  access-list      Configure access control list
  arp              Configure ARP
  banner           Configure banner message
  class-map        Configure a Class map
  do               EXEC command
  end              Exit from configure mode
  exit             Exit from configure mode
  interface        Configure an interface
  ip               Configure IP features
  ldap-server      Configure LDAP related parameters
  no               Negate a command or set its defaults
  parameter-map    Configure a parameter map
  policy-map       Configure a policy map
  radius-server    Configure RADIUS related parameters
  service-policy   Enter service policy to be applied to this context
```

```
  snmp-server     Configure snmp server
  ssh             Configure SSH parameters
  tacacs-server   Configure TACACS+ server related parameters
  timeout         Configure the maximum timeout duration
  username        Configure user information.
```

Similarly, Tom, the SLB admin, sees only commands related to his role:

```
ACE-1/sap# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
ACE-1/sap(config)# ?
Configure commands:
  access-group    Activate context global access-list
  arp             Configure ARP
  class-map       Configure a Class map
  do              EXEC command
  end             Exit from configure mode
  exit            Exit from configure mode
  interface       Configure an interface
  ip              Configure IP features
  no              Negate a command or set its defaults
  parameter-map   Configure a parameter map
  policy-map      Configure a policy map
  probe           Configure probe
  rserver         Configure rserver
  script          Configure script file and tasks
  serverfarm      Configure serverfarm
  service-policy  Enter service policy to be applied to this context
  snmp-server     Configure snmp server
  ssh             Configure SSH parameters
  timeout         Configure the maximum timeout duration
  username        Configure user information.
ACE-1/sap(config)#
```

With the use of domains, these policies can be further restricted. In the following example, Tom, the ERP administrator, is allowed to modify only the ERP policy-map:

```
domain ERP
  add-object policy-map ERP-policy
user tom pass cisco123 role SLB-Admin domain ERP
```

When Tom attempts to edit the Supply Chain Management policy, he is blocked. However, when Tom edits the allowed ERP-policy, the action is permitted:

```
ACE-1/sap(config)# policy-map type load first SCM-policy
Error: object being referred to is not part of User's domain
ACE-1/sap(config)# policy-map type load first ERP-policy
ACE-1/sap(config-pmap-lb)#
```

Similarly, the security admin can also be further restricted. In this example, Mark, the security admin, is allowed to perform the pre-defined security-admin functions, but only on the client VLAN (VLAN 962):

```
domain client
  add-object interface vlan 962
username mark password cisco123 role Security-Admin domain client

ACE-1/sap# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
ACE-1/sap(config)# int vlan 968
Error: object being referred to is not part of User's domain
ACE-1/sap(config)# int vlan 962
ACE-1/sap(config-if)#
```

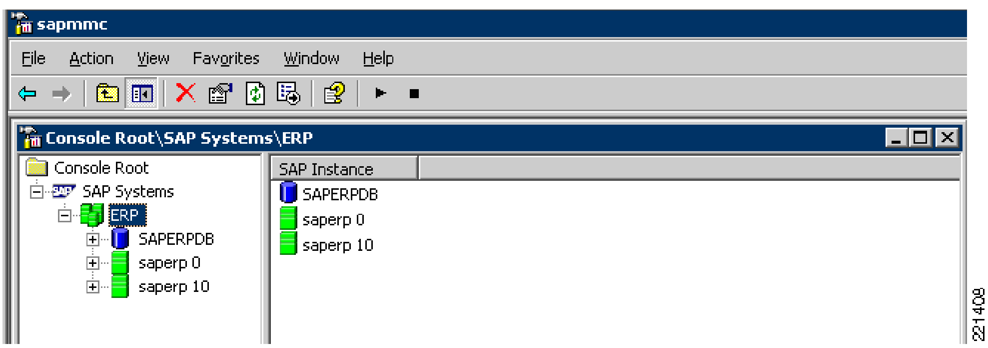
Monitoring Server Health

Another valuable function of ACE is to periodically check the health of each server. If the servers are unable to respond to consecutive probes, ACE removes them from the server farm rotation and periodically checks back with the server to determine whether it is back online. This section describes how to configure various types of probes to help ACE optimize the server selection process.

The goal in configuring a health probe is to determine whether the server is up and available to service incoming requests. A successful ping, for example, shows that the machine and the operating system are up and that the network connection is good. However, the SAP service must also be running to service requests. To test this, ACE can be configured to send HTTP or SSL probes and to evaluate either the response code or a hash of the web page itself. In many cases, the receipt of a 200OK is an adequate measure for verifying server health. As discussed below, some URLs work better than others as a reliable probe.

To check the state of the SAP service, open the SAP Management console window (sapmmc) as shown in Figure 3 for the SAP ERP CI. All the icons are green, which indicates the service is running properly.

Figure 3 SAP Management Console—Healthy State

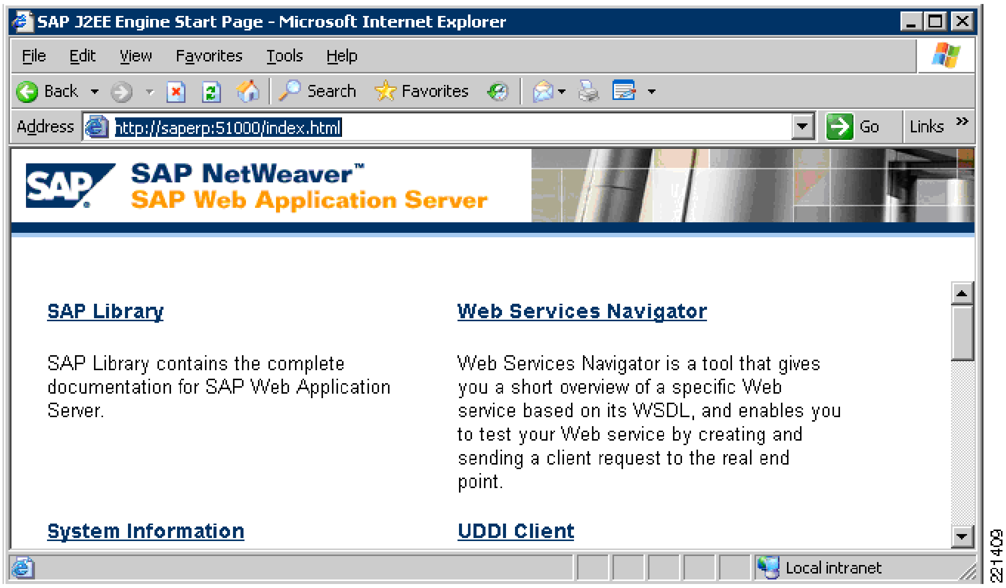


The link to the ERP home page is http://saperp:51000/index.html. A probe to this link is configured on ACE as follows:

```
probe http erp
  port 51000
  interval 2
  faildetect 2
  passdetect interval 2
  request method get url /index.html
  expect status 200 200
```

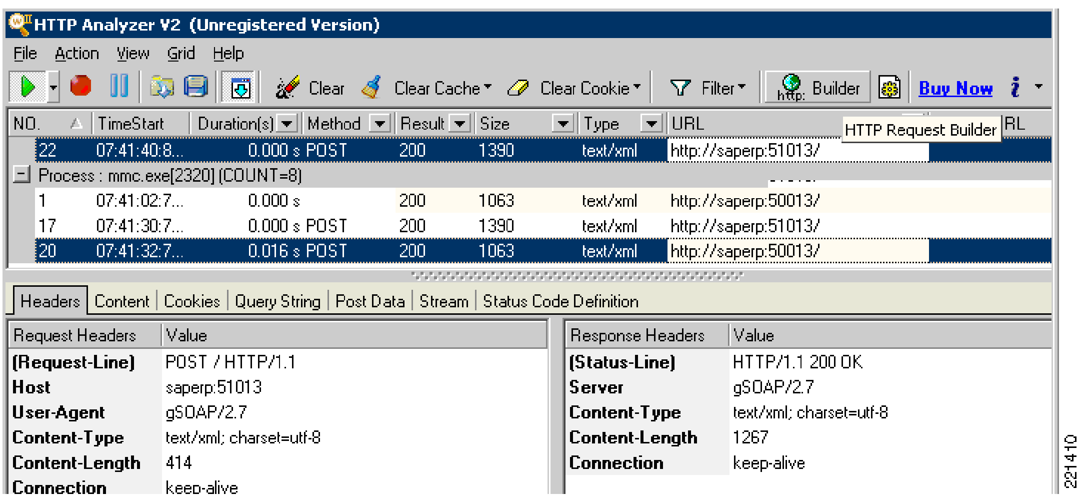
When the probe is successful, it returns the window shown in Figure 4.

Figure 4 ERP Successful Index Window



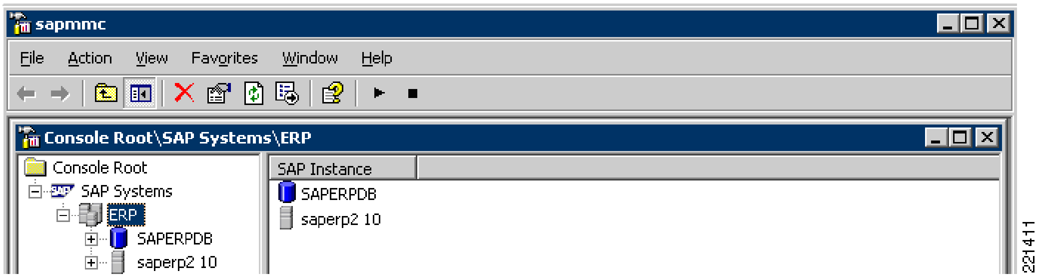
In this case, ACE is looking for the status code, which is 200OK as shown in Figure 5.

Figure 5 200OK



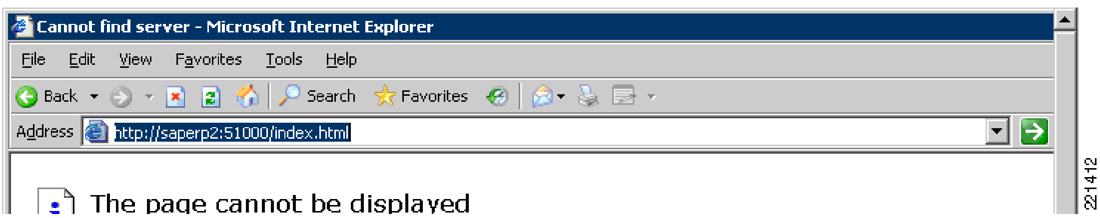
If the ERP service is not running, this probe fails. Consider the saperp2 host shown in Figure 6 where the ERP service console shows the ERP service greyed out in a stopped state.

Figure 6 SAP Management Console—ERP Service Down



When the same probe is run on this host, it fails and the server is taken out of rotation, as shown in Figure 7.

Figure 7 Index Window when ERP Service is Down




