

webMethods EntireX

High Availability

Innovation Release

Version 9.9

October 2015

This document applies to webMethods EntireX Version 9.9 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1997-2015 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: EXX-HIGHAVAIL-99-20171128

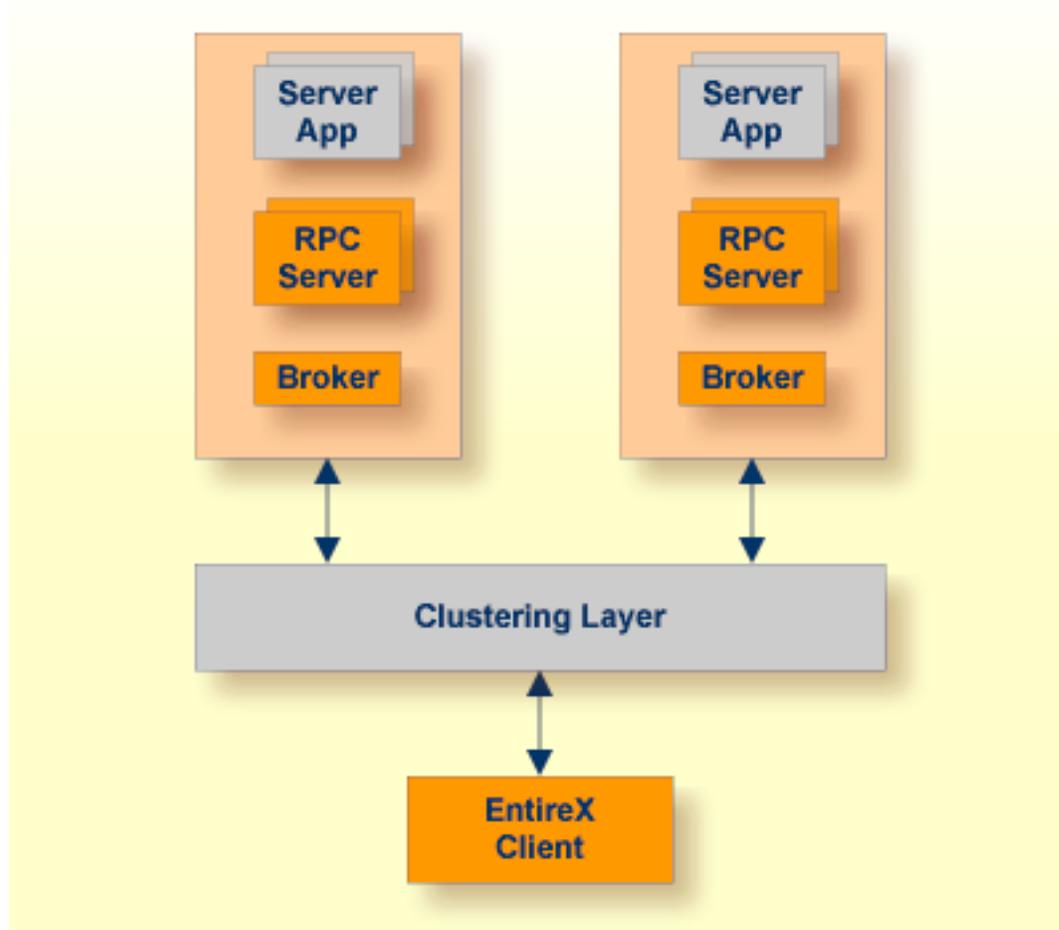
Table of Contents

High Availability in EntireX	v
1 Introduction to High Availability	1
Purpose of Clustering	2
Why is High Availability Important	2
Clustering for High Availability, Load Balancing and Fault Tolerance	3
Advantages of Network-based Clustering	3
Virtual IP Addressing	4
Client Considerations	4
2 Commonly Used High Availability Terms	7
3 High Availability with Sysplex under z/OS	9
4 Setting up your EntireX Environment for Sysplex	11
Setting Broker Attributes	12
Broker Cluster Considerations	13
Configuring Redundant RPC Servers	13
Verifying your Configuration	14
Managing Brokers and RPC Servers	17
5 High Availability with NLB under Windows	21
6 Setting up your EntireX Environment for NLB	25
Setting Broker Attributes	26
Broker Cluster Considerations	27
Configuring Redundant RPC Servers	28
Managing Brokers and RPC Servers	28
Restrictions	31
7 High Availability with Other Clustering Technologies	33
8 Setting up Your EntireX Environment for High Availability with Other Clustering Technologies	35
Setting Broker Attributes	36
Broker Cluster Considerations	37
Configuring Redundant RPC Servers	37
Managing Brokers and RPC Servers	38

High Availability in EntireX

Under High Availability we understand an environment with engineered redundancy which, if any one component fails, guarantees the integrity of the system as a whole. To achieve high availability, EntireX uses existing third-party clustering technology.

This document provides an introduction to clustering technology and describes how to set up in EntireX the redundant RPC servers and brokers you will need to achieve high availability. Please note that this document applies to simple synchronous scenarios only; see [Client Considerations](#) for details and prerequisites.



The scenario you choose depends on the platform where your clustering environment is set up, typically the environment where your broker is running:

- [High Availability with Sysplex under z/OS](#)

- *High Availability with NLB under Windows*
- *High Availability with Other Clustering Technologies*

See also list of *Commonly Used High Availability Terms* and *Introduction to High Availability*.

1 Introduction to High Availability

- Purpose of Clustering 2
- Why is High Availability Important 2
- Clustering for High Availability, Load Balancing and Fault Tolerance 3
- Advantages of Network-based Clustering 3
- Virtual IP Addressing 4
- Client Considerations 4

Purpose of Clustering

When determining how to increase availability and to decrease downtime for important applications, there are many different clustering solutions to consider. It is important to start any availability improvement discussion however, with a preemptive understanding of what it is you want to improve. Begin by looking at the history of your application failures, downtime scenarios, and the underlying causes of availability issues. Map out your application topology and network infrastructure such that you can ascertain potential weak spots or single points of failure and prioritize component redundancy based on exposed risk to the overall system availability. Ask yourself the following questions:

- What is it that I want to accomplish from clustering?
- Who are the stakeholders?
- How will I measure availability improvement?

Once you have a prioritized list of availability improvement objectives, look for solutions that address these points of failure and look to provision an implementation plan that ensures a pre-arranged level of operational performance will be met during a contractual measurement period. Improvement must be measurable in both the technical and business perspectives.

Why is High Availability Important

Many of the world's largest organizations including financial institutions, manufacturing, transportation, and communication companies along with large government agencies, rely on the availability and reliability of applications to deliver their most important business transactions and data. These large-scale information systems consist of several hardware and software components, each of which performs a particular function and is a critical-path to successful daily operations. If any of these components fail however, the outcome could vary drastically - from a single user experiencing a slow-down in their order response time, to thousands of retail bank customers not being able to access any of their cash assets. Creating a highly available system topology removes any single points of failure from a large system, enabling another redundant component to effectively take over the workload of the failed component. Improving availability ultimately leads to a reduction in downtime, improved business performance, and a better user experience.

Another IT concern is how to apply maintenance and upgrades to these important systems without affecting users. In a highly available world, a system in need of maintenance can be taken out of the workload pool and updated while the rest of the system continues to process service requests.

Clustering for High Availability, Load Balancing and Fault Tolerance

As previously mentioned, there is a wide variety of clustering techniques that are designed to accomplish specific improvement in application availability based on planned or unplanned events. Planned events are typically scheduled maintenance activities associated with specific fixes or general upgrades. In this case, clustering can be utilized to maintain processing workloads while certain instances or services of the cluster are brought down, updated, and rejoined to the cluster.

Unplanned events require a means of automated failover whereby work is picked up by a pooled resource. Cluster architectures vary in how they handle fault tolerance, recovery, and guarantee delivery. Each high availability solution has a different architecture or technique in which work is redirected to or picked up by an available process. There are shared memory solutions (e.g. Terracotta Server Array), shared data store (e.g. Integration Server cluster), shared message queue (e.g. Universal Messaging), and shared virtual IP (e.g. EntireX Broker) among the list of possible solutions. Each technique provisions a group or cluster of common processes working on in-flight data or messages that may or may not be persisted and coordinated by the state of the application endpoints.

Advantages of Network-based Clustering

While a loosely coupled system such as network clustering cannot recover or coordinate distributed work, it does protect against a wide range of failures up and down the stack including hardware, OS, and application failures. Network-based HA solutions are relatively easy to configure and work transparently with stateless applications.

Another advantage is to address system availability during planned events such as applying maintenance patches or upgrades. For example, when a major or minor update is required to be performed to a broker, it is important that the system remains operational during this planned event. In this case, individual broker instances are taken out of the cluster without impacting the overall operation of the system. As updates are completed, Brokers can individually be added back into the cluster independent of their version.

Virtual IP Addressing

Traditionally, an IP address is associated with each end of a physical link (or each point of access to a shared-medium LAN), and the IP addresses are unique across the entire visible network, which can be the Internet or a closed intranet. The majority of IP hosts have a single point of attachment to the network, but some hosts (particularly large server hosts) have more than one link into the network.

A TCP/IP host with multiple points of attachment also has multiple IP addresses, one for each link. Within the IP routing network, failure of any intermediate link or adapter disrupts end user service only if there is not an alternate path through the routing network. Routers can route IP traffic around failures of intermediate links in such a way that the failures are not visible to the end applications or IP hosts. However, because an IP packet is routed based on ultimate destination IP address, if the adapter or link associated with the destination IP address fails, there is no way for the IP routing network to provide an alternate path to the stack and application.

