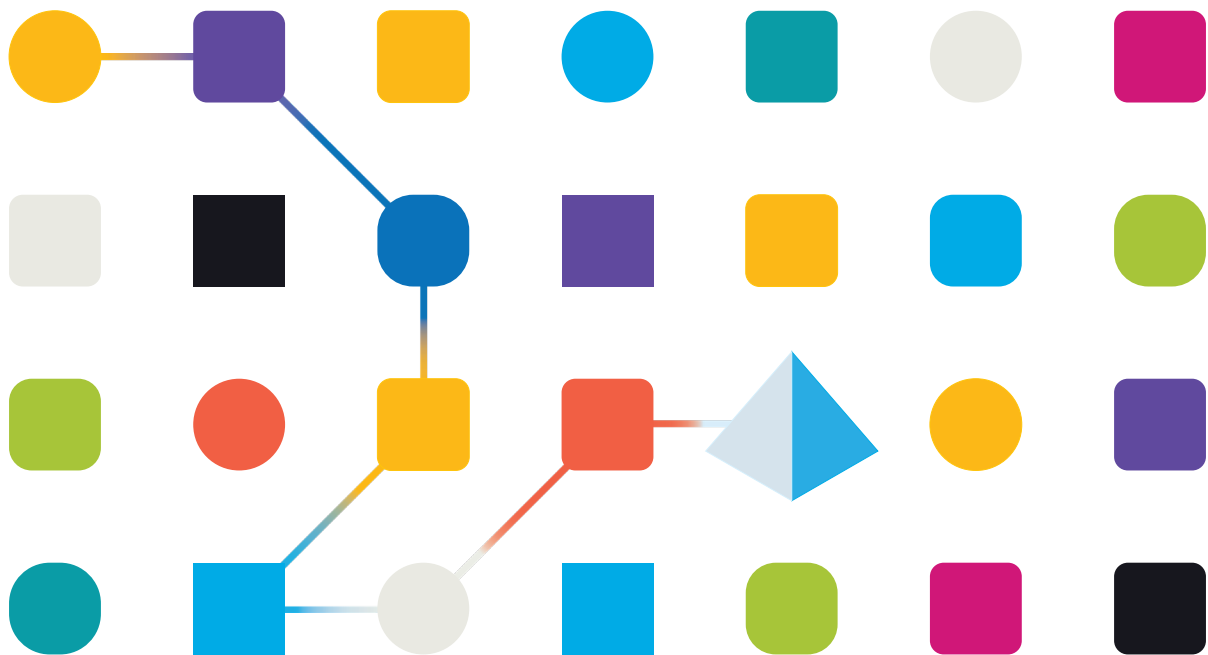


Blue Prism 6 Load Balancing Reference Guide

Document Revision 1.0



Trademarks and copyright

The information contained in this document is the proprietary and confidential information of Blue Prism Limited and should not be disclosed to a third party without the written consent of an authorised Blue Prism representative. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying without the written permission of Blue Prism Limited.

© **Blue Prism Limited, 2001 – 2021**

®Blue Prism is a registered trademark of Blue Prism Limited

All trademarks are hereby acknowledged and are used to the benefit of their respective owners.
Blue Prism is not responsible for the content of external websites referenced by this document.

Blue Prism Limited, 2 Cinnamon Park, Crab Lane, Warrington, WA2 0XP, United Kingdom
Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: www.blueprism.com

Contents

- Introduction 1
 - Audience 1
 - About this document 1
- Terminology and Key Concepts..... 2
 - Terminology 2
 - Key Concepts 6
 - Other Configuration Considerations..... 7
- Design and Configuration Scenarios..... 11
 - System Design for High Availability and Redundancy 11
 - Configuring Blue Prism for Transport Encryption 14
- Summary of Blue Prism Recommendations..... 17
 - Blue Prism Recommendations 17
 - Troubleshooting an Issue in a Secure Deployment 17

Introduction

Audience

This reference guide is intended for use by system architects or designers who are seeking to gain an understanding of the options for performing load balancing within a Blue Prism environment. This guide is supplementary to the High Availability topics in the Infrastructure Reference Guide. The Infrastructure Reference Guide is subject to a non-disclosure agreement (NDA) and is available on request from Blue Prism Customer Support.

About this document

The document introduces the key concepts involved in load balancing, the deployment options, and the implications for the Blue Prism application operation. Some understanding of the basic concepts of Blue Prism and architecture is expected of someone designing a solution that involves a load balancing solution with version 6. There is also an assumed understanding of basic networking concepts.

Unless otherwise specified, this document refers exclusively to WCF-based connection modes.

If using .NET remoting connections in version 6 (which is supported for backwards compatibility only) please refer to the version 5 Blue Prism Reference Guide - Load Balancing, available from the Blue Prism portal, as there are differences in the required configuration of a load balancing solution.

Terminology and Key Concepts

Load balancing is a broad technical concept with a wide range of technical solutions – often with differing capabilities and terminology. This section explains the key concepts of load balancing within the context of how they can be used in a Blue Prism environment. Some of the terminology is not industry standard and may vary between vendors but should be applicable in most scenarios.

The only documented and supported scenario for load balancing is between Runtime Resources or Interactive Clients and Application Servers.

Terminology

This section outlines and explains some of the key terms often used in load balancing documentation - these may differ between vendors but will remain generally the same.

Host

Sometimes also known as the Server or Node and generally refers to the server that will receive traffic from the load balancer. This is synonymous with the IP address of the physical server and, in the absence of a load balancer, would be the IP address that the server name would resolve to. In the context of the Blue Prism solution, this would relate to an Application Server host.

Service

Also known as the Member or Node, a Service is usually a little more defined than a Host in that it includes the TCP port of the application that will be receiving traffic. For instance, a server named bpappserver001 may resolve to an address of 172.16.1.10. With a Blue Prism Application Server Service running on port 8199 (the Blue Prism default), this results in a Service address of 172.16.1.10:8199.

