Tivoli Netcool Support's Guide to the Microsoft EWS Probe

by

Jim Hutchinson

Document release: 2.2

Table of Contents

1 Introduction

1.1 Overview

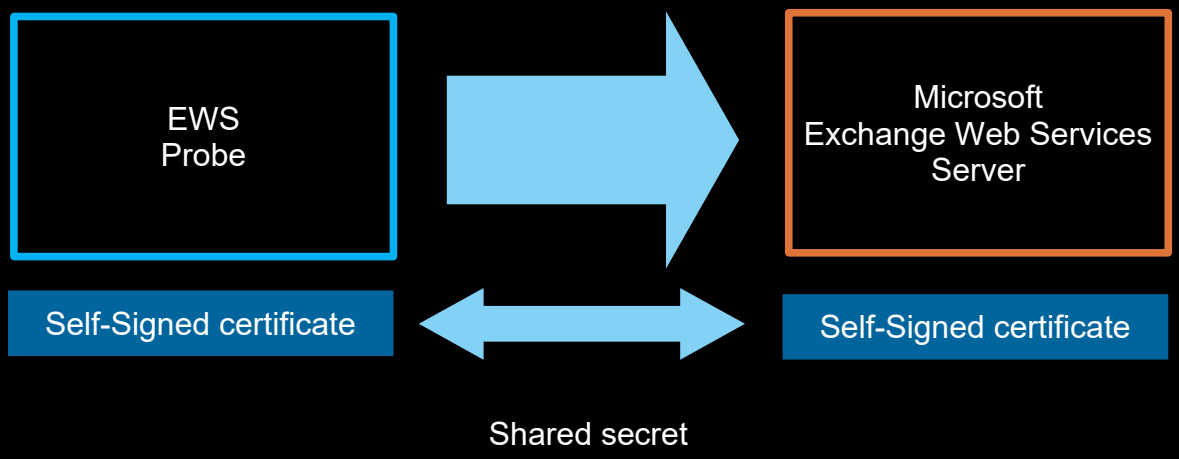
The Microsoft EWS probe manual is the best point of reference for the configuring the EWS probe. The Support's guide to the EWS probe is provided as a supplement to the main documentation.

The Microsoft Exchange Web Services probe connects to the Microsoft Exchange Server using a dedicated user and a number of authentication methods.

1.2 Connecting to the EWS Server

OAuth2 uses the shared secret method to obtain server trust. The simplest method is to share a self-signed certificate with the EWS server.

For OAuth2 authentication, use either ClientCertification or ClientSecret, but not both. If both are specified, the probe will use the ClientSecret setting.



1.3 Best practice property settings

By using the best practice settings, problems with probe behaviour in extreme conditions can be avoided.

```
# Best practice
NetworkTimeout          : 15
PollServer              : 60
# Buffer settings
# BufferSize / FlushBufferInterval equals approximate expected EPS
Buffering               : 1
BufferSize              : 200
FlushBufferInterval     : 9
# Heartbeating
ProbeWatchHeartbeatInterval : 60
# Tuning
DisableDetails          : 1
MaxEventQueueSize       : 50000
# To disable the pid lock file
DisablePidFileLock      : 'true'
```

2 Connection authentication methods

2.1 Basic authentication

Review the manual and confirm the settings are as discussed here.

Connecting using Basic authentication

To connect using Basic authentication mode, configure the following probe properties:

• AuthenticationType
Set this property to Basic:
```
AuthenticationType : 'Basic'
```

• ServiceURL
The URL for the Exchange Server or Exchange Online service.
```
ServiceURL : 'https://outlook.office365.com/EWS/Exchange.asmx'
```

If ServiceURL is not specified, the probe will perform AutoDiscovery to detect the service URL (Exchange Online only).

