Jive Interactive Intranet

9.x Administrator Guide

Platform Administration

Notices

For details, see the following topics:

Notices

This document is proprietary and confidential to Aurea and is available only under a valid non-disclosure agreement. No part of this document may be disclosed in any manner to a third party without the prior written consent of Aurea. The information in these materials is for informational purposes only and Aurea assumes no responsibility for any errors that may appear therein. Aurea reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of Aurea to notify any person of such revisions or changes.

You are hereby placed on notice that the software, its related technology and services may be covered by one or more United States (“US”) and non-US patents. A listing that associates patented and patent-pending products included in the software, software updates, their related technology and services with one or more patent numbers is available for you and the general public’s access at https://markings.ip-dynamics.ai/esw/ (the “Patent Notice”) without charge. The association of products-to-patent numbers at the Patent Notice may not be an exclusive listing of associations, and other unlisted patents or pending patents may also be associated with the products. Likewise, the patents or pending patents may also be associated with unlisted products. You agree to regularly review the products-to-patent number(s) association at the Patent Notice to check for updates.

Aurea and Aurea Software are registered trademarks of Aurea Software, Inc. in the United States and/or other countries. Additional Aurea trademarks, including registered trademarks, are available at: https://www.aurea.com/legal/trademarks/. Jive is a registered trademark of Jive Software, Inc. in the United States and/or other countries. Additional Jive trademarks, including registered trademarks, are available at: https://www.jivesoftware.com/legal/.

Third-party acknowledgments

The following third-party trademarks may appear in one or more Jive guides:

- Amazon is a registered trademark of Amazon Technologies, Inc.
- Apache and Derby is a trademark of Apache Software Foundation.
- Chrome is a trademark of Google Inc.
- Eclipse is a registered trademark of the Eclipse Foundation, Inc.
- HP-UX is a registered trademark of Hewlett-Packard Development Company, L.P.
- IBM, AIX, DB2, and WebSphere are registered trademarks of International Business Machines Corporation.
- Intel and Pentium are registered trademarks of Intel Corporation in the U.S. and/or other countries.
- JBoss is a registered trademark, and CentOS is a trademark, of Red Hat, Inc. in the U.S. and other countries.
- Linux is a registered trademark of Linus Torvalds.
- Microsoft, Active Directory, Internet Explorer, SharePoint, SQL Server, Visual Studio, and Windows are registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Mozilla and Firefox are registered trademarks of the Mozilla Foundation.
- Oracle and Java are registered trademarks of Oracle and/or its affiliates.
- Progress and OpenEdge are registered trademarks of Progress Software Corporation or one of its subsidiaries or affiliates in the U.S. and other countries.
- Red Hat and Red Hat Enterprise Linux are registered trademarks of Red Hat, Inc. in the U.S. and other countries.
- SAP and SAP NetWeaver are registered trademarks of SAP SE in Germany and in several other countries.
- SUSE is a registered trademark of SUSE, LLC.
- Ubuntu is a registered trademark of Canonical Limited in the United States and/or other countries.
- UNIX is a registered trademark of The Open Group in the United States and other countries.

All other marks contained herein are for informational purposes only and may be trademarks of their respective owners.

Table of Contents

Aurea global support

If you encounter a problem while using an Aurea product or require assistance with downloading the software or upgrading a product release, please, try to:

- Search the articles on the Aurea Knowledge Base for solutions to your issues.
- Search the product documentation and other product-related information that are also available on Support Central.

If you still cannot find a solution, open a ticket on Aurea Support Central. Information about the support organization is available on Support Portal as well.

You can also find the setup files on Support Portal.

For information about purchasing an upgrade or professional services, contact your account executive. If you do not know who your account executive is, or for other queries, contact us through our website.

1

Administering Jive platform

This section includes information about administering and managing the platform, including run books, configuration information, and performance tuning help.

For details, see the following topics:

- Jive and high-availability
- Clustering in Jive
- In-memory caching
- Monitoring your Jive environment
- Operations cookbook
- Fine-tuning performance
- Application management command reference

Jive and high-availability

Jive has been deployed in a wide variety of high-availability (HA) configurations. Here you can find how to design your Jive configuration for high-availability, and how each of the application's components handles failover and recovery.

Supported Jive high-availability configurations

This section describes the supported HA configurations: single data center and multiple data centers.

Jive supports two types of HA configurations:

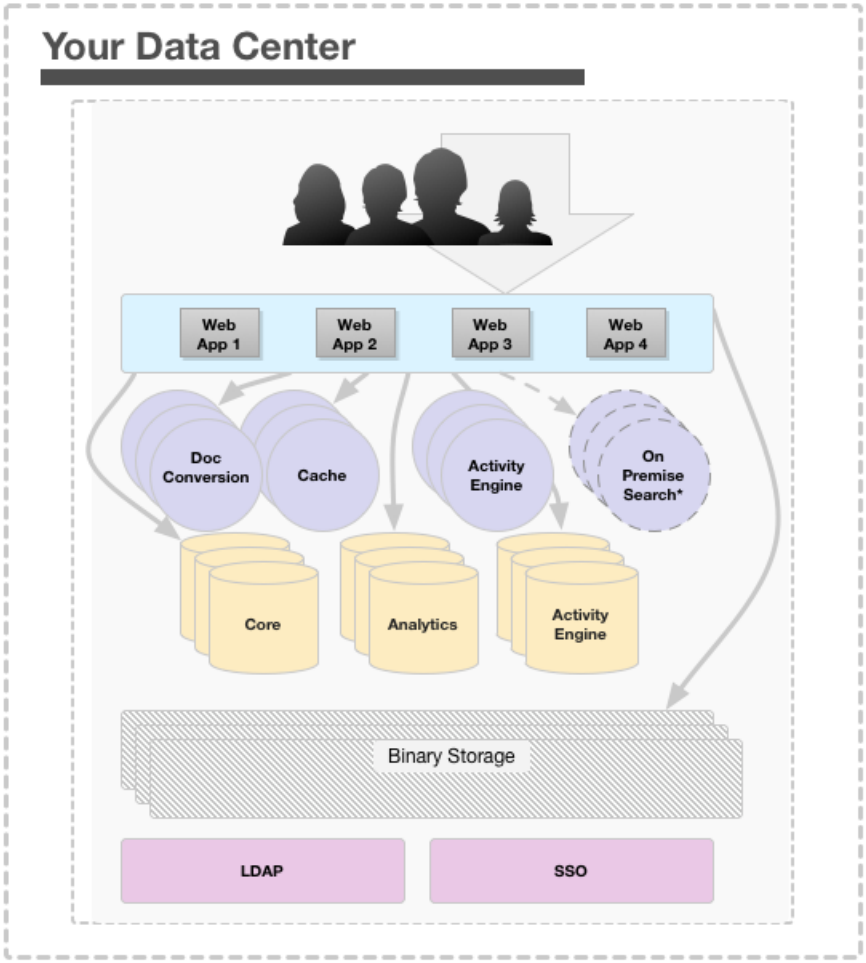
Local high-availability in a single data center	This ensures availability in a single data center deployment through the use of redundant and load-balanced nodes for the web application, cache, document conversion, and databases.
Geographically-distributed high-availability across multiple data centers	This ensures availability across multiple data centers in the event of a disaster affecting the active data center through the use of redundant (but not hot) nodes for the web application, cache, document conversion, and databases.

Designing single data center HA configuration

The single data center HA configuration ensures availability through the use of redundant and load-balanced nodes for the web application, cache, document conversion, and databases. This configuration requires that all of the nodes be physically located in the same data center.

Note: You may choose to configure redundant databases and replicate data between them. Jive does not provide database replication configuration or support. Your data center or DBA team, or both, must provide database support for you and your configuration. We have seen customers successfully deploy Oracle RAC and SQL server HA configurations with Jive.

As an example, here is how a single data center HA configuration might look (your configuration may vary):



**Arrows indicate flow of information retrieval.

*The Cloud Search Service is the default for customers in the Jive Data Center. The On Premise Search alternative is available for customers unable to use the default Search Service and may not contain all features available with the default Search Service.

In this configuration, the web application nodes are configured in a cluster and deployed behind a load balancer, preferably an enterprise-grade load balancer, such as the F5 BIG-IP. For more information about how to set up a cluster, see Clustering in Jive on page 46). At Jive, we have observed that most customers deploy multiple web applications nodes (usually three or more) in a single data center configuration.

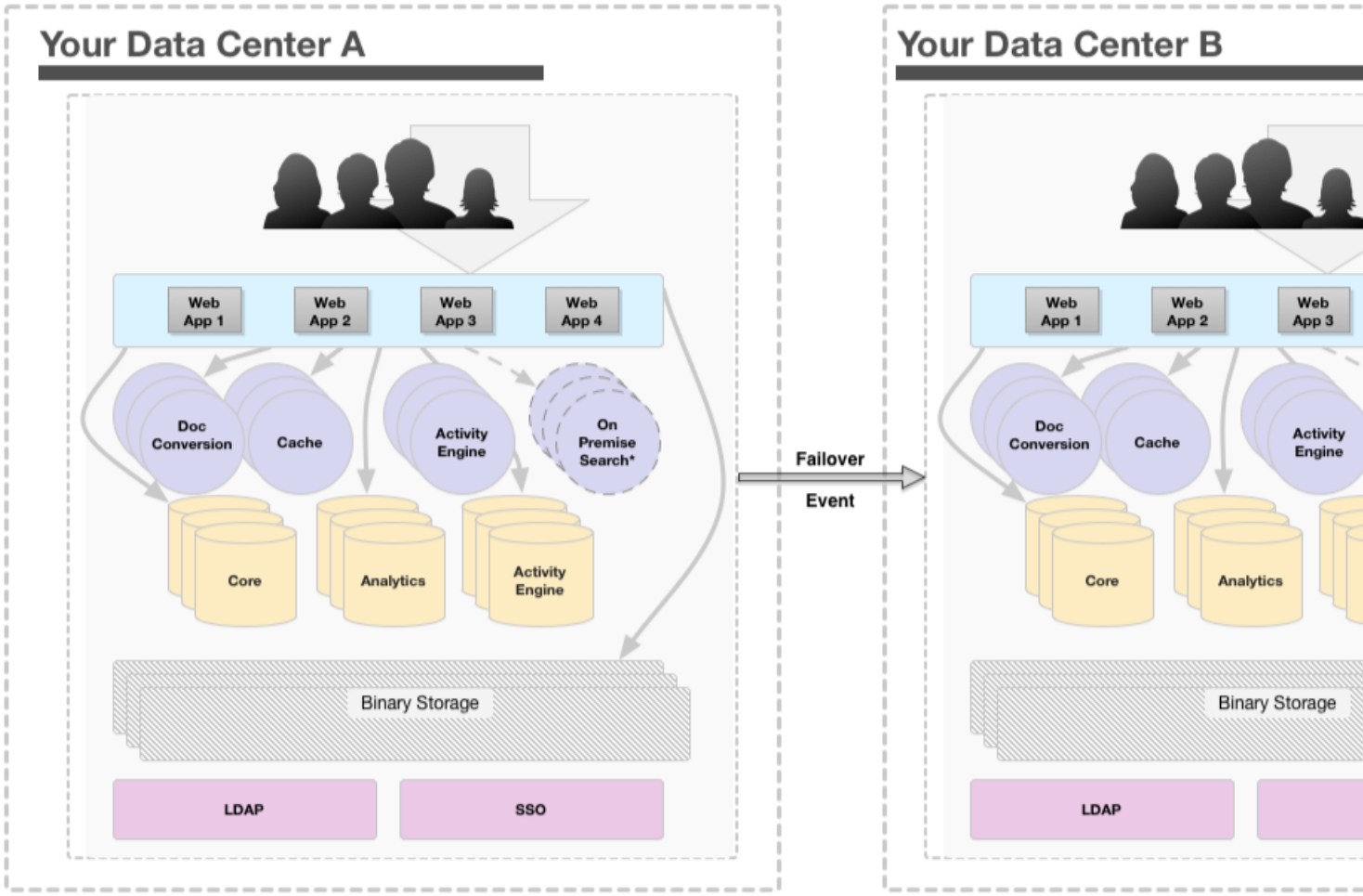
Alternatively, you could choose to deploy additional web application nodes in the cluster as passive participants which are online and standing by, and available to come online if necessary. For example, the Jive application may be running on these nodes, but not serving requests.

Designing multiple data center HA configuration

The multiple data centers HA configuration ensures availability across geographically-distributed and redundant Jive platforms as an active or passive configuration. Note that you cannot have Jive running in multiple data centers simultaneously.

Note: You may choose to configure redundant databases and replicate data between them. Jive does not provide database replication configuration or support. Your data center or DBA team, or both, must provide database support for you and your configuration. We have seen customers successfully deploy Oracle RAC and SQL server HA configurations with Jive.

As an example, here is how a multiple data center HA configuration might look (your configuration may vary).



**Arrows indicate flow of information retrieval.

*The Cloud Search Service is the default for customers in the Jive Data Center. The On Premise Search alternative is available for customers unable to use the default Search Service and may not contain all features available with the default Search Service.

In this configuration, the web application nodes are configured in a cluster and deployed behind a load balancer, preferably an enterprise-grade load balancer such as the F5 BIG-IP. For more information about how to set up a cluster, see Clustering in Jive on page 46.

In the passive standby data center system, you can leave the web application nodes booted up at the operating system level, but not the Jive application (while the active production data center is running). However, the cache nodes, the Document Conversion service nodes, the Activity Engine nodes, and the database nodes in the passive standby data center may be left on.

Caution: The web app nodes in the passive standby data center cannot be up and available all the time. If you attempted to do this, the web application nodes in the passive data center would communicate with the active production cluster as if they were part of it, which would cause catastrophic issues in your production cluster.

For information on how to bring up Data Center B in the case of a failure, see Starting up after failover on page 45.

Supported HA Search

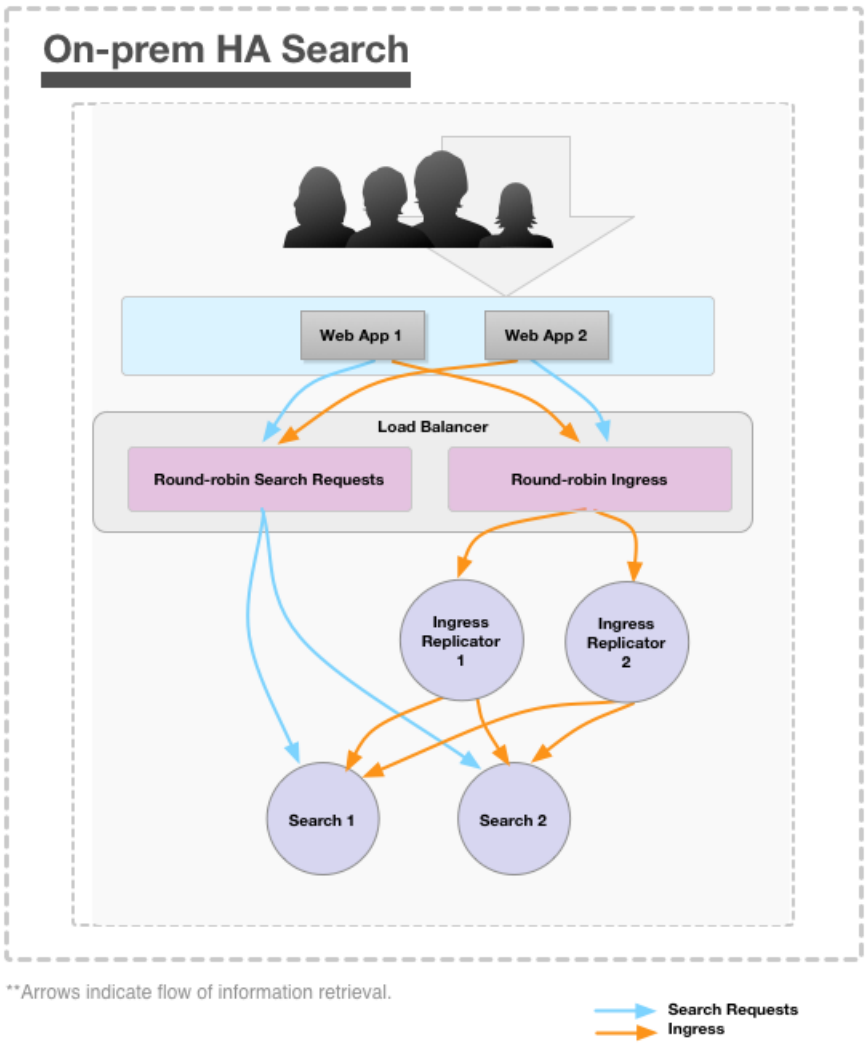
On-premise HA Search supports two kinds of HA configurations: single data center and multiple data centers.

You can configure your On-Premise Search nodes in either of these HA configurations:

- Single data center, described in Designing single data center HA configuration on page 8
- Multiple data centers, described in Designing multiple data center HA configuration on page 9

To understand what you need for your HA search configuration, see Capacity and scaling considerations on page 23 and Required nodes for an On-Premise HA Search service on page 23.

The following diagram shows the simplest HA configuration for search.



You might also find it helpful to understand how a single on-premise search node works before diving into Configuring the Search Service for High-Availability. For more information, see How On-Premise Search works and Configuring On-Premise Search service for high-availability on page 21.

Configuring Jive for high-availability

This section describes the special design and configuration recommendations for each component in a highly available Jive configuration.

Note: The jive_startup.xml located on the web application nodes in `/usr/local/jive/applications/[name of your application]/home/` stores the connection string of the Core Application databases. The connection string for all other nodes, including all other databases, is set via the Admin Console and stored in the Core Application databases.

Configuring Web Application servers for high-availability

If you're setting up a web application node as a template, then copying the home directory (such as `/usr/local/jive/applications/instance_name/home`) to the other web application nodes in the cluster, you must remove the node.id file and the crypto directory from the home directory before starting the server. The application will correctly populate them.

Although not required, we highly recommend that you regularly (at least once a week, but preferably once a day) replicate the contents of `/jiveHome` from one of the web application nodes in the active production data center to all of the web nodes in the warm standby data center. For more information on what is important to persist in a disaster recovery situation, see Restoring Web Application server file system on page 41.

For more information on how to configure the core application databases, see Configuring Core Application database for high-availability on page 17.

Caution: When making a copy of `jiveHome` on the production instance, the Jive application must NOT be running (if it is running, the Lucene index may get copied in a corrupt state and the resulting copied index will be invalid). The easiest way of working around this issue is to have a web application node that participates in the production cluster but does NOT serve HTTP requests. Shut down the Jive application on it on a nightly basis, make a copy of `jiveHome`, and then, after the copy is complete, restart the Jive application on that node.

The following items are stored in `/jiveHome` and they are particularly important to replicate. For more on what is important in `/jiveHome`, see Restoring Web Application server file system on page 41.

jive_startup.xml	This file contains all of the configuration information for the platform.
/search	This is the most important directory to synchronize. It contains the Lucene search index, which is what all of the people search is based on. Note that if you choose not to replicate this directory on a regular basis (weekly or daily), when or if a failover occurs from the production data center to the standby data center, every web application node in the new standby production data center will begin the process of updating its local search index, *which is a CPU and an I/O-intensive process that could take hours*, depending on how much content has been added since the last update.
/themes	This directory contains all of the theme information for your community.

Setting up connection string

The jive_startup.xml located on the web application nodes in `/usr/local/jive/applications/[name of your application]/home/` stores the connection string of the core application databases. The connection string for all other nodes, including all other databases, is set via the Admin Console and stored in the core application databases.

