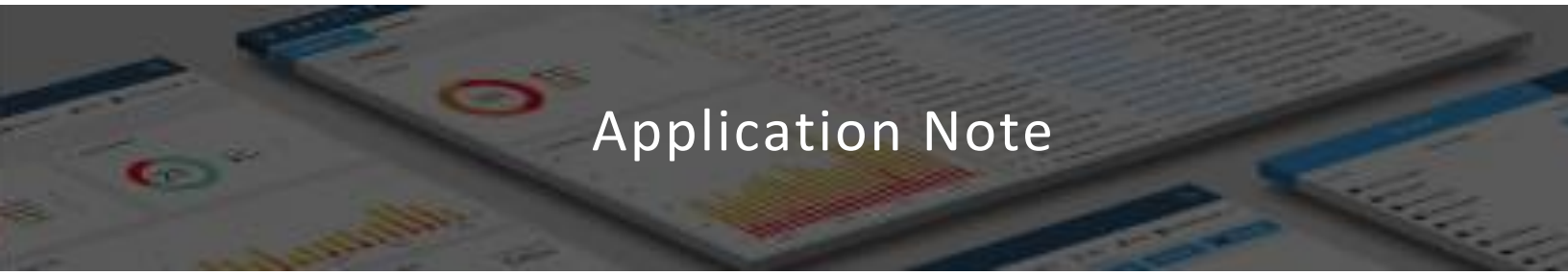
ENTUITY
NETWORK ANALYTICS



Application Note

Use of non-Groovy scripts with Configuration Management

23rd April 2019

John Diamond

Table of Contents

1 Overview 3

2 Managing the security sandbox 3

3 Executing non-Groovy scripts 4

1 Overview

The Configuration Management facilities of Entuity leverage Groovy scripts to control the required interaction with managed devices. The Groovy language is derived from Java and allows the execution of scripts without explicit compilation by a user. A “Just in time” approach to compilation is incorporated that provides the convenience of a scripted environment with the efficiency of a compiled language. Furthermore, given that Groovy is derived from Java it can leverage Java libraries and can include Java syntax if that is the preference of the script implementer. A set of such scripts is included with the product and installed by default. Suitably authorized users can add their own scripts to control additional configuration management functions that they can then add to the system. The scripts form the executable definitions for the “Steps” performed by Configuration Management “Tasks”. The execution of these Groovy scripts is controlled by a functional module called the “scriptEngine”. Any script executed by the scriptEngine is assumed to be written in Groovy.

This application note describes how to implement a simple Groovy script that launches another script that is implemented in some other scripting language. The Windows Command Prompt will be used as an example script processing engine as it is capable of executing batch files.

2 Managing the security sandbox

There is syntax within the Groovy language that allows external executables including alternative script processors or shells to be executed. The use of external executables is dependent on suitable adjustments being made to the security sandbox configuration in which the scriptEngine operates. The scope of this sandbox is defined by the etc\scriptEngine.policy configuration file which, in Entuity 17.0, has the following default settings:

```
// This file contains permissions for the scriptEngine

grant codeBase "file:${ENTUITY_HOME}/-" {
    permission java.security.AllPermission;
};


// Permissions for config management groovy scripts.
// These have a code base of /com/entuity/configManager/script/<HOSTNAME>
// Where HOSTNAME is the originating server name as known to Entuity

grant codeBase "file:/com/entuity/configManager/script/-" {
    permission java.util.PropertyPermission "ENTUITY_HOME", "read";
    permission java.io.FilePermission "${ENTUITY_HOME}/configManagement/-", "read, write, delete";
    permission java.io.FilePermission "${ENTUITY_HOME}/cm_transfer/-", "read, write, delete";
    permission java.io.FilePermission "${ENTUITY_HOME}/cm_archive/-", "read, write, delete";
    permission java.io.FilePermission "${ENTUITY_HOME}/log/expect.log", "read, write, delete";
    permission java.io.FilePermission "${ENTUITY_HOME}/log/expect.log.1", "read, write, delete";
    permission java.io.FilePermission "${ENTUITY_HOME}/log/expect.log.2", "read, write, delete";
    permission java.io.FilePermission "${ENTUITY_HOME}/log/expect.log.3", "read, write, delete";
    permission java.io.FilePermission "${ENTUITY_HOME}/log/expect.log.4", "read, write, delete";
};
```

Each permission statement within the grant section for `"file:/com/entuity/configManager/script/-"` defines a directory or file that the scriptEngine is allowed some level of access to. Any directory or file not mentioned is, by implication, outside the accessible reach of the scriptEngine. Each entry can have any combination of “read”, “write”, “delete” or “execute” permission granted. Adding the following line to the same grant block enables the scriptEngine to launch the Windows Command Prompt so that batch files can be executed:

```
permission java.io.FilePermission "C:/Windows/System32/cmd.exe", "execute";
```

Note that the file separator used is "/" regardless of whether this is a Windows or Linux installation.

