

Administering the Platform

jive

Contents

Administering the Platform	2
Jive Platform Overview	2
System Components.....	2
Platform Conventions, Management and Tools.....	3
What Happened to jiveHome?.....	5
Platform Run Book (Linux)	6
Jive HTTPD.....	6
Jive HTTPD Networking.....	7
Jive HTTPD Log Files.....	8
Jive-Managed Applications.....	8
Jive Database Server.....	10
Operations Cookbook	11
Enabling SSL Encryption.....	12
Disabling the Local Jive System Database.....	12
Changing the Configuration of an Existing Instance.....	12
Performing a Jive System Database Backup.....	13
Backup and Storage Considerations.....	14
Using an External Load Balancer.....	15
Enable Application Debugger Support.....	15
Setting Up a Local Cluster.....	15
Configuration Support for External Client Access.....	16
Adding Fonts to Support Office Document Preview.....	18
Fine-Tuning Performance	19
Client-Side Resource Caching.....	19
Server-Side Page Caching.....	19
Configuring External Static Resource Caching.....	20
Performance Recommendations for High Traffic Sites.....	20
Adjusting Java Virtual Machine (JVM) Settings.....	21
Assorted Performance Tuning Tips.....	21
Application Management Command Reference	21
appadd Command.....	21
appls Command.....	23
apprm Command.....	24
appsnap Command.....	24
appstart Command.....	25
appstop Command.....	25
appsupport Command.....	25
dbbackup Command.....	26
manage Command.....	27
upgrade Command.....	27

Administering the Platform

This guide includes information on administering the platform, including run books, configuration guides, and performance tuning help.

Jive Platform Overview

The platform introduced in version 3 includes important changes that affect the way you install and administer the application. This topic provides an overview of those changes, the platform, and the tools that come with it.

Through a design that supports a more focused set of components (in particular, application server and database servers), the platform provides a standard, reliable, and better-integrated environment that makes it easier to optimize the applications deployed on it. The platform -- especially its deployment as an package -- also makes installation much easier by removing the need to follow instructions (including component-specific issues and limitations) specific to particular combinations of application server, DBMS, and so on.

Note: If you're upgrading, Jive Software provides tools for migrating your existing deployment. Be sure to see the Installation Guide for more on *Migrating Data from an Unsupported DBMS* and the contents of your jiveHome directory.

Here's a high-level list of the things that are new or changed with the platform's introduction:

- [System Components](#) (page 2) were chosen to create an integrated, standard platform that can be more fully optimized.
- Database server support was tuned to the PostgreSQL and Oracle DBMSes.
- Application server support is optimized to the Apache Tomcat instance included with the platform. Other application servers aren't supported.
- Supported operating systems include Red Hat and SuSE Enterprise Linux.
- Tools were included to make managing the application easier and more reliable.
- The jiveHome directory became the jive.instance.home environment variable.

Related Content

You might be interested in other documentation related to the platform. The following topics include references and guides for making the most of the platform and applications installed on it.

Document	Description
Platform Run Book (Linux) (page 6) ,)	Basic system commands for managing the platform. If you need to handle something quickly, start here.
Operations Cookbook (page 11)	Sample configurations and script examples for long-term operations.
Application Management Commands (page 21)	A reference for commands you can use to maintain your managed instance.
<i>System Requirements</i>	System components and technologies supported in this release.
<i>Installing and Upgrading</i>	Step by step instructions for installing the platform.

System Components

The Jive SBS platform consists of several high-level components that work together to provide overall system functionality. The following sections provide an overview of these components and their role in the architecture. By default, the platform will install the following components to start and stop in the correct order during system startup and shutdown.

Apache HTTPD Server

The platform contains a customized version of the Apache Foundation's *Apache HTTPD web server* software. Among other things, this software is used to process HTTP and HTTPS-encrypted requests between the platform and end users. The Apache HTTPD server uses the JK protocol to communicate with the back-end application server described in the next section.

Tomcat Application Server

The *Apache Tomcat application server* is used to host back-end application and business logic as well as to generate dynamic HTTP traffic served by the Apache HTTPD Server. The application server is also responsible for processing email delivered by the system, and for scheduling background tasks such as search index management and integration with third-party applications and systems.

PostgreSQL Database Server

The *PostgreSQL database server* is an RDBMS (Relational Database Management Server) service that is hosted internally within the platform. You can use this service or a PostgreSQL or Oracle system of your own.

Jive System Configuration

As part of the platform installation, the Jive SBS package will tune various settings such as shared memory and maximum file handles. The details of these changes can be found in the "jive-system" script, located in the standard "/etc/init.d" directory.

Platform Conventions, Management and Tools

The "jive" User

By default, Jive SBS will attempt to install a local system user named "jive" belonging to group "jive" with a home directory of "/usr/local/jive". If a user named jive already exists on the system where the installation is being performed, the platform will use the existing user. Most binaries, scripts and configurations used by the platform are owned by the jive user with a small number of files owned by root.

Application Abstractions

The Jive SBS platform is designed to manage one or more logical "application" servers, each serving a variety of functional concerns.

Master Application Template:

The master application template is the core set of Jive software used to create all subsequent instances. By convention, the files in the master template are stored in the jive user's "applications" directory located at:

```
/usr/local/jive/applications/template
```

Application Instances:

Upon installation, and at the request of system administrators, the platform will create instances of the master application template, each with a unique name. Each instance is managed, configured and runs independently of other applications. By design, an instance of an application runs in a dedicated operating system process, providing separation between applications such that one application instance cannot directly exhaust the resources of another in a way that cannot be stopped with standard operating system commands (such as kill).

Managed Application Instances:

An application instance is considered managed if it is created, updated, and its lifecycle controlled by the [platform tooling](#) (page 21) (appadd, appstop, appstart, etc.). Managed applications are automatically upgraded when the platform is upgraded.

In some circumstances, you might want to create applications that are not fully managed. For example, applications created with the "--no-link" options to the appadd tool will not be directly upgraded by

platform upgrades. Non-managed applications are not directly upgraded by the Jive tooling and must be upgraded manually by a system administrator.

System Management Tools

Ultimately, most Jive SBS tools delegate to standard Linux systems management tools such as bash scripts, the “ps” command, and so on. It is possible to monitor Jive-managed components using standard tools such as ps and top as described in the Platform Run Book ([Linux](#) (page 6) ,).

Database Management Tools

When using the local database (installed as part of Jive SBS) as the system of record for a deployment, the platform will make an attempt to tune and back up the underlying database. While sufficient for smaller databases, larger databases will have storage and backup considerations unique to their individual deployments; you'll likely want to alter the platform defaults.

Auto Backups:

Upon installation, the Jive SBS platform database is scheduled for daily full backups via an entry in the cron.daily section of the system cron task scheduler. Specifically, the installation process creates a symlink that maps /usr/local/jive/bin/dbmaint to /etc/cron.daily/jive-dbmaint. This script performs a full analysis and backup of the contents of the local database. Backups are stored to /usr/local/jive/var/data/backup/postgres/full and are stamped with the date that the backup began. Logs for the backup and maintenance are stored in the standard platform location at /usr/local/jive/var/logs, in files dbmaint.log, dbback.log and dbanalyze.log.

In addition to full backups, the platform captures PostgreSQL WAL (Write-Ahead Log) segments to /usr/local/jive/var/data/backup/postgres/wal. For more information about WAL segments, see the *PostgreSQL documentation*.

