



Robotic Process Automation Software

Load Balancing

REFERENCE GUIDE

Version: 5.0.29

For more information please contact:

info@blueprism.com | UK: +44 (0) 870 879 3000 | US: +1 888 757 7476

www.blueprism.com

Table of Contents

1. Introduction	3
1.1. Intended audience.....	3
1.2. About this document.....	3
2. Load Balancing – Key Concepts and Terminology	4
2.1. Using DNS Round Robin.....	4
2.2. Using a Load Balancer (Application Delivery Controller)	5
2.3. Host	5
2.4. Service	5
2.5. Pool.....	5
2.6. Load Balancing Algorithm	7
2.7. Health Monitoring	8
2.8. Other Configuration Considerations.....	9
2.8.1. Session Stickiness (Persistence)	9
2.8.2. Direct Routing vs NAT Routing	9
3. Expected Behaviour and Design Scenarios.....	10
3.1. Expected Failure Behaviour	10
3.1.1. Blue Prism 5.0.24 and above.....	10
3.1.2. Blue Prism 5.0.23 and below	11
3.2. Designing for High Availability and Redundancy.....	12
4. Summary of Blue Prism Recommendations	14

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 – 2016

*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, Centrix House, Crow Lane East, Newton-le-Willows, WA12 9UY, United Kingdom
Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: www.blueprism.com

1. Introduction

1.1. Intended audience

This reference guide is intended for use by system architects and 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 topics on High Availability, which are covered in the Infrastructure Reference Guide.

1.2. About this document

The document provides an introduction to the key concepts involved in Load Balancing, the deployment options and implications for the Blue Prism application operation. Some understanding of the basic concepts and architecture would be expected of someone designing a solution that involves a load balancing solution.

2. Load Balancing – Key Concepts and Terminology

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 may be used in a Blue Prism environment. Some of the terminology is not industry standard and may vary between vendors, but should be generally applicable in most scenarios.

Note – The only documented and supported scenario for Load Balancing is between Runtime Resources or Interactive Clients and Application Servers.

2.1. Using DNS Round Robin

A basic level of Load Balancing between Blue Prism clients or Runtime Resources and the Blue Prism Application Server 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 with or without a hardware load balancer. When using DNS round robin, it is important to set a relatively low Time to Live (TTL) setting, to ensure that the DNS record is renewed quickly in event of the failure of an application server.

In its simplest implementation, Round-robin DNS works by responding to DNS requests with a list of potential IP addresses corresponding to several servers that host identical services. The order in which IP addresses from the list are returned is the basis for the term round robin. With each DNS response, the IP address sequence in the list is permuted. Usually, basic IP clients attempt connections with the first address returned from a DNS query, so that on different connection attempts, clients would receive service from different providers, thus distributing the overall load among servers.

Whilst this approach can be effective in distributing the workload of the Blue Prism environment between multiple Application Servers, it does have some limitations:

1. DNS caching – Caching within the DNS hierarchy can cause unpredictable results and, on its own, is **not a reliable mechanism for providing High Availability**, as the record for a server will continue to be served, even if that server is down.
2. Load distribution limitations – As the round robin effectively just alternates the order of the address records each time a name server is queried, this is not the most effective mechanism for load distribution.
3. There is no accounting for transaction time, server load or network congestion and no Application awareness, in event of an outage of a server.

2.2. Using a Load Balancer (Application Delivery Controller)

A load balancer (sometimes known as an ADC) is a device that distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) 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. Some examples of hardware load balancers would be F5 Big-IP Local Traffic Manager (LTM), Cisco Load Balancer, Citrix Netscaler and Fortinet. Examples of Software Load Balancers would include HAProxy and NGINX.

Load balancers are generally grouped into two categories: Layer 4 and Layer 7.

- 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 generally infers some form of Network Address Translation is employed. As .NET remoting is not supported over NAT, Blue Prism does not recommend the use of Layer 7 Load balancing.

Load balancers ensure reliability and availability by monitoring the health of applications and only sending requests to servers and applications that can respond in a timely manner, therefore there is an advantage in 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 section 2.6.

2.3. Host

Sometimes also known as the Server or Node. 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.

