Geronimo Modules

{scrollbar}

INLINE

Geronimo is composed of a lightweight core (or kernel) and many modules. Each module may include system code (such as a thread pool or web container) or may be an application (such as the management console or a user-deployed application).

There are several advantages to this structure:

- All modules in Geronimo, whether system modules or application modules, can be individually started or stopped
- It's possible to disable or remove the modules for features you don't need (perhaps CORBA or the EJB container)
- It's possible to add new features to Geronimo by installing additional modules
- It's easy to distribute modules to other servers or users in the form of Geronimo Plugins

One of the side effects of this structure is that every module has a Module ID, and you will become very used to seeing module IDs when dealing with Geronimo.

Module

A module ID uniquely identifies a specific module, including its name, version, etc. In fact, there are four components of a module ID:

- **Group ID** A name identifying a group of related modules. This may be a project name, a company name, etc. The important thing is that each artifact ID should be unique within the group. If no group is specified when declaring the module ID for a module or application, it will get the group ID default. If no group is specified for a module ID used to identify a dependency, it's treated as a wildcard and the group is not used to help identify the exact dependency.

- **Artifact ID** A name identifying the specific module within the group. For example, there may be a single group ID for an application, with separate artifact IDs for the web application and EJB modules that make up that application. Every module ID must include an explicit artifact ID. If no module ID at all is specified when deploying a module, the artifact ID defaults to the files name of the module file.

- **Version** Each module has a version number, normally in a format like 1.2.3. If no version number is specified when declaring the module ID for a module or application, it will get a numeric timestamp as its version number (e.g. System.currentTimeMillis()). Each time the module is redeployed, it will get a new timestamp. If no version number is specified for a module ID used to identify a dependency, it means that any available version will be used. If there are multiple versions, Geronimo favors any version that might already be loaded, or else typically the newest version will be used.

- **Type** A module's type is normally either car (for a system module), or the file extension for an application module (ear, war, jar, etc.). If no type is specified, the type will be set appropriately by the deployer when the module is deployed.

Module IDs are normally printed with slashes between the four components, such as `GroupID/Artifact-ID/Version/Type`. A module ID with all 4 components is known as `fully resolved`. Any module that has been deployed will have a fully resolved module ID. However, you may use unresolved module IDs when declaring dependencies on other modules – normally omitting the version number to indicate that any version will suffice.
Module IDs are also used to refer to common libraries included with Geronimo, and available for use by system modules or applications. These common libraries are typically JARs, but use the same identifier format of `GroupID/ArtifactID/Version/Type`.