

Installing on Solaris



Contents

Installing on Solaris	2
Overview	2
Before You Install.....	2
Installing the Package	5
Prerequisites.....	5
Installing the Software.....	6
Troubleshooting Installation.....	8
Upgrading from Prior to 3.0	9
Before You Upgrade.....	9
Upgrading the Software.....	10
After You Upgrade.....	10
Troubleshooting Upgrade.....	10
Transferring jiveHome.....	11
Migrating Data from Another DBMS.....	13
Upgrading a Package Installation (Version 3.0 and Later)	17
Upgrading the Package.....	17
After You Upgrade.....	18
Post-Installation Tasks	18
Installation Known Issues	18

Installing on Solaris

Instructions for installing the Jive SBS package using the package manager.

Overview

The distribution you received is probably all you need to install the application onto a supported operating system.

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 operating system should have a package manager (such as the RPM manager on Linux or pkgadd on Solaris) already installed. When installing, you'll invoke that manager to install the package that contains Jive SBS.
- Install the application package 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 package manager.
- The application includes (and installs) a PostgreSQL DBMS, but you needn't use it. If you have a separate 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 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 (*Linux, Solaris*), *Application Management Commands*, 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.

Note: If you're upgrading to a version whose database schema is different from the previous version, the application will automatically detect the difference after you've upgraded. When you next start the application and navigate to the admin console, you'll be prompted to start the database upgrade. This is a necessary step before you can start using the application after upgrade. You don't need to run any database-related upgrade scripts.

- 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, 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 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*. 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.
- Confirm that the drivers required for the DBMS you'll be using are installed. Due to licensing restrictions, Jive is unable to ship and install all of the drivers for supported DBMSes. Use the table below to determine whether the driver you need is included, and install the driver if it's not.

To install the driver, place the driver's JAR file in the application file system where it will be on the classpath, such as at `/usr/local/jive/applications/template/application/WEB-INF/lib`.

DBMS	Driver Included	Action Needed
Oracle	OCI driver required, but not included. (Thin driver is included for evaluation purposes only and should not be used under load.)	For more information on installing the OCI driver, see the Oracle <i>Instant Client page</i> . Please consult the Oracle documentation to choose the version of the JDBC driver best for you.
PosgreSQL	Required driver included.	None.
MySQL	Required driver not included.	You'll need to install the correct driver.
SQL Server	Required driver not included.	You'll need to install the correct driver.

DBMS Issues and Notes:

MySQL:

MySQL Character Encoding Issues

MySQL does not have proper Unicode support, which makes supporting postings in non-Western languages difficult. However, the MySQL JDBC driver has a workaround which you can enable by adding `<mysql><useUnicode>true</useUnicode></mysql>` to the `<database>` section of your `jive_startup.xml` file. When using this setting, you should also set the Jive character encoding to utf-8 in the Admin Console.

MySQL 4.1 introduced better support for character encodings than previous versions. This functionality assigns a default character encoding to the database and its tables and columns. It's best to set the default character encoding for your database before installing the Jive schema so that you can be sure that you will not have encoding problems in the future. After creating your database, execute the following line in the MySQL console:

```
ALTER DATABASE <database name> DEFAULT CHARACTER SET <character set>;
```

For example, if you plan on using UTF-8 in Clearspace, you should enable the JDBC driver workaround mentioned above and then execute this line in the MySQL console:

```
ALTER DATABASE <database name> DEFAULT CHARACTER SET utf8;
```

MySQL Max Attachment Size Issues

You can fix the max attachment size problem on a MySQL server by following the directions here: <http://dev.mysql.com/doc/refman/5.0/en/packet-too-large.html>

The easiest way to change the MySQL server setting on Windows is to add a line to your `my.ini` file (you'll find this file in a location such as `C:\database\mysql\5.0.19`); on Linux, look for the `my.cnf` file instead. Add the following line after the `[mysqld]` section heading:

```
max_allowed_packet = 500M
```

After you add the line, restart MySQL.

Adjust the InnoDB Buffer Pool Size

When you have the database running on a dedicated server, you should increase the InnoDB buffer pool size from the default (8 MB) to up to 80 percent of the computer's available memory. If the computer has 2 GB of RAM or less, you should think about setting the buffer to something less than 80 percent to ensure that the operating system has enough memory to avoid swapping.

See the *MySQL documentation* for more on configuration.

SQL Server:

Case Sensitivity Issues

Clearspace requires a DBMS that matches strings with case sensitivity; SQL Server is case insensitive by default. You can set SQL Server database and/or server collation (the character set used to store data in text fields) to support case sensitivity. If database collation is not specified, server collation will be used. The collation value you'll need to use is `SQL_Latin1_General_CP1_CS_AI`.







