

Installing and Upgrading

Contents

Installing and Upgrading	2
Before You Install	2
About the Jive SBS Platform.....	2
Database Prerequisites.....	2
High-Level Installation Steps	3
Installing the Application	3
Installing the Software.....	4
Troubleshooting Installation.....	6
Upgrading from an Earlier Version	8
Upgrading the Software.....	9
Troubleshooting Upgrade.....	9
Transferring jiveHome.....	10
Migrating Data from an Unsupported DBMS.....	12
Migration Troubleshooting.....	15
Back Up Your jiveHome Directory.....	16
Back Up Your Database.....	16
Remove Plugins Before Upgrading.....	16
Post-Installation Tasks	16
Installation Known Issues	17
Application Management Command Reference	18
appadd	18
appls	20
apprm	20
appsnap	20
appstart	21
appstop	21
appsupport	22
dbbackup	23
manage	23
upgrade	23

Installing and Upgrading

Here you'll find instructions for installing the application and upgrading from a installation prior to version 3.

The RPM distribution you received is probably all you need to install the application onto a support Linux operating system. The RPM (formerly "Red Hat Package Manager") format is standard for enterprise Linux software. In addition to making installation much easier, the familiar RPM format provides powerful features for validating the target installation environment. It also allows for sophisticated upgrade capabilities managed through Jive Software.

Installing with the RPM distribution is part of a greatly simplified platform configuration. With the version 3 release, the distribution you receive includes key required pieces (such as an application server, Java VM, and optionally a database). At the most basic level, you need only provide hardware with the required operating system. You can, of course, supply your own database and user directory, in keeping with the *system requirements*.

In This Section

- Your Linux operating system should have the RPM manager installed. When installing, you'll invoke that manager to install the RPM package that contains Jive SBS.
- Install the application RPM before you upgrade. You'll use the tools described here to move data into the new installation. You'll manage future upgrades by working with Jive Software and using the RPM manager.
- The application includes (and installs) a PostgreSQL DBMS, but you needn't use it. If you have a separate PostgreSQL or Oracle DBMS you want to use, you can specify its location when you're setting up the application after installation. You can also use your own user directory server.
- The distribution includes an application server; you won't need to install one.

Before You Install

Here are a few things you'll want to do and be aware of before you start installing Jive SBS.

About the Jive SBS Platform

The Jive SBS platform introduced in version 3 includes a few important changes from versions prior to version 3. Through research and conversation with its customers, Jive arrived at a platform that is not only well-suited to its customers' needs, but also enables Jive to more fully optimize the application. For more specific information, see the *System Requirements* and the *Platform Overview*. Once you've installed, you might also want to check out the *Platform Run Book*, [Application Management Commands](#) (page 18) , or the *Operations Cookbook*

Database Prerequisites

You'll find that both installation and upgrade go more smoothly if most database-related tasks are done before you begin installing. This is especially true if the database the application will be using is administered separately by a DBA. This section describes the database tasks that are required for a successful installation.

- Make a backup of your existing database. This probably goes without saying.
- Ensure that the target database allows access through which the application's setup tool can create tables. After you install the application using RPM, you'll finish installation (or upgrade) by using the application's admin tools to connect to the target database. The tool will create or upgrade the tables needed by this version of the application. You'll want to be sure that permission to create tables is granted at least until that process is completed.

For example, if you'll be using an Oracle database for a new installation, you'll be creating a user representing a schema. That schema will be empty until the application setup tool creates tables in it. Here's an example script for creating such a user:







```
create user jiveuser identified by changeme default tablespace jivedata;
grant create session, create table, create view, create sequence, create procedure, create
  synonym to jiveuser;
alter user jiveuser quota unlimited on jivedata;
```



The username can be any legal Oracle identifier -- "jiveuser" is just an example. The application does not require a dedicated tablespace, but many DBAs have that practice. The tablespace must be created separately. For more on the *create user* statement, see the Oracle documentaion.

- Create or migrate your application database. How you do this will vary depending on whether you're installing a new instance or upgrading.
 - If you're installing a new instance:
 - If you're using the included PostgreSQL DBMS, installation handles database setup for you. You don't need to create a database.
 - If you're using an external PostgreSQL or Oracle DBMS, create the empty target database before you begin installing the application. Ensure that the new database allows the application setup tool to create tables. After you install, and during setup, the application will connect to your existing database and create the necessary tables.
 - If upgrading an existing instance:
 - If you're upgrading from an instance that uses a DBMS supported for this version:
 - If you're upgrading "in place" to continue to use the existing database, then the application handles the change after installation when you use the upgrade console. After you install, and during setup, the application will connect to your existing database and upgrade it.
 - If you're upgrading from an instance that uses a DBMS *not* supported on this version, create the empty target database before you begin installing the application. Migrate the data from the unsupported database to the supported one using the process described in [Migrating Data from an Unsupported DBMS](#) (page 12) . Ensure that the new database allows the application setup tool to create tables. After you install, and during setup, the application will connect to your existing database and upgrade it.

High-Level Installation Steps

The following illustration lays out the high-level steps for installing Jive SBS and upgrading from a previous version. These steps are described in detail in this guide. Note that you're strongly encouraged to complete the database tasks described above before you start installing the application.

Installation	Upgrade	Performed By
1 Copy the RPM platform package to the target host.		
2 Install the RPM platform package.		
	3 Back up existing jiveHome directory.	
	4 Stop the existing instance.	
	5 Transfer jiveHome information from the existing application to the target host.	
3 Navigate to the admin console to complete the installation by configuring.	6 Navigate to the admin console to complete the installation by upgrading.	