The application requires you to add a DNS name or IP address for the database server. The database connection string can either be an IP address or a domain name but should be dynamic in case of a failure at the database layer.

Here's how to set it up:

- If a DNS name is used to specify the location of the database servers (this is the preferred method), the names must resolve to a database server local to the data center.

  Using the web node names from the example above, but substituting a DNS name (`db01.example.com`) in the configuration instead of an IP address, the DNS name must resolve to the database server `db01-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com`, and must resolve to `db01-dcB.example.com` when requested by either of the web application nodes `wa01-dcB.example.com` or `wa02-dcB.example.com`.

- If an IP address is used to specify the location of the database servers, it must be a virtual IP address that resolves to a database server in the local data center.

  For example, given web application nodes `wa01-dcA.example.com` and `wa02-dcA.example.com`, both in data center A, and web application nodes `wa01-dcB.example.com` and `wa02-dcB.example.com` in data center B, all pointing to the virtual IP address `172.16.100.3` in `/usr/local/jive/applications/[name of your application]/home/jive_startup.xml`, the IP address must resolve to the database server `db01-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com`, and must resolve to `db01-dcB.example.com` when requested by either of the web application nodes `wa01-dcB.example.com` or `wa02-dcB.example.com`.

Configuring Activity Engine server for high-availability

You configure the Activity Engine service in the Admin Console. You must enter either a DNS name (preferred) or an IP address, specifying the location of one or more Activity Engine services.

The Jive core platform requires a separate server and database to manage users' activity streams and recommendations. The Activity Engine service handles a number of key features in the Jive application including the All Activity stream, streams for places and people, watch emails, trending content and people, and personalized recommendations (through a connection to a Cloud service). For more information about the Activity Engine database in an HA configuration, see Designing single data center HA configuration on page 8.

You configure the Activity Engine service in the Admin Console at **System** > **Settings** > **Activity Engine** . You must enter either a DNS name (preferred) or an IP address, specifying the location of one or more Activity Engine services.

In both the single data center and the multiple data centers high availability configurations, Jive Software recommends that you configure the service with a DNS name that resolves to a machine local to its data center.

For example, given web application nodes `wa01-dcA.example.com` and `wa02-dcA.example.com`, both in data center A and web nodes `wa01-dcB.example.com` and `wa02-dcB.example.com` in data center B, all pointing to the DNS name `activity-service.example.com` via the Admin Console setting, the name must resolve to the Activity Engine server `activity-service-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com` and must resolve to `activity-service-dcB.example.com` when requested by either of the nodes `wa01-dcB.example.com` or `wa02-dcB.example.com`.

For more information single and multiple data centers high availability configurations, see Designing single data center HA configuration on page 8 and Designing multiple data center HA configuration on page 9.

Configuring Cache servers for high-availability

In a multiple data center HA configuration, there are special requirements for handling the cache servers.

Every Jive deployment larger than a single node is going to require at least a single node to provide caching services. Within a single data center, high availability can be achieved through the use of three or more cache servers (two cache servers within a single data center is not supported and can cause data loss), all of which you configure via the Admin Console. For more information on single data center configuration, see Designing single data center HA configuration on page 8.

Caution:

If you are implementing an HA caching configuration with Jive, *you must use three or more cache servers*. Two are not supported. This is because each cache PUT operation must succeed on two different cache servers. Therefore, be aware that HA implementations may result in significant performance issues to accommodate successful writes across multiple cache servers.

Use `jive set cache.hostnames list_of_hostnames` to set the cache machine addresses. You can use the comma-separated list of IP addresses or domain names, but be consistent with the format (use IP addresses or domain names, but not both) and order you use. This list should be the same on all cache servers, and well as in the Admin Console. For more information on setting up cache servers, see Adding cache server machines on page 58.

Install the cache server on a machine separate from the web application nodes in the cluster. The cache server is available to all the web app nodes in the cluster; in fact, you can't create a cluster without declaring the address of a cache server. For more information about how caching works in the application, see In-memory caching on page 54.

Caution: Only one data center can be the active (live) system. Therefore, caching requests should never be sent from a web app node in data center A to a cache node in data center B.

Setting up connection string

The application requires you to add a DNS name or IP address for every cache server deployed with the application. You set this connection string via the Admin Console at **System** > **Settings** > **Caches** . This string is then stored in the Core Application databases. For more information on what must be persisted in the core application database during disaster recovery, see Restoring database with persistent properties on page 42.

Because the web application nodes require low latency, the web application nodes must not make cache requests across geographically-distributed data centers. To deal with this in a multiple data center HA configuration, you need to correctly set up the DNS name or IP address of the cache servers. Here's how you do that:

- If a DNS name is used to specify the location of the cache servers (this is the preferred method), the names must resolve to a cache server short name on the web application nodes in each respective data center.

  Using the web node names from the example above, but substituting a DNS short name `ex:(cache01)` in the configuration instead of an IP address, the DNS name must resolve to the cache server `cache01-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com`, and must resolve to `cache01-dcB.example.com` when requested by either of the web application nodes `wa01-dcB.example.com` or `wa02-dcB.example.com`.

- If an IP address is used to specify the location of the cache servers, it must be a virtual IP address that resolves to a cache server in the local data center.

  For example, given web application nodes `wa01-dcA.example.com` and `wa02-dcA.example.com`, both in data center A, and web application nodes `wa01-dcB.example.com` and `wa02-dcB.example.com` in data center B, all pointing to the virtual IP address `172.16.100.3` in the Admin Console cache server configuration page, the IP address must resolve to the cache server `cache01-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com`, and must resolve to `cache01-dcB.example.com` when requested by either of the web application nodes `wa01-dcB.example.com` or `wa02-dcB.example.com`.

For more information on multiple data center HA configuration, see Designing multiple data center HA configuration on page 9.

Configuring Document Conversion server for high-availability

Here you can find how to configure the Document Conversion server for a single or multiple data center HA configuration.

Jive gives users the ability to upload Office and Adobe PDF documents to the community for easy content sharing and collaboration. The Document Conversion service converts uploaded documents to a standard PDF format and then converts them again to Adobe Flash (.swf files) so that they can then be viewed in a web browser without needing to open the document's native software application.

The Document Conversion service must run on a separate node in the deployment because it consumes a significant amount of CPU and memory.

Setting up connection string

The application requires you to add a DNS name or IP address for the Document Conversion server deployed with the application. You set this connection string via the Admin Console at **System** > **Settings** > **Document Conversion** . This string is then stored in the Core Application databases. For more information on what must be persisted in the Core Application database during disaster recovery, see Restoring database with persistent properties on page 42.

You must enter a DNS name (preferred) or an IP address specifying the location of the Document Conversion server so that when a user uploads one of the supported document conversion types to the community, the web application can first save the document to the storage service, and then send a request to the Document Conversion service to perform the conversion.

In both of the supported HA configurations, described in Supported Jive high-availability configurations on page 7, we recommend that you configure the Document Conversion service with a DNS name that resolves to a machine local to that data center.

For example, if you have web application nodes `wa01-dcA.example.com` and `wa02-dcA.example.com`, both in data center A, and web application nodes `wa01-dcB.example.com` and `wa02-dcB.example.com` in the data center B, all pointing to the DNS name `conversion-service.example.com` via the Admin Console setting, the name must resolve to the Document Conversion server `conversion-service-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com`, and must resolve to `conversion-service-dcB.example.com` when requested by either `wa01-dcB.example.com` or `wa02-dcB.example.com`.

Additionally, because the Document Conversion service nodes are stateless, you can configure the service to live behind a load balancer, thereby making the Document Conversion server itself fault-tolerant. As an example, given the above scenario of two web nodes pointing to a DNS name `conversion-service.example.com`, you could configure the DNS name to use round-robin to load balance the requests across multiple Document Conversion service nodes, or it could resolve to the IP address of a load balancer, such as an F5 BIG-IP, which itself load balances and provides fault-tolerance across the Document Conversion services.

Configuring Core Application database for high-availability

The core application database supports the following: Microsoft SQL, Oracle, Postgres, and MySQL.

All of the database information here assumes that you have successfully deployed your database system of choice in an HA configuration, ensuring that the database server itself is not a single point of failure.

Note: You may choose to configure redundant databases and replicate data between them. Jive does not provide database replication configuration or support. Your data center or DBA team, or both, must provide database support for you and your

configuration. We have seen customers successfully deploy Oracle RAC and SQL server HA configurations with Jive.

Location of web application database configuration information

The web application database configuration information is stored on the web application nodes in an XML file that lives in Jive home (usually `/usr/local/jive/applications/[name of your application]/home/jive_startup.xml`). For more information on how to set up the core application database string on the web application nodes, see Configuring Web Application servers for high-availability on page 13. For more information on what must be persisted in the core application database during disaster recovery, see Restoring database with persistent properties on page 42.

Supported Core Application databases

Microsoft SQL Server

Jive supports the JTDS database driver for communication between the Jive instance and Microsoft SQL Server. While Jive does not specifically perform load or functional tests against Microsoft SQL Server in a cluster or failover configuration, the JTDS driver does appear to support SQL Server clustering.

Jive does not currently support the Microsoft JDBC driver. Jive is aware of and is actively working with customers who have deployed Jive in an HA configuration using Microsoft SQL Server.

Oracle

Jive supports the OCI database driver for both the core web application and the Activity Engine application, which is supported by Oracle with their Oracle RAC database system deployments. While Jive does not specifically perform load or functional tests against an Oracle in a RAC cluster or failover configuration, the OCI driver does appear to support Oracle RAC. Jive is aware of and is actively working with customers who have deployed Jive in an HA configuration using Oracle RAC.

Postgres

Jive supports several version of Postgres, as listed in Supported database engines. The Postgres 9 supports two different types of high availability: hot standby, streaming replication and warm standby, and log shipping which, in theory, would allow for transparent and automatic failover, assuming there is a way to automatically redirect all traffic from the live production database server to the hot standby backup server. Jive is not aware of any customers who have deployed against a Postgres instance configured in this manner. If a mechanism exists for failing over a database server from one node to another, or from one data center to another, without disrupting the web application nodes, Jive supports the configuration.

MySQL

Similar to Postgres, there are multiple ways of deploying a highly available MySQL database system. Also similar to Postgres, Jive is not aware of any customers who have deployed against a MySQL instance configured in this manner. If a mechanism exists for failing over a database server from one node to another, or from one data center to another, without disrupting the web application nodes, Jive will support the configuration.

Configuring Analytics database for high-availability

The Analytics database has special HA considerations because its connection string is stored in the core application's database.

All of the database information here assumes that you have successfully deployed your database system of choice in an HA configuration, ensuring that the database server itself is not a single point of failure.

The Analytics service supports the following database types:

- Oracle
- Postgres

Location of Analytics database configuration information

The Analytics database connection string, username, and password are stored in a table in the core application database in an encrypted format (not in an XML file).

Setting up connection string

The application requires you to add a DNS name or IP address for the Analytics database server deployed with the application. You set this connection string via the Admin Console at **Reporting** > **Settings** > **Analytics** . This string is then stored in the Core Application databases. For more information on what must be persisted in the core application database during disaster recovery, see Restoring database with persistent properties on page 42.

In the event of a failover, there are specific ways that the application uses this connection string to determine which Analytics server to failover to. Therefore, be especially careful when setting up the connection string as follows:

- If a DNS name is used in the connection string (this is the preferred method), the name must resolve to an Analytics database server that is resolvable and reachable by the web application nodes in each respective data center (A or B).

  Using the web node names, but substituting a DNS name (`analytics-virtual.example.com`) in the connection string instead of an IP address, the DNS name must resolve to the Analytics database server `analytics01-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com` and must resolve to `analytics01-dcB.example.com` when requested by either of the nodes `wa01-dcB.example.com` or `wa02-dcB.example.com`.

- If an IP address is used in the connection string, it must be a virtual IP address that points to the Analytics database server that's available from both data centers.

  For example, given web nodes `wa01-dcA.example.com` and `wa02-dcA.example.com`, both in the data center A, and web nodes `wa01-dcB.example.com` and `wa02-dcB.example.com` in data center B, all pointing to the virtual IP address `172.16.100.2` in the Analytics database connection string, said IP address must resolve to the Analytics database server `analytics01-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com` and must resolve to `analytics01-dcB.example.com` when requested by either of the web application nodes `wa01-dcB.example.com` or `wa02-dcB.example.com`.

Configuring Activity Engine database for high-availability

The Activity Engine database supports several types of databases. For the database connection string, you must use a dynamic link, which can be either an IP address or a domain name.

All of the database information here assumes that you have successfully deployed your database system of choice in an HA configuration, ensuring that the database server itself is not a single point of failure.

The Activity Engine service supports the following database types:

- SQL Server
- Oracle
- Postgres
- MySQL

Locating of Activity Engine database configuration information

The Activity Engine nodes maintain their database configuration information in the core application database. The database connection string can either be an IP address or a domain name, but should be dynamic in case of a failure at the database layer.

Setting up connection string

The application requires you to add a DNS name or IP address for the Activity Engine database server deployed with the application. You set this connection string via the Admin Console at **System** > **Settings** > **Activity Engine** . This string is then stored in the core application databases. For more information on what must be persisted in the core application database during disaster recovery, see Restoring database with persistent properties on page 42.

In the event of a failover, there are specific ways which the application uses the connection string to determine which Activity Engine database server to failover to. Therefore, be especially careful when setting up the connection string as follows:

- If a DNS name is used in the connection string (this is the preferred method), the name must resolve to an Activity Engine database server that is resolvable and reachable by the web application nodes in each respective data center (A or B).

  Using the web node names, but substituting a DNS name (activityeng-virtual.example.com) in the connection string instead of an IP address, the DNS name must resolve to the Activity Engine database server `acteng01-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com` and must resolve to `acteng01-dcB.example.com` when requested by either of the nodes `wa01-dcB.example.com` or `wa02-dcB.example.com`.

- If an IP address is used in the connection string, it must be a virtual IP address that points to the Activity Engine database server that's available from both data centers.

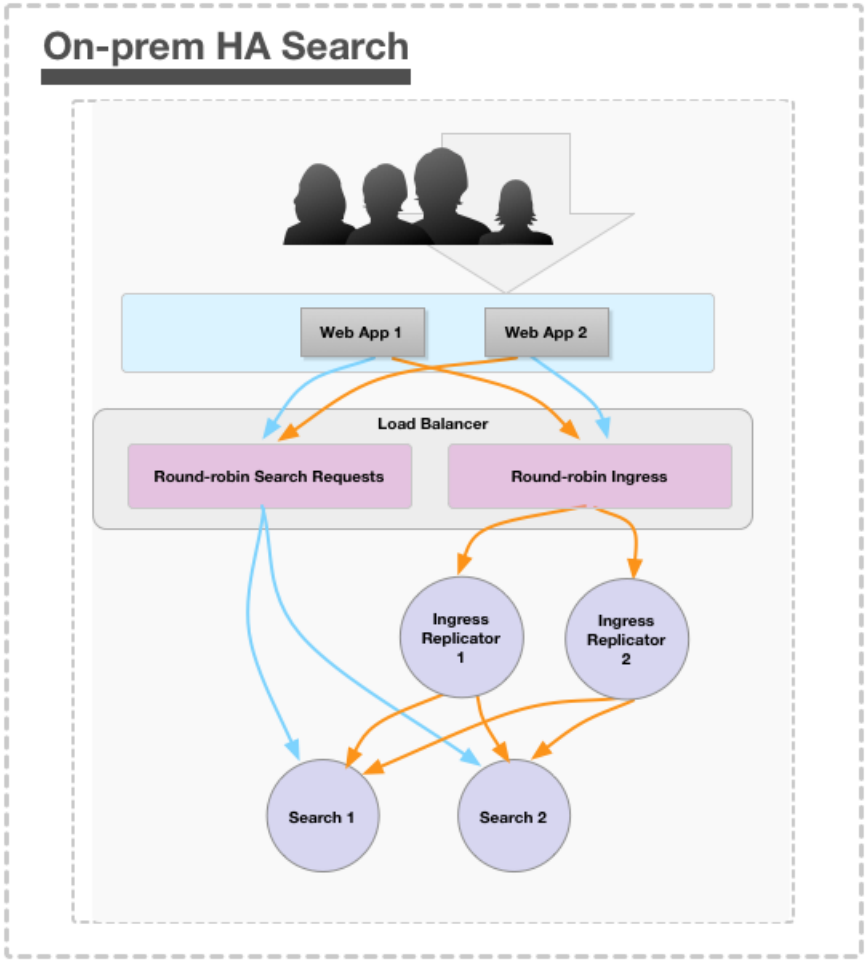
For example, given web nodes `wa01-dcA.example.com` and `wa02-dcA.example.com`, both in the data center A and web nodes `wa01-dcB.example.com` and `wa02-dcB.example.com` in data center B, all pointing to the virtual IP address `172.16.100.2` in the Activity Engine database connection string, said IP address must resolve to the Activity Engine database server `acteng01-dcA.example.com` when requested by either `wa01-dcA.example.com` or `wa02-dcA.example.com` and must resolve to `acteng01-dcB.example.com` when requested by either of the web application nodes `wa01-dcB.example.com` or `wa02-dcB.example.com`.

Configuring On-Premise Search service for high-availability

To create an on-premise HA search service, you need separate search nodes configured as part of the larger HA deployment.

The following diagram shows the simplest HA configuration for On-Premise Search. Jive has been deployed in a wide variety of HA configurations. This is only an example of an HA search configuration. Your configuration may vary depending on your specific requirements.

Note: The ingress replicator and search nodes have built-in health checks via `host:port/loadbalance/eligible`. Therefore, the load balancer can maintain the pool of available nodes via the health check and then round-robin requests across available nodes. In this way, the load balancer can detect any failures of ingress replicator or search nodes.


On-prem HA Search
Web App 1
Web App 2
Load Balancer
Round-robin Search Requests
Round-robin Ingress
Ingress Replicator 1
Ingress Replicator 2
Search 1
Search 2
**Arrows indicate flow of information retrieval.
Search Requests
Ingress

Capacity and scaling considerations

Understanding how failures occur may help you determine the number of On-Premise Search nodes you need in your HA deployment.

There are two types of On-Premise Search failures that can occur:

Search failure	In this case, new search requests are not serviced during an outage. To design your Jive configuration to guard against this, you need to have more than one search broker and have them on separate nodes. This also means that you need to deploy an ingress replicator, which can be co-located with each of the search brokers.
Ingressfailure	In this case, new content is not indexed during an outage. To design your Jive configuration to guard against this, you need to deploy multiple ingress replicators and on separate nodes. Each can be co-located with a search broker, if desired. Note that the protection against index failure is not absolute; for example, while new content continues to be indexed during an outage, a small amount of content that was created just before the outage could fail to be indexed until the ingress replicator comes back online.

Generally, it makes sense to keep capacity considerations separate from your decision about your HA search configuration. Having additional search brokers adds capacity to services search requests, but a deployment would need to be very large and very active before a single search broker could not handle the requests. In that case, it would be simpler to add a CPU rather than a new search broker.

The key capacity consideration is the amount of memory available to the search brokers. Remember that the data is not shared, so each search broker needs to have enough memory to effectively handle the size of the index. Therefore, if HA is not needed, adding a second search broker for the purpose of scaling is a big investment because you would need to commit more memory to it.

For capacity planning guidelines of your Jive configuration, see Deployment sizing and capacity planning.

Required nodes for an On-Premise HA Search service

To configure your on-premise search nodes for HA, you need to split the search service from the ingress replicator so that each service can be made redundant. You need a load balancer to direct traffic to each set of services. Here you can find an example with four nodes described: two for ingress and two for search. Your configuration may vary depending on your requirements.