Endpoint (source or destination) IP adapters and links thus constitute single points of failure. While this might be acceptable for a client host, where only a single user will be cut off from service, a server IP link might serve hundreds or thousands of clients, all of whose services would be disrupted by a failure of the server link.

The virtual IP address (VIPA) removes the adapter as a single point of failure by providing an IP address that is associated with a stack without associating it with a specific physical network attachment. Because the virtual device exists only in software, it is always active and never experiences a physical failure. A VIPA has no single physical network attachment associated with it.

Client Considerations

Only synchronous, non-conversational application scenarios are supported. Additional prerequisites apply to client applications:

- [No Persistent Sockets \(Socket Pooling\)](#)
- [Socket Reconnect](#)
- [Security Handling](#)

- [Matrix of Supported Features](#)

No Persistent Sockets (Socket Pooling)

Socket pooling needs to be explicitly disabled for all EntireX clients, except webMethods EntireX Adapter for Integration Server. See [Matrix of Supported Features](#).

Socket Reconnect

Client applications connected to a broker instance may need to react when this broker instance becomes unavailable and the cluster system establishes connection to a different broker instance.

Most EntireX clients support some reconnect logic on socket disconnect if the cluster system routes the connection to a different broker instance. However, the Java RPC and XML/SOAP client Java API do not support automatic reconnect. This needs to be handled by the client application logic. The XML/SOAP Listener does not support socket reconnect.

Security Handling

If the Brokers in the cluster have security enabled, client applications need to re-authenticate with a new Broker instance on reconnect.

For Java RPC and XML/SOAP client Java API, re-authentication has to be completely handled by the client application. The XML/SOAP Listener does not support automatic re-authentication.

Matrix of Supported Features

In the table below, “yes” means the feature is handled automatically and no user action or configuration is required; “no” means the feature is not supported; and “application” means that the client application must be adapted accordingly.



Note: This table assumes you are using the latest version of the components listed.

RPC Client	No Persistent Sockets	Socket Reconnect	Security Handling
EntireX Adapter for IS	yes	yes	yes
RPC-ACI Bridge	Specify <code>socketpoolsize=0</code> as part of the broker ID. See <i>Socket Parameters for TCP and SSL/TLS Communication under Writing Advanced Applications - EntireX Java ACI</i> .	yes	yes
WebSphere MQ Listener	Specify <code>socketpoolsize=0</code> .	yes	yes
SAP XI Adapter	Specify <code>socketpoolsize=0</code> .	yes	yes
Java RPC	Specify <code>socketpoolsize=0</code> .	application	application
XML/SOAP client API	Specify <code>socketpoolsize=0</code> .	application	application

RPC Client	No Persistent Sockets	Socket Reconnect	Security Handling
XML/SOAP Listener	Specify <code>socketpoolsize=0</code> .	no	no
Natural RPC	Specify <code>ETB_SOCKETPOOL=OFF</code> . See <i>Support of Clustering in a High Availability Scenario under Administration of Broker Stubs</i> in the platform-specific administration documentation.	yes	application
C RPC	Specify <code>ETB_SOCKETPOOL=OFF</code> .	yes	application
COBOL RPC	Specify <code>ETB_SOCKETPOOL=OFF</code> .	yes	application
PL/I RPC	Specify <code>ETB_SOCKETPOOL=OFF</code> .	yes	application
.NET Wrapper	Specify <code>ETB_SOCKETPOOL=OFF</code> .	yes	application
DCOM Wrapper	Specify <code>ETB_SOCKETPOOL=OFF</code> .	yes	application

2 Commonly Used High Availability Terms

Clustering

Clustering is a technique to improve availability by configuring redundant resources such that if one instance fails it does not disrupt system availability to users.

DVIPA

Dynamic virtual IP address (DVIPA) is a z/OS Communications Server feature used to describe the automatic failover of TCP/IP stack instances within a Sysplex cluster.

See [Setting up your EntireX Environment for Sysplex](#).

Failover

Feature used to describe the control switch between redundant resources in the event of a resource failure.

Fault Tolerance

Describes the process of delivering continuous operation in the event of an unplanned outage.

High Availability

Under High Availability we understand an environment with engineered redundancy which, if any one component fails, guarantees the integrity of the system as a whole. To achieve high availability, EntireX uses existing third-party clustering technology.

HiperSockets

HiperSockets is an IBM technology for offering TCP/IP communications between partitions at in-memory speed.

Load Balancing

A network-based technique to distribute common workload across multiple back-end services that can be evenly distributed or routed based on weighted inference of a key performance attribute (commonly called filtering).

LPAR

A logical partition (LPAR) describes the virtualization of the System z hardware platform; if one LPAR goes down it has no effect on any others running on the same hardware.

NLB

Network Load Balancing (NLB) is a clustering technology for Windows servers that enhances the scalability and availability of mission-critical, TCP/IP-based services, such as Web, Terminal Services, virtual private networking, and streaming media servers.

See [Setting up your EntireX Environment for NLB](#)

Node Convergence

The process of converging static machine addresses into a clustered IP address on Windows.

Queued Direct IO

Queued Direct IO is a System z hardware feature offering an enhanced data transfer architecture for improving data transfer speed and efficiency for TCP/IP traffic.

Sysplex

A Parallel Sysplex is a cluster of IBM mainframes acting together as a single system image with z/OS. Parallel Sysplex combines data sharing and parallel computing to allow a cluster of up to 32 systems to share a workload for high performance and high availability.

See [Setting up your EntireX Environment for Sysplex](#)

Transaction Recovery

A feature for recovering resources in the event of an unplanned outage.

VIPA

Virtual IP addressing is an IP address assigned to multiple static IP addresses to make them accessible as one address for the purpose of improving availability or load balancing.

WLM

IBM Workload Manager is a Sysplex based load balancer offering workload filtering.

24x7

A business definition for highly available applications representing 24 hours, 7 days a week availability.

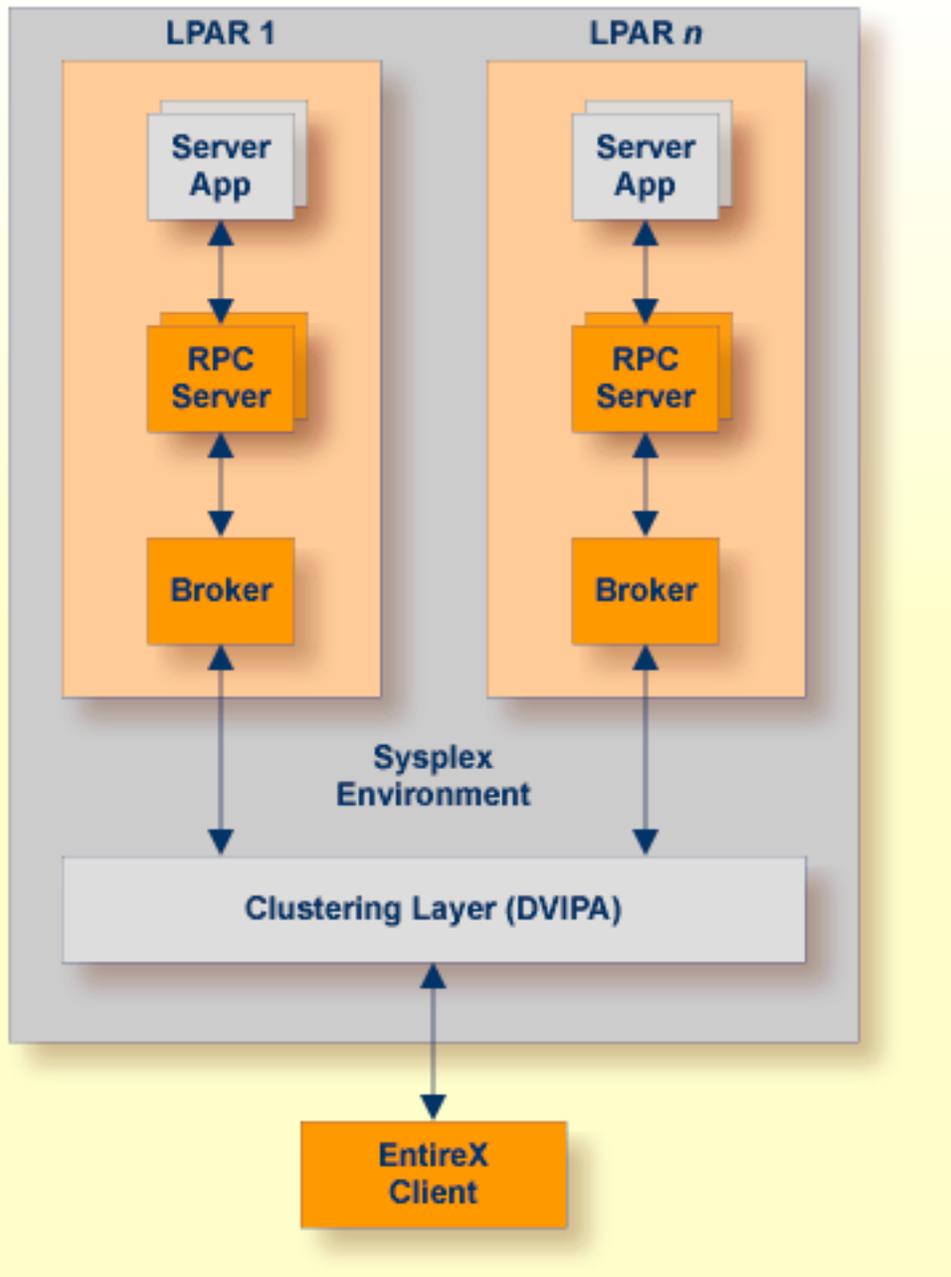
99.999

A business definition for highly available applications representing a percentage very close to 100 % availability.

