

External Load Balancing of WAAE Application Servers

Since AutoSys 11.0, CA has been using a 3 tier architecture, primarily to eliminate the need for distributed database clients on agent and client hosts. Now, all client calls are issued against an application server which handles security and database transactions on behalf of the client. Some examples of clients would be:

- Workload Control Center
- Commands such as jil, sendevent, autorep, etc
- SDK Applications
- Other application servers (in cross instance configurations)

There are many benefits to the 3 tier architecture, but one of the biggest benefits is also one of its drawbacks – consolidation. This means that all distributed load which may have been occurring on hundreds of client machines, is now occurring on one application server – that can be a “stressful” situation and usually shows its limits in environments that are very large, or where client commands are heavily used as part of the job/workload.

“Out of the box”, CA offers some limited failover/HA functionality, in that you can specify a comma separated list of application servers. This comma separated list is a sequential list of servers which the clients will use – a failover order, if you will. If the first application server is not available to the client, it will time out, and then the next in the sequence is tried, followed by the third, etc... Distributing load is limited to staggering this sequential list, or using exclusive lists for certain high-load clients. Also, there’s a bit of delay assumed by this repeated failover, at least until the application server is restored.

Such problems are textbook reasons for load balancing, and not entirely unique to WAAE. In fact, with load balancing so prevalent in the enterprise today, its likely there has been a solution already implemented and available for your use today.

Over the past few years I’ve performed and assisted in the successful implementation of ½ dozen or so load balanced configurations. I’ve tested and demonstrated the process and functionality, to both CA and CA customers alike. At the time of my writing this document, CA has acknowledged the benefits of external load balancing and has agreed that it is a viable configuration option – much like vmware, clustering, and LDAP, in that load balancing, when done correctly, is completely transparent to the application, and thus should not require any special “support”. In every case, the load balancer was already present in the environment, and utilization came at no additional cost beyond setup time. Your mileage may vary, but is possible that the solution of external load balancing is within your arm’s reach, and at no extra cost/chargeback to your organization.

There are several vendors and types of load balancers to choose from, but in principal, they all work the same. Whether you have access to F5 LTM/GTM, or perhaps have a popular linux distribution that includes HAPROXY, most any load balancer will fit the bill. I personally have extensive knowledge with configuring HAPROXY, and F5 load balancers – both with fantastic results.

Why and why not to:

When might you not wish to externally load balance application servers?

- If you do not stand to benefit from such a configuration. Just because you are able to do something, doesn't always mean that you should.
- If your load balancer is already saturated, and does not have the bandwidth to accommodate speedy traffic distribution.
- The load balancer is located somewhere over the WAN, and latency is introduced delayed response time. Just like the bandwidth point, a slow load balancer is going to introduce a delay into every single client call. You'll want to make sure this delay is minimal, and should not exceed what would be considered a reasonable ping delay.
- Certain network configurations that rely heavily on specialized routing or firewall rules may be prohibitive. There are also "one armed" and "two armed" load balancers, which may have an impact on the ability to properly route traffic.
- Knowledge deficiencies – typically, there are 2 parties (sometimes 3) involved in the end-to-end configuration. The application party (AutoSys), and the Load balancer or Network party. Usually, neither party knows what the other is talking about, and a successful configuration takes some additional effort by one of the parties to bridge the knowledge gap.

This document is intended to help facilitate the knowledge transfer between all parties, and since you are reading it, you might be implying your willingness to bridge the application/network gap. Note that while CA has officially stated that load balancers will work with its WAAE Application servers, they do not provide any special support or functionality to employ external load balancing. So you cannot expect CA Technical support to bridge these knowledge gaps for you – load balancing is a transparent technology, just like vmware, clustering or LDAP. Regardless, if you are not willing to understand the concepts within this document, and are therefore not willing to accept the responsibility of properly communicating or testing the configuration, an externally load balanced architecture might create more problems for you than it solves.

With the “why not to” out of the way, why might you want or need to externally load balance your application servers?