Note that an ingress replicator service an run on the same host as the search service.

Table 1: On-Premise HA Search Configuration Node Requirements

Search component	Nodes required	Description
Ingress replicator	2 separate nodes	The ingress replicators journal everything to disk to guarantee all ingressed activities (such as new content or content updates) will be delivered to a search node at least once.
Search service	2 separate nodes	The search nodes handle incoming search queries and return search results. For a diagram of how this works, see Configuring On-Premise Search service for high-availability on page 21.

HA Search setup overview

When you set up the Jive platform for On-Premise HA search, you perform several steps in a specific sequence.

Note: As of Jive 7.0, the search index rebuild process has been improved so that you no longer have to rebuild the search index on one node and then copy it to all of the other search nodes. In versions 7.0+, the ingress replicators automatically send rebuild traffic to all search nodes. Because of this change, all of the search nodes must be available before starting a search rebuild. This ensures that the search index on the search service nodes are always consistent.

In the following topics, we provide an example of an HA search configuration setup that uses the following ports and hosts. Your configuration may vary depending on your requirements.

- 2 search nodes: `search01.yourdomain.com`, `search02.yourdomain.com`, port `30000`
- 2 ingress replicator nodes: `ir01.yourdomain.com`, `ir02.yourdomain.com`, port `29000`
- 1 haproxy node: `haproxy.yourdomain.com`, load balancing the search nodes on port `20000` and load balancing the ingress replicator nodes on port `19000`

Step	What you're installing	Required or optional	Installation instructions
1	–	Required	Understand the supported HA search configurations, described in Supported HA Search on page 11.
2	–	Required	Determine how many nodes you need in your HA search configuration, as described in Required nodes for an On-Premise HA Search service on page 23. Typically, this includes two search nodes and two ingress replicators, but you may have more of each, depending on your requirements.
3	Application RPM	Required	Install the Jive Linux package on each node of your HA search configuration (the search servers and the ingress replicator servers). For more information, see Installing Jive package and starting up.
4	Search servers	Required	Add another search server to your configuration, as described in Installing one or more Search servers on page 25.
5	Ingress replicators	Required	Add another ingress replicator server to your configuration, as described in Installing one or more Search Ingress replicators on page 26.
6	HA search proxy	Required	Add an HA proxy to your configuration, as described in Setting up the HA search proxy on page 26.
7	–	Required	Include a JSON services file on each search server in the configuration, as described in Services directory for HA Search on page 28.

Installing one or more Search servers

Use these steps to add a search server to your on-premise HA search configuration.

To install a Search server:

1. Install the Jive Linux package on the search servers that will be part of your HA search service configuration. For more information, see Installing Jive package and starting up.
2. Enable the search service to start by typing the following command as the jive user:

```
jive enable search
```

3. Verify that the port is correct for your setup in the main-args.properties file (located in `/usr/local/jive/services/search-service/`). In this example, it looks like this:

   ```
   PORT=30000
   ```

4. Restart the search service by using the following command as the jive user:

   ```
   jive restart search
   ```

Installing one or more Search Ingress replicators

Use these steps to add an ingress replicator server to your On-Premise HA Search configuration.

To add an ingress replicator server:

1. Install the Jive Linux package on the ingress replicator servers that will be part of your HA search service configuration.

   For more information, see Installing Jive package and starting up.

2. Enable the ingress replicator to start by typing the following command as the jive user:

   ```
   jive enable ingress-replicator
   ```

3. Verify that the following properties in the main-args.properties file (located in `/usr/local/jive/services/ingress-replicator-service/`). In this example, they look like this:

   ```
   PORT=29000
   REPLICATE_INGRESS_TO_THESE_COMMA_SEPARATED_HOST_COLON_PORT_TUPLES=guaranteed:all:search0
   REPLICATE_REBUILD_TO_THESE_COMMA_SEPARATED_HOST_COLON_PORT_TUPLES=search01.eng.yourdomai
   REPLICATE_INDEX_MANAGE_TO_THESE_COMMA_SEPARATED_HOST_COLON_PORT_TUPLES=search01.eng.your
   ```

4. Restart the ingress replicator service using the following command as the jive user:

   ```
   jive restart ingress-replicator
   ```

Setting up the HA search proxy

To configure a redundant HA search environment, you need a proxy server to load balance requests to each search server and ingress replicator.

Below is an example of two search nodes, two ingress nodes, and one proxy that load balances each pair. We also recommend setting up one reporting endpoint for the load balancer itself.

The proxy should be running on its own server. This example uses `CentOS` and `haproxy`. You may use other proxy services depending on your requirements.

To setup a proxy server for search environment:

1. Install a proxy server using yum.

   ```
   yum install haproxy
   ```

2. Set the proxy so that it starts up automatically whenever the server restarts.

   ```
   chkconfig haproxy on
   ```

3. Edit the proxy's config file (located in `/etc/haproxy/haproxy.cfg`) as follows and save the changes. The proxy listens on port 20000 for search and port 19000 for ingress. It also exposes a status UI on port 8085.

```
frontend  main
    bind haproxy.yourdomain.com:20000,haproxy.yourdomain.com:19000
    acl ingress-r   dst_port 19000
    use_backend             ingress-replicator if ingress-r
    default_backend         search

backend search
    balance     roundrobin
    option httpchk GET /ping
    server  search01 search01.yourdomain.com:30000 check
    server  search02 search02.yourdomain.com:30000 check

backend ingress-replicator
    balance     roundrobin
    option httpchk GET /ping
    server  ir01 ir01.yourdomain.com:29000 check
    server  ir02 ir02.yourdomain.com:29000 check

listen status haproxy.yourdomain.com:8085
   stats enable
   stats uri /
```

4. Restart the haproxy.

   The control scripts are located in `/etc/init.d/haproxy`.

5. Test the setup by sending search queries through curl and ensuring they are showing up in the logs of the destination machines.

6. In the Admin Console, go to **System** > **Settings** > **Search** and update the search service host field with `haproxy.yourdomain.com` and the search service port to `20000`.

7. Restart the application.

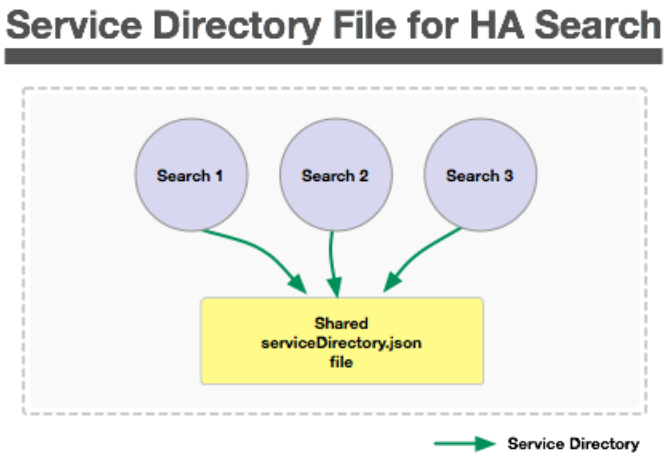
Services directory for HA Search

The Search service relies on a file named serviceDirectory.json. This file should be identical for all of the search servers (not the ingress replicators) in your HA configuration.

Here you can find a sample file.

- All hosts and ports point to load balancer-exposed addresses.
- The entries for `directory` and `search` should all point to the load balancer address for search.
- The `activityIngress`, `rebuildSearchIndex`, and `searchIndexManage` entries should point to the load balancer address for the ingress replicator.
- If you are connecting only one Jive instance to your HA search configuration, you do not need to modify the second section of serviceDirectory.json (`tenantSpecific-ServiceDirectory`). `tenantSpecificServiceDirectory` allows you to uniquely configure multiple instances to a shared search service.

```
{
  "defaultServiceDirectory" : {
    "directory" : {
      "host" : "haproxy.yourdomain.com",
      "port" : 20000
    },
    "search" : {
      "host" : "haproxy.yourdomain.com",
      "port" : 20000
    },
    "searchIndexManage" : {
      "host" : "haproxy.yourdomain.com",
      "port" : 19000
    },
    "rebuildSearchIndex" : {
      "host" : "haproxy.yourdomain.com",
      "port" : 19000
    },
    "activityIngress" : {
      "host" : "haproxy.yourdomain.com",
      "port" : 19000
    }
  }
}
```



Adding an On-Premise HA Search server

There are two basic steps to adding (or removing) a search node in an already existing HA search environment: introducing the new search node and then configuring the ingress replicators to recognize the new node. The following example assumes you have two search nodes and you are adding a third.

Adding new search node to your configuration

Here you can find how to add a new search node to your configuration.

1. Take all ingress replicators out of the load balancer rotation.
2. Wait for all ingress replicators to deliver pending activities. To do this, from the command line run: `curl http://ir0x.yourdomain.com:29000/logging/metric/list-Counters | grep InQueue`

   This should return something like the following, which indicates that you are replicating to `search01.yourdomain.com:30000` and `search02.yourdomain.com:30000`, and that the InQueue metric is `0.0`, which means that all activities have been delivered:

   ```
   {"key":"counter>com.jivesoftware.service.activity.stream.replicator.ActivityStreamReplic

   {"key":"counter>com.jivesoftware.service.activity.stream.replicator.ActivityStreamReplic
   ```
3. Shut down the ingress replicators:

   ```
   jive stop ingress-replicator
   ```
4. Install the third search service 3 (`search03.yourdomain.com`).

   For more information, see Installing one or more Search servers on page 25.
5. Update the ingress replicator settings for the additional search node.

   For more information, see Pointing to new search node on page 31.
6. Rebuild the search index. You can use one of the following ways:

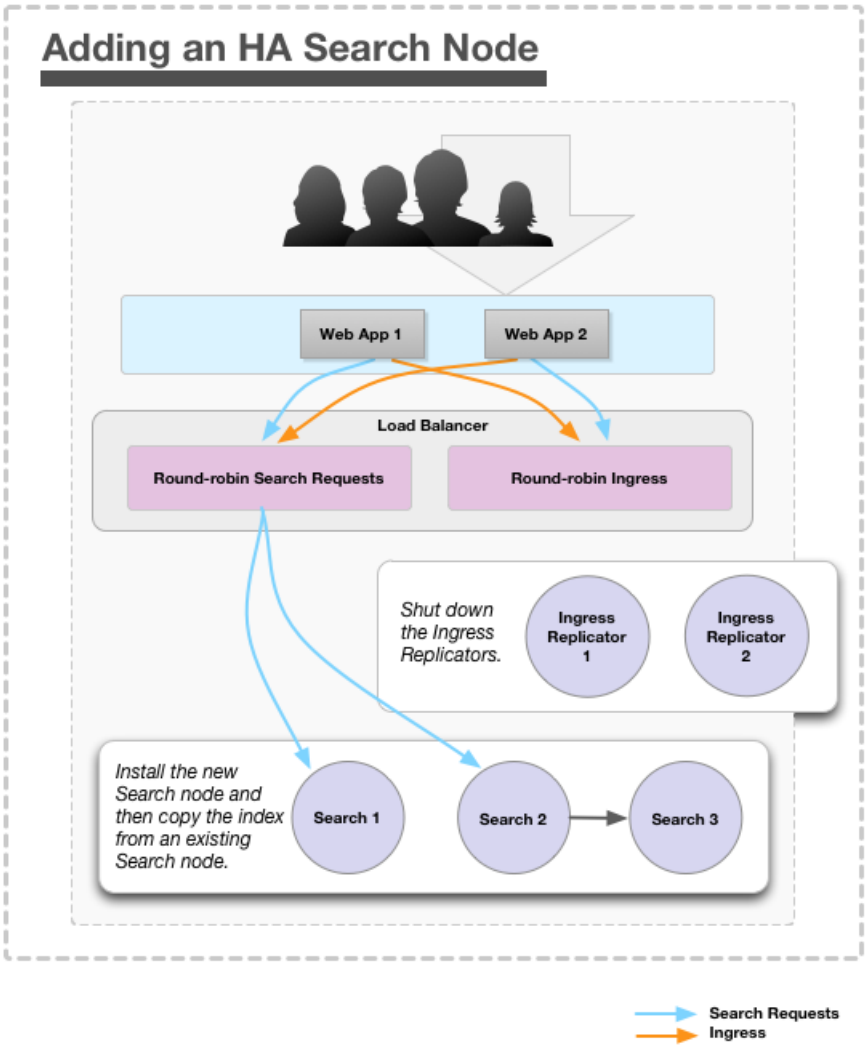
Options	Description
If the index is large (multiple gigabytes)	It's faster to copy the active search index from search service 1 or 2 to the new search service 3. You can find the active search index by looking in `/usr/local/jive/services/search-service/main-args.properties` file for `CONTENT_SEARCH_HOME_DIRECTORY=var/data/contentSearch/`. This property lists the location of the search indexes.
If the index is small (less than two gigabytes) or you do not care how long the rebuild takes	1. Complete Steps Step 7 on page 30-Step 9 on page 30. 2. Start a rebuild in the Admin Console at **Admin Console: System** > **Settings** > **Search** .

7. Start search service 3.

   ```
   jive start search
   ```

8. Add the third search node to your load balancer.
9. Start up all of the ingress replicators and make sure all of the search services are running.

   ```
   jive start ingress-replicator
   ```



Pointing to new search node

You should point the other nodes to the new search node you have added.

After you've successfully added the new node, as described in Adding new search node to your configuration on page 29, you need the other nodes to point to the new search service node. Here is how to do that:

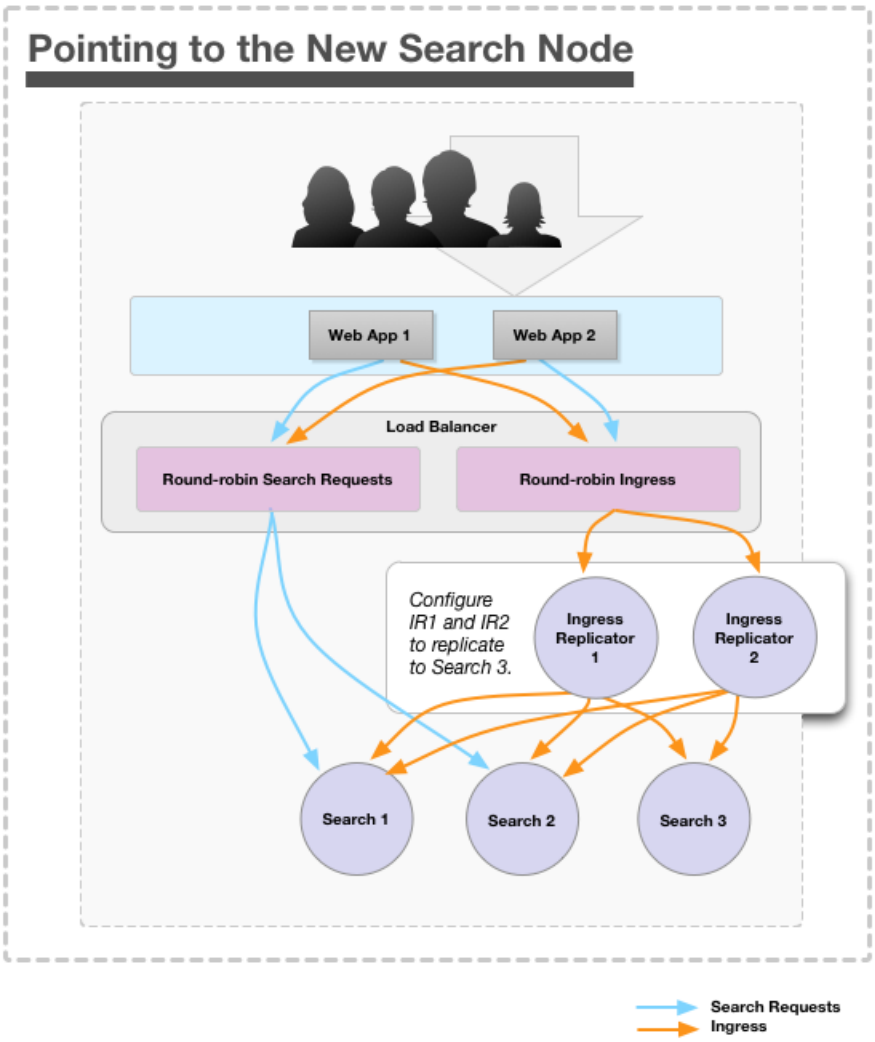
1. In `/usr/local/jive/services/ingress-replicator-service/`,re-configureingress replicator 1 and 2 to include search service 3. Set the load balancer to forward the ingress service to all machines holding the ingress replicator, and the ingress replicators to forward to all machines where the search service runs. The relevant parameters in the main-args.properties file for this example look like this:

```
PORT=29000
REPLICATE_INGRESS_TO_THESE_COMMA_SEPARATED_HOST_COLON_PORT_TUPLES=guaranteed:all:search0
REPLICATE_REBUILD_TO_THESE_COMMA_SEPARATED_HOST_COLON_PORT_TUPLES=search01.eng.yourdomai
REPLICATE_INDEX_MANAGE_TO_THESE_COMMA_SEPARATED_HOST_COLON_PORT_TUPLES=search01.eng.your
```

2. Restart the ingress replicator services.

jive restart ingress-replicator

3. Add the ingress replicator services back into the load balancer configuration.



Failover behavior of HA servers

This section describes the expected failover and data recovery process for each component in a high-availability Jive configuration.

System failover

If a failover occurs, the length of an outage depends on several factors.

In the event of a failover, the length of your outage depends on whether or not the system can correctly redirect everything to the passive system, how long it takes for either the manual or automatic processes to run where you're switching DNS/IP addresses, and the time it takes to bring up the application servers in the passive-now-active system.

Therefore, you should ensure that you've set up everything correctly, as described in Configuring Jive for high-availability on page 12.

If you've set up the configuration so that all of the nodes correctly redirect to the passive system, here is an overview of what happens during a failover.

Single data center	Community users would not see any disruption. For more information on single data center configuration, see Designing single data center HA configuration on page 8.
Multiple data center	The length of disruption to end users would be between five and ten minutes, during which time, users would see a maintenance page when attempting to view community pages. Downtime depends, of course, on how fast you have set up database replication and how quickly your community administrators react to the outage of Data Center A. For more information on multiple data center configuration, see Designing multiple data center HA configuration on page 9.

For more information on how to start up the system after a failover, see Starting up after failover on page 45.

Failover and data recovery in caches

If or when a cache server becomes unavailable, the web application nodes continue indefinitely to attempt to communicate with the cache server until it is brought back online.

A web application node determines that a cache server is unavailable after 20 consecutive failed requests to the cache server. The web app node then waits for two seconds before attempting to communicate again with that cache server. This process continues indefinitely until the failed cache server is brought back online. In the meantime, the web app nodes automatically redirect to the next available cache servers.

If the cache server is unavailable at the startup of a web application node, the web node uses local caching. The web app node continues to try to communicate with the external cache server, and does so indefinitely, until the external cache server is brought back online.

