

NEX Troubleshoot, Manage & Monitor ECP/EDX

Version: 1.1

Date: 04.12.2024

<u>1</u>	<u>OVERVIEW.....</u>	<u>3</u>
<u>2</u>	<u>CORE CONCEPT: CERTIFICATES</u>	<u>4</u>
<u>3</u>	<u>CORE CONCEPT: MESSAGE PATH (MP).