3

High Availability with Sysplex under z/OS

This graphic shows a cluster environment using IBM Sysplex technology to achieve high availability. Clustering is a technique to improve availability by configuring redundant resources - in the case of EntireX this means brokers and RPC servers. This redundancy ensures that if one instance fails the integrity of the system is not affected. Up to 32 logical partitions (LPARs) - often geographically distributed - can act together as a single system image. Dynamic virtual IP address (DVIPA) is a z/OS Communications Server feature used to describe the automatic failover of TCP/IP stack instances within a Sysplex cluster.



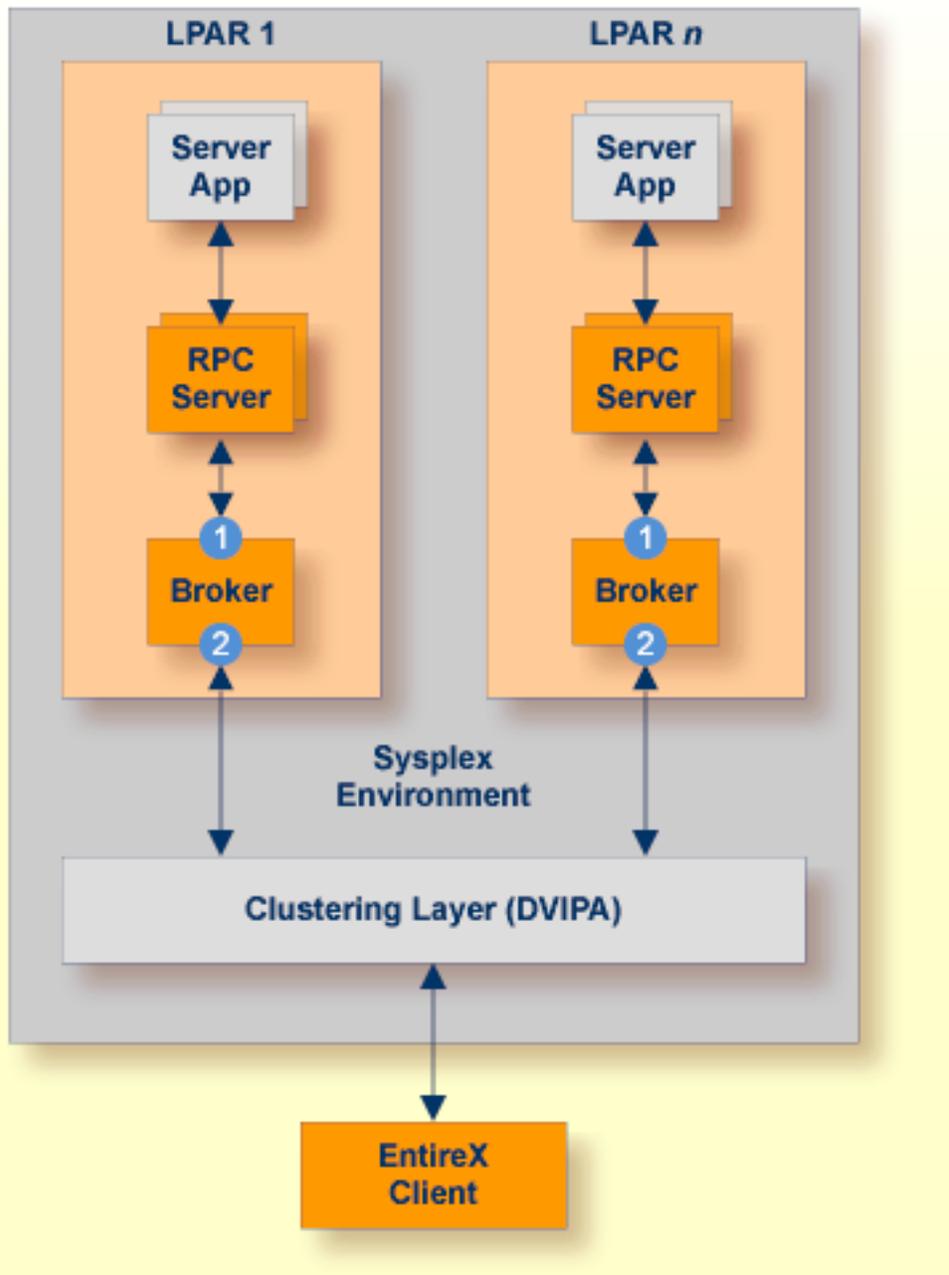
For more details, see [Setting up your EntireX Environment for Sysplex](#).

4 **Setting up your EntireX Environment for Sysplex**

- Setting Broker Attributes 12
- Broker Cluster Considerations 13
- Configuring Redundant RPC Servers 13
- Verifying your Configuration 14
- Managing Brokers and RPC Servers 17

Scenario: "I want to use Sysplex for my high availability cluster."

Setting Broker Attributes



Segmenting dynamic workload from static server and management topology is critically important. Using broker TCP/IP-specific attributes, define two separate connection points:

- One for RPC server-to-broker and admin connections. ⁽¹⁾
- The second for client workload connections. ⁽²⁾

See *TCP/IP-specific Attributes* (DEFAULTS=TCP) under *Broker Attributes* in the platform-independent administration documentation. Sample attribute file settings:

	Sample Attribute File Settings	Note
(1)	PORT=1972	In this example, the HOST is not defined, so the default setting will be used (localhost).
(2)	HOST=10.20.74.103 (or DNS) PORT=1811	In this example, the HOST stack is the virtual IP address. The PORT will be shared by other brokers in the cluster.

Broker Cluster Considerations

Platform-independent Considerations

We recommend the following:

- Share configurations - you will want to consolidate as many configuration parameters as possible in the attribute setting. Keep separate yet similar attribute files.
- Isolate workload listeners from management listeners.
- Monitor Brokers through SMH.

z/OS-specific Considerations

- Use Started Task names that match EntireX Broker naming conventions and have logical context.
- z/OS supports multiple TCP/IP stacks per LPAR. EntireX Broker supports up to eight separate listeners on the same or different stacks.

Configuring Redundant RPC Servers

In addition to broker redundancy, you also need to configure your RPC servers for redundant operations. We recommend the following best practices when setting up your RPC servers:

- **General Hints**
 - Make sure your definitions for CLASS/SERVER/SERVICE are identical across the clustered brokers. Using identical service names will allow the broker to round-robin messages to each of the connected RPC server instances.

- For troubleshooting purposes, and if your site allows this, you can optionally use a different user ID for each RPC server.
- RPC servers are typically monitored using SMH as services of a broker. Optionally, for example for troubleshooting purposes, the RPC servers can be configured with a unique TCP port number for SMH.



Note: SMH port is not supported by the Natural RPC server.

- Establish the broker connection using the static `Broker name:port` definition.

■ CICS RPC Server

- CICS RPC Server instances can be configured through the *ERXMAIN Macro* that contains default settings and naming conventions. You can use the *RPC Online Maintenance Facility* to control and monitor these instances online.
- Establish an unused port through the CICS default stack to start the SMH Listener. See *SMH*.
- If you are using server-side mapping files, share the server-side mapping container across RPC server instances. A server-side mapping file is an EntireX Workbench file with extension `.svm`. See *Server Mapping Files for COBOL*. Use VSAM RLS or simple share options to keep a single image of the server-side mapping container across all or groups of RPC servers (for example CICS, IMS, Batch). See *Job Replacement Parameters* under *Simplified z/OS Installation Method* in the z/OS Installation documentation.

■ Batch RPC Server

- Make sure logging is distinguishable through each RPC Server instance for troubleshooting purposes (e.g. `JES SYSOUT`).

■ Natural RPC Server

- Maintain separate parameter files for each Natural RPC Server instance.

