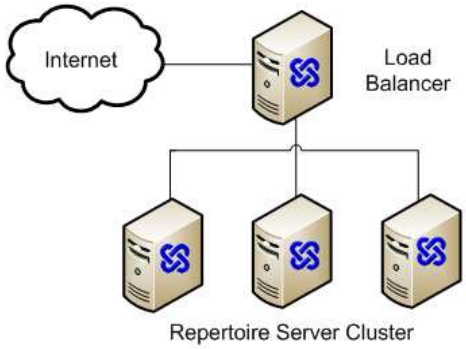
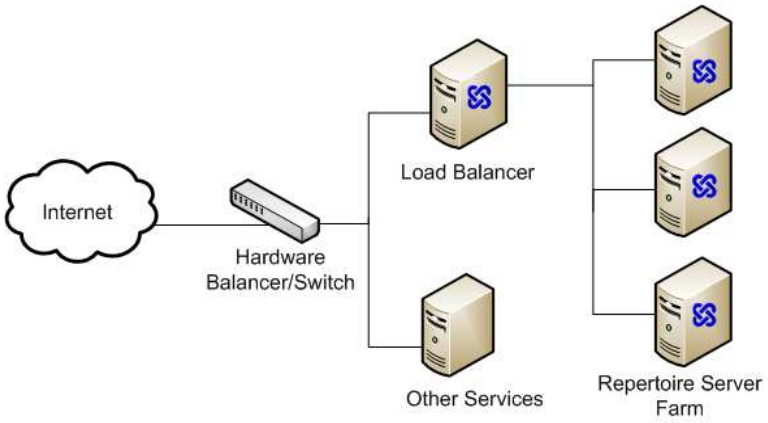
ELIXIR LOAD BALANCER 2

Overview

Elixir Load Balancer for Elixir Repertoire Server 7.2.2 or greater provides software solution for load balancing of Elixir Repertoire Servers. As a pure Java based software application, it is platform independent and can run on any J2SE enabled OS platforms that has Java version 1.5.0 or greater. Elixir Load Balancer supports a wide range of platforms and operating systems and is able to run on commodity hardware. Therefore, its performance can easily and cost-effectively grow as required.



Elixir Load Balancer can complement other hardware-only load balancing solutions in very high traffic environments. Hardware Load Balancers are deployed at the edge of hosting provider's network, making simple packet switching decisions at very high speed. Load Balancer is used to handle Repertoire Server specific requests. It is also able to handle almost any TCP/IP-based protocol.

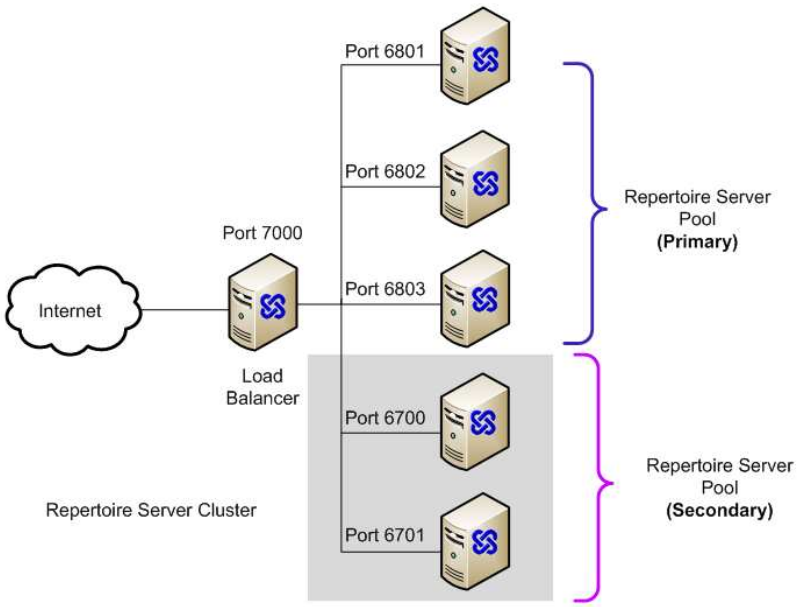


Availability

There are two ways to configure the Load Balance to improve availability, which are Static Server Configuration or Dynamic Server Configuration.

(A) Static Server Configuration

Load Balancer provides seamless Repertoire Server fail over. A primary and secondary server cluster needs to be setup. The primary cluster is a group of Repertoire Servers that will service report requests while the secondary cluster consists of Repertoire Servers are in hot standby. In the event of a Repertoire Server failure, the requests will be piped automatically to a Repertoire Server in the secondary cluster. The affected Repertoire Server(s) in the primary cluster will be removed from the server pool. User can reactivate the Repertoire Server(s) once the issue is resolved via the web console. The detection of failure Repertoire Server is only done when connection request is triggered to reduce unnecessary network traffic which is usually caused by active monitoring.