Simply put, the service includes the definition of the application port as well as the IP address of the physical server. Note that a Host may be running multiple instances of an application server service. This is important when determining the appropriate definition of Pools (see below).

This guide uses the term *Application Server Service* for Services.

Pool

Load balancing allows organizations to distribute inbound traffic across multiple back-end destinations (in our case – Application Servers). It is therefore necessary to have the concept of a collection of back-end destinations. Pools, also known as clusters or farms, are collections of similar services available on any number of hosts.

The key concept here is that all systems have a collective object that refers to all similar services and makes it easier to work with them as a single unit. This collective object, a pool, is almost always made up of services, rather than hosts, though it can be either. From a Blue Prism perspective, Application Server Services should exist in pools, so they are logically grouped to represent the desired distribution of connections.

For example, it would be logical to separate all Development (DEV) application server Services from User Acceptance Testing (UAT) services within a pool, even though it is feasible that these Services may be collocated on the same Host.

Pools may also be used to allow a greater degree of control over availability and maintenance scheduling. This is explained further in [Design and Configuration Scenarios](#).

Health Monitoring

One of the advantages of a dedicated load balancing solution is that the load balancer can be configured to monitor the health of its managed hosts or services. Once again, there is variation between solutions, but in general the options fall into the following categories:

- **Simple Monitoring** - Determines whether a Host is available by pinging it using ICMP or TCP_ECHO. This form of monitoring is not capable of determining the availability of the Blue Prism application, only the availability of the server itself. Therefore we would recommend the use of Active or Passive monitoring instead.
- **Active Monitoring** - Checks the status of a pool member or service on an ongoing basis using one of the various, generally available methods. In version 6 of Blue Prism the Application Server Service exposes an HTTP/S (depending on configuration) endpoint, the availability of which can be used to indicate health of the Service.

An endpoint for bpappserver001 on port 8199 using WCF: Transport Encryption, with (or without) Windows Authentication would be configured as below:

```
https://bpappserver001:8199/bpserver
```

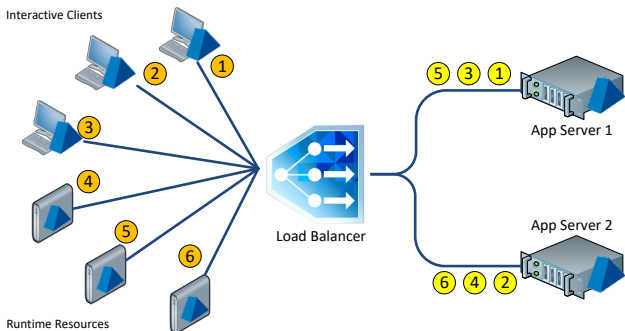
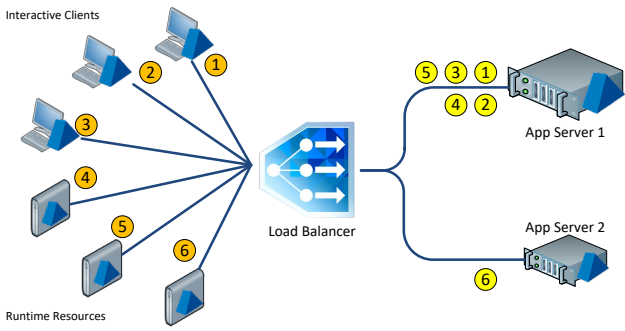
Conversely an endpoint for bpappserver001 on port 8199 using WCF: Message Encryption with Windows Authentication or WCF: Insecure would be configured as:

```
http://bpappserver001:8199/bpserver
```

- **Passive Monitoring** - Passive monitoring occurs as part of a client request. This type of monitoring checks the health of a pool member based on a specified number of connection attempts, or data request attempts, that occur within a specified time period. If, after the specified number of attempts within the defined interval, the system cannot either connect to the server or receive a response, or if the system receives a bad response, the system marks the pool member as unavailable.

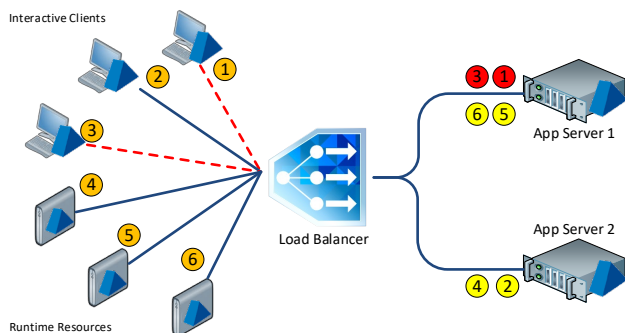
Load Balancing Algorithm

The method by which a connection is routed to an Application Server Service is determined by an algorithm. Blue Prism does not recommend one algorithm over another; algorithms have some variation between solutions and customers should use one that suits their solution. The most common algorithms are defined below:

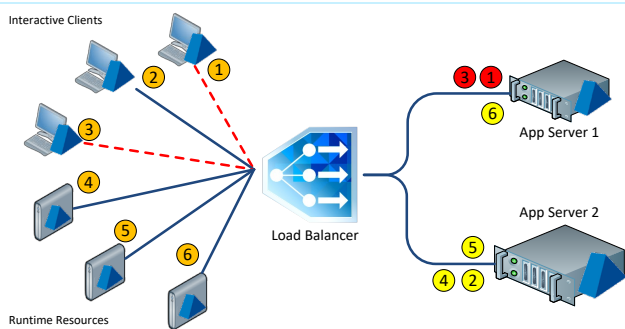
Algorithm	Explanation
 <p>Round Robin</p>	<p>The concept of Round Robin is identical to using DNS Round Robin, except that more intelligence can be applied, using health monitoring and load balancing Pools, to ensure that sessions are not allocated to an Application Server Service which is unavailable.</p>
 <p>Weighted Round Robin</p>	<p>The Weighted Round Robin is like the Round Robin in that requests assigned to the nodes is still cyclical, but a node with higher specification can be apportioned a greater number of requests.</p> <p>When the load balancer is configured, weights are assigned to each node. The node with the higher specification should of course be given the higher weight.</p> <p>Usually weights are specified in proportion to actual capacities. For example, if Server 1's capacity is five times more than Server 2's, it could be assigned it a weight of five and Server 2 a weight of one. It would generally be expected that Application servers in a pool would have identical specifications, therefore this scenario is not normally applicable.</p>

