



# Shepherd Installation and Configuration

*J&J Computer Consulting Shepherd Server Publications*

## Installing the Shepherd Server Engine

The Shepherd Server Engine comes packaged in a zip file for Windows and OS/2 or in a tar.gz file for Linux and BSD. Once the file has been downloaded to your system, create a directory for the Shepherd Server and extract the archive, maintaining directory information, into the new directory. The following sub-directories will be created:

- ◆ **doc** - Program documentation in PDF format.
- ◆ **dsp** - Directory Service Providers included with Shepherd.
- ◆ **log** - Empty. Log files can be directed here or elsewhere.
- ◆ **services** - Service files can be stored here or elsewhere.
- ◆ **setup** - Setup files.
- ◆ **samples** - Sample files in LDIF format.

For Windows and OS/2 users, if you create an icon for Shepherd or run Shepherd from the startup file, make sure that the installation directory is set as the working directory before starting Shepherd. For Unix users, you must set LD\_LIBRARY\_PATH to the Shepherd installation directory and set the installation directory as the working directory before running Shepherd. Check the Shepherd installation directory for sample shell scripts to start Shepherd automatically.

## Getting Started

By default, Shepherd comes configured to work with a JDB (J&J Computer Consulting Database) Directory Service Provider (DSP) that allows all directory information to be stored in two files on the Shepherd Server. The default configuration includes the file samples/min.ldif which will initialize the directory with a base of c=US, the server cn=Shepherd, c=US, and the administrator as cn=admin, c=US.

In many cases this setup is not appropriate. If so, there are several options that must be configured. The best place to start is the shepherd.ini file found in the installation directory. This file gives the Shepherd Server Engine as much information as it needs outside the directory for startup. The format used is similar to that found in Windows. Table 1 describes each of the sections and the attributes that can be set in the initialization file.

**Table 1: Shepherd Initialization File Specification**

<b>Section: [engine]</b>	The engine section of the initialization file provides information to the engine on the contents of the directory and how to access the directory.
<b>basedn</b>	The base distinguished name is the start of the directory tree for this server. Shepherd currently does not support the concept of root, so all directories must start somewhere below the root.
<b>cache</b>	The directory uses a cache to speed up access to directory information. This is separate and independent of any cache used by the DSP and is different in that access to objects in the Shepherd Directory cache will come back faster than data in the DSP cache due to fewer routines having to be called to get to the data. The cache should be given in bytes.
<b>server</b>	The server entry points Shepherd to information about itself in the directory. The server entry is used to lookup services that the server should start. Shepherd will support multiple servers running from a single directory with the aid of Shepherd.Replicate.
<b>user</b>	The distinguished name under which the engine should run. This can be any ShepherdAccount in the system.
<b>password</b>	The password for the ShepherdAccount specified in the user option.
<b>Section: [dsp]</b>	The dsp section tells Shepherd which Directory Service Provider to use, and optionally, passes attributes to the dsp during initialization. The only attribute in this section that is required in all cases is the package file. Other attributes are specific to the particular DSP. The DSP package file shows default values for possible options in its dsp section.
<b>package</b>	The name of the Directory Service Provider initialization file.

Once the shepherd.ini file is configured, the DSP package file must be configured. As with the shepherd.ini file, the DSP package file is a simple, sectioned attribute file. Table 2 describes the contents of the DSP package file.

**Table 2: DSP Package File Specifications**

<b>Section: [dll]</b>	The dll section identifies the file name of the DSP dynamic link library or shared library.
<b>name</b>	Sets the name of the dynamic link or shared library file. Typically, these are presented without the .dll or .so extension so as to maintain portability in configuration between platforms. The extension is assumed based on platform by Shepherd, however, the extension can be included if preferred.
<b>Section: [api]</b>	Tells the DSP the names of the API functions in the library. Not user configurable.
<b>Section: [userdata]</b>	This section gives samples of additional attributes that can be set in the shepherd.ini file. If the attributes are not set in shepherd.ini, the defaults shown in this section will be used instead. All parameters are DSP specific.
<b>Section: [dspdata]</b>	This section is for use by the DSP only. While these options can be changed, they should be changed only at the request of the DSP maker.

While the DSP package file rarely requires changes, it is important to view the [userdata] section to make sure all of the appropriate options have been set in the shepherd.ini file. Based on the DSP being used, there may be other configuration requirements to make Shepherd work with the directory as required. The JDB-DSP is described in the next section.