Verifying your Configuration

Here are some sample commands for verifying your cluster environment:

➤ To display the Netstat Dynamic VIPA status

- Enter command

```
D TCPIP,TCPIPEXB,N,VIPADYN
```

Status must be "ACTIVE".

```
EZZ2500I NETSTAT CS V1R13 TCPIPEXB 187
DYNAMIC VIPA:
  IP ADDRESS      ADDRESSMASK      STATUS      ORIGINATION      DISTSTAT
  10.20.74.103    255.255.255.0   ACTIVE      VIPADEFINE        DIST/DEST
  ACTTIME:       11/17/2011 09:29:13
```

➤ To display the Netstat Dynamic VIPA info

- Enter command

```
D TCPIP,TCPIPEXB,N,VIPADCFG
```

This shows multiple static definitions to one dynamic definition:

```
EZZ2500I NETSTAT CS V1R13 TCPIPEXB 190
DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
  IP ADDRESS      ADDRESSMASK      MOVEABLE      SRVMGR FLG
  -----
  10.20.74.103    255.255.255.0   IMMEDIATE     NO
  VIPA DISTRIBUTE:
  IP ADDRESS      PORT      XCF ADDRESS      SYSPT      TIMAFF      FLG
  -----
  10.20.74.103    18000    10.20.74.104     NO         NO
  10.20.74.103    18000    10.20.74.114     NO         NO
```

➤ To display Sysplex VIPA Dynamic configuration

- Enter command

```
D TCPIP,TCPIPEXB,SYS,VIPAD
```

This verifies that multiple LPARs have been defined (MVSNAMEs):

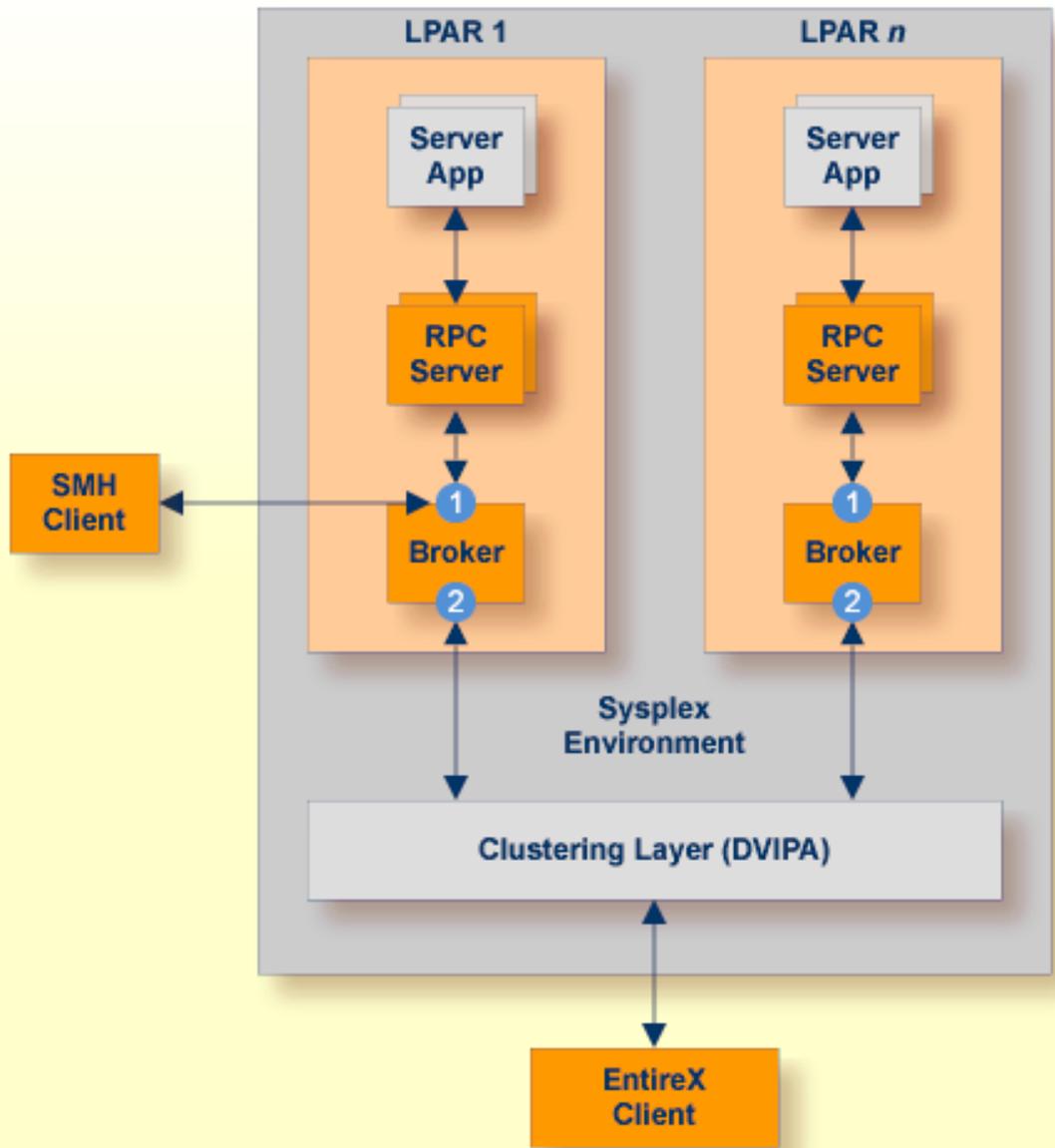
```
EZZ8260I SYSPLEX CS V1R13 166
VIPA DYNAMIC DISPLAY FROM TCPIPEXB AT AHST
IPADDR: 10.20.74.103 LINKNAME: VIPLOA144A67
ORIGIN: VIPADEFINE
TCPNAME MVSNAME STATUS RANK ADDRESS MASK NETWORK PREFIX
-----
TCPIPEXB AHST ACTIVE 255.255.255.0 10.20.74.0
TCPIPEXB BHST BACKUP 001
```

Managing Brokers and RPC Servers

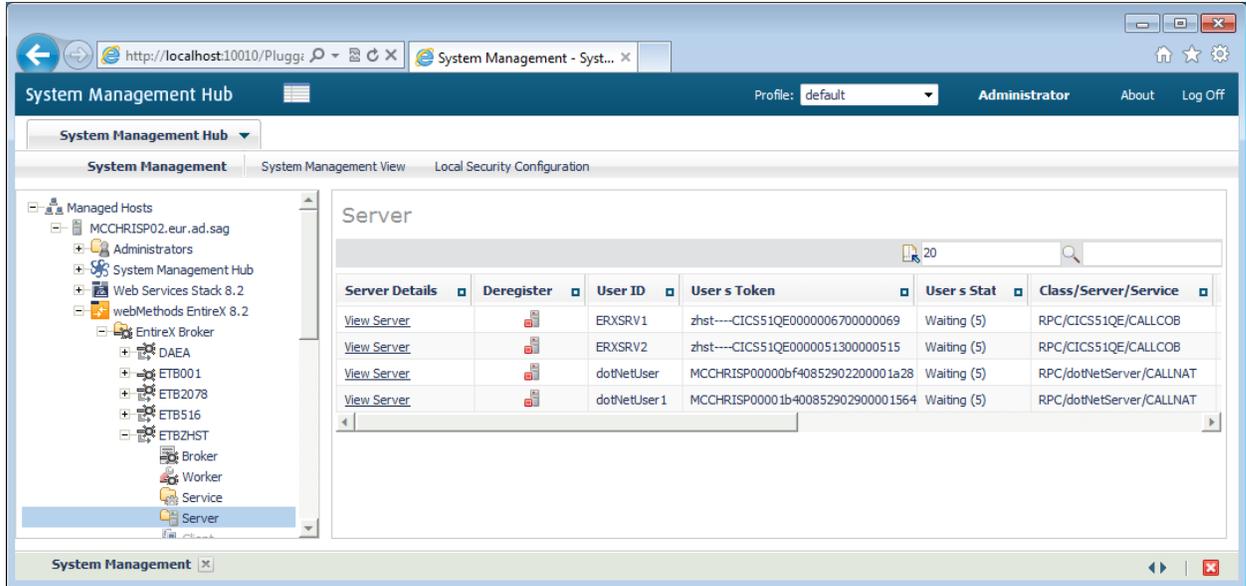
- [Using System Management Hub](#)
- [Lifecycle Management of Brokers](#)
- [Lifecycle Management of RPC Servers](#)

Using System Management Hub

Use the System Management Hub to monitor the status of the broker and RPC server instances using their respective `address:port` connections. Set up each connection with logical instance names.



The following screen shows two pairs of redundant RPC servers registered to the same broker from the Server view:



- Each broker requires a static TCP port for RPC server and management communications.
- This must be maintained separately from the broker's virtual IP address configuration.
- Onboard TCP-based RPC server connections *can* support HiperSockets requiring segregated static addresses (otherwise the address can be shared).

Lifecycle Management of Brokers

An important aspect of high availability is during planned maintenance events such as lifecycle management, applying software fixes, or modifying the number of runtime instances in the cluster. Using a virtual IP networking approach for broker clustering allows high availability to the overall working system while applying these tasks.

See *Starting and Stopping the Broker* under *Setting up Broker Instances* in the z/OS administration documentation.

