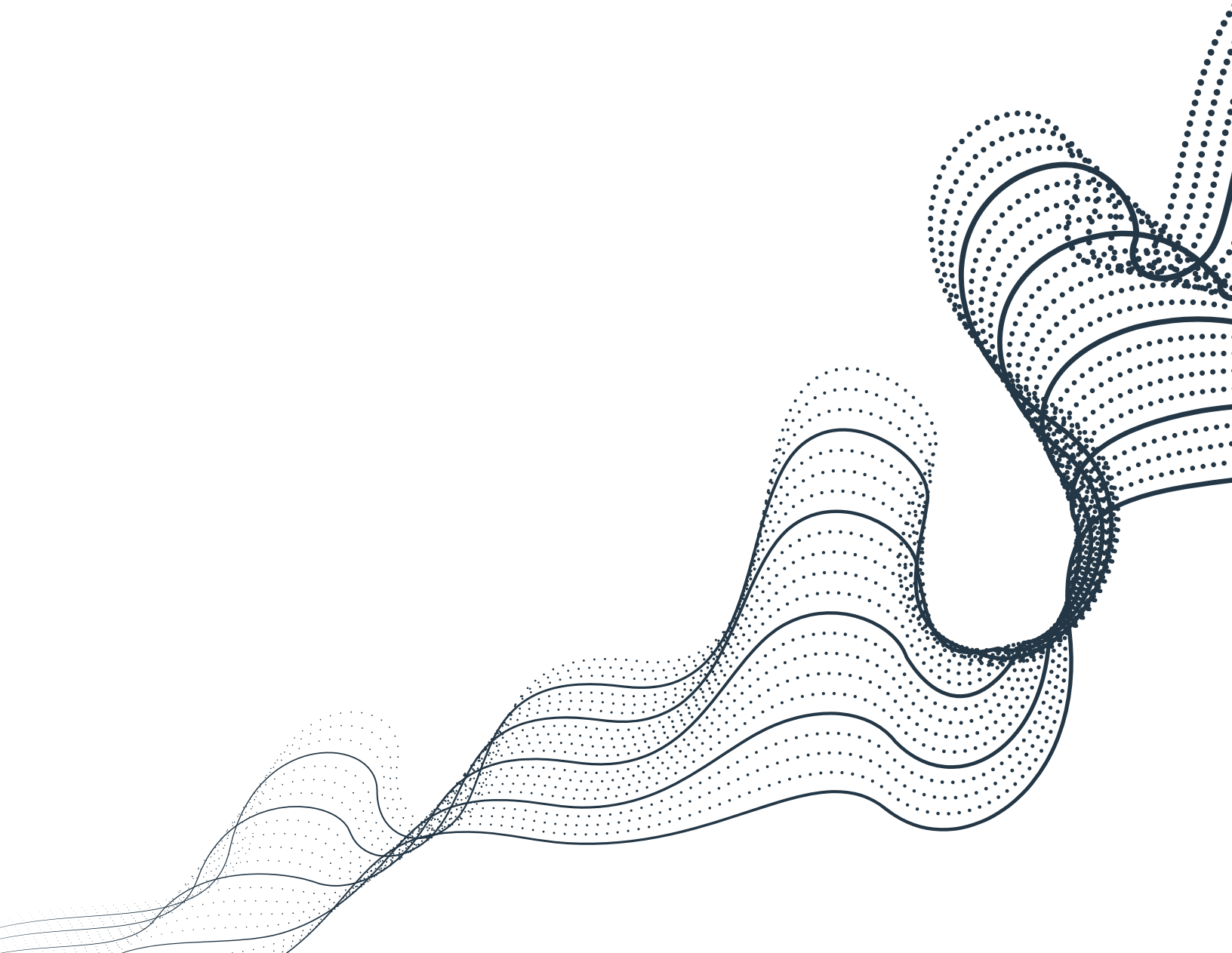
EMBOTICS

vCommander®

High Availability Solution Guide



Summary

Large enterprise organizations typically require production systems to have some level of failover or high availability setup to achieve uptime targets. Embotics® vCommander® is an application that can be run in an Active-Passive clustered configuration to meet this common enterprise requirement.

This document explains the process by which vCommander is made highly available by using a two-node cluster of application servers. To further protect the availability, install the application against a highly available Microsoft SQL database. There is no way to configure failover when vCommander has been installed with the default Postgres SQL database.