## Configuring the JDB Directory Service Provider

As stated previously, the JDB-DSP comes with enough data to allow most users to get started. However, if the configuration provided does not match your needs, the system should be re-configured outside Shepherd to prevent any future problems.

The following attributes are available to help in the configuration process:

- ◆ **datafile** - Name of the data file used to store the DSP data. The default is sds.jdb stored in the dsp sub-directory.
- ◆ **logfile** - Name of the log file used to store the DSP data. The default is sds.log stored in the dsp sub-directory.
- ◆ **cache** - This is the number of pages cached by the DSP in memory. A page size of 4096 bytes is used by the DSP, so the default of 49 pages requires approximately a 200k cache. When configuring this parameter, keep in mind that Shepherd uses a cache of directory objects independent of this cache. In that case, make sure you look at both values in addition to the size of Shepherd and its component libraries to determine memory usage.
- ◆ **atsetup** - Specifies a file to use for setting up additional attributes. A sample file is stored in setup\attrib.setup and contains all of the default attributes. The file consists of the name of the attribute, any amount of white space, and the type of the attribute. The type is not currently used, but you should attempt to use the model provided by attrib.setup to avoid future problems. This attribute should be specified only once to initialize the attribute definition database.
- ◆ **import** - This option will perform an import from the indicated LDIF file. Though Shepherd's CLI also has an import, this one works without any respect for security or structure and is the best way to initialize the base object and administrator for the directory. Once both objects have been created, future imports are more easily handled through Shepherd's import function.

To start the directory over with new data, create an LDIF file with the data you would like to load. Minimally, this needs to be the base object such as o=Your Organization with an ACL for Admin access granted to the administrator, an administrator object, and a ShepherdServer object. The file used to create the default configuration can be used as a model and is available in setup/min.ldif.

Set the import option in the dsp section of the shepherd.ini file to the LDIF file you created, set the atsetup attribute to setup/attrib.setup, and do one of two things with the file attribute. You can either leave it alone and delete the identified file (to remove the default configuration and create a new one), or change it to another file.

You can now start Shepherd by running shepherd.exe from the installation directory. If you have configured everything correctly, you will see no error messages and be given a Shepherd command prompt. By typing "dir" and pressing <Enter>, you will be able to see a list of objects in the base of your directory tree.

## Installing and Configuring Shepherd Services

Each protocol implementation such as HTTP, POP3, Chat, etc. is considered a "Service" that runs under the Shepherd Server Engine. Each of these Services comes with both a dynamic link library (shared library for Unix users), a Service initialization file (.svc extension), and possibly other files specific to the Service.

Services are packaged in zip or tar-gzipped files and can be unzipped in the Shepherd installation directory. This puts the services in the "services" sub-directory. The services sub-directory is only a recommended location for the files. Services can actually be placed on any hard drive accessible to the Shepherd system.

Included with each Shepherd system is the remote administration service, Shepherd.Admin. Shepherd.Admin allows a user with a Shepherd account to log on to the system and perform administrative functions on the server. Other Services are add-ons to the system.

To setup a Service, the appropriate initialization file should be available and should have its dll attribute set to the location of the Service dll. Services are configured to work from the services sub-directory of the Shepherd installation. If you move them, make sure that the dll attribute is set appropriately so that the Shepherd Engine can find the Service dynamic link or shared library.

Once these files are available, you need to add a ShepherdService object to your system for the appropriate Service. "An Introduction to Shepherd Directory Services" describes the ShepherdService object and its associated attributes in detail to help you create the appropriate object. There is also an admin.ldif file in the setup directory that shows a default configuration of Shepherd.Admin. This file can be modified to suit your needs and imported with the Shepherd CLI import command.

## Using the Shepherd Command Line Interface (Server Console)

The Shepherd Command Line Interface (CLI) provides a wide variety of functions in addition to letting the user view and manipulate the Shepherd Directory. Commands available include:

**add <name> {\*(<attribute>=<value>);}**

The add command inserts an object into the directory. <name> can be specified as distinguished or relative distinguished. A sample add command would be:

```
add cn=user1 {objectclass=ShepherdAccount;cn=user1;userPassword=secret}
```

**cd <pathname> | /B | ..**

Changes the current context (path) to the indicated path. <pathname> can be a relative distinguished name (RDN) or distinguished name (DN). The /B option is used to move immediately to the base of the directory tree. The .. option moves to the next highest level of the tree.

**delete <name>**

Delete removes an object from the directory. <name> can be a DN or RDN.

**dir**

Displays a list of objects in the current context much like the dir or ls command available to operating systems.