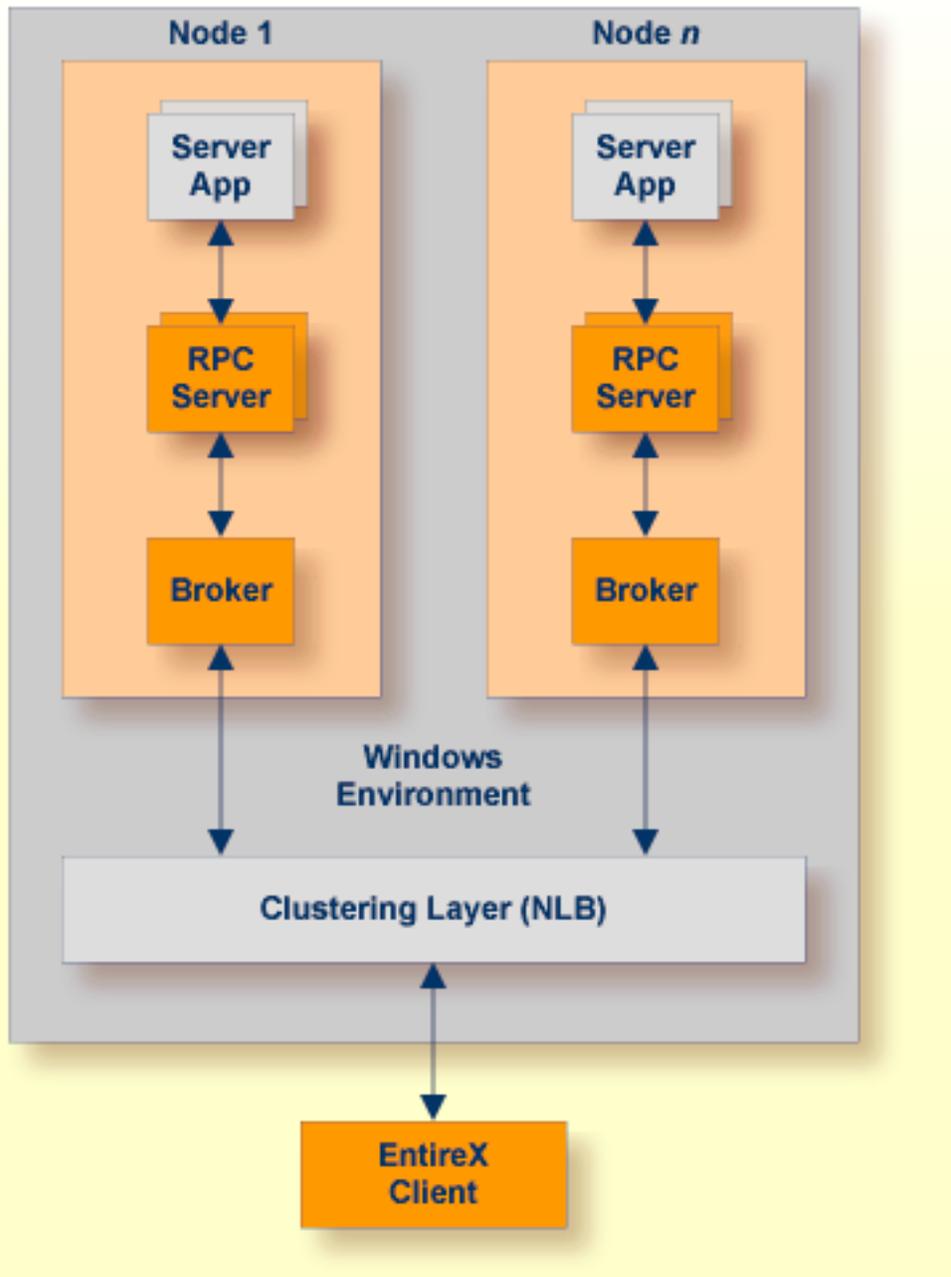
You can ping and stop a broker using the command-line utility *ETBCMD*.

Lifecycle Management of RPC Servers

Starting, pinging and stopping an RPC server is described in the EntireX documentation for CICS | Batch | IMS RPC servers.

See also *Operating a Natural RPC Environment* in the Natural documentation.

5 High Availability with NLB under Windows



The Network Load Balancing (NLB) feature in Windows Server (2008 R2 or 2012) enhances the availability and scalability of EntireX Broker and other mission-critical applications. NLB is intended for atomic applications that do not have long-running-in-memory or conversational states. Essentially, each client call to a stateless application is a separate transaction, so it is possible to distribute the requests among multiple servers to share the work load. One attractive feature of NLB is that all servers in a cluster monitor each other with a heartbeat signal, so there is no single point of failure.

In theory, a single broker instance running on Windows Server provides a limited level of server reliability and scalable performance. However, by combining the resources of two or more Brokers into a single virtual cluster, NLB can deliver the reliability and performance that mission-critical servers such as EntireX need.

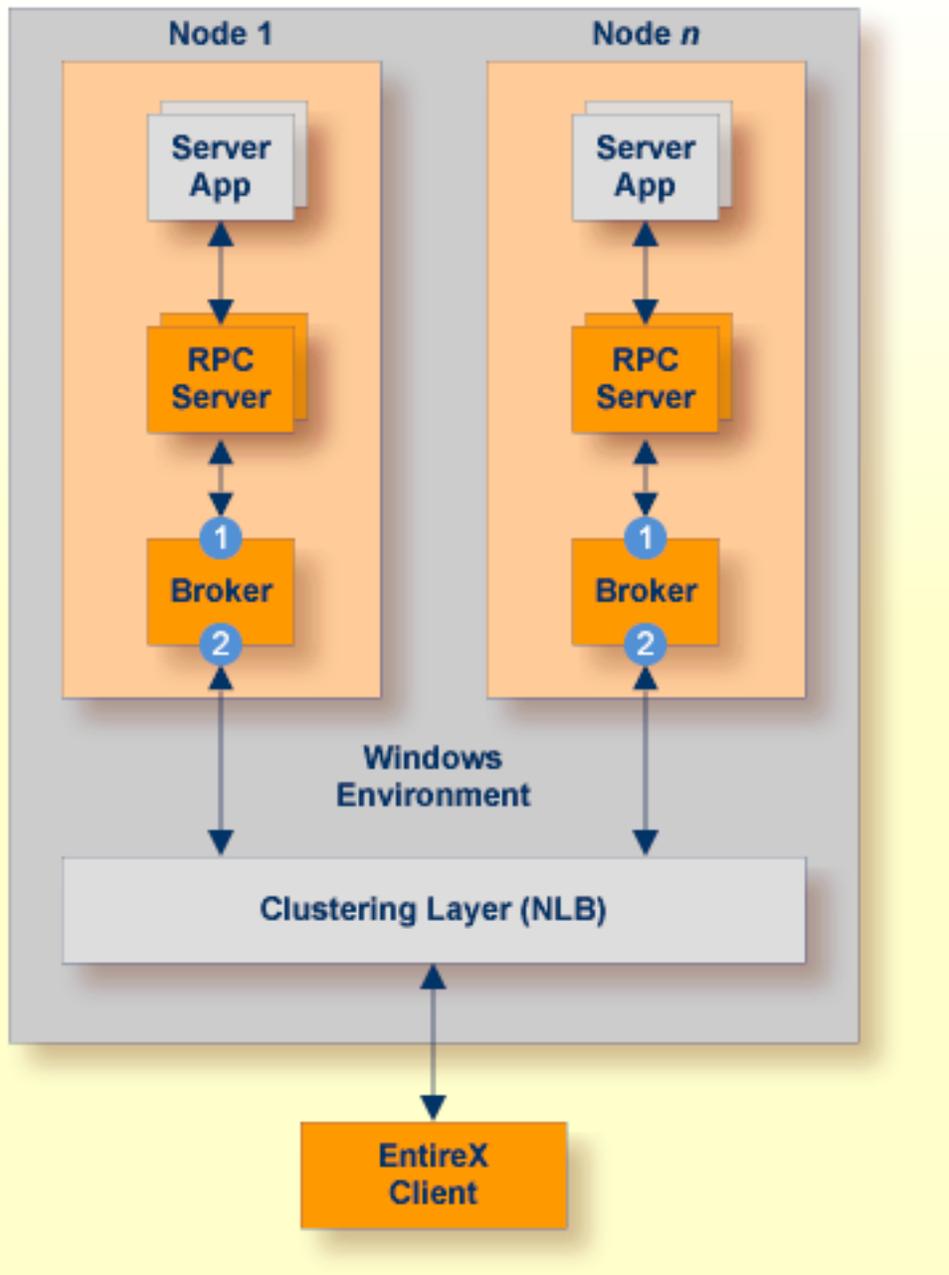
See [Setting up your EntireX Environment for NLB](#) for more details.

6 Setting up your EntireX Environment for NLB

- Setting Broker Attributes 26
- Broker Cluster Considerations 27
- Configuring Redundant RPC Servers 28
- Managing Brokers and RPC Servers 28
- Restrictions 31

Scenario: "I want to use Windows NLB for my high availability cluster."

Setting Broker Attributes



Segmenting dynamic workload from static server and management topology is critically important. Using broker TCP/IP-specific attributes, define two separate connection points:

- One for RPC server-to-broker and admin connections. ⁽¹⁾
- The second for client workload connections. ⁽²⁾

See *TCP/IP-specific Attributes* (DEFAULTS=TCP) under *Broker Attributes* in the platform-independent administration documentation. Sample attribute file settings:

	Sample Attribute File Settings	Note
(1)	PORT=1972	In this example, the HOST is not defined, so the default setting will be used (localhost).
(2)	HOST=10.20.74.103 (or DNS) PORT=1811	In this example, the HOST stack is the virtual IP address. The PORT will be shared by other brokers in the cluster.

Broker Cluster Considerations

Platform-independent Considerations

We recommend the following:

- Share configurations - you will want to consolidate as many configuration parameters as possible in the attribute setting. Keep separate yet similar attribute files.
- Isolate workload listeners from management listeners.
- Monitor Brokers through SMH.

Windows-specific Considerations

- The network load balancing service for all the machines should have the correct local time. Ensure the Windows Time Service is properly configured on all hosts to keep clocks synchronized. Unsynchronized times will cause a network login screen to pop up which doesn't accept valid login credentials.
- You have to manually add each load balancing server individually to the load balancing cluster after you've created a cluster host.
- To allow communication between servers in the same NLB cluster, each server requires the following registry entry: a DWORD key named "UnicastInterHostCommSupport" and set to 1, for each network interface card's GUID (HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WLBS\Parameters\Interface{GUID})
- NLB may conflict with some network routers, which are not able to resolve the IP address of the server and must be configured with a static ARP entry.

