Tivoli Netcool Support's Guide to Probe Common Features

by

Jim Hutchinson

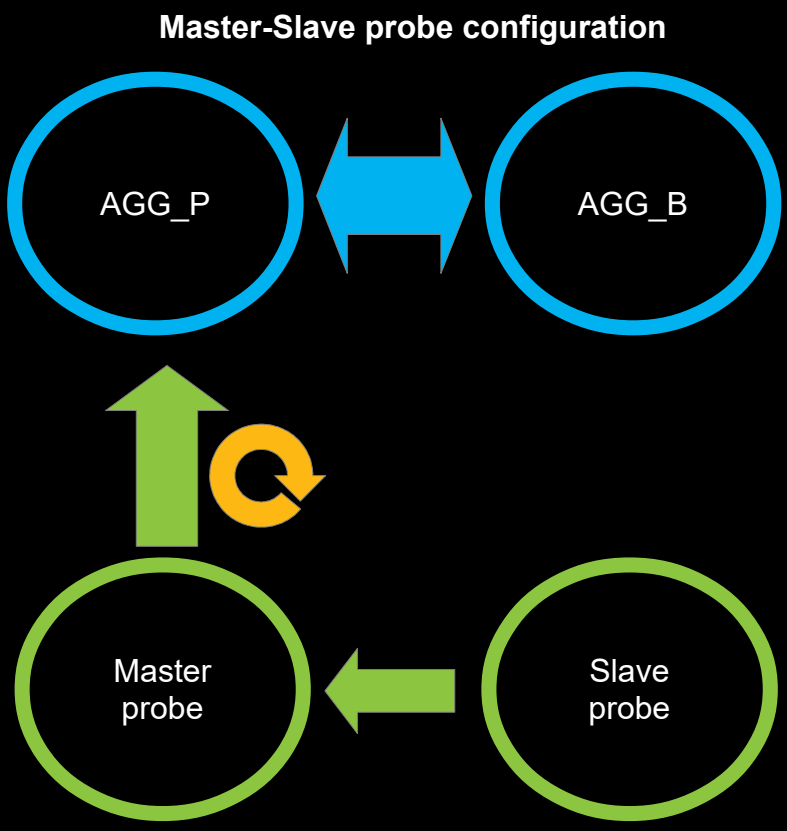
Document release: 2.0

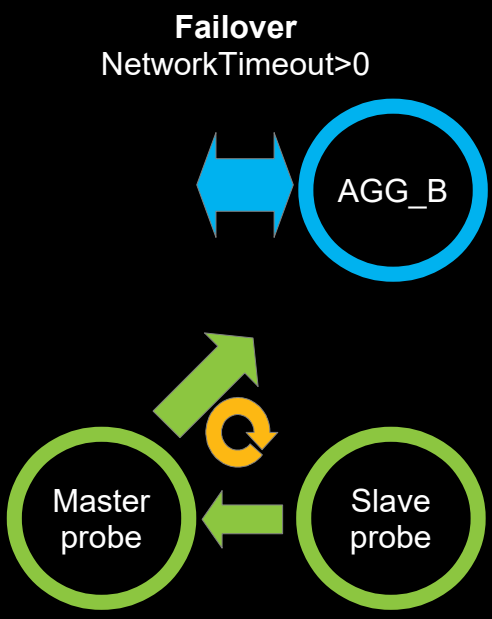
Table of Contents

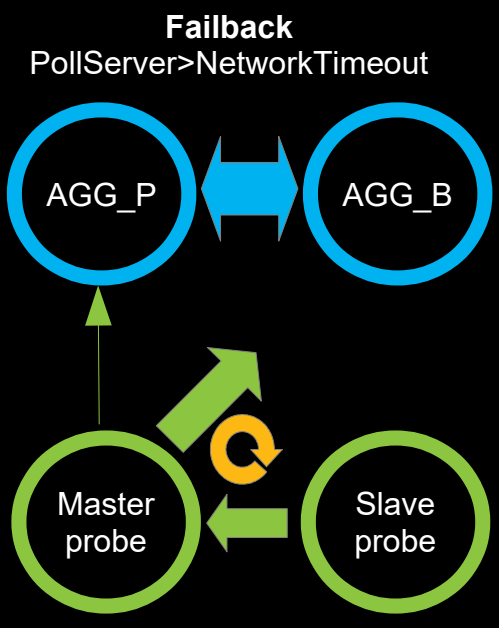
1 Introduction

1.1 Overview

There are a number of common probe properties, which allows the probe behaviour to be extended and improved.