```
ACE-1/sap# sh probe ep

 probe       : ep
 type        : HTTP, state : ACTIVE
----------------------------------------------
   port      : 51000   address     : 0.0.0.0         addr type  : -
   interval  : 2       pass intvl  : 2               pass count : 3
   fail count: 2       recv timeout: 10

--------------------- probe results ------------
   probe association   probed-address  probes   failed   passed   health
   ------------------- ---------------+----------+----------+----------
   serverfarm  : ep
     real      : ep[51000]
                       12.20.57.10     617257   251204   366053   FAILED
```

Indirect Application Failures

Other cases are more ambiguous. For example, the service is running but in an error state (shown as yellow on the SAP console) when the connection is lost to the database. The ability to reply to the probe depends on the URL that is used. Some pages, such as http://saperp:51000/index.html used above, continue to respond to probes and fail to reflect the true state of the server. As a result, it is better to use a page that depends on a successful database lookup to return a 200 OK. That is the case with the initial login screen of the ERP server (http://saperp:51000/irj/portal). When this URL is used, a database failure results in an unsuccessful probe and the server is taken out of rotation.

If in fact the database has failed, all the application servers are unable to service requests, and the traffic must either be routed to a backup site or a sorry server. Thus, there is the need for a backup server farm definition in the ACE. This is a policy that goes into effect when all the servers are considered unavailable.

If it is not possible to find a URL that depends on a database lookup for a successful probe, ACE has a facility for “out-of-band” probes that rate server availability based on the health of an external resource such as the database server.

In the following example, a second probe is created that contacts the database directly rather than using the implied approach above (db). This is a TCP-based probe that connects to port 1433 on the database server. Like the ep probe, it is applied to the server farm. However, where the ep probe inherited the IP address properties of the server farm, the database probe specifies the IP address of the database server that is outside of this server farm. The **routed** keyword is necessary for this to work.

```
probe tcp db
  ip address 12.20.53.14 routed
  port 1433
  interval 2
  faildetect 2
  passdetect interval 2
  passdetect count 2

serverfarm host ep
  probe ep
  probe db
  rserver ep 51000
    inservice
  rserver ep1 51000
    inservice
  rserver ep2 51000
    inservice
```

If the db probe fails, all the servers are taken out of service:

```
ACE-1/sap# sh server ep
 serverfarm     : ep, type: HOST
 total rservers : 3
 ---------------------------------
                                                      --connections--
       real                  weight state        current    total
   ---+---------------------+------+------------+----------+-----------
   rserver: ep
       12.20.57.10:51000     8      PROBE-FAILED 0          1
   rserver: ep1
       12.20.57.11:51000     8      PROBE-FAILED 0          1
   rserver: ep2
       12.20.57.12:51000     8      PROBE-FAILED 0          0
```

Future connections are not sent to these servers until the probes are successful. By including the backup statement in the loadbalance policy map, new connection requests are sent to the backup server farm.

```
serverfarm host ep-backup
  probe ep
  rserver ep 51000
    inservice
  rserver ep1 51000
    inservice
  rserver ep2 51000
```

This is shown with the load balancing policy configuration below. There are four parts to building a basic load balancing policy:

- VIP class map—Specifying the virtual IP address with any port restrictions
- Loadbalance policy map—Mapping a traffic class to primary and backup server farm
- Multi-match policy map—Mapping the load balance policy to the VIP and set VIP options

- Service-policy—Binding the policy to the interface that receives the candidate traffic

```
class-map match-all basic-slb-vip
  3 match virtual-address 12.20.99.202 tcp any
policy-map type loadbalance first-match basic-slb
  class class-default
    serverfarm ep backup ep-backup aggregate-state
policy-map multi-match SLB-policies
  class basic-slb-vip
    loadbalance vip inservice
    loadbalance policy basic-slb
    loadbalance vip advertise active
interface vlan 962
  description client VLAN
  service-policy input SLB-policies
```

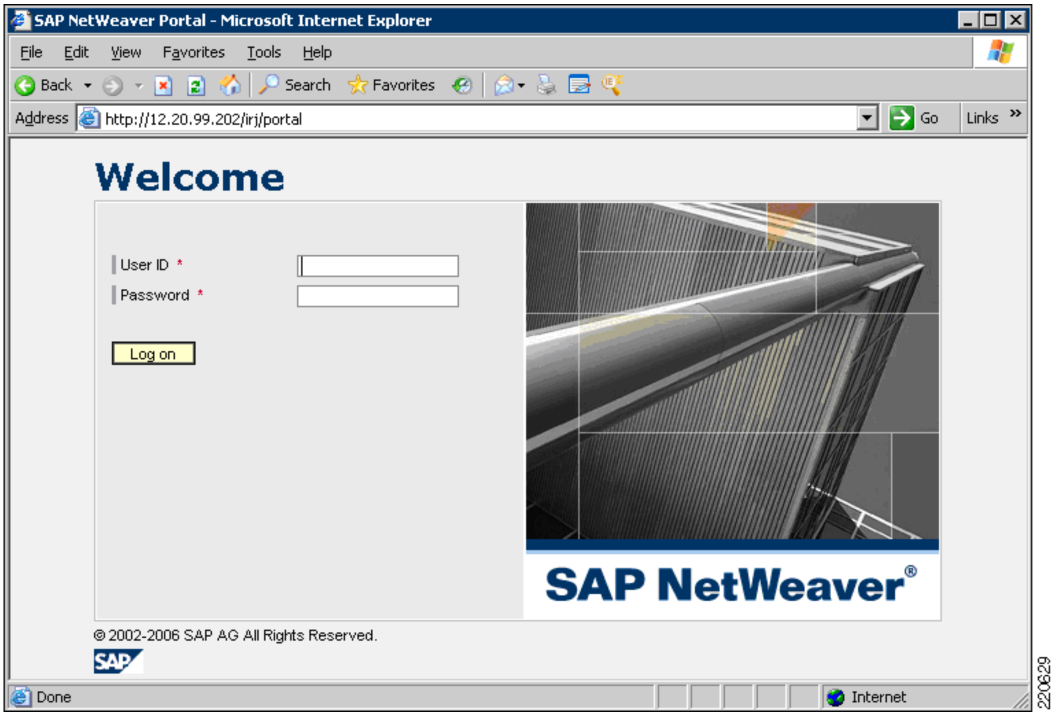
Note In this case, a server farm called "ep-backup" is defined, with the ep probe but without the db probe. It uses the same ep servers for illustration purposes (the IP address of the database probe was changed in this example, so that the real database is still available.)