Algorithm

Explanation

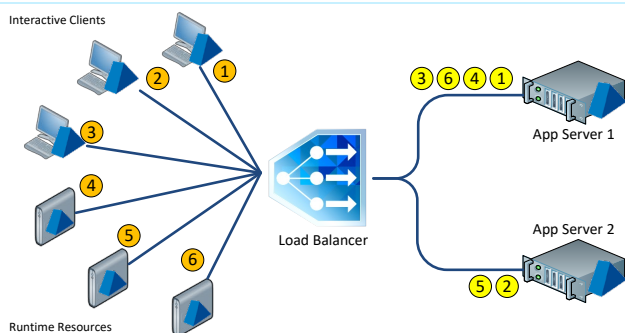
**Least Connection**

This algorithm takes into consideration the number of current connections each server has. When a client attempts to connect, the load balancer attempts to determine which server has the least number of connections and assigns the new connection to that server. In the example diagram, if clients 5 and 6 attempt to connect after 1 and 3 have been disconnected and 2 and 4 are still connected, the load balancer will assign both connections to Server 1. In a Round Robin scenario, the load balancer is unaware of disconnections and would have assigned one of the connections to each server, leaving an imbalance.

**Weighted Least Connection**

Weighted Least Connection builds on the concept of Least Connection by introducing a weight to the algorithm.

A load balancer that implements the Weighted Least Connection algorithm takes into consideration two things: the weights/capacities of each server and the current number of clients currently connected to each server. In the example, the higher Weight of Server 2 could cause connection 5 to be assigned to Server 2, despite it having more connections.

**Random**

This algorithm matches clients and servers randomly using an underlying random number generator. In cases where the load balancer receives many requests, a Random algorithm will be able to distribute the requests evenly to the nodes. Like Round Robin, the Random algorithm is enough for clusters consisting of nodes with similar configurations (CPU, RAM, etc.).

Key Concepts

This section outlines some of the key concepts associated with load balancing scenarios. Similar to the [terminology](#), there may be slight variations between vendors; however the concepts should remain mostly the same.

Using a Load Balancer/Application Delivery Controller

Load balancers distribute network or application traffic across several servers - they are used to increase capacity and reliability of applications. They improve the overall performance of applications by decreasing the burden on servers associated with managing and maintaining application and network sessions, as well as by performing application-specific tasks.

Both Software and Hardware load balancing solutions are available. It is not within the scope of this document to address the pros and cons of each. Examples of hardware load balancers include F5 Big-IP Local Traffic Manager (LTM), Cisco Load Balancer, Citrix NetScaler and Fortinet and examples of Software load balancers include HAProxy and NGINX.

Load balancing functionality can be grouped into two categories:

- **Layer 4** load balancers act upon data found in network and transport layer protocols (IP, TCP, FTP, UDP).
- **Layer 7** load balancers distribute requests based upon data found in application layer protocols such as HTTP. Layer 7 load balancing is imperative for the utilisation of the more advanced features of load balancing.

It is not unusual for a load balancer to be configurable to work as both a Layer 4 and Layer 7 load balancer.

Load balancers ensure reliability and availability by monitoring the health of applications, only sending requests to servers and applications that can respond in a timely manner. This provides advantages to their use when compared to a simple DNS round robin.

Requests are received by both types of load balancers and they are distributed to a particular server based on a configured algorithm. The common algorithms are explained in [Load Balancing Algorithm](#).

DNS Round Robin

A basic level of load balancing may be achieved using a DNS round robin approach, where connections are routed to an application server based on the DNS responses, according to a statistical model. This may be achieved using either a hardware or software load balancing technology.

DNS Round Robin is not recommended for use in version 6 with WCF connection modes because the way that messages are transmitted differs to .NET Remoting. If using .NET Remoting in Blue Prism version 6, please refer to the version 5 Blue Prism Reference Guide - Load Balancing.

Other Configuration Considerations

Whilst there are variations between vendors and solutions, the following considerations would generally apply in most load balancing solutions.

Session Persistence (Stickiness)

The persistence, or stickiness of a session refers to the concept of ensuring that a connection from a device is always served to the same server. When a client connects to a service, the load balancer remembers the last connection for a specified period. If that same client connects again within that period, it is sent to the same server it connected to previously — bypassing the load-balancing mechanisms. When a connection occurs outside the time window, it is handled according to the rules in place.

The connections that are established by Blue Prism over a WCF connection require session persistence to be set. As a guideline, session persistence should be configured for the duration of the longest running process plus an amount of headroom to allow for environmental or system variation.

If using .NET remoting, which is supported in version 6 only for backwards compatibility, please refer to the version 5 Blue Prism Reference Guide - Load Balancing.