• Username
The email address of the user whose mailbox is to be accessed by the probe.
```
Username : 'exampleuser@ewsprobe.microsoft.com'
```

• Password
The password of the user whose mailbox is to be accessed by the probe.
```
Password : 'password'
```

• TrustStore
The full path to the truststore used for TLS authentication (Exchange Server only).
```
TrustStore : '/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/ewsporbe/ms-truststore.jks'
```

• TrustStorePassword
The password to access the truststore used for TLS authentication (Exchange Server only).
```
TrustStorePassword : 'password'
```

To check the required certificates on the server use openssl.

```
openssl s_client -connect outlook.office365.com:443
```

The default HTTPS port is 443.

Example return.

```
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
depth=1 C = US, O = DigiCert Inc, CN = DigiCert Cloud Services CA-1
depth=0 C = US, ST = Washington, L = Redmond, O = Microsoft Corporation, CN =
outlook.com
---
Certificate chain
 0 s:/C=US/ST=Washington/L=Redmond/O=Microsoft Corporation/CN=outlook.com
   i:/C=US/O=DigiCert Inc/CN=DigiCert Cloud Services CA-1
 1 s:/C=US/O=DigiCert Inc/CN=DigiCert Cloud Services CA-1
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Global Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
...
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
```

It is best to obtain the required certificates from Microsoft or the Exchange servers administrator.

The JKS store file can be created in a number of ways, depending on the format of the certificates provided.

2.2 ClientCertification OAuth2 authentication

For OAuth2 Authentication, you can use either ClientCertification or ClientSecret.
The ClientCertification method is the simplest to implement.

The ClientCertification Oauth2 authentication uses a self-signed certificate to identify the EWS probe to Exchange server.

2.2.1 ClientCertification

Create the probe instance directory.

```
mkdir $NCHOME/omnibus/probes/linux2x86/ewsprobe
```

Create the probe instances PKCS12 keystore using the probes Java:

```
$NCHOME/platform/linux2x86/jre64_1.8.0/jre/bin/keytool -genkey \
-alias ews_oauth2 \
-storetype PKCS12 \
-keyalg RSA \
-keystore $NCHOME/omnibus/probes/linux2x86/ewsprobe/ews_oauth2_keystore.p12 \
-validity 3650


Enter keystore password: <password>
Re-enter new password: <password>
What is your first and last name?
[Unknown]: <FQDN> or <instance user>
What is the name of your organizational unit?
[Unknown]: Support
What is the name of your organization?
[Unknown]: IBM
What is the name of your City or Locality?
[Unknown]: New York
What is the name of your State or Province?
[Unknown]: New York
What is the two-letter country code for this unit?
[Unknown]: US

Is CN=<FQDN>, OU=Support, O=IBM, L=Unknown, ST=New York, C=US correct?
(type "yes" or "no") [no]: yes


ls -l $NCHOME/omnibus/probes/linux2x86/ewsprobe/ews_oauth2_keystore.p12
```

Extract the public key from the keystore file and export to .crt format to upload to Azure Active Directory.

```
openssl pkcs12 \
-in $NCHOME/omnibus/probes/linux2x86/ewsprobe/ews_oauth2_keystore.p12 \
-clcerts -nokeys \
-out $NCHOME/omnibus/probes/linux2x86/ewsprobe/ews_oauth2_keystore.crt

Enter Import Password: <password>
MAC verified OK

ls -l $NCHOME/omnibus/probes/linux2x86/ewsprobe/ews_oauth2_keystore.crt
```

Follow the steps in the manual for the 'Connecting using OAuth2 authentication' and the Microsoft Exchange server, to obtain the strings related to the uploaded ews_oauth2_keystore.crt.

Probe property settings

```
# Mail server login
ServiceURL : 'https://outlook.office365.com/EWS/Exchange.asmx'
Username   : 'netcooluser@company.com'
Password   : '<netcooluser-password>'
#
# Connect using OAuth2
#
AuthenticationType : 'OAuth2'
Authority  : 'https://login.microsoftonline.com/########-####-####-####-############'
Scope      : 'https://outlook.office.com/.default'
#
# Connect using the ClientId from the Azure Active Directory
#
# ClientId          : '########-####-####-####-############'
#
# OR
#
# Using the shared self-signed certificate
#
ClientCertificate : '$NCHOME/omnibus/probes/linux2x86/ewsprobe/ews_oauth2_keystore.p12'
ClientCertificatePassword : '<password>'
```

2.3 ClientSecret OAuth2 Authentication

The ClientSecret method tends to involve a copy and paste of text from the Microsoft user interface which can be problematic due to character encodings. If there are problems, type out the string from the Microsoft user interface into the probe property file before debugging further.