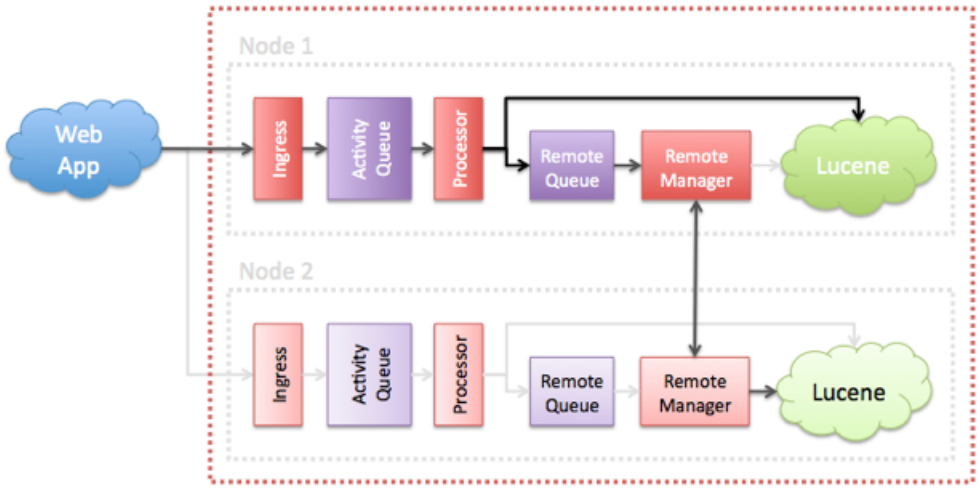
Failover and data recovery in Activity Engine

Here you can find an overview of how the Activity Engine handles activity ingress, egress, and failures.

Activity ingress

In a clustered Jive configuration, when a Jive web application node sends an activity, it selects an appropriate Activity Engine node to which to send the activity. The web app node selects an Activity Engine node based on the following criteria:

User affinity	All Jive web apps in a cluster attempt to route activity from a particular user to the same Activity Engine node.
Sensitive ordering	User affinity is based on node ordering. We recommend keeping consistent the list of Activity Engine nodes defined in the Activity Engine setup.
Failover behavior	If an Activity Engine node becomes unavailable, the web application bans it and reroutes applicable users to the next node in the list. The new routing is kept for a small number of subsequent requests (30) before it is discarded and the original route is again attempted.



After the web application selects an available Activity Engine node, the web application attempts the delivery. If it fails for any reason, the activity is journaled and attempted again later (potentially against a different node). Delivery failure may occur due to the following:

Unavailable node	The node is unreachable or otherwise unavailable between its selection and activity delivery.
Activity not written to the Activity Engine database	The node is unable to write the activity to the database for any reason.
Activity not queued for processing	The node is unable to add the activity to its processing queue for any reason.

Streams Durability

Each Activity Engine node is backed by its own Lucene stream service. While ingress remains unchanged, each node is responsible for replicating its processed data to all other nodes in the Activity Engine cluster. Each node is made aware of its siblings during the web application's registration process.

The replication procedure works as follows:

Disk queue	Fully-processed activities, as well as events (for example, reads, hides, moves), are queued to disk in a simple pipeline destined for all siblings.
Connection pool	Each Activity Engine node establishes up to 10 connections to each of its siblings on the same port used by the web app. Note: Each node will, therefore, accept and establish up to 10 x #-of-siblings additional connections to handle the replication requirement.

Activity Engine Failover and Recovery

If an Activity Engine node goes down:

Unprocessed activity is in limbo	There is no way to reclaim activities that are still queued on disk and activities or events in the replication queue. You must bring the failed node back online for its queue to be processed.
Queues lost to disk failure are recoverable	In the event of a complete disk failure, you can recover unprocessed activities by running a stream rebuild on all nodes. While this guarantees that the unprocessed activities are correctly reflected in all indexes, it is a resource-intensive procedure. Therefore, Jive Software recommends avoiding this scenario if possible.
Lucene is completely mirrored	Because each node has its own complete Lucene index, no stream data goes missing or becomes unreachable. All other data (e.g., follows and email notifications) are persisted to the database and are unaffected by an Activity Engine node failure.
Recoverable cross-communication	If an Activity Engine node is unable to reach one of its siblings during replication, activities/events destined for that sibling are pushed to a "retry" queue where they are reattempted at a later time. After a failed node recovers, replication to that node should resume within approximately 1 minute; so there will be a brief period of stream inaccuracy on the affected node.

After an Activity Engine node recovers:

Processing resumes	The Activity Engine node will continue processing its disk queue, as well as activities/events in the replication queue. No further action is required.
Corrupted queues	In the event that the failure has corrupted one or more disk queues and activities or events in the replication queue, those queues are copied off with the suffix ".corrupt" and can potentially be copied and analyzed. But there is no way to recover corrupt queues. If your replication queue is corrupt, we strongly recommend that you run a stream rebuild on all Activity Engine nodes.

If you decommission an Activity Engine node:

Unprocessed activity can be recovered	If you decommission an Activity Engine node with unprocessed activity in its queues, then we recommend you run a stream rebuild on all nodes. This is a resource-intensive procedure, but we recommend it over attempting to move the unprocessed queue data to another node.
Cross-communication adjusts accordingly	When you remove an Activity Engine node from the web app's Activity Engine configuration, all remaining nodes are made aware of the change. Replication to the decommissioned node ceases immediately.

If you add or re-add a node:

Cross-communication adjusts accordingly	When you add an Activity Engine node to the web app's Activity Engine configuration, all current nodes are made aware of the change. Replication to the commissioned node begins immediately.
Stream rebuild is initiated	When an Activity Engine node is registered with the web app, it is able to detect whether it was a new addition (or re-addition). The node flags itself for a stream rebuild to ensure that its index contains all required stream data.

Note that, while we have made extra efforts to ensure that consistency is maintained automatically during the management of an Activity Engine cluster, in the event of any unexpected inconsistency or missing data, you can correct the issue by initiating a stream rebuild on the affected nodes. The Activity Engine node remains completely accessible while performing a rebuild, during which time streams will fill with users' newest activity first. In addition, Jive supports complex "in/out" configurations which play a large part in activity ingress.

Activity Egress

Because each node maintains its own Lucene index, if an Activity Engine node is in the process of performing a stream rebuild, it is possible for users routed to a particular Activity Engine node to have partly-diminished streams. This state is temporary and resolves itself automatically (similar to a search or browse index rebuild). The most recent stream activity will always be available first, and in no event will any user activity ever be permanently lost. There are some background tasks which are only performed by an elected node. However, as of Jive 6.0, the "per-user stream rebuild" task is no longer applicable and has been removed. Therefore, the only tasks requiring node election are upgrade migrations. In addition, Jive 6.0+ supports complex "in/out" configurations which play a large part in activity egress.

Failover and data recovery in Document Conversion Service

If your Document Conversion server fails, you only lose the ability to convert new documents, you do not lose all the previously converted documents.

Once documents are converted, they are kept in your file storage. So if your Document Conversion server fails, you only lose the ability to convert new documents, until you bring the service back online.

Failover and data recovery in Storage

Any Jive deployment beyond a simple pilot or proof-of-concept must use file system storage, where each web application node reads and writes from a shared disk system via NFS.

Jive supports two types of binary file storage:

Database	This is the default binary file storage.
File system	This is the preferred binary file storage.

External storage failover in single data center HA configuration

In a single data center HA configuration, it's assumed that the external storage is redundant within the data center. For more information, see Designing single data center HA configuration on page 8.

In the event of a local (within the single data center) or a catastrophic (the entire data center) failure, it is assumed that the storage layer that Jive is configured with is redundant and that recovery is handled transparently by the underlying storage system.

External storage failover in multiple data center HA configuration

In a multiple data center HA configuration, it's assumed that the external storage is being replicated across the data centers transparently. For more information, see Designing multiple data center HA configuration on page 9.

Failover and data recovery in Core database

Here you can find how you can prepare for disaster recovery with the Jive Core database.

Note: You may choose to configure redundant databases and replicate data between them. Jive does not provide database replication configuration or support. Your data center or DBA team, or both, must provide database support for you and your configuration. We have seen customers successfully deploy Oracle RAC and SQL server HA configurations with Jive.

Core database failover in single data center HA configuration

For details about a single data center HA configuration, refer to Configuring Core Application database for high-availability on page 17.

Core database failover in multiple data center HA configuration

Disaster Recovery (DR) architecture varies greatly from data center to data center. In a strategy where all systems are fully replicated to a DR facility, you may be able to architect the Jive Platform as you would with a single data center HA configuration. In many cases, however, the DR strategy is more manual and requires a multiple data center HA configuration. For more information, see Designing multiple data center HA configuration on page 9 and Designing single data center HA configuration on page 8.

You may follow one of these possible DR strategies:

- Create backup at a remote location (simplest DR strategy) on page 38
- Use Cold/Warm state servers on page 38

Create backup at a remote location (simplest DR strategy)

If a recent full backup of the database is available at a remote location, it is possible to recover the system to the point in time of the available backup. Upon declaration of a disaster, perform the following:

1. Set up a new cluster of application servers and point them to a new empty database at the recovery facility.
2. After completion of initial setup, save specific property values from the new database's jiveProperty table somewhere or copy them to a backup table. For more information on which properties need to persist, see Restoring database with persistent properties on page 42.
3. Restore the database backup over the newly created database.
4. Apply the properties that should persist (as determined in Step 2 on page 38) from the new database to the restored database jiveProperty table. For more information, see Restoring database with persistent properties on page 42.
5. Restart the application server.

Note: Data loss in the event of a disaster can be minimized by pushing incremental backups, and transaction write-ahead logs to the remote facility. Point-in-time recovery may be performed up to the point of the disaster if backups are available.

Use Cold/Warm state servers

Streaming replication, or Write-Ahead Logging (WAL), maintains a recent copy of the database at the remote facility. You need to make sure a cluster of application servers are already set up and attached to an empty database. Upon declaration of disaster, the replicated database should replace the empty database with specific values in jiveProperty persisted to reflect the DR environment. For more information, see Restoring database with persistent properties on page 42.

It is important to consider which properties should be replaced with values from the original production site, and which values should reflect values of the new facility. The persisted values depend on system configuration, as well as any customizations that could impact the jiveProperty table. Review, validation, and live testing of a system failover eliminates any potential issues that could arise during an actual disaster.

Failover and data recovery in Analytics database

The Analytics database connection string, user name, and password are stored in a table in the Core Application database in an encrypted format. In the event of a failure, you want to be sure the web app nodes are calling the correct Analytics database server.

Note: You may choose to configure redundant databases and replicate data between them. Jive does not provide database replication configuration or support. Your data center or DBA team, or both, must provide database support for you and your configuration. We have seen customers successfully deploy Oracle RAC and SQL server HA configurations with Jive.

Analytics database failover in single data center HA configuration

In a single data center HA configuration, if Analytics database 1 fails over to Analytics database 2, this failure is transparent to the web application nodes due to the database driver layer controlling traffic between the web app nodes and the analytics database. So if the Analytics database 1 failed, community users would not notice the failure because the driver would automatically redirect web app node requests over to Analytics database 2.

For more information, see Designing single data center HA configuration on page 8.

Analytics database failover in multiple data center HA configuration

The Analytics database server connection string, user name, and password are stored in a table in the Core Application database in an encrypted format (not in an XML file). Because of this, how you set up the connection string affects how the web application nodes call and resolve the Analytics database server. In the event of a failure in a multiple data center HA configuration, you want to be sure the web app nodes are calling the correct Analytics database server. Therefore, be especially careful when setting up the Analytics database connection string.

For more information, see Designing multiple data center HA configuration on page 9 and Configuring Analytics database for high-availability on page 19.

Failover and data recovery in Search Service

In the case of a failure, your ingress replicators or search service nodes may be unreachable. This topic describes what happens during an outage.

Note: To avoid non-recoverable disk failures, we recommend that you configure the ingress replicator journals and search service indexes so that they are written

to durable storage. For each ingress replicator, allocate at least 20 GB for journal storage. For each search service, allocate at least 50 GB for index storage. Monitor these storage volumes for remaining capacity, maintaining 25% free capacity.

In the case of a failure of any given node in your HA search configuration, here is an overview of what happens.

Ingress replicator node fails

The ingress replicator journals everything to disk to guarantee all ingressed activities are delivered at least once. If the service fails or is stopped, it sends any remaining journaled events when it starts back up. If the service cannot come back up due to a non-recoverable disk failure, then a full rebuild is required.

If both ingress replicators fail (or you have only one and it fails), for the duration of the outage no new content is indexed; but, when the ingress replicator comes back online, the search service catches up with the indexed content (due to local caching on the web application nodes); therefore, the search service does not miss anything.

For more information on rebuilding search index, see Rebuilding On-prem HA Search Service on page 45.

Search service node fails

If search service 1 or 2 is offline for any reason, the ingress replicator retains the undelivered activities. When search service 1 or 2 is restored to a healthy state, the undelivered activities are sent to the restored service. While previously undelivered activities are being fed into the newly restored service, the search indexes will be out of sync. After all undelivered activities have been received by the restored service, the indexes are synced.

If the service cannot be restored due to a non-recoverable disk failure, then you need to remove and re-add the affected search service.

If you leave a search service down for a very long period of time (such as for several weeks), you may run out of disk space because the ingress replicator services will be persisting to disk until the configured search service is restored. If you don't plan to restore the offline search service, then remove the offline search service from all ingress replicator configuration files and restart the ingress replicators.

For more information, see Adding an On-Premise HA Search server on page 29.

Recovering Jive after failure

This section describes which system properties, files, and directories need to be restored when recovering Jive after a failure.

You should configure Jive for High-Availability, as described in Configuring Jive for high-availability on page 12. Your HA configuration should enable you to automatically or manually start up after a failover. The following provides system properties, files, and directories that need to be restored in the case of a failure without any failover configuration:

- On the Core Application database nodes, restore the properties in the jiveProperty table that need to be persisted. For a list of these properties, see Restoring database with persistent properties on page 42.
- On the web application nodes, restore the files and directories in `/jiveHome` that need to be recovered. For a list of these files, see Restoring Web Application server file system on page 41.

Restoring Web Application server file system

The Web Application server file system contains certain files and directories that should persist in a disaster recovery situation.

You should prepare for a disaster situation by configuring the web application servers for high-availability, as described in Configuring Web Application servers for high-availability on page 13. If you don't have a warm standby data center, you should copy the following files and directories found in `/usr/local/jive` to the same location in your new web application servers:

- `applications/*/home/search`
- `applications/*/home/themes`
- `applications/*/home/attachments/cache`
- `applications/*/home/images/cache`
- `applications/*/home/documents/cache`
- `applications/*/home/cache/jiveSBS`
- `applications/*/home/jms`
- `applications/*/home/jive_startup.xml`
- `applications/*/home/attachments/*.txt`
- `applications/*/home/images/*.bin`
- `*.pid`
- `applications/saasagent`
- `tomcat/lib/postgresql*.jar`
- `etc/httpd/sites/proxies/maint.con`

Tip: If you have a text extraction location set up for search, as described in Configuring text extraction location, you should copy that directory over to the new system to save time while reindexing.

Restoring database with persistent properties

In the jiveProperty table, there are several properties that may need to be changed when restoring data in a Disaster Recovery (DR) situation.

For more information, see Failover and data recovery in Core database on page 37.

Table 2: Properties that may need to be changed on failover0

This table provides a list of properties that may need to be changed when restoring your Core Application database.

Property	Definition and notes
`cache.clustering.enabled`	Enables caching in a clustered environment. Possible values are either `true` or `false`. This should be set to `true` unless the new instance (disaster recovery instance) does not have clustering (more than one web application node) enabled, in which case this value should be set to `false`.
`jive.storage-provider.FileStorage-Provider.rootDirectory`	Specifies the path to the mount point for binary storage if you configured your system to store binary content on the file system versus the database, which is the default. This should be the same across data centers, but it can change if the mount points are different.
`jive.storage-provider.cache.enabled`	If you use NFS for your binstore file system, then you should set this property to `false` to prevent excessive traffic on the network from caching the NFS mounts.
`jive.auth.forceSecure`	Possible values are either `true` or `false`. Set this value to `true` as part of forcing all traffic to the instance over SSL. You only need to change this value if the failover data center has a different HTTP configuration, for example, if SSL is not enabled.

Property	Definition and notes
`jive.master.encryption.key.node`	This property represents one of the `node.id` files in the cluster. Remove the value in the event of a failover. Upon restart, the web applications populates it automatically with the correct value.
`antivirus.virusScannerUri`	Determines the URI of the virus scan server. This may change if you failover to a new data center. The URI must be in one of the following formats: • ClamAV: `tcp://hostname/clamav` • McAfee: `icap://hostname:port/RESPMOD`

Table 3: Wildcards that extract additional properties

This table shows wildcards that extract additional properties. Each wildcard includes an example of properties that were extracted and may change during a failover. Your actual results will vary depending on how your system is set up.

Wildcard	Notes
`jive.cluster.jgroup.servers.address%`	Remove all values in the event of a failover. Upon restart, the web application populates this table automatically. For example, remove the values in the following property: `jive.cluster.jgroup.servers.address.8aa5ce1c-f15b-4d2e-ba90-f9cf424af3b2`
`jive.cache.voldemort.servers.address%`	If you use a DNS name or a virtual IP, then the values of extracted properties do not need to change. However, if the IP address of the cache server in the new data center is different than the IP address of the cache server in the failed data center, then this property and any of its children should be updated with the valid IP address or DNS name. For example, you may need to update the value in `jive.cache.voldemort.servers.address.1`

Wildcard	Notes
`__jive.analytics.%`	If you use a DNS name or a virtual IP, then the values of extracted properties do not need to change. However, if the IP address of the cache server in the new data center is different than the IP address of the cache server in the failed data center, then the `__jive.analytics.database.serverURL` property should be updated with the valid IP address or DNS name. In addition, you may need to update `__jive.analytics.database.username` and `__jive.analytics.database.password` if they change, but generally they don't. Extracted properties that might change: • `__jive.analytics.database.serverURL` • `__jive.analytics.database.username` • `__jive.analytics.database.password`
`jive.dv.%`	The `jive.dv.service.hosts` may need to change if the IP address or domain name changes for the new data center, making it different from the failed data center. Extracted properties that might change: • `jive.dv.service.hosts`
`%smtp%`	The `mail.smtp.host` or `mail.smtp.port` may need to change if the host or port changes for the new data center making it different from the failed data center. Extracted properties that might change: • `mail.smtp.host` • `mail.smtp.port`
`%ldap%`	The `ldap.host`, `ldap.port`, and `ldap.sslEnabled` may need to change if the values change for the new data center, making them different from the failed data center. Extracted properties that might change: • `ldap.host` • `ldap.port` • `ldap.sslEnabled`

Wildcard	Notes
`%checkmail%`	The `checkmail.host`, `checkmail.port` or `checkmail.protocol` may need to change if the values change for the new data center, making them different from the failed data center. Extracted properties that might change: • `checkmail.host` • `checkmail.port` • `checkmail.protocol`
`%activity%`	The `jive.activitymanager.endpoints` may need to change if the value changes for the new data center, making it different from the failed data center. Extracted properties that might change: • `jive.activitymanager.endpoints`

Rebuilding On-prem HA Search Service

As of Jive 7.0, the HA search rebuild process has been simplified. The ingress replicators now send rebuild traffic to all of the search nodes. Therefore, ensure that all of the search service nodes are available before you start a rebuild.

Fastpath: Admin Console: System > Settings > Search

To start a search rebuild:

- In the Admin Console, go to **System** > **Settings** > **Search** and click **Rebuild Index**.

For more information, see Reindexing content search and Reindexing user search.

Starting up after failover

Depending on whether you're able to correctly set up dynamic redirects for the nodes, you start up the newly active system automatically or manually.

Starting up automatically

If you have correctly set up the dynamic redirects, as described in Configuring Jive for high-availability on page 12, after a failover, you start up the new web application nodes by running the `jive start` command on each enabled web app node in the new system. Doing this starts up all of the other nodes and services in the new configuration.