Configuring Redundant RPC Servers

In addition to broker redundancy, you also need to configure your RPC servers for redundant operations. We recommend the following best practices when setting up your RPC servers:

■ General Hints

- Make sure your definitions for `CLASS/SERVER/SERVICE` are identical across the clustered brokers. Using identical service names will allow the broker to round-robin messages to each of the connected RPC server instances.
- For troubleshooting purposes, and if your site allows this, you can optionally use a different user ID for each RPC server.
- RPC servers are typically monitored using SMH as services of a broker. Optionally, for example for troubleshooting purposes, the RPC servers can be configured with a unique TCP port number for SMH.



Note: SMH port is not supported by the Natural RPC server.

- Establish the broker connection using the static `Broker name:port` definition.

■ Natural RPC Server

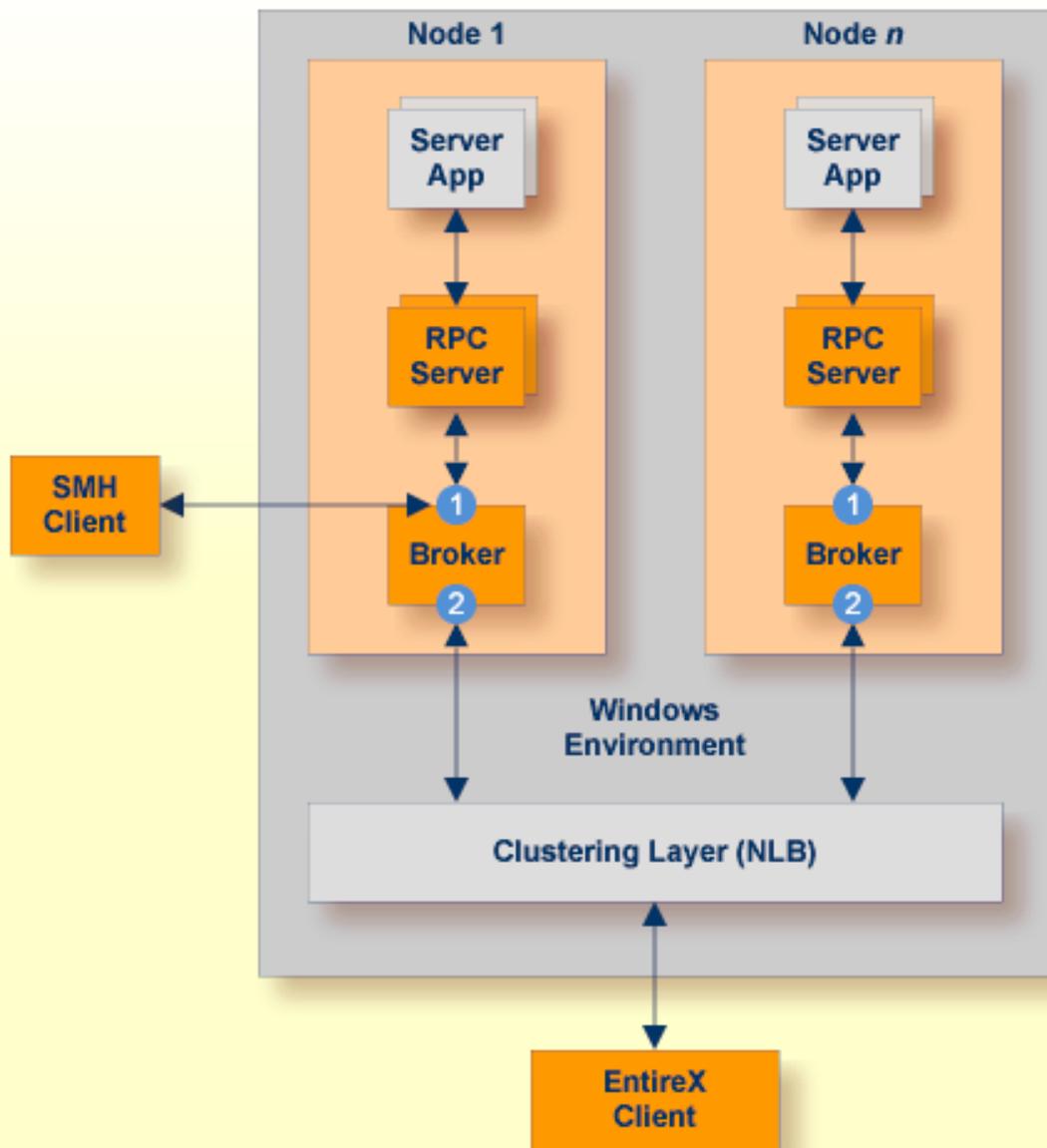
- Maintain separate parameter files for each Natural RPC Server instance.

Managing Brokers and RPC Servers

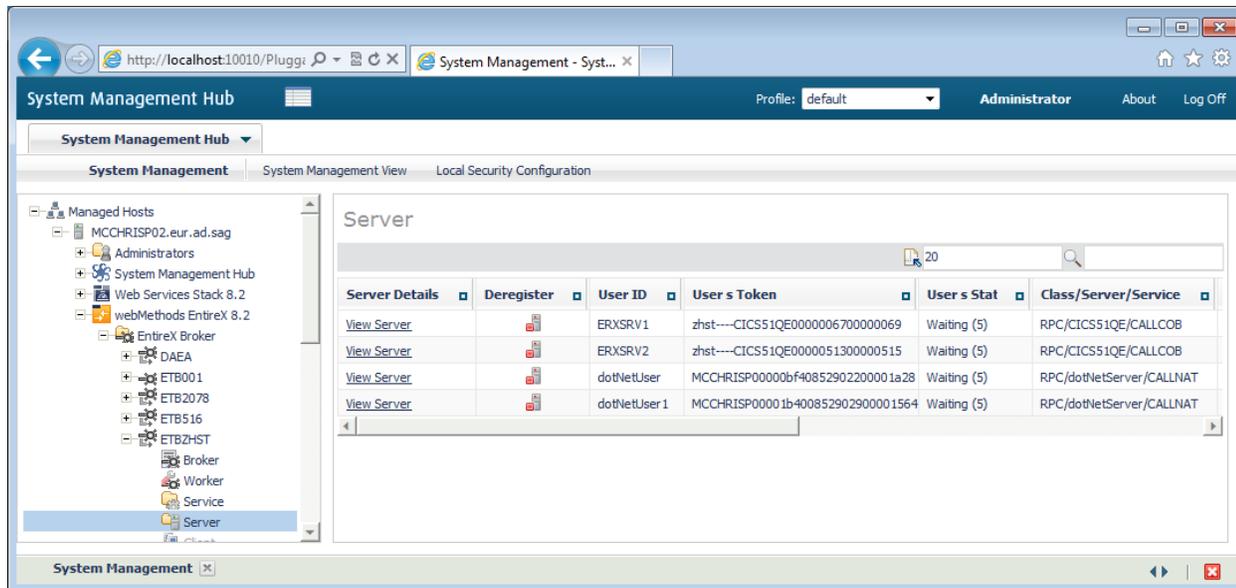
- [Using System Management Hub](#)
- [Lifecycle Management of Brokers](#)
- [Lifecycle Management of RPC Servers](#)

Using System Management Hub

Use the System Management Hub to monitor the status of the broker and RPC server instances using their respective `address:port` connections. Set up each connection with logical instance names.



The following screen shows two pairs of redundant RPC servers registered to the same broker from the Server view:



- Each broker requires a static TCP port for RPC server and management communications.
- This must be maintained separately from the broker's virtual IP address configuration.

Lifecycle Management of Brokers

An important aspect of high availability is during planned maintenance events such as lifecycle management, applying software fixes, or modifying the number of runtime instances in the cluster. Using a virtual IP networking approach for broker clustering allows high availability to the overall working system while applying these tasks.

Broker administrators, notably on UNIX and Windows systems, have the need to start, ping (for Broker alive check) and stop Broker as well as RPC servers from a system command-line, prompt or from within batch or shell scripts. To control and manage the life cycle of brokers, the following commands are available:

- `etbsrv BROKER START <broker-id>`
- `etbsrv BROKER PING <broker-id>`
- `etbsrv BROKER STOP <broker-id>`
- `etbsrv BROKER RESTART <broker-id>`

If only one user is to be permitted to execute commands, enter the command

```
etbsrv SECURITY ENABLE TRUSTED-USER=YES
```

The trusted user can then execute commands without any additional authentication. There can only be one trusted user. To change the trusted user, modify the property `install.user` in file `entirex.config`.

See *Administration Service Commands* in the EntireX Windows Administration documentation for more information.

Lifecycle Management of RPC Servers

➤ To start an RPC server

- See *Starting the RPC Server* in the documentation of the respective RPC server.

➤ To ping an RPC server

- Use the following Administration Service command:

```
etbsrv BROKER PINGRPC <broker-id> <class/server/service>
```

Return code 0 means the broker is running; any other value means the broker has stopped.

➤ To stop an RPC server

- See *Stopping the RPC Server* in the documentation of the respective RPC server.