This guide uses as its example the Embotics HA configuration with a front-end load balancer, but the same configuration will work for any service-aware load balancing solution.

Contents

Requirements

Find below the requirements necessary to fulfill the complete configuration as described in this document.

- vCommander 5.7.9 or later
- Embotics vCommander REST API PowerShell libraries
- PowerShell v4 installed on the vCommander application server
- vCommander Node Quarantine script
- vSphere PowerCLI installed on the vCommander application server
- Load Balancer capable of monitoring service status
- Highly available SQL configuration of your choice

You must have the Embotics PowerShell REST client installed to initialize the first node.

Access to the vCommander REST API is also required on both nodes in the cluster to test to determine if logins are successful.

The vCommander Node Quarantine script is used to shut down a failed node in the cluster. This is done to ensure that a non-responsive node cannot come back online unexpectedly, and is highly recommended over manual shutdown. The script is designed to shut down a vCommander node running on vCenter using PowerCLI, and must be edited with your environment's details.

Your load balancer must be capable of monitoring the service status. This document provides examples for Kemp and F5 load balancers. If you are using a load balancer from a different vendor, please consult its documentation on how best to monitor the vCommander service status in a manner like the examples provided.

The database must also be highly available. Configuration of the database as highly available is the responsibility of Embotics' customers using their preferred solution.

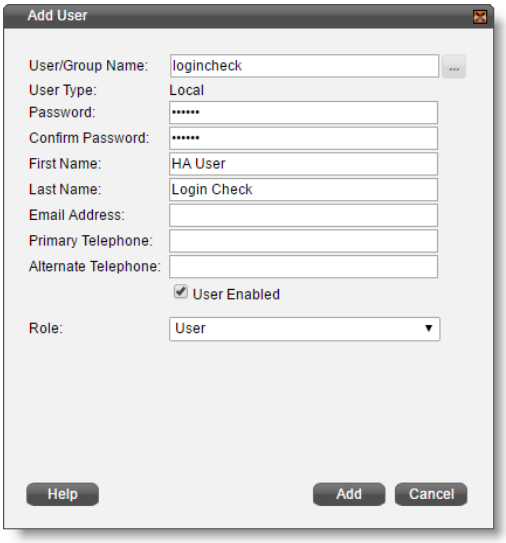
Before You Begin

As when making any substantial change to a production system, Embotics strongly recommends taking measures to ensure a successful roll-back is possible in case unexpected failures occur. These include making sure there are current backups of the vCommander SQL database and snapshots of the vCommander application server.

Additionally, you may want to implement and test the configuration in a staging environment to make sure it operates as expected prior to putting it into production. Embotics provides customers with staging or lab licenses upon request. Contact your Customer Advocate for more information about these limited, supplementary licenses.

Creating an HA User

You must create a specific account to handle the high available configuration monitoring activity. As a best practice, name this account so that it is easily recognizable. Doing so facilitates tracking actions undertaken by this solution by parsing vCommander events for the account.



- Simply using a local vCommander account with the User role will be sufficient. However, you can use a directory services (AD/LDAP) if necessary.
- Provide a name that will be recognizable in the events and logs as the HA configuration user.

Preparing the Node Quarantine Script

You should have already downloaded and extracted the script to your vCommander nodes as detailed in the Requirements section. Embotics typically recommends storing all scripts called by vCommander in a specific location, using sub-folders to identify the functions of scripts. Later on in this document we describe how to configure file replication to ensure both nodes have access to the scripts used in your workflows.

However, the Node Quarantine script used for high availability must not be replicated. Each node will have its own copy of the script, and must be edited to name the other node. Additionally, edits are required to identify the vCenter on which the vCommander nodes are running, and the location of your encrypted credentials file.