The ClientCertification method avoids these issues.

Follow the steps in the manual for the 'Connecting using OAuth2 authentication' and the Microsoft Exchange server.

Probe property settings

```
# Mail server login
ServiceURL : 'https://outlook.office365.com/EWS/Exchange.asmx'
Username   : 'netcooluser@company.com'
Password   : '<netcooluser-password>'
#
# Connect using OAuth2
#
AuthenticationType : 'OAuth2'
Authority  : 'https://login.microsoftonline.com/########-####-####-####-############'
Scope      : 'https://outlook.office.com/.default'
#
# Connect using the ClientId from the Azure Active Directory
# ClientId          : '########-####-####-####-############'
# OR
# Using the shared self-signed certificate
# ClientCertificate : '$NCHOME/omnibus/probes/linux2x86/ewsprobe/ews_keystore.p12'
# ClientCertificatePassword : '<password>'
# ClientSecret
#
ClientId           : 'xxxxxx-xxxxx-xxxx-xxxx-xxxxxxxxxxx'
ClientSecret       : 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
#
# It is best to take a copy of the Java cacerts if being used in the probe property file
#
# To list the keystrore
# keytool -list -keystore $NCHOME/omnibus/probes/linux2x86/cacerts.jks -storepass changeit
# To check certificates use openssl
# openssl s_client -connect host:port
#
TrustStore         : "$NCHOME/platform/linux2x86/jre64_1.8.0/jre/lib/security/cacerts"
TrustStorePassword : "changeit"
TrustStoreType     : "JKS"
```

3 Connecting via a Proxy Server

The EWS probe supports setting the Web Proxy server settings in the probe property settings. Additional settings are required and can be set in the EWS probes environment file.

3.1 Proxy Server property settings

The probe support Web proxy settings for connections via a proxy server.

```
ProxyDomain
Use this property to specify the domain of the web proxy server.
The default is "".

ProxyHost
Use this property to specify the web proxy server address when its usage is required.
The default is "".

ProxyPort
Use this property to specify the port number to use in web proxy server.
Default is 80.

ProxyUserName
Use this property to specify the username for the web proxy credential.
The default is "".

ProxyPassword
Use this property to specify the password for the web proxy credential.
The default is "".
```

3.2 Configuring the EWS probe

For the EWS probe to connect to the Microsoft service via a proxy server, the following is required.

- The proxy server must support the HTTP, HTTPS, SOCKS protocols
- The proxy server must be able to perform internet DNS lookups

Additionally, the probe server must be able to perform DNS lookups, either via the proxy server or a DNS server.

To check that the probe server can perform internet DNS lookups use nslookup.
For example.

```
nslookup google.com
```

The EWS probes environment file requires these additional settings.

```
File : $NCHOME/omnibus/probes/java/nco_p_ews.env

###
# Enable PROXY server Host and Port for HTTPS and SOCKS
NCO_JPROBE_JAVA_FLAGS="-Dhttps.proxyHost=192.168.2.1 -Dhttps.proxyPort=8080 ${NCO_JPROBE_JAVA_FLAGS}"
NCO_JPROBE_JAVA_FLAGS="-Djdk.http.auth.tunneling.disabledSchemes='' ${NCO_JPROBE_JAVA_FLAGS}"
NCO_JPROBE_JAVA_FLAGS="-DsocksProxyHost=192.168.2.1 -DsocksProxyPort=9090 ${NCO_JPROBE_JAVA_FLAGS}"
```

The EWS probe property settings define the Web Proxy Host and Port.

```
###
# Web Proxy settings
# ProxyDomain: ""
ProxyHost: "192.168.2.1"
ProxyPort: 8080
# ProxyUserName: ""
# ProxyPassword: ""
```

Here all of the proxy server services are on a single host and port.
The IP Address must be used for the proxy server settings.