2.4. Service

Sometimes also known as the Member or Node. For the purposes of this guide, we will continue to use the term "Service". A Service is usually a little more defined than a server/node in that it includes the TCP port of the actual application that will be receiving traffic. For instance, a server named bpappserver001 may resolve to an address of 172.16.1.10, which represents the Host, and will have an application server running (by default on TCP port 8199), making the service address 172.16.1.10:8181. 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)

2.5. Pool

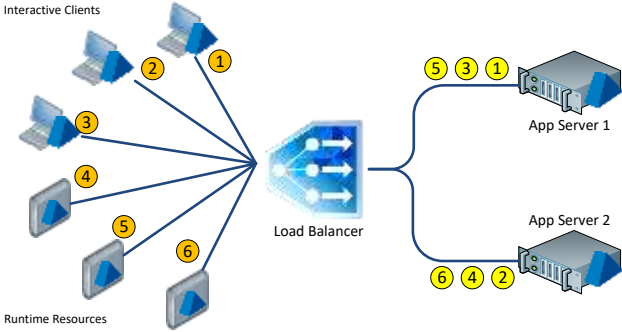
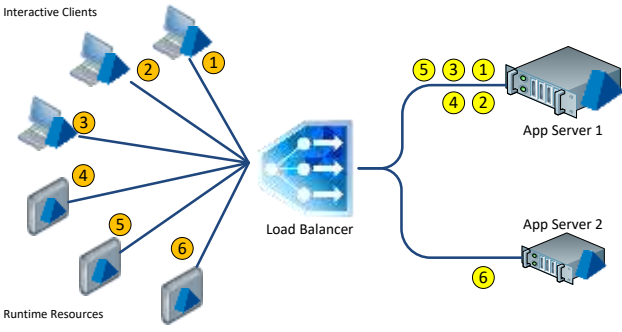
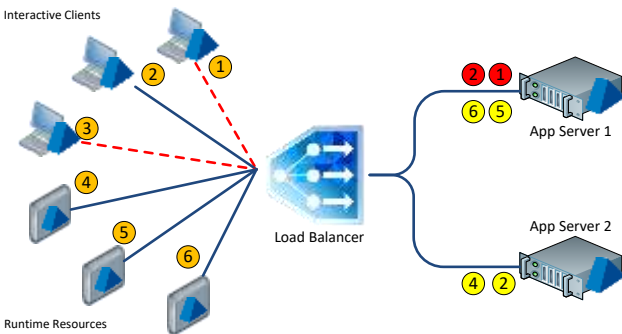
Load balancing allows organizations to distribute inbound traffic across multiple back-end destinations (in our case – Application Servers). It is therefore a necessity to have the concept of a collection of back-end destinations. Pools, also known as clusters or farms, are collections of similar services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

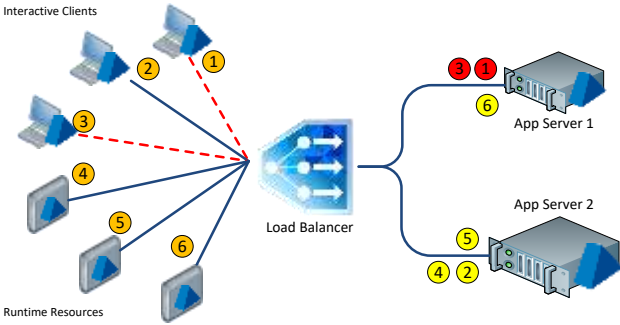
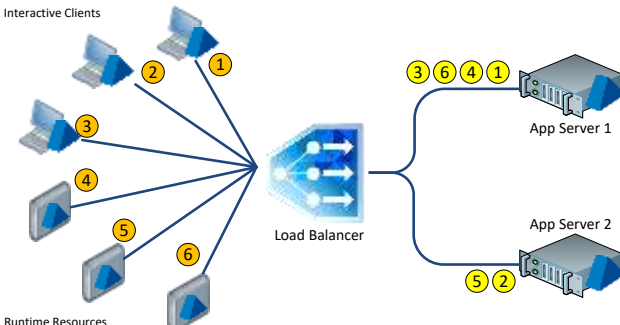
The key concept here is that all systems have a collective object that refers to "all similar services" and makes it easier to work with them as a single unit. This collective object—a pool—is almost always made up of services, not hosts (though it can be either). In our case, we would want to group application server "Services" into pools that are logically grouped to represent the desired distribution of connections. For example, it would be logical to

separate all DEV application server Services from 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 section 3.