Database Storage Considerations:

The platform backup infrastructure will purge full backups that are 30 days or more old. The platform will not purge WAL segment backups described in the previous section. As a general rule, you can safely remove WAL segments older than the most recent full backup. You should keep WAL segments since the previous full backup so that you have it for a partial recovery of the database in the event of corruption.

System administrators should evaluate the storage capacity of their systems and the size consumption to arrive at the best strategy for purging backup files.

Application Management

The platform manages one or more instances of the Jive SBS software. Each instance represents a dedicated Apache Tomcat application server that ultimately runs as a dedicated server process, independent of other application servers on the host machine. By convention, each application instance is represented by a directory entry beneath the “applications” directory located in the jive user’s home directory (/usr/local/jive/). For example, an application named “biodome” will have a corresponding application directory located at:

```
/usr/local/jive/applications/biodome
```

Each application has a standard set of subdirectories beneath its top-level application directory:

- <app_name>/bin – Scripts and environment configuration used by this instance. Notably:
 - manage – Used to start, stop and restart the application server instance in a graceful manner and intended to minimize the possibility of data loss.
 - setenv – Script for failsafe environment configuration invoked by the manage script. This script should not be edited except under the direction of Jive support.
 - instance – Per-instance configuration information. The contents of this script customize the environment for the named application instance. For example, to change the HTTP address the application server listens on, the platform will write shell script environment variables to this file.
- <app_name>/conf – Application server runtime configuration files as well as container logging configuration (log4j.properties)

- <app_name>/home – Application instance home directory. This directory represents the instance-specific storage for items such as search indexes, local file system caches, plugin caches, and database configuration.
- <app_name>/application – Commonly a symbolic link to the shared Jive application binaries. If an application instance is created by the appadd tool with the “-no-link” option, the application directory will be a full copy of the shared application binaries at the time the application was created and will not be upgraded during a package upgrade.

Troubleshooting and Snapshot Utilities

The Jive SBS platform installation captures meaningful system data to various log files as well as providing application and system snapshot capabilities. With these, Jive support can more easily diagnose issues with the platform.

Support Information Gathering:

The primary mechanism for gathering support-related information on the platform is via the [appsupport](#) (page 25) command, typically executed as the jive user.

When run, the appsupport command will combine multiple useful data points from the system and application logs, then aggregate the data to the console’s standard output stream. Alternatively, you can use the “-o” flag, along with the path to a file where the aggregate system information should be appended (if the file does not exist, it will be created).

System Thread Dumps:

In some cases, Jive support may suggest that you capture application thread dumps from a running system. The platform includes the [appsnap](#) (page 24) tool for this purpose. As with other commands, appsnap is commonly performed as the jive user.

The most common options used with the appsnap tool are the “-count” and “-interval” options. Count determines the number of samples taken. The interval option determines the number of seconds between successful samples. The tool writes snapshot output to the console’s standard output or appends it to the file given for the “-o” option.

What Happened to jiveHome?

If you're upgrading from a version prior to version 3, you might wonder about the jiveHome directory. The jiveHome directory was where you put (and found) the working files for your community. Caches, themes, plugins, logs -- that sort of thing.

As of version 3, the jiveHome directory is replaced by the jive.instance.home environment variable. Each instance depends on three environment variables to identify its place in the bigger picture of its environment:

- jive.home -- The root of all Jive software on the filesystem. Shared resources like Apache and Tomcat binaries are found here, as are individual instances.
- jive.name -- Unique, human-readable identifier of an instance within the local deployment (and importantly, not across all deployments); this value influences where the instance lives in a managed deployment (i.e., platform) and various other subtle artifacts like log file names.
- jive.instance.home -- Previously jiveHome. This is where the application puts working files, including caches, plugins, themes, etc. For example, by default this would be /usr/local/jive/applications/sbs/home/.

When you migrate your application to the platform, the contents of the jiveHome directory are distributed to places that are better suited to the platform. The following table list the new locations of resources previous found in jiveHome.

Resource	Location Prior to Version 3	Location in the Platform
Application logs (including those for database backups)	<jiveHome>/logs	/usr/local/jive/var/logs

Resource	Location Prior to Version 3	Location in the Platform
Installed plugins	<jiveHome>/plugins	/usr/local/jive/applications/ <app_name>/home/plugins
User-created themes	<jiveHome>/themes	/usr/local/jive/applications/ <app_name>/home/themes
Cached attachments	<jiveHome>/attachments	/usr/local/jive/applications/ <app_name>/home/attachments
Cached images	<jiveHome>/images	/usr/local/jive/applications/ <app_name>/home/images
Cached data	<jiveHome>/cache	/usr/local/jive/applications/ <app_name>/home/caches
Resources directory	<jiveHome>/resources	/usr/local/jive/applications/ <app_name>/home/resources
Local system database	<jiveHome>/database	/usr/local/jive/var/work/sbs/data/ postgres-8.3
jive_startup.xml file	<jiveHome>/jive_startup.xml	/usr/local/jive/applications/ <app_name>/home/jive_startup.xml

Setting jive.instance.home

Jive SBS uses the following convention to determine where jive.instance.home actually exists on the file system. These rules follow the general notion of more to less specific -- for example, if a user has gone to the trouble of specifying a JNDI property for the home directory, honor that over a more generic system property, honor that over a more generic environment property.

1. JNDI environment property named "jive.instance.home" in the context "java:comp/env/"
2. System property named "jive.instance.home"
3. JNDI legacy environment property named "jiveHome" in the same context as #1
4. System property named "jiveHome"
5. Default to OS default (/usr/local/jive/applications/\${name}/home or c:\jive\applications\\${name}\home} - \${name} is described above represents the -Djive.name property with a default of "sbs"

Platform Run Book (Linux)

This document covers basic system-administration commands for managing the platform.

Use the information here for tasks that an average system administrator would need to perform without familiarity of the actual functionality of the system. This guide assumes a basic understanding of common Unix or Linux commands and concepts. You'll find the application management commands mentioned here documented in [Application Management Commands](#) (page 21) .

For a higher-level look at the platform, be sure to see the [Platform Overview](#) (page 2) .

Jive HTTPD

The Jive HTTPD service is the main access point for HTTP and HTTPS access to the Jive SBS system by web browser.

Start Jive HTTPD

To start the jive-httpd service, execute the following command as root:

```
[root@biodome:~]$ /etc/init.d/jive-httpd start
Starting jive-httpd:
```

OK

If the command completes successfully, an OK message will be printed to the console and the exit code of the command will be zero.

Stop Jive HTTPD

To stop the jive-httpd service, execute the following command as root:

```
[root@biodome:~]$ /etc/init.d/jive-httpd stop
Stopping jive-httpd:
OK
```

If the command completes successfully, an OK message will be printed to the console and the exit code of the command will be zero.

Monitoring Jive HTTPD

The jive-httpd service supports a "status" command issued to the standard init script located at "/etc/init.d/jive-httpd". An example of checking the service status as the root user:

```
[root@biodome:~]$ /etc/init.d/jive-httpd status
JIVE_HOME set to /usr/local/jive
Running: 2393 (2396, 2397)
```

In the above example, the parent process of the jive-httpd system daemon is 2393, with child processes of 2396 and 2397.