```
ACE-1/sap# sh server ep-backup
 serverfarm     : ep-backup, type: HOST
 total rservers : 3
 ---------------------------------
                                                        --connections--
       real                  weight state        current    total
   ---+---------------------+------+------------+----------+----------
   rserver: ep
       12.20.57.10:51000     8      OPERATIONAL  0          0
   rserver: ep1
       12.20.57.11:51000     8      OPERATIONAL  0          0
   rserver: ep2
       12.20.57.12:51000     8      OPERATIONAL  2          2
```

SAP and Session Persistence

This basic load balancing setup is not sufficient for a working system because persistence is not defined. Without persistence, every click gets assigned in round robin fashion to a different server. HTTP is a stateless protocol, but the session with an SAP server must be stateful. Cookies are used by the server to help load balancers maintain this state. This can be seen with an attempt to access the Employee Self-Service utility. The first screen the user sees when connecting to the SAP Enterprise Portal (EP) is the logon screen shown in Figure 8.

Figure 8 SAP Enterprise Portal Logon Screen

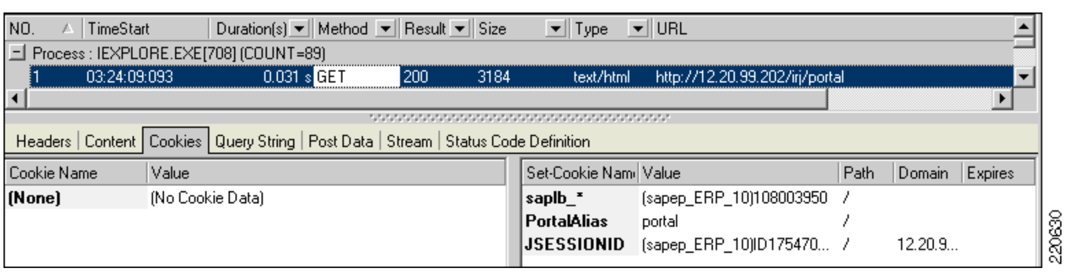


It takes 12 HTTP requests to render this page. By analyzing the individual requests, you can see to which server you are connected and how cookies are set with SAP. SAP sets various cookies throughout a session.

The first two set-cookie messages, saplb_* and JSESSIONID, come in the very first response. These cookies include the SAP DNS name for that server: sapep, sapep1, or sapep2. One of these names is shown in the cookie. The next set-cookie message, MYSAPSS02, comes in request 13 after the user has entered login credentials.

The analysis in Figure 9 shows the cookie-related details of HTTP request #1, with the client request on the left and the server response on the right.

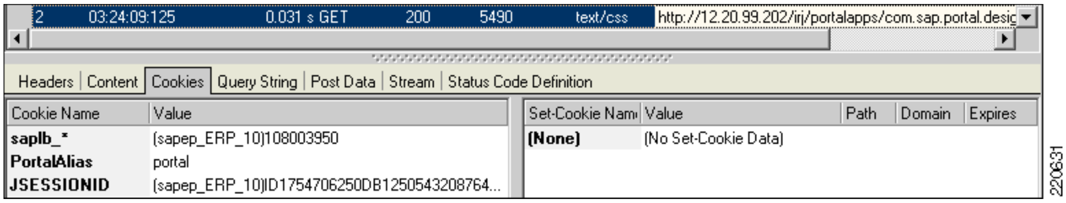
Figure 9 Cookie Analysis for HTTP Request #1



The client comes in with no cookie set and receives two set-cookies: "saplb_*" and "JSESSIONID". Note that both these responses contain the hostname "sapep". This indicates that ACE has sent the request to the sapep server.

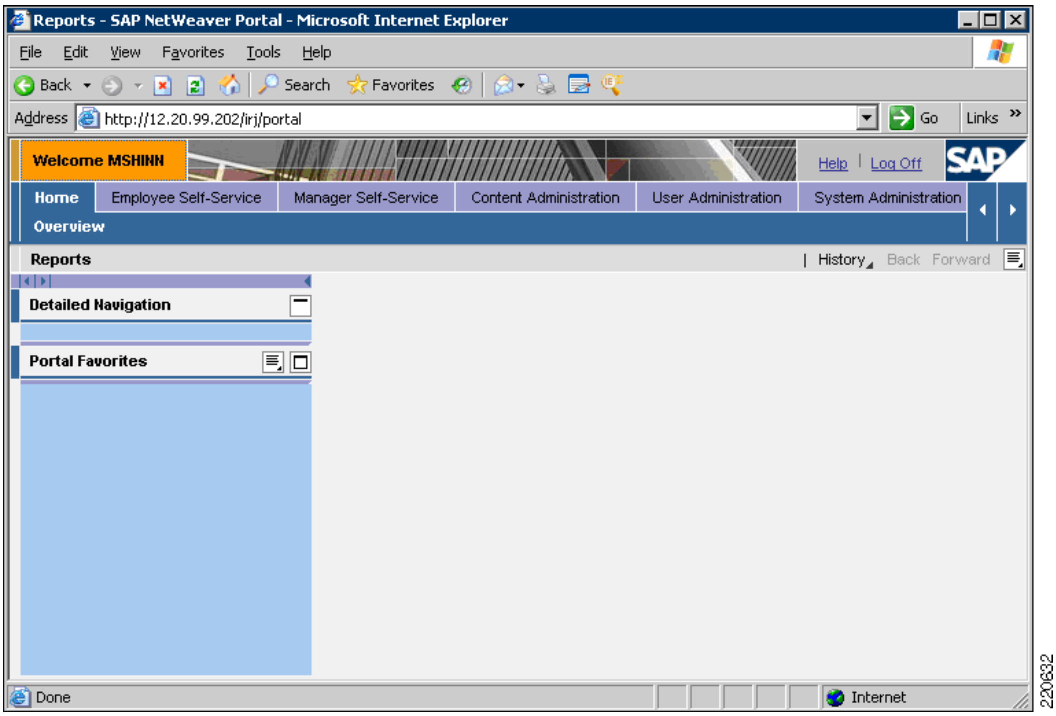
Message #2 indicates that the client now has incorporated both cookies, as shown on the left of Figure 10.

Figure 10 Message #2



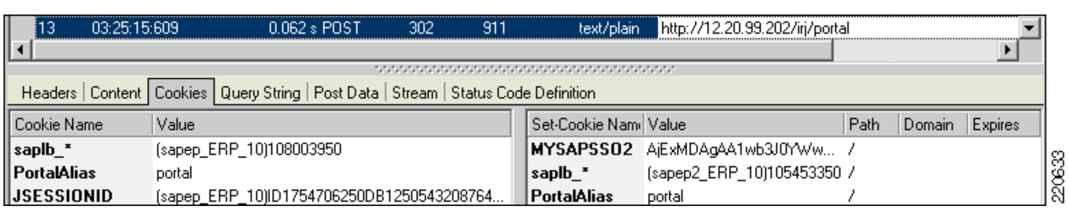
This continues on to the next event, where the user enters the username and password and clicks **OK** to login. What follows are HTTP requests 13–88, which render the home page of the portal (see Figure 11).

Figure 11 Portal Home Page



A closer look at message 13 shows that the client comes in with the cookie for sapep, the server it visited originally, yet it is getting a new set-cookie request from sapep2, as can be seen by the set-cookie for saplb_* (see Figure 12).

Figure 12 Message 13

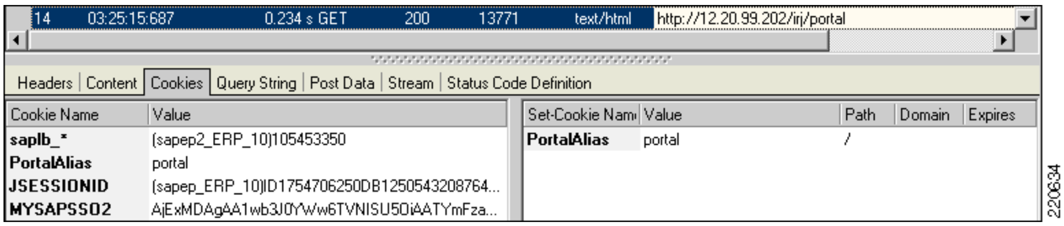


This request has gone to sapep2, which recognizes that the saplb_* cookie is not right, so it sends a new set-cookie message for itself. Note that a new cookie has not been requested for JSESSIONID, while an entirely new cookie (that does not reflect the hostname) is being requested for MYSAPSS02, the cookie that SAP issues after a successful login.

The client is now at an entirely different server, sapep2, that has just been sent the login information that was intended for the original sapep server. At this point, SAP has not complained; it is unusual that the login information did not come in with the right cookie set, but the server sets a new saplb_* cookie and logs the client in, because the username and password are valid.

Requests 14–88 follow, which all have the new saplb_* cookie set in the client browser for sapep2, as shown by the example of message 14 shown in Figure 13.

Figure 13 Message 14

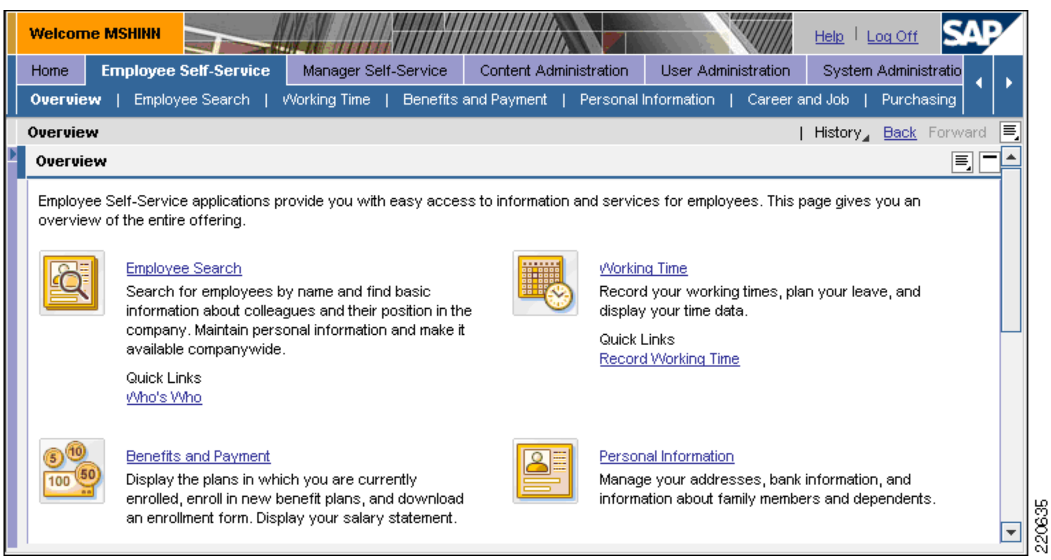


The JSESSIONID cookie has not changed (still indicating the original host), and the new MYSAPSS02 cookie is now set in the browser.

This continues until the next click on Employee Self-Service, which occurs at request 89.

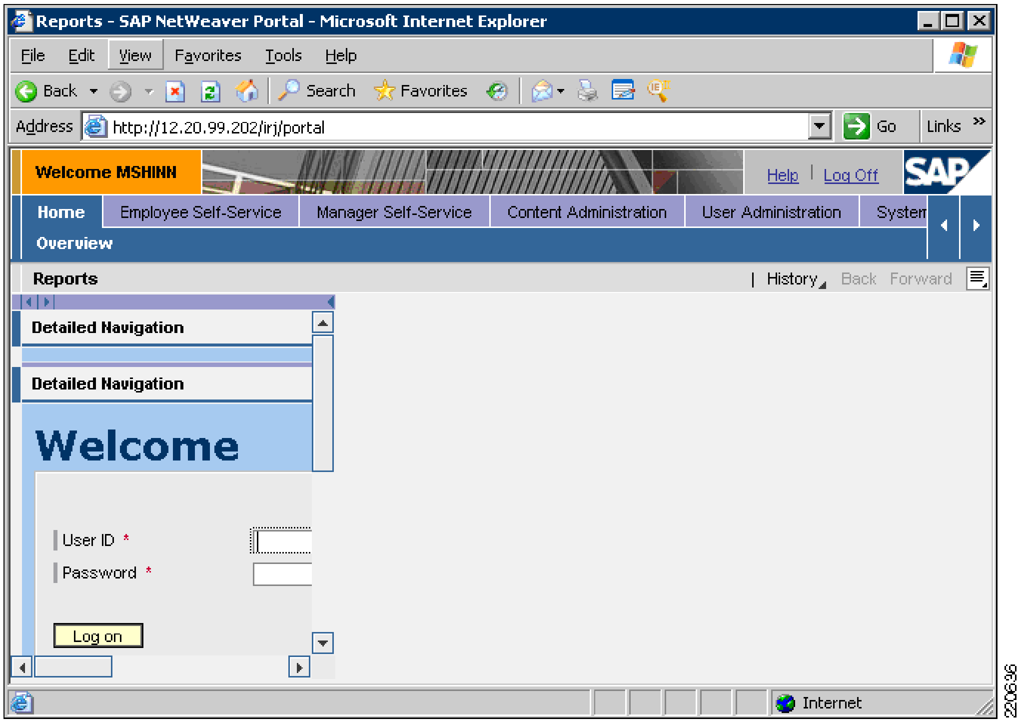
Figure 14 shows what the portal is supposed to return when you click on Employee Self-Service.

Figure 14 Employee Self-Service Screen



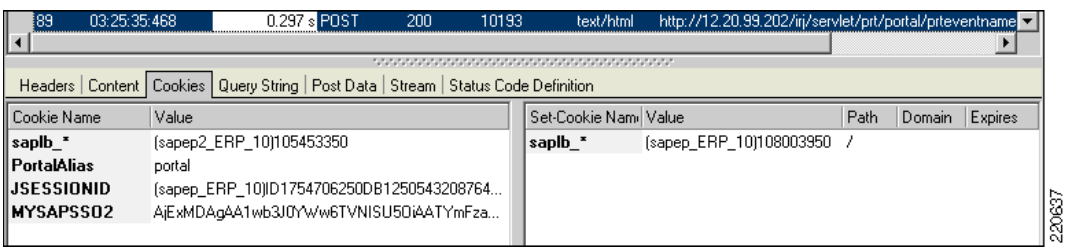
However, the return screen is instead a confused page where the login screen seems to be in the navigation pane of a window seen by a user already logged in (see Figure 15). At this point, the session is completely broken.

Figure 15 Broken Session



A look at the offending request shows what happened (see Figure 16). The client is now connected back to sapep (as indicated by the saplb_* set-cookie), which has no knowledge of the successful login sent to sapep2.

Figure 16 Message Analysis



There clearly needs to be persistence so that client requests stay with the same server throughout the session. There are a number of mechanisms to do this, as described in the next section.

Source IP Persistence

Session persistence can be established by tying the session to an IP address, or by a cookie that uniquely identifies the client.

For persistence based on source IP, a few modifications are required to the configuration shown in the earlier steps:

- Create a sticky-group

