Tivoli Netcool Supports Guide to IBM CORBA Probes

by

Jim Hutchinson

Document release: 2.0

Table of Contents

1Introduction....2

1.1Overview....2
1.2UNIX file overview....3
1.3Connection Overview....4
1.4Connection Example....5

2Patch considerations....6

2.1Java runtime environment....6
2.2Non-native probe....7
2.3CORBA Framework....8
2.3.1dumpns tool....8
2.4SDK Java....8

3Common Properties....9

3.1Non-Native Probe....9
3.1.1FlushBufferInterval....9
3.1.2NetworkTimeout Property....9
3.2Probe framework....10
3.2.1Command port properties....10
3.2.2Recovering lost connections....10
3.2.3Event Synchronisation....10
3.2.4Internal event queue....10
3.3Additional probe specific properties....11
3.3.1Resynchronisation properties....11
3.3.2Stream capture properties....11

4IOR (Interoperable Object Reference) files....12

4.1IOR Parser Tool....12

5Examples....13

5.1Huawei T2000 probe....13
5.1.1IOR files....13
5.1.2Host, Port, Object 13
5.2Generic TMF814 Probe....14
5.2.1Basic settings....14
5.2.2IOR file properties....14
5.2.3Host, Port, Object properties....14
5.2.4Types of log file messages:....15

6Additional Logging....16

6.1IBM CORBA debug logging....16
6.2Non-native debug logging....16
6.3SSL debug logging....16
6.4CLASSPATH debugging....16

1 Introduction

1.1 Overview

The IBM CORBA probes replace the Visibroker CORBA probes.

To determine if a probe uses CORBA review the Summary section of the probes manual and confirm the connection method is defined as CORBA:-

Connection Method : CORBA

Some examples of CORBA probes

- nco_p_generic_tmf814
- nco_p_alcatel_5620_sam_3gpp_v8
- nco_p_cisco_ctm_corba_v9
- nco_p_alcatel_wnms
- nco_p_lucent_snms
- nco_p_huawei_M2000_corba
- nco_p_huawei_N2000_corba
- nco_p_huawei_T2000_corba
- nco_p_nokia_netact_3gpp_v6
- nco_p_siemens_tnms_TMF814
- nco_p_zte_e300
- nco_p_zte_corba_wcdma
- nco_p_zten31_fixednms_corba
- nco_p_spectrum_corba_v9
- nco_p_eci_lightsoft_tmf814
- nco_p_siemens_corba_v2

IBM CORBA probes use these related packages:

- IBM Object Request Broker (IBM ORB, supplied with Netcool/OMNIbus)
- probe-corba-framework
- probe-sdk-java
- probe-nonnative-base
- probe-command-port

Visibroker CORBA probes uses the Visibroker package, whose related JAR files are:

- vbjorb.jar
- vbsec.jar

With the newer versions of the Non-Native probe script, it is possible to add debug lines to the probes environment script to echo out important details, including the CLASSPATH.
e.g.

```
cd $NCHOME/omnibus/probes/java
vi nco_p_generic_tmf814.env

echo "** nco_p_generic_tmf814_env"
echo "NCO_PROBE_JRE=$NCO_PROBE_JRE"
$NCO_PROBE_JRE/bin/java -version
echo CLASSPATH=$CLASSPATH"
#EOF
```

1.2 UNIX file overview

The IBM CORBA probes install a link to the generic non-native probe patch. The non-native probe handles the connection to the Object Server and initiates the correct PROGRAM JAR file, used to connect to the specific Element Management System (ems).

e.g.

```
nco_p_huawei_M2000_corba -> nco_jprobe
```

Not all probes that use the Non-native probe patch are IBM CORBA probes, but all IBM CORBA probes use the Non-native probe patch.