**export <filename>**

Exports all directory data to an LDIF file. The export command should be used frequently to perform backups of the directory.

**import <filename>**

Imports data from an LDIF formatted file into the directory.

**kill <pid>**

Kills a process running on the same system as Shepherd by its process identification number. The task command shows the process identification number as the first column of its output.

**load <servicename>**

Starts a Shepherd Service. <servicename> must be the distinguished name of a Shepherd Service object in the Shepherd Directory.

The load command is useful for starting a newly installed Service without taking the Shepherd Server offline. The new Service object can be added to the directory and immediately loaded without any down time. It can also be used to load a Service temporarily for administrative functions.

**log (view | rotate | truncate) <logname>**

Shepherd allows log files handled through Shepherd's standard log file system to be manipulated through the CLI. <logname> is the name of the ShepherdLog object on the server. If the log specified is not actively being used by a Service, you will **not** be able to manipulate it. The view option displays the log file. The rotate option stores all the data in the log in a file named the same as the log file plus the current date. Truncate simply erases the current log file and starts fresh.

**modify <name> {\*((add|replace|delete):<attribute=<value>;)}**

The modify command changes an object in the directory. <name> specifies the DN or RDN of the object. A sample modify command would be:

```
modify cn=user1 {add:objectclass=person;add:description=User 1}
```

## **pwd**

Displays the current directory context. Much like default Unix shells, the Shepherd command prompt does not display the current path. Since the current path can easily be very long, the pwd command exists to provide the user a simple method of checking the current context.

## **reboot**

Reboots the computer system on which Shepherd is running. File system caches and Shepherd caches are properly flushed to disk, but changes to the OS/2 desktop will not be saved.

## **Restart [<servicename>]**

Restart applies only to the Shepherd Server Engine and acts much like performing a shutdown followed immediately by starting Shepherd again. If the optional service name parameter is included, Shepherd will stop and restart only the identified service. This is equivalent to unloading and loading a service.

## **show <objectname>**

Show displays the contents of the directory object specified. The <objectname> can be given as either a DN or RDN. Shepherd uses an LDIF-like display format.

## **shutdown**

Shuts down the Shepherd Server. All connections are closed, Services unloaded, and DSP changes flushed to disk.

## **task**

Displays a list of tasks running on the computer running Shepherd.

## **unload <servicename>**

Stops a running Shepherd Service. <servicename> must be the distinguished name of a Shepherd Service object in the Shepherd Directory.

The unload command is useful for stopping a running Service without stopping the Shepherd Server.

All CLI use is governed by the access control set in the directory. At the Shepherd console, the account specified in the shepherd.ini file is used for access to the directory. For Shepherd.Admin, an account and password are specified before the CLI session starts.

## Using the Shepherd.Admin Service (Remote CLI)

The preferred method of using Shepherd.Admin is through the Shepherd.Admin Java Client. To install the Java client, create a directory on your system, and unzip the Shepherd.Admin Client archive into it maintaining directory structure. A Java 1.1 or newer runtime is required for the Client. A command file for OS/2 is included as an example of how to start the Shepherd.Admin Client. This file should be easily adaptable to other platforms.

The Shepherd.Admin Client allows you to create profiles for each Shepherd server in your organization and connect to and manage those servers. Once you have connected to a server, you will be presented with a tree view of your directory. You can expand containers by clicking the plus (+) sign to the left of the container and contract them by clicking the minus (-) sign. Objects are displayed by clicking the object, and their attributes can be modified through the panel on the lower right. To add an object, click on the container in which it should be created and click Selected, New Object.

For a more detailed explanation of the Shepherd.Admin Client, refer to the Shepherd.Admin Client User's Guide.

The Shepherd.Admin Service also supports telnet logins through the port specified in the Shepherd.Admin Service object. Once connected to the Service, you must perform an authentication step before gaining access to the CLI. This consists of the user and pass(word) commands. The user command looks as follows:

```
user cn=admin, c=US
```

Shepherd.Admin will not respond at all to the user command. Immediately following the user command, you should enter the password command. The password command looks like:

```
pass secret
```

If authentication succeeds, you will get no response and will be able to start sending CLI commands. If the authentication fails, an error message will be displayed followed immediately by the connection being terminated.

When using CLI commands with Shepherd.Admin, some commands will, due to their function, disconnect your Shepherd.Admin session. Restart, shutdown, and reboot all disconnect the current connection. Once these commands are applied, you will need to either wait on Shepherd.Admin to be started again (if possible), or you will need an alternate method of accessing your server to start Shepherd again.