Affinity Patterns

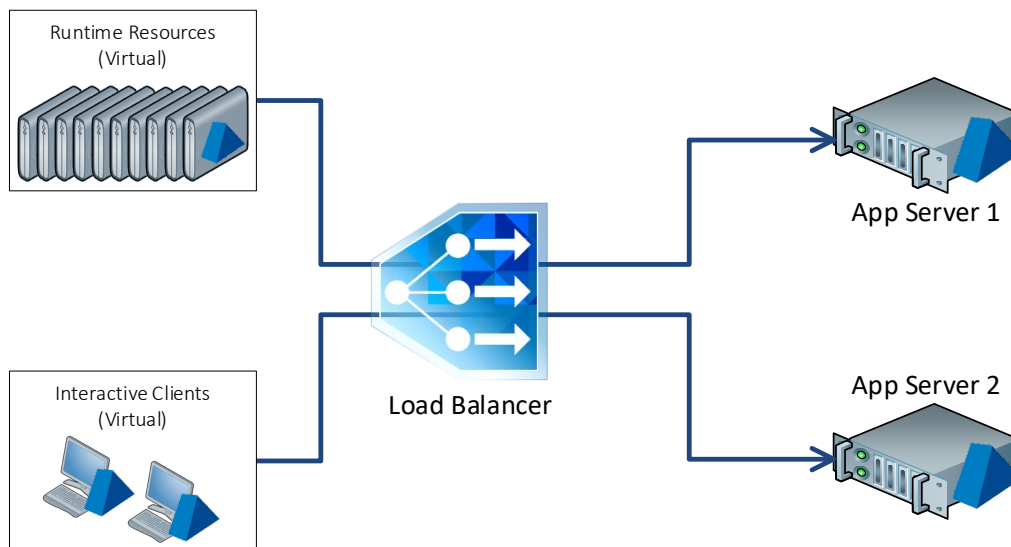
An Affinity Pattern is the mechanism by which session persistence is achieved. It defines which properties of the traffic requiring Session Persistence are used, to ensure it is routed to the correct server in the Pool. There are a variety of options available to define affinity behaviour. Some of the common patterns are:

- Source IP Address
- Destination IP Address
- SSL (usually based on properties of the SSL session)
- Custom Hash
- Cookie-Based

Cookie-based Affinity Patterns are not currently supported because Blue Prism Clients and Robots do not track and apply cookies returned in responses to subsequent requests in the same way that a web browser would.

Network Address Translation

The introduction of WCF in Blue Prism 6 removes the need to configure call-back ports as responses are made directly back to the port from which the request was sent. WCF messages are more like Internet traffic and therefore do not need a long running channel to be held open. This means that sessions are persisted on the Application Server rather than in transport. In addition to this difference, WCF response messages are capable of Network Address Translation meaning that there is no requirement for 'direct server return', as required with .NET Remoting connection modes.



Load Balancing HTTPS Connections

When configuring load balancing for connections that use the HTTPS protocol (WCF: Transport Encryption with Windows Authentication or WCF: Transport Encryption) there are several configuration options that are available depending on the scenario.

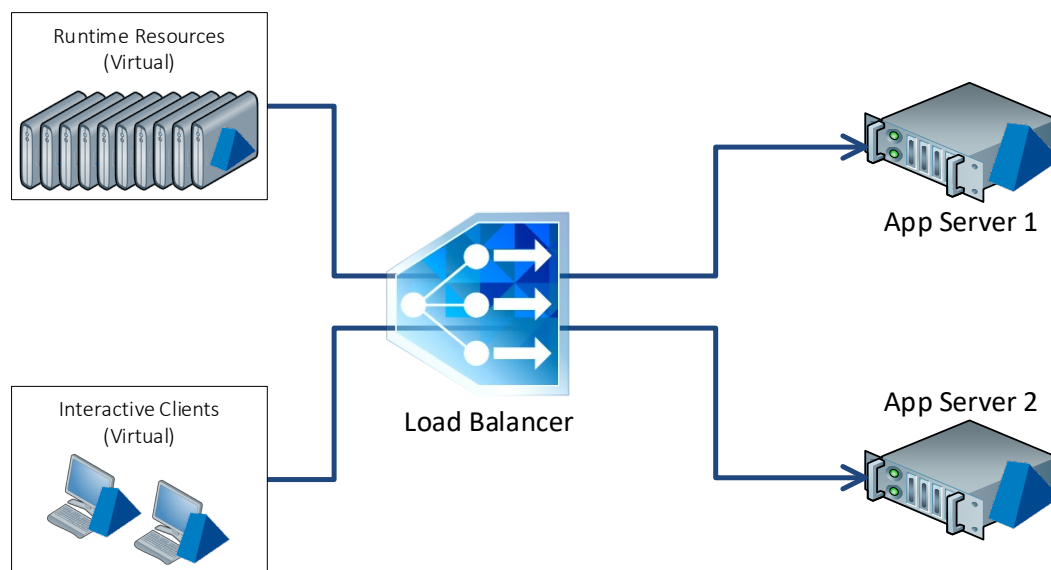
This section details the following HTTPS load balancing methodologies:

- SSL Passthrough – Supported.
- SSL Termination – Supported.
- SSL Offloading – Not supported. For further information, see [SSL Offloading](#).

The examples below assume that a suitable Affinity Pattern has been configured - see [Session Persistence \(Stickiness\)](#).