Starting up manually

If, for whatever reasons, you are unable to set up dynamic redirects, then, in the event of a failure, you would need to manually do the following in the passive-now-active data center before starting it up:

1. On the web application nodes, edit the jive_startup.xml file to point to the new data center as described in Configuring Web Application servers for high-availability on page 13.
2. On the core application database nodes, edit the Activity Engine database property (`jive.eae.db.url`) in the jiveProperty table to point to the new Activity Engine database.

   For a connection string configuration example, see Configuring Activity Engine database for high-availability on page 20.
3. On the core application database nodes, edit the Analytics database property (`__jive.analytics.database.serverURL`) in the jiveProperty table to point to the new Analytics database.

   For a connection string configuration example, see Configuring Analytics database for high-availability on page 19.

After you have manually performed the above tasks, start up the new web application nodes by running the `jive start` command on each enabled web app node. Doing this starts up all of the other nodes and services in the newly active configuration.

Clustering in Jive

Here you can find an overview of the system that supports clustered installations of Jive.

Clustering overview

While they're different services, the clustering and caching systems interoperate. In fact, an application cluster requires the presence of a separate cache server for caching data for use by all application server nodes in the cluster.

For information on installing the application on a cluster, see Setting up cluster on page 50.

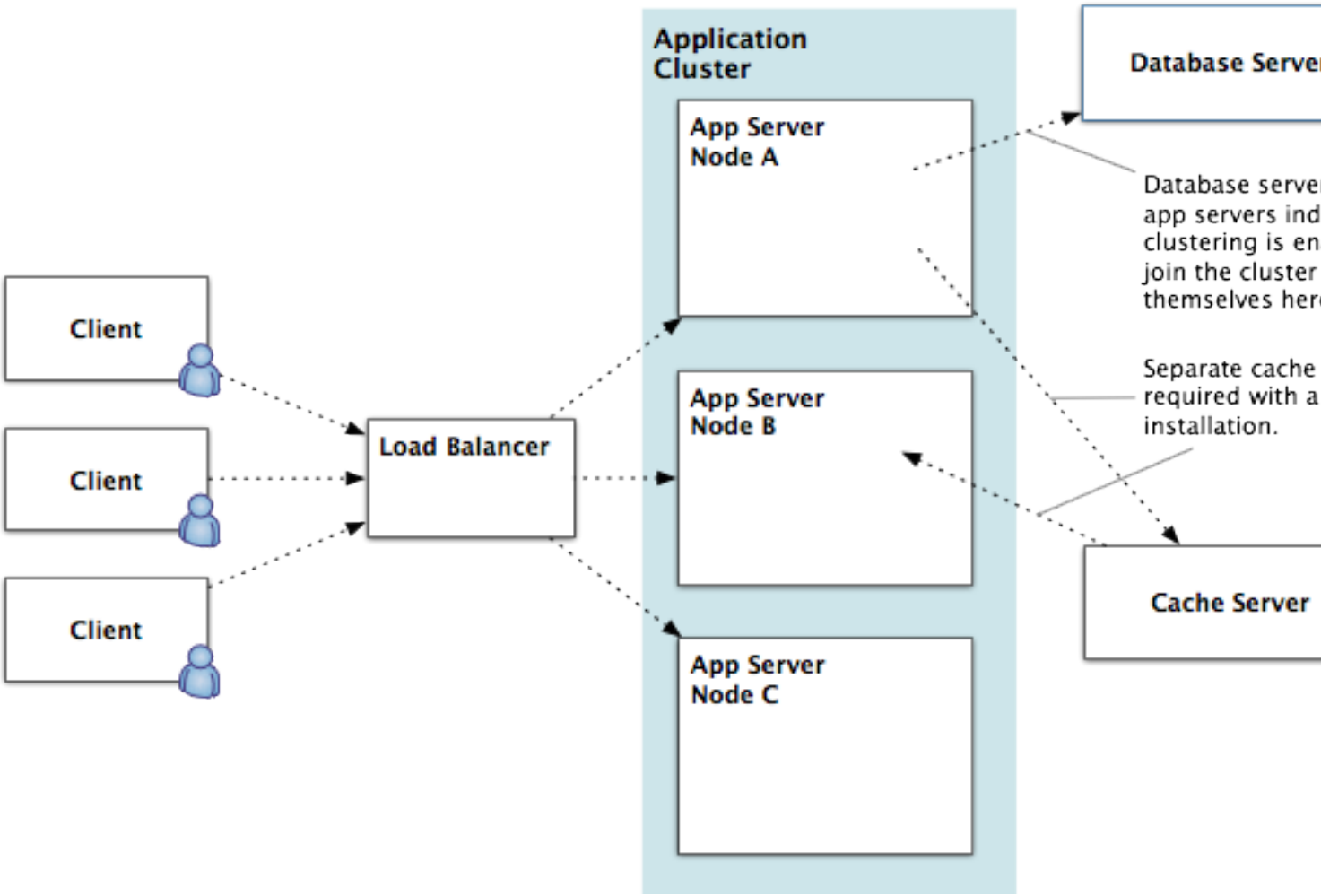
Parts of clustering system

Application servers	In the middle-tier, multiple application servers are set up and the clustering feature is enabled. Caches between the application instances are automatically synchronized. If a particular application server fails, the load balancer detects this and removes the server from the cluster.
Cache server	On a separate machine from application servers is a cache server that is available to all application server nodes in the cluster. Note that you can't create a cluster without declaring the address of a cache server.

Database server	All instances in a cluster share the same database.
Load balancer	Between users and the application servers is a load-balancing device. The device may be hardware- or software-based. Every user has a session (represented by a unique cookie value) that allows stateful data to be maintained while they are using the application. Each session is created on a particular application server. The load balancer must be *session-aware*, meaning that it inspects the cookie value and always sends a given user's requests to the same application server during a given session. Without session-aware load balancing, the load balancer could send requests to any application server in the cluster, scrambling results for a given user.

Typical cluster configuration

A typical cluster configuration is shown in the diagram below. Note that the database server and cache server are separate nodes, but not part of the cluster.



The existence of a cluster is defined in the database, which stores the TCP endpoint for each node in the cluster. A node knows it's supposed to be in a cluster because the database it is using shows that clustering is enabled. Nodes in a cluster use the application database to register their presence and locate other nodes. Here's how that works at startup:

1. When an application server machine starts up, it checks the database to discover the TCP endpoint (IP address and port) it should bind to.
2. If the node can't find its TCP endpoint in the database (because this is the first time is has started and tried to join a cluster, for example), it looks for the first non-loopback local address it can use. It tries to bind to a default port (7800). If it fails, it scans up to port 7850 until it finds a port it can bind to. If this fails, the node doesn't join the cluster.
3. Having established an endpoint for itself, the node notes the other node addresses it found in the database.
4. The node joins the cluster.

Clustering best practices

Here are a few best practice suggestions for clustered installations.

- Ensure that the number of nodes in your cluster is greater than what you'll need to handle the load you're getting. For example, if you're at capacity with three nodes, then the cluster will fail when one of those nodes goes down. Provision excess capacity so that your deployment can tolerate a node's failure.
- If you have document conversion enabled, and one of the machines is faster than the others, start that one first.

Managing application cluster

The clustering system is designed to make it easy to add and remove cluster nodes. By virtue of connecting to an application database that other cluster nodes are using, a node automatically discovers and joins the cluster on startup. You can remove a node using the Admin Console.

For a high-level view of how clustering works, Clustering in Jive on page 46.

Enabling and disabling clusters

You can enable or disable clustering in the Admin Console. For more information, see Configuring cluster node (optional) for more information.

Adding cluster nodes

When you add a new node to the cluster, you must first manually copy the encryption keys from the `/usr/local/jive/applications/app_name/home/crypto` directory to each of the other nodes. Then restart every node in the cluster to ensure that the new node is seen by the others.

You might also be interested in Setting up cluster on page 50, which describes the process for installing or upgrading the application on an entire cluster.

Fastpath: Admin Console: System > Settings > Cluster

To add a note to a cluster:

1. Install the application on the new node, as described in Installing Jive package and starting up.
2. Finish setting up the new node, restart it, and let it get up and running.

   By default, the node scan for the TCP endpoint and register itself in the database. You can also specify a particular endpoint in the Admin Console, as described in Configuring cluster node (optional).
3. Restart all nodes in the cluster so that the other nodes can become aware of the new node.

Removing cluster nodes

When you want to be sure that a node's registration is removed from the database, you can remove a node from a cluster by using the Admin Console.

Fastpath: Admin Console: System > Settings > Cluster

1. Ensure that the node you want to remove is shut down.
2. In the Admin Console for any of the nodes in the cluster, on the Cluster page, locate the address of the node you want to remove.
3. Next to the node's address, select the **Remove** check box.
4. Click **Save** to save settings and remove the address from the database.

   Settings will be automatically replicated across the cluster.

Clustering FAQ

Here are some frequently asked questions and answers about clustering.

Do all cluster members need to be on the same local network?

Yes. It's better for performance.

Is it possible to have more than one cluster per physical network?

You can have two deployments (for example, CommunityA and CommunityB) on the same physical network operating as two clusters. You cannot have a single deployment (for example, CommunityA) separated into two clusters.

How do configuration files work in a cluster?

All configuration data (except bootstrap information such as database connection information) is stored in the database. Changing configuration settings on one cluster member will automatically update them on all other cluster members.

Can I set a cluster node's TCP endpoint to a particular value?

Yes. If you have an address and port you want to force a node to bind to, you can do that by setting those values in the Admin Console. If you do that, the node tries that address and port only; it won't scan for an address and port if the one you specify fails. For more information, see Configuring cluster node (optional).

How will I know if a cluster node has failed or can't be found by the cluster?

The Cluster page in the Admin Console displays a list of nodes in the cluster. See Configuring cluster node (optional) for more information.

Setting up cluster

Before you set up the nodes in a cluster you should have already configured a cache server, as described in the high-level steps below. The cluster requires the presence of a cache server in order to cache data that should be available to all nodes in the cluster.If your cache server isn't configured and running, you won't be able to set up the cluster.

Note: Your license determines whether or not clustering is enabled and how many nodes are supported. To check on the number of clustered servers your license allows, see the license information after logging into the admin console.

Topology

The nodes in a cluster need to be installed on the same subnet, and preferably on the same switch. You cannot install nodes in a cluster across a WAN.

Important notes

Before you get started, let me show start and related to setting up a cluster.

Important: If you're upgrading and copying the home directory (such as `/usr/local/jive/applications/<instance_name>/home`) from the older installation, you must preserve the node.id file and the crypto directory from the home directory before starting the server. The value stored in this file must be unique among the cluster nodes; that is, each node in a cluster must have a unique value in the node.id file. You must preserve the node.id file because it plays a role in storing encrypted information in the cluster; if that file is lost, you can lose access to the encrypted information.

- If you are deploying a new cluster, it is permissible to copy the contents of the home directory from the first node (where you set up clustering) to subsequent nodes — with the exception of the node.id file. Do not copy the node.id file to subsequent nodes. If the node.id file does not exist, the application generates a new file on startup.
- Always wait for the first node in the cluster to be up and running with clustering enabled before you start other cluster nodes. Waiting for a minute or more between

starting each node ensures the nodes are not in competition. As the senior member, the first node you start has a unique role in the cluster. For more information, see Clustering in Jive on page 46.

- The cache server must be cleared and restarted before upgraded application server nodes are started and try to talk to the cache.
- If you're upgrading a plugin, clear the cache for that plugin and shut down the cache server first.
- The clocks on all machines must be synchronized in order for caching to work correctly. For more information, see Managing in-memory cache servers on page 57. Also, if you're running in a virtualized environment, you must have VMware tools installed in order to counteract clock drift.
- If you're running in a virtualized environment, you must have VMware tools installed in order to counteract clock drift.
- If your deployment places a firewall between cluster nodes and cache servers, be sure to leave the following ports open between machines: 6666, 6667 and 6650. Caching won't work correctly unless these are open.
- Port 6650 should be blocked to *external* access (but not between the cluster nodes!) so that any access outside of the data center is prohibited. This is to prevent operations allowed by JMX not to be executed on the cache server.

High-level new installation steps

Important: If, as part of your new installation, you're setting up one node as a template, then copying the home directory (such as `/usr/local/jive/applications/<instance_name>/home`) to other nodes in the cluster, you must **remove** the node.id file and the crypto directory from the home directory before starting the server. The application will correctly populate them.

To set up a cluster:

1. Use the application package (such as the RPM on Linux) to set up a cache server on a separate machine. For more information, see Setting up cache server. Note the cache server address for use in setting up application servers.
2. Before proceeding, make sure the cache server you set up is running. It must be running while you set up application server nodes.
3. On each node in the cluster, install the application instance using the package (RPM on Linux or package on Solaris), but don't run the Setup wizard yet.

   For the installation instructions about installing the application with a package, see Installing Jive package and starting up.
4. Start one node and navigate to its instance with a web browser. In the setup screen provided, enter the address of the cache server you installed, then complete the Setup wizard.
5. After you've finished with the Setup wizard, restart the application server.
6. Manually copy the encryption keys from the `/usr/local/jive/applications/app_name/home/crypto` directory to each of the other nodes.
7. Start the application server on each of the other nodes. Because it's connecting to the same database used by the server on the node you've already set up, the

server on each subsequent node detects that clustering is enabled. Each also picks up the configuration you set on the first node.

8. After setting up all of the application server nodes and running them once, restart all servers in the cluster to ensure that the address of each node in the cluster is known to all the other nodes. The entire cluster must be bounced after all the nodes are set up.

Upgrading claster

When you're upgrading from a version prior to 4.5.0 (and after 3.0.0), you should follow the high-level steps listed here. Note that you must upgrade from version 4.0.0 or later.

To upgrade a cluster:

1. Stop all application server nodes in the cluster.
2. Use the application package (such as the RPM on Linux) to set up a cache server on a separate machine. For more information, see Setting up cache server. Note the cache server address for use later in setting up application servers.
3. Before proceeding, make sure the cache server you set up is running. It must be running while you set up application server nodes.
4. On each node in the cluster, upgrade the application instance by using the package, but don't run the Setup wizard yet.

   For more information on upgrading the package, see Upgrading Jive package.
5. Start one node and navigate to its instance with a web browser. Work through the upgrade tool, allowing it to run the upgrade tasks it lists.
6. Restart the application server you've upgraded, then go to it with a web browser again.
7. Enter the address of the cache server you installed, then complete the Setup wizard.
8. After you've finished with the Setup wizard, restart the application server.
9. Start the application server on each of the other nodes. Because it's connecting to the same database used by the server on the node you've already set up, the server on each subsequent node detects that clustering is enabled. Each also picks up the configuration you set on the first node.
10. After setting up all of the application server nodes and running them once, restart all servers in the cluster to ensure that the address of each node in the cluster is known to all the other nodes. The entire cluster must be bounced after all the nodes are set up.

Troubleshooting caching and clustering

This topic lists caching- or clustering-related problems that can arise, as well as tools and best practices.

Log files related to caching

If a cache server machine name or IP address is invalid, you get verbose messages on the command line. You also get the messages in log files found in `$JIVE_HOME/var/logs/`.

- cache-gc.log — Output from garbage collection of the cache process.
- cache-service.out — Cache startup messages, output from the cache processes, showing start flags, restarts, and general errors.

Misconfiguration through mismatched cache address lists

If you have multiple cache servers, the configuration list of cache addresses for each must be the same. A mismatched configuration shows up in the cache-service.out file.

For example, if two servers have the same list, but a third one doesn't, the log includes messages indicating that the third server has one server but not another, or that a key is expected to be on one server, but is on another instead.

For more information on adding a cache server to a cluster, see Adding cache server machines on page 58. For more information on setting up cache servers for high-availability, see Configuring Cache servers for high-availability on page 15.

Cache server banned under heavy load

Under extreme load, an application server node may be so overwhelmed that it may ban a remote cache server for a small period of time because responses from the cache server are taking too long. If this occurs, you see it in the application log as entries related to the `ThresholdFailureDetector`.

This is usually a transient failure. However, if this continues, you should take steps to reduce the load on the application server to reasonable levels by adding more nodes to the cluster. You might also see this in some situations where a single under-provisioned cache server (for example, a cache server allocated just a single CPU core) is being overwhelmed by caching requests. To remedy this, ensure that the cache server has an adequate number of CPU cores. For more information on hardware requirements, see Cache Server machine.

Banned node can result in near cache mismatches

While the failure of a node doesn't typically cause caching to fail across the cluster (cache data lives in a separate cache server), the banning of an unresponsive node can adversely affect near caches. This shows up as a mismatch visible in the application user interface.

An unresponsive node is removed from the cluster to help ensure that it doesn't disrupt the rest of the application (other nodes will ignore it until it's reinstated). Generally, this situation resolves itself, with the intermediate downside of an increase in database access.

If this happens, recent content lists can become mismatched between nodes in the cluster. That's because near cache changes, which represent the most recent changes, are batched and communicated across the cluster. If the cluster relationship is broken, communication fails between the banned node and other nodes.

After first start up, a node is unable to leave then rejoin the cluster

After the first run of a cluster — the first time you start up all of the nodes — nodes that are banned (due to being unresponsive, for example) might appear not to rejoin the cluster when they become available. That's because when each node registers itself in the database, it also retrieves the list of other nodes from the database. If one of the earlier nodes is the cluster coordinator — responsible for merging a banned cluster node back into the cluster — it is unaware of a problem if the last started node becomes unreachable.

To avoid this problem, after you start every node for the first time, bounce the entire cluster. That way, each is able to read node information about all of the others.

For example, imagine you start nodes A, B, and C in succession for the first time. The database contained no entries for them until you started them. Each enters its address in the database. Node A starts, registering itself. Node B starts, seeing A in the database. Node C starts, seeing A and B. However because node C wasn't in the database when A and B started, they don't know to check on node C — if it becomes unreachable, the won't know and won't inform the cluster coordinator. Note that the coordinator might have changed since startup.

If a node leaves the cluster, the coordinator needs to have the full list at hand to re-merge membership after the node becomes reachable again.

In-memory caching

The in-memory caching system is designed to increase application performance by holding frequently-requested data in memory, reducing the need for database queries to get that data.

The caching system is optimized for use in a clustered installation, where you set up and configure a separate external cache server. In a single-machine installation, the application uses a local cache in the application's server's process, rather than a cache server.

Note: Your license must support clustering in order for you to use an external cache server.

In-memory caching overview

Here you can general information about how in-memory caching works in Jive.

Parts of in-memory caching system

In a clustered installation, caching system components interoperate with the clustering system to provide a fast response to client requests while also ensuring that cached data is available to all nodes in the cluster.

Note: For more on setting up caching in a clustered installation, see Setting up cache server.

Application server	The application manages the relationship between user requests, the near cache, the cache server, and the database.
Near cache	Each application server has its own near cache for the data most recently requested from that cluster node. The near cache is the first place the application looks, followed by the cache server, then the database.
Cache server	The cache server is installed on a machine separate from application server nodes in the cluster. It's available to all nodes in the cluster. Note that you can't create a cluster without declaring the address of a cache server.
Local cache	The local cache exists mainly for single-machine installations, where a cache server might not be present. Like the near cache, it lives with the application server. The local cache should only be used for single-machine installations or for data that should not be available to other nodes in a cluster. An application server's local cache does not participate in synchronization across the cluster.
Clustering system	The clustering system reports near cache changes across the application server nodes. As a result, although data is not fully replicated across nodes, all nodes are aware when the content of their near caches must be updated from the cache server or the database.