2.6. Load Balancing Algorithm

The method by which a connection is routed to a Host / Service is determined by an algorithm. These again have some variation between solutions, but the most common are defined below:

Algorithm	Explanation
 <p>Round Robin</p>	<p>The concept of Round Robin is identical to when using DNS, except that more intelligence can be applied, by means of health monitoring and the use of Load Balancing Pools, to ensure that sessions are not allocated to a server which is unavailable.</p>
 <p>Weighted Round Robin</p>	<p>The Weighted Round Robin is similar to the Round Robin in a sense that the manner by which requests are assigned to the nodes is still cyclical, however a node with higher specs 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 specs should of course be given the higher weight. Usually weights are specified in proportion to actual capacities. So, for example, if Server 1's capacity is 5x more than Server 2's, it could be assigned it a weight of 5 and Server 2 a weight of 1. It would generally be expected that Application servers within a pool would have identical specifications, therefore this scenario is not normally applicable.</p>
 <p>Least Connections</p>	<p>This algorithm takes into consideration the number of current connections each server has. When a client attempts to connect, the load balancer will try to determine which server has the least number of connections and then assign the new connection to that server. This can be beneficial where the time that a connection will remain persistent is unpredictable (as can be the case with a mixture of Runtime Resource and Interactive Client connections), as this could cause one server to become more overloaded in a Round Robin scenario.</p> <p>So in the example diagram, if clients 5 and 6 attempt to connect after 1 and 3 have been disconnected, but 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.</p>

Algorithm	Explanation
 <p style="text-align: center;">Weighted Least Connections</p>	<p>Weighted Least Connections builds on the concept of Least Connection by introducing a weight to the algorithm.</p> <p>A load balancer that implements the Weighted Least Connections algorithm now takes into consideration two things: the weights/capacities of each server AND the current number of clients currently connected to each server, therefore in the example, the higher Weight of server 2 could cause connection 5 to be assigned to Server 2, despite it having more connections.</p>
 <p style="text-align: center;">Random</p>	<p>As its name implies, this algorithm matches clients and servers by random, i.e. using an underlying random number generator. In cases wherein the load balancer receives a large number of requests, a Random algorithm will be able to distribute the requests evenly to the nodes. So like Round Robin, the Random algorithm is sufficient for clusters consisting of nodes with similar configurations (CPU, RAM, etc).</p>

2.7. Health Monitoring

One of the advantages of a dedicated Load Balancing solution vs basic DNS round robin 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 simply by pinging it using ICMP or TCP_ECHO. This form of monitoring is not capable of determining the availability of the Blue Prism application, 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. Various methods are available, but the most commonly used method for Blue Prism environment would be to check the availability of the listening port of the application server.
- Passive Monitoring - Passive monitoring occurs as part of a client request. This kind 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 down.

2.8. Other Configuration Considerations

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

2.8.1. Session Stickiness (Persistence)

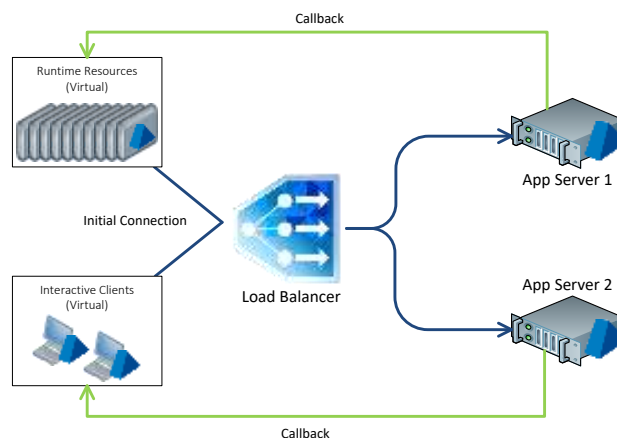
The stickiness (or persistence) of a session refers to the concept of ensuring that a connection from a device is always served to the same server. When a client connects to a service, the Load Balancer remembers the last connection for a specified period of time. If that same client IP address connects 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 Runtimes and Clients are established only once - when the connection is first established - therefore stickiness does not generally have a significant impact.