JAR files used by the Huawei M2000 probe:

JAR file	Product
nco_p_huawei_M2000_corba.jar	Probe specific patch
NSProbe.jar	Non-native probe patch
IntegrationsSupport.jar	IBM CORBA Integrations Support
jlog.jar	Java logging

JAR files used by the Generic TMF814 probe [IBM CORBA probe]:

JAR file	Product
nco_p_generic_tmf814.jar	Probe specific patch
ProbeServices.jar	Non-native probe additional features
NSHeadServices.jar	Non-native probe ObjectServer features
framework.jar	Non-native probe base
TestServices.jar	Service availability support
JSON4J_Apache.jar	Apache JSON support
corba_v2.0.jar	Version specific CORBA framework
CorbaFrameworkBase.jar	CORBA framework base
CorbaFrameworkTMF814.jar	TMF814 CORBA framework

1.3 Connection Overview

The IBM CORBA probes use the CORBA interface to collect data from various Element Management Systems (ems). The connection methods are common, using host, port, object connections, however the design of each Element Management Systems varies.

IOR files can be used to define the host, port, object to be connected to. The IOR files are provided by the Element Management System. The port numbers used by the ems may be dynamic, and it is important to ensure that the customer is aware of this, especially if the CORBA probe is installed remote from the ems.

Example IOR string [blanked to preserve security]:

```
IOR:000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000
```

This string defines the type of service, as well as the host, port and object;

e.g.

```
Name: com/ericsson/nms/cif/service/NMSNAConsumer
Type: IDL:3gppsa5.org/NotificationIRPSystem/NotificationIRPOperations:1.0
Host: 192.168.226.54
Port: 31488
```

Most probes allow IOR or host/port/object name definition in the probes properties files. These properties are mutually exclusive.

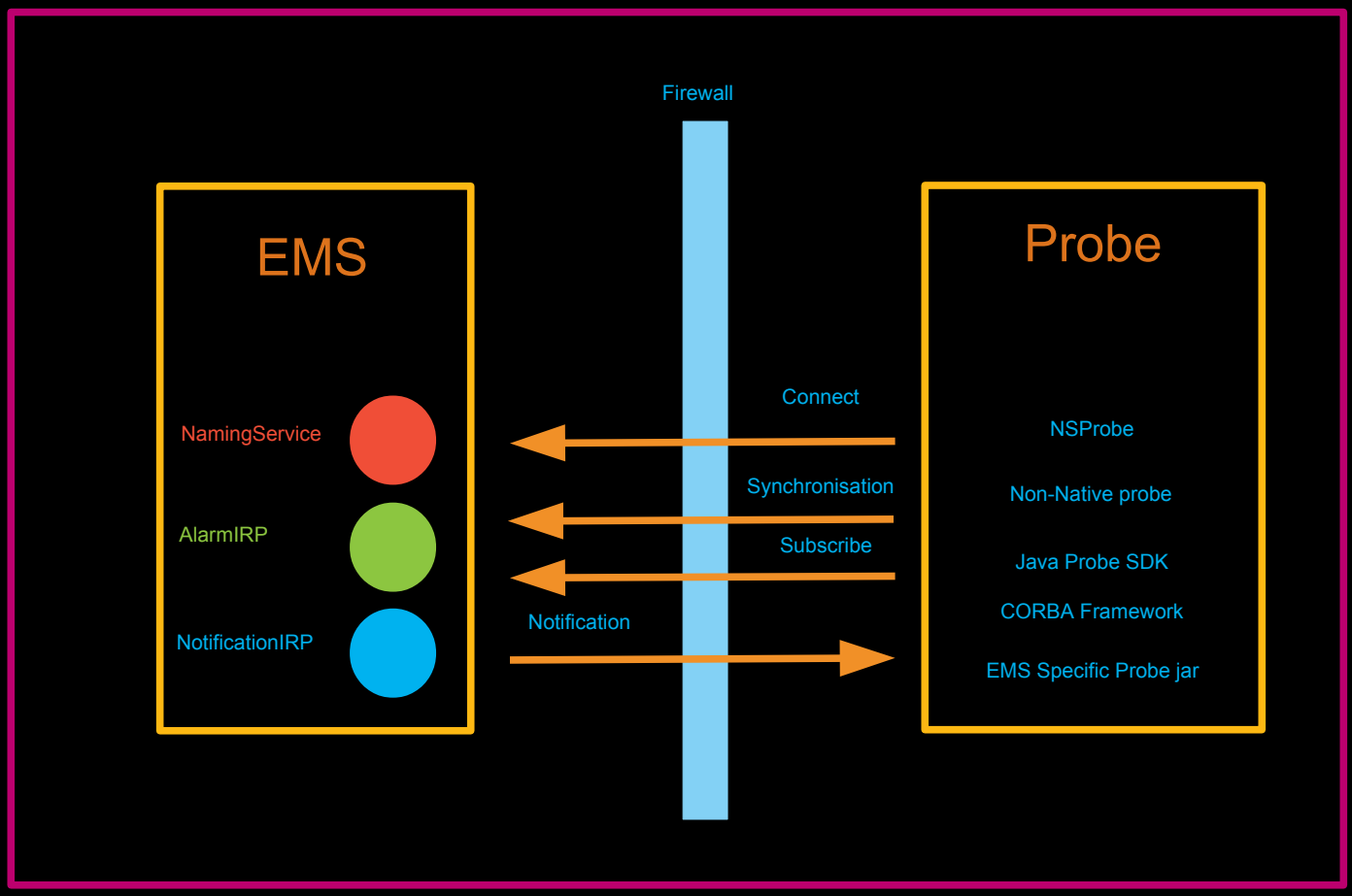
The connection method at this level is over TCP/IP using the IBM CORBA software to create the probes ORB (**Object Requesting Broker)**.

