Tivoli Netcool Support's Guide to the EXEC Probe

by

Jim Hutchinson

Document release: 2.0

Table of Contents

1 Introduction

The EXEC probe was written as a general integration for Netcool/OMNIbus. It is capable of running any command including custom wrapper scripts that can be capable of complex tasks.

The probes manual gives an example usage that involves TELNET:

```
ExecCommand : 'telnet sol10-build2 8001'
Header : 'CNK1'
Footer : 'END JOB'
ParseAsLines : 1
```

The event data can be read using a number of properties as described in the manual:

The EntrySequence property specifies the chat in string and the ExitSequence property specifies the chat out string.

The probe can be configured to recognize the beginning (Header) and the end (Footer) of an event. The probe can also alter the format of the event and how it is processed using the TokenSize, MessageSize, and TerminationCharacter properties.

Due to the numerous possible usages of the EXEC probe it is not possible to provide much detail other than what configuration settings are available and their potential.

1.1 Overview

The main properties manage the way and what EXEC command is run:

1. EntrySequence
2. **ExecCommand**
3. ExitSequence

The other main properties define how the event stream is processed once the EXEC command is running

- Header
- ParseAsLines
  - WhiteSpaces
  - QuoteCharacters
  - BreakCharacters
  - TerminationCharacter
- Footer

The EXEC probe is designed to exit when the EXEC command completes and if this is not the case, the command may of become a zombie process or the probe may have a bug.
e.g.

```
PATCH probe-nco-p-exec-4
REVISION 3:
-----------
        IV96982 - Probe now will send ExitSequence before closing event parser. This is
to avoid probe
                  from missing the last event coming from process before it terminates.
REVISION 2:
-----------
        IV56635 - Probe was found to include and create tokens for data that is not
between the Header
                  and Footer of events which may cause problems with rules file handling.
This issue
                  has been fixed.
REVISION 1:
-----------
        IV29561 - Probe now will shutdown if the exec process terminates.
```

2 Event Parsing

2.1 Example input data

```
HEADER
ServerName=AGG_P
ServerSerial=224045
FOOTER
```

2.1.1 Working settings

Property settings:

```
ParseAsLines     : 1
Footer           : 'FOOT'
Header           : 'HEAD'
```

Event tokens:

```
Header:     HEADER
Token0:     ServerName=AGG_P
Token1:     ServerSerial=224045
Footer:     FOOTER
```

2.1.2 Example#1 settings

Property settings:

```
ParseAsLines     : 1
```

Event tokens:

```
Token01:    HEADER
Token01:    ServerName
Token02:    AGG_P
Token01:    ServerSerial
Token02:    224045
Token01:    FOOTER
```

2.1.3 Example#2 Settings

Property settings:

```
ParseAsLines     : 0
```

Event tokens:

```
Token01:    HEADER
Token01:    ServerName
Token02:    AGG_P
Token01:    ServerSerial
Token02:    224045
Token01:    FOOTER
```

2.1.4 Example#3 Settings

Property settings:

```
ParseAsLines     : 0
Footer           : 'FOOT'
Header           : 'HEAD'
```

Event tokens:

```
Header:     HEADER
Footer:     FOOTER
```

3 Example Wrapper Script

The best way to run the EXEC commands is using a wrapper script that can include more complex commands that may be run in a single line.

3.1 Multiline Script

File : multiline_example.sh

```
#! /bin/sh
while true
do
cat /tmp/multiline_example.txt
sleep 10
done
#EOF
```

3.2 Example Data

File : multiline_example.txt

```
HEADER
ServerName=AGG_P
ServerSerial=224045
FOOTER
```

3.3 Property file

```
ExecCommand     : '/opt/IBM/tivoli/netcool/omnibus/utils/multiline_example.sh'
ParseAsLines    : 1
Footer          : 'FOOT'
Header          : 'HEAD'
```

3.4 Example output

```
Header:    HEADER
Token0:    ServerName=AGG_P
Token1:    ServerSerial=224045
Footer:    FOOTER
```

4 Encrypted Secrets Script

It maybe the password or other secret needs to be encrypted in the EXEC probe properties. This can be done by extracting the secrets from the running wrapper script and placing them into the command line options for the wrapper script. These secrets can then be encrypted in the probe property file using the AES crypt method.