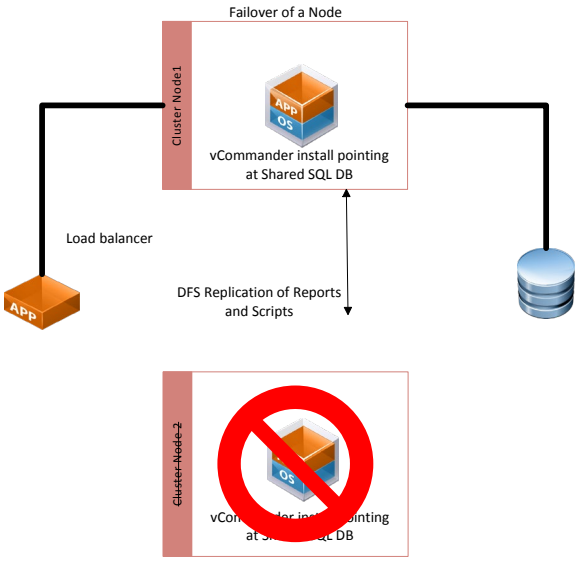
```
5   # Edit these lines to specify the vSphere host, the location of the credential file,
6   # and the name of the vm that should be quarantined
7
8   $VIServer = "your.vcenter.address.com"
9   $CredFile = "C:\scripts\cred.cred"
10  $VMName = "your.vcommander.name"  # VMname as it appears in vCenter
```

Setting	Description
$VIServer	The hostname or IP address of the vCenter server on which the other node's vCommander application server is running.
$CredFile	The credentials file which handles access to your vCommander. For more details, refer to this knowledgebase article.
$VMName	The name of the VM on which the other node's vCommander application server is running.

Clustering vCommander™

The figure below demonstrates how vCommander is installed on two application servers behind a load balancer. Each installation connects to the same highly available Microsoft SQL Server, but they cannot be connected simultaneously. The passive node activates when one or more failure indicators are observed. These failure indicators are the lack of a recent heartbeat record written to the database by the active node, an inability of the passive node to login to the active node via the API, or a combination of both.

Microsoft Distributed File System (DFS) replicates key directories between the nodes in real time, to ensure the passive node is fully synchronized with the active node. These directories contain binaries such as scripts, reports and log files.



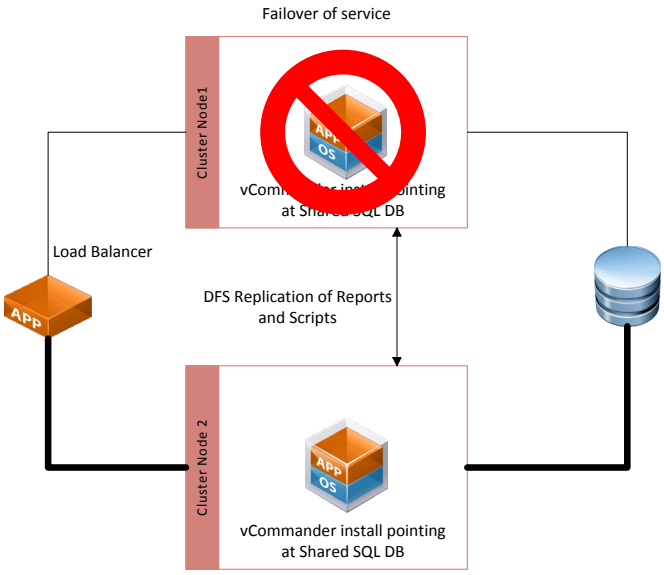
When the service on the active node fails, the failover follows this process:

1. The passive node determines that the active node is no longer functioning via its monitoring.

2. The passive node initiates a quarantine of the active node.

3. The passive node becomes the active node. It remains the active node even when the other node comes back online.

4. The newly active node executes a script to shut down the failed node.



There will always be a small amount of downtime as the passive node becomes active. When the failover is planned, Embotics recommends using the Service Portal Message of the Day to notify users of an upcoming outage window.

Installation Process

Before beginning, create two databases on the SQL Cluster. One database will be the production database where your actual data is stored and against which the active node operates, and the other will be a dummy database that is used to install or upgrade the application nodes, after which the connection strings will be reset to the production database. If you already have a production vCommander installed, the instructions remain the same. Your active system is Node 1.

For more information on the network and database requirements, please refer to the Embotics vCommander Installation Guide.

The steps below reference configuration points that are required for installation in a cluster, but there are other settings you may wish to configure to your own preferences. Please see HA.Properties Configuration File Reference for more details.

Note that these instructions include the use of the vCommander Control Panel. Once database information is entered in a vCommander node's HA.properties file, this is how the database connection is established. Modifying the database connection using the vCommander Control Panel will have no effect.