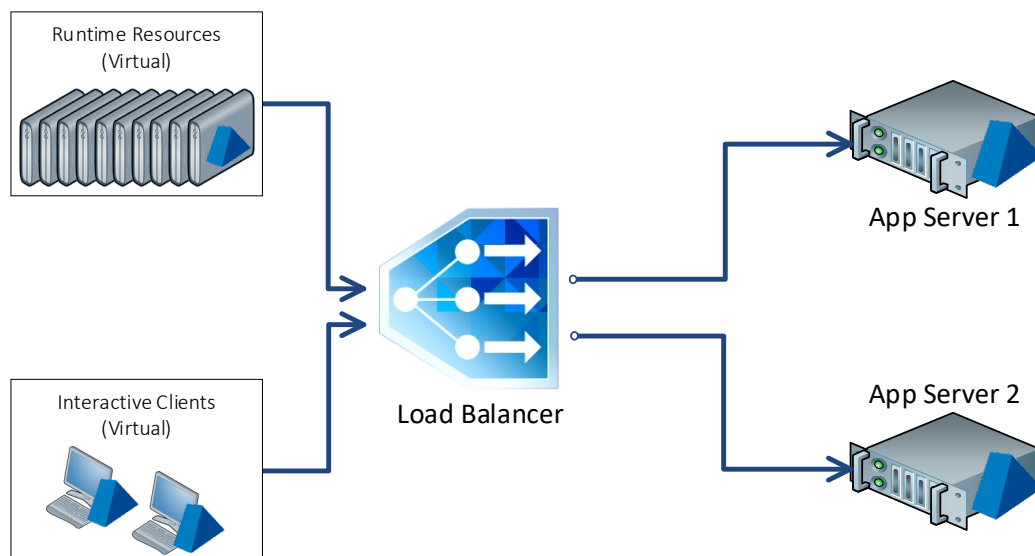
SSL Passthrough

Layer 4 load balancing is often referred to as SSL Passthrough and is the simplest method of load balancing HTTPS connections. It involves directing traffic to the Application Server Service using a single connection without decrypting any packets that are sent to the load balancer. The initial connection from the Runtime Resource or Interactive Client is carried all the way through to the Application Server, as shown in the diagram below.



SSL Termination

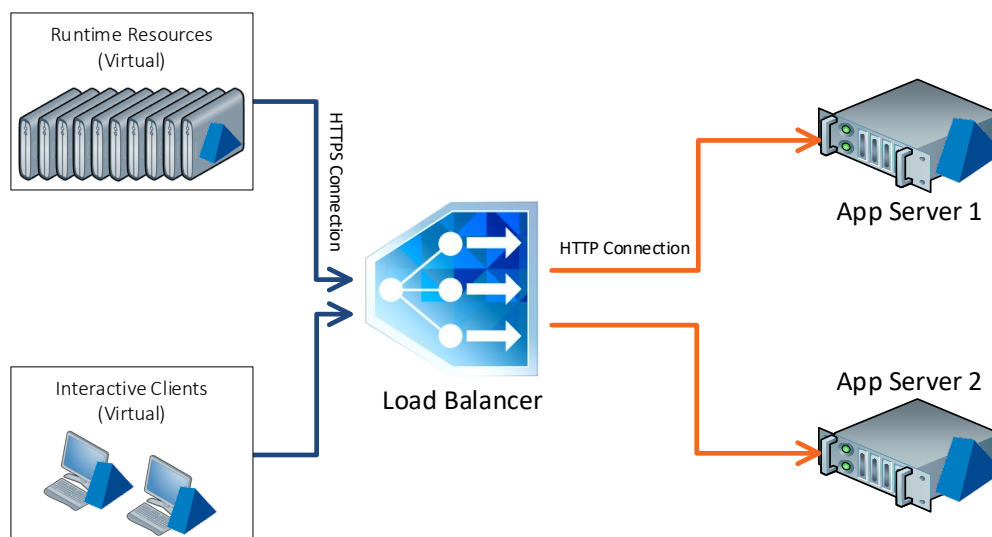
SSL Termination is like SSL Passthrough but differs in that a new connection is established between the load balancer and the Application Server. The first connection is made to the load balancer where the content is decrypted before being re-encrypted and sent to the Application Server using a new connection. When a response is received from the Application Server this is returned to the source using the original connection. This is shown below - note the separation of connections at the load balancer.



SSL Offloading

SSL Offloading is conceptually like SSL Termination, the only difference being that traffic to the Application Servers is not re-encrypted following the initial connection. This is common practice in web farms as communication inside the datacentre is considered secure. Once data has arrived it is unnecessary to re-secure any traffic, dramatically reducing response times.

SSL Offloading is not supported in Blue Prism and is included here for reference. This is because the WCF Reliable Session Initiation message sent from the Runtime Resource or Interactive Client negotiates a secure WCF Reliable Session over HTTPS - this message cannot be received via an HTTP connection.



Design and Configuration Scenarios

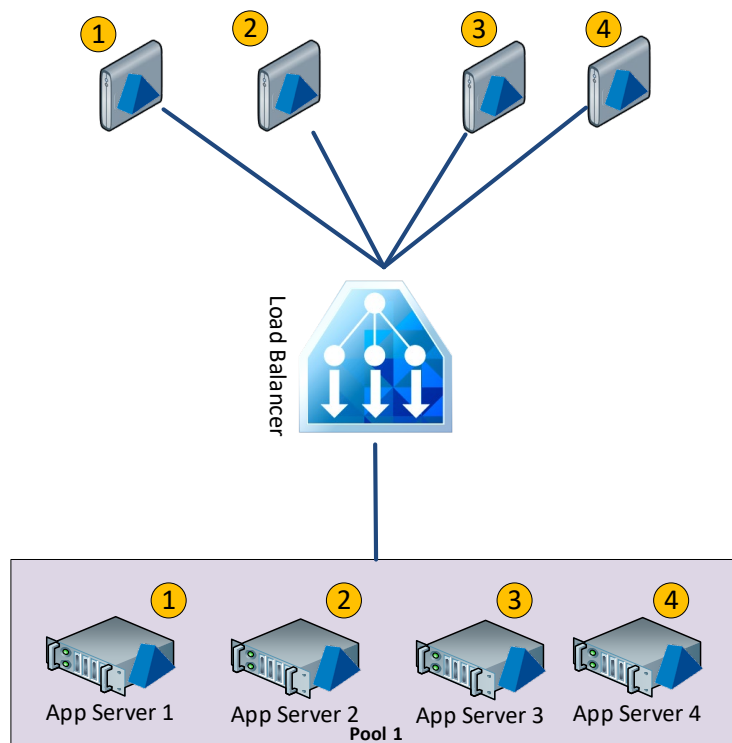
It is important to understand the requirements for maintenance and failover between components and develop a strategy that satisfies these. The following sections are intended to explain the key concepts and configuration items involved, including configuration of Blue Prism for WCF: Transport Encryption connection modes.