```
sticky ip-netmask 255.255.255.255 address source ep-sourceIP-sticky
```

```
  timeout 10
  serverfarm ep backup ep-backup aggregate-state
```

- Change the server farm to the sticky-group:

```
policy-map type loadbalance first-match basic-slb
  class class-default
    sticky-serverfarm ep-sourceIP-sticky
```

Note Note that the backup statements have moved from the load balance policy map configuration to the sticky-group configuration.

To see which server a given client is stuck to, enter the following:

```
ACE-1/sap# show sticky data client 12.20.15.15
sticky group : ep-sticky
type         : IP
timeout      : 10            timeout-activeconns : FALSE
  sticky-entry       rserver-instance       time-to-expire flags
  ------------------+---------------------+-------------+-------+
  202641167             ep1:51000             599             -
```

This matches the hostname shown in the client cookie:

```
Cookie: saplb_*=(sapep1_ERP_10)105453250; PortalAlias=portal;
```

Cookie Persistence

Persistence based on the client IP address works well unless there is a proxy. In that case, all the client connections appear to have the same IP address and are all sent to the same server, defeating the purpose of load balancing. In this type of situation, it is better to use a cookie for persistence. ACE can dynamically learn the cookie *value* from the server when the cookie *name* is specified.

The question is which cookie to use. SAP uses multiple cookies throughout a session: saplb_*, JSESSIONID, and MYSAPSS02 are all seen in the example above. These cookies are applied at different times. MYSAPSS02 is applied when the user submits login credentials. JSESSIONID and saplb_* are both applied before the credentials are submitted at the initial user logon. The difference, however, is that the server always checks the hostname contained in the saplb_* cookie to see whether it matches the server hostname. If it does not, it resets the cookie. However, after the JSESSIONID cookie is set, the server does not reset it. It is deleted only if the browser is closed, in which case the server assigns it a new one at the beginning of the session.

As a result, the saplb_* cookie works best for ensuring that the session is sent to the right server. As it turns out, this cookie is also designed for load balancers. For more information, see the following URL: http://help.sap.com/saphelp_erp2005vp/helpdata/en/f2/d7914b8deb48f090c0343ef1d907f0/frameset.htm. For troubleshooting purposes, it can be useful to compare the saplb_* and the JSESSIONID hostnames. You can rely on saplb_* to reliably tell you to which server you are currently connected, while JSESSIONID tells you only the server into which you initially logged. If the server does change, either because persistence was not configured or the cookie was aged out of the sticky database, the discrepancy between saplb_* and JSESSIONID enables you to see what happened.

In this example, saplb_* is selected for the cookie name to track persistence. The following shows a new configuration where ACE sticks all requests to the server based on the saplb_* cookie setting. The main differences from the basic server load balancing setup are the following:

1. A sticky server farm is created by mapping the saplb_* cookie name to server farm "ep". This causes the associated cookie value to be learned dynamically and to stick the session.
2. Cookie sticky groups require L7 traffic classification. Previously, traffic was made eligible in the load balance policy map by simply using the class-default catch-all, which applies to any traffic. In this case, the traffic needs to be specified at L7. To do this, there is a second class map that specifies a match on any (.*.) URL.
3. The load balance policy map references the URLs in this class and ties them to the sticky server farm instead of the server farm directly.

For the cookie sticky to work properly on the server farm ep, it should be the only sticky group associated with it. To configure a new cookie-based sticky group with the ep server farm, the source-IP sticky connection to ep must be removed:

```
sticky ip-netmask 255.255.255.255 address source ep-sticky
timeout 10
no serverfarm ep
```

Instead, a new server farm (ep-cookie) is created, with the same servers, that maps to a new VIP address, 12.20.99.203, so that both methods can be demonstrated depending on which VIP is used. As a result, there is also a new class added to the SLB-policies multi-match map.

```
sticky http-cookie saplb_* ep-cookie
  timeout 10
  replicate sticky
  serverfarm ep-cookie backup ep-backup aggregate-state

class-map match-all epL7-vip
  match virtual-address 12.20.99.203 tcp any

class-map type http loadbalance match-any epL7
  match http url .*.

policy-map type loadbalance first-match epL7-policy-1
  class epL7
   sticky-serverfarm ep-cookie

policy-map multi-match SLB-policies
  class epL7-vip
    loadbalance vip inservice
    loadbalance policy epL7-policy-1
    loadbalance vip advertise active

interface vlan 957
  description EP-server
  ip address 12.20.57.2 255.255.255.0
  alias 12.20.57.1 255.255.255.0
  peer ip address 12.20.57.3 255.255.255.0
  no normalization
  access-group input anyone
  no shutdown
interface vlan 962
  description client VLAN
  ip address 12.20.62.2 255.255.255.0
  alias 12.20.62.1 255.255.255.0
  peer ip address 12.20.62.3 255.255.255.0
  no normalization
  service-policy input SLB-policies
```

```
  access-group input anyone
  no shutdown
```

With this configuration, all requests stick to the server originally selected and the session remains intact. An important factor to consider is the length of time the cookie is stored in the cookie database. By default the timeout is 10 minutes, shown in the cookie database in seconds.