You'll find more information on adjusting SQL Server settings in the SQL Server documentation:



Setting and Changing the Server Collation

Setting and Changing the Database Collation

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 platform package to the target host.		
2 Install the 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 Package

In This Section

- You'll need to have SSH access to the target host to copy the distribution there and execute commands.

What You'll Need

To install Jive SBS, 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 operating system installed. See the *System Requirements* for supported versions.
- A copy of the application package. You'll also need SSH access to the host computer so you can copy the package 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.

Prerequisites

The Jive SBS Solaris package depends on additional third-party packages for certain functionality, including as cryptographic libraries and system tools. The following libraries must be installed prior to installing the Jive SBS package:

Component Dependency	Library	What It Does
Platform Scripting	coreutils-7.4-sol10-sparc-local	Scripting infrastructure, more functional than Solaris versions of common commands
Platform Scripting	findutils-4.4.1-sol10-sparc-local	Improved version of find command
Platform Scripting	gmp-4.2.1-sol10-sparc-local	coreutils dependency
Platform Scripting/Apache2/PostgreSQL	libgcc-3.4.6-sol10-sparc-local	Core GNU c library
Platform Scripting	libiconv-1.11-sol10-sparc-local	popt dependency

Component Dependency	Library	What It Does
Platform Scripting	libintl-3.4.0-sol10-sparc-local	logrotate dependency
Apache2	logrotate-3.7.6-sol10-sparc-local	Log rotation capabilities
Apache2/PostgreSQL	popt-1.14-sol10-sparc-local	Command line option parsing capabilities
PostgreSQL	readline-5.2-sol10-sparc-local	PostgreSQL command line dependency

Each of these libraries is available for download from *Sunfreeware*. As a convenience, Jive provides the `solaris-host.sh` script to download the required dependencies. This script will download the correct versions of the correct libraries. Note that the script must be run from a machine with internet access and with the `wget` command available.

Installing the Software

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

1. Ensure that that application database has been created as described in database prerequisites.
2. From the command line, access the target host as root.
3. If you haven't yet done so, copy the application package to the target host.
4. Unzip the package file using the `gunzip` command.

```
bash-3.00# /usr/bin/gunzip JiveSBS-3.0.5-SunOS-5.10-sparcv9.gz
```

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

To set these, use the `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.	Debugging information isn't displayed.
JIVE_APPLICATION_STARTUP_SERVICE	Prevents 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.

6. As root, execute the `pkgadd` command with the `-d` option and the decompressed package file from the previous step as an argument. (Note that your copy of the package file -- here, `JiveSBS-3.0.5-SunOS-5.10-sparcv9` -- might have a slightly different name.)

When executing the command, the system will prompt for the package to process, enter "all" or press return to select the default of all.

The `pkgadd` command will then prompt that the `/usr/local/jive/httpd/bin/suexec` file will be installed with `setuid` and/or `setgid` permissions. Enter 'y' to accept.

Finally, pkgadd will ask if it should continue with installation of the JiveSBS package. Enter 'y' to accept.

Console output should appear similar to the following:

```
bash-3.00# pkgadd -d JiveSBS-3.0.5-SunOS-5.10-sparcv9

The following packages are available:
  1 JiveSBS      JiveSBS for SunOS 5.10 sparcv9
                    (sun4u) 3.0.0

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]:

Processing package instance <JiveSBS> from </root/JiveSBS-3.0.5-SunOS-5.10-sparcv9>

JiveSBS for SunOS 5.10 sparcv9(sun4u) 3.0.0
Jive Software, Inc.
## Executing checkinstall script.
## Processing package information.
## Processing system information.
  1 package pathname is already properly installed.
## Verifying package dependencies.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

The following files are being installed with setuid and/or setgid
permissions:
  /usr/local/jive/httpd/bin/suexec <setuid root>

Do you want to install these as setuid/setgid files [y,n,?,q] y

This package contains scripts which will be executed with super-user
permission during the process of installing this package.

Do you want to continue with the installation of <JiveSBS> [y,n,?] y
Installing JiveSBS for SunOS 5.10 sparcv9 as <JiveSBS>

## Executing preinstall script.
Preparing clean installation.
## Installing part 1 of 1.
/etc/jive/jiverc
/usr/local/jive/.bash_profile
/usr/local/jive/applications/template/README
/usr/local/jive/applications/template/application/404.jsp
/usr/local/jive/applications/template/application/500.jsp
/usr/local/jive/applications/template/application/META-INF/MANIFEST.MF
... (omitted for brevity)
## Executing postinstall script.
Executing Jive post-install configuration.
Marking all upgrades as complete.
Sun Microsystems Inc.  SunOS 5.10      Generic January 2005
Initializing database for first use.
Sun Microsystems Inc.  SunOS 5.10      Generic January 2005
Starting Jive System Database.
Staging Jive Application.
Sun Microsystems Inc.  SunOS 5.10      Generic January 2005
Validating configuration.
Staging application from template: /usr/local/jive/applications/template
Linking application to master binary at '/usr/local/jive/applications/template/
application'.
System memory requirements invalid. Application will likely run out of memory during
normal usage.
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.
Jive post-install configuration complete.

Installation of <JiveSBS> was successful.
```