4.1 Testing the script

File : sendPassword.sh

```
#! /bin/sh

if [ $# -ne 1 ]
then
        echo "Usage : `basename $0` [password]"
        exit
fi

export PASSWORD
PASSWORD=$1
# Comment out after testing
echo "PASSWORD=$PASSWORD"

while true
do
echo "abc=123"
sleep 1
done

# Exit
echo "Exit"
exit
#EOF
```

4.2 Probe properties

```
# Always set NetworkTimeout and PollServer
Server          : 'AGG_P'
ServerBackup    : 'AGG_B'
NetworkTimeout  : 15
PollServer      : 60
# AES Encryption
ConfigCryptoAlg : 'AES'
ConfigKeyFile   : '$NCHOME/etc/security/keys/netcool.exec.probe.keygen'
# Un AES Encrypted
#ExecCommand    : '/opt/IBM/tivoli/netcool/omnibus/utils/sendPassword.sh password'
ExecCommand     :
'@152:XL9IQFFyPJKVi+wW4/S9FfG2zEMutu4fnuMkk94LuJKl/K9sK/5waFaD37eRumKBMkseP7wJt6u8Sdu
+XW9XT7bPHnOMeaHGBUw8lAcqiWZEbJx0ihuDP1mH/SmD1lzeeA0XyxN0HxyZX2e7VhVAyA==@'
# Keep object server connection alive
HeartBeat : 60
#EOF
```

4.3 Encrypting Properties

Create the probe specific key file

```
$OMNIHOME/bin/nco_keygen -o $NCHOME/etc/security/keys/netcool.exec.probe.keygen
```

The Property is encrypted as:

```
$OMNIHOME/bin/nco_aes_crypt -c AES -k
$NCHOME/etc/security/keys/netcool.exec.probe.keygen
'/opt/IBM/tivoli/netcool/omnibus/utils/sendPassword.sh password'

@152:XL9IQFFyPJKVi+wW4/S9FfG2zEMutu4fnuMkk94LuJKl/K9sK/5waFaD37eRumKBMkseP7wJt6u8Sdu+
XW9XT7bPHnOMeaHGBUw8lAcqiWZEbJx0ihuDP1mH/SmD1lzeeA0XyxN0HxyZX2e7VhVAyA==@
```

5 Simple listener

It is possible to create a simple listener using the EXEC Probe along with netcat.

Note : There is a SOCKET Java Probe provided within the Netcool/OMNIbus portfolio, this example is provided as a simple demonstration of how flexible the probe is.

5.1 Script

File : run_netcat.sh

```
#! /bin/sh
while true
do
/bin/nc -l <FQDN> 10101
done
#EOF
```

5.2 Property settings

The property settings can be configured according to the input that is required to be read.
The main property settings here are the EXEC command and events being parsed as lines.

```
ExecCommand     : '/opt/IBM/tivoli/netcool/omnibus/utils/run_netcat.sh'
ParseAsLines    : 1
```

5.3 Example data

INPUT line :

```
column1 "column 2" "col umn 3"
```

Probe tokens:

```
Token01:   column1
Token02:   column 2
Token03:   col umn 3
```

6 Using SSH

You can automatically login to a system using SSH as given here:

Copy the login user's public key to the authorized_keys file in the remote users .ssh directory.
To create the public key run the ssh-keygen command.
Ensure that you do not use a passphrase
Afterwards copy the entry in ~/.ssh/id_rsa.pub to the remote users authorized_keys file in their .ssh direcotry, ~/.ssh.

It is generally best to include this detail in the command script for general usage:
e.g.

```
#! /bin/sh
# Script : run_some_command.sh
#
# To prevent password prompt copy user's public key to the authorized_keys file
# in the remote users .ssh's directory.
# To create the public key run ssh-keygen
# Do not use a passphrase
# Copy the entry in ~/.ssh/id_rsa.pub to the remote users authorized_keys file in ~/.ssh
#
/bin/ssh nrv81@server.ibm.com /opt/nrv81/IBM/tivoli/netcool/omnibus/utils/some_command.sh

#EOF
```