3.3 Example debug logging

```
Information: I-JPR-000-000: Probe started
Information: I-UNK-000-000: Probewatch: Running ...
...
Debug: D-UNK-000-000: 0 buffered alerts
Information: I-JPR-000-000: No target has been registered with the CommandService
...
Information: I-JPR-000-000: Exchange service is using Web Proxy connection
Debug: D-JPR-000-000: [OAuthProvider] Using Client Secret to request for token...
Debug: D-JPR-000-000: [OAuthProvider] Could not get token from cache, Reason: Token not found in the cache
Information: I-JPR-000-000: [OAuthProvider] Successfully acquired new access token from Microsoft Azure.
Debug: D-JPR-000-000: [OAuthProvider] Access token will expire at YYYY-MM-DD HH:MM:SS
Information: I-JPR-000-000: [OAuthProvider] Successfully updated ExchangeService.
Information: I-JPR-000-000: [OAuthProvider] Successfully completed OAuth authentication with Microsoft Azure.
2023-09-14T08:34:57: Information: I-JPR-000-000: Connected to Exchange Web Services
2023-09-14T08:34:57: Debug: D-JPR-000-000: Exchange server has Version x.x.x.x (schema VYYYY_MM_DD)
2023-09-14T08:34:57: Debug: D-JPR-000-000: Subscribing for EWS notifications on new email
Debug: D-JPR-000-000: Subscribed to EWS notifications successfully
...
Information: I-JPR-000-000: Resynchronizing Probe
...
Debug: D-JPR-000-000: Scheduled resync time = <n> : Time now =  <n>
Information: I-JPR-000-000: Waiting on resync to finish
Debug: D-JPR-000-000: Received no folder item changes
Information: I-JPR-000-000: No new email to process
Debug: D-JPR-000-000: Probe resynchronization took [nnnn msecs]
Information: I-JPR-000-000: Resync finished
...
Information: I-JPR-000-000: Probe connected
...
Information: I-JPR-000-000: RESYNCHREQUEST 'Changes in mailbox folder is detected, performing resynchronization
to check for any new email'
...
Information: I-JPR-000-000: Resynchronizing Probe
Debug: D-JPR-000-000: Received a batch of 1 folder item changes
Debug: D-JPR-000-000: Inspecting 1 new candidate email(s)...
Debug: D-JPR-000-000: Filtered email with subject of "#NETCOOL - test email" and ID of [xxxx] which does not
match configured filter due to field value is not matched
Information: I-JPR-000-000: Loaded 0 new qualified email(s) for further parsing
Debug: D-JPR-000-000: Probe resynchronization took [nnnn secs]
```

4 Creating a Microsoft365 keystore

The full microsoft365 certificates are provided on the Microsoft support wite. These certificates can be used to create an all encompassing keystore for use with the email probe and EWS probe.

4.1.1 Using the import_all_pems.sh script to create a JKS store file

ALL m365 ROOT CAs:

```
./import_all_pems.sh m365_root_certs.pem
$NCHOME/platform/linux2x86/jre64_1.8.0/jre/bin/keytool netcool microsoft365.jks
```

ALL m365 inrermediate CAs:

```
./import_all_pems.sh m365_intermediate_certs.pem
$NCHOME/platform/linux2x86/jre64_1.8.0/jre/bin/keytool netcool microsoft365.jks
```

office365 certificate:

```
./import_all_pems.sh office365.cert
$NCHOME/platform/linux2x86/jre64_1.8.0/jre/bin/keytool netcool microsoft365.jks
```

Check the JKS store file:

```
keytool -list -keystore microsoft365.jks -storepass netcool | grep -v finger
```

4.1.2 Obtaining a valid certificate chain

The server certificate can be found using the following method.