If changes are made to scriptEngine.policy while the Entuity server is running, they will have no effect until the scriptEngine is restarted. A restart of the scriptEngine happens when the server is restarted but can be performed in isolation, without the need for a complete system restart, from the command line using "bin\stop scriptEngine".

3 Executing non-Groovy scripts

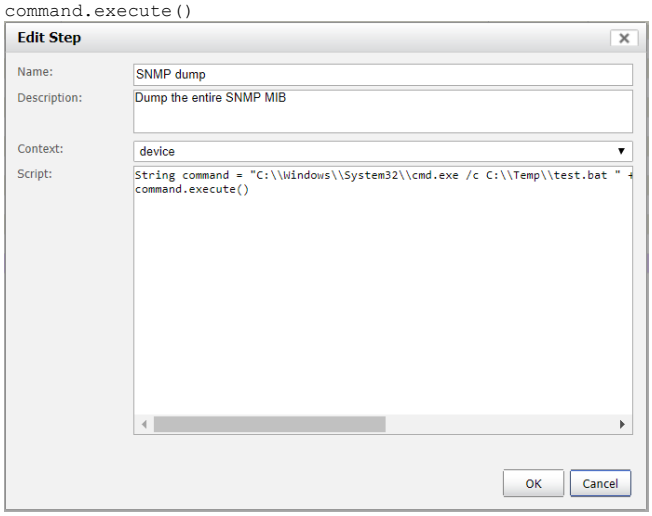
An example of how the above configuration change can be exploited would be to run the "snmpdump.exe" utility from a Configuration Management Task. A batch file called "test.bat" was placed in "C:\Temp" (a more suitable directory would be recommended for a production deployment). The contents were as follows:

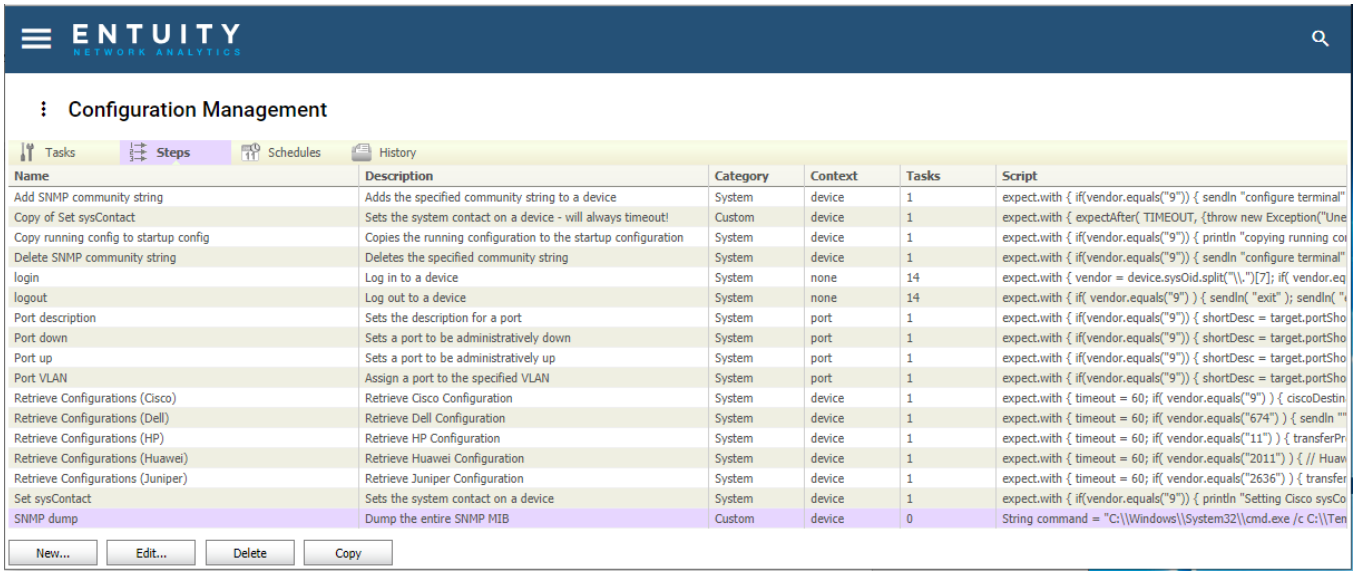
```
C:\Entuity16.5\lib\tools\snmpdump -v2c -c %3 %2 >C:\Temp\%1.txt
```

Note that the full path of the snmpdump executable was specified and an output file redirection was included. Additionally, three command line parameters were passed containing the device name (%1), polled IP address (%2) and read-only community string (%3).

