

Core Concepts

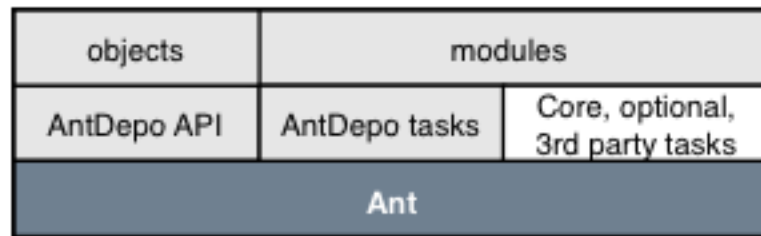
1. Overview

This section describes some core concepts that underly the AntDepo software and its use.

2. Framework

AntDepo contains a command-dispatching framework built using the AntDepo API, to lookup and execute specified commands. The command-dispatching capability is also exposed as an Ant task which is part of a set of AntDepo Ant tasks. The framework also provides depots for managing command modules, management objects and their property data. Several utilities are included to execute commands and administer the framework.

Figure 1: AntDepo Software



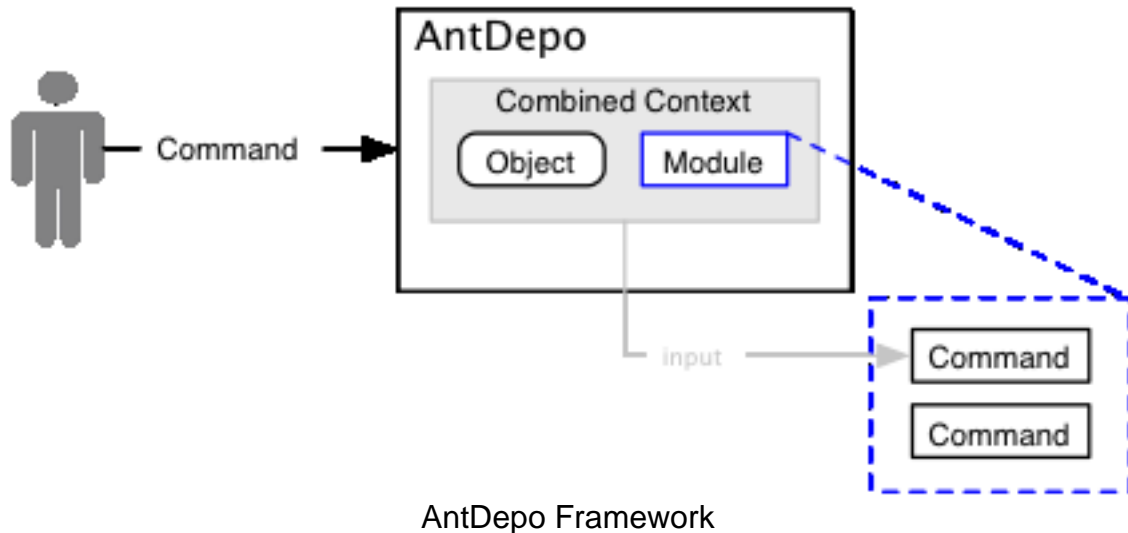
AntDepo Software

Note:

AntDepo includes a distribution of Ant to support the framework.

The figure below shows how a user executes a command via the AntDepo framework. The command-dispatcher receives the request to run a command, looks up the command's handler, and then, provides as input, a data context that can be managed within the framework.

Figure 2: AntDepo Framework



3. Modules

A module is a packaged set of commands created to execute some set of procedures. Modules can be likened to a plugin to the framework. Once a module has been installed the framework provides access to execute its commands.

Modules can be organized into a structure of super- and sub-modules to support generic/specific procedures and provide a means for better reuse.

3.1. Commands

Commands are named procedures in the module. Typically, commands are used to control the deployment life cycle of applications but they might also be used to control processes and tools. The implementation of a command is called a *command handler*. Command handlers are really Ant build files that follow a few AntDepo conventions.