Get the office365.cert using openssl.

```
openssl s_client -connect outlook.office365.com:993
```

Add the certificate to a file, office365.cert.

```
keytool -printcert -file office365.cert

Owner: CN=outlook.com, O=Microsoft Corporation, L=Redmond, ST=Washington, C=US
Issuer: CN=DigiCert Cloud Services CA-1, O=DigiCert Inc, C=US
Serial number: 49c38e3545b3a0715289f8bfeb52004
Valid from: Tue Dec 21 16:00:00 PST 2021 until: Thu Dec 22 15:59:59 PST 2022
```

The Microsoft 365 encryption chains are obtained from microsoft:

```
https://docs.microsoft.com/en-us/microsoft-365/compliance/encryption-office-365-
certificate-chains?redirectSourcePath=%252fen-us%252farticle%252foffice-365-
certificate-chains-0c03e6b3-e73f-4316-9e2b-bf4091ae96bb&view=o365-worldwide

Download:
m365_intermediate_certs_20201013.p7b
m365_root_certs_20201012.p7b
```

4.1.3 Example manual command line usage

Check files:

```
openssl pkcs7 -print_certs -in m365_intermediate_certs_20201013.p7b
openssl pkcs7 -print_certs -in form der -inm365_root_certs_20201012.p7b
```

Create PEMS from pkcs7 files.

ALL intermediate CA's.

```
openssl pkcs7 -print_certs -in m365_intermediate_certs_20201013.p7b | grep -v subject
| grep -v issuer | grep . > m365_intermediate_certs.pem
```

ALL ROOT CA's.

```
openssl pkcs7 -print_certs -inform der -in m365_root_certs_20201012.p7b | grep -v
subject | grep -v issuer | grep . > m365_root_certs.pem
```

Get m365 server certificate.

```
openssl s_client -connect outlook.office365.com:993

CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, CN = DigiCert Cloud Services CA-1
verify return:1
depth=0 C = US, ST = Washington, L = Redmond, O = Microsoft Corporation, CN =
outlook.com
verify return:1
---
Certificate chain
 0 s:/C=US/ST=Washington/L=Redmond/O=Microsoft Corporation/CN=outlook.com
   i:/C=US/O=DigiCert Inc/CN=DigiCert Cloud Services CA-1
 1 s:/C=US/O=DigiCert Inc/CN=DigiCert Cloud Services CA-1
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Global Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
…
-----END CERTIFICATE-----
subject=/C=US/ST=Washington/L=Redmond/O=Microsoft Corporation/CN=outlook.com
issuer=/C=US/O=DigiCert Inc/CN=DigiCert Cloud Services CA-1

# EOF
```

4.2 Create Intermediates

```
File : create_intermediate_pem_from_p7b.sh
#! /bin/sh
#
# Create Intermediate CA PEM file from p7b file
#
if [ $# -ne 1 ]
then
echo "Usage: `basename $0` [filename]"
echo
echo "Where the input file is filename.p7b"
exit
fi

MYPEMFILE=$1

# Set variables
export INPUTFILE OUTPUTFILE MYPEMFILE
INPUTFILE=${MYPEMFILE}.p7b
OUTPUTFILE=${MYPEMFILE}.pem

if [ ! -f ${INPUTFILE} ]
then
echo "Cannot locate file : ${INPUTFILE}"
exit
fi

if [ -f ${OUTPUTFILE} ]
then
echo "Remove or renanme file : ${OUTPUTFILE}"
echo "and try again"
exit
fi

# Check file contents
echo "CONTENTS -->"
openssl pkcs7 -print_certs -in ${INPUTFILE}
echo "<-- END CONTENTS"

# Confirm conversion
echo "Continue? <retrun or contrl-c to exit>"
read ans

# Convert to PEM
openssl pkcs7 -print_certs -in ${INPUTFILE} | grep -v subject | grep -v issuer | grep
. > ${OUTPUTFILE}

if [ -f ${OUTPUTFILE} ]
then
echo "Created file ${OUTPUTFILE}"
ls -l ${OUTPUTFILE}
else
echo "File does not exist : ${OUTPUTFILE}"
fi
# EOF
```