2 Authentication

2.1 Probe user

2.2 AES_FIPS Password encryption

Create a keyfile directory in $NCHOME/etc/security to hold key files, this will keep them clear of any SSL/TLS files.

```
mkdir $NCHOME/etc/security/crypt
```

Check the file you want to create, does not already exist.

```
ls -l $NCHOME/etc/security/crypt/probe.keyfile
```

If it does not exist then create the file.

```
$NCHOME/omnibus/bin/nco_keygen -l 256 -o $NCHOME/etc/security/crypt/probe.keyfile
```

General guidance for secure passwords.
Minimum password length is 14 characters.
A password must have at least one lower case character, one upper case character, and one digit or special character.
Each character must not occur more than three times in a password.
No more than two consecutive characters of the password can be identical.
Characters are in the standard ASCII printable character set within the range from 0x20 to 0x7E inclusive.

Example password

```
12345678901234
NetcoolN3tc00l
```

How to encrypt a password:

```
$OMNIHOME/bin/nco_aes_crypt -c AES_FIPS -k
$NCHOME/etc/security/crypt/objectserver.keyfile "NetcoolN3tc00l" -o
/tmp/crypted_password.txt

cat /tmp/crypted_password.txt
@44:dV7x30dVSnHfZe2kpriX0nfGZNQZPz4avczhrtL56LY=@
```

Add the text from the encrypted password file to the property file with the required property settings to enable AES_FIPS:

```
# AES_FIPS settings
ConfigCryptoAlg : 'AES_FIPS'
ConfigKeyFile   : '$NCHOME/etc/security/crypt/probe.keyfile'
```

Password part

```
AuthPassword : '@44:dV7x30dVSnHfZe2kpriX0nfGZNQZPz4avczhrtL56LY=@'
AuthUserName : 'probe_user'
```

3 Peer-to-Peer

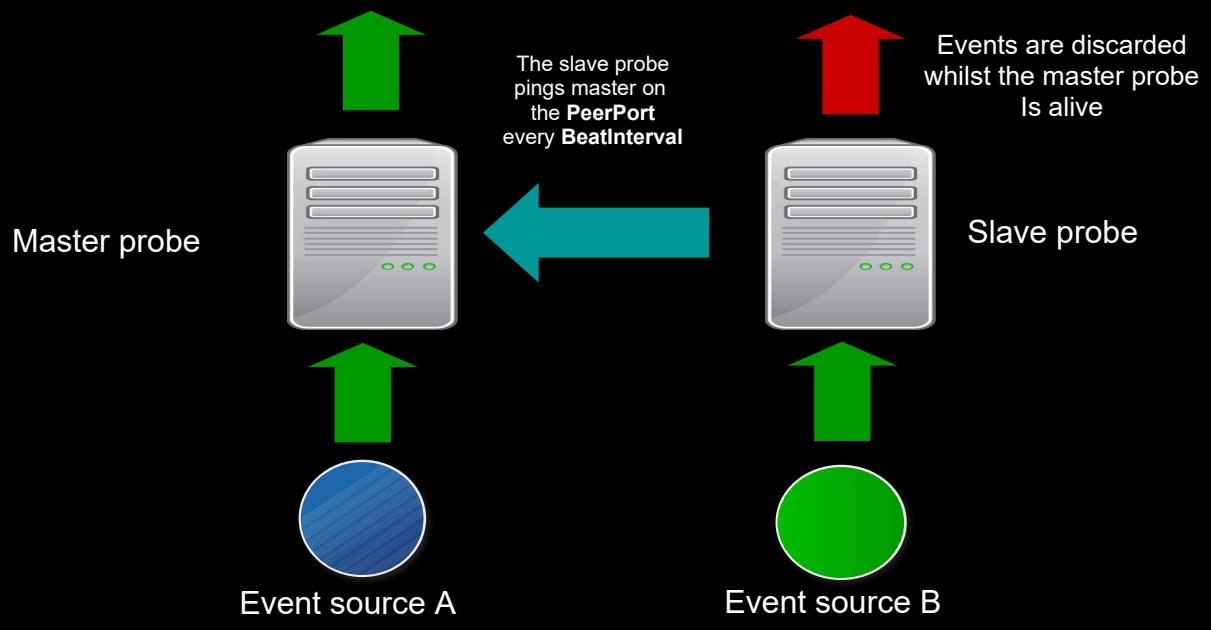
The peer-to-peer feature is designed to provide resilience in the events arriving at the collection layer.