System Design for High Availability and Redundancy

Using a load balancing solution for the purposes of High Availability rather than application distribution requires an additional degree of care and attention that must, as a minimum, consider the following:

- Maintenance approach – How the Application Servers will be grouped and maintained, to ensure the availability of a device for the Runtime and Interactive Client connections.
- Infrastructure Availability vs Business Process Availability – Considering the availability of the Application servers will not, on its own, guarantee availability of the Business processes being managed by the Runtime Resources. This should also take into account the number and distribution of resources between Application servers and load balancer pools, against the available resources for a given Business Process.

In order to ensure the highest level of availability for Business Processing, it is recommended that a combination of Balanced Application Server Pools and resource grouping is used. The table below outlines some common and recommended scenarios, along with the implications for High Availability.



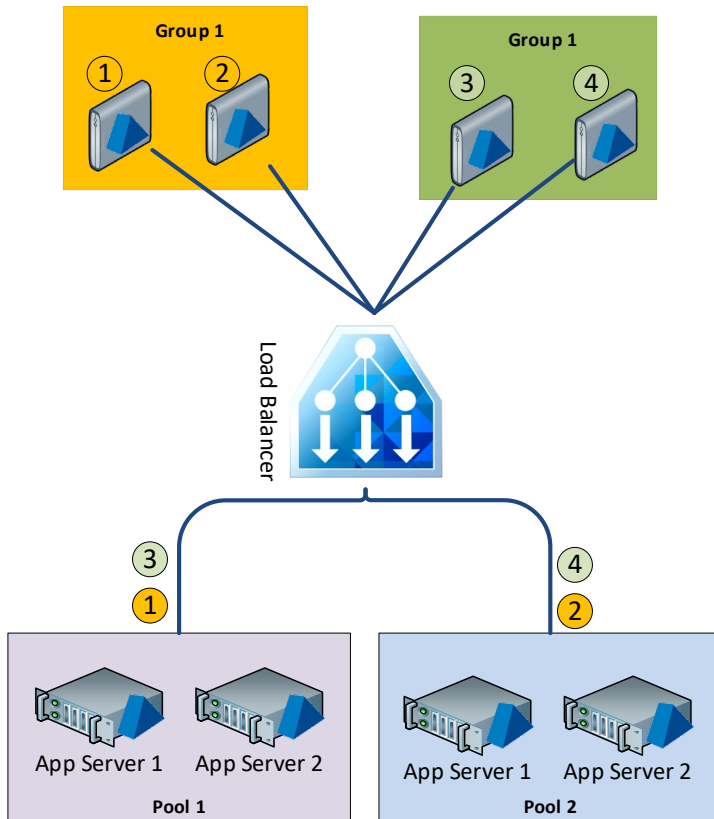
Single Pool

In a scenario with one single load balancing pool, it is difficult to ensure that the distribution of Runtime Resources is predictable. The diagram shows an even distribution, however it is just as likely that distribution will not be equitable, depending on the number of connections and the algorithm used.

If this deployment scenario is used, then care must be taken to ensure maintenance of the servers is carried out in a sequential manner, such that connections can be redistributed to an active server as efficiently as possible.

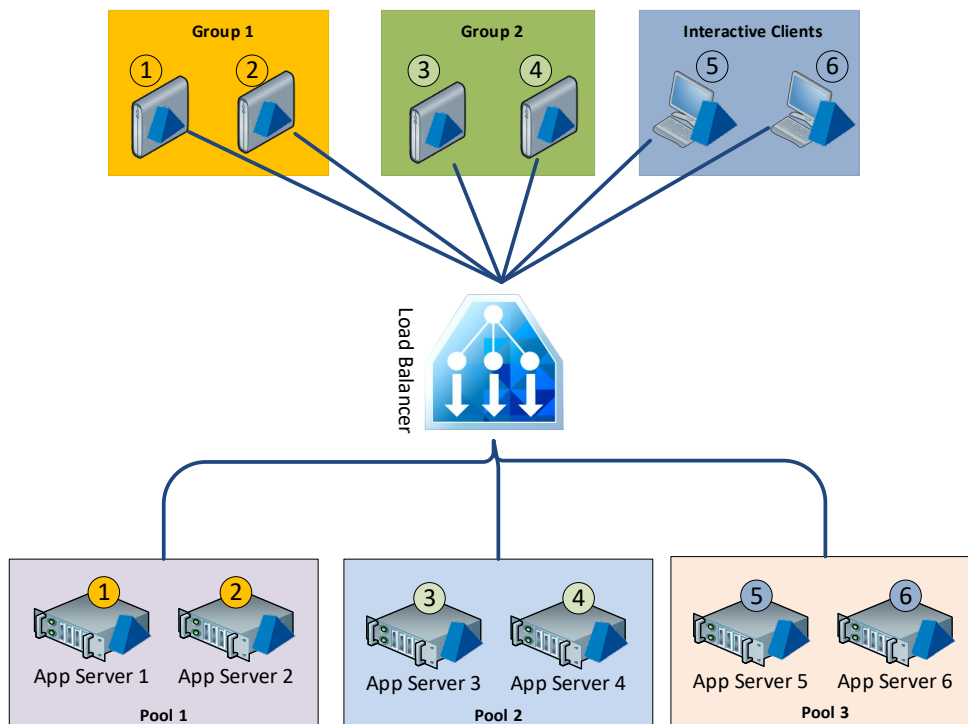
Work Queues should always be monitored during planned maintenance or after an outage, to manage any process exceptions.

Logical Load Balanced Pools



- Application Servers are grouped into logical load balanced pools. This may also be based on location, to account for DR scenarios.
- Runtime Resources are grouped, with distribution between two or more load balancer pools
- Maintenance schedules for the servers should ensure that Servers in Pool 1 are maintained separately to Pool 2.
- Business critical processes should always be allocated to Runtime Resources from each group, ensuring that a Resource will always be available to continue (albeit with a diminished throughput) in event of a failure or maintenance of the Servers within either pool.
- Whilst this does offer additional redundancy and protection during a failure or maintenance, Queues should be actively monitored to manage any exceptions.
- Interactive Clients are not shown on this diagram. These should be configured with connections to two or more pools, to allow for manual redirection in the event of an outage of an entire pool.