How in-memory caching works

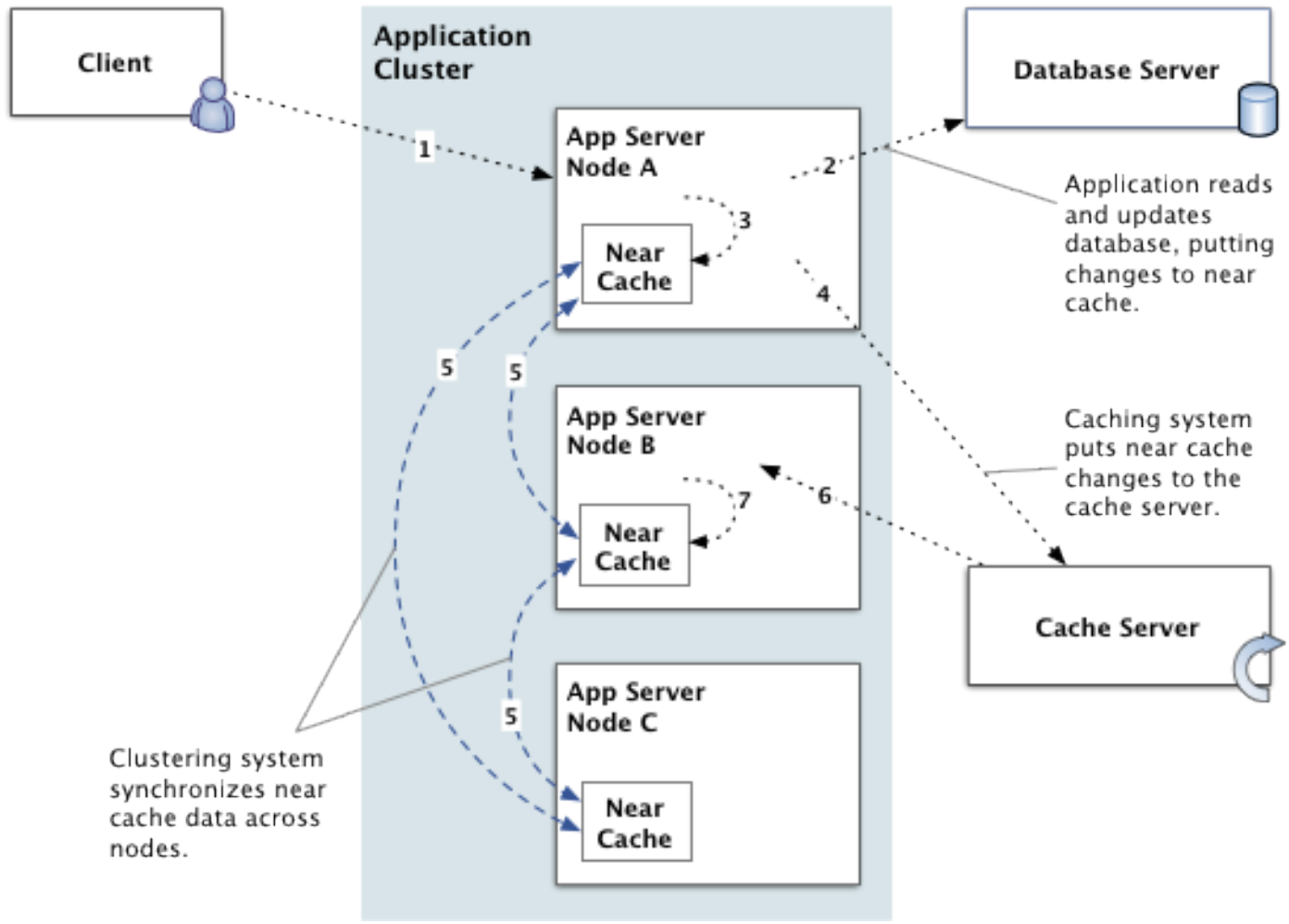
For typical content retrievals, data is returned from the near cache (if the data has been requested recently from the current application server node), from the cache server (if the data has been recently requested from another node in the cluster), or from the database (if the data is not in a cache).

Data retrieved from the database is placed into a cache so that subsequent retrievals are faster.

Here's an example of how changes are handled:

1. Client makes a change, such as an update to a user profile. Their change is made through node A of the cluster, probably via a load balancer.
2. The node A application server writes the change to the application database.
3. The node A app server puts the newly changed data into its near cache for fast retrieval later.
4. The node A app server puts the newly changed data to the cache server, where it will be found by other nodes in the cluster.
5. Node A tells the clustering system that the contents of its near cache have changed, passing along a list of the changed cache items. The clustering system collects change reports and regularly sends them in a batch to other nodes in the cluster. Near caches on the other nodes drop any entries corresponding to those in the change list.

6. When the node B app server receives a request for the data that was changed, and which it has removed from its near cache, it looks to the cache server.
7. Node B caches the fresh data in its own near cache.



Cache server deployment design

In a clustered configuration, the cache server should be installed on a machine separate from the clustered application server nodes. That way, the application server process is not contending for CPU cycles with the cache server process. It is possible to have the application server run with less memory than in a single-machine deployment design. Also, note that it is best if the cache servers and the application servers are located on the same network switch. This helps reduce latency between the application servers and the cache servers.

Note: For more information about the hardware configuration, see System requirements.

Choosing the number of cache server machines

A single dedicated cache server with four cores can easily handle the cache requests from up to six application server nodes running under full load. All cache server processes are monitored by a daemon process which automatically restarts the cache server if the JVM fails completely.

In a cluster, the application continues to run even if all cache servers fail. However, performance degrades significantly because requests previously handled via the cache are transferred to the database, increasing its load significantly.

Adjusting near cache memory

The near cache, which runs on each application server node, starts evicting cached items to free up memory once the heap reaches 75 percent of the maximum allowed size. When you factor in application overhead and free space requirements to allow for efficient garbage collection, a 2GB heap means that the typical amount of memory used for caching will be no greater than about 1GB.

For increased performance (since items cached in the near cache are significantly faster to retrieve than items stored remotely on the cache server) larger sites should increase the amount of memory allocated to the application server process. To see if this is the case, you can watch the GC logs or use other tools, noting if the amount of memory being used never goes below about 70 percent even after garbage collection occurs.

Adjusting cache server memory

The cache server process acts similarly to the near cache. However, it starts eviction once the heap reaches 80 percent of the maximum amount. On installations with large amounts of content, the default 1GB allocated to the cache server process may not be enough and should be increased.

To adjust the amount of memory the cache server process will use, set the `cache.jvm_heap_max` and `cache.jvm_heap_min` values, as shown in the following example:

```
jive set cache.jvm_heap_max 2048
jive set cache.jvm_heap_min 2048
```

Note that you should set the min and the max to the same value — otherwise, evictions may occur prematurely. If you need additional cache server memory, recommended values are the default of 2048 (2GB) or 4096 (4GB). You need to restart the cache server for this change to take effect.

For more information, see Startup property reference and Managing in-memory cache servers on page 57.

Managing in-memory cache servers

Here you can find information on how you can manage the cache server nodes in a cluster. This includes starting and stopping servers, adding and removing nodes, and moving a node.

For more information about installing cache servers in a cluster, see Setting up cache server.

Synchronizing server clocks

Cache servers determine the consistency of cached data between cache servers partially based on the timestamp used when storing and retrieving the data. As a result, all the clocks on all machines (both cache server machines and app server nodes) must be synchronized. It is common to use an NTP daemon on each server synchronized to a common time source. For more information about NTP, refer to The NTP FAQ and HOWTO at http://www.ntp.org/ntpfaq/.

Note that clock synchronization becomes even more important when running within a virtualized environment; some additional steps may be required for proper clock synchronization as outlined in the vendor's documentation. Also, if you're running in a virtualized environment, you must have VMware tools installed in order to counteract clock drift.

Starting and stopping cache servers

You can start and stop cache servers by using the commands described here. Note that all cached data on that machine is lost when its cache server is shut down

---

Note: If you're logged in as root, you can use `su - jive` to become the jive user.

To start a cache server, use the following command as a jive user:

```
jive start cache
```

To stop a cache server, use the following command as a jive user:

```
jive stop cache
```

Adding cache server machines

Adding a cache server to a cluster that has existing cache machines requires additional steps beyond a fresh installation. In particular, you need to shut down the entire cluster (both application and cache servers) before you add a new cache server.

Note: Having multiple cache servers is common only to high-availability configurations. For more information, see Configuring Cache servers for high-availability.

To add a cash server to a cluster that has existing cache machines:

1. Add the new cache server machine as follows:
   a. In the Admin Console, go to **System** > **Settings** > **Caches** .
   b. In **Cache Servers**, add the new cache server machine, then save the settings.
2. Shut down every node in the cluster.
3. Install the new cache server, as described in Setting up cache server.

4. On each of the existing cache machines, set the cache machine addresses by typing `jive set cache.hostnames list_of_hostnames` as a jive user. You can use a comma-separated list of IP addresses or domain names, but be consistent with the format (use IP addresses or domain names, but not both) and order you use.
5. Start up all cache servers before starting the application servers.

Removing cache server machines

Removing a cache server from an existing cluster is very similar to adding one.

Note: Having multiple cache servers is common only to high-availability configurations. For more information, see Configuring Cache servers for high-availability.

To remove a cache server machine from a cluster:

1. Remove the cache server machine from the list as follows:
   a. In the Admin Console, go to **System** > **Settings** > **Caches** .
   b. In **Cache Servers**, remove the cache server machine, then save the settings.
2. Shut down every node in the cluster.
3. On each of the existing cache machines, set the cache machine addresses by typing `jive set cache.hostnames list_of_hostnames` as a jive user. You can use a comma-separated list of IP addresses or domain names, but be consistent with the format (use IP addresses or domain names, but not both) and order you use.
4. Start up all cache servers before starting the application servers.

Moving a cache server to another machine

Moving a cache server from an existing cluster is very similar to adding a machine.

Note: Having multiple cache servers is common only to high-availability configurations. For more information, see Configuring Cache servers for high-availability.

To move a cache server to another machine:

1. Update the list of cache servers as follows:
   a. In the Admin Console, go to **System** > **Settings** > **Caches** .
   b. In **Cache Servers**, change the address for the cache server machine you're going to move, then save the settings.
2. Shut down every node in the cluster.

3. On each of the existing cache machines, set the cache machine addresses by typing `jive set cache.hostnames list_of_hostnames` as a jive user. You can use a comma-separated list of IP addresses or domain names, but be consistent with the format (use IP addresses or domain names, but not both) and order you use.
4. Start up all cache servers before starting the application servers.

Configuring In-Memory Caches

In-memory caching reduces the number of trips the application makes to its database by holding often-requested data in memory. When you configure cache servers, you give each server the list of all cache server machines. For example, you might edit the list of cache server machines when you're adding or removing servers.

For information on adding and removing cache servers, see Managing in-memory cache servers. For information on installing cache servers, see Setting up cache server.

The Caches page in the Admin Console lists the application's caches and provides information on how well they're being used. This information is for use in troubleshooting if you need to call Jive support.

Fastpath: Admin Console: System > **Settings** > **Caches**

Registering cache servers

You must register cache servers in the Admin Console of your instance.

You need to register the cache servers in the **Cache Servers** box of the **Caches** page in the Admin Console.

- In the Admin Console, go to **System** > **Settings** > **Caches** and specify the cache servers in the **Cache Servers** box.

If you have more than one cache server, such as with a high-availability configuration, they must all be listed a comma separated list of either IP addresses or domain names. You must be consistent with the format (use IP addresses or domain names, but don't combine them) and order you use. For more information on setting up cache servers for high-availability, see Configuring Cache servers for high-availability.

For more information about adding, removing, and moving cache servers, see Managing in-memory cache servers.

Getting cache performance information

You can get information about caches in the Admin Console.

In the **Cache Performance Summary** table on the **Caches** page, you can find a list the individual kinds of data cached. Many represent content, such as blog posts and documents. Others represent other data that can be performance-expensive to retrieve from the database.

For each cache, you can find the following information:

Column Name	Description
Cache name	You can click the cache name to view advanced statistics about the cache. You might use these statistics when working with the support team to resolve cache-related issues. General information about the advanced statistics is provided below.
Objects	Each object in the cache represents a different instance of the item. For example, if the Blog cache has 22 objects in it, it means that 22 of the community's blogs are represented there.
Hits	A cache hit is recorded when a query to the cache for the item actually finds it in the cache;
Misses	A cache miss is when the item isn't found in the cache and the query much go to the database instead.
Hit percentage	The effectiveness number — a percentage — is a good single indicator of how well a particular cache is serving your application. When a cache is being cleared often (as might happen if memory constraints are being reached), the ratio of cache hits to misses is lower.
Local cache evictions in the last 30 seconds	The number of times cache was cleared in the last 30 seconds.
Clear cache check box	If you need to clear a cache, select its check box, then click the **Clear Selected** button at the bottom of the cache list table.

Troubleshooting caching and clustering

This topic lists caching- or clustering-related problems that can arise, as well as tools and best practices.

Log files related to caching

If a cache server machine name or IP address is invalid, you get verbose messages on the command line. You also get the messages in log files found in `$JIVE_HOME/var/logs/`.

- cache-gc.log — Output from garbage collection of the cache process.
- cache-service.out — Cache startup messages, output from the cache processes, showing start flags, restarts, and general errors.

Misconfiguration through mismatched cache address lists

If you have multiple cache servers, the configuration list of cache addresses for each must be the same. A mismatched configuration shows up in the cache-service.out file.

For example, if two servers have the same list, but a third one doesn't, the log includes messages indicating that the third server has one server but not another, or that a key is expected to be on one server, but is on another instead.

For more information on adding a cache server to a cluster, see Adding cache server machines on page 58. For more information on setting up cache servers for high-availability, see Configuring Cache servers for high-availability on page 15.

Cache server banned under heavy load

Under extreme load, an application server node may be so overwhelmed that it may ban a remote cache server for a small period of time because responses from the cache server are taking too long. If this occurs, you see it in the application log as entries related to the `ThresholdFailureDetector`.

This is usually a transient failure. However, if this continues, you should take steps to reduce the load on the application server to reasonable levels by adding more nodes to the cluster. You might also see this in some situations where a single under-provisioned cache server (for example, a cache server allocated just a single CPU core) is being overwhelmed by caching requests. To remedy this, ensure that the cache server has an adequate number of CPU cores. For more information on hardware requirements, see Cache Server machine.

Banned node can result in near cache mismatches

While the failure of a node doesn't typically cause caching to fail across the cluster (cache data lives in a separate cache server), the banning of an unresponsive node can adversely affect near caches. This shows up as a mismatch visible in the application user interface.

An unresponsive node is removed from the cluster to help ensure that it doesn't disrupt the rest of the application (other nodes will ignore it until it's reinstated). Generally, this situation resolves itself, with the intermediate downside of an increase in database access.

If this happens, recent content lists can become mismatched between nodes in the cluster. That's because near cache changes, which represent the most recent changes, are batched and communicated across the cluster. If the cluster relationship is broken, communication fails between the banned node and other nodes.

After first start up, a node is unable to leave then rejoin the cluster

After the first run of a cluster — the first time you start up all of the nodes — nodes that are banned (due to being unresponsive, for example) might appear not to rejoin the cluster when they become available. That's because when each node registers itself in the database, it also retrieves the list of other nodes from the database. If one of the earlier nodes is the cluster coordinator — responsible for merging a banned cluster node back into the cluster — it is unaware of a problem if the last started node becomes unreachable.

To avoid this problem, after you start every node for the first time, bounce the entire cluster. That way, each is able to read node information about all of the others.

For example, imagine you start nodes A, B, and C in succession for the first time. The database contained no entries for them until you started them. Each enters its address in the database. Node A starts, registering itself. Node B starts, seeing A in the database. Node C starts, seeing A and B. However because node C wasn't in the database when A and B started, they don't know to check on node C — if it becomes unreachable, the won't know and won't inform the cluster coordinator. Note that the coordinator might have changed since startup.

If a node leaves the cluster, the coordinator needs to have the full list at hand to re-merge membership after the node becomes reachable again.

Monitoring your Jive environment

You should set up your monitoring systems so that you're alerted before things go wrong.

We strongly recommend that your system administrators set up monitoring systems for Jive platforms that are deployed on-premise. (Monitoring for hosted customers is performed automatically by Jive Software).

Monitoring the health of the nodes in your Jive deployment and setting up system alerts can help you avoid costly community downtime, and can also be helpful for correctly sizing the hardware of your deployment. For more information on how to properly size your community, see Deployment sizing and capacity planning.

Basic monitoring recommendations

Here are some monitoring recommendations that are relatively easy to implement.

Consider monitoring the following items using a monitoring tool, such as check_MK, Zenoss, Zyrion, IBM/Tivoli, or other monitoring tools. Polling intervals should be every five minutes.

Caution: If you are connecting Jive to other resources, such as an LDAP server, SSO system, SharePoint, or Netapp storage, we strongly recommend setting up monitoring on these external and shared resources. Most importantly, if you have configured Jive to synchronize against an LDAP server, or if you have configured Jive to authenticate against an SSO, we strongly recommend that you configure monitoring and alerting on that external resource so that you can properly troubleshoot login issues. Note that we see outages related to the LDAP server not being available in our hosted customer environments.

Node	What you should monitor	Why you should monitor it
On all nodes	• Memory utilization • CPU load • Disk space • Disk I/O activity • Network traffic • Clock accuracy	These checks help you monitor all the basics and should be useful for troubleshooting. We recommend performing each of the following checks every five minutes on each server. • Memory utilization: If your memory utilization is consistently near 75%, consider increasing the memory. • CPU load: On healthy web application nodes, we typically see CPU load between 0 and 10 (with 10 being high). In your environment, if the CPU load is consistently above 5, you may want to get some thread dumps by using the `jive snap` command. Then you need to open a support case with Support. For more information, see Getting started with the new Jive Support Portal on Worx. • Disk space: On the web application nodes, you need enough disk space for search indexes (which can grow large over time) and for attachment, image, and binary content caching. The default limit for the binstore cache is 512MB, you can configure it at **Admin console: SystemSettingsStorage Provider**). We recommend starting with 512MB for the binstore cache. Note that you also need space for generated static resources. • Network traffic: While you may not need a specific alert for this, monitoring this is helpful for collecting datapoints. This monitor can be helpful for understanding when traffic dropped off. • Clock accuracy: In clustered deployments, ensuring the clocks are accurate between web application nodes is critical. We strongly recommend using NTP to keep all of the server clocks in sync.
Jive web applications		

Node	What you should monitor	Why you should monitor it
	We recommend running a synthetic health check against your Jive application (by using a tool such as WebInject). • Individual webapplication server • Through the load balancer's virtual IP address	WebInject interacts with the web application to verify basic functionality. It provides functional tests beyond just connecting to a listening port. Checking individual servers, as well as the load balancer instance, verifies proper load balancer behavior. We recommend setting these checks every five minutes initially. To minimize false alarms, we require two failures before an alert is sent. If you find that these settings are resulting in too many false alarms, then adjust your settings accordingly. We recommend setting up WebInject tests that perform the following: • Request the Admin Console login page. This verifies that Apache and Tomcat are running. • Log in to the Admin Console. This verifies that the web application node can communicate with the database server. • Request the front-end homepage. This verifies at a high level that everything is okay. For an example of WebInject XML code that performs all of the above, see WebInject code example on page 71.