 System Administrator
 Application Administrator

Installing the Application

In This Section

- You'll need to have SSH access to the target host to copy the distribution there and execute commands.
- You'll use the RPM manager built into Linux on the target host to install the distribution.
- The process managed by RPM takes care of most of the installation itself; you'll finish the process using the web-based setup console.

What You'll Need

To install Jive SBS using the RPM, you'll need the following:

- A server that meets the minimum specified hardware requirements. See the *System Requirements* for more information.
- The host computer must have a Jive-certified Linux operating system installed. See the *System Requirements* for supported versions.
- A copy of the application RPM. You'll also need SSH access to the host computer so you can copy the RPM there for installation.
- Root access to the host where the installation is performed, commonly via SSH or less commonly through user interface access such as VNC.

Installing the Software

The following installation steps represent the most common approach to installing the Jive SBS Platform.

1. Ensure that that application database has been created as described in [database prerequisites](#) (page 2) .
2. From the command line, access the target host as root. For example, the following illustrates using the `ssh` command to access the server at targethost as the root user.

```
joe@joesbox ~ $ ssh root@targethost
root@targethost's password:
Last login: Tue Jan 20 14:00:56 2009 from joesbox.example.com
```

3. If you haven't yet done so, copy the application RPM to the target host. Here's an example using the Linux `scp` command to copy from a computer named edda to a target host at targethost:

```
scp -v joe@joesbox:/Users/joe/jive_sbs-3.0.0-78305.x86_64.rpm root@targethost:/root
```

4. Set options for installation. If you want, you can set installer options (listed in the following table). Ordinarily you won't need these, but they can be useful in some cases.

To set these, use the Linux `export` command to set them as environment variables. All package-level variables are enabled by setting their value to a non-empty string. For example, the following example turns on debugging information:

```
export JIVE_DEBUG=1
```

You can clear the variable with a command such as the following:

```
unset JIVE_DEBUG
```

Option	Description	Default
JIVE_DEBUG	Exposes installation debugging information, listing actions the installer is performing. This is in addition to the information displayed by the RPM <code>-v</code> (verbose) flag.	Debugging information isn't displayed.

Option	Description	Default
JIVE_APPLICATION_ON_SERVICE	Prevents the package from starting Jive SBS services immediately after installation.	By default, the application starts immediately after installation; you'll be able to navigate to its setup tool using a web browser.

5. Install the application RPM using an `rpm` command such as the following. Here, the "i", "h", and "v" options are provided to indicate install with hash indicators, and to be verbose during the installation. (Note that your copy of the RPM file -- here, `jive_sbs-3.0.0-78305.x86_64.rpm` -- might have a slightly different name.)

```
rpm -ihv jive_sbs-3.0.0-78305.x86_64.rpm
```

Note: You can find out more about *rpm command syntax* at the Fedora web site.

The following shows console output for a successful installation using the preceding command. In this case, the RPM file was in the `/root` directory of the target host.

```
[root@targethost ~]# ls -l
total 191536
drwxr-xr-x 2 root root      4096 Dec 18 15:08 Desktop
-rw-r--r-- 1 root root 195742683 Mar  9 09:14 jive_sbs-3.0.0.RHEL-5.x86_64.rpm
-rw-r--r-- 1 root root   177294 Jan  2 2008 sysstat-7.0.2-1.el5.x86_64.rpm
-rwxr-xr-x 1 root root    1707 Jan 16 10:42 updateDNS.sh
Preparing...                               ##### [100%]
Preparing clean installation.
Pre-install tasks complete.
 1:jive_sbs                                 ##### [100%]
Writing installation version.
Wrote installation version.
Executing Jive post-install configuration.
Creating jive group jive.
Creating jive system user jive.
Marking all upgrades as complete.
Starting Jive System Daemon.
Performing Jive system configurations.

Initializing database for first use.
Starting Jive System Database.
Starting jive-database:
server starting

Configuring scheduled database maintenance.
Configuring log rotation maintenance.
Staging Jive Application.
Validating configuration.
Staging application from template: /usr/local/jive/applications/template
Linking application to master binary at '/usr/local/jive/applications/template/
application'.
Creating application configuration at: /usr/local/jive/applications/sbs/bin/instance
Application context set to '/'.
Creating proxy configuration for default HTTPD virtual host.
Staging cryptography.
Creating private key to /usr/local/jive/applications/sbs/home/crypto/sbs.pem
Creating public key to /usr/local/jive/applications/sbs/home/crypto/sbs.pub
Successfully created application at '/usr/local/jive/applications/sbs'.
Starting Jive applications.
Handling applications ['sbs']
Starting sbs...
Executing /usr/local/jive/applications/sbs/bin/manage start
sbs started successfully.
All applications started successfully (1 total).
Starting jive-httpd:

Jive post-install configuration complete.
```

When it's finished, the RPM indicates that the post-install configuration has completed and that the SBS application has been started successfully.

6. If you're doing a new installation (rather than upgrading), you're finished! With a supported web browser, navigate to `http://<hostname>/`, where `hostname` is the DNS resolvable name of the

server where the RPM was installed. There, you can finish configuring the application with the setup console.

If you're upgrading, you'll find the rest of the steps you need in [Upgrading from an Earlier Version](#) (page 8) .

Troubleshooting Installation

The Jive SBS installation uses Linux RPM, a widely tested and used application that is very unlikely to fail. However, there are a few dependences that Jive SBS requires (that you are likely to have already). And if you do run into trouble, you can delete and start over as described below.