1. Install vCommander on **Node 1**, against the database you intend to use for production.
   Ensure vCommander has been activated with a license first, before proceeding further.

2. Create a read-only SQL user on the vCommander production database. You cannot use a domain account.

3. Login to vCommander on **Node1** and create a local user account with the Superuser role. This is the account that the passive node will use to perform login checks.

4. Install vCommander on **Node 2**, against the dummy database you will not use in production. The path to the installation directory, service account and installation package must be identical to that used for the installation on **Node 1**.

5. Verify that you can login to both installations without issue.

6. On **both Nodes**, stop the vCommander Windows service.

7. On **Node 1**, open the vCommander **HA.properties** file, located at:
   \Program Files\Embotics\vCommander\tomcat\common\classes

   i. Set the following to configure **Node 1** for HA:
      - `embotics.cluster.enabled=true`

   ii. Set your read-only SQL credentials:
      - `hibernate.connection.username=your SQL user account`
      - `hibernate.connection.password=your SQL user password`

   iii. Set the active database SQL user credentials:
      - `embotics.cluster.dbuser=your SQL user account`

- `embotics.cluster.dbpasswd=your SQL user password`

iv. Set the account to be used for login health checks that you created in the section Creating an HA User:
- `embotics.cluster.rest.user=your vCommander user account`
- `embotics.cluster.rest.password=your vCommander user password`

v. Set the IP address or hostname of the other node in the cluster, including the port number if other than 443:
- `embotics.cluster.rest.url=xxx.xxx.xxx.xxx:xxxx`

vi. Set the unique ID of the current node, used for logging:
- `embotics.cluster.service.id=Node_ID1`

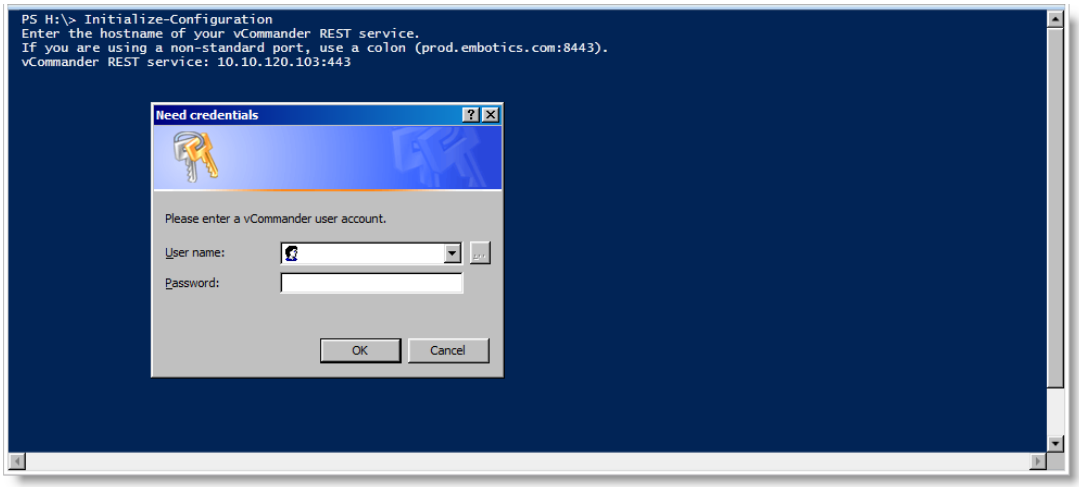
8. On **Node 2**, modify the same **HA.properties** file. All the values will be the same as what you set for **Node 1** except the address of the other node in the cluster (`embotics.cluster.rest.url`) and the unique ID for the current node (`embotics.cluster.service.id`).

9. Start the vCommander service on **Node 1** and wait until you the vCommander login page appears before proceeding further.

10. Set **Node 1** as active by invoking the following REST API call:

    `Initialize-Configuration`

    You will be prompted for the URL, credentials for the HA user account, and whether to ignore certificate errors (if you are using a self-signed certificate).