Logical Load Balanced Pools with Separated Interactive Clients



This scenario is identical to the last, except that Interactive Client connections are explicitly separated from the Application Servers serving the Runtime Resources. This ensures that maintenance of servers can be performed with a greater degree of certainty of the impact on users directly interacting with the Blue Prism Client than would otherwise be the case.

This design is only expected to be necessary or practical in the largest of environments, as many will not have sufficient Application servers to warrant it.

Configuring Blue Prism for Transport Encryption

This section is only relevant for the WCF: Transport Encryption and WCF: Transport Encryption with Windows Authentication connection modes. It is assumed that the required certificates are present and available to the BPService.exe program and the Blue Prism Server Service can start successfully.

[SSL Passthrough](#) and [SSL Termination](#) require slightly different Blue Prism configurations to ensure that communication works correctly. This is due to the following factors:

- The way that the load balancer handles incoming requests.
- The certificate which should be used depending on the load balancer configuration.

If the Blue Prism server certificate is configured incorrectly then the system will not function as the endpoint to which clients connect cannot be deemed trusted.

When using one of the WCF: Transport Encryption connection modes, the Application Server is usually configured to use a certificate with a common name (CN) of the Application Server, or an appropriate wildcard. The certificate common name is the key attribute and the certificate that is returned must match the server name configured in AutomateConfig.exe on the Runtime Resource or Interactive Client, as shown in the screenshot below.

Connection Configuration

Store and recall different database connections

Saved Connections

Default Connection

Current Connection

Connection Name: Default Connection
The name by which this connection will be remembered

Connection Type: Blue Prism Server
The type of connection to use

Blue Prism Server: myloadbalancer.mydomain.local
The hostname of the Blue Prism Server

Connection Mode: WCF: SOAP with Transport Encryption & Windows Authentication
This must match the mode configured on the Blue Prism Server(s)

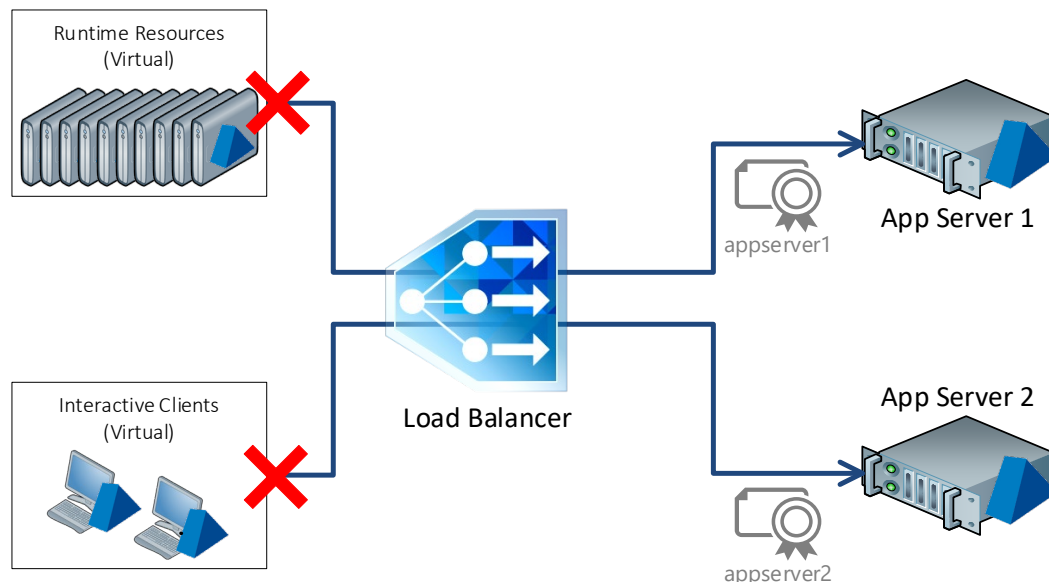
Server Port: 8199
This must match the listening port configured on the Blue Prism Server(s)

Test Connection

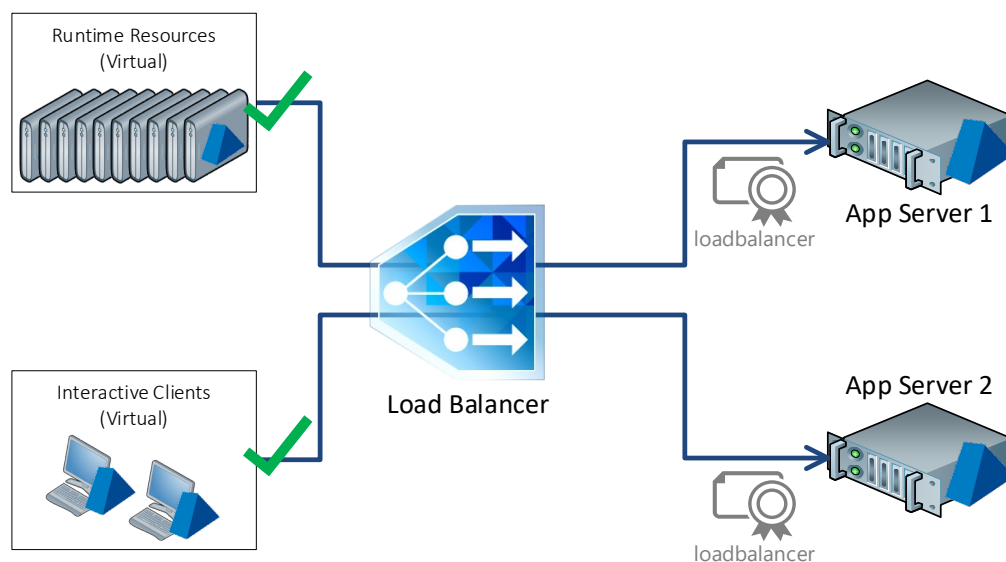
New Connection Delete Connection Create Database Upgrade Database Configure Database OK Cancel

Configuring Application Servers for SSL Passthrough

Whilst SSL Passthrough is a simpler concept than SSL Termination, the Blue Prism configuration for this mode is not as intuitive. With a load balancer configured to use SSL Passthrough, the connection from the Runtime Resource or Interactive Client is forwarded to the appropriate Application Server and returned by the load balancer with no interruption. This means that the certificate configured in the BPServer.exe application is returned. In this situation, a connection will be made to the load balancer and a certificate for an Application Server will be returned. This causes an issue because the expected certificate common name does not match the target - as shown below.