Probes with identical event feeds are configured to use the peer-to-peer feature, on two different servers. Whilst the master probe is determined to be alive by the slave probe, the slave probe discards its events. If the master probe becomes unavailable the slave probe forwards its current discard events buffer, and forwards events to the object server.

Tuning of the discard events buffer and tolerance of the feature is performed using the BeatInterval and BeatThreshold.

The BeatInterval is the time (seconds) between peer-to-peer checks on the other peer-to-peer host. The BeatThreshold is the time (seconds) that is allowed for the peer-to-peer host to respond.
Whilst peer-to-peer prevents event loss, provided the two event sources are synchronised, peer-to-peer does not prevent duplicate events being forwarded to the collection layer.
The maximum delay in sending events to the object server is given by.

```
BeatInterval + BeatThreshold + 1
```



3.1 Master probe property settings

```
MessageLevel            : 'info'
# P2P
Mode                    : 'master'
PeerHost                : 'slaveprobe.company.com'
Peerport                : 8888
BeatInterval            : 25
BeatThreshold           : 5
```

3.2 Slave probe property settings

```
MessageLevel            : 'info'
# P2P
Mode                    : 'slave'
PeerHost                : 'masterprobe.company.com'
Peerport                : 8888
BeatInterval            : 25
BeatThreshold           : 5
```

3.2.1 Checking connectivity

If the master and slave probes are connecting well.

```
netstat -na | grep <PeerPort>
```

Return the following.

Master probe:

```
tcp6       0      0 masterhost:1234567    slavehost:8888     ESTABLISHED
```

Slave probe:

```
tcp6       0      0 :::5555               :::*               LISTEN
tcp6       0      0 slavehost:8888        masterhost:1234567 ESTABLISHED
```

You can use curl to check port access from the master probe.

```
curl telnet://masterhost:8888
```

3.3 P2P logging

3.3.1 Normal running

Master probe

```
[P2P] : Successfully sent duplex heartbeat
```

Slave probe

```
[P2P] : Received heartbeat from Master
```

3.3.2 Master probe goes down

Master probe

```
n/a
```

Slave probe

```
[P2P] : Failed to read data message. (-27:Connection closed)
[P2P] : All events will now be forwarded
```

3.3.3 Slave probe goes down

Master probe

```
[P2P] : Problem receiving heartbeat. (-27:Connection closed)
[P2P] : Unable to connect to slave 'masterprobe.company.com' on port '8888'.
(111:Connection refused)
```

Slave probe

```
n/a
```

3.3.4 Other P2P situations

When the slave probe is busy the timestamps coincide.

```
[P2P] : Received heartbeat from Master
[P2P] : Received heartbeat from Master
```

When the master probe is busy the events are forwarded by the slave probe.

```
[P2P] : All events will now be forwarded
```

3.4 Troubleshooting

Ensure the master and slave probe servers are synchronised in time, so that the logging can be compared.