11. Once you have entered this information, issue the command:

    `Connect-client`

    and then:

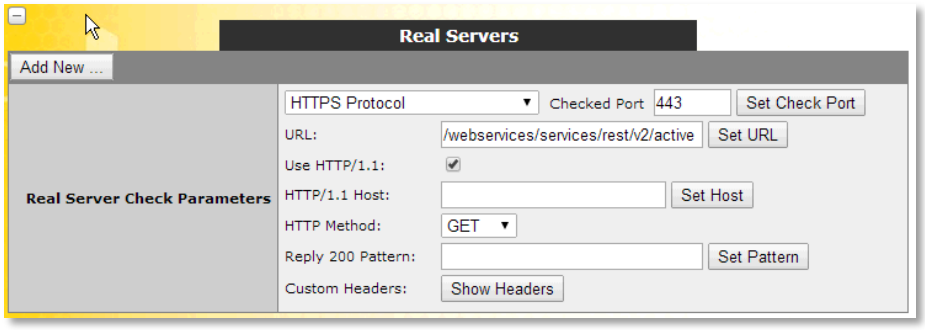
```
Invoke-Activate -quarantine $false
```

12. Launch the vCommander control panel on **Node 2**. Switch to the **DB Management** tab and change the connection details to match those used by **Node 1** to connect to the production database. Test the connection and start the vCommander service on **Node 2**.

13. vCommander starts in warm standby mode. You will be able to log in to confirm the system is running, but not make any changes, as read-only DB credentials are used for the passive node's connection.

14. To monitor your load balancer service, there are a few things that you can use, depending on the vendor. The URL to verify which node is active is **/webservices/services/rest/v2/active**.

    For some load balancers, such as Kemp, this is all you will need to specify. The load balancer receives the 200 (OK) status from the header for the active node.

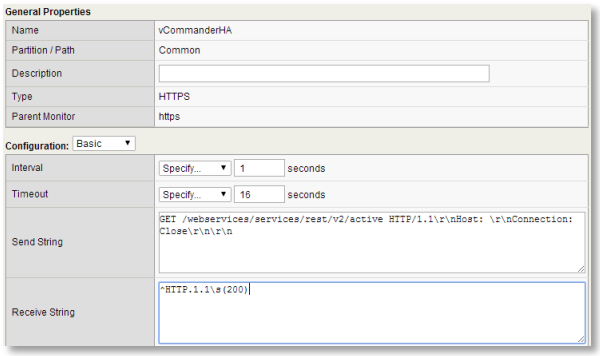


    Other load balancers such as F5 require a slightly more granular configuration. In the Embotics test lab, the F5 required the send string:

```
GET /webservices/services/rest/v2/active HTTP/1.1\r\nHost:
\r\nConnection: Close\r\n\r\n
```

    and a receive string:

```
^HTTP.1.1\s(200)
```



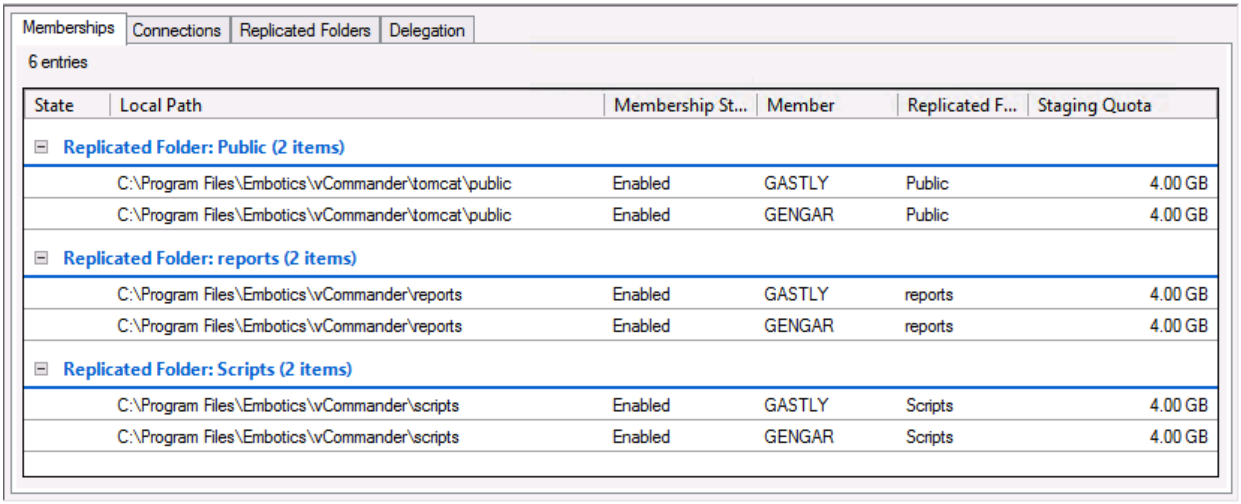
    If your load balancer is not capable of the monitoring methods above, but can search

for content, you can monitor the active node for the string `cluster_node_active` on the **/webservices/services/rest/v2/active** page.