- Precise control over app server load using a variety of load balancing methods. Using algorithms such as least connection, predictive, or custom rule based load balancing can ensure that you are getting optimal utilization, and the best overall performance possible out of your application servers.
- Traffic reporting can help provide decision support in capacity planning and traffic shaping.
- Traffic shaping – the ability to align the right resources with the right demands. In other words, segregating WCC, generic client, and batch client activity. There are various methods of doing so, but you can avoid client contention by using a load balancer.
- Solving bandwidth limitations – Application servers have a concurrent thread limit of up to 128 transactions (32 by default). A client command could be comprised of many transactions, with each transaction consuming a database connection. In very large environments, a load balancer that permits concurrent connection limits, can help protect against overwhelming an individual application server, resulting in the “62 second timeout” and lost transactions.
- Alerting – the availability of an application server is not always known. From time to time an application server can go down without any notice. A load balancer, when properly configured, is going to be aware of an application server’s absence, and can therefore trigger an alert bringing attention to a crash that might otherwise go unnoticed.
- Failover errors like “CAUAJM_E_10029”. The native, sequential method of failover will error on app server attempts that fail, despite the ultimate success of the command. So even though a command may be successfully serviced by a secondary or tertiary app server, the exit still results in an error. If you have automated processes that run various client commands, you might receive an error when you should have received a warning. The load balancer however, solves this problem by not sending client traffic to app servers that are down – so clients never have to commit time to the initial failure, and the “CAUAJM_E_10029” never occurs unless the command actually fails.
- Rolling application server reboots are “invisible” to clients, requiring zero down time.

Service ports

A service port is the tcp port that a load balancer will be distributing traffic over.

In the current WAAE architecture, client/server communication is facilitated by CSAM (aka CA Secure Socket Adapter). This component uses a physical TCP Port (7163) to communicate bidirectionally between clients and the app servers. The client initiates communication via a listener port (port 9000), at which point an ephemeral port will be chosen by the application server (49152-50176). It's important to note that CSAM is only utilizing 1 physical port during these client transactions, and that port is TCP 7163. All other ports (9000, 49152-50176) are virtual ports, which are "multiplexed" or "tunneled" through port 7163. So in the context of external load balancing, port 7163 is the service port we need to be concerned about.

In addition to CSAM, some may choose to use ECLI functionality within WCC. The ECLI functionality is provided by another component called the iGateway. This component utilizes TCP port 5250, bidirectionally. So if an externally load balanced application server configuration is to be used, and ECLI functionality is desired, a "wildcard" port must be used for the service port. This port is typically TCP port "0", and is used in place of both 7163 and port 5250 in the load balancer config.

Health Monitoring Port

As the name implies, this port is what the load balancer will use to monitor the application status. In other words, to verify that the application server is alive. This will help the load balancer determine the state of the application, and whether it should keep the server in the pool of available application servers.

The perfect port for health monitoring is the Application Server Auxiliary port. This port is defined in the \$AUTOUSER/config.\$AUTOSERV file as the "AppSrvAuxiliaryListeningPort". The default port is TCP 7500. This unidirectional port is intended to serve as a listener port for WAAE System Agents, and is often used during message queue management and actions such as WCC validation. Since CSAM is a service that runs independent of the Application server, port 7163 cannot be relied upon to determine the health of the application server. So TCP port 7500 is best suited for this purpose, as it is a port directly associated with the application server.

Configuration

As mentioned earlier in this document, most all load balancers operate on the same principles. Really, the most significant difference is usually the terminology used to describe the load balancer configuration.

Some synonymous terms and their meanings:

Back end, application server list, virtual server pool: logical grouping of application servers/nodes to receive traffic. In the context of this document, a per WAAE instance list of hosts having as_server running on them.

Client, source address: clients such as jil, sendevent, autorep, autostatus, WCC and their respective addresses.

Session stickiness, affinity, source address/ip persistence: the managed relationship between clients and application servers. Once this relationship is established, the load balancer will continue to direct traffic from a particular client to the same application server. That is, until a rule, session timeout, expiration, or application availability dictates otherwise.