1.4 Connection Example

The IBM CORBA probes are varied and it is not possible to provide a definitive example for all probes. The diagram below illustrates the possible number of connections for the current probe portfolio and how the connection from the IBM CORBA probe to the Element Management System is tiered through a number of components.

When firewalls exist between the IBM CORBA probe and Element Management System, it becomes necessary to define the port numbers being used by both the probe and Element Management System.

The probes connectivity can be defined using ORBLocalHost and ORBLocalPort, where these properties are available. If they are not available, a request for enhancement must be raised.



2 Patch considerations

2.1 Java runtime environment

Ensure that the correct version of java is available as defined in the probes readme file or in the probes documentation;

You can set the correct version of java using the NCO_PROBE_JRE variable in the $PROGRAM.env file where PROGRAM is the probes script name.

e.g.

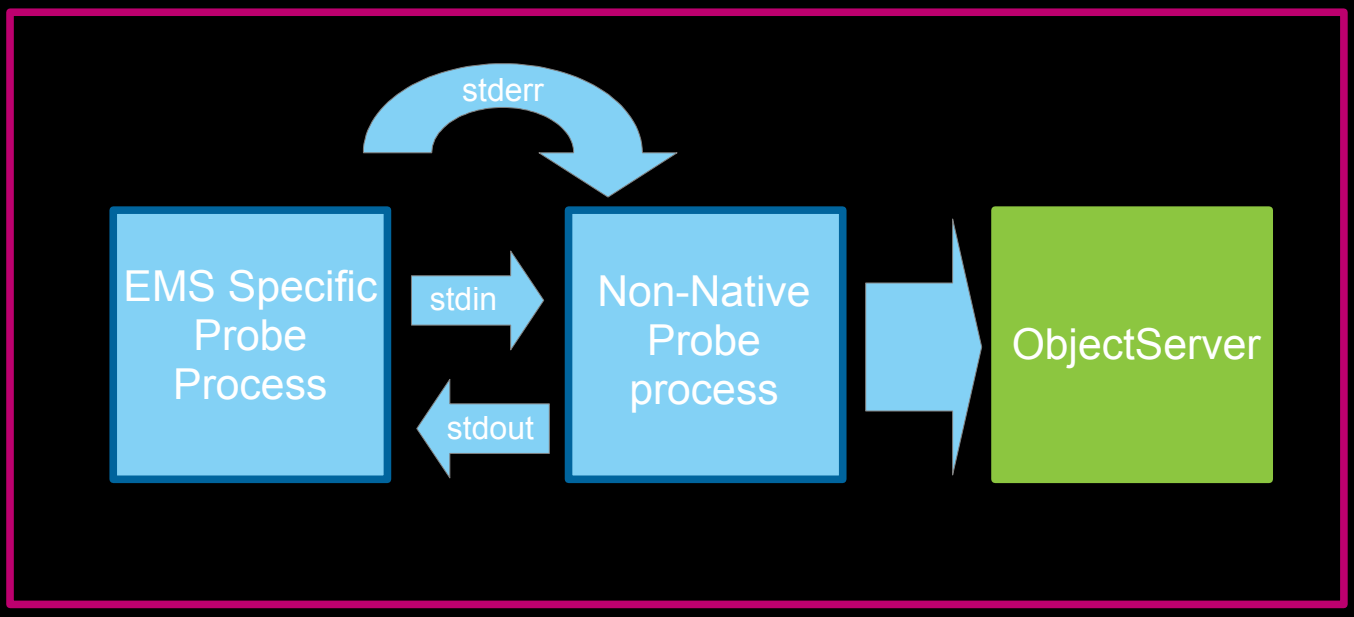
```
cd $NCHOME/omnibus/probes/java
vi nco_p_huawei_M2000_corba.env
#! /bin/sh

. $OMNIHOME/probes/java/set_ibm_jre.sh
# Specified java directory explicitly
NCO_PROBE_JRE=$NCHOME/platform/solaris2/jre_1.6.7/jre
#EOF
```