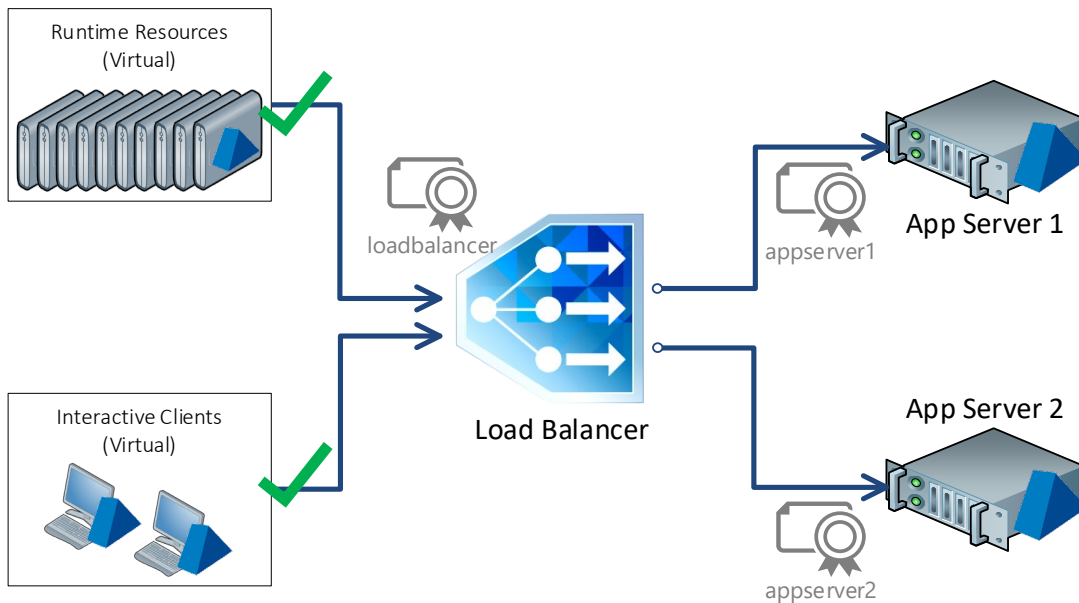
To ensure correct functionality when using SSL Passthrough, all Application Servers must be configured with a certificate that is applicable to the load balancer. The diagram below shows the Blue Prism Application Server Services configured to use a certificate configured for the load balancer.



Configuring Application Servers for SSL Termination

Due to the load balancer creating a new, separate connection to the target host in an SSL Termination scenario, individual, specific certificates can be deployed on Blue Prism Application Servers. In addition to Application Server certificates, the load balancer needs to have a certificate to ensure that the initial connection from Runtime Resource or Interactive Client is secure. As with the Application Server, the certificate used by the load balancer must have a common name that matches how Runtime Resources and Interactive Clients will make requests to the load balancer.

As the initial connection from Runtime Resource to Interactive Client is held whilst a separate connection is made to the Application Server, there is a need for different certificates in multiple locations.



Summary of Blue Prism Recommendations

This section outlines recommendations made by Blue Prism for ensuring smooth load balancer functionality and provides some troubleshooting and debugging information.

Blue Prism Recommendations

The following is a summary of the recommendations, in relation to the design of a Load Balanced Blue Prism solution in version 6:

- If available, use a Least Connections type of algorithm rather than Round Robin.
- If available, use active or passive health monitoring.
- Consider creating separate load balancing pools, aligned with a maintenance strategy, for the Blue Prism servers.
- Consider distributing runtime connections between different load balancing pools of Blue Prism application servers. This can be further enhanced by allocating workload to Runtimes that are connected to separate pools.
- In large environments, consider separating Runtimes and Interactive Client connections between different Blue Prism application servers and load balancing pools.
- Consider the availability of the load balancer itself - the design of the load balancer solution needs to be equal or greater than the availability profile for the Blue Prism environment.
- Ensure that the configured load balancer port matches that of the Blue Prism Application Server. For example, if the Application Servers are listening on port 8199, ensure that the load balancer is also listening on port 8199.
- If using a secure connection to the Application Server, ensure that the correct cryptographic libraries are configured and available on the load balancer. Sometimes, by default, these are not installed.

Troubleshooting an Issue in a Secure Deployment

The Application Server service in Blue Prism version 6 exposes a URL that is the main interaction point with the service. When accessing the load balancer URL via a web browser it can also be used to ascertain successful certificate configuration. The exposed URL would look like the URL below:

```
https://<loadbalancename>:<port>/bpserver
```

If a default WCF page is returned, then the configured certificates are correct. However, should a security warning be displayed then the certificates need to be corrected. For more information on this please see the *v6.3 Data Sheet - Securing Network Connectivity*, available from the Blue Prism Portal.

The successful display of the default WCF page is not itself indicative that Blue Prism will function correctly - its use here is to identify potential issues with certificate configuration. To ensure expected functionality, click **Test Connection** in AutomateConfig.exe which can be seen in [Configuring Blue Prism for Transport Encryption](#).