Node	What you should monitor	Why you should monitor it
Cache server	• Java Management Extensions (JMX) hooks (heap) • Disk space (logs)	JMX provides a means of checking the Java Virtual Machine's heap size for excessive garbage collection. Disk space checks ensure continued logging. • Heap: If your heap is consistently near 75%, consider increasing the heap size. For more information, see Adjusting Java Virtual Machine (JVM) settings on page 86.

Databases (Activity Engine, Analytics, and web application)	• Stats for:
	  • Connections
	  • Transactions
	  • Longest query time and slow queries
	• Verify ETLs are running
	• Disk space
	• Disk I/O activity	

Node	What you should monitor	Why you should monitor it
		Database checks show potential problems in the web application server which can consume resources at the database layer (such as excessive open connections to the database). • Connections: More connections require more memory. If you need to increase the maximum number of connections allowed by the Jive installation to the core database, consider adding more memory to the database server while ensuring that the database server has enough memory to handle the database connections. The maximum number of connections to the core database is the maximum number of connections allowed for each web application node times the number of webapp nodes in the Jive installation (For more information, see Getting basic system information). Out-of-the-box settings for the core database connections are 25 minimum, 50 maximum. For high-traffic sites in our hosted environment, we set the core database to 25 and 125. Note that additional nodes should be used instead of more database connections for managing additional traffic. Default connection settings for the analytics database are 1 minimum and 15 maximum (you may need to adjust this based on usage and load). For the activity engine database the defaults are 1 minimum and 50 maximum. • Transactions: If the database provides an easy way to measure this number, it can be helpful for understanding the overall traffic volume. However, this metric is less important than monitoring the CPU, memory, and IO utilization for capacity planning and alerting. • Longest query time and slow queries: It's helpful to monitor slow query logs for the database server that they're provisioned against. In our hosted (PostgreSQL) deployments, we log all slow queries (queries that take more than 1000ms seconds) to a file and then monitor those to help find any queries that might be causing issues that could be helped by database indexes. • Verify ETLs are running: This is important only for the Analytics database. The easiest way to monitor this is by querying the `jivedw_etl_job` table with something

Node	What you should monitor	Why you should monitor it
		like this: `select state, start_ts, end_ts from jivedw_etl_job where etl_job_id = (select max(etl_job_id) from jivedw_etl_job);` If the state is `1`, the ETL is running. If any state is `3`, there is a hard failure that you need to investigate. If the difference between `start_ts` and `end_ts` is too big, you may need to increase the resources for the Analytics database. • Disk space: On the web application nodes, you need enough disk space for search indexes (which can grow large over time) and for attachment, image, and binary content caching. The default limit for the binstore cache is 512MB; you can configure it at **Admin console: System** > **Settings** > **Storage Provider** . We recommend starting with 512MB for the binstore cache. Note that you also need space for generated static resources. The most critical place to monitor disk space is on the database server; you should never have less than 50% of your disk available. We recommend setting an alert if you reach more than 50% disk utilization on the database server. • Disk I/O activity: This is good to record because it can be important if you see slow performance on the web application nodes and excessive wait time.
Document conversion		The various service statistics are exposed via JMX's mbean and can be accessed the same way as JMX on the web application node's Tomcat's Java Virtual Machine.

Node	What you should monitor	Why you should monitor it
	• Tomcat I/O • Heap • Queue statistics (for example, average length and wait times) • Running OpenOffice service statistics • Overall conversion success rate for each conversion step	
Activity Engine	• Activity Engine service • Java Management Extensions (JMX) hooks (heap) and ports • Queue statistics (e.g., average length and wait times)	JMX provides a means of checking the Java Virtual Machine's heap size for excessive garbage collection. Disk space checks ensure continued logging. • Heap: If your heap is consistently near 75%, consider increasing the heap size. For more information, see Adjusting Java Virtual Machine (JVM) settings on page 86. • For more information about the queue depths for the Activity Engine, see Configuring Activity Engine.

Jive logs

Jive provides logs that gather application and service information, warnings, and

errors that can help you troubleshoot issues.

By default, your logs can be found in `<Jive installation directory>/var/logs`. You can change the log directory by setting `main.log_dir`:

```
jive set main.log_dir
```

Using `logrotate` script

You can find a `logrotate` script at `<Jive installation directory>/sbin/logrotate`. This script cleans up old gc log files and runs the logrotate tool with configuration from `<Jive installation directory>/etc/conf/logrotate.conf`. The RPM installation creates a symlink in `/etc/cron.hourly` to the logrotate script so that it is executed each hour.

Note: If your Jive installation directory is not the default `/usr/local/jive` or you have modified the `main.log_dir` startup property, you need to modify the logrotate script so that it references the actual installation directory.

WebInject code example

Here is an example of XML code for WebInject for performing several basic checks on a web application node.

Note: For more information about monitoring, see Monitoring your Jive environment on page 63.

This script is designed to perform the following checks on a web application node:

- Request the Admin Console login page (`case id="1"`). This verifies that Apache and Tomcat are running.
- Log in to the Admin Console (`case id="2"`). This verifies that the web application node can communicate with the database server.
- Request the front-end homepage (`case id="3"`). This verifies at a high level that everything is okay.
- Request the index page (`case id="4"`).

In addition, consider monitoring the time it takes this check to run and set an alert threshold at N seconds to ensure this check succeeds in a timely manner.

```
<testcases repeat="1">
<testvar varname="BASEURL">http://my-jive-instance.my-domain.com:80</testvar>
<testvar varname="LOGIN">admin</testvar>
<testvar varname="PASSWORD">admin-password\</testvar>

<case
   id="1"
   description1="Hit main page"
   description2="Verify 'SBS' exists on page"
   method="get"
   url="${BASEURL}/admin/login.jsp?url=main.jsp"
   verifypositive="SBS"
```

```
/>

<case
   id="2"
   description1="Log in as admin user"
   description2="Follow redirect"
   method="post"
   url="${BASEURL}/admin/admin_login"
   postbody="url=main.jsp&login=false&username=${LOGIN}&password=${PASSWORD}"
   verifyresponsecode="302"
   parseresponse="Location:|\n"
/>

<case
   id="3"
   description1="Get main.jsp"
   description2="Check for 'System'"
   method="get"
   url="{PARSEDRESULT}"
   verifypositive="System"
/>

<case
   id="4"
   description1="Get index.jspa"
   description2="Check for 'Welcome'"
   method="get"
   url="${BASEURL}/index.jspa"
   verifypositive="Welcome|Location: ${BASEURL}/wizard-step\!input.jspa|Location:
 .*/terms-and-conditions\!input.jspa"
/>

</testcases>
```

Advanced monitoring recommendations

These advanced monitoring recommendations require intermediate experience with monitoring systems.

Consider monitoring the following items using a monitoring tool, such as check_MK, Zenoss, Zyrion, IBM/Tivoli, or other monitoring tools. Polling intervals should be every five minutes.

Caution: If you are connecting Jive to other resources, such as an LDAP server, SSO system, SharePoint, or Netapp storage, we strongly recommend setting up monitoring on these external and shared resources. Most importantly, if you have configured Jive to synchronize against an LDAP server, or if you have configured Jive to authenticate against an SSO, we strongly recommend that you configure monitoring and alerting on that external resource so that you can properly troubleshoot login issues. Note that we see outages related to the LDAP server not being available in our hosted customer environments.

JMX data points

Node	Data type	JMX object name	JMX attribute name	Data point
Jive web applications	JVM heap memory	java.lang:type=Memory	HeapMemoryUsage	max
	JVM heap memory	java.lang:type=Memory	HeapMemoryUsage	used
	Voldemort cache average operation time	voldemort.store.stats.aggregate:type=aggregate-perf	averageOperationTimeInMs	milliseconds
	Voldemort cache average operation time	voldemort.store.stats.aggregate:type=aggregate-perf	averageOperationTimeInMs	milliseconds
Cache server	JVM heap memory	java.lang:type=Memory	HeapMemoryUsage	max
	JVM heap memory	java.lang:type=Memory	HeapMemoryUsage	used
Activity Engine	JVM heap memory	java.lang:type=Memory	HeapMemoryUsage	max
	JVM heap memory	java.lang:type=Memory	HeapMemoryUsage	used

PostgreSQL data points

We collect the PostegreSQL data points for the core application database and the Activity Engine database. You may choose to also collect these data points for the Analytics database; we do not do this at Jive Software.

Query method	Type	Data points
`poll_postgres.py` script	Connections	`Total`, `Active`, `Idle`
This script makes one query to the database. The query returns all of the following data points at once.	Locks	`Total`, `Granted`, `Waiting`, `Exclusive`, `Access Exclusive`
	Latencies	`Connection latency`,`SELECT Query latency`
	Tuple Rates	`Returned`, `Fetched`, `Inserted`, `Updated`, `Deleted`

Operations cookbook

This section is intended to provide sample configurations and script examples common to long-term operation of a Jive installation. These operations are common to a new installation, but generally not for the day-to-day operation of the platform.

Configuring SSL on load balancer

Configuring SSL termination at the load balancer, which is required, involves configuring your load balancer pool with your SSL certificate information and the addresses of your web app nodes, then ensuring your `JiveURL` property matches the load balancer.

This procedure describes how to configure SSL termination at the load balancer, which is required to effectively secure your installation. Running the Jive site behind a load balancer allows you to operate your Jive web application nodes on a separate, non-public network. For this reason most customers find it sufficient to terminate SSL at the load balancer and proxy http connections to the web application nodes. For information on how to also configure SSL encryption between your load balancer and each web application node, see Configuring SSL between load balancer and web app nodes on page 75.

Note: To ensure consistent results, you should enable SSL for your UAT environment as well as your production instance of Jive. To properly test and implement SSL, you need certificates for `community.yourdomain.com` (Production) as well as `community-uat.yourdomain.com` and `apps.community-uat.yourdomain.com` (UAT). If you're a hosted customer, you can contact Support instead of using the steps below to apply the certificates. For more information about Apps subdomain security, see Creating secure subdomains for apps.

To configure SSL termination at the load balancer:

1. Configure your load balancer pool to use the SSL certificates you've acquired for your sites.
2. Create a DNS record for each domain that resolves to your load balancer pool's IP address.
3. Add all of your site's web application node addresses and ports to the balancer pool. For example, add:

```
http://myapp-wa01.internal.mycompany.com:8080
http://myapp-wa02.internal.mycompany.com:8080
http://myapp-wa03.internal.mycompany.com:8080
```

4. On each of the webapp nodes, set the required proxy-related properties and restart. For example:

```
jive set webapp.http_proxy_name community.mycompany.com
jive set webapp.http_proxy_port 443
jive set webapp.http_proxy_scheme https
```

5. Make sure that the `jiveURL` property in Jive's core database is set to the address of the load balancer by going to **System** > **Management** > **System Properties** and checking the setting of the `jiveURL` system property.
6. Restart Jive on all the web application nodes.

Configuring SSL between load balancer and web app nodes

Configuring SSL encryption between your load balancer and each web application node is not required, but if you plan to do it, you need to acquire an SSL certificate for each node.

To set up SSL encryption to each node:

1. On each webapp node, enable SSL by assigning the following startup properties:

```
jive set httpd.ssl_enabled True
jive set httpd.ssl_certificate_file /path/to/your/crt/file
jive set httpd.ssl_certificate_key_file /path/to/your/key/file
```

2. Change your load balancer pool's members to reflect the new SSL port. For example:

```
https://myapp-wa01.internal.mycompany.com:8443
https://myapp-wa02.internal.mycompany.com:8443
https://myapp-wa03.internal.mycompany.com:8443
```

3. Restart httpd on all the web application nodes.

Configuring session affinity on load balancer

Jive requires session affinity to be configured on your load balancer.

Session affinity on the load balancer is required. For an F5 BigIP load balancer, you can use a default cookie persistence profile. For more information, see the recommended F5 settings in the F5 documentation. If you have another type of load balancer, which doesn't create its own cookies for session affinity, you can use the JSESSIONID cookie that Jive sets. For more information, see the Apache HTTPD documentation for examples.

To configure session affinity:

1. Set a route string for each balancer member in your load balancer configuration. For example, use the string `node01` in the balancer pool configuration for `myapp-wa01.internal.mycompany.com`.
2. Set the corresponding startup property on that web application node. If you used the route string `node01`, you might set:

   ```
   jive set webapp.app_cluster_jvmroute node01
   ```

3. Follow the same pattern for your other web application nodes.
4. Restart the web application on all the web application nodes.

Restricting Admin Console access by IP address

You can secure the Admin Console by allowing or denying specific IP addresses.

To specify who can access the Admin Console based on IP address:

1. Locate the `/usr/local/jive/etc/httpd/sites/default.conf` file.
2. Allow or deny IP addresses by adding and modifying the following code:

```
<Location /admin>
    Order Deny,Allow
    Allow from <IP ADDRESS>
    Allow from <IP ADDRESS>
    Allow from <IP ADDRESS>
    Allow from <IP ADDRESS>
    Allow from <IP ADDRESS>
    Deny from all
    </Location>
```

Changing configuration of existing instances

When you change the configuration of an existing instance, you should update environment variables in your `/usr/local/jive/applications/app_name/bin/instance` file to reflect new configuration settings.

In some circumstances, it may be desirable to change the default configuration of platform-managed application server instances. For example, on a larger server-class machine, an application instance benefits from allocation of more RAM for the JVM heap.

To change this or other settings, edit the `instance` file for the desired application (`sbs` by default) located at `/usr/local/jive/applications/`*`app_name`*`/bin/instance`.

The contents of this file vary from release to release. Generally, the entries in this file correspond to either:

- Environment variable values in the `setenv` script located in the same directory
- Tokenized configuration attributes for the `conf/server.xml` file in the application directory

For any managed application, all files except the binaries for the web application (by default, each application is linked to these binaries located at `/usr/local/jive/applications/template/application`) are not managed by the application platform. As a result, any changes to files, such as for `instance`, are durable across application upgrades.

Changing ports

As an example, to change the port that the managed application listens for AJP connections, edit the `instance` file to alter the port for `AJP_PORT`.

Prior to edit, the `instance` file looks similar to the following.

```
[0806][jive@melina:~/applications/sbs/bin]$ cat instance
export JIVE_HOME="/usr/local/jive"
export AJP_PORT="9002"
export APP_CLUSTER_ADDR="224.224.224.224"
export JIVE_APP_CACHE_TTL="10000"
export APP_CLUSTER_PORT="9003"
export HTTPD_ADDR="0.0.0.0"
export AJP_BUFFER_SIZE="4096"
export HTTP_ADDR="127.0.0.1"
export JIVE_APP_CACHE_SIZE="10240"
export SERVER_PORT="9000"
export JIVE_NAME="sbs"
export HTTP_PORT="9001"
export AJP_ADDR="127.0.0.1"
export JIVE_CONTEXT=""
export AJP_THREADS_MAX="50"
```

To alter the `AJP_PORT` to listen on port 11000, edit the `instance` file to appear similar to the following:

```
[0806][jive@melina:~/applications/sbs/bin]$ cat instance
export JIVE_HOME="/usr/local/jive"
export AJP_PORT="11000"
export APP_CLUSTER_ADDR="224.224.224.224"
export JIVE_APP_CACHE_TTL="10000"
export APP_CLUSTER_PORT="9003"
export HTTPD_ADDR="0.0.0.0"
export AJP_BUFFER_SIZE="4096"
export HTTP_ADDR="127.0.0.1"
export JIVE_APP_CACHE_SIZE="10240"
export SERVER_PORT="9000"
export JIVE_NAME="sbs"
export HTTP_PORT="9001"
export AJP_ADDR="127.0.0.1"
export JIVE_CONTEXT=""
export AJP_THREADS_MAX="50"
```

Changing heap min and max values

For more information about changing the JVM min and max values, see Adjusting Java Virtual Machine (JVM) settings on page 86.

Configuring JVM route name of nodes

To configure the route name of your web application nodes, add a lines to the `instance` file in `/usr/local/jive/applications/<app_name>/bin` as follows, where `node01` is your desired route name:

```
export APP_CLUSTER_JVMROUTE="node01"
```

When configuring multiple nodes with `jvmRoute` attributes, each node should have a different value.

Using external load balancer

In order to integrate the Jive platform with an external load balancer, you configure the load balancer for cookie-based session affinity between each host running the platform.

For more information on setting up cookie-based session affinity, see Configuring session affinity on load balancer on page 76. Note that all Jive's testing of load balancers is cookie-based. As of Jive 7, the load balancer is required to perform SSL session termination as described in Configuring SSL on load balancer on page 74. You may also wish to configure SSL encryption between the load balancer and each web application node. For more information, see Configuring SSL Between a Load Balancer and Web App Nodes.

Depending on the load balancer, it may be necessary to add JVM route information to the outgoing JSESSIONID HTTP cookies sent to remote agents. For more information about using Apache HTTPD as a load balancer, see Apache's documentation about load balancer stickiness at http://httpd.apache.org/docs/2.2/mod/mod_proxy_balancer.html#stickyness_implementation. For more information on how to configure the route name (`jvmRoute` variable) of your nodes in Jive, see Configuring JVM route name of nodes on page 78.

Some load balancers require a "magic" HTML file in the site root to make the node available. If your load balancer requires this, add the following line to this default configuration file `/usr/local/jive/etc/httpd/sites/default.conf`:

```
ProxyPass /magicfile.html !
```

For more information about Apache's ProxyPass and how it works, see Apache Module mod_proxy at http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxypass.

Enabling application debugger support

The Jive web application is capable of accepting remote Java debuggers. To enable debugging, you set the necessary additional Java arguments before starting the managed application to be debugged.

: You should not run this operation on a production site.

To enable application debugging:

1. Run `jive list webapp.custom_jvm_args` to check whether you have an override value already set for `webapp.custom_jvm_args`.
2. If you have not set a value for this property, run the following command:
   ```
   jive set webapp.custom_jvm_args " -Xdebug
   -Xrunjdwp:transport=dt_socket,address=9090,suspend=n,server=y"
   ```
3. If you already have a value for this property, run the following command:
   ```
    jive set webapp.custom_jvm_args " PREVIOUS_VALUE_HERE -Xdebug
   -Xrunjdwp:transport=dt_socket,address=9090,suspend=n,server=y"
   ```
4. To apply your changes, run `jive restart webapp`.

Setting up Document Conversion

Some documents — including PDFs and some of the Office documents — are supported in a preview view in Jive. To convert content from its native format into a form that can be previewed without altering the original document, you need the Document Conversion module. This module you deploy on a server that is separate from your core Jive production instances.

Jive gives users the ability to upload Office and Adobe PDF documents to the community for easy content sharing and collaboration. The Document Conversion service converts uploaded documents to a standard PDF format and then converts them again to Adobe Flash (.swf files) so that they can then be viewed in a web browser without needing to open the document's native software application.

We support converting the following file types:

- doc
- ppt
- docx
- pptx
- xls
- xlsx
- pdf

Note: For more information about managing conversion attempts and reconverting documents if necessary, see Managing Document Conversion.

Here is an overview of the steps you perform to set up Document Conversion:

1. Set up a production instance of the Jive application. You should devote one node in your installation to document conversion. For more information, see Installing Jive package and starting up.
2. Install the Jive platform RPM on your conversion node machine, as described in Setting up conversion machine. Then disable the services not related to document conversion. Download and install the correct RPM for the PDF2SWF utility on the conversion node machine. You can find the RPMs at https://static.jiveon.com/doc-converter.

   Note: If you receive the following error when installing either the `jive_sbs` rpm or `jive_pdf2swf`, use the `--replacefiles` flag on the document conversion node.