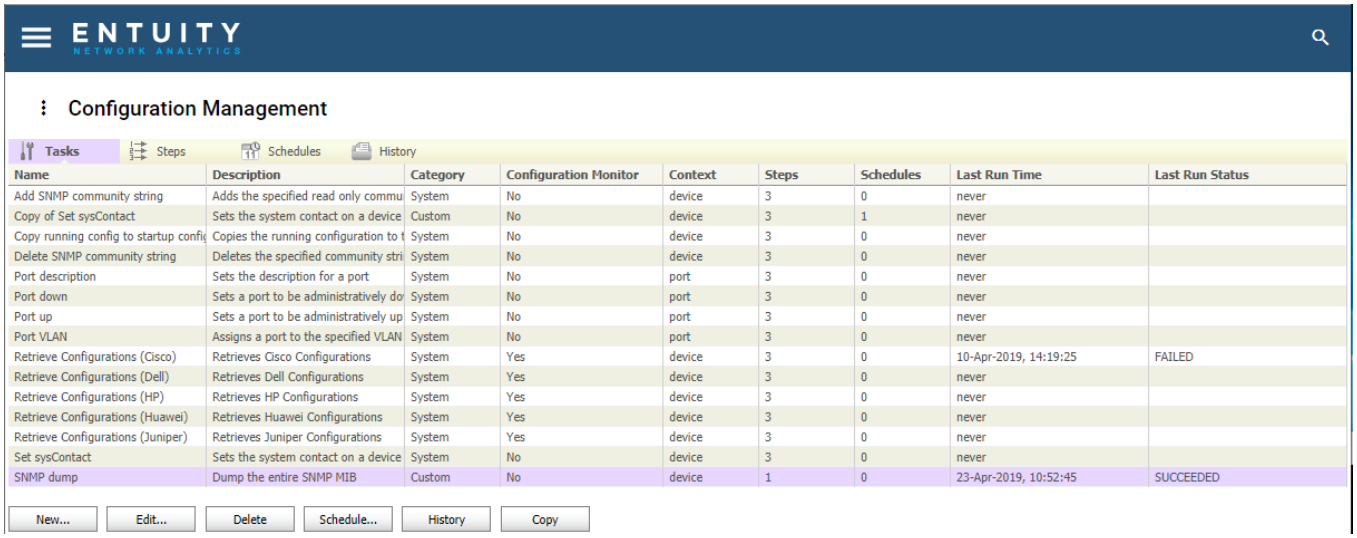
The batch file was called from a Configuration Management Step script that looked like this:

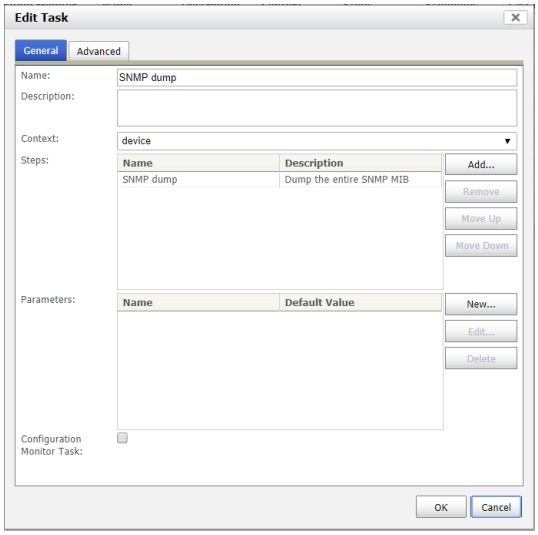
```
String command = "C:\\Windows\\System32\\cmd.exe /c C:\\Temp\\test.bat " + target.name + " " + target.devPolledIpAddr +
" " + target.snmpCommunity
command.execute()
```

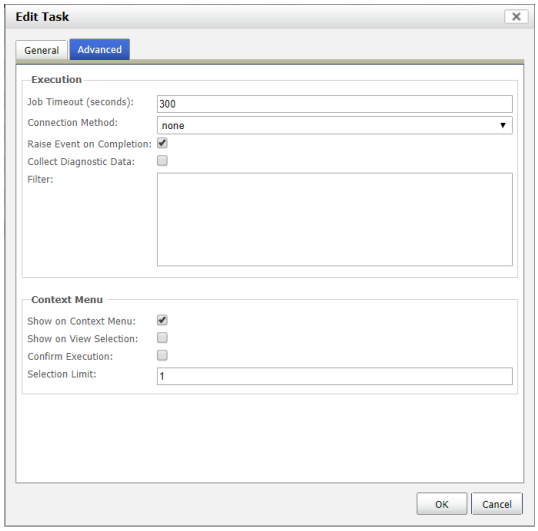




The “SNMP dump” Step was invoked by a Task called “SNMP dump”:







Note the syntax used in the Step script to obtain the device name, polled IP address and read-only community string from the corresponding StormWorks data structure for the selected device. Also note that the `command` string defined in the Groovy script which is then executed by `command.execute()` did not contain any file redirection syntax as this is only supported by the Command Tool and any batch file that it then executes. The above configuration allows an authorized Entuity user to request an SNMP walk of the entire MIB to be performed and written to a file in "C:\Temp" named with the device name.