You can also use the command-line utility `etbcmd`. Example:

```
etbcmd -b <broker-id> -d SERVICE -o IMMED -m <class/server/service>
```

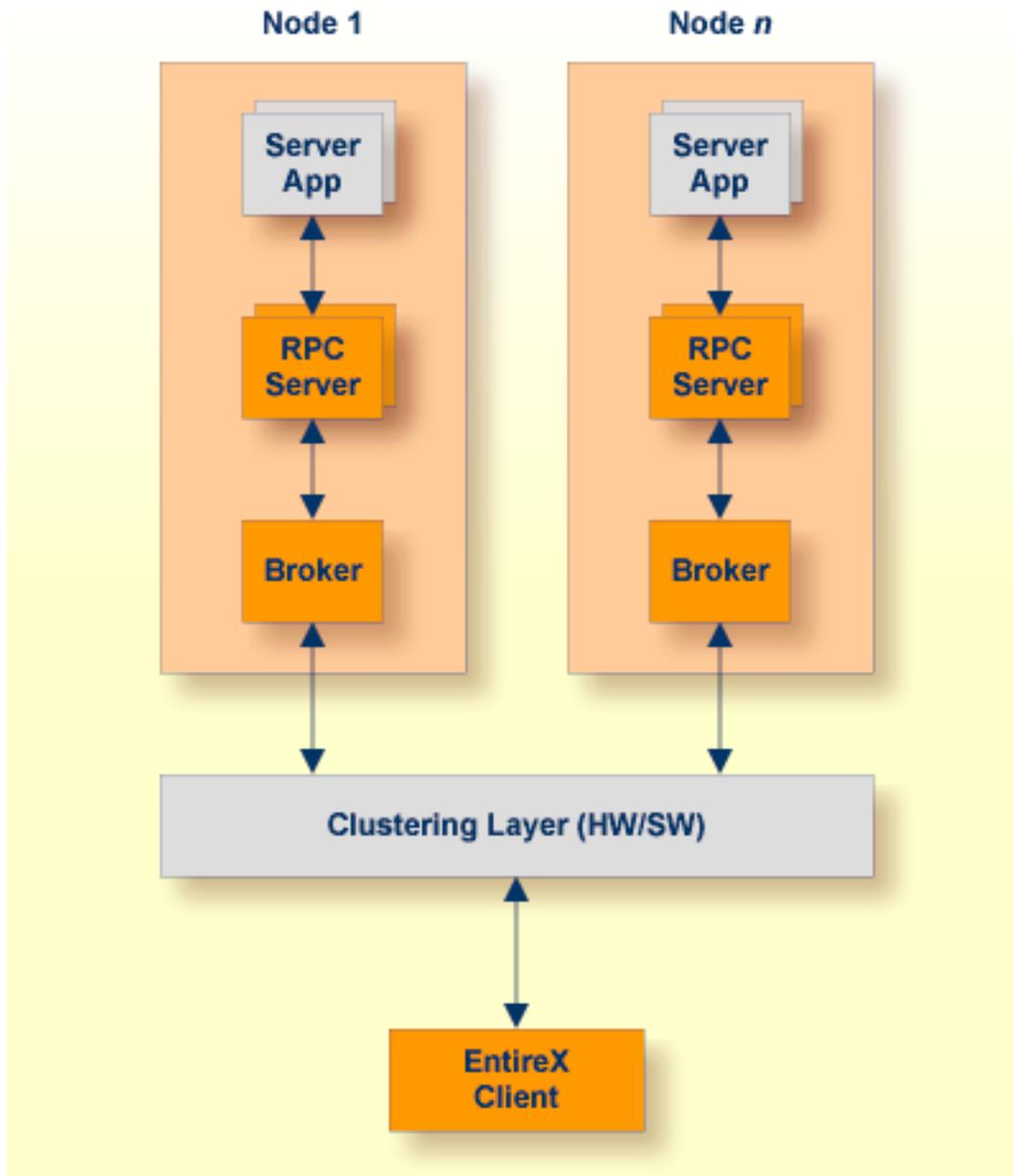
Restrictions

- All hosts in the NLB cluster must reside on the same subnet and the cluster's clients are able to access this subnet.
- When using NLB in multicast or unicast mode, routers need to accept proxy ARP responses (IP-to-network address mappings that are received with a different network source address in the Ethernet frame).
- Make sure the Internet control message protocol (ICMP) to the cluster is not blocked by a router or firewall.
- Cluster hosts and the virtual cluster IP need to have dedicated (static) IP addresses. This means you must request static IPs from your Network Services group.

- NLB clustering is a stateless failover environment that does not provide application or in-flight message recovery.
- Only TCP/IP is configured on the network interface that the NLB is configured for.

7 High Availability with Other Clustering Technologies

EntireX Brokers can also be clustered for the purpose of enhancing application availability and scalability with other IP-based load balancing solutions. From an EntireX Broker point of view, there is no fundamental difference between a HW or a SW clustering solution. It is up to the customer and their networking organization to reconcile the requirements of the EntireX project with what any particular technology can or cannot deliver. The following information represents the generic configuration of EntireX components using a network IP virtualization or clustering solution.

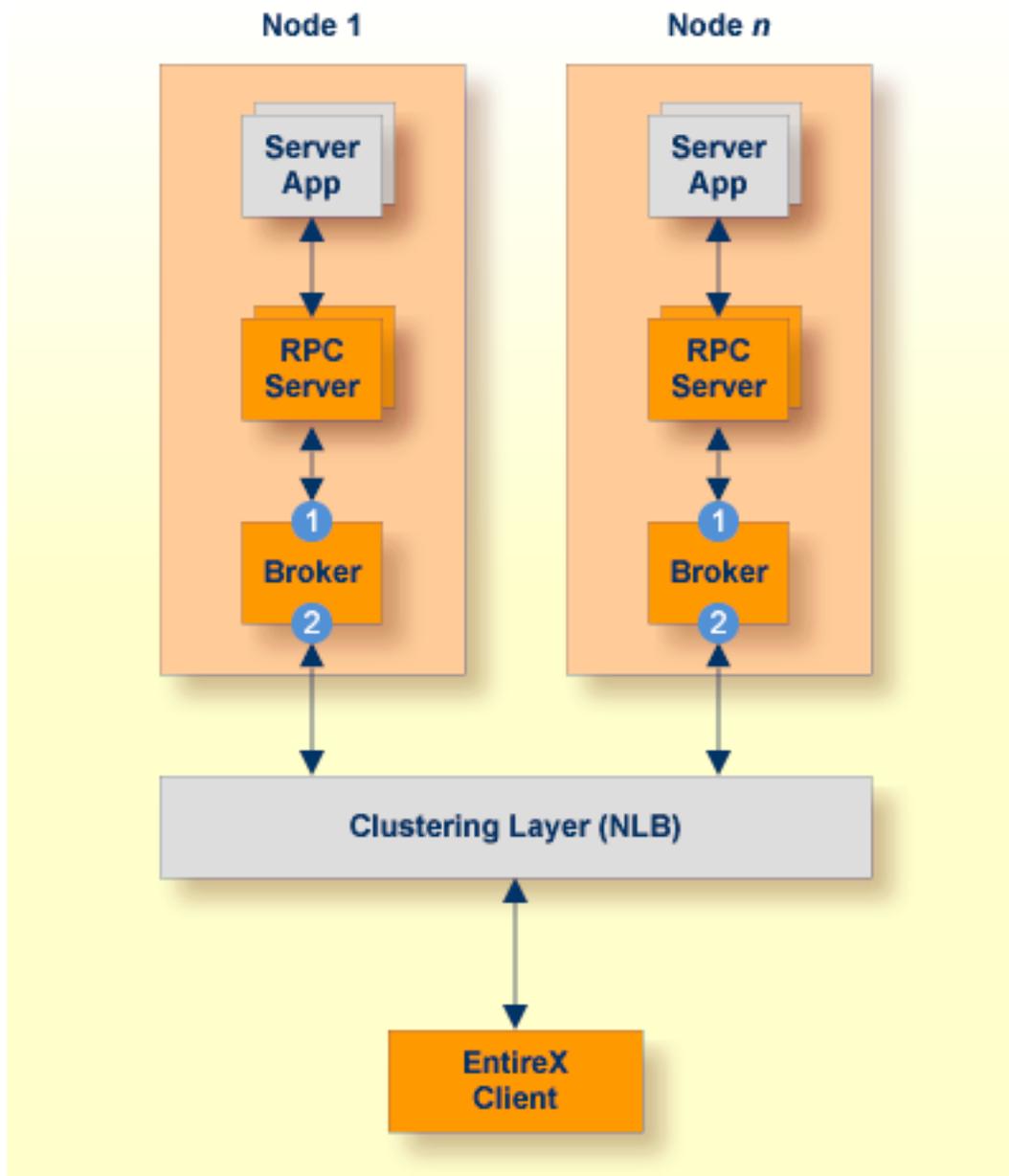


See also [Setting up Your EntireX Environment for High Availability with Other Clustering Technologies](#) for more details.

8 Setting up Your EntireX Environment for High Availability with Other Clustering Technologies

- Setting Broker Attributes 36
- Broker Cluster Considerations 37
- Configuring Redundant RPC Servers 37
- Managing Brokers and RPC Servers 38

Setting Broker Attributes



Segmenting dynamic workload from static server and management topology is critically important. Using broker TCP/IP-specific attributes, define two separate connection points:

- One for RPC server-to-broker and admin connections. ⁽¹⁾
- The second for client workload connections. ⁽²⁾

See *TCP/IP-specific Attributes* (DEFAULTS=TCP) under *Broker Attributes* in the platform-independent administration documentation. Sample attribute file settings:

	Sample Attribute File Settings	Note
(1)	PORT=1972	In this example, the HOST is not defined, so the default setting will be used (localhost).
(2)	HOST=10.20.74.103 (or DNS) PORT=1811	In this example, the HOST stack is the virtual IP address. The PORT will be shared by other brokers in the cluster.