In addition, we provide a service wrapper that can be installed as part of Windows Services to monitor Java Virtual Machine. The service is able to automate the restarting of the Load Balancer in the event of a JVM fault (crash) or restart of hardware.

Auto detection of recovery of failed Repertoire Server is provided. On detection of the server recovery, it will be included as part of the server pool through active monitoring of the IP and port. To increase fault tolerance, user should use a hardware/OS clustering solution to handle hardware failure.

(B) Dynamic Server Configuration

Load balancer can be configured to support dynamic server discovery. A multi cast IP address and port is setup for the Load Balancer and the Repertoire Servers. At intervals, the Load Balancer will broadcast a message and listen for responses from available the Repertoire Servers. New servers will be added to the Load Balancer's list of available servers. If any of the servers in list failed to response to the Load Balancer broadcast, it will be removed from that list.

Before the Repertoire Server is added or be removed from list, the Load Balancer will probe to reconfirm the availability of the server on the actual Repertoire Server port.

In order for the Dynamic server configuration to work, firstly configure the Repertoire Server to support dynamic server discovery, the MBean, *ERS2:name=PingListener* needs to be enabled in the ERS2.xml file of Repertoire Server configuration directory.

```
<ers:mbean name="ERS2:name=PingListener" class="com.elixirtech.ers2.ping.Ping">
<ers:property name="Port">7001</ers:property>
        <ers:property name="Address">224.5.12.24</ers:property >
        <ers:property name="Timeout">30000</ers:property >
</ers:mbean>
```

As for the Load Balancer configuration, MBean, *ERS2:name=DynamicInstanceConfigurator* needs to be enabled in ERS2-LoadBalancer.xml configuration file of the Load Balancer configuration directory.

```
<ers:mbean name="ERS2:name=DynamicInstanceConfigurator"
class="com.elixirtech.ers2.lb.DynamicInstanceConfigurator">

        <ers:property name="Port">7001</ers:property>

        <ers:property name="Address">224.5.12.24</ers:property>

        <ers:property name="Timeout">5000</ers:property>

        <ers:property name="BroadcastInterval">30000</ers:property>

</ers:mbean>
```

The IP address is the standard multicast group is specified by a class D IP address and by a standard UDP port number. Class D IP addresses are in the range 224.0.0.0 to 239.255.255.255, inclusive. The address 224.0.0.0 is reserved and should not be used.

Server Affinity

Load Balancer allows the client to connect to the same Repertoire Server after the http session is authenticated. This association of session with backend server is call sticky session. The Repertoire Server origin is encoded in the session cookie for Load Balancer to locate the appropriate Repertoire Server to send the request.

In the Load Balancer configuration file, ERS2-LoadBalancer.xml, set the parameter *UseServerAffinity* to true.

```
<ers:mbean name="ERS2:name=LoadBalancer" class="com.elixirtech.ers2.lb.LoadBalancer">
 ……
        <ers:property name="UseServerAffinity">true</ers:property>

</ers:mbean>
```

Repeat the same for the Repertoire Server configuration file, ERS2.xml, at the Jetty MBean configuration set as enable.

```
<ers:mbean name="ERS2:name=Jetty"  class="com.elixirtech.jetty.JettyLauncher">
…….
        <ers:property name="ServerAffinityEnabled">true</ers:property>
</ers:mbean>
```

The session cookie name can also be modified. This allow you configure a different cookie name i.e. JSESSION to a different one. The properties set in the

```
<ers:mbean name="ERS2:name=LoadBalancer" class="com.elixirtech.ers2.lb.LoadBalancer">
  ……
        <ers:property name="ServerAffinityCookieName">JSESSIONID</ers:property>

</ers:mbean>
```

Administration

The Load Balancer provides a Web administration screen to allow a user to view the configuration and the current state of the Load Balancer. It is based on JMX J2EE platform.

Load Balancer Web administration: http://localhost:8090

Configuration is fairly straightforward. All configurations are done via an xml file where you can specify the Load Balancer's port and configure the servers as primary and secondary server pool.

Load Balancing Algorithm

Load Balancer provides Lowest Workload and Round Robin balancing algorithms. For Round Robin balancing algorithm, new requests are being distributed between the Repertoire Servers evenly. As for the Lowest Workload balancing algorithm, new requests are being assigned to the Repertoire Server instance with the lowest workload. This maximizes the Repertoire Server resources to handle load.

Configuration of Load Balancer

System Requirement

- Java VM J2SE version 1.5.0 or greater
- Recommended minimum physical memory of 128MB
- Shipped with Windows service wrapper. For non-Windows daemon, please see http://wrapper.tanukisoftware.org/doc/english/index.html for details

Configuration

All the configuration information is located in the Load Balancer\config\ERS2-LoadBalancer.xml.