```
ACE-1/sap# show sticky data
sticky group : ep-cookie
type         : HTTP-COOKIE
timeout      : 10            timeout-activeconns : FALSE
  sticky-entry          rserver-instance                 time-to-expire flags
  ---------------------+-------------------+--------------+-------+
  6394345763420148261   ep2:51000          598            -
```

After the cookie times out, the next connection is load balanced to a new server. By default, the timeout is ten minutes. This means that if someone walks away from their desk for a few minutes, when they come back, it may appear as if the system is broken, as was seen from Figure 15 where persistence was undefined. The timeout can be changed to any value up to 65,535 minutes, as follows:

```
ACE-1/sap(config)# sticky http-cookie saplb_* ep-cookie
ACE-1/sap(config-sticky-cookie)# timeout ?
  <1-65535>    Enter the timeout value in minutes
```

Persistence and High Availability

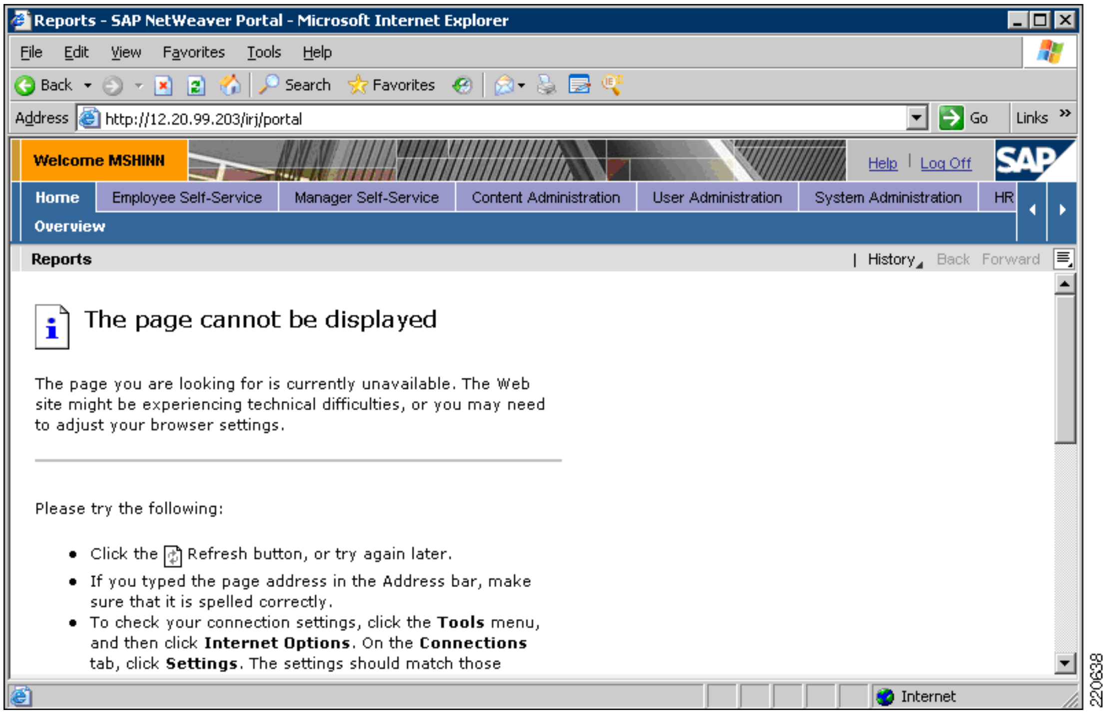
The cookie database is maintained between ACE modules so that even in the event of maintenance or a failure, the sessions are maintained correctly. The following shows the same cookie replicated across both ACE modules.

```
ACE-1/sap# show sticky data
sticky group : ep-cookie
type         : HTTP-COOKIE
timeout      : 10            timeout-activeconns : FALSE
  sticky-entry          rserver-instance      time-to-expire    flags
  ---------------------+----------------------+--------------+-------+
  6394345763420148261   ep:51000                447            -