In addition to the status script, it is possible to check the status of the jive-httpd daemon using standard Unix commands. For example, the following ps command will list all jive-httpd processes running on the host:

```
[root@biodome:~]$ ps -ef | grep jive-httpd | grep -v grep
root      2393      1  0 14:41 ?        00:00:00 /usr/local/jive/httpd/bin/jive-httpd -f /usr/
local/jive/etc/httpd/conf/httpd.conf -k start
jive      2395    2393  0 14:41 ?        00:00:00 /usr/local/jive/httpd/bin/jive-httpd -f /usr/
local/jive/etc/httpd/conf/httpd.conf -k start
jive      2396    2393  0 14:41 ?        00:00:00 /usr/local/jive/httpd/bin/jive-httpd -f /usr/
local/jive/etc/httpd/conf/httpd.conf -k start
jive      2397    2393  0 14:41 ?        00:00:00 /usr/local/jive/httpd/bin/jive-httpd -f /usr/
local/jive/etc/httpd/conf/httpd.conf -k start
jive      2398    2393  0 14:41 ?        00:00:00 /usr/local/jive/httpd/bin/jive-httpd -f /usr/
local/jive/etc/httpd/conf/httpd.conf -k start
jive      2399    2393  0 14:41 ?        00:00:00 /usr/local/jive/httpd/bin/jive-httpd -f /usr/
local/jive/etc/httpd/conf/httpd.conf -k start
```

Jive HTTPD Networking

The jive-httpd server by default listens for connections on port 80, on all available network interfaces. If configured for SSL (see the [Operations Cookbook](#) (page 11)), the server will also listen for connections on port 443. The following commands will show if the jive-httpd service is listening on the designated ports.

```
[root@melina ~]# lsof -n -i TCP:80 -i TCP:443
COMMAND  PID USER  FD  TYPE DEVICE SIZE NODE NAME
jive-http 3094 root   3u  IPv6 30661    TCP *:http (LISTEN)
jive-http 3098 jive   3u  IPv6 30661    TCP *:http (LISTEN)
jive-http 3099 jive   3u  IPv6 30661    TCP *:http (LISTEN)
jive-http 3100 jive   3u  IPv6 30661    TCP *:http (LISTEN)
jive-http 3101 jive   3u  IPv6 30661    TCP *:http (LISTEN)
jive-http 3102 jive   3u  IPv6 30661    TCP *:http (LISTEN)
jive-http 3104 jive   3u  IPv6 30661    TCP *:http (LISTEN)
jive-http 3105 jive   3u  IPv6 30661    TCP *:http (LISTEN)
jive-http 3273 jive   3u  IPv6 30661    TCP *:http (LISTEN)
```

```
jive-http 3274 jive 3u IPv6 30661 TCP *:http (LISTEN)
jive-http 3275 jive 3u IPv6 30661 TCP *:http (LISTEN)
```

In the above example, multiple jive-httpd processes are providing the "http" service. If listening for SSL or TLS connections, the "https" service will also be present.

Jive HTTPD Log Files

All log files for jive-httpd are stored in the standard platform log directory - /usr/local/jive/var/logs. The following command illustrates how to view the available logs.

```
[root@melina logs]# ls -l /usr/local/jive/var/logs/*http*
-rw-r--r-- 1 root root 224 Feb 23 16:12 /usr/local/jive/var/logs/httpd-error.log
-rw-r--r-- 1 root root 19454 Feb 26 08:25 /usr/local/jive/var/logs/jive-httpd-access.log
-rw-r--r-- 1 root root 854 Feb 23 16:13 /usr/local/jive/var/logs/jive-httpd-error.log
-rw-r--r-- 1 root root 854 Feb 23 16:13 /usr/local/jive/var/logs/jive-httpd-ssl-access.log
-rw-r--r-- 1 root root 854 Feb 23 16:13 /usr/local/jive/var/logs/jive-httpd-ssl-error.log
```

In the above example, startup logs are captured to the "httpd-error.log" file. Requests handled by the standard jive-httpd server are maintained in "jive-httpd-access.log" file while errors during normal runtime are captured to "jive-httpd-error.log". Likewise, SSL or TLS encrypted requests are captured to the corresponding log files with "ssl" appended to the name of the file.

Jive-Managed Applications

An installation of the Jive SBS platform will host one or more distinct applications. All managed applications have both system-level scripts that are invoked at system startup and shutdown, as well as scripts locally available to the "jive" system user created when the platform is installed. The following operations are available for managing and monitoring managed applications.

Start Jive-Managed Applications

To start all Jive-managed applications, a standard "init" script is available for use by the root user. This script is invoked at system boot to start all managed applications.

```
[root@biodome:~]$ /etc/init.d/jive-application start
JIVE_HOME set to /usr/local/jive
Starting jive-application:
All applications started successfully (1 total).
```

In addition to the system scripts, the jive user may start any individual application or all managed applications using the [appstart](#) (page 25) command. The appstart command is automatically added to the jive user's interactive shell path and may be run after becoming the jive user (commonly via the "su" command). The following example demonstrates the root user using the "su" command to become the jive user and then the use of the appstart command to start the "sbs" application.

```
[root@biodome:~]$ su - jive
[1456][jive@biodome:~]$ appstart --verbose sbs
Handling applications ['sbs']
Starting sbs...
Executing /usr/local/jive/applications/sbs/bin/manage start
sbs started successfully.
```

Stop Jive-Managed Applications

To stop all Jive-managed applications, execute the system stop script.

```
[root@biodome:~]$ /etc/init.d/jive-application stop
JIVE_HOME set to /usr/local/jive
Stopping jive-application:
```

All applications stopped successfully (1 total).

Similar to the `appstart` command, the `appstop` (page 25) command may be executed to stop managed applications as the `jive` user.

```
[1457][jive@biodome:~]$ appstop --verbose
Stopping sbs...
Executing /usr/local/jive/applications/sbs/bin/manage stop
sbs stopped successfully.
Cleaning sbs application work directory at /usr/local/jive/var/work/sbs.
All applications stopped successfully (1 total).
```

Monitoring Jive-Managed Applications

To show all running Jive-managed applications, execute the `appls` (page 23) command with the `--running` flag as the `jive` user as in the following example.

```
[1507][jive@biodome:~]$ appls --running
stage      running (pid=2799)
```

In this example, the "stage" application is currently running with a process ID of 2799. To monitor the individual process, standard tools like the `ps` command can be used with the process ID from `appls` output as in the following example.

```
[1542][jive@biodome:~]$ ps -ef | grep 2799 | grep -v grep
jive      2799      1  0 15:06 pts/0    00:00:16 /usr/local/jive/java/bin/java -XX:
+PrintClassHistogram -XX:+PrintTenuringDistribution -XX:+UseParNewGC -XX:+UseConcMarkSweepGC
-Djava.awt.headless=true -Djava.net.preferIPv4Stack=true -Xloggc:/usr/local/jive/var/
logs/stage-gc.log -Xmx2048m -Xms2048m -XX:MaxPermSize=512m -Djive.home=/usr/local/jive -
Djive.instance.home=/usr/local/jive/applications/stage/home -Djive.name=stage -Djive.context=
stage -Djive.logs=/usr/local/jive/var/logs -Djive.application=/usr/local/jive/applications/
stage/application -Djive.work=/usr/local/jive/var/work/stage -Djive.app.cache.ttl=10000
-Djive.app.cache.size=10240 -Dserver.port=9500 -Dhttp.addr='127.0.0.1' -Dhttp.port=9502
-Dajp.addr=127.0.0.1 -Dajp.port=9501 -Dajp.buffer.size=4096 -Dajp.max.threads=50 -
Dlog4j.configuration=file:///usr/local/jive/applications/stage/conf/log4j.properties -
Dtangosol.coherence.clusteraddress='224.224.224.224' -Dtangosol.coherence.clusterport=9503
-Dcatalina.base=/usr/local/jive/applications/stage -Dcatalina.home=/usr/local/jive/tomcat -
Djava.io.tmpdir=/usr/local/jive/var/work/stage -classpath /usr/local/jive/applications/stage/
bin//bootstrap.jar:/usr/local/jive/applications/stage/bin/tomcat-juli.jar::/usr/local/jive/
java/lib/tool.jar org.apache.catalina.startup.Bootstrap start
```