   ```
   file /usr/local/jive/bin/pdf2swf from install of rpm_name conflicts with file
   from package other_rpm_name the flag
   ```

3. Enable the Document Conversion service by using the following commands:

   ```
   jive enable docconverter
   jive start
   ```

4. On the application node, configure the application to communicate with the conversion machines, as described in Configuring Document Conversion node connection.
5. If you want to set up secure communication to the conversion machine, see Setting up SSL for Document Conversion.

Setting up SSL for Document Conversion

If you have an SSL certificate, you can set up secure communication by editing the `docconverter/conf/server.xml` file and specifying the new secure URL in your Document Conversion Settings.

Before you can set up secure communication with your Document Conversion server, you need to acquire an SSL certificate.

To add an SSL certificate to your instance:

1. Edit the `/usr/local/jive/services/docconverter/conf/server.xml` file and add a connector to listen on port `8443`.

   For example:

   ```
   <Connector port="8443" maxThreads="200" scheme="https" secure="true"
   SSLEnabled="true" "SSLCertificateFile="/usr/local/jive/services/docconvert-
   er/home/jive.crt" SSLCertificateKeyFile="/usr/local/jive/services/doc-
   converter/home/jive.key" clientAuth="optional" />
   ```

where `SSLCertificateFile` is the certificate file and `SSLCertificateKeyFile` is the key file.

For more information on setting up Tomcat and https, see Apach documentation at http://tomcat.apache.org/tomcat-8.0-doc/ssl-howto.html.

2. Make sure the SSL engine is on.

   For example:

   ```
   <Listener className="org.apache.catalina.core.AprLifecycleListener"
   SSLEngine="on"/>
   ```

3. Restart the document conversion service by running the following command as the jive user:

   ```
   jive restart docconverter
   ```

4. Go to **System** > **Settings** > **Document Conversion Settings** and edit **Conversion Service Settings** to specify the new secure URL and port.

   For example:

   ```
   https://conversion-node:8443/conversion/v1
   ```

5. Verify that you can run all conversion tests successfully.

Configuring Document Conversion node connection

You use the Document Conversion Settings page to configure the node that hosts the core Jive application, so the main application knows how to communicate with any conversion machines you've set up. Before you use the settings on this page, you should already have set up a document conversion node. For an overview of Document Conversion setup, see Setting up Document Conversion*.*

Fastpath: > **Admin Console: System** > **Settings** > **Document Conversion**

The Document Conversion Settings page is pre-populated with the default values for most installations. Use the following steps to ensure your setup is correct.

To enable document conversion:

1. Under **Document Conversion**, select **Enabled** to enable this feature.
2. Click **Add a Node** to start configuring a conversion machine.
3. Enter the IP address or hostname for the conversion machine.
4. Switch the way you access the conversion service URL from http to https by setting this in the **Conversion Service Settings** field.
5. If you want to exclude some file types from conversion, enter a comma-separated list in **Disabled Extensions**. For example, if you don't want to convert Excel files, type `xls, xlsx`.

Troubleshooting Document Conversion machine setup

If your document conversion tests are failing, try investigating the areas described here to resolve the problem.

- Check that port `19003` is open between the application node and the conversion node by executing a telnet command. For example, run `telnet 10.61.32.156 19003`.
- Check the log for the DocConverter service for startup exceptions. You can find the log at `/usr/local/jive/var/logs/docconverter.out`.
- If the Office to PDF test is failing on the Document Conversion Setup page, execute `jive status -v` to verify that DocConverter has open ports `8820`, `8821`, `8822`, `8823`, `8824`.

Adding fonts to support Office document preview

You should install your licensed True Types fonts on the Document Conversion server to enable accurate previews of uploaded Office documents.

Note: If you need to use languages such as Chinese, Japanese, Korean, and Arabic in an on-premise installation, you need to install the proper licensed fonts to enable proper text display in document preview mode. Licensing limitations prevent Jive from distributing these fonts with the installation package. (If your Jive community is hosted by Jive, this custom font feature is not supported, but Jive installs language-specific fonts for supported languages.)

To add fonts to support Office document preview:

1. Locate the font packages you want to install.
2. Connect to the Document Conversion server as root.
3. Using the operating system's package manager, install the fonts on the Document Conversion server.

   The fonts should now have been added to fontconfig on your system. You can verify that a particular font is installed and ready to be used by the document conversion service by typing **fc-list** and making sure the font is listed.
4. As root, restart the Document Conversion service (`/etc/init.d/jive-docconverter restart`).

Sharing Exchange calendars in an HTML Text widget

If you are using an Exchange 2010 SP1 or later email server, you can set up a community widget to show user Exchange calendars, with customizable levels of visible calendar details.

Important: We do not recommend that you use widgets and widgetized Overview pages in your community. For more information, see Understanding pages in places.

Caution: Calendar sharing uses Exchange Web Services to make HTML and iCal versions of users calendars available. Depending on your Exchange topology, this can (and will) publish calendar URLs to the Internet, where they could be viewed by anyone. If you want to prevent this, make sure you have a secure firewall in place.

To share an Exchange calendar, first, you set up sharing profiles on the Exchange side, then publish shared calendars in your community.

To set up your Exchange server for sharing:

1. Create a calendar sharing profile.
2. Enable the calendar sharing profile for each user for whom you want to have a visible calendar in the community.

Note: You cannot share calendars contained in public folders. A shared calendar must be a user mailbox.

To publish shared calendars in your community:

1. Ensure that calendar publishing is enabled on your Exchange server. To do this, you can use the following Exchange PowerShell commandlet:

   ```
   Get-OwaVirtualDirectory | ft server, Name, CalendarPublishingEnabled
   ```

2. Enable calendar publishing with the following command:

   ```
   Set-OWAVirtualDirectory "SERVER\owa (Default Web Site)"
   -CalendarPublishingEnabled:$true
   ```

3. From the Exchange Management Shell, create a new calendar sharing profile and enable anonymous shares:

   ```
   New-SharingPolicy -Name "Calendar Sharing Policy"
   ```

4. Set the sharing policy on user mailboxes who wish to share their calendars:

   ```
   Set-Mailbox -Identity User.Mailbox -SharingPolicy "Calendar Sharing Policy"
   ```

5. Notify the target users to share their calendars either via Outlook 2010 or via Outlook Web Access.
6. When the user publishes a shared calendar, copy the full text of the **Link for viewing calendar in a web browser**. This link usually looks like this:

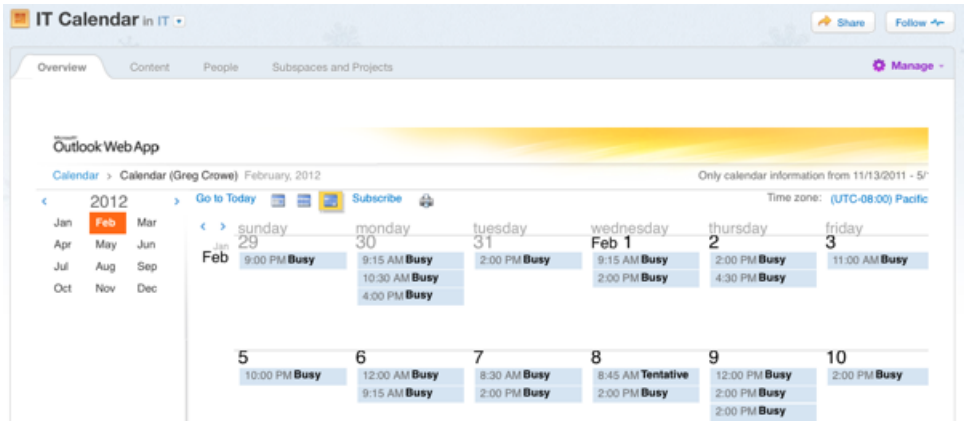
   ```
   https://YOUR.MAIL.SERVER/owa/calendar/GUID@YOURDOMAIN.PUBLIC/DIFFERENT_GUID/calendar.htm
   ```

7. In the community place where you want to share calendars, add an HTML widget.
8. In the widget, include the link you copied in Step 6 in an iframe tag as follows:

   ```
   <iframe
   src="https://YOUR.MAIL.SERVER/owa/calendar/GUID@YOURDOMAIN.PUBLIC/DIFFERENT_GUID/calenda
    width="1200" height="800"></iframe>
   ```

9. Save the HTML tile.
10. Save and publish your changes to the place.

A page with a shared calendar may look like this:



Fine-tuning performance

Through adjustments to caches, JVM settings, and other settings, you can make sure that the application is performing well.

You may want to adjust application settings from their defaults shortly after you're up and running. In particular, you can monitor caching, but there are other things you can do to ensure that the application is performing as well as possible. Here you can find tuning suggestions.

Client-side resource caching

The platform HTTPD server is pre-configured for optimal caching of static production content. Default configuration values for content caching can be found on a Jive-managed server at `/usr/local/jive/etc/httpd/conf.d/cache.conf`. *You can edit these settings to change default cache time or headers for specific scenarios (changing length of time static images are cached, for example). Changes to this file are preserved across upgrades to a file named cache.conf.rpmnew. If this file is changed, you must check for new enhancements when upgrading.*

Note: Certain resources in plugins and themes are cached for 28 days by default. These include the following file types: .js, .css, .gif, .jpeg, .jpg, and .png. This means that clients won't see updated versions of those files until their cache expires or is cleared. Note that changing the resource's file name also causes it to be downloaded because it isn't yet cached.

Configuring external static resource caching

If you're using a lightweight content delivery network (CDN), you can configure the community to tell clients to retrieve static resources from your CDN server. This improves performance by reducing load on the Jive server. You can make this setting in the Admin Console.

Fastpath: Admin Console: System > **Settings** > **Resource Caching**

Configuring external CDN software to retrieve static resources

This feature assumes that you've set up and configured your CDN software to retrieve static resources from the application server when necessary. Here are the basic steps:

1. Set up your CDN, configuring it to be aware of your Jive server.
2. Configure the resource caching feature with the CDN base URL where static resources can be found when requested.
3. At run time, when building pages for a client, Jive rewrites static resource locations so that their URLs point to your CDN server.
4. When completing the page request, the client uses the CDN URL to retrieve static resources.
5. If the CDN server has the resource, it returns it; if not, it retrieves the resource from the Jive server, return it to the client, and cache it for future requests.

Configuring external static resource caching

To configure the feature,

1. In the Admin Console, go to **System** > **Settings** > **Resource Caching** .
2. Select the **Enable external caching** check box.
3. Enter the CDN URL where static resources can be retrieved by clients, and save the settings.

Adjusting Java Virtual Machine (JVM) settings

As with any Java-based web application, you can sometimes improve performance by assigning particular values to the Java Virtual Machine options.

You can edit the JVM minimum and maximum memory settings on a node by editing the values for the `jvm_heap_max` and `jvm_heap_min` variables from the command line. These values are expressed in MB. For example, to set the minimum and maximum heap available on the web application node to 4GB, from the command line interface you would type the following:

```
jive set webapp.jvm_heap_max 4096
jive set webapp.jvm_heap_min 4096
```

The default JVM values for each of the nodes are listed in Startup property reference. The command settings are listed in Startup property commands on page 90. Note that your particular community may need to decrease or increase the default values depending on the size and traffic of your community. For sizing capacity recommendations, see Deployment sizing and capacity planning.

JVM recommendations

Node	Recommendations
Jive Web applications	To ensure that the appropriate resources are available to the running application, we recommend setting the `jvm_heap_min` and `jvm_heap_max` to the same value on the web application nodes. In a clustered environment, these min and max values should be the same for all of the web application nodes. For larger communities, that is, communities that get more than 100,000 page views per day or that contain a large amount of content (more than 100,000 messages, documents, or blog posts), you may need to increase the JVM heap min and max settings to be both 4096 or both 6144.
Additional cluster nodes (if your configuration includes these optional nodes)	These values should match those of the primary web app nodes.
Activity Engine	None.
Cache servers (if your configuration includes these optional nodes)	None.
Document Conversion (if you have this optional module)	We recommend not changing the default settings. They have consistently performed well in all pre-release quality, stress, and performance tests.

Configuring search index rebuild

In rare cases, particularly after a version upgrade and depending on your configuration, you may experience long search index rebuild times. In this case, you may wish to adjust the search index rebuild system properties to increase the limit on the amount of resources used, potentially improving performance.

Fastpath: Admin Console: System > **Management** > **System Properties**

Search index performance can vary greatly depending on the size and number of binary documents and attachments, as well as user activity, in your community. By default, the search index parameters are set to use as few memory and CPU resources as possible during a rebuild. If you experience extremely long search index rebuild times (for example, because your community has created a large amount of content) and you have additional CPU and memory resources to spare, contact Support about adjusting the search index rebuild system properties.

Using Content Distribution tool with Jive

Many of Jive Software's customers rely on a third-party content distribution and content delivery network (CDN) tool to help their Jive pages load faster for

globally-dispersed users. In this section, we describe some best practices for using Jive with these tools.

Note: The application can be configured to work with most CDN tools. While there are a number of hardware appliances that customers use inside their firewall, Jive has found that the majority of on-premise customers choose to deploy behind devices by F5 (http://www.f5.com/).

Recommended settings for F5

In most cases, your Jive configuration should rely on the default settings in F5. However, there are a few settings that Jive Software's hosting engineers commonly customize to optimize hosted Jive deployments.

Generally, we recommend using the default settings in F5 because F5 is already optimized for them and customizations you create may require more processing, and thus, more load.

The following tables list the settings that our hosting engineers typically change in F5. These are general guidelines. Your needs may be different. Contact your Jive Software representative with specific questions.

Table 4: Node configuration

Setting	Description
ICMP Health Monitor	A simple ICMP request (`PING`) to the node to confirm it is online and operating at its most basic level.

Table 5: Pool configuration

Setting	Description
TCP Health Monitor	This is necessary because HTTP does not always show it is down when the Jive application goes into a maintenance mode. We depend on Web Injections via a separate monitoring service to determine whether a node in a pool is operational or not. Therefore, if a TCP connection fails to the port that is specified by the VIP, the node is considered down and removed from the pool. Note that a node is not considered down if the Jive application is down but the service is still running. This is why we use Web Injections to do more appropriate application level up time validation. For more information, see Monitoring your Jive environment.
Load balancing method: Least Connections (node)	This causes the Jive application to load balance based on the number of connections to the node, regardless of whether the connections are related to the pool traffic. Therefore, load is balanced overall between individual nodes.

Table 6: HTTP VIP configuration

Setting	Description
OneConnect /32 profile	This profile is used to accommodate the CDN fronting the Jive application access. This setting allows F5 to properly handle multiple HTTP requests within the same TCP connection, as you would see when using a CDN. For more information, refer to F5's documentation (http://support.f5.com/kb/en-us/solutions/public/7000/900/sol7964.html?sr=11716037).
HTTP Profile (this applies only if you are using F5 VIP's with SNAT).	This is a customized profile based on the parent HTTP profile to insert the true client source IP using either Request Header Insert or Insert X-Forwarded-For. This is for HTTP logging because F5 acts as a reverse proxy to the Jive web application nodes.
Set the SNAT Pool to Auto Map.	F5 acts as a reverse proxy to the Jive web application nodes; the Jive application needs the traffic response from the web application nodes to respond back through F5. This setting isn't required, but we recommend it as a best practice for configuring the F5 in a one-armed mode.
Set the default persistence profile to cookie	This maintains session persistence based on an inserted cookie.
Keep iRules as simple as possible.	At Jive Software, our hosting engineers try to keep iRule use to a minimum because they are evaluated each time traffic passes the VIP to which it is attached. Because this adds processing load, we recommend keeping it simple and adding as few iRules as possible.

Use an iRule or HTTP Class Profile for redirect from HTTP to HTTPS.	To keep processing to a minimum, we recommend using the configuration options built into F5 rather than iRules to accomplish HTTP to HTTPS redirects. However, note that using an HTTP Class Profile for redirects uses a 302 redirect (Temporary), not a 301 redirect (Permanent). This may cause problems with your configuration. If this is acceptable for you, then you can use an HTTP Class Profile to accomplish your redirect; otherwise, you need to use an iRule. Here is an example of each:
	• iRule:
	`when HTTP_REQUEST {`
	`HTTP::respond 301 Location "https://[HTTP::host][HTTP::uri]" }`
	• HTTP Class Profile: use the Send To option and select **Redirect To**. Then, in the Redirect to Location, set it to https://[HTTP::host][HTTP::uri]

Table 7: HTTPS VIP configuration

Setting	Description
Set everything the same as above in HTTP VIP Configuration, except the following:	
Use the default HTTP Profile (this applies only if you are using F5 VIP's with SNAT).	The HTTP profile cannot be used to insert the true client source IP into the header of an HTTPS connection. This must be done by using an iRule for HTTPS traffic. Here is a simple example: `when HTTP_REQUEST { HTTP::header insert JiveClientIP [IP::remote_addr] }`
Set the Client SSL Profile to cover your SSL certificate, key, and chain.	We recommend leaving everything else as the default parent profile of clientssl. You may want to consider removing the renegotiation option from the parent clientssl profile for security reasons. **Attention:** Note that there is a potential DoS risk here. For more information, see https://community.qualys.com/blogs/securitylabs/2011/10/31/tls-renegotiation-and-denial-of-service-attacks.

Application management command reference

You can use the `jive` command to perform tasks on your instance.

The `jive` command is located in `<yourjivehome>/python/bin`. This path is automatically added to your `$PATH` variable by .bash_profile.

You can run `jive --help` to see a full list of available commands. You can use the commands is as follows:

```
jive [-h] [--version]
[command{start,stop,restart,status,enable,disable,list,set,del,doc,setup,snap}]
```

Startup property commands

Here you can find the commands you can use with any of the startup properties. These commands enable you to set, list, or delete startup properties on any of the nodes in the configuration.

jive doc	Shows Help for the most commonly modified startup properties.
jive list	Shows all the startup properties you have overridden.
jive list [substring_match]	Lists properties matching the specified substring.
jive list -p	Shows properties in a props file format that you can easily parse with scripts.
jive list -v	Shows all of the available startup properties.

jive set [property_name] [prop_value]	Overrides the existing value of the specified startup property with the new value.
jive del [property_name]	Removes the override for the specified property so that the default value will be used.

For more information on startup properties, see Startup property reference.

Executing startup property commands

These commands must be executed as the jive user.

For example, if you've got ssh access as root to your host machine, use the following command to switch to the jive user:

```
sudo su - jive
```

Services properties commands

Here you can find the commands you can use with any of the services. These commands enable you to check the status of any of the services, as well as stop, start, or restart them.

jive status	Shows all of t service their runnin status, and th enable status.
jive status -v	Shows the po on whi the service are listeni
jive enable [servicename{cache,docconverter,eae,ingress-replicator,search,webapp,httpd}]	Enable the specifi service so tha starts when you ru `jive start`.
jive disable [servicename{cache,docconverter,eae,ingress-replicator,search,webapp,httpd}]	Disabl the specifi service

jive start	Starts enable service
jive start [servicename{cache,docconverter,eae,ingress-replicator,search,webapp,httpd}]	Starts the on specifi service
jive stop	Stops runnin service
jive stop [servicename{cache,docconverter,eae,ingress-replicator,search,webapp,httpd}]	Stops the on specifi service
jive restart	Restar all runnin service
jive restart [servicename{cache,docconverter,eae,ingress-replicator,search,webapp,httpd}]	Restar the on specifi service
jive restart --graceful httpd	Perfor a gracef restart the htt service
jive snap	Takes system or service snapsh

Executing service property commands

These commands must be executed as the jive user.

For example, if you've got ssh access as root to your host machine, use the following command to switch to the jive user:

```
sudo su - jive
```