Figure 3: Command handlers

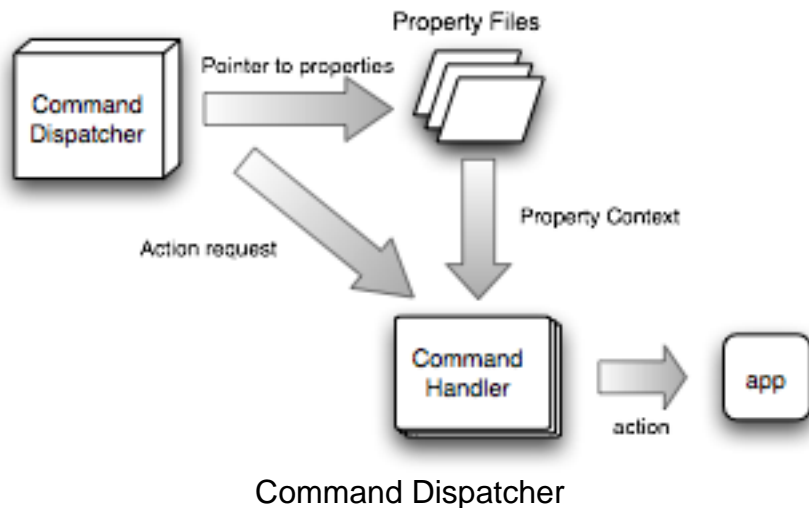


Command Handlers

Core Concepts

The framework's command dispatcher knows how to lookup handlers in the module library, set a data context, and then execute them.

Figure 4: Command dispatcher mechanism



A fundamental tenet in AntDepo is to "soft code" commands by stating important configuration detail in property files in order to separate the procedural logic from environment detail. This leads to commands that are more flexible and reusable in different environments.

4. Context

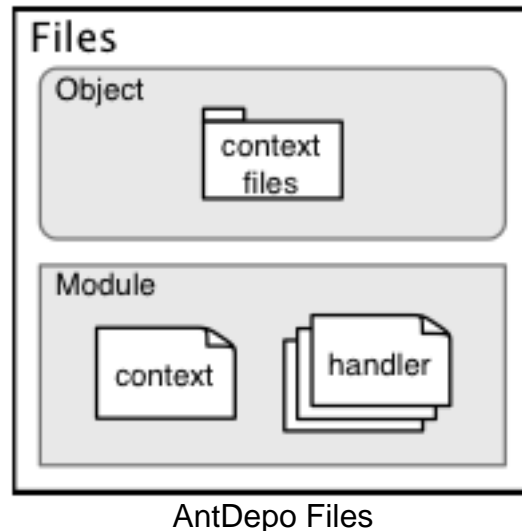
As mentioned earlier, commands in AntDepo are driven by a data context defined by a set of Ant (i.e., Java) properties. The AntDepo framework provides standard places for you to define and manage the property data but you are free to create and use property data in your own files.

The kind of information you might store in the property files include:

- configuration data: app and environment settings
- procedural data: management commands
- dependency data: bindings to other objects
- any thing else your procedures need

The framework uses several configuration files where global configuration data is stored. Every command handler will see this global data as it is loaded by the command-dispatcher before the handler is loaded.

Figure 5: Files



4.1. Module Context

The first place to define management data is in the module's property files. The module has two standard property files:

1. `module.properties`: This contains metadata about the module.
2. `commands.properties`: This contains metadata about each command

Your handler will load this property data when executed. Other data can be stored in other property files and read in by the handler using the Ant `property` task.

4.2. Object Context

When there is a lot of data to manage for commands, or when you need to coordinate procedures by invoking commands across modules, the AntDepo framework provides the capability of defining objects to organize the management data and enable a scheme to coordinate procedure.

Objects are typically created to represent deployments. Each object has its own property file. Object property files use an AntDepo property naming convention to organize the management data into several aspects. Below are examples of the aspects supported by AntDepo:

1. `command`: What commands can be run for this object. Objects are controlled via a module.