Note: You'll find the installation log files on the target computer at `/usr/local/jive/var/logs`.

Unsatisfied Dependencies

The application RPM depends on the presence of several low-level system packages that are common to nearly all configurations of Jive's supported Linux distributions. Also, the application RPM depends on three high-level packages. If any of these packages (system or high-level) are not present, the RPM subsystem will warn you, then refuse to install. When you see these warnings, simply install the missing packages using RPM, then install Jive SBS as described in the instructions.

Unsatisfied dependencies appear as an error when attempting to install the RPM:

```
[root@targethost ~]# ls -l
total 202068
-rw-r--r-- 1 root root 206701420 Jan 20 16:03 jive_sbs-3.0.0-78310.i386.rpm
-rwxr-xr-x 1 root root 1347 Oct 7 16:14 updateDNS.sh
[root@targethost ~]# rpm -ivh jive_sbs-3.0.0-78310.i386.rpm
error: Failed dependencies:
    bash >= 3.2 is needed by jive_sbs-3.0.0-78310.i386
    sysstat >= 7 is needed by jive_sbs-3.0.0-78310.i386
```

Depending on the host configuration, it may be possible to install the dependencies directly using system tools. For example, in RedHat Enterprise Linux, the "yum" command can install dependencies via network repositories. The following demonstrates how to install the dependencies shown in the error above. Note that it is necessary to type "y" for yes when prompted:

```
[root@targethost ~]# yum install bash-3.2 sysstat
Loading "installonlyn" plugin
Setting up Install Process
Setting up repositories
extras          100% |=====| 1.1 kB 00:00
updates                    951 B 00:00
base              100% |=====| 1.1 kB 00:00
addons            100% |=====| 951 B 00:00
Reading repository metadata in from local files
primary.xml.gz    100% |=====| 90 kB 00:00
##### 295/295
primary.xml.gz                    369 kB 00:03
##### 796/796
primary.xml.gz          100% |=====| 853 kB 00:01
##### 2458/2458
Parsing package install arguments
Resolving Dependencies
--> Populating transaction set with selected packages. Please wait.
--> Downloading header for sysstat to pack into transaction set.
sysstat-7.0.2-1.el5.i386. 100% |=====| 15 kB 00:00
--> Package sysstat.i386 0:7.0.2-1.el5 set to be updated
--> Downloading header for bash to pack into transaction set.
bash-3.2-21.el5.i386.rpm 100% |=====| 55 kB 00:00
--> Package bash.i386 0:3.2-21.el5 set to be updated
--> Running transaction check
Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
```

```

  sysstat                i386          7.0.2-1.el5          base                168 k
Updating:
  bash                   i386          3.2-21.el5           base                1.9 M
Transaction Summary
=====
Install      1 Package(s)
Update      1 Package(s)
Remove      0 Package(s)
Total download size: 2.0 M
Is this ok [y/N]: y
Downloading Packages:
(1/2): sysstat-7.0.2-1.el1 100% |=====| 168 kB    00:00
(2/2): bash-3.2-21.el5.i3 100% |=====| 1.9 MB    00:02
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Updating      : bash                ##### [1/3]
  Installing    : sysstat            ##### [2/3]
  Cleanup      : bash                ##### [3/3]
Installed: sysstat.i386 0:7.0.2-1.el5
Updated:   bash.i386 0:3.2-21.el5
Complete!

```

After dependencies have been resolved, the package should install as normal.

Insufficient System Memory

The Jive SBS platform requires a minimum of 3GB RAM to operate effectively for an enterprise environment. If insufficient memory is not available on the target installation system, the installer will provide a warning at installation time similar to the example below.

```

[root@targethost ~]# rpm -ivh jive_sbs-3.0.0-78310.i386.rpm
Preparing... ##### [100%]
 1:jive_sbs   ##### [100%]
Writing installation version.
Wrote installation version.
Executing Jive post-install configuration.
Creating jive group jive.
Creating jive system user jive.
useradd: warning: the home directory already exists.
Not copying any file from skel directory into it.
Marking all upgrades as complete.
WARNING: this host does not have sufficient RAM to run a production Jive SBS system.
A minimum of 3GB is required to host the application and HTTPD servers. 4GB is required to
run a locally hosted database.
Starting Jive System Daemon.
Performing Jive system configurations.
Disabling CPU frequency stepping.

Initializing database for first use.
Starting Jive System Database.
Starting jive-database:
server starting

Configuring scheduled database maintenance.
Configuring scheduled database backups.
Configuring log rotation maintenance.
Staging Jive Application.
Validating configuration.
Staging application from template: /usr/local/jive/applications/template
Linking application to master binary at '/usr/local/jive/applications/template/application'.
Creating application configuration at: /usr/local/jive/applications/sbs/bin/instance
Application context set to '/'.
Creating proxy configuration for default HTTPD virtual host.
Staging cryptography.
Creating private key to /usr/local/jive/applications/sbs/home/crypto/sbs.pem
Creating public key to /usr/local/jive/applications/sbs/home/crypto/sbs.pub
Successfully created application at '/usr/local/jive/applications/sbs'.
Starting Jive applications.
Handling applications ['sbs']
Starting sbs...
Executing /usr/local/jive/applications/sbs/bin/manage start
Failed to start application sbs. See log file at '/usr/local/jive/var/logs/sbs.out'.

