**Policy文件的语法格式与说明**

一个Policy文件实质上是一个记录列表，它可能含有一个“keystore”记录，以及含有零个或多个“grant”记录。其完整格式如下（斜体字代表变量）：

keystore "*some\_keystore\_url*", "*keystore\_type*", "*keystore\_provider*";

grant signedBy "*signer\_names*", codeBase "*URL*",

principal *principal\_class\_name* "*principal\_name*",

principal *principal\_class\_name* "*principal\_name*",

... {

permission *permission\_class\_name* "*target\_name*", "*action*", signedBy "*signer\_names*";

permission *permission\_class\_name* "*target\_name*", "*action*", signedBy "*signer\_names*";

...

};

keystore记录（可选）

* 一个keystore是一个私有密钥（private keys）数据库和相应的数字签名，例如X.509证书。Policy文件中可能只有一条keystore记录（也可能不含有该记录），它可以出现在文件中grant记录以外的任何地方。Policy配置文件中指定的keystores用于寻找grant记录中指定的、签名者的公共密钥（public keys），如果任何grant记录指定签名者（signer\_names），那么，keystore记录必须出现在policy配置文件中。
* “some\_keystore\_url”是指keystore的URL位置，”keystore\_type”是指keystore的类型。第二个选项是可选项，如果没有指定，该类型则假定由安全属性文件（java.security）中的”keystore.type”属性来确定。keystore类型定义了keystore信息的存储和数据格式，用于保护keystore中的私有密钥和keystore完整性的算法。Sun Microsystems支持的缺省类型为“JKS”。 “keystore\_provider”是指keystore的提供者。

grant记录

* SignedBy（可选）指定签名者（signer\_names）
* Principal（可选）指定一些由java.security.cert定义的用户证书
* CodeBase（可选）指定一个或多个代码，关于路径通配符见下文
* 在Policy文件中的每一个grant记录含有一个CodeSource（一个指定的代码）及其permission(许可)。
* Policy文件中的每一条grant记录遵循下面的格式，以保留字“grant”开头，表示一条新的记录的开始，“Permission”是另一个保留字，在记录中用来标记一个新的许可的开始。每一个grant记录授予一或多个指定的代码（CodeBase）一套许可（Permissions）。
* permission\_class\_name必须是一个合格并存在的类名，要写全，例如java.io.FilePermission，不能使用缩写target\_name用来指定目标类的位置，action用于指定目标类拥有的权限。
* target\_name可以直接指定类名（可以是绝对或相对路径），目录名，也可以是下面的通配符：

directory/ 目录下所有class文件，不包括jar文件

directory/\* 目录下的所有文件，不包括子目录

directory/- 目录下的所有文件，包括子目录

其实，在实际使用中，我们可能不会像上面介绍的那么复杂，特别是在不使用数字签名（keystore）时。这时，我们完全可以参考借鉴JDK 1.x提供给我们的现成的\jdk1.x\jre\lib\security\java.policy文件，根据我们的需要作相应的修改。（其中keystore、SignedBy、Principal在小规模平台的使用场合经常被忽略）以下给出一些常用的简单例子。

* 常用举例：

1. java.net.SocketPermission，action：listen,accept,connect,resolve

举例： grant { permission java.net.SocketPermission "localhost:\*","accept,listen,connect,resolve"; };

其中当accept（接受）,listen（监听）,connect（连接）被添加时，resolve（翻译为解析）也默认被添加了，resolve指忽略端口的检测。

1. java.io.FilePermission，action： read, write, delete, execute,readlink和<<ALL FILES>>

其中readlink表示允许类java.nio.file.Files的方法readSymbolicLink读取symbolic link（符号链接）

<<ALL FILES>>表示允许读写文件系统的所有文件，不建议使用。

举例： grant {permission java.io.FilePermission "/tmp/\*", "read,write"; };

1. java.util.PropertyPermission， action: read, write

其中read：允许调用System.getProperty write：允许调用System.setProperty

举例: grant { permission java.util.PropertyPermission "java.vendor", "read"; };

1. 还有更多permission例如AllPermission、MBeanPermission、RuntimePermission等等

请查阅官方文档https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html

本文教程参考JAVA7官方文档，更多详细请参阅：

<https://docs.oracle.com/javase/7/docs/technotes/guides/security/PolicyFiles.html>