Alternatively, the following example combines both operations into a single command.

```
[1539][jive@biodome:~]$ ps -ef | grep 'appls --running | awk -F=' '{print $2}' | tr -cd
[:digit:]'
jive      2799      1  0 15:06 pts/0    00:00:16 /usr/local/jive/java/bin/java -XX:
+PrintClassHistogram -XX:+PrintTenuringDistribution -XX:+UseParNewGC -XX:+UseConcMarkSweepGC
-Djava.awt.headless=true -Djava.net.preferIPv4Stack=true -Xloggc:/usr/local/jive/var/
logs/stage-gc.log -Xmx2048m -Xms2048m -XX:MaxPermSize=512m -Djive.home=/usr/local/jive -
Djive.instance.home=/usr/local/jive/applications/stage/home -Djive.name=stage -Djive.context=
stage -Djive.logs=/usr/local/jive/var/logs -Djive.application=/usr/local/jive/applications/
stage/application -Djive.work=/usr/local/jive/var/work/stage -Djive.app.cache.ttl=10000
-Djive.app.cache.size=10240 -Dserver.port=9500 -Dhttp.addr='127.0.0.1' -Dhttp.port=9502
-Dajp.addr=127.0.0.1 -Dajp.port=9501 -Dajp.buffer.size=4096 -Dajp.max.threads=50 -
Dlog4j.configuration=file:///usr/local/jive/applications/stage/conf/log4j.properties -
Dtangosol.coherence.clusteraddress='224.224.224.224' -Dtangosol.coherence.clusterport=9503
-Dcatalina.base=/usr/local/jive/applications/stage -Dcatalina.home=/usr/local/jive/tomcat -
Djava.io.tmpdir=/usr/local/jive/var/work/stage -classpath /usr/local/jive/applications/stage/
bin//bootstrap.jar:/usr/local/jive/applications/stage/bin/tomcat-juli.jar::/usr/local/jive/
java/lib/tool.jar org.apache.catalina.startup.Bootstrap start
```

List Jive-Managed Applications

A list of all managed applications can be obtained by executing the `appls` command as the `jive` user as shown in the following example.

```
[1507][jive@biodome:~]$ appls
      stage      running (pid=2799)
      development stopped (pid=None)
```

In the output above, the "stage" application is running with process ID 2799, the "development" application is not running.

Jive-Managed Application Networking

The network ports and addresses used by a managed Jive application will vary depending on usage. The default Jive SBS application will work on the following addresses and ports.

Service	Protocol	Address
Application server management	TCP	127.0.0.1:9000
HTTP	TCP	127.0.0.1:9001
AJP	TCP	127.0.0.1:9002
Multicast Cluster	UDP/Multicast	224.224.224.224:9003

Note that managed applications should not be accessed directly via the HTTP 9001 port and it is recommended that a firewall prevent access to that port. Its existence is for troubleshooting and support purposes only and is not intended for production use.

To validate that the TCP services are present for a default install, execute the following command.

```
[root@melina ~]# lsof -n -P | grep jive | grep java | grep LISTEN
java      3204      jive    30u      IPv6      31631      TCP 127.0.0.1:9001 (LISTEN)
java      3204      jive    31u      IPv4      31632      TCP 127.0.0.1:9002 (LISTEN)
java      3204      jive    39u      IPv4      38046      TCP 127.0.0.1:9000 (LISTEN)
```

Jive-Managed Application Logs

Log files for Jive-managed applications are located in the var/logs directory of the jive user's home directory (/usr/local/jive/var/logs). The following log files can be consulted for further information on the status of individual applications. Each file will be prefixed with the name of the corresponding application. For example, for the "stage" application, the container log file will be named "stage-container.log".

- <name>.log - Primary log file for a managed application; most log entries will be located here.
- <name>-container.log - Early bootstrap log file for the application server container hosting the web application.
- <name>-session.log - Log file capturing creation and eviction of user session data.
- <name>.out - Redirection of standard out and standard error for the application process; may contain data not in the main log file.
- <name>-gc.log - Java garbage collection logs for the application.

Jive Database Server

The Jive SBS platform ships with a local PostgreSQL database server. The following operations are available for the database server.

Start Jive Database Server

To start the database server, execute the following system command as the root user.

```
[root@biodome:~]$ /etc/init.d/jive-database start
JIVE_HOME set to /usr/local/jive
Starting jive-database:
server starting
```

Stop Jive Database Server

To stop the database server, execute the following system command as the root user.

```
[root@biodome:~]$ /etc/init.d/jive-database stop
JIVE_HOME set to /usr/local/jive
Stopping jive-database:
waiting for server to shut down.... done
server stopped
```

Note that stopping the database while managed applications are using the database will result in applications that cannot service requests. Additionally, stopping the database while applications are connected may result in a lengthy shutdown time or a failed shutdown.

Monitoring Jive Database Server

Monitoring the database server can be done as the root user with system scripts, or with traditional Unix commands.

To check the status of the jive database, execute the following command as the root user.

```
[root@biodome:~]$ /etc/init.d/jive-database status
pg_ctl: server is running (PID: 3211)
/usr/local/jive/postgres/bin/postgres "-D" "/usr/local/jive/var/data/postgres-8.3"
```

The output of the above command lists the parent process of the database system (3211 in this example) and shows the command used to start the database.

A healthy, running database system will have multiple processes. The following command will show all running database processes on the system:

```
[root@biodome:~]$ ps -ef | grep post | grep -v grep
jive      3211      1  0 17:13 ?        00:00:00 /usr/local/jive/postgres/bin/postgres -D /usr/
local/jive/var/data/postgres-8.3
jive      3214    3211  0 17:13 ?        00:00:00 postgres: writer process
jive      3215    3211  0 17:13 ?        00:00:00 postgres: wal writer process
jive      3216    3211  0 17:13 ?        00:00:00 postgres: autovacuum launcher process
jive      3217    3211  0 17:13 ?        00:00:00 postgres: archiver process
jive      3218    3211  0 17:13 ?        00:00:00 postgres: stats collector process
```

Jive Database Server Networking

In the default configuration, the included database service listens for connections on TCP address 127.0.0.1 port 5432. To verify that the database is listening for connections, execute the following command.

```
[root@melina ~]# lsof -n -P | grep jive | grep postgres | grep LISTEN
      postgres  2990      jive    3u      IPv4    21499      TCP
127.0.0.1:5432 (LISTEN)
```

Jive Database Server Logs

Logs for the database server are maintained in the platform log directory at "/usr/local/jive/var/logs/postgres.log".

Operations Cookbook

This section is intended to provide sample configurations and script examples common to long-term operation of a Jive SBS installation.

As opposed to the Platform Run Book ([Linux](#) (page 6) ,) these operations are common to a new installation, but generally not for day-to-day operation of the platform.