```

```
Failed to start Jive applications. Please check the error logs at '/usr/local/jive/var/logs/sbs.log'.
Disabling existing Apache HTTPD server.
Starting jive-httpd:
[ OK ]
Jive post-install configuration complete.
```

In the above example, note the warning message "WARNING: this host does not have sufficient RAM to run a production system. A minimum of 3GB is required to host the application and HTTPD servers. 4GB is required to run a locally hosted database."

Despite this warning, the RPM does install correctly, however further errors are noted on the output line "Failed to start application sbs. See log file at '/usr/local/jive/var/logs/sbs.out'." Upon further examination, the contents of this log file indicate:

```
[root@targethost ~]# cat /usr/local/jive/var/logs/sbs.out
SCRIPT_DIR=/usr/local/jive/applications/sbs/bin
JIVE_BASE=/usr/local/jive/applications/sbs
```

```
Creating temp directory at /usr/local/jive/var/work/sbs.
Starting application sbs
Error occurred during initialization of VM
Could not reserve enough space for object heap
```

Starting Over

In the unlikely event that something goes wrong during installation and you want to start over, you can uninstall. When uninstalling, you don't specify the RPM file name, as you did when installing. Instead, you give the logical name by which RPM now knows the application: `jive_sbs`. Here's an example using the `rpm` command with the `-e` switch for uninstalling:

```
rpm -e jive_sbs
```

If you want to be sure you've removed all remnants of the installation, delete the destination directory created by RPM. Here's how that command looks:

```
rm -rf /usr/local/jive
```

Upgrading from an Earlier Version

You can upgrade from an earlier installation, version 2.5 and higher (if you're on a version prior to 2.5, you'll need to upgrade to 2.5 before upgrading to this version). If you're experienced with previous upgrades, you'll notice that the new platform presented with Jive SBS means that the process has changed considerably. While the steps you'll take here to upgrade to Jive SBS include special steps to get from a previous version to this one, you'll only do these steps for this upgrade; future upgrades are made much simpler because they're managed through the RPM.

The following steps are written for the scenario where an existing installation resides on a separate host from the installation of the Platform RPM. If the RPM is installed to the same host where the previous instance is running, the steps below are the same, with the exception that you do not need to copy files between the two hosts.

The following assumes that the Jive SBS Platform RPM has been installed to the target host and that the installation produced no warnings or errors.

In This Section

- Jive Software supports upgrades to Jive SBS from version 2.5.x and later.
- These instructions assume you've already installed the application on the target host. If you haven't, you can use the [installation instructions](#) (page 4) to do so.

- You'll export information from your jiveHome directory, the home for application environment-related data.
- If you're upgrading from an application instance that uses a DBMS not supported on version 3, you should already have migrated your database to a supported DBMS. Take a look at the [database prerequisites](#) (page 2) for more information.
- Upgrading from an earlier version includes steps you'll take only for upgrades from a version older than 3.0. Future upgrades are made much simpler by the RPM manager this version uses.

Note: Upgrading an existing standalone or source build to version 3 is not supported. Make sure to [back up your jiveHome directory](#) (page 16) and [back up your database](#) (page 16) before doing an upgrade.

Upgrading the Software

1. Ensure that you've [installed the platform package](#) (page 4) to the new host.
2. Verify that you've [backed up your jiveHome directory](#) (page 16) . You should also have a backup of your existing database.
3. [Remove plugins](#) (page 16) from your older instance. After you've got the upgraded instance running, re-install the plugins using the admin console.
4. Ensure that the existing application database has been migrated if it uses a DBMS not supported for this release. See the [database prerequisites](#) (page 2) for more information.
5. Stop the default SBS application instance. By default, the Jive SBS Platform RPM installs an empty, unconfigured instance of the application software to the target host. Before upgrading, stop this instance as the jive system user:

```
[1635][jive@targethost:~]$ appstop --verbose
Stopping sbs...
Executing /usr/local/jive/applications/sbs/bin/manage stop
sbs stopped successfully.
All applications stopped successfully (1 total)
```

6. Transfer the jiveHome directory by using the tools provided with this distribution. See [Transferring jiveHome Information](#) (page 10) for instructions.
7. With a supported web browser, navigate to `http://<hostname>/`, where hostname is the DNS resolvable name of the server where the RPM was installed. There, you can proceed with the upgrade using the upgrade console.
8. After you've finishing the upgrade with the upgrade console, you'll need to restart the application. To do this, log into the target host as the jive user and execute the following commands:

```
[1520][jive@kuato:~]$ appstop --verbose
Stopping sbs...
Executing /usr/local/jive/applications/sbs/bin/manage stop
sbs stopped successfully.
All applications stopped successfully (1 total).
[1520][jive@kuato:~]$ appstart --verbose
Starting sbs...
Executing /usr/local/jive/applications/fsbs/bin/manage start
sbs started successfully.
All applications started successfully (1 total).
```

9. You're finished!

Troubleshooting Upgrade

This section details common issues that arise during the upgrade process as well as their solutions.