2.2 Non-native probe

The Non-native probe is used to allow java based probe integrations to communicate with the Object Server. The probe package may be upgraded independently of the EMS specific probe package, and it is advised that the latest non-native probe patch is installed following an installation of the EMS specific probe package, unless advised otherwise by the probes README file or online documentation. Enhancements to the non-native probe patch include performance improvements, the FlushBufferInterval property and probe specific environment configuration.

Overview of Non-native probe communications



Related files:

$NCHOME/omnibus/probes/<*platform*>/nco_p_nonnative

$NCHOME/omnibus/probes/java:

- NSProbe.jar

2.3 CORBA Framework

The CORBA framework was created to replace the Visibroker software used in earlier probes. It makes use of the IBM CORBA features included with IBM Java, as well as Apache JACORB. The dumpns command is provided as a supported replacement for the earlier Visibroker based DumpNameServices utility.

The following JAR files are included in the CORBA framework:

$NCHOME/omnibus/probes/java/corba:

- corba_v2.0.jar
- CorbaFrameworkBase.jar
- CorbaFrameworkTMF814.jar

$NCHOME/omnibus/probes/java/corba/jacorb-3.3/lib:

- jacorb.jar
- slf4j-api-1.6.4.jar
- slf4j-jdk14-1.6.4.jar

2.3.1 dumpns tool

The dumpns tool is provided with the CORBA Framework version 3 to allow the contents of the Naming Service to be output to the commend line. It uses the Apache JACORB support jar to connect to the naming service, which is provided with the latest CORBA framework package.

Example:

```
cd $NCHOME/omnibus/probes/java/corba
./dumpns nshost nsport
```

Where
nshost is the IP Address or FQDN for the Naming Service host
nsport is the port number for the Naming Service

With the output from dumpns containing:

```
INFO: ClientConnectionManager: found ClientGIOPConnection to nshost:nsport
   NotificationService
   AlarmIRP
   NotificationIRP
```

2.4 SDK Java

The SDK Java package provides the supporting JAR files for common Java probe features.

$NCHOME/omnibus/probes/java:

- framework.jar
- IntegrationsSupport.jar
- JSON4J_Apache.jar
- NSHeadServices.jar
- ProbeServices.jar
- TestServices.jar

3 Common Properties

3.1 Non-Native Probe

3.1.1 FlushBufferInterval

The FlushBufferInterval property allows a maximum time to be set for which events are held in the non-native probes buffer, when Buffering is enabled;
e.g. if the probe rate is around 10 events per second [EPS]

```
Buffering           : 1
BufferSize          : 100
FlushBufferInterval : 11

BufferSize =~ EPS * FlushBufferInterval
```

Where EPS is the events per second, and the interval is set to the nearest prime or odd number to the required value.
The buffer can be filled during the FlushBufferInterval, with events being sent to the object server, but the FlushBufferInterval should be set to a required maximum amount of time that is acceptable for events to be delayed at the probe level.