Health Monitor, check port, health check: A method and port used by the load balancer to verify the application state

Virtual Server, frontend, load balanced name: the ip and logical hostname that represents the load balanced application. Traffic that goes to this virtual name will be directed towards a pool of physical servers – it's the core of the load balancer configuration

Node, backend server, application server: an application server which has an as_server/application server installed on it, and may receive client traffic from the load balancer

Service port, forwarded port, balanced port, application port: The active port which client traffic will be distributed on. Unless prohibited, tcp port "0" should be used as it includes all ports. Otherwise, tcp 7163 is the WAAE service port, and 5250 is the igateway port.

Mode, virtual server type: the type of load balancing used, i.e. "tcp", "layer 4", or "performance layer 4". In the context of this document, you should never have HTTP or application layer load balancing.

To setup external load balancing for WAAE application servers, there are multiple areas of configuration. In an effort to explain this in the most logical/portable way, I've split the various areas up based on component:

Load balancer config

This configuration assumes you are using an F5 load balancer. I've performed extensive testing with HAProxy on BSD as well, and I find the results very impressive. So the decision to provide an F5 example is not necessarily an endorsement, its just that F5 is probably the most common load balancer in use at the time of my writing this document.

1. Obtain or define the virtual ip and hostname to be used for the virtual server. This might involve a request/change with your DNS/Networking team, if your F5 team is not a "one stop shop" for all things network.
2. Define the nodes for your WAAE instance, these are the application hosts which you will later add to the virtual server pool
3. Define a Virtual Server Pool which includes the application server nodes from step 2
4. Define the Virtual Server using:
 - a. "Performance Layer 4" type
 - b. "0" for the tcp service port
 - c. Provide the virtual server name you obtained in step 1
 - d. "source_addr" persistence
 - e. "Automap SNAT" for the source address translation (F5 calls it secure address translation)
 - f. "Predictive" for the load balancing method – you may use "least connection" as a substitute
 - g. Set the persistence timeout – 30 seconds or lower is best.
5. Define a Health Monitor
 - a. Use the "TCP-Half Open" method
 - b. "7500" is the monitor port (note 7500 is the default, yours might differ)

WAAE Client config

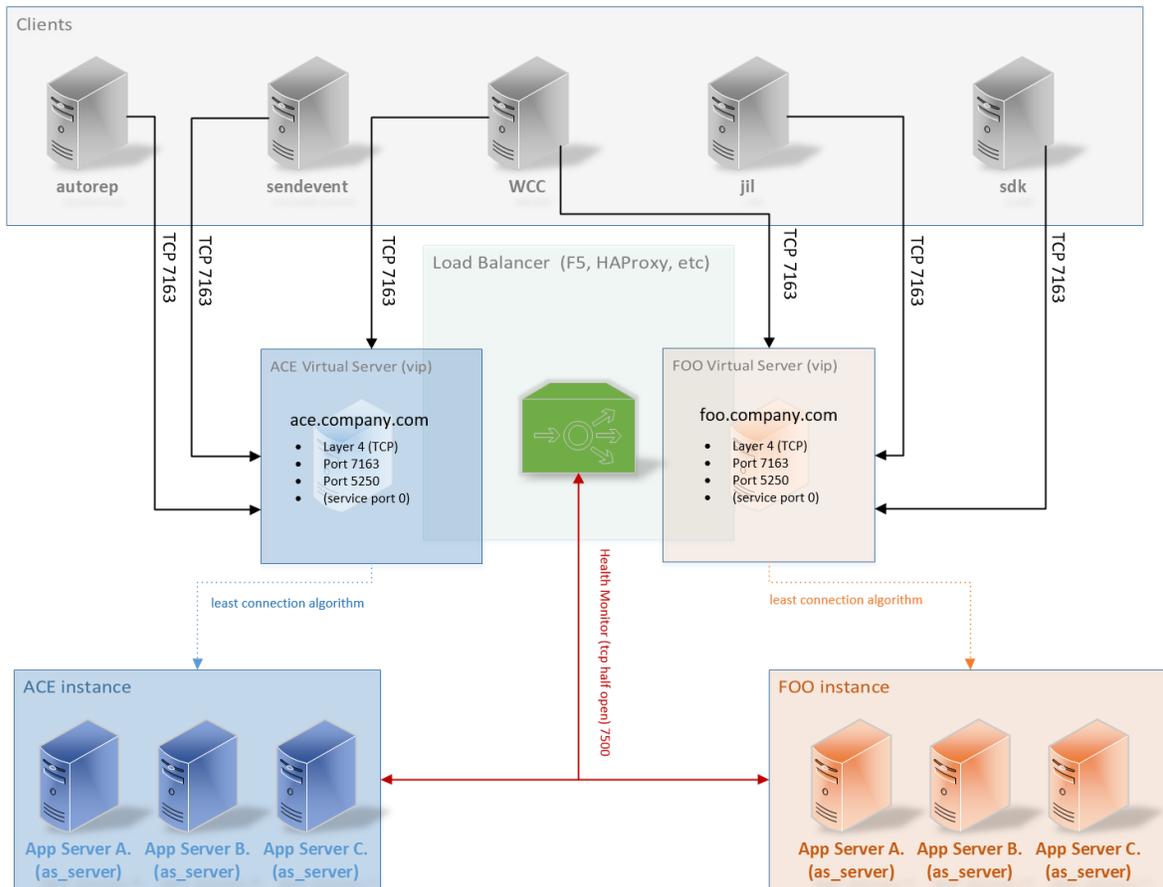
1. Edit the client configuration file \$AUTOUSER/config.\$AUTOSERV to contain the following:
AutoServer=<virtual_server_name>,server_a,server_b,server_c