Driver Not Found Errors

Due to licensing restrictions, Jive is unable to ship database drivers for all previously-supported databases. Namely, JDBC drivers for DB2 and MySQL are not included with the Jive SBS Platform RPM. The platform will attempt to install these drivers when you install the platform, but in some cases automated installation will not be possible. If the RPM cannot download the appropriate drivers, an error

message will be printed to the console similar to “No suitable driver found for...”. For example, in the following example, the MySQL database driver is not found by the runtime system:

```
[1616][jive@targethost:~]$ ./bin/migration/bin/migration ./migration.properties
java.sql.SQLException: No suitable driver found for >jdbc:mysql://soul:3306/csc252
at java.sql.DriverManager.getConnection(DriverManager.java:602)
at java.sql.DriverManager.getConnection(DriverManager.java:185)
at com.jivesoftware.migration.schema.DbUtil.getConnection(DbUtil.java:133)
at com.jivesoftware.migration.etl.DbConfig.getInputConnection(DbConfig.java:161)
at com.jivesoftware.migration.etl.DbConfig.getVersion(DbConfig.java:263)
at com.jivesoftware.cli.Main.exportBlobs(Main.java:198)
at com.jivesoftware.cli.Main.run(Main.java:100)
at com.jivesoftware.cli.Main.main(Main.java:66)
Exception in thread "main" java.lang.RuntimeException: java.lang.NullPointerException
at com.jivesoftware.cli.Main.main(Main.java:68)
Caused by: java.lang.NullPointerException
at com.jivesoftware.migration.etl.DbConfig.getVersion(DbConfig.java:273)
at com.jivesoftware.cli.Main.exportBlobs(Main.java:198)
at com.jivesoftware.cli.Main.run(Main.java:100)
at com.jivesoftware.cli.Main.main(Main.java:66)
```

To resolve this issue, download the MySQL JDBC driver and place the JAR file for the driver in the jive user’s bin/migrate/lib directory. The following sequence demonstrates downloading the file and unpacking to the /usr/local/jive/bin/migration/lib directory of the target host.

```
[1632][jive@targethost:~]$ wget http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-
java-5.1.7.tar.gz/from/http://mysql.llarian.net/
--16:32:11-- http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.7.tar.gz/
from/http://mysql.llarian.net/
Resolving dev.mysql.com... 213.136.52.29
Connecting to dev.mysql.com|213.136.52.29|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://mysql.llarian.net/Downloads/Connector-J/mysql-connector-java-5.1.7.tar.gz
[following]
--16:32:13-- http://mysql.llarian.net/Downloads/Connector-J/mysql-connector-java-5.1.7.tar.gz
Resolving mysql.llarian.net... 209.221.142.116, 2001:5d8:11::14
Connecting to mysql.llarian.net|209.221.142.116|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8640154 (8.2M) [application/x-gzip]
Saving to: `mysql-connector-java-5.1.7.tar.gz'
100%[=====>] 8,640,154 5.11M/s
in 1.6s
16:32:14 (5.11 MB/s) - `mysql-connector-java-5.1.7.tar.gz' saved [8640154/8640154]
[1632][jive@targethost:~]$ tar xzf mysql-connector-java-5.1.7.tar.gz
[1632][jive@targethost:~]$ mv mysql-connector-java-5.1.7/
build.xml docs/ README
CHANGES EXCEPTIONS-CONNECTOR-J README.txt
COPYING mysql-connector-java-5.1.7-bin.jar src/

[1632][jive@targethost:~]$ mv mysql-connector-java-5.1.7 bin/migration/lib/
mv: cannot move 'mysql-connector-java-5.1.7' to `bin/migration/lib/mysql-connector-java-5.1.7':
Permission denied
[1632][jive@targethost:~]$ chmod 755 bin/migration/lib
[1632][jive@targethost:~]$ mv mysql-connector-java-5.1.7 bin/migration/lib/
[1632][jive@targethost:~]$ ./bin/migration/bin/migration ./migration.properties
```

The preceding example downloads the MySQL JDBC driver and using the “tar” command uncompresses the file. The tar archive uncompresses to a directory which can be moved directly to the /usr/local/jive/bin/migration/lib directory.

Note that in the preceding example, it is necessary to change the permissions on the /usr/local/jive/bin/migration/lib directory to move the expanded tar.gz file to the lib directory. This directory is read-only by default on an RPM-managed host.

Transferring jiveHome

When you upgrade from version 2.5.x to Jive SBS, you need to transfer the jiveHome directory from its existing form and location to the new ones. This ensures that the information your installation needs will be correctly suited to the new platform. For example, as of version 3, the contents of the former

jiveHome directory are no longer all contained in a single directory. Use the tools described here to make that transfer easier.

All previous versions of this software required a locally-mounted writeable directory that a Jive deployment could use for instance-specific storage. This directory must be migrated to the new Jive-managed environment controlled by the RPM installation.

Export the Existing Jive Home Directory

To make moving the home directory easier, Jive SBS includes an export tool you can execute on the current host where the home directory resides. This export tool is packaged as a Java JAR file that you can execute directly from any Java 1.5-compliant Java Runtime Environment. This tool is intended to be executed with the JRE used to host the existing instance.

You'll find the tool in an RPM installation on the target host at:

```
/usr/local/jive/bin/export/export.jar
```

1. Locate the export tool in an RPM installation on the target host at:

```
/usr/local/jive/bin/export/export.jar
```

2. Copy this file to the source host containing the existing home directory:

```
[1638][jive@targethost:~/bin/export]$ scp export.jar root@oldhost:
root@oldhost's password:
export.jar
```

3. From the source machine, use the export tool to archive the existing home directory (formerly referred to as jiveHome) from the file system.

Note: Before invoking the following commands, be sure to stop the current instance (stopping an existing instance will depend on the current application server).

You should invoke the export tool from a command shell (for Windows cmd.exe, for Unix or Linux any standard shell). Commonly, the java command will be on the path of the shell executing the command. If it's not, prefix the following examples with the full path to a Java 1.5 JRE on the local system.

To invoke the export tool, pass the "-jar <path_to_export.jar>" option to the java command in the working shell, as well as the path to the current home directory.