Enabling SSL Encryption

The Jive SBS platform is capable of encrypting HTTP requests via SSL or TLS. Enabling encryption of HTTP traffic requires the following steps on a platform-managed host:

- Copy cryptographic materials to the host. By default, the Jive HTTPD server attempts to load an X.509 certificate file from the path `"/etc/jive/httpd/ssl/jive.crt"` and the corresponding key from `"/etc/jive/httpd/ssl/jive.key"`. The paths to these files are configured in the default Apache HTTPD virtual host file located at `"/etc/jive/httpd/sites/default.conf"` and can be changed to any path desired.
- Enable SSL in the HTTPD server by specifying the `"-D SSL"` option in the Apache HTTPD configuration extension file located at `"/etc/jive/conf/jive-httpd"`. To enable SSL, open (or create) this file and add `'OPTIONS="-D"'` to the file.
- With either Jive SBS's HTTP server or behind a third-party load balancer, add two attributes to the file at `/usr/local/jive/applications/<app_name>/conf/server.xml`. To the first (HTTP) `/Server/Connector` element, add this: `scheme="https" proxyPort="443"`.
- After making the changes above, restart the Jive HTTPD server as described in the run book for [Linux](#) (page 6) or .

Note: Except where noted above, if a third-party load balancer or external HTTP proxy is performing SSL termination upstream of the Jive HTTPD server, it is not necessary to configure the Jive HTTPD server for HTTP encryption in addition to the load balancer.

Note: If the private key file installed to the server is encrypted, the HTTPD server will interactively prompt the user for the password to decrypt the key.

Disabling the Local Jive System Database

Many deployments will not wish to use the locally managed platform database instead, choosing to use an RDBMS that is controlled by an internal corporate IT group. In this case, the Jive SBS local database should be disabled. To disable the database, as the root user, execute the following script:

The following terminal output demonstrates deactivation and of the Jive database service:

```
[root@biodome ~]# /etc/init.d/jive-database deactivate
Jive Database deactivated.
[root@biodome ~]# /etc/init.d/jive-database activate
Jive Database activated. The database will start automatically on the next system restart.
```

Note: Disabling the database does not stop the service if it is running. Likewise, re-enabling the database does not start the database service. Also, disabling the local system database will un-schedule all standard local system database maintenance tasks.

Changing the Configuration of an Existing Instance

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 will benefit 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 will 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

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 will look 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"
```

Many values contained in the application setenv script can be overridden in the instance configuration. Commonly modified values include:

- JVM_HEAP_MAX – the maximum JVM heap size in megabytes
- JVM_HEAP_MIN – the minimum JVM heap size in megabytes (should usually be set to the same as the JVM_HEAP_MAX value)

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 instance will be durable across application upgrades.

Performing a Jive System Database Backup

Jive SBS-managed databases will perform automatic backups as described in [Auto Backups](#) (page 4) in the [Platform Overview](#) (page 2) . In some situations, for example prior to an upgrade of the platform, you should perform a full database backup manually.

To manually perform a full backup of the managed database, execute the [dbbackup](#) (page 26) script as the jive user.

```
[0801][jive@melina:~]$ ./bin/dbbackup
/bin/tar: Removing leading '/' from member names
```

The command will not produce any further output if successful and will return zero if successful, non-zero otherwise.

You can restore from a backup by using PostgreSQL commands. In the PostgreSQL documentation, the *section on recovering from your backup* is probably the most specific. For a broader view, be sure to see the *contents of their documentation on backing up and restoring*.

Backup and Storage Considerations

Storage Reliability

It is highly recommended that the Jive system home directory (`/usr/local/jive`) be mounted on redundant external storage (preferably SAN storage via redundant HBAs and SAN fabric). When redundant external storage is not available, the local system volume for `/usr/local/jive` should be mirrored across multiple physical disks to prevent the loss of a single disk as a single point of failure.

The total storage requirements for this directory will vary from installation to installation. As a basic guide for capacity planning, consider the following:

- Core binaries - The base installation requires 500MB storage (200MB on disk, an additional 300MB needed during upgrades of the platform).
- Total system traffic - The system writes all logs to `/usr/local/jive/var/logs`. While the system will by default rotate log files to reduce disk space consumed, larger installations may wish to retain log files for analysis over time (HTTPD access logs for example). In a default installation, allocating 5GB for log storage should provide ample room to grow.
- Cache efficiency - For each application, local caches of binary content including attachments and images are maintained. The more space available to those caches, the more efficient the system will be at serving binary requests and the smaller the strain on the backing RDBMS. As a capacity guideline, plan on roughly .25 the planned total binary (BLOB) storage in the RDBMS for efficient caching.
- Search index size - Each node stores local copies of the system search index. As a general rule of thumb, plan for search indexes to be 1x the total database storage consumption (.5 for active indexes, .5 for index rebuilds).
- Local database backups - When using the Jive SBS platform-managed database, the database will regularly be backed up to `/usr/local/jive/var/data/backup/full` and database checkpoint segments backed up to `/usr/local/jive/var/data/backup/wal`. When an instance is using this database, approximately 35x the total database size will be required in the `/usr/local/jive/var/data/backup` location with a default configuration. This number can be lowered by more aggressively removing full backup archives stored in `backup/full` and by more aggressively removing WAL segments after a full backup has been performed.

Storage Monitoring

As with any system, disk consumption should be regularly monitored and alerts generated when the system approaches disk capacity. Most disk consumption will occur in three areas:

- Application instance home directory -- By default, the platform manages a single application instance located at `/usr/local/jive/applications/sbs` with a home directory of `sbs/home`
- Platform logs -- All platform log files are stored in `/usr/local/jive/var/logs`
- Platform database -- If the local platform database is used, data files will be stored in `/usr/local/jive/var/data/postgres-8.3` and backups in `/usr/local/jive/var/data/backup`

System Backups

In addition to performing regular backups of reliable storage, you should perform backups of the Jive system home. The most simple backup solution is to simply backup the entire contents of `/usr/local/jive`. A more selective option is to backup only `/usr/local/jive/applications` and `/usr/local/jive/etc`. In either case, you should make backups in accordance with standard backup practices.

Before upgrading Jive SBS, you should make a full backup of `/usr/local/jive`.

When you're using the platform-managed database, it's a good idea to maintain copies of `/usr/local/jive/var/data/backup` on a separate storage volume that's immune from corruption that may occur on the `/usr/local/jive` volume.

System Database Credentials

The Jive SBS local system database is intended for use only as the application's main database. Under most circumstances you shouldn't need to separately connect to it. For those cases when you do, the default connection information is as follows:

- **Connection URL:** `jdbc:postgresql://localhost:5432/sbs`
- **User name:** `sbs`
- **Password:** Passwords for database accounts are generated during installation and written to hidden files in `/usr/local/jive/etc/postgres/`. For example, you'll find the password for the local system database in `/usr/local/jive/etc/postgres/.cs-password`

Using an External Load Balancer

In order to integrate the Jive SBS platform with external load balancers, configure the load balancer for session affinity between each host running the platform. If the load balancer is performing SSL session termination (recommended), the load balancer should be configured to route traffic to port 80 each Jive managed server. If the load balancer is not performing SSL session termination, the load balancer should be configured to route traffic to port 443 and each server configured for SSL as described in the above cookbook recipe.

Depending on the load balancer, it may be necessary to add JVM route information to the outgoing JSESSIONID HTTP cookies sent to remote agents. In this case, add a `jvmRoute` attribute to the `Engine` element of the `server.xml` file located in the application's conf directory (`/usr/local/jive/applications/sbs/conf/server.xml` by default). For example, to name the route "r1", the `server.xml` `Engine` element would read:

```
<Engine defaultHost="localhost" name="Catalina" jvmRoute="r1">
```

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

Enable Application Debugger Support

Applications managed by Jive SBS are capable of accepting remote Java debuggers. To enable debugging, export environment variables "DEBUG" and "JPDA_TRANSPORT" prior to starting the managed application to be debugged.

For example, to debug via remote socket connection, start the desired application as shown below.

