



# Shepherd Agent Specification

*J&J Computer Consulting Shepherd Server Publications*

## Overview

Shepherd Agents provide an extension to the Shepherd Engine by granting a separate thread of execution to a module outside of Shepherd. Much like Shepherd Services, Shepherd Agents reside in a Dynamic Link Library (DLL) and implement methods required by the Shepherd Agent interface. Possible uses for Shepherd Agents include:

- ◆ A Job Scheduler
- ◆ Synchronization with external directory data (Meta-Directory Services)
- ◆ Monitoring

While external programs can do many of the same things, Shepherd Agents have the advantage of being fully manageable through Shepherd's administration services. Shepherd allows loading and unloading of Agents just like Shepherd Services, and all configuration data required by Agents resides in SDS.

This document describes how to setup existing Agents and how to develop new Shepherd Agents.

## Setting Up Shepherd Agents

Shepherd Agents consist of two files: <agent>.dll and <agent>.agt.

Typically, these two files should be installed in the "agents" subdirectory of the Shepherd installation directory, but Shepherd makes no requirements for the location of Shepherd Agents. In some cases, a Shepherd Agent may be part of a larger application and will reside outside the Shepherd directory structure.

The <agent>.agt file contains the information necessary for Shepherd to interface to the DLL containing the Agent code. The format matches that of other initialization files within Shepherd. As an example:

```
[agent]
dll=agents\agent.dll
initialize=agentInitialize
terminate=agentTerminate
run=agentRun
version=agentVersion
```

The "dll" value specifies the path to the Agent's DLL, relative or full. The other values specify the

function names used in the Agent DLL to implement Shepherd's Agent interface.

Once the DLL and AGT file are in place, all of the configuration for the Agent moves to SDS. Within SDS, an object with objectclass "ShepherdAgent" is defined for each Agent. A sample from SDS Replication Services looks like:

```
dn: cn=replication, c=US
cn: replication
objectclass: ShepherdAgent
agentModule: replicate\sdsrepl.agt
hostServer: cn=Master, c=US
log: cn=replicationErrorLog, c=US
log: cn=replicationAuditLog, c=US
```

Shepherd locates the AGT file through the agentModule directive. In this case, since SDS Replication Services are part of a larger application, agentModule points to sdsrepl.agt in the replicate subdirectory of the working directory.

The hostServer attribute identifies which Shepherd system should run the Agent. Shepherd supports multiples values for the hostServer attribute in case the Agent should run on multiple servers with the same configuration.

Like Shepherd Services, Agents use log files, if required. The Agent's documentation should identify the log types supported.

Once configured properly, the Shepherd Agent will start on the next restart or in response to a "LOAD" directive through the Shepherd Console or by Shepherd.Admin. To successfully load the Agent, though, the user must have Administrative access on the Agent object.

## Developing Shepherd Agents

Shepherd Agents must implement the following methods:

```
int initialize( AgentInitializer* initdata, AgentConfiguration** config );
int terminate( AgentConfiguration* config );
int run( AgentConfiguration* config );
int version( str* version );
```

The main body of the Agent takes place in the run method. The initialize and terminate methods get called before and after execution of run respectively. The initialize method has the opportunity to stop execution of the Agent by returning a non-zero value. The version method allows the Agent to return version information to the user. The data returned can be of any format.

When the Agent loads, Shepherd calls the initialize method. The first parameter, AgentInitializer, is defined as follows:

```
struct AgentInitializer {
    Directory* directory;
    ServerConsole* serverConsole;
    CommandProcessor* commandProcessor;
    DirectoryObject agentObject;
    DirectoryObject serverObject;
    DirectorySession session;
    str baseObjectName;
    ShepherdLog** logs;
};
```

The information provided in this structure gives the Agent access to SDS equivalent to that of the Shepherd Engine. Please consult the SDS Programming Reference for further information on the classes used in this structure.

The second parameter, AgentConfiguration, acts as a storage area for any data needed by the Agent. Shepherd lets the agent handle memory management for the AgentConfiguration object by passing a reference to a ShepherdConfiguration pointer. If the Agent uses this storage, it should allocate the object as necessary. Typically, this is accomplished by creating an AgentConfiguration derived object within the Agent and returning a pointer to that object.

If the Agent returns a non-zero value for the initialize method, Shepherd stops the loading process and reports an error to the Shepherd Console or to the Shepherd.Admin Service as necessary. If the Agent returns zero, Shepherd proceeds to call the run method, the first parameter of which is the AgentConfiguration object allocated during the initialization phase.

The run method of a Shepherd Agent **MUST BE FAULT TOLERANT**. Shepherd preemptively kills the thread running the Shepherd Agent in response to a shutdown, restart, reboot, or unload request.

Once the Agent has been stopped, Shepherd calls the terminate method. The terminate method takes responsibility for any cleanup required by the Agent **INCLUDING** freeing memory associate with the AgentConfiguration object allocated in the initialize method.

The version method of a Shepherd Agent is called in response to a version command issued through the Shepherd Console or by the Shepherd.Admin Service for the Agent.