Once the application has initialized and you have confirmed normal operations, best practices dictate that you perform testing of the failover capability to ensure service startup on the passive node.

File System Replication

Configure the folders listed below for DFS replication.



- **\Program Files\Embotics\vCommander\reports**
  - This folder is the location where custom reports created by Embotics are stored. These reports are dropped-in as JAR files to be automatically loaded into vCommander on startup.
- **\Program Files\Embotics\vCommander\scripts**
  - This folder is the location where workflow scripts are stored.
- **\Program Files\Embotics\vCommander\tomcat\public**
  - This folder is the location where custom themes and integrations are stored.

In addition, you can also configure the following:

- **\Program Files\Embotics\vCommander\tomcat\reports\data**
  - This folder is the location where reports generated from vCommander are stored.
- **\Program Files\Embotics\vCommander\tomcat\wfplugins**
  - This folder is the location where plug-in workflow steps are stored, and is needed only if you are using plug-in workflow steps.

You may have your workflow scripts stored in another location. If this is the case, make sure to include that location in addition to those listed above.

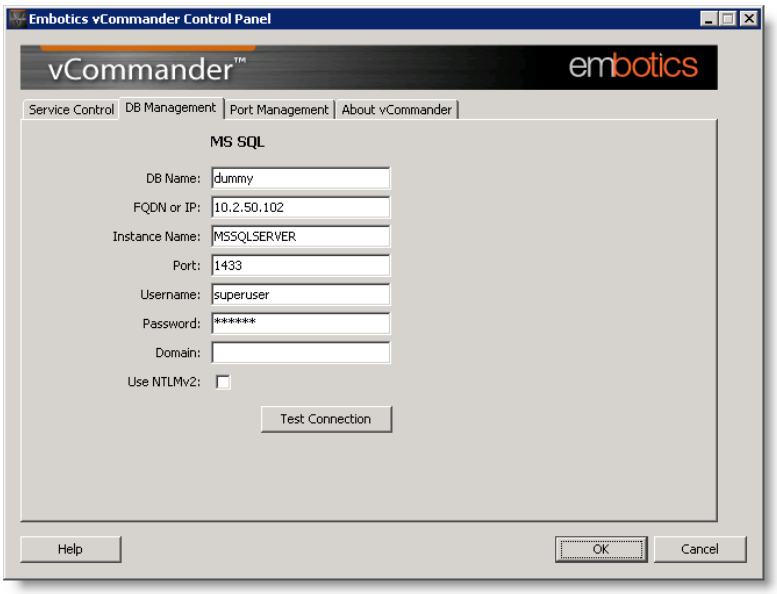
Note: If you are using plug-in workflow steps, you must install the plug-in workflow steps on both vCommander systems. To learn how to download and install workflow plug-in steps, Adding plug-in workflow steps.

Upgrade Process

The vCommander installer does not expect a clustered configuration, so the upgrade process involves making both the passive node and the active nodes believe that they are standalone applications.

Part 1: Upgrade the passive node

1. On the passive node, stop the vCommander service and the vCommander Identity service.
2. On the passive node, back up the following file to a location outside of the vCommander installation path (for example, your Desktop):
   \Program Files\Embotics\vCommander\tomcat\common\classes\HA.properties
3. On the passive node, edit the **\Program Files\Embotics\vCommander\tomcat\common\classes\HA.properties** file as follows:
   a. Set the following property:
      ```
      embotics.cluster.enabled=false
      ```
   b. Comment out the DB credential properties that start with **hibernate.connection.***
      ```
      # Read-only database credentials used by the passive node
      #hibernate.connection.username=none
      #hibernate.connection.password=none
      ```
4. On the passive node, from the vCommander program group, launch the Embotics vCommander Control Panel. Switch to the **DB Management** tab.



Enter the **DB Name** for your dummy database.

Click **Test Connection**. Confirm that the test is successful.