Note on windows the AutoSys Administrator utility can be used – version 11.3.6 and up now support a configuration file.

WCC config

1. On the config tab, click on the "server name" which you wish to modify
2. Locate the "Application Server Host" field, and prepend the virtual server name that represents the instance, as configured in step 4 of the "load balancer config" steps above.

The diagram below shows the distribution of client connections, the associated ports. The virtual server name “ace.company.com” is representative of the “ACE” WAAE instance, and all associated application servers. Also included is another instance called “FOO”, demonstrating the need for a unique virtual server configuration for each instance you wish to load balance.



The client config file “\$AUTOUSER/config.\$AUTOSERV” would contain the following, in order to support the configuration in the diagram:

```
AutoServer=ace.company.com,server_a.company.com,server_b.company.com,server_c.company.com
```

For the next instance, FOO, the config would look like so:

```
AutoServer=foo.company.com,server_a.company.com,server_b.company.com,server_c.company.com
```

The configuration would require a separate virtual server for each and every autosys instance.

Initial testing of your configuration

It goes without saying, but testing should be performed before deploying to the enterprise. A good test would obtain clients from several subnets, while obtaining a reasonable representation of the platforms used in the environment.

An initial test would consist of the following:

1. Modify the \$AUTOUSER/config.\$AUTOSERV on the test client so that the virtual server name is the only value for the parameter "AutoServer":
AutoServer=ace.company.com
2. With the modified config file, go ahead and run various commands such as autorep, sendevent, jil, etc.
3. Assuming the service levels permit within the environment you are testing in, go ahead and kill - 9 one of the application servers and run step 2 again. Perform this action successively until all application servers are down and the following error is received:
CAUAJM_E_10029 Communication attempt with the CA WAAE Application Server has failed! [ace.company.com:9000]
4. At this point start up one of the application servers and perform step 2 again. If you choose to, you can run a while true loop, with the desired command in step 2, and watch the behavior as you kill and start application servers.
5. As a substitution or additional series of tests, you can have the load balancer administrator progressively bring down each node in the virtual server pool. This test may be preferred in situations where a pure test environment is not available due to service level requirements, since only those clients which are configured to use the load balancer are affected.
6. Once testing is satisfied, and various subnets are all verified, you can move to updating the WCC configuration. On the config tab prepend the load balancer virtual server name to the list of application servers listed in the "Application Server Host Address" – NOTE: Unlike the client, WCC must have a comma separated list of ALL application servers, in addition to the virtual server, for it to function properly. This is because WCC needs to validate the actual application server name. The load balancer virtual server name will allow communication to initiate, but WCC will become aware of the actual application server it is working with, and therefore must have that server in the comma separated list of hosts. Simply putting the virtual server name in the configuration, with no other entries, will break WCC. Also, if using WCC HA in 11.4, the configuration will be shared with all other member WCC servers, because WCC now stores its configuration in the database:

CA Workload Automation My Profile | EEM | Help

Configuration

• = Required [Servers](#) | [Preferences](#)

[Expand All](#) | [Collapse All](#)

Property Name	Property Value
CA Workload Automation AE	
General	
Server Name	ace_instance
Product Version	11.3.6.4 ▼
Instance Name	ACE
Cluster Name	Default
Application Server	
• Application Server Host Address	ace.company.com,server_a.company.com,server_b.company.com,server_c.company.com
• Application Server Host Port	9000
Compatibility	
• Legacy Compatibility Port	4444
Monitor User	

After initial testing is completed, you can modify the client configurations so that the virtual server (load balanced name) is the first in the AutoServer order, followed by each application server in the instance. This gives extra protection in the event that the load balancer is not available, the normal failover sequence will resume. I would recommend performing a “burn in” through the various environments, to be sure that the solution is thoroughly tested before rolling out into production.

Troubleshooting tips

Troubleshooting a load balancer configuration can be frustrating to the uninitiated, which is why I offer some of these tips.

1. The application is actually host aware. The CSAM component sends packets which contain the host of origin, which means that although a load balancer is used, a server will likely respond directly back to the client instead of through the load balancer on subsequent replies. This becomes significant in certain environments where there routing may prohibit or create problems with asynchronous communication. I've only seen this one time and testing exposed the issue almost immediately.
2. Have tcpdump installed, or wireshark. These are extremely helpful utilities and any load balancer admin will be well versed in their use. When looking at these utilities, use the service port 7163, and check all source/destination traffic. The same reports can be provided from the load balancer so you can see what is happening, end to end.
3. Reduce the complexity to make troubleshooting easier. In your client configuration, for example, you can specify only the load balancer server, or possibly one of the application server nodes. If the problem occurs without the load balancer, it will most certainly occur when the load balancer is added to the configuration. So rule out the load balancer.
4. On one occasion I had seen where a customer had opted to use a DNS load balancer instead of their F5 load balancer. The difference being that the DNS would resolve the virtual name to a different ip, each time a lookup was done. For this to work, you would need to be mindful of nscd, or any other type of name service caching on the client.
5. Use CSAM logging – it does not seem to be as helpful as it used to be, but it can sometimes help. The CSAM log resides under `$CSAM_SOCKETADAPTER/log`.
6. Enable COMM trace on the application server via the config file:
 - a. `ISDBGACTIVE=COMM`
 - b. You're looking for errors or abnormalities. I recommend starting with a properly functioning client/application server config, and observing the normal behavior.
7. Enable COMM trace with the client
 - a. `ISDBGACTIVE=COMM; export ISDBGACTIV`
 - b. As with 6b. you're looking for abnormalities which will be seen in standard out.
8. Simple timing can be an indicator. If it takes 20 seconds to timeout, it's likely that the client communication is going out, but communication from the application server is not making it back.
9. Ping has limited use with the load balancer, and will verify something is listening, but that's about it. Regardless, it can be a good first test.
10. Traceroute on a virtual server name can result in looping indefinitely, or until the max hops are reached.
11. CA Technologies Technical support will not be of much help in troubleshooting, unless the issue can be reproduced in a non-load balanced environment. As mentioned earlier, there are no special features or design elements in place to exploit or permit an externally load balanced configuration. That being said, your load balancer vendor support might be able to assist with further troubleshooting.
12. Other sources for load balancing documentation and support are:
 - a. <https://devcentral.f5.com>
 - b. www.haproxy.org