Core Concepts

2. deployment: Related deployments which may be upstream or down stream application dependencies.
3. document: Application configuration files and their templates used during the deployments of new releases.
4. package: Software package dependencies.
5. setting: Configuration setting information.

Project Depots

The framework repository organizes objects into project depots. For each object, the framework provides a file structure. Each object is allocated its own directory which can be used as a workspace for command executions.

5. Distributed Management

5.1. Nodes

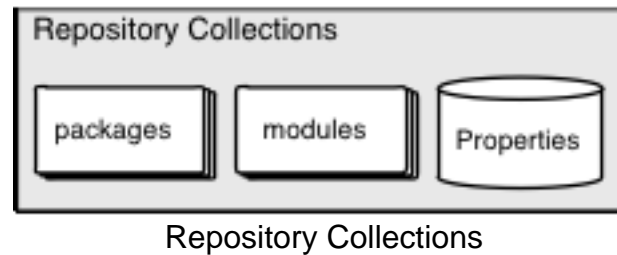
A configured AntDepo software instance on a machine is referred to as a node. By convention a node can take on one of two roles, target node or manager node. A target node is one that receives commands from a manager and executes them. A manager node is a central server from which remote management of target nodes will be performed. Typically, the manager node hosts the repository described below.

5.2. Repository

An important resource for managing a network of AntDepo nodes is its central repository. Once command modules have been developed they should be stored in this repository so that nodes that depend on them can download them. The AntDepo repository is a file server based on WebDAV. The AntDepo manual includes information about setting up a WebDAV server.

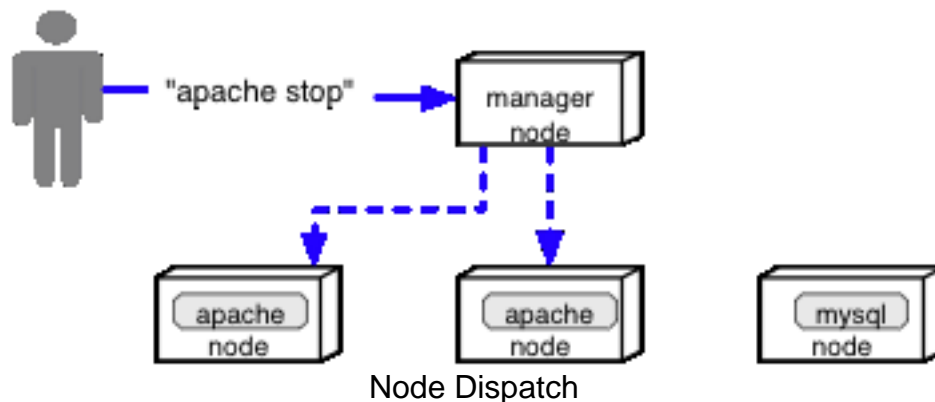
The Repository uses a content structure divided into the following categories:

- **Modules:** A collection of packaged AntDepo control modules that can be deployed to AntDepo nodes.
- **Packages:** A collection of software packages that will be deployed to the AntDepo nodes. These packages may be of various file types such as zip, tgz, rpm, war, jar, ear, etc. Common package types like those listed above have a corresponding AntDepo module which is used to carry out its installation life-cycle.
- **Configuration:** A set of property files that include information about nodes and their deployments. One such file, `deployments.properties`, describes on which nodes objects and their modules should be installed.



5.3. Node Dispatch

The internal command dispatch mechanism supports an execution strategy referred to as `nodedispatch`. Using this strategy, users can target commands to modules or objects without having to specify the nodes they reside on. The command dispatcher is able to lookup the node of the target, dispatching the command remotely or locally as appropriate. This feature provides network transparency to command execution and makes management of server pools and clusters more convenient.



6. Glossary

- Command - Named procedure that can be executed via AntDepo
- Command handler - An implementation of a Command
- Context - Data context made available during the execution of a handler
- Framework - A command dispatching mechanism and repository of command modules and management data.
- Module - A packaged set of Commands.
- Object - A management entity that has its own workspace and an associated module.
- Project - A repository of management objects. Also, often referred to as a "depot".