Click **OK** to apply, and click **OK** again when notified that the vCommander service will restart.

5. If the vCommander service does not start automatically, start the vCommander service on the passive node.
6. Once the service has restarted, confirm that vCommander is not connected to your production database by logging in.
7. On the passive node, upgrade your vCommander installation by right-clicking the executable and running as Administrator.
8. Once the upgrade is complete, log in to vCommander with the superuser account to ensure that upgrade was successful.
9. Stop the vCommander service and vCommander Identity service on the passive node.
10. On the passive node, restore the HA.properties file from the backup location to: **\Program Files\Embotics\vCommander\tomcat\common\classes\HA.properties**

Note: Do not start the vCommander services on the passive node at this time.

Part 2: Upgrade the active node

1. On the active node, stop the vCommander service and the vCommander Identity service.
2. On the active node, back up the following file to a location outside of the vCommander installation path (for example, your Desktop): **\Program Files\Embotics\vCommander\tomcat\common\classes\HA.properties**
3. On the active node, edit the file **\Program Files\Embotics\vCommander\tomcat\common\classes\HA.properties** as follows:
    a. Set the following property:

    ```
    embotics.cluster.enabled=false
    ```

    b. Comment out the DB credential properties that start with **hibernate.connection.***

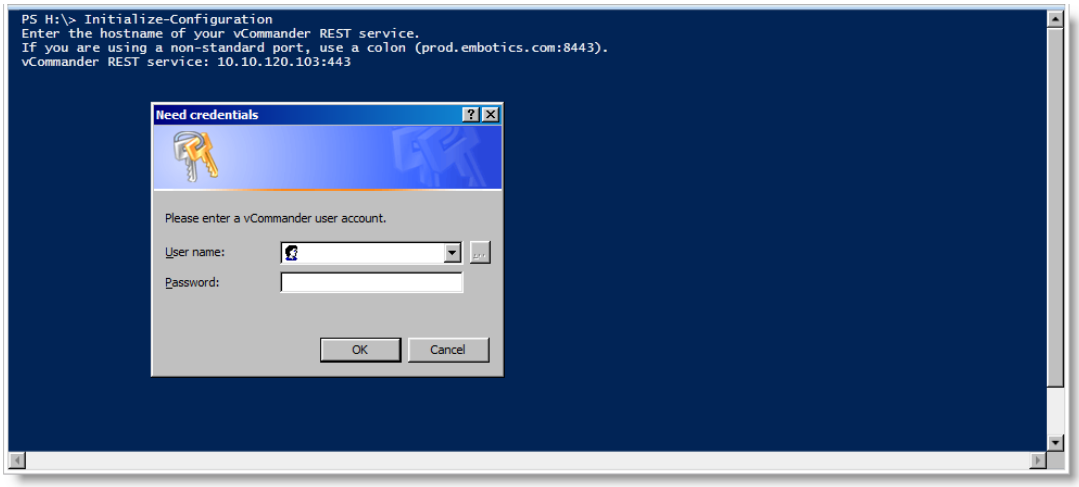
    ```
    # Read-only database credentials used by the passive node
    #hibernate.connection.username=none
    #hibernate.connection.password=none
    ```

4. Upgrade your vCommander installation on the active node by right-clicking the executable and running as Administrator.
5. Once the upgrade is complete, log in to vCommander with the superuser account to ensure that the upgrade was successful.

6. Stop the vCommander service and vCommander Identity service on the active node.
7. On the active node, restore the HA.properties file from the backup location to: **\Program Files\Embotics\vCommander\tomcat\common\classes\HA.properties**
8. Start the vCommander service on the active node.
9. Confirm that this node is the active node by accessing the following address: https://<activenode>/webservices/services/rest/v2/active
10. The message cluster_node_active indicates success. If the node is not active, activate it using the vCommander REST API PowerShell client, using the following calls:

```
Import-Module VCommanderRESTClient
Initialize-Configuration
```

You will be prompted for the URL, the credentials for the HA user account, and whether to ignore certificate errors (if you are using a self-signed certificate).



