

Operations Cookbook

jive

Contents

Operations Cookbook	2
Enabling SSL Encryption	2
Disabling the Local Jive System Database	2
Changing the Configuration of an Existing Instance	3
Performing a Jive System Database Backup	4
Backup and Storage Considerations	4
Storage Reliability.....	4
Storage Monitoring.....	4
System Backups.....	5
System Database Credentials.....	5
Using an External Load Balancer	5
Enable Application Debugger Support	5
Setting Up a Local Cluster	6

Operations Cookbook

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

As opposed to the Run Book (*Linux, Solaris*), these operations are common to a new installation, but generally not for day-to-day operation of the platform.

Enabling SSL Encryption

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

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

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

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

Disabling the Local Jive System Database

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

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

On Linux

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

On Solaris

```
bash-3.00# svcadm -v disable -s application/jive/database
svc:/application/jive/database:default disabled.
```

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

Changing the Configuration of an Existing Instance

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

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

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

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

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

Prior to edit, the instance file will look similar to the following.

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

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

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

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

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

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

Performing a Jive System Database Backup

Jive SBS-managed databases will perform automatic backups as described in *Auto Backups*. In some situations, for example prior to an upgrade of the package, it may be desirable to perform a full database backup manually.

To manually perform a full backup of the managed database, execute the *dbbackup* script as the jive user.

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

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

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

Backup and Storage Considerations

Storage Reliability

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

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

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

Storage Monitoring

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

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

System Backups

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

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

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

System Database Credentials

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

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

Using an External Load Balancer

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

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

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

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

Enable Application Debugger Support

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

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

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

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

Setting Up a Local Cluster

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

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

```
apprm sbs
```

3. Use the `appadd` command to add two new application instances that will make up your cluster.
 - a. You can add the first instance with default settings, specifying a context path:

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

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

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

4. Restart the `jive-httpd` service (*Linux, Solaris*).
5. Start the first instance with the `appstart` command.

```
appstart cluster1
```

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

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

7. Stop the first instance with the `appstop` command.

```
appstop cluster1
```

8. Change the `APP_CLUSTER_PORT` in the `/usr/local/jive/applications/cluster2/bin/instance` file to be the same as `cluster1`'s cluster port. For more on making this change, see [Changing the Configuration of an Existing Instance](#) (page 3) .

Note: In version 3.0.0, to support clustering you'll need to change configuration to override a default setting. [Edit the instance file](#) (page 3) for the instance to add the following (where `<addr>` is a unique address -- `224.224.224.224` might work.):

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

9. Start the second instance.

```
appstart cluster2
```

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

```
appstop cluster2
```

12. Start both instances.

```
appstart
```

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