switch/sap# show sticky data
sticky group : ep-cookie
type         : HTTP-COOKIE
timeout      : 10            timeout-activeconns : FALSE
  sticky-entry          rserver-instance      time-to-expire   flags
  ---------------------+-----------------------+--------------+-------+
  6394345763420148261   ep:51000                591            -
```

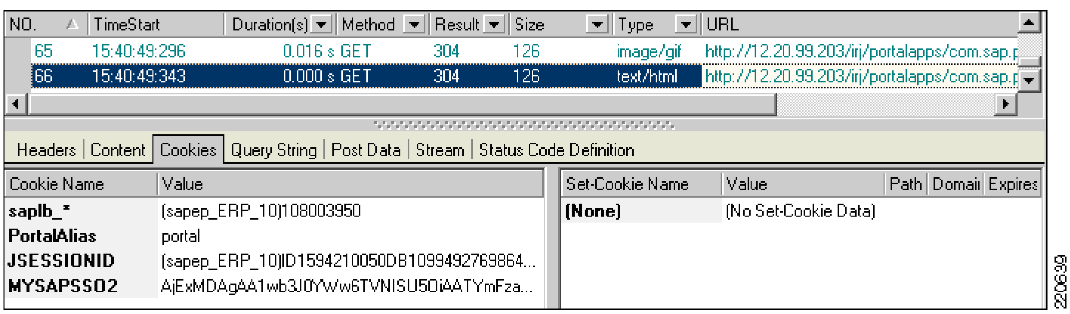
As a result, when ACE-1 fails, ACE-2 takes over the session and maintains the same sticky database. The testing showed that immediately after the ACE-1 failure, the first click failed. (See Figure 17.)

Figure 17 First Click Failure



However, the next click is successful and the session remains stuck to the ep server. (See Figure 18.) This provides for the maintenance of session persistence even in the case of an ACE module failure.

Figure 18 Session Remains Stuck to EP Server



Application Analysis

While ACE is primarily concerned with the management and scalability of the application, WAAS improves the response time to end users and/or to improve the bandwidth utilization on WAN links. In evaluating the potential for improving application performance, it is useful to know where the problem area is; bandwidth, latency, congestion, protocol performance, and so on. The Cisco Application Analysis Solution (AAS) is a useful tool for this (see http://www.cisco.com/en/US/products/ps6362/prod_bulletin0900aecd80582266.html).

Cisco AAS provides an in-depth look at the traffic patterns between client and server.

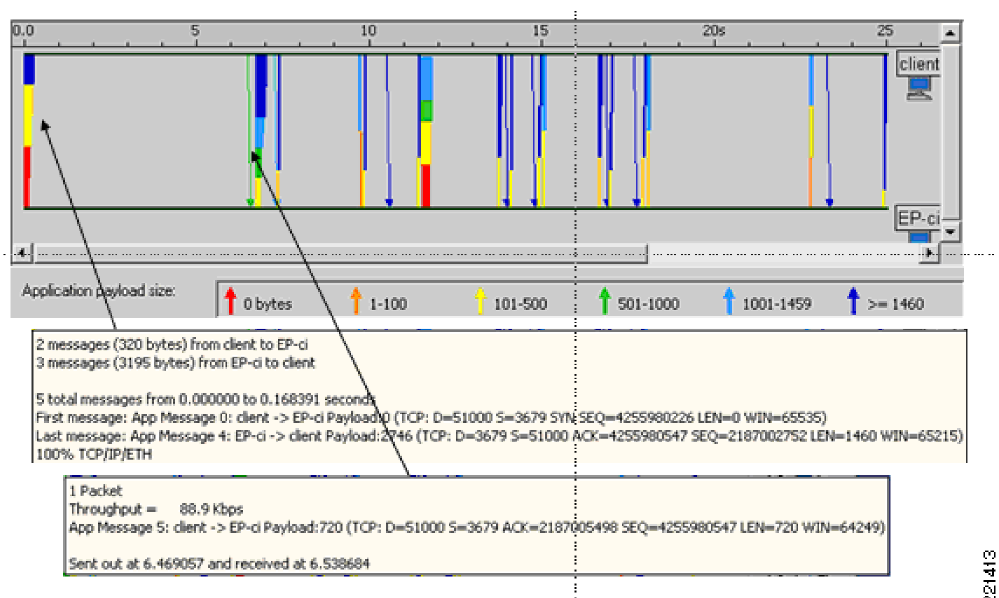
The AAS collects application traffic traces from multiple sources such as the Cisco Network Analysis Module as well as software agents that can be installed on each device in the path of the transaction. In theory, this process can account for all components involved in a transaction. For example, numerous hosts are involved to complete an SAP employee lookup transaction: the client, the EP central instance, EP dialog instance, EP database server, as well as the ERP CI/DI and the ERP database server.

Ideally, an agent is placed on each host, and the transaction can be traced from beginning to end. In practice, however, the application server and the database server are constantly chatting back and forth, and AAS is unable to correlate which messages relate to the client session being monitored. So in this case, only the client and application server probes are merged, and the backend communication between application server, ERP server, and database servers are not explicitly accounted for.

The traffic pattern shown in this example is a web client connecting to the SAP Enterprise Portal (EP) website and accessing the Employee Self Service application to do an employee lookup. The bandwidth is T1 and the one-way delay is 52 ms. The complete transaction requires several clicks, so when the total response time is measured, it reflects time for thinking on the part of the user as well as network delay and application processing delay. The response time for the entire transaction was measured at 25.1 seconds with 135.2 KB of application data transferred.

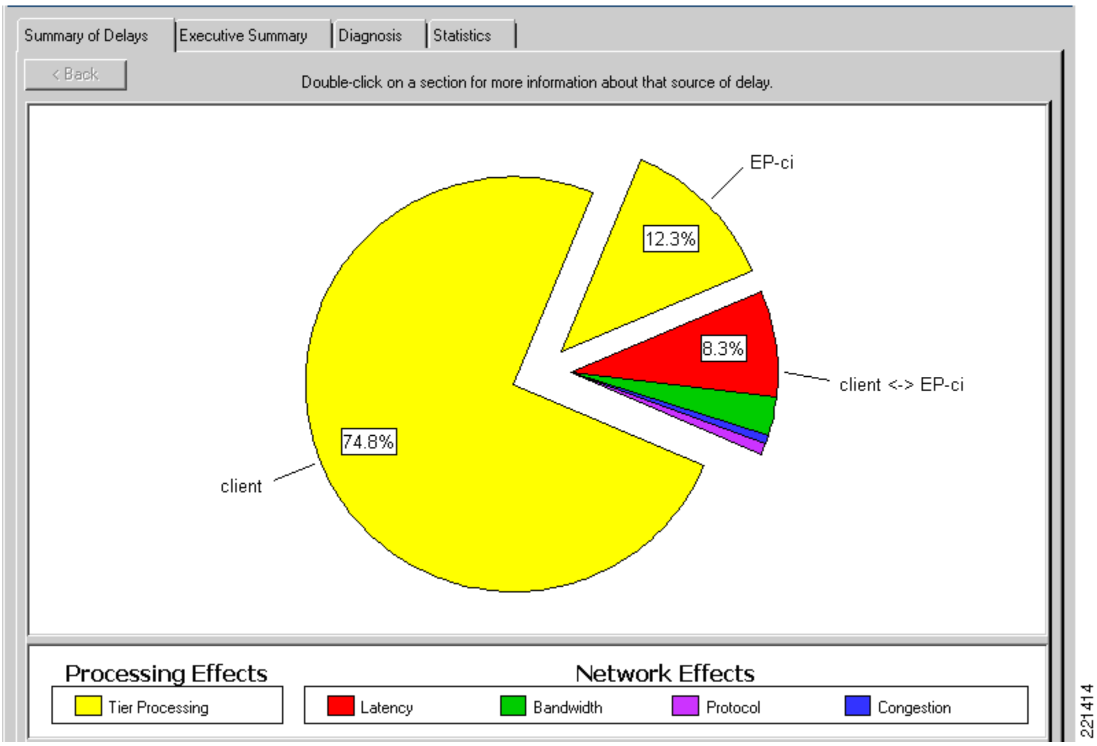
AAS provides an examination of the performance aspects of the transaction. First, the delay chart shows a timeline of when each message occurs. The 25-second ESS transaction is displayed visually as a series of message bursts interspersed with processing time on the server and think time for the client. The colored lines represent the messages; the thicker the line, the more messages. If a line has an arrow, it indicates a single message. The slant of the arrow indicates the delay from when it leaves until it arrives. When the mouse is positioned over the message lines, the details are presented as shown in Figure 19. This provides a picture of what is occurring over the duration of the transaction time.

Figure 19 SAP ESS Transaction—Message Timeline



From the gaps between the messages shown above, it is clear that there is a good percentage of time when no packets are traversing the network and either the client or server is "processing" (see Figure 20). AAS includes an Application Doctor that shows who is responsible for each component of the delay. It shows that by far the biggest delay is "think time" by the client; 74.8 percent of the total. This number varies depending on how the user interacts with the application, such as pausing to talk on the phone or getting a cup of coffee. All these human variables factor into this number. Processing delay by EP, including all the database and inter-process communications, is the next biggest component at 12.3 percent. Network latency, the time it takes for packets to traverse the network, is 8.3 percent, and bandwidth accounts for approximately 3 percent. Protocol processing is less than 1 percent. This leaves approximately 15 percent of the total delay that can be tuned in the network.

Figure 20 SAP ESS Transaction—Summary of Delays

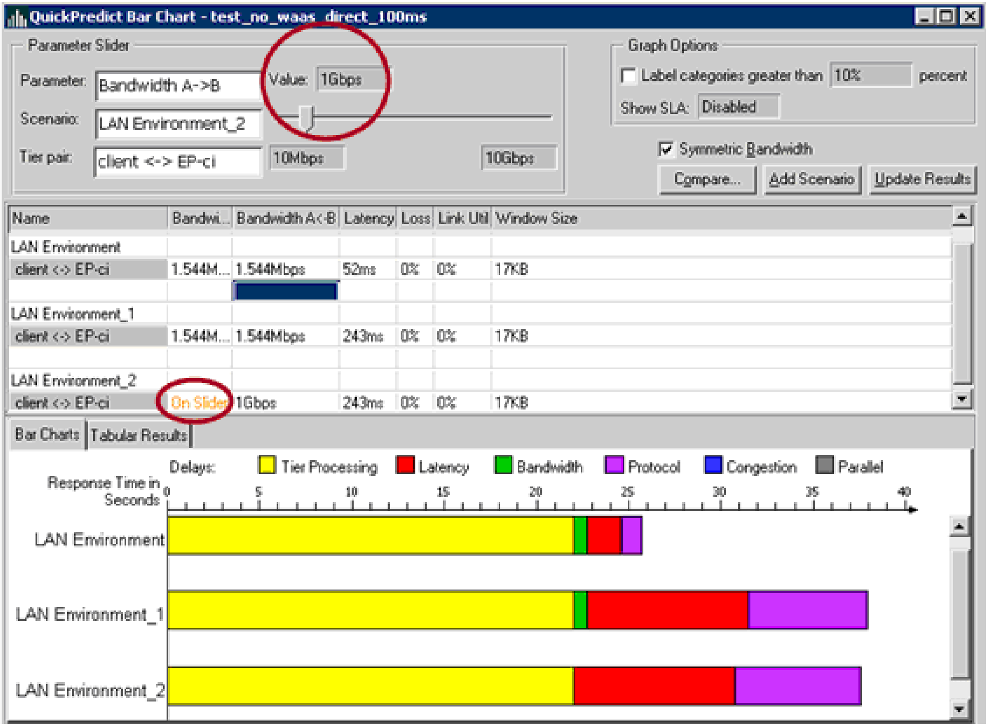


The quick predict bar tool provides a way to create scenarios with different values for bandwidth and delay and to calculate the effect on the total transaction time. This is useful, for example, if testing is performed at a central site that captures the basic application behavior, and estimates are required for remote sites with different bandwidth and latency values.

Figure 21 shows the following three scenarios:

- LAN Environment has the baseline values as measured in the pie chart shown in Figure 20, except that it shows the time impact in terms of seconds rather than a percentage. Client and server tiers account for 22 seconds of the total, while bandwidth contributes somewhat less than a second and latency approximately two seconds.
- In LAN Environment_1, the latency is increased to 243 ms and transaction delay goes to 38 seconds.