Broker Cluster Considerations

We recommend the following:

- Share configurations - you will want to consolidate as many configuration parameters as possible in the attribute setting. Keep separate yet similar attribute files.
- Isolate workload listeners from management listeners.
- Monitor Brokers through SMH.

Configuring Redundant RPC Servers

In addition to broker redundancy, you also need to configure your RPC servers for redundant operations. We recommend the following best practices when setting up your RPC servers:

■ General Hints

- Make sure your definitions for CLASS/SERVER/SERVICE are identical across the clustered brokers. Using identical service names will allow the broker to round-robin messages to each of the connected RPC server instances.
- For troubleshooting purposes, and if your site allows this, you can optionally use a different user ID for each RPC server.
- RPC servers are typically monitored using SMH as services of a broker. Optionally, for example for troubleshooting purposes, the RPC servers can be configured with a unique TCP port number for SMH.



Note: SMH port is not supported by the Natural RPC server.

- Establish the broker connection using the static Broker name:port definition.

■ **Natural RPC Server**

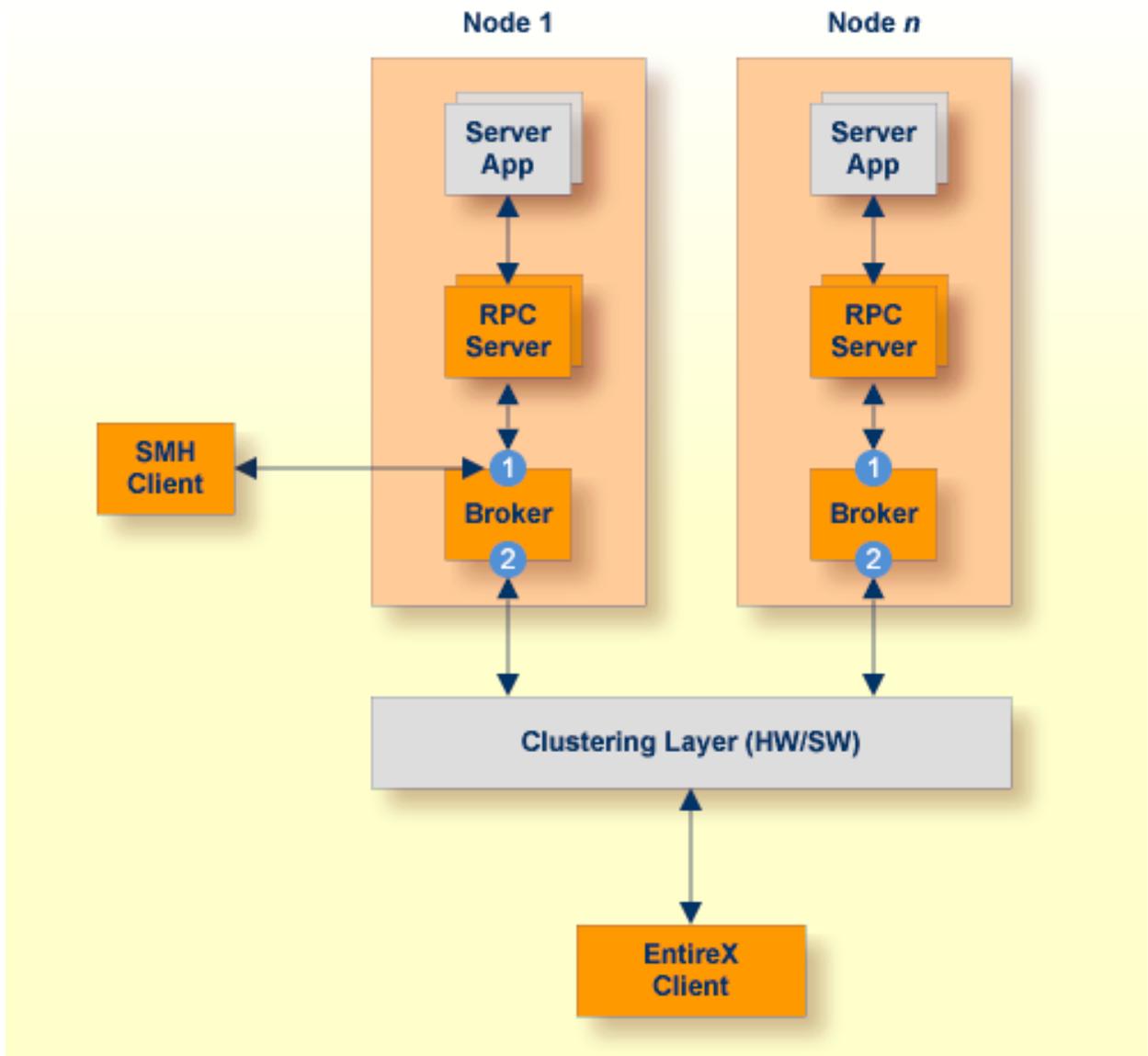
- Maintain separate parameter files for each Natural RPC Server instance.

Managing Brokers and RPC Servers

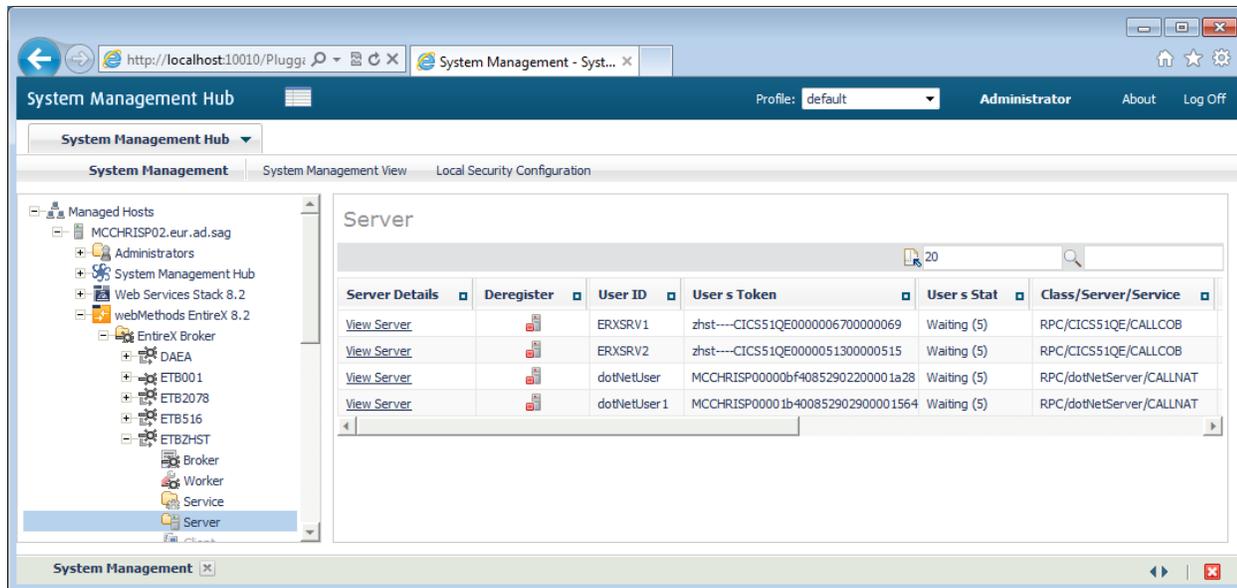
- [Using System Management Hub](#)
- [Lifecycle Management of Brokers](#)
- [Lifecycle Management of RPC Servers](#)

Using System Management Hub

Use the System Management Hub to monitor the status of the broker and RPC server instances using their respective `address:port` connections. Set up each connection with logical instance names.



The following screen shows two pairs of redundant RPC servers registered to the same broker from the Server view:



- Each broker requires a static TCP port for RPC server and management communications.
- This must be maintained separately from the broker's virtual IP address configuration.

Lifecycle Management of Brokers

An important aspect of high availability is during planned maintenance events such as lifecycle management, applying software fixes, or modifying the number of runtime instances in the cluster. Using a virtual IP networking approach for broker clustering allows high availability to the overall working system while applying these tasks.

Broker administrators, notably on UNIX and Windows systems, have the need to start, ping (for Broker alive check) and stop Broker as well as RPC servers from a system command-line, prompt or from within batch or shell scripts. To control and manage the life cycle of brokers, the following commands are available:

- `etbsrv BROKER START <broker-id>`
- `etbsrv BROKER PING <broker-id>`
- `etbsrv BROKER STOP <broker-id>`
- `etbsrv BROKER RESTART <broker-id>`

If only one user is to be permitted to execute commands, enter the command

```
etbsrv SECURITY ENABLE TRUSTED-USER=YES
```

The trusted user can then execute commands without any additional authentication. There can only be one trusted user. To change the trusted user, modify the property `install.user` in file `entirex.config`.

See *Administration Service Commands* in the UNIX|Windows administration section of the documentation for more information.

Lifecycle Management of RPC Servers

➤ To start an RPC server

- See *Starting the RPC Server* in the documentation of the respective RPC server.

➤ To ping an RPC server

- Use the following Administration Service command:

```
etbsrv BROKER PINGRPC <broker-id> <class/server/service>
```

Return code 0 means the broker is running; any other value means the broker has stopped.

➤ To stop an RPC server

- See *Stopping the RPC Server* in the documentation of the respective RPC server.

You can also use the command-line utility `etbcmd`. Example:

```
etbcmd -b <broker-id> -d SERVICE -o IMMED -m <class/server/service>
```