There are three sets of configurable items provided.

(i) Load Balancer

This allows users to configure the port and the number of servers to be allocated in the primary server pool.

Port	This determines the port that requests are submitted to from the client.
ActivationCount	This determines the number of servers that are setup to be in the primary pool.

Setup example: When the Load Balancer is setup to listen to incoming client report request connection for port 7000 and three report servers setup to process report from the primary pool while the rest will be in the secondary server pool in hot standby mode.

```
<ers:mbean name="ERS2:.name=LoadBalancer" class="com.elixirtech.ers2.lb.LoadBalancer">
        <ers:property name="Port">7000</ers:property>
        <ers:property name="ActivationCount">3</ers:property>
</ers:mbean>
```

Configuring Limit

When the number of clients on each instance hits the limit set in ClientCountLimit, a warning message will be shown in the log file. When the number of clients on each instance hits the limit specified in ClientCountMaximumLimit, that particular instance will be removed from the primary server pool and will not accept any new requests. During this period of time, any connection time that is greater than the value specified in ClientCountMaximumAgeLimit will be cleared from the instance. The instance will wait for the time specified in InstanceWaitTime before it will reset and go back to the primary server pool to accept new requests.

SocketTimeout	Enable/disable SocketTimeout with the specified timeout, in milliseconds. The timeout must be > 0. A timeout of zero is interpreted as an infinite timeout.
SocketLingerTimeout	Controls how long it waits to close when there are still unsent data.
SocketAlive	Whether or not to have socket keep alive turned on. It is either set to true or false.
RecoveryWaitTime	The amount of waiting time in milliseconds before the recovery of an instance. If set to zero, there will be no recovery of dead instances.
ClientCountLimit	The count limit for the number of clients on each instance. When the limit is reached, there will be a warning in the log. If set to zero, there will be no limit to the number of clients.
ClientCountMaximumLimit	The maximum number of clients on each instance. When the limit is reached, the instance will be removed from the primary server pool.
InstanceWaitTime	The time in milliseconds the instance will wait before it will reset.
ClientCountMaximumAgeLimit	Any connection greater than the client age limit will be cleared

	from the instance.

Server Affinity

ServerAffinityEnabled	Enable the Load balancer to direct the request to its origin.
ServerAffinityCookieName	This is the session cookie name use by the Repertoire to keep session based information. Load Balancer uses this to locate which server instance the connection originates from.

(ii) Repertoire Server Instance

(A) Static Server Configuration

This specifies the Reportoire Server TCP/IP address or host name and its port that a report request can be directed to.

Host	Repertoire Server host name or TCP/IP address.
Port	Repertoire Server listener port.
ERS2Instance:name	Internal name that the Load Balancer refers to. Determined by the user.

```
<ers:mbean name="ERS2Instance:name=First" class="com.elixirtech.ers2.lb.ERS2Instance">
        <ers:property name="Host">192.168.1.1</ers:property>
        <ers:property name="Port">6998</ers:property>
</ers:mbean>
```

(B) Dynamic Server Configuration

This allows the auto discovery Repertoire Server instance in the network.

Address	Broadcast address (Multicast IP)
Port	Broadcast port.
BroadcastInterval	Length time (ms) before it broadcast for response from servers

Timeout	Length time (ms) to wait for incoming Repertoire Servers responses.

(iii) Web Console Configuration

This specifies the port that the user can connect to the web management console of the Elixir Load Balancer. Load Balancer Web console can be disabled by commenting it out.

Port	Port that the user can connect to via the web browser.

```
<ers:mbean name="ERS2:name=html" class="com.sun.jdmk.comm.HtmlAdaptorServer">
        <ers:property name="Port">8089</ers:property>
</ers:mbean>
```

(iv) Monitoring of Failed Servers (optional)

In the event that a failed server is detected, it will be taken out of the pool and place in a monitoring queue. At regular intervals (user configurable), the Load Balancer will check. If the server responses, it will be restored to the queue. This monitoring behavior can be removed by commenting it out of the configuration script.

Wait	Time to wait till the next monitoring for server activity. (in milliseconds)

```
<ers:mbean name="ERS2:name=MonitorDeadServers"
class="com.elixirtech.ers2.lb.MonitorDeadServerInstance">
        <ers:property name="Wait">3600000</ers:property>
</ers:mbean>
```

Configuration As Service

Elixir Load Balancer can run as a Window service or Unix daemon via Java Service wrapper. Shipped in our package is the Windows version of the service wrapper. For those who want the Load Balancer in a non-Windows environment, please download the respective OS configurable at:

http://wrapper.tanukisoftware.org/doc/english/index.html

When the Load Balancer is run as a Windows service or Unix daemon, the startup and shutdown process is controlled by the service wrapper.