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

7. If you'll be using a MySQL database, copy its driver into the application's class path. Due to licensing restrictions, Jive is unable to ship all drivers for databases it supports. In particular, the JDBC driver for MySQL is not included. Download the MySQL JDBC driver and place its JAR file in the application file system where it will be on the classpath. In an uncloned instance, this will be at: /usr/local/jive/applications/template/application/WEB-INF/lib. In a cloned instance, it will be /usr/local/jive/applications/<application_name>/application/WEB-INF/lib. After you copy the driver into the classpath, you'll need to restart the application service.
8. 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 package 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](#).

Troubleshooting Installation

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

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.

```
System memory requirements invalid. Application will likely run out of memory during normal usage.
```

Despite this warning, the package does install correctly. However, be sure to see the log file at '/usr/local/jive/var/logs/sbs.out'. Upon further examination, the contents of this log file might indicate:

```
bash-3.00# 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
```

If you get the warning about memory, you should add more memory to the host machine (at least 3 GB).

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 package file name, as you did when installing. Instead, you give the logical name by which the package manager now knows the application: JiveSBS. Here's an example using the `pkgrm` command:

```
pkgrm JiveSBS
```

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

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

Upgrading from Prior to 3.0

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 package manager.

The following steps are written for the scenario where an existing installation resides on a separate host from the installation of the package. If the package 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 package 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 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 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 package 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 and back up your database before doing an upgrade.

Before You Upgrade

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

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.

Upgrading the Software

1. Ensure that you've installed the platform package to the new host.
2. Verify that you've backed up your jiveHome directory. You should also have a backup of your existing database.
3. Remove plugins 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 for more information.
5. Stop the default SBS application instance. By default, the Jive SBS Platform package 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 for instructions.
7. With a supported web browser, navigate to `http://<hostname>/`, where hostname is the DNS resolvable name of the server where the package was installed. There, you can proceed with the upgrade using the upgrade console.
8. After you've finished 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!

After You Upgrade

Be sure to see the post-installation tasks for suggested steps to take after you install the software.

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 package. 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 package 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-
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 a 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 package installation.

Export the Existing jiveHome 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 a package installation on the target host at:

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

1. Locate the export tool in a package 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 jiveHome 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 package 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 with the package. Unzip the exported home directory to the application's home directory. In a standard package 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 Another 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.

Note: If you're also migrating an existing jiveHome directory as part of upgrade, you might need to re-run the application setup tool. This might be necessary in order to specify connection parameters for the database you're migrating to. You'll know that you need to do this if, when you start the application after

migrating both database and Jive home, you're not greeted by the setup tool. For information on running the setup tool, see *Setting Up the Community*.

You don't need to use this tool if you're upgrading from a DBMS that's supported for version 3. (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:oci:@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:oci:@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

Upgrading a Package Installation (Version 3.0 and Later)

With a new package distribution in hand, you can easily upgrade your existing package to the later one.

What You'll Need

To upgrade using the package, you'll need the following:

- An existing installation of the Jive SBS package.
- A copy of the application package you're upgrading with. You'll also need SSH access to the host computer so you can copy the package there for installation.
- Root access to the host where the installation is performed, commonly via SSH.

Upgrading the Package

Rather than using an explicit package upgrade, you simply make a backup of your existing instance, remove the existing instance, then install the new one.

1. Make a backup of your existing instance.
2. From the command line, access the target host as root.
3. Upgrade the package using the same command you used to install it:

```
bash-3.00# pkgadd -d JiveSBS-3.0.6-SunOS-5.10-sparcv9
```

Under normal circumstances, the upgrade output will include a list of files and conclude by telling you that installation was successful. If you previously left the default SBS application installed with the platform (perhaps because you're using it), you'll receive a warning that the name "sbs" is already in use. That's because the package is designed to install the default application when it runs. If that application already exists, the package manager will notice the presence of an application with the same name as the default and will bypass this step, not installing the default application. The platform will still be upgraded and the SBS application will remain intact.

4. You're finished.

After You Upgrade

Be sure to see the post-installation tasks for suggested steps to take after you install the software.

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 (*Linux, Solaris*), 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*.

Enabling SSL Encryption

Be sure to see the Operations Cookbook for information on *enabling SSL encryption*.

Disabling the Local Jive System Database

The Operations Cookbook describes how you can *disable the local Jive system database* if you won't be using it (because you have your own DBMS).

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.