

Jive Platform Overview



Contents

Jive Platform Overview	2
System Components	2
Apache HTTPD Server.....	2
Tomcat Application Server.....	3
PostgreSQL Database Server.....	3
Jive System Configuration.....	3
Platform Conventions, Management and Tools	3
The "jive" User.....	3
Application Abstractions.....	3
System Management Tools.....	4
Database Management Tools.....	4
Application Management.....	4
Troubleshooting and Snapshot Utilities.....	5
What Happened to jiveHome?	5
Setting jive.instance.home.....	6

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 a 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 your data 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 specific DBMSes. Other database servers aren't currently supported for applications deployed on the platform.
- Application server support is optimized to the Apache Tomcat instance included with the platform. Other application servers aren't supported.
- Supported operating systems include Solaris, 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 (<i>Linux, Solaris</i>)	Basic system commands for managing the platform. If you need to handle something quickly, start here.
<i>Operations Cookbook</i>	Sample configurations and script examples for long-term operations.
<i>Application Management Commands</i>	A reference for commands you can use to maintain your managed instance.
<i>System Requirements</i>	System components and technologies supported in this release.
Installation guides for Linux and Solaris	Step by step instructions for installing the platform using a package manager.

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 package manager 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 supported 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, the Jive SBS package 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 package installation is being performed, the package 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 installed through the package 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* (appadd, appstop, appstart, etc.). Managed applications are automatically upgraded when the package 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 OS 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, Solaris*).

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* 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* 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
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"