```
[0832][jive@melina:~]$ export DEBUG=1 && export JPDA_TRANSPORT=dt_socket && apstart sbs
```

Note that only one managed application may be debugged at a time. When running in DEBUG mode, the application JVM will halt until a debugger is attached.

Setting Up a Local Cluster

You might find it easier to isolate cluster-related issues by creating a cluster of instances on a single machine. The following steps describe how you can create a simple two-node cluster on a single machine. This assumes you have a license that supports more than one node in a cluster.

1. *Install the platform* using the default settings.
2. Use the `apprm` (page 24) command to remove the application you've installed, which installed it into the root ("/") context.

```
apprm sbs
```

3. Use the [appadd](#) (page 21) command to add two new application instances that will make up your cluster.

- a. You can add the first instance with default settings, specifying a context path:

```
appadd --context-path=/cluster1 cluster1
```

- b. Add the second instance with the following suggested settings:

```
[joe@targetmachine]$ appadd --ajp-port=9004 --server-port=9005 --http-port=9006 --cluster-port=9007 --context-path=/cluster2 cluster2
```

4. Restart the jive-httpd service. For more information see [Jive HTTPD \(Linux\)](#) (page 6) .
5. Start the first instance with the [appstart](#) (page 25) command.

```
appstart cluster1
```

6. With a browser, navigate to the setup tool (usually at <http://<hostname>/cluster1>, where hostname is the DNS resolvable name of the server where the package was installed) to configure the instance.

Note: Be sure to use a license that supports more than one node in a cluster.

7. Stop the first instance with the [appstop](#) (page 25) command.

```
appstop cluster1
```

8. Change the APP_CLUSTER_PORT in the `/usr/local/jive/applications/cluster2/bin/instance` file to be the same as cluster1's cluster port. For more on making this change, see [Changing the Configuration of an Existing Instance](#) (page 12) .

Note: In version 3.0.0, to support clustering you'll need to change configuration to override a default setting. [Changing the Configuration of an Existing Instance](#) (page 12) for the instance to add the following (where `<addr>` is a unique address -- 224.224.224.224 might work.):

```
export CUSTOM_OPTS="-Dtangosol.coherence.clusteraddress=<addr>
```

9. Start the second instance.

```
appstart cluster2
```

10. Use the setup tool to configure the instance (<http://<hostname>/cluster2>). When configuring the database, choose "external database," then use settings for the database from cluster1.
11. Stop the second instance.

```
appstop cluster2
```

12. Start both instances.

```
appstart
```

13. Use the admin console to turn on clustering on both instances. You'll find that setting at System > Settings > Caches.

Configuration Support for External Client Access

As of version 4, Jive SBS optionally supports access to the community from several new kinds of clients. This topic describes the features provided by Jive to help you ensure that these connections are secure.

The following lists the clients for which you might want to make special provision for secure access (each of these is an optional feature):

- Mobile device. Jive supports access from mobile applications such as its iPhone application.

- Bridged instances. Using a bridge, you can connect two Jive communities together. Through this bridge, people in one community (who are members the bridged communities) can see activity from the bridged communities. It's possible that the bridge might cross network boundaries.
- An add-in within Microsoft Office. Jive offers the Jive Desktop Office add-in through which people can upload and synchronize Office documents while working within the Office application on their desktop.
- Social Media Console. Among the set of features included with Jive Market Engagement is the Social Media Console, an external web application that provides a way to collect and organize market data.

Features designed to help you ensure secure access include:

- URL conventions that you can use to filter requests. See below for more on these.
- Admin console support for turning REST web services on or off. For more, see *Setting Access for Web Service Clients*.
- Admin console support for enabling or denying access via mobile device based on user name and device ID. For more, see *Managing iPhone Access*.

URL Conventions

Each of the client types listed above requires access to the community in order to exchange data about content, people and activity. Each communicates with the community using REST web services. To help you secure that access, Jive uses a URL convention that you can use to filter requests so that only those relevant to the services you're supported are allowed. Each client uses a different base URL to make requests.

Each REST URL for the clients listed begins with `__services` and is followed by a convention specific to the client type -- `/mobile`, `/bridging`, `/office`, and `/sme`. Using this convention you can filter access to permit valid requests. For example, imagine that you want to allow a bridge from a public community outside your firewall to a private community that's inside it. You could create a filter that permits URLs of the form `/__services/bridging/**`. Depending on your network topology and conventions, you could permit these URLs through your firewall, or you could set up a reverse proxy that would forward requests made to these URLs.

The following table lists base URLs for each client type:

Client	Base URL	Notes
iPhone	<code>/__services/mobile/v1/</code>	Services can be enabled or disabled in the admin console.
Bridged instance	<code>/__services/bridging/</code>	Services can be enabled or disabled in the admin console.
Jive Desktop Office add-in	<code>/__services/office/</code>	Services can not be disabled via the

Client	Base URL	Notes
		console if the feature is installed.
Social Media Control	/__services/sme/	Services can not be disabled via the console if the feature is installed.

Security for Client Requests

You can enable Secure Sockets Layer (SSL) for each type of client, although how you do so varies among the clients.

The following table describes how SSL is enabled for each client type:

Client	SSL Handling
iPhone	You can force SSL specifically for the iPhone from the admin console as described in <i>Setting Access for Web Service Clients</i> . Note that if you'll be using SSL to secure iPhone access, your certificate must be valid. For example, it must be created by a trusted authority such as Verisign, rather than self-created.
Bridged instance	You force SSL for access from bridges when you force it for REST web services in general. See <i>Setting Access for Web Service Clients</i> for more information about that setting.
Jive Desktop Office add-in	To force SSL for the Jive Desktop add-in, you must force SSL for the entire site, including for browser-based requests. For more information, see Enabling SSL Encryption (page 12) .
Social Media Console	To force SSL for the Social Media Console, you must force SSL for the entire site, including for browser-based requests. For more information, see Enabling SSL Encryption (page 12) .

Adding Fonts to Support Office Document Preview

When someone uploads a Microsoft Office document requiring a font that is not installed on the system running Jive SBS, the document's preview in the community can appear with text in an alternate font. This affects only the preview, not the original document.

You can install your licensed True Type fonts on the server where the application is installed. (The font files will have a .ttf or .TTF extension.) Install the fonts in two locations:

```
/usr/local/share/swftools/fonts/
/usr/local/jive/opt/openoffice.org2.4/share/fonts/
```

Note: On Solaris, you'll also need to add the following environment variable:

```
export FONTCONFIG_PATH=/etc/fonts
```

After installing the fonts, restart the jive-joosd service.

Fine-Tuning Performance

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

It's almost certain that you'll want to adjust application settings from their defaults shortly after you're up and running. In particular, you'll want to keep an eye on caching, but there are other things you can do to ensure that the application is performing as well as possible. See the following for 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 this file to change default cache time or headers for specific scenarios (changing length of time static images are cached, for example). Changes to this file will be preserved across upgrades to a file named `cache.conf.rpmnew`. If this file is changed, be sure to 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`, `.jpe`, `.jpg`, and `.png`. This means that clients won't see updated versions of those files until their cache expires or is cleared. Of course, changing the resource's file name will also cause it to be downloaded because it isn't yet cached.

Server-Side Page Caching

You can adjust server-side page caching for anonymous users when their having the very freshest content is less of a concern. With server-side caching on, the server caches pages that are assembled dynamically from data and resources. Retrieving a page from the cache can save the time needed to assemble a fresh page. However, if the data that makes up the page has changed, the page in the cache won't be as fresh as a new one would be.

With server-side page caching disabled (and for registered users, whether or not caching is enabled), Jive SBS sends its default HTTP headers. With page caching enabled, in addition to the server-side page cache stored in memory, the application also sets the HTTP header in the response to `Cache-Control max-age=3600`.

The value set for `max-age` is configurable as described below.

You can set these with system properties in the admin console.

Property	Description	Values
<code>jive.pageCache.enabled</code>	Enables server-side page caching.	false (default) to disable page caching; true to enable it. When enabled, only anonymous or guest users will receive cached content.
<code>jive.pageCache.maxage.seconds</code>	Sets the age after which the server will create a fresh page rather retrieve the page from the cache.	Defaults to 60 seconds. This sets <code>Cache-Control: max-age=60</code> in the HTTP headers for the page.
<code>jive.pageCache.expiration.seconds</code>	Sets the number of seconds after which a page will be removed from the cache.	Defaults to 30 seconds

Property	Description	Values
jive.pageCache.maxEntries	Sets the maximum number of pages that can be maintained in the cache. Note that increasing this value might require that you provide more system resources for the application.	Defaults to 1000 entries.

Note that turning on developer mode by setting the `jive.devMode` property to true will disable the `maxAgeFilter` setting (effectively setting `jive.maxAgeFilter.enable` to false). The `jive.devMode` property is intended for situations when you're developing themes or plugins, In those situations, caching can hinder you from seeing the results of your development work.

In the UI: Admin Console: System > Management > System Properties

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 SBS server. You can make this setting in the admin console.

In the UI: Admin Console: System > Settings > Resource Caching

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 SBS 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 SBS will rewrite static resource locations so that their URLs point to your CDN server.
4. When completing the page request, the client will use the CDN URL to retrieve static resources.
5. If the CDN server has the resource, it will return it; if not, it will retrieve the resource from the Jive SBS server, return it to the client, and cache it for future requests.

To configure the feature, go to the Resource Caching Settings page and select the **Enable external caching...** check box. Enter the CDN URL where static resources can be retrieved by clients.

Performance Recommendations for High Traffic Sites

Here are a few recommendations for high-traffic deployments in which most users are anonymous. Anonymous users tend to visit merely to read content rather than contribute. Because they're anonymous, by default there's very little about their experience that will be customized. As a result, you can typically afford to serve them cached content rather than freshly rendered dynamic content.

- Recognize anonymous users from the cookies the application sets. These cookies are available in 1.x versions as of 1.10.3 and in version 2.0.1, but are unavailable in version 2.0. If you're using a version in which they're not available, you should consider upgrading.
 - The application sets a `jive.user.loggedIn` cookie whose value is true if the user is registered and logged in.
 - Use the `jive.server.info` cookie to preserve consistency between user requests. This cookie contains information about the server from which the user receive content. Its value might be:

```
serverName=myhost:serverPort=8080:contextPath=/our-
community:localName=myhost:localPort=8080:localAddr=127.0.0.1
```

- Offload page caching to dedicated cached-content servers. With large numbers of anonymous users, you should consider using a content delivery network (CDN) such as Akamai or Limelight to serve cached content to anonymous users.

- Avoid sending a max age page cache header back to the client if the anonymous user will be able to log in. If the browser caches the page, then the newly logged in user might not receive fresh dynamic content.
- Consider caching resources such as CSS, Javascript, and decorator image files, which tend to rarely change. In fact, consider caching these resources for days if you expect not to be deploying new versions of them.
- Use the Query Stats feature to track database queries. The Query Stats page of the admin console can display SQL statements executed, how often they're executed, and how long the query takes. The number of queries you see when browsing your front pages should be low. For more information about this feature, see *Examining Database Queries*.
- When you have a large percentage of anonymous users but the overall number of page views is smaller, use the application's built-in server-side page caching. Starting with version 2.0.0, the application can [Server-Side Page Caching](#) (page 19) . You can enable the cache and set maximum age and expiration thresholds.

Adjusting Java Virtual Machine (JVM) Settings

As with any Java-based web application, you can sometimes improve performance by assigning particular values to Java Virtual Machine options. The default location for environment configuration is located at /usr/local/jive/applications/sbs/bin/instance. Within this file, it is possible to edit JVM maximum memory settings by adding a value for JVM_HEAP_MAX. This value is expressed in MB, so for example, to set the max heap available to the application to 4GB, add the following lines to the instance file:

```
JVM_HEAP_MAX=4096
```

```
JVM_HEAP_MIN=4096
```

Jive recommends setting the MIN and MAX values the same to ensure that the system is capable of allocating the desired maximum value and that an OutOfMemoryException will not occur during runtime.

Note that in addition to the heap designation above, a typical Jive SBS instance will consume 512M of JVM "PermGen" heap. In the above example, increasing the MIN and MAX values to 4GB would result in a JVM process consuming slightly more than 4.5GB RAM.

Assorted Performance Tuning Tips

Here are a few ways to get the application running the most efficiently.

- Consider setting the default for threaded discussions to "flat." People will still be able to set thread mode their own views to "threaded," but setting the default will ensure the "flat" mode for new users.
- If you're using a load balancer, make sure it's configured for session affinity/sticky sessions.
- On Oracle 11g, use the prepared statement cache to reduce database overhead.
- When using Oracle as an RDBMS, the OCI driver should be used as opposed to the "thin" JDBC Type4 driver. Using the OCI driver will require installation of Oracle native binaries compatible with the operating system hosting the Jive SBS installation.

Application Management Command Reference

Use these commands to perform maintenance tasks on your managed instance. Except where noted, you'll find these in /usr/local/jive/bin.

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

```
sudo su - jive
```

appadd Command

Jive Application Addition tool (appadd). Adds a new application configuration and configuration to the standard locations. Optional parameters may be overridden for hosting multiple instances on a physical host however such configurations are not recommended.

```
appadd [options] name
```

Short	Long	Description
	--version	Show program's version number and exit.
-h	--help	Show help message and exit.

HTTPD Options

Configures the HTTPD integration configuration options.

Short	Long	Description
	--httpd-addr	HTTPD listen address [default 0.0.0.0]
	--vhost	Create a virtual host for HTTPD integration as opposed to proxy directives (default is to create proxy directives only). Default: False
'	--dedicated-httpd-enable	Enable Dedicated HTTPD Server.
'	--dedicated-httpd-port=PORT	Port to use for dedicated httpd server.

Application Options

Configures general application server options.

Short	Long	Description
-s PORT	--server-port=PORT	Application server management port [default 9000]
-j OPT	--java-options=OPTS	Additional JRE options to use with the Java runtime.
'	--custom-option=OPTS	Additional application option to use with the Java runtime.
-c PORT	--cluster-port=PORT	Multicast cluster port [default 9003]
-m ADDR	--cluster-addr=ADDR	Multicast cluster address [224.224.224.224]
'	--cluster-jvm-route=ROUTE	Cluster JVM Route setting for Apache-based load balancing.
'	--cluster-local-port-enable	Enable local multicast cluster port [optional].
'	--cluster-local-port=PORT	Local multicast cluster port [default 9005]
'	--cluster-local-addr=ADDR	Local multicast address [optional].
'	--snmp-enable	Enabled SNMP monitoring [optional]
'	--snmp-port=PORT	SNMP port [optional] [default 10161]

HTTP Options

Configures application server HTTP options.

Short	Long	Description
-z	--http-port	Application server http port [default 9001]
'	--http-addr=ADDR	Application server http listen address [default 127.0.0.1]

AJP Options

Configures application server AJP options.

Short	Long	Description
-a PORT	--ajp-port=PORT	Application server ajp port [default 9002]
	--ajp-addr=ADDR	Application server ajp listen address [default 127.0.0.1]
-t THREADS	--ajp-threads=THREADS	Maximum number of application server AJP worker threads [default 50]
-b BYTES	--ajp-buffer=BYTES	Size in bytes for AJP connection buffers [default 4096]

Application Options

Configures application options.

Short	Long	Description
-p PATH	--context-path=PATH	Application context [default \'/\']
	--app-cache-size=BYTES	Static cache size in bytes [default 10240 (10M)]
	--app-cache-ttl=MS	Static cache TTL in ms [default 10000 (10 minutes)]

General Options

General configuration not specific to any subsystem. Most should only be used for testing.

Short	Long	Description
-v	--verbose	Be verbose about what actions are being taken. Default: False
-d	--debug	Show debug information. Default: False
	--source=SOURCE	Use the given application template path and not the default in JIVE_HOME. Default: None
	--destination=DEST	Output the application to the given path and not the default in JIVE_HOME. Default: None
	--overwrite	Overwrite any existing application artifacts. Default: False
	--force	Ignore any warnings and proceed, potentially causing conflicts with other applications. Default: False
	--no-link	Use copies instead of symlinks when creating the application [default is to link]
'	--auto-port	Automatically determine ports to use.

appls Command