For example, the following commands demonstrate how to export the home directory at "/usr/local/jive/jiveHome" when the "java" command is on the local user's path:

```
root@oldhost ~/opt/jive/home $ cd /home/eonnen/opt/jive/home/
root@oldhost ~/opt/jive/home $ java -jar export.jar csc csc.zip
```

For detailed export instructions, the "--help" option may be passed to the command which results in the following output:

```
root@oldhost ~ $ cd opt/jive/home/
root@oldhost ~/opt/jive/home $ java -jar export.jar --help
usage: java -jar export.jar [options] jive_home archive_output
```

Where jive_home is the path to the existing jive home directory and archive_output is the path where the export archive will be written.

-a,--all	Include all artifacts in the existing jive home.
-c,--caches	Include local attachment and plugin caches [default disabled].
-h,--help	Display this message.
-i,--indexes	Include calculated search indexes [default disabled].
-l,--logs	Include log files [default disabled].
-o,--overwrite	Overwrite an existing artifact if one exists [default disabled].
-v,--verbose	Be verbose.

```
-x,--no-exports Do not include database exports [default enabled].
-z,--no-themes Do not include themes [default enabled].
```

Home directory is required.

In most cases, you should use the "--all" option. In large deployments, the size of index files may be too large to copy practically across a network. If this is the case, you can specify the "--indexes" option to exclude the index files from the export.

Import the Contents of the Jive Home Directory

After you complete the preceding export step, you should copy the exported file to the installation host where the Jive SBS platform was installed.

1. Before using the export, make sure the target application is stopped. In a default Jive SBS Platform RPM installation, a single application named "sbs" will be running. Stop this application by using the "appstop" command as the jive user.

```
[root@targethost ~]# su - jive
[1055][jive@targethost:~]$ appstop --verbose sbs
Stopping sbs...
Executing /usr/local/jive/applications/sbs/bin/manage stop
sbs stopped successfully.
```

2. After stopping the application, back up the existing home directory installed by the RPM. Unzip the exported home directory to the application's home directory. In a standard RPM installation, the application's home directory will be "/usr/local/jive/applications/sbs/home". In the following example, the csc.zip archive of the exported home is located in the user's home directory and is readable by the jive user.

```
[root@targethost ~]# su - jive
[1104][jive@targethost:~]$ appstop --verbose sbs
Stopping sbs...
Executing /usr/local/jive/applications/sbs/bin/manage stop
sbs stopped successfully.
sbs stopped successfully.
[1104][jive@targethost:~]$ cd applications/sbs/home/
[1104][jive@targethost:~/applications/sbs/home]$ cd ~
[1104][jive@targethost:~]$ cd applications/sbs
[1105][jive@targethost:~/applications/sbs]$ ll
total 16K
lrwxrwxrwx 1 jive jive 49 Jan 20 15:52 application
-> /usr/local/jive/applications/template/application
drwxr-x--- 2 jive jive 4.0K Jan 20 15:52 bin
drwxr-x--- 2 jive jive 4.0K Jan 20 15:51 conf
drwxr-x--- 16 jive jive 4.0K Jan 22 11:02 home
-rw-r--r-- 1 jive jive 694 Jan 20 15:20 README
[1105][jive@targethost:~/applications/sbs]$ mv home home.bak
[1106][jive@targethost:~/applications/sbs]$ mkdir home
[1107][jive@targethost:~/applications/sbs]$ cd home
[1107][jive@targethost:~/applications/sbs/home]$ unzip ~/csc.zip
Archive: /usr/local/jive/csc.zip
[1108][jive@targethost:~/applications/sbs/home]$ ll
total 32K
drwxr-xr-x 2 jive jive 4.0K Jan 22 11:08 conf
drwxr-xr-x 2 jive jive 4.0K Jan 22 11:08 database
drwxr-xr-x 2 jive jive 4.0K Jan 22 11:08 geoip
-rw-r--r-- 1 jive jive 958 Jan 22 2009 jive.license
-rw-r--r-- 1 jive jive 1.7K Jan 22 2009 jive_startup.xml
drwxr-xr-x 3 jive jive 4.0K Jan 22 11:08 resources
drwxr-xr-x 2 jive jive 4.0K Jan 22 11:08 spelling
drwxr-xr-x 3 jive jive 4.0K Jan 22 11:08 themes
```

Migrating Data from an Unsupported DBMS

Jive provides a migration utility that is designed to ease the process of migrating from a DBMS that's no longer supported to one that is. The utility copies data from the old schema and (no longer

supported) DBMS to a new database on a supported DBMS. Along the way, the tool manages occasional incompatibilities between DBMS systems.

You don't need to use this tool if you're upgrading from a DBMS that's supported for version 3. (This includes PostgreSQL and Oracle; see the *System Requirements* for more information.)

The easiest way to use the command-line migration utility is by specifying a properties file that contains the parameter values you'll need. You'll find the utility in the migration directory of your application distribution. You'll find the utility on the target computer at the following path:

```
/usr/local/jive/bin/migration/bin
```

Syntax

```
./migration_path/to/filename.properties
```

Parameters

Except where noted, all of the parameters are required.