It's recommended to set the FlushBufferInterval property to values greater than 10 seconds, however, if the probe connects to a dedicated collection layer object server, the FlushBufferInterval property can be set to 1 to ensure event delivery is not delayed due to buffering.

From a performance perspective, sending events in large blocks should be more efficient than sending events one at a time or in smaller blocks. However, there is a point whereby the loading of the object server with event blocks reaches a levelling point; this point is different for every system.

3.1.2 NetworkTimeout Property

The NetworkTimeout is set to zero by default, and can result in the probe being unable to notice when an ObjectServer connection is lost, due to network problems. It is therefore best practice to set NetworkTimeout to some value even when the probe does not connect a backup ObjectServer.

e.g.

```
NetworkTimeout : 120
```

When a backup ObjectServer is connected to, NetworkTimeout should be less than PollServer, otherwise the poll will never discover the Primary ObjectServer has returned, preventing failback from occurring.

Default setting:

```
NetworkTimeout                  : 0
```

3.2 Probe framework

3.2.1 Command port properties

The command port should be set to any available port on the system. If the probe is being used within a firewall environment, the firewall administrator must allocate a port or be notified of the allocated port. If a number of custom user tools are being used, the the CommandPortLimit may need to be increased along with the number of connections setting for the nco_pad process.

e.g.

```
nco_pad -connections 128
```

To check the port is available on UNIX use:

```
netstat -na | grep 6970
```

Default:

```
CommandPort                    : 6970
CommandPortLimit               : 10
```

3.2.2 Recovering lost connections

The probe includes a number of features to ensure that the connection to the EMS is maintained and is recoverable. By default the probe does not exit due to inactivity or retry the connection. The probe checks the EMS connection periodically every 60 seconds by default.

Example settings:

```
RetryCount                     : 3
RetryInterval                  : 11
HeartbeatInterval              : 60
Inactivity                     : 3600
```

Default:

```
RetryCount                     : 0
RetryInterval                  : 0
HeartbeatInterval              : 60
Inactivity                     : 0
```

3.2.3 Event Synchronisation

By default the probe does not attempt to synchronise the events in the EMS.

Default:

```
InitialResync                  : 'false'
ResyncInterval                 : 0
```

3.2.4 Internal event queue

The internal event queue, which is the number of events allowed to be held between the EMS specific probe jar and the non-native probe jar, is set to 10,000 by default. If the probe logs NSProbe messages regarding the queue being full, it is best to examine why the probe is unable to process the event volumes, as it is an indication that there is a performance problem at the ObjectServer or possibly the probe server.

Default:

```
MaxEventQueueSize              : 10000
```

3.3 Additional probe specific properties

Probes have specific properties that can be set to improve performance and gather data.

For example the Generic TMF814 probe:

3.3.1 Resynchronisation properties

The size of the batches collected from the EMS can affect performance, and the probes Buffering settings should be adjusted according to the batch size, so as to ensure that data is processed efficiently. The Generic TMF814 probe allows basic filtering, with the filtering being exclusion filters.

For example, to exclude cleared, indeterminates and minor alarms use:

```
ResyncSeverityFilter              : 'PS_CLEARED;PS_INDETERMINATE;PS_MINOR'
```

Default settings:

```
ResyncBatchSize                   : 100
ResyncProbableCauseFilter         : ''
ResyncSeverityFilter              : ''
```

3.3.2 Stream capture properties

The Stream capture files can be used to check what data arrives via the CORBA interface and is useful for in depth analysis of the event data. It is not a feature that is useful in a production environment.

Default settings:

```
StreamCapture                     : 0
StreamCaptureFilePath             : ''
```

4 IOR (Interoperable Object Reference) files

IOR (Interoperable Object Reference) files are used by CORBA (Common Object Requesting Broker Architecture) to define objects. CORBA is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together.

CORBA uses an interface definition language (IDL) to specify the interfaces that CORBA objects present to the outside world. CORBA implementation are usually written in C++ or Java.

The IOR files are provided by the ems (Element Management System) the probe connects to, when using the CORBA interface. Some IOR files are dynamic, and are subject to change, usually when the ems is restarted, and may need to be updated.

Most CORBA probes allow IOR files to be replaced by the host, port, object instead. The definition of each CORBA service (host/port/object) is usually provided by the ems, or else the current IOR files can be gathered manually by the ems administrator.