You can use tcpdump to check the ports behaviour for extreme cases.

```
tcpdump -i any -s 0 -w p2pport8888.pcap port 8888
```

4 Buffering

The Buffering probe property is common to all probes although the flush property setting varies. Use the -dumpprops option to check the available probe property settings.

The best practice settings are configured for a nominal event rate rate of around twenty events per second, with the flush interval set to an odd number around the required period [for example 10s].

```
# Best practice
NetworkTimeout          : 15
PollServer              : 60
# Buffering - 20 events per second
Buffering               : 1
BufferSize              : 200
FlushBufferInterval     : 9
# Or
# BufferFlushInterval     : 9
```

By default the events are processed in the rules file, one at a time, and forwarded to the object server afterwards. When Buffering is enabled, the events are processed in the rules file engine until either the number of processed events equals the BufferSize or the flush interval is reached. If the flush interval is not set then events are not sent to the object server, until the Buffer is full.

If the event rate is 200 events per second, the buffer is filled every second. In this case the BufferSize needs to set accordingly, to accommodate the larger event rate. In 10 seconds, the number of stored event will be 2000, so a BufferSize of 2000 is appropriate at this higher event rate.

The higher the event rate, the larger the performance impact of Buffering is, since the accumulated time saved by Buffering, is larger.

5 Failover/Failback

The probe property file supports failover and failback to collection object servers. When probes connect to the aggregation layer, they can rely on the backup object servers trigger disconnect_all_clients to failback to the primary object server.

The failover/failback properties are.

- Server
- ServerBackup
- NetworkTimeout
- PollServer

The probe will use the Server and ServerBackup for object server connectivity. Attempting to reconnected to the Server, whilst connected to the ServerBackup. Taking the PollServer setting, as the period for the availability check for the Server. The NetworkTimeout setting determines how long the probe waits for a response from object servers. This means that the NetworkTimeout must be set to a value less than PollServer value, otherwise the probe never determines the Server's availability.

```
# Failover/Failback settings
Server                  : 'AGG_P'
ServerBackup            : 'AGG_B'
# Best practice
NetworkTimeout          : 15
PollServer              : 60
```

5.1 NetworkTimeout

The NetworkTimeout and PollServer should be set too, to ensure good behaviour of the probes connection to the object server. By default NetworkTimeout is unlimited so the probes connection to the object server is unable to detect network problems as well as object server performance problems. Events are stored in memory, rather than being sent to the object server, and eventually the probe process will run out of memory and exit.

6 Store and Forward

Store and forward is a feature which allows the probe to store the probes SQL statements to a file, when the object server connection is lost.

6.1 AutoSAF

AutoSAF allows the probe to start-up without any object server connection.

```
# Allow the probe to start-up without an object server connection
AutoSAF                 : 1
```

6.2 Circular Store and Forward

```
# > Circular Store and Forward
NetworkTimeout          : 5
PollServer              : 60
# SAF settings
StoreAndForward         : 2
MaxSAFFileSize          : 10240
SAFPoolSize             : 10
RollSAFInterval         : 90
# Buffering
Buffering               : 1
BufferSize              : 10000
FlushBufferInterval     : 3
# < EoCSaF
```

The multiple file circular store and forward files are best written to a custom probe instance directory, to prevent cluttering the $NCHOME/omnibus/var directory with files.

```
# Create the $OMNIHOME/var/[probe instance] directory
# before running the probe
SAFFileName             : '$NCHOME/omnibus/var/[probe instance]/SAF'
```

7 Command line interface

The probes command line interface supports HTTPS and user authentication.