2.8.2. Direct Routing vs NAT Routing

Load balancers offer 2 mechanisms for routing. The configuration of this is critical to the successful use of Load Balancing solution for Blue Prism. Blue Prism employs the use of .NET remoting, which does not support a callback over NAT, thus the NAT routing scenario is not appropriate and Direct Routing must be used.

To ensure the successful communication between components, each Application server MUST be able to directly resolve the address of the client or Runtime, as only the initial connection will be initiated via the Load Balancer, after which the server must be able to directly address the calling client or Runtime.



3. Expected Behaviour and Design Scenarios

It is important to understand the expected behaviour of the Blue Prism application and define a strategy for maintenance and failover between components. The following sections are intended to explain the key concepts involved.

3.1. Expected Failure Behaviour

The table below outlines the expected behaviour of a Runtime and Interactive Client, in event of a dropped connection to its allocated Application Server.

3.1.1. Blue Prism 5.0.24 and above

Impacted Component	Expected Behaviour
Runtime Resource (5.0.24+)	<p>Runtime Resource connection would fail to communicate with the App Server. Presuming health monitoring is in place, the load balancer will have marked the service as "DOWN" and will redirect the connection to another application server service within its cluster.</p> <p>If this connection is achieved within 5 seconds of the communication failure, then the resource will immediately connect. Any active process will continue uninterrupted at this point, as long as the runtime is not attempting to read or write anything except session log data to the database. If the Runtime is attempting to read or write any other type of data at the point of communication failure, then the process would terminate.</p> <p>If there is no response within 5 secs, the active process will terminate and any queue item(s) locked by the process will be marked as an exception. The Runtime Resource attempts to reconnect every 2 minutes.</p> <p>When it reconnects, it will become online and available for work again.</p>
Interactive Client (5.0.24+)	<p>Interactive Client connection would fail to communicate with the App Server. Presuming health monitoring is in place, the load balancer will have marked the service as "DOWN" and will redirect the connection to another application server service within its cluster.</p> <p>If this connection is achieved within 5 seconds of the communication failure, then the client should immediately reconnect without error.</p> <p>If there is no response within 5 secs, the user will receive a number of on screen notifications within the GUI until the connection is re-established.</p>

3.1.2. Blue Prism 5.0.23 and below

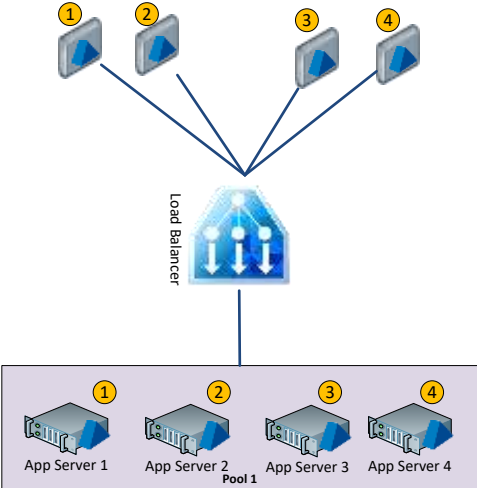
Impacted Component	Expected Behaviour
Runtime Resource (5.0.23 and below)	<p>Runtime Resource connection would fail to communicate with the App Server. Presuming health monitoring is in place, the load balancer will have marked the service as “DOWN” and will redirect the connection to another application server service within its cluster.</p> <p>If the connection is re-established to the SAME IP address, within 5 seconds of the communication failure, then the resource will immediately connect. Any active process will continue uninterrupted at this point, as long as the runtime is not attempting to read or write anything except session log data to the database. If the Runtime is attempting to read or write any other type of data at the point of communication failure, then the process would terminate.</p> <p>If the connection is routed to a different IP address (which would be expected in a load balancing scenario), then the process will terminate and the runtime will need to be restarted to ensure that it picks up the correct IP address from DNS.</p> <p>If there is no response within 5 secs, the active process will terminate and any queue item(s) locked by the process will be marked as an exception. The Runtime Resource attempts to reconnect every 2 minutes.</p>
Interactive Client (5.0.23 and below)	<p>Interactive Client connection would fail to communicate with the App Server. Presuming health monitoring is in place, the load balancer will have marked the service as “DOWN” and will redirect the connection to another application server service within its cluster.</p> <p>If the connection is re-established to the SAME IP address, within 5 seconds of the communication failure, then the client will immediately reconnect without error.</p> <p>If there is no response within 5 secs, or the connection is established with the DIFFERENT IP address, the user will receive various errors and popups within the GUI and will need to restart the client.</p>