Example:-

Huawei M2000 probe

```
EntryPointIrpFtpCommand       : "ftp://netcool:password@192.168.22.31/"
EntryPointIrpFile             : "/opt/m2000/etc/entry.ior"
FtpPassword                   : "ECEDBJAGFKDIGD"
```

4.1 IOR Parser Tool

IOR File Parser allows the host/port/object name to be confirmed;

http://www2.parc.com/istl/projects/ILU/parseIOR/

e.g.

```
IOR:000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000
```

Might return:-

```
Name: com/ericsson/nms/cif/service/NMSNAConsumer
Type: IDL:3gppsa5.org/NotificationIRPSystem/NotificationIRPOperations:1.0
Host: 192.168.226.54
Port: 31488
```

5 Examples

5.1 Huawei T2000 probe

Example configuration of the Huawei T2000 probe;

```
Loading IOR file
</opt/CiscoTransportManagerServer/openfusion/domains/OpenFusion/localhost/NameService/NameSingleton/NameSingleto
n.ior>
Attempting to decode Naming Service IOR
Failed to parse IOR
Converting string IOR to reference
Narrowing reference to NamingContext
Narrowed reference.
Dumping contents of Naming Service
(Format is ContextName.ContextKind/ContextName2.ContextKind2/ObjectName.ObjectKind)
Name: TMF_MTNM.Class/Cisco Systems.Vendor/
CiscoTransportManager.EMSInstance/7_0.Version/CTM.EMS/SessionFactory.EmsSessionFactory
Type: IDL:mtnm.tmforum.org/emsSessionFactory/EmsSessionFactory_I:1.0
Host: 192.168.1.6
Port: 61696
IOR:000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000
Name: TMF_MTNM.Class/Cisco Systems.Vendor/Cisco Transport
Manager.EMSInstance/7_0.Version/CTM.EMS/NotificationChannel
Type: IDL:omg.org/CosNotifyChannelAdmin/EventChannel:1.0
Host: 192.168.1.6
Port: 9472
IOR:000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000
Name: services/NotificationService
Type: IDL:omg.org/CosNotifyChannelAdmin/EventChannelFactory:1.0
Host: 192.168.1.6
Port: 9472
IOR:000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000
End of Naming Service Dump
```

5.1.1 IOR files

Property file example:

```
IORFile                  : "/opt/netcool/omnibus/var/EmsSessionFactory.ior"
NamingContextIORFile     : "/opt/netcool/omnibus/var/NameSingleton.ior"
```

5.1.2 Host, Port, Object

Property file example:

```
ORBInitialHost     : "192.168.1.6"
ORBInitialPort     : 32805
NamingContextPath : "TMF_MTNM.Class/Cisco Systems.Vendor/Cisco
TransportManager.EMSInstance/7_0.Version/CTM.EMS/SessionFactory.EmsSessionFactory"
```

5.2 Generic TMF814 Probe

The EMS connection can be checked using the dumpns tool if the EMS administrator is unable to confirm details or when the eMS connectivity needs to be confirmed.
e.g.
./dumpns emsserver.company.com 56468
…
INFO: ClientConnectionManager: found ClientGIOPConnection to <*IP-ADDRESS*>:56468 (c4ef1)
NotificationService
AlarmIRP
NotificationIRP
EMSSessionFactory
...

5.2.1 Basic settings

Ensure that the correct version and character encodings are set.
It is good practice to set the ORBLocalHost and ORBLocalPort event when there is no firewall between the EMS and the probe, as it ensures that the local port is known.
e.g.

```
# EMS Version details
ReleaseTMF814                   : 'V3.0'
# Character encoding
ORBCharEncoding                 : 'ISO8859_1'
ORBWCharDefault                 : 'UTF16'
#
ORBLocalHost                    : 'probeserver.company.com'
ORBLocalPort                    : 9876
```

5.2.2 IOR file properties

The IOR files from the EMS can used to test probe connectivity:

e.g.

```
NamingServiceIORFile  : '/opt/IBM/tivoli/netcool/omnibus/var/ns.ior'
IORFile               : '/opt/IBM/tivoli/netcool/omnibus/var/emssessionfactory.ior'
```