7.1 HTTP command line interface

The standard HTTP command line interface is enabled in the probe property settings.

```
NHttpd.EnableHTTP                : TRUE
NHttpd.ListeningPort             : 9001
NHttpd.AccessLog                 : "$NCHOME/omnibus/log/simnet.nhttpd.access.log"
```

Example nco_http command.

```
nco_http -uri http://localhost:9001/probe/common
```

7.2 HTTP command line interface with BasicAuth

To add user authentication to the standard HTTP command line interface add theNhttpd.BasicAuth property to the probe property settings.

```
NHttpd.EnableHTTP                : TRUE
NHttpd.ListeningPort             : 9001
Nhttpd.BasicAuth                 : 'username:DJSCkqeVjjjwIx7aDqgOVQ=='
NHttpd.AccessLog                 : "$NCHOME/omnibus/log/simnet.nhttpd.access.log"
```

The Nhttpd.BasicAuth property string takes the format "*username*:*password*", where *username* is any set of characters that does not include a colon (:) and the *password* is the AES-128 hash of the password generated by the nco_crypt utility.
e.g.

```
nco_crypt -aes password
DJSCkqeVjjjwIx7aDqgOVQ==
```

Example nco_http command.

```
nco_http -uri http://localhost:9001/probe/common -username username -password password
```

7.3 HTTPS command line interface

For the HTTPS service, a certificate needs to be created and added to the omni.kdb on on the probe server.

The certificates label and Common Name needs to match the host naming used in the nco_http command.

7.3.1 Create the probes SSL certificate

The example is for localhost.

Note: The CN must be a fully qualified domain name (fqdn) for remote access

```
$NCHOME/bin/nc_gskcmd -certreq -create -db "$NCHOME/etc/security/keys/omni.kdb" -stashed \
-label "localhost" \
-size 2048 \
-dn "CN=localhost,O=IBM,OU=Support,L=SouthBank,ST=London,C=GB" \
-file "$NCHOME/etc/security/keys/localhost_req.arm"
```

Sign the probes certificate using the CA omni.kdb:

```
$NCHOME/bin/nc_gskcmd -cert -sign -db "$NCHOME/etc/security/keys/MASTER_CA.kdb" -stashed \
-label "MASTER_CA" \
-target "$NCHOME/etc/security/keys/localhost_signed.arm" \
-expire 3600 \
-file "$NCHOME/etc/security/keys/localhost_req.arm"
```

Receive the probes signed certificate:

```
$NCHOME/bin/nc_gskcmd -cert -receive -db "$NCHOME/etc/security/keys/omni.kdb" -stashed \
-file "$NCHOME/etc/security/keys/localhost_signed.arm"
```

To check the contents of the keystore:

```
$NCHOME/bin/nc_gskcmd -cert -list -db $NCHOME/etc/security/keys/omni.kdb -stashed
$NCHOME/bin/nc_gskcmd -certreq -list -db $NCHOME/etc/security/keys/omni.kdb -stashed
```

7.3.2 HTTPS command line interface probe properties

```
NHttpd.SSLEnable         : TRUE
NHttpd.SSLListeningPort  : 9002
NHttpd.SSLCertificate    : 'localhost'
# NHttpd.SSLCertificatePwd : 'netcool' # Deprecated in OMNIbus 8.1 fixpack 15
AccessLog                : "$NCHOME/omnibus/log/simnet.nhttpd.access.log"
# EOF
```

The nco_http.props is enabled for HTTPS.

File : $NCHOME/omnibus/etc/nco_http.props

```
NHttpd.SSLEnable: TRUE
```

Example nco_http command.

```
nco_http -uri https://localhost:9002/probe/common
```

7.3.3 HTTPS command line interface probe properties with BasicAuth

```
NHttpd.SSLEnable         : TRUE
NHttpd.SSLListeningPort  : 9002
NHttpd.SSLCertificate    : 'localhost'
Nhttpd.BasicAuth         : 'username:DJSCkqeVjjjwIx7aDqgOVQ=='
# NHttpd.SSLCertificatePwd : 'netcool' # Deprecated in OMNIbus 8.1 fixpack 15
AccessLog                : "$NCHOME/omnibus/log/simnet.nhttpd.access.log"
# EOF
```

Where the password was encrypted using nco_crypt.

```
nco_crypt -aes password
DJSCkqeVjjjwIx7aDqgOVQ==
```

Example nco_http command.

```
nco_http -uri https://localhost:9002/probe/common -username username -password password
```