4.3 Create Roots

File : create_root_pem_from_p7b.sh

```
#! /bin/sh
#
# Create ROOT CA PEM file from MS p7b file
#
if [ $# -ne 1 ]
then
echo "Usage: `basename $0` [filename]"
echo
echo "Where the input file is filename.p7b"
exit
fi

MYPEMFILE=$1

# Set variables
export INPUTFILE OUTPUTFILE MYPEMFILE
INPUTFILE=${MYPEMFILE}.p7b
OUTPUTFILE=${MYPEMFILE}.pem

if [ ! -f ${INPUTFILE} ]
then
echo "Cannot locate file : ${INPUTFILE}"
exit
fi

if [ -f ${OUTPUTFILE} ]
then
echo "Remove or renanme file : ${OUTPUTFILE}"
echo "and try again"
exit
fi

# Check file contents
echo "CONTENTS -->"
openssl pkcs7 -print_certs -inform der -in ${INPUTFILE}
echo "<-- END CONTENTS"

# Confirm conversion
echo "Continue? <retrun or contrl-c to exit>"
read ans

# Convert to PEM
openssl pkcs7 -print_certs -inform der -in ${INPUTFILE} | grep -v subject | grep -v
issuer | grep . > ${OUTPUTFILE}

if [ -f ${OUTPUTFILE} ]
then
echo "Created file ${OUTPUTFILE}"
ls -l ${OUTPUTFILE}
else
echo "File does not exist : ${OUTPUTFILE}"
fi
# EOF
```

4.4 Import all PEMS script

File : import_all_pems.sh

```
#! /bin/sh
#
# Use the script and cer file from the openssl to create a JKS store file:
#
#For example:
#
# ./import_all_pems.sh all.pem $NCHOME/platform/linux2x86/jre64_1.8.0/jre/bin/keytool
netcool all.jks
#
# Which will create a JKS file called all.jks with password netcool
# using the multiple certificate file all.cer
#
# Check arguments
#
if [ $# -ne 4 ]
then
echo "Usage: `basename $0` [PEM_FILE] [KEYTOOL] [PASSWORD] [KEYSTORE]"
exit
fi
# Set key variables
export PEM_FILE KEYTOOL PASSWORD KEYSTORE
export FILENAME COUNT ALIAS
PEM_FILE=$1
KEYTOOL=$2
PASSWORD=$3
KEYSTORE=$4
# Check input exists
if [ ! -f $PEM_FILE ]
then
echo "File does not exist $PEM_FILE" ]
exit
fi

# Check KEYTOOL exists
if [ ! -x $KEYTOOL ]
then
echo "Cannot run $KEYTOOL"
ls -l $KEYTOOL
exit
fi
```

```
# Count the number of certs in the PEM file
CERTS=$(grep 'END CERTIFICATE' $PEM_FILE| wc -l)
echo "CERTS=$CERTS"
# For every certificate in the PEM file,
# extract it and import into the JKS keystore
for COUNT in $(seq 0 $(($CERTS - 1)))
do
  FILENAME=`basename ${PEM_FILE}`
  ALIAS="${FILENAME}_$COUNT"
  echo "Creating $ALIAS"
  cat $PEM_FILE |
    awk "n==$COUNT {print}; /END CERTIFICATE/ {n++}" |
    $KEYTOOL -noprompt -import -trustcacerts \
            -alias $ALIAS -keystore $KEYSTORE -storepass $PASSWORD
done

if [ -f $KEYSTORE ]
then
echo "keytool -list -keystore $KEYSTORE -storepass $PASSWORD"
keytool -list -keystore $KEYSTORE -storepass $PASSWORD
else
echo " ERROR: Store file was not created : $KEYSTORE"
fi

# EOF
```