5.2.3 Host, Port, Object properties

It is recommended that the host, port, and object name are used to define the EMS connection as this will ensure that probe can connect, even when the IOR files are updated, usually following an EMS restart.

e.g.

```
NamingServiceHost     : 'emsserver.company.com'
NamingServicePort     : 56468
NamingContextPath     : 'EMSSessionFactory'
```

5.2.4 Types of log file messages:

- Invalid port:

```
Error: E-JPR-000-000: Failed to get IOR Object : org.omg.CORBA.ORBPackage.InvalidName:
NameService:org.omg.CORBA.TRANSIENT: java.net.ConnectException: Connection refused:host=<IP>,port=5646  vmcid:
IBM  minor code: E02  completed: No
```

- Invalid listening port:

```
Resolving initial references to NamingService
com.ibm.tivoli.netcool.omnibus.probe.services.impl.SimpleCommandService$BidiThread.run
ENTERING
com.ibm.tivoli.netcool.omnibus.probe.services.impl.SimpleCommandService$BidiThread.run
EXITING
…
timeout
```

- Valid host/port, invalid object name:

```
Successfully narrowed reference
Resolving System reference
Failed to get the System reference from the naming service! :
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0
Failed to resolved to Naming Context :org.omg.CosNaming.NamingContextPackage.NotFound:
IDL:omg.org/CosNaming/NamingContext/NotFound:1.0
```

- Incorrect object name:

```
Resolving System reference
Retrieveing EMS Session via IOR object...
Failed to get IOR Object : initial and forwarded IOR inaccessible
Failed to connect: org.omg.CORBA.TRANSIENT: initial and forwarded IOR inaccessible
vmcid: IBM  minor code: E07  completed: No
```

6 Additional Logging

6.1 IBM CORBA debug logging

Edit probes environment script and add the following:

```
vi $NCHOME/omnibus/probes/java/nco_p_<probe>.env
# IBM CORBE Debug logging
NCO_JPROBE_JAVA_FLAGS="-Dcom.ibm.CORBA.Debug=true -Dcom.ibm.CORBA.CommTrace=true
-Dcom.ibm.CORBA.Debug.Output=$OMNIHOME/log/ibm_orb_trace.log $NCO_JPROBE_JAVA_FLAGS"
#EOF
:wq
```

6.2 Non-native debug logging

Edit probes environment script and add the following:

```
vi $NCHOME/omnibus/probes/java/nco_p_<probe>.env

# ENABLE NON-NATIVE PROBE DEBUG LOGGING
NDE_DEFAULT_LOG_LEVEL=debug
NDE_FORCE_LOG_MODULE=$OMNIHOME/log/nonnative_forced.log
NCO_P_NONNATIVE_TRANSCRIPT=$OMNIHOME/log/nonnative_debug.log
export NDE_DEFAULT_LOG_LEVEL NDE_FORCE_LOG_MODULE NCO_P_NONNATIVE_TRANSCRIPT
#EOF
:wq
```

6.3 SSL debug logging

Edit probes environment script and add the following:

```
vi $NCHOME/omnibus/probes/java/nco_p_<probe>.env
# SSL debug logging
NCO_JPROBE_JAVA_FLAGS="-Djavax.net.debug=ssl:handshake:verbose $NCO_JPROBE_JAVA_FLAGS"
#EOF
```

6.4 CLASSPATH debugging

It may be necessary to examine the CLASSPATH at the point the probe is exec'ed.
If this is required, then the non-native probe script must be edited, rather than the probe specific environment script.
e.g.

```
cd $NCHOME/omnibus/probes
mv nco_p_generic_tmf814 nco_p_generic_tmf814.ga
cp nco_p_generic_tmf814.ga nco_p_generic_tmf814
vi nco_p_generic_tmf814
# Added CLASSPATH listing before exec block to debug
echo "** CLASSPATH:"
echo $CLASSPATH | awk -F: '{for(i=1;i<=NF;i++)print $i}'
echo "**"
:wq
```

After troubleshooting, remember to revert back to the original GA script to prevent patching problems.