Once you have entered this information, issue the command:

```
Connect-client
```

and then:

```
Invoke-Activate -quarantine $false
```

Again, use the following address to confirm that the node is active:

https://<activenode>/webservices/services/rest/v2/active

Part 3: Start vCommander on the passive node

1. On the passive node, from the vCommander program group, launch the Embotics vCommander Control Panel.
   a. Switch to the **DB Management** tab.
   b. Enter the details for your production database.
   c. Click **Test Connection**. Confirm that the test is successful.
   d. Click **OK** to apply, and click **OK** again when notified that the vCommander service will restart.
2. If the vCommander service does not start automatically, start the vCommander service on the passive node.
3. Confirm that both the vCommander service and the vCommander Identity service have started on the passive node.
4. Confirm that the passive node has started, but is not active, by accessing the following address:

   https://<passivenode>/webservices/services/rest/v2/active

5. The state should be returned as "offline".

Note: After the upgrade is completed, vCommander performance may be affected for about an hour.

Command Reference

The table below provides the commands you may issue using the vCommander PowerShell client to control the HA configuration.

Action	Command	Notes
Transition vCommander from running in standby to fully operational.	Initialize-Configuration	You will be prompted to provide URL and credentials for the HA user.
Check if the vCommander service is in the ready state.	$s = Get-SystemState $s.SystemStatus.serviceState	Result will be either ACTIVE or STANDBY to indicate the current node's state.
Activate the service.	Invoke-Activate -quarantine $false	-

HA.Properties Configuration File Reference

Property	Description	Values
embotics.cluster.enabled	Indicates whether vCommander is deployed as standalone or part of an application cluster.	Initially set to **False.** **True** / **False**
hibernate.connection.username hibernate.connection.password	Read-only database credentials used by the passive node.	Initially set to **None**. Username and password pair providing read-only access to the vCommander database. Must be a SQL user.
embotics.cluster.dbuser embotics.cluster.dbpasswd	Database credentials used by the active node.	Initially set to **None**. Username and password pair providing full access to the vCommander database. Must be a SQL user.
embotics.cluster.rest.user embotics.cluster.rest.password	vCommander credentials used by the passive node to check login capability on the active node.	Initially set to **None**. Username and password pair for a vCommander user.
embotics.cluster.rest.url	The address of the other node in the cluster, including port number if not the default 443.	Initially set to **localhost**. IP address or hostname for the other vCommander node. Examples: ***vcom.foo.com:7443*** ***10.10.25.25***
embotics.cluster.service.id	A unique text string that differentiates nodes in the cluster from one another. Used to populate events in vCommander for audit purposes.	Initially set to **node_id_1**. Any value can be used, providing it will be provide the information you need to identify the node when auditing logs or events at a later date.

Property	Description	Values
embotics.cluster.startupDelay	The time in milliseconds that vCommander will wait before performing health checks. In the event both nodes start simultaneously, this setting prevents the passive node from taking the active node offline prematurely.	Initially set to **60000**. Any numeric value can be used, but this value should always be greater than the time it takes for your vCommander nodes to initialize, and greater than the other time configurations in the file.
embotics.cluster.healthCheckInterval	The frequency in milliseconds with which the passive node will perform health checks against the active node, following the startup delay.	Initially set to **30000.** Any numeric value can be used.
embotics.cluster.errorRetryInterval	The frequency in milliseconds with which the passive node will perform health checks against the active node, following an error.	Initially set to **5000**. Any numeric value can be used.
embotics.cluster.heartBeatErrors	The number of heartbeat errors that must be observed before the passive node will activate.	Initially set to **3**. Any numeric value can be used.
embotics.cluster.loginErrors	The number of login errors that must be observed before the passive node will activate.	Initially set to **10**. Any numeric value can be used.
embotics.cluster.twoFactorErrors	The number of heartbeat errors and login errors that must be observed in the same health check before the passive node will activate.	Initially set to **2**. Any numeric value can be used.
embotics.cluster.heartBeatFrequency	The frequency in milliseconds with which this vCommander node will generate heartbeats.	Initially set to **30000.** Any numeric value can be used.
embotics.cluster.quarantine.script	The path to the external script called to quarantine the node being deactivated.	Initially set to **c:\\scripts\\ShutdownVM_VC.ps1** Any valid UNC path or file system path to the file. Any backslashes used in the path must be doubled.
embotics.cluster.quarantine.continueOnFail	Indicates whether or not the passive node will continue to activate if the quarantine of the active node failed.	Initially set to **False**. **True** / **False**