Parameter	Description
inputUrl	The JDBC URL used to connect to the source database.
inputDriver	The Java class name of the JDBC driver used to connect to source database.
inputUser	Source database username.
inputPassword	Source database password.
outputUrl	The JDBC URL to connect to the target database.
outputDriver	The JDBC driver used to connect to target database.
outputUser	Target database username.
outputPassword	Target database password.
workDir	Directory where the migration utility will store the binary data, logs, processing scripts during the migration.
Optional Parameters	
customSchemas	Comma-separated list of files that contain definitions of tables that are not part of the Jive SBS schema. The table definitions must be in the schema format used by Jive SBS. See plugin documentation for additional information on how to write a schema file.
useSqlDump	Specifies that all of the data should be put into an XML file, then used later for importing. Values are <code>true</code> or <code>false</code> .
steps	<ul style="list-style-type: none">• A comma-separated list of steps to perform during migration. Use this property to change the default steps performed in a standard migration.• Use one or more of the following values:<ul style="list-style-type: none">• <code>exportBlobs</code>: Export the blob data to the work directory.• <code>exportSql</code>: Export the data to a set of XML files.• <code>createTargetSchema</code>: Create the schema and custom tables in the target database.• <code>writeEtlis</code>: Write ETL (extract, transform and load) scripts to the working directory.• <code>runEtlis</code>: Execute ETL scripts from the working directory.• <code>validate</code>: Validate the migrated data against the source database.
threadCount	Number of threads. Default is 1.

Migration Example

Here's an example of the properties you'll need as you might set them in a properties file:

```
inputUrl=jdbc:mysql://soul:3306/csc252
inputDriver=com.mysql.jdbc.Driver
inputUser=csc252
inputPassword=password
outputUrl=jdbc:oracle:thin:@oracle-utf8:1521:ORAUTF8
outputDriver=oracle.jdbc.driver.OracleDriver
outputUser=KUATO
outputPassword=Kuato
workDir=/usr/local/jive/var/work/migrate
steps=exportBlobs, exportSql, writeEtl
```

The following sample demonstrates running the migration tool from the jive user's home directory with a configuration file of migration.properties:

```
[1600][jive@targethost:~]$ ls -l
total 2408
drwxr-xr-x  4 jive jive   4096 Jan 22 14:05 applications
drwxr-xr-x  6 jive jive   4096 Jan 22 15:05 bin
-rw-r--r--  1 jive jive 2413509 Jan 22 11:01 csc.zip
drwxr-xr-x  5 jive jive   4096 Jan 22 14:05 etc
drwxr-xr-x 15 jive jive   4096 Jan 22 14:05 httpd
drwxr-xr-x  7 jive jive   4096 Jan 22 15:04 java
-rw-r--r--  1 jive jive   332 Jan 22 15:52 migration.properties
drwxr-xr-x  8 jive jive   4096 Jan 22 14:13 postgres
drwxr-xr-x  6 jive jive   4096 Jan 22 14:13 python
drwxr-xr-x  2 jive jive   4096 Jan 22 15:04 sbin
drwxr-xr-x 10 jive jive   4096 Jan 22 15:04 tomcat
drwxr-xr-x  7 jive jive   4096 Jan 22 14:05 var

[1600][jive@targethost:~]$ cat migration.properties
inputUrl=>jdbc:mysql://soul:3306/csc252
inputDriver=com.mysql.jdbc.Driver
inputUser=csc252
inputPassword=password
outputUrl=jdbc:oracle:thin:@oracle-utf8:1521:ORAUTF8
outputDriver=oracle.jdbc.driver.OracleDriver
outputUser=KUATO
outputPassword=Kuato
workDir=/usr/local/jive/var/work/migrate
steps=exportBlobs, exportSql, writeEtl

[1818][jive@targethost:~]$ ./bin/migration/bin/migration ./migration.properties
log4j:WARN No appenders could be found for logger (com.jivesoftware.migration.schema.DbUtil).
log4j:WARN Please initialize the log4j system properly.
Number of open connections:
Connection URL:jdbc:mysql://10.61.130.77:3306/csc252, count:1
Number of open connections:
Connection URL:jdbc:mysql://10.61.130.77:3306/csc252, count:0
Number of open connections:
Connection URL:jdbc:mysql://10.61.130.77:3306/csc252, count:1
Number of open connections:
Connection URL:jdbc:mysql://10.61.130.77:3306/csc252, count:0
Number of open connections:
Connection URL:jdbc:mysql://10.61.130.77:3306/csc252, count:1
Wrote file:/usr/local/jive/var/work/migrate/jiveAttachData/1001.bin
Wrote file:/usr/local/jive/var/work/migrate/jiveAttachData/1002.bin
Wrote file:/usr/local/jive/var/work/migrate/jiveAttachData/1003.bin
Wrote file:/usr/local/jive/var/work/migrate/jiveAttachData/1004.bin
Wrote file:/usr/local/jive/var/work/migrate/jiveAttachData/1005.bin
...
jiveAnswer:2000
jiveDocTypeElement:2001
jiveDocBodyVersion:2002
jiveDraftImage:2003
jiveAttachVersion:2004
jiveImageVersion:2005
jiveWFCurrStepPrev:2006
jiveUsrRelGrApr:2007
jiveUsrRelGrNtf:2008
jiveUserRel:2009
```

```
jiveUsrRelListMap:2010
jiveCollaboration:2011
jiveSGroupMember:2012
jivePTracker:2013
jiveSectionElement:3000
Step: writeEtlS completed in:0
Memory used: 18764656
```

Migration Troubleshooting

The migration utility has been extensively tested on a variety of database and operating system platforms. Still, it is possible that a specific scenario may have been overlooked. This FAQ will try to answer some of the issues that you may encounter during the migration.

What are the disk space requirements for migration?