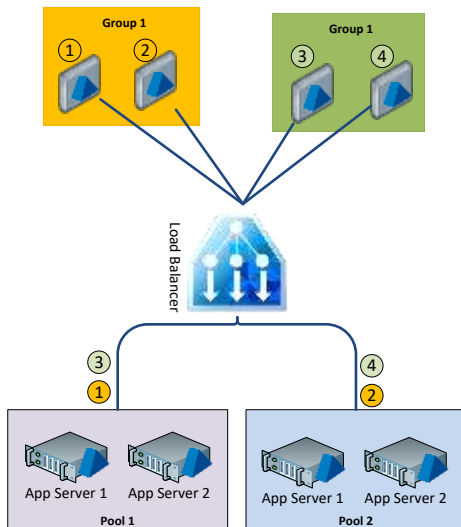
3.2. Designing for High Availability and Redundancy

Using a Load Balancer solution for the purposes of High Availability vs application distribution requires an additional degree of care and attention that must take into account (as a minimum) the following:

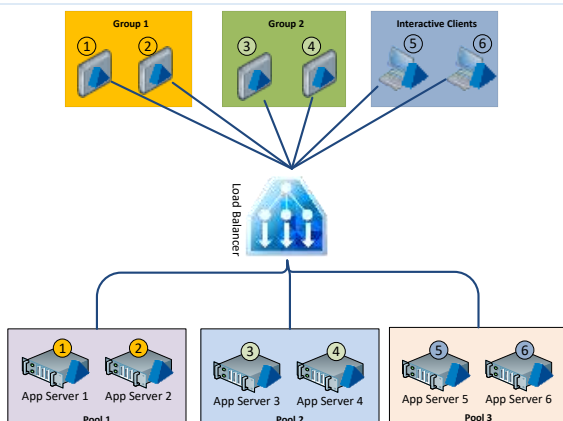
- Maintenance approach – How the Application Servers will be grouped and maintained, to ensure the availability of a device for the Runtime and Interactive Client connections.
- Infrastructure Availability vs Business Process Availability – Simply considering the availability of the Application servers 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 to use a combination of Balanced Application Server Pools and resource grouping. The table below outlines some common and recommended scenarios, along with the implications for High Availability.

Scenario	
	<ul style="list-style-type: none"> • In a scenario with one single Load Balancing pool, it is difficult, if not impossible, 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.



- Application Servers are grouped into logical Load Balanced Pools. This may be also based on location, to account for DR scenarios.
- Runtime Resources are grouped, with distribution between 2 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, thus 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 2 or more pools, to allow for manual redirection, in the event of an outage of an entire pool



- 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.

4. Summary of Blue Prism Recommendations

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

- Use a Low TTL setting, if using DNS Round Robin
- Consider using a hardware or software LB vs DNS if High Availability is required and / or the non-functional requirements of the solution require it. If they do not, a load balancer may add unnecessary complexity.
- Use a Least Connections type of algorithm vs Round Robin, if available
- Use active or passive health monitoring if using a hardware or software Load Balancer
- Consider creating separate Load Balancing Pools, aligned with a 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
- Use Direct routing vs NAT routing
- Consider the availability of the Load Balancer itself - the design of the Load Balancer solution needs to be equal or greater to the availability profile for the Blue Prism environment.