- In LAN Environment_2, the latency is kept at 243 ms and bandwidth is increased to 1 Gbps. Note that the slider at the top can be used to see the effect in real-time of increasing the bandwidth or the latency variable on total transaction time.

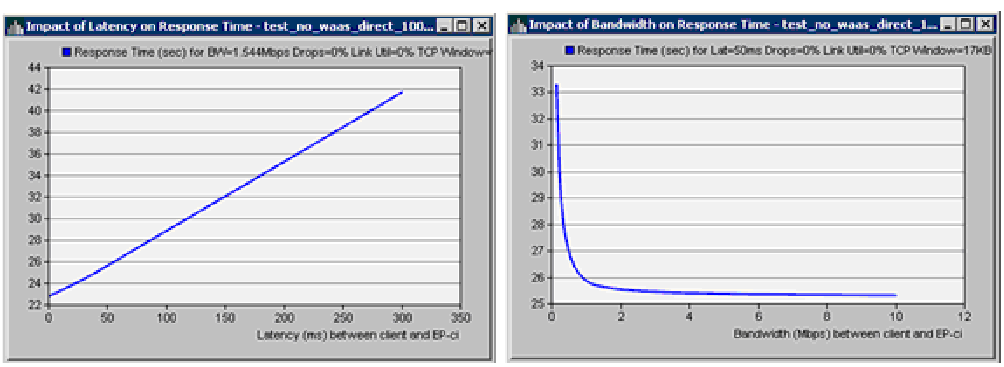
Figure 21 SAP ESS Transaction—Sensitivity Analysis



221415

Figure 22 shows how bandwidth and latency affect the transaction time. The transaction time is shown on the vertical access as a function of increasing values of latency on the left and bandwidth on the right. Latency is linear, while bandwidth quickly shows diminishing returns. (This does not account for the congestion latency that occurs when there is not enough bandwidth because of contention with other sessions.)

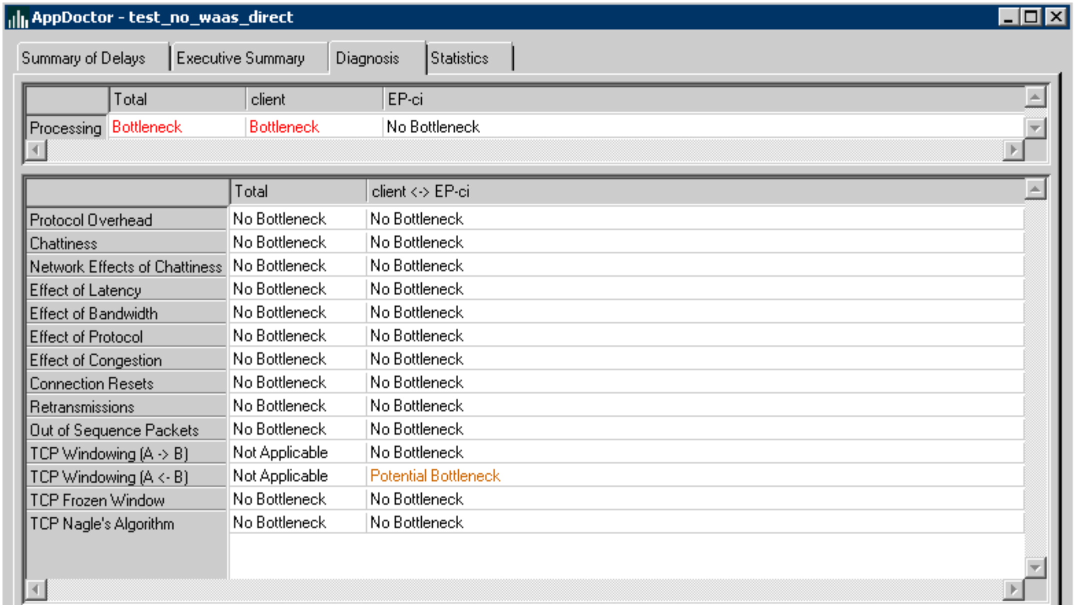
Figure 22 SAP ESS Transaction—Bandwidth and Delay Impacts



221416

The Application Doctor also provides a very granular look at potential bottleneck points in the network, considering a range of variables. As shown in Figure 23, TCP windowing is an area for improvement.

Figure 23 SAP ESS Transaction—Bottleneck Identification

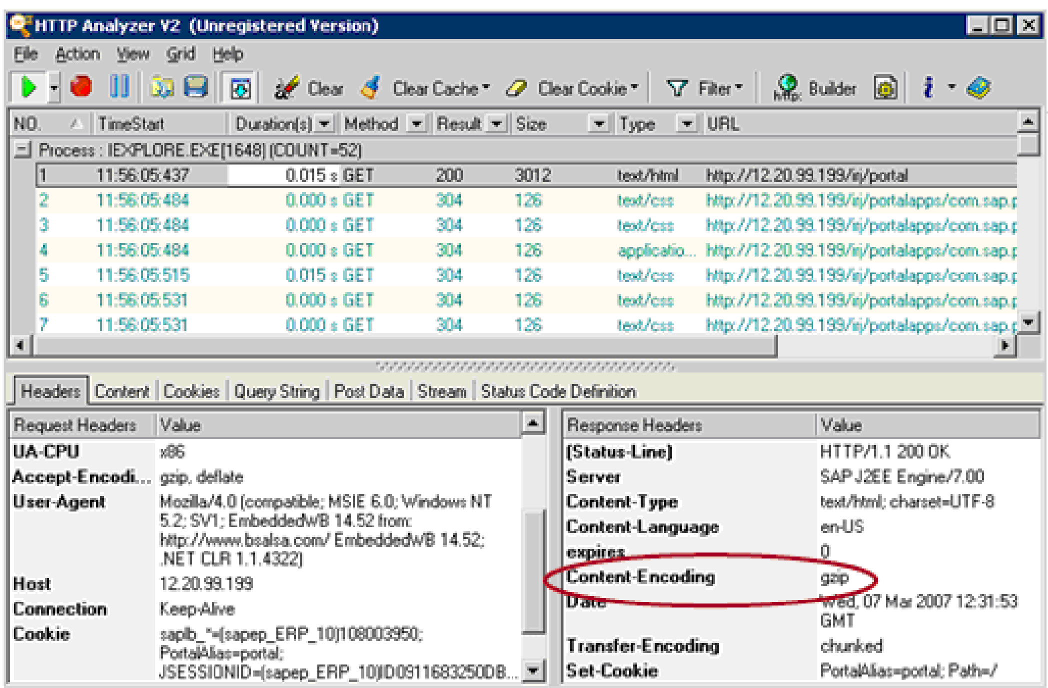


221417

Compression

Another factor that affects the ability to improve application performance is compression. Figure 24 shows an analysis of an HTTP response from the EP server. It shows that the EP server is already performing gzip compression on the data.

Figure 24 HTTP Response from the EP Server



183880

WAAS Network Integration

One of the options for further optimizing the SAP session is WAAS, using techniques such as Data Redundancy Elimination and TCP optimization. A WAAS deployment is symmetrical, meaning there is a Wide-Area Application Engine (WAE) at each end of the transaction. The SAP servers are located in the data center and the users in most need of optimization are located out in the branch offices. At the branch office, the most common solution is a WAE either as a standalone device or an integrated router module that intercepts traffic using WCCP. At the head end, the WAEs can be located in the data center or at the WAN edge.

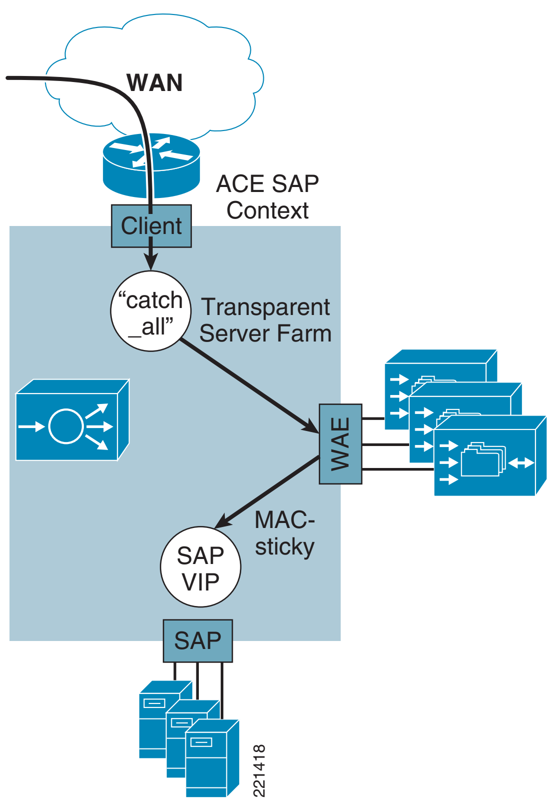
In most cases, there is more than a single WAE at the point of aggregation for a branch office network. At a minimum, there are two for redundancy, and for a large branch office network, there may need to be a large farm of WAEs to handle the load from all the remote sites. Aggregating multiple WAEs at the central site can be done with WCCP or a server load balancing device such as ACE. Both these options were tested with the SAP environment, and each have advantages. WCCP has been used for years and is well understood. Its chief limitation is that it scales only to 32 devices. For very large-scale deployments, ACE provides an alternative that scales to hundreds or thousands of WAEs.

To use ACE for scaling WAAS at the head end, consider the two scenarios described in the following sections.

WAAS and ACE in the Data Center—Overview

Figure 25 shows a solution for deploying WAAS in the data center.

Figure 25 WAAS in the Data Center



User sessions come in from the WAN, go through the data center core, and then into an ACE context located in the aggregation switch. The ACE context shown is a routed context with three interfaces: client, WAE, and SAP. ACE is the default gateway for the WAE and SAP VLANs. The session consists of a client at the branch office connecting to the virtual IP (VIP) address of the SAP server farm. ACE sends the session to the best available WAE to be optimized, and then on to an SAP server for application processing.

Following are some of the details of this session flow:

1. The client at a remote branch (not shown) browses to the SAP Enterprise Portal (EP). This is a farm of several SAP servers that DNS resolves to a single VIP on the ACE at the data center. The branch office router intercepts the session with WCCP and forwards it to the branch office WAE. The WAE sets the TCP options and forwards it.
2. When the packet arrives at ACE, a service-policy, “waas_policy”, located on the “client” interface redirects any inbound traffic to the WAE farm. At this point, ACE normally translates the destination address to the selected real server (WAE in this case) address. However, because the WAE server farm is defined as “transparent”, ACE does not perform destination NAT and the VIP address is preserved.
3. The core WAE selected by the ACE load balancing algorithm records the packet TCP options, adds its own identifier as the last WAE to inspect the packet, and returns the packet to the ACE on the WAE VLAN interface. This interface is configured with “mac-sticky” so that ACE remembers which WAE sent the packets to it. This is required because the return traffic must traverse the same WAE on which it came in to be optimized.
4. The “SLB_policies” service-policy on the WAE interface references the SAP load-balancing policy, selects the best SAP server, uses NAT for the destination address to it, and forwards the traffic.
5. The SAP server processes the request and sends the response back to ACE without any TCP options attached. The “waas_policy” service-policy configured on the client interface is also configured on the SAP interface. At this point, ACE normally forwards the traffic to the best available WAE. However, because mac-sticky was applied at the WAE interface, the traffic is forwarded back to the same WAE.
6. The core WAE returns the packet back to ACE, where it is then routed back through the branch router and branch WAE, and ultimately back to the client.

WAAS and ACE in the Data Center—Configuration

In terms of changes to the ACE SAP context configuration, introducing the WAAS element involves the following procedure.

Step 1 Create a WAAS server farm and probe:

```
probe icmp waas
  interval 10
  faildetect 2
  passdetect interval 2

rserver host waas-5
  ip address 12.20.29.5
  inservice
rserver host waas-6
  ip address 12.20.29.6
  inservice
rserver host waas-7
  ip address 12.20.29.7
  inservice
```

```
rserver host waas-8
  ip address 12.20.29.8
  inservice

serverfarm host waas
  transparent
  predictor leastconns
  probe waas
  rserver waas-5
    inservice
  rserver waas-6
    inservice
  rserver waas-7
    inservice
  rserver waas-8
    inservice
```

Step 2 Create WAAS VIP, load balancing, and multi-match policies:

```
class-map match-all waas
  2 match virtual-address 0.0.0.0 0.0.0.0 tcp any
policy-map type loadbalance first-match waas-policy
  class class-default
    serverfarm waas
policy-map multi-match waas-policy
  class waas
    loadbalance vip inservice
    loadbalance policy waas-policy
```

Step 3 Move the SAP EP policy from the client interface to the WAAS interface and apply the WAAS policy to the client and EP server interfaces:

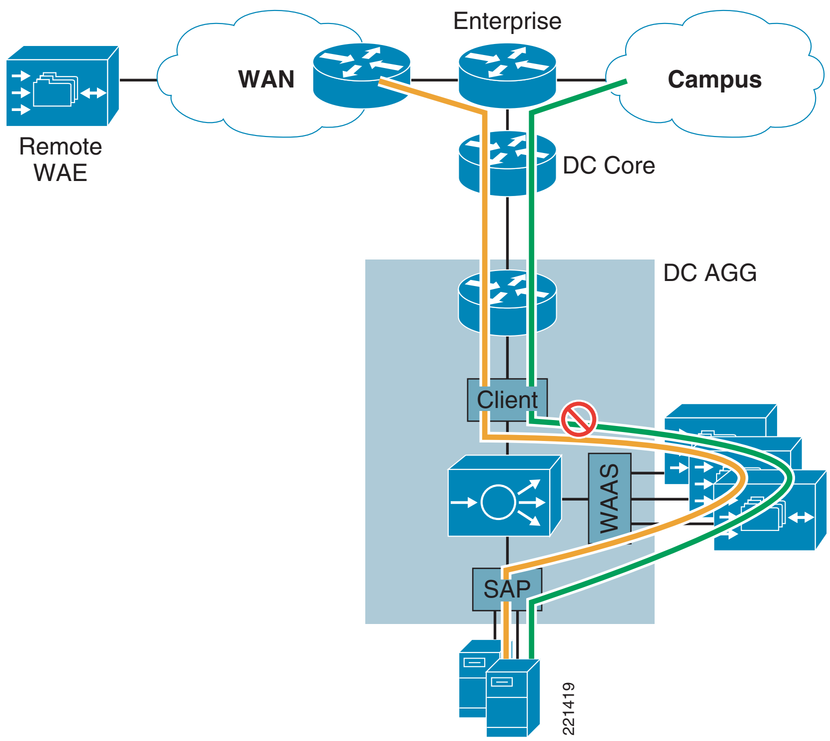
```
interface vlan 962
  description client VLAN
  ip address 12.20.62.2 255.255.255.0
  alias 12.20.62.1 255.255.255.0
  peer ip address 12.20.62.3 255.255.255.0
  no normalization
  access-group input anyone
no service-policy input SLB-policies
  service-policy input waas-policy

  no shutdown
interface vlan 957
  description EP-server
  ip address 12.20.57.2 255.255.255.0
  alias 12.20.57.1 255.255.255.0
  peer ip address 12.20.57.3 255.255.255.0
  no normalization
  access-group input anyone
  service-policy input waas-policy
  no shutdown
interface vlan 29
  description WAAS
  no normalization
  mac-sticky enable
  no icmp-guard
  access-group input anyone
  service-policy input SLB-policies
```

Although this works, there are a number of limitations with this approach:

- The WAEs must be located on an interface of the SAP context itself. Designing a dedicated WAAS context in the server farm that is shared by all other contexts is problematic. That means the most leverage you can get out of this WAE farm is the traffic destined to the SAP servers. This severely limits the utility of the WAE farm because it is unable to be used by servers on other virtual contexts. Similarly, it also reduces the likelihood that there will be a need to scale past the 32-device constraint of WCCP.
- The introduction of the WAE interface on the SAP context requires a "routed mode" deployment. This precludes the use of bridged mode in environments where this is desirable, such as the "VRF sandwich" approach, where routing updates need to be forwarded across a transparent ACE context.
- Campus and branch flows need to be separated so that campus traffic, which is not optimized, stays off the WAE devices. However, there is not currently a good way to separate these traffic flows when the core WAEs are in the server farm. ACE can differentiate based on source address using "http loadbalance" policy maps. However, when these maps are used, the WAAS TCP options are lost, preventing optimization. Another approach might be to apply different WAN and campus policies to different VIPs, but this is not practical. It is unlikely the administrator would want to maintain different VIPs based on client location in the network. Consequently, there is an unnecessary load on the central WAE farm, and it has less capacity to service the branch sessions. Figure 26 shows the traffic flow.

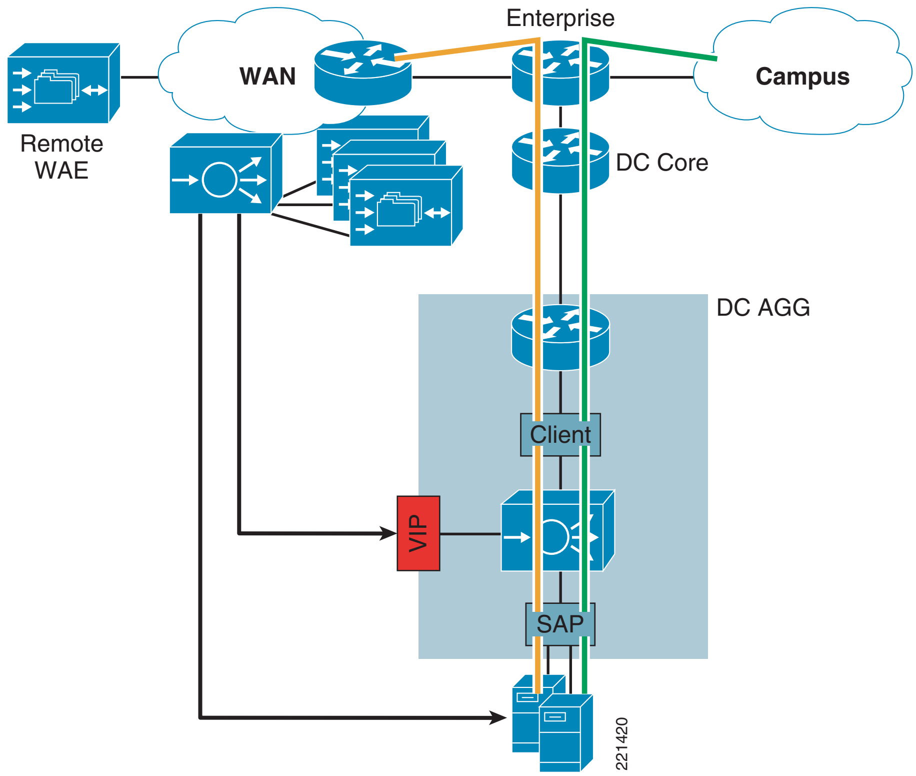
Figure 26 Campus and Branch Traffic Flows with WAAS in the Server Farm



WAAS and ACE at the WAN Edge

Considering the limitations outlined in the previous section, a better solution is to locate the core WAEs at the WAN edge, as shown in Figure 27.

Figure 27 Locating Core WAAS and ACE at the WAN Edge



The ACE is positioned in-line between the WAN edge router and the core. This way, only WAN traffic traverses the ACE and campus traffic flows directly to the data center servers. The configuration is basically the same, except that the rservers reference downstream VIPs rather than physical servers.

Positioning an in-line ACE at the WAN edge provides a solution for large-scale WAAS deployments. However, it is also potentially problematic, because it breaks the continuity of routing between the core and the WAN edge. It cannot be deployed in bridged mode, and there is no routing protocol support. Depending on the customer network requirements, the best solution in many cases will continue to be WCCP.

Conclusions

The design for the Cisco data center solutions for SAP includes Cisco ACE, FWSM, and WAAS. Among these, ACE provides a central role in managing the server farm by providing scalability, server offload, high availability, integration of WAN acceleration components, and security.

This guide focuses on a scalable design approach for SAP using ACE, AAS, and WAAS. With the virtualization capabilities in ACE, SAP can be deployed into an existing server farm with minimal impact, simply by creating a new context on an existing ACE and assigning VLANs to it. Role-based access control extends virtualization within a context to administrators with different responsibilities. To scale the SAP application, this guide shows how to measure the health of individual application servers as well as the entire system by monitoring external variables such as database health. When a host-level probe fails, that server is taken out of rotation. When a system-level failure occurs, all connections are routed to a backup server farm. This guide also shows what users might experience when persistence is not properly established and how to configure source IP and cookie-based persistence with SAP using the saplb_ cookie. Testing showed that normalization must currently be disabled for cookie-based persistence. In addition, the default cookie timer value of 10 minutes should probably be adjusted to several hours to avoid unnecessary calls to the help desk.

Persistence is also accounted for during failover events. The cookie database is maintained statefully across modules, so that the persistence of existing sessions is maintained if the primary ACE module fails.

The performance of SAP was also analyzed over a simulated WAN using T1 bandwidth and 52 ms one-way delay. The Cisco Application Analysis Solution breaks down the performance of a sample SAP transaction showing a comprehensive insight into the components of delay. WAAS, which is frequently used to reduce transaction delay and overall usage across the WAN, is then considered in terms of head-end deployment options, at the data center and the WAN-edge. Although it is possible to deploy WAAS in the data center with ACE, the best results are achieved when core WAEs are deployed at the WAN edge.