The utility needs approximately 1.5-2 times the size of the source database of free, local disk space. So if you have a database with 10 GB of data, ensure that you have at least 15 GBs of disk space available on the server on which you are running the utility.

How long does it usually take to do the migration?

The time taken will be completely dependent on hard-disk and network speeds and the current load on source and target databases. On a fast disk and reasonably fast network, we have been able to migrate data at the rate of 15-18GB/hour with validation.

Can the migration be broken into multiple steps?

Yes, but it is recommended you run all of the steps that are required in every run. The migration internally is broken into multiple steps:

1. Export the data from Source database: Steps required are ***exportBlobs, writeEtlS***
2. Import the data to the target database: Steps required are ***createTargetSchema, runEtlS***
3. Validate the imported data: Steps required are ***validate***

The ***validate*** step will generally be the most time consuming and may be excluded, though it is recommended that you always run it. If you exclude this step, check the ***missing_fk.log*** and ***truncated.log*** files for errors that may have occurred during migration.

Where are the migration logs stored?

All the logs files are stored in the *workDir*.

Why might the migration fail?

There are two reasons why data may not migrate completely from the source to the target database.

- Foreign Key Constraints: Some records in the source database may have constraints that point to records that have been deleted. The migration utility will not import these record. When it finds a record that has Foreign Key constraints that do not exist, a entry will be generated in *missing_fk.log*. In most migrations, the total number of records that have missing foriegn keys should be around 0.5-1% of the total number records in the database.
- Data truncation: It is possible that size of certain text/varchar/char columns in the target database may not be large enough to accommodate data from same columns in the source database. The migration utility checks the size of every text/varchar/char column before inserting the data into the target database. If the size of the text to be inserted exceeds the column limit, it will truncate the text to fit the column size. In addition, it will create an entry in the ***truncated.log*** log file for every record that has been truncated with the original and truncated text.

How do I know whether the migration succeeded?

At the end of the migration, a summary.html file is generated, that will give a quick summary of the migration. The file contains an entry for each table that is migrated. The entry will contain row counts, count of rows that differ (the delta), and a link to the log file that contains the diff between the source and target. Below is an example of the summary.html:

DB MIGRATION REPORT

Fri Jan 09 11:19:59 PST 2009

Source Database Name:	jdbc:mysql://10.61.130.171:3306/jive
Destination Database Name:	jdbc:postgresql://10.61.130.171:5432/jive
Migration started:	Fri Jan 09 11:19:59 PST 2009
Migration ended:	Fri Jan 09 13:43:17 PST 2009
Total number of tables:	168

Summary

TableName	SourceDir	DestinationDir	Delta RowCount	LogFile
jiveContainerAprvr		1	0	/tmp/db/bs_ver_25x/jiveContainerAprvr.diff.log
jiveCommunityProp		56	1	/tmp/db/bs_ver_25x/jiveCommunityProp.diff.log

Back Up Your jiveHome Directory

The jiveHome directory is the place where the application stores a number of things about your environment. The database connection information is stored there, as well as logs, cached attachments, your license file, and the local system database files (if used). You should back up this directory before upgrading.

Back Up Your Database

You should back up your database before you upgrade. For now, the best way to manage database backups is to follow the recommendations of your DBA or the recommendations of your database software. There are a number of tools built in to various databases. Here are a couple examples:

- MySQL — Use the "mysqldump" tool
- Postgres — Use the "pg_dump" tool

There are many tools for each database; pick one that suits your environment.

Remove Plugins Before Upgrading

Before starting your upgrade be sure to remove any plugins you've installed. For those plugins that aren't compatible with the version you're upgrading to, you'll need to separately upgrade your plugin code (or get upgraded versions of the plugins from their developer), then install the upgraded versions after you've upgraded to this version.

Post-Installation Tasks

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 Run Book, these operations are common to a new installation, but generally not for day to day operation of this platform.

Using Commands to Work with Your Managed Instance

Jive SBS includes several command-line tools you can use to perform maintenance tasks with your managed instance. With these tools, you can start and stop the application, upgrade the application, collect information needed by Jive support, and more.

You'll find these documented in the [Application Management Command Reference](#) (page 18) .

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.
- After making the above changes, restart the Jive HTTPD server using the command `"/etc/init.d/jive-httpd restart"`.

Note that if a 3rd 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 that 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 SBS platform database instead, choosing to use an RDBMS that is controlled by an internal corporate IT group. In this case, the local database should be disabled. To disable the application database, as the root user, execute the following script:

```
The following terminal output demonstrates deactivation and of the Jive database service:  
[root@targethost ~]# /etc/init.d/jive-database deactivate  
Jive Database deactivated.  
[root@targethost ~]# /etc/init.d/jive-database activate  
Jive Database activated. The database will start automatically on the next system restart.
```

Note that disabling the database does not stop the service if it is running. Likewise, re-enabling the database does not start the database service.

Installation Known Issues

Poor Performance When Citrix Is Used

Jive SBS will not work properly when accessed through Citrix. This is because Citrix doesn't support AJAX, a technology the application uses for accessing the server asynchronously in the background.

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

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
	<code>--version</code>	Show program's version number and exit.
<code>-h</code>	<code>--help</code>	Show help message and exit.

HTTPD Options

Configures the HTTPD integration configuration options.

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

Application Options

Configures general application server options.

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

Short	Long	Description
'	--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

```
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
-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

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

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
	--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

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

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.

Short	Long	Description
-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

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

Backs up the application database. To create a full backup, omit the options. For more on backups, see the *Operations Cookbook*.

```
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

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

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

upgrade

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

Short	Long	Description
-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