```
appls [options]
```

Lists information about platform-managed applications.

Short	Long	Description
	--version	Show program's version number and exit.
-h	--help	Show help message and exit.
-f	--failed	Show only failed application instances. Default: False
-r	--running	Show only running application instances. Default: False
-s	--stopped	Show only stopped application instances. Default: False

Short	Long	Description
-q	--quiet	Remove unnecessary text for combination with other utilities. Default: False
-b	--brief	Remove even more unnecessary text for combination with other utilities.
-n	--name-only	Show only names meeting filter criteria. Default: False
-v	--verbose	Be verbose about what actions are being taken. Default: False
-d	--debug	Show debug information. Default: False

apprm Command

Jive application removal tool (apprm). Stops and removes a managed application given a valid application name.

```
apprm [options] name
```

Short	Long	Description
-h	--help	Show help message and exit.
'	--version	Show program's version number and exit.
-s	--no-stop	Do not stop the application before removing. Default: True
-v	--verbose	Be verbose about what actions are being taken. Default: False
-d	--debug	Show debug information. Default: False

appsnap Command

Jive Application Snapshot tool (appsnap). Passively gathers information about a running system and applications.

```
appsnap [options] [name]
```

Short	Long	Description
	--version	Show program's version number and exit.
-h	--help	Show help message and exit.

Snapshot Options

Defines the interval and count of snapshots taken, system or application.

Short	Long	Description
	--no-sys	Do not gather top-level system information. Default: True
-c COUNT	--count=COUNT	Sample count to take [default 1]
-i INTERVAL	--interval=INTERVAL	Time between samples [default 1]
	--jstack=PATH	Use the given path to the jstack binaries for sampling applications. Default: None

Short	Long	Description
	--jstack-opts=OPTS	Pass the given options to jstack binary for sampling detail. Default: None
'	--force=FORCE	Force Java thread dump. This may be fatal to the process resulting in failure if the dump is successful.
-o OUTPUT	--out=OUTPUT	Append output to the given file creating if the file does not exist [default STDOUT]

General Options

General configuration not specific to any subsystem. Most should only be used for testing.

Short	Long	Description
-v	-verbose	Be verbose about what actions are being taken. Default: False
-d	-debug	Show debug information. Default: False

appstart Command

Jive Application Start tool (appstart). Starts a template-configured application by name. If no name is given, all configured applications are started.

```
appstart [options] name
```

Short	Long	Description
	--version	Show program's version number and exit.
-h	--help	Show help message and exit.
-v	--verbose	Be verbose about what actions are being taken. Default: False
-d	--debug	Show debug information. Default: False

appstop Command

Jive Application Stop tool (appstop). Stops a template-configured application by name. If no name is given all configured applications are stopped.

```
appstop [options] name
```

Short	Long	Description
	--version	Show program's version number and exit.
-h	--help	Show help message and exit.
-r	--retain	Retain work directory for the application (default false)
-v	--verbose	Be verbose about what actions are being taken. Default: False
-d	--debug	Show debug information. Default: False

appsupport Command

Jive Application Support tool (appsupport). Passively gathers information about a running system for communicating with Jive support.)

appsupport [options]

Short	Long	Description
	--version	Show program's version number and exit.
-h	--help	Show help message and exit.

System Options

Fine tune the system information captured by the support dump. By default, all data is captured.

Short	Long	Description
-m	--no-mem	Do not gather memory information. Default: True
-c	--no-cpu	Do not gather CPU information. Default: True
-u	--no-uptime	Do not gather uptime information. Default: True
-s	--no-os	Do not gather operating system information. Default: True
-z	--no-limits	Do not gather ulimit information. Default: True
-x	--no-sysctl	Do not gather sysctl information. Default: True
-n	--no-network	Do not gather network information. Default: True
-f	--no-firewall	Do not gather firewall information. Default: True
-l	--no-logs	Do not gather system log information. Default: True

Jive Options

Fine tune the jive-specific information captured by the support dump. By default, all data is captured.

Short	Long	Description
-e	--no-jive-config	Do not gather Jive-specific configuration information.
-a	--no-jive-apps	Do not gather jive application data.
-j	--no-jive-logs	Do not gather jive-specific logs.
-i	--no-httpd-logs	Do not gather Jive HTTPD logs.
-p	--no-db-logs	Do not gather local system database logs.
-t BYTES	--limit-logs=BYTES	Limit log captures to a given number of bytes [default no limit]

General Options

General configuration not specific to any subsystem. Most should only be used for testing.

Short	Long	Description
-o OUTPUT	--out=OUTPUT	Append output to the given file creating if the file does not exist [default STDOUT]
	--no-time	Do not add timestamps to output. Default: True
-v	--verbose	Be verbose about what actions are being taken. Default: False
-d	--debug	Show debug information. Default: False

dbbackup Command

Backs up the application database. To create a full backup, omit the options. For more on backups, see the [Performing a Jive System Database Backup](#) (page 13) .

```
dbbackup [options]
```

Short	Long	Description
-s SEGMENT	' FILE	Database segment to archive.
-d DATA_PATH	'	Path to the data source.
-p DESTINATION_PATH	'	Path to which the archive should be written.

manage Command

Starts, stops, or restarts the application. Also checks the status of the application.

This command operates on the application it is installed with (*app_name*).

Location: /usr/local/jive/applications/<app_name>/bin

```
manage [start | stop | restart | status]
```

upgrade Command

Jive platform upgrade tool. Used to detect if an upgrade is in progress and which behaviors need to be executed, optionally executing them depending on command options. When invoked with no options, returns 0 if an upgrade is in progress, 1 otherwise.

```
upgrade [options]
```

Short	Long	Description
	--version	Show program's version number and exit.
-h	--help	Show help message and exit.

Upgrade Options

Configures upgrade behaviors.

Short	Long	Description
-x	--dry-run	Show upgrade tasks that would be performed. Default: False
-e	--execute	Perform any pending platform upgrade tasks. Default: False
-r	--reset	Mark all unperformed upgrade tasks as complete and exit. Default: False
-f PATH	--database-file=PATH	Use alternate upgrade database file. Default: None

General Options

General configuration not specific to any subsystem. Most should only be used for testing.

Short	Long	Description
-v	-verbose	Be verbose about what actions are being taken. Default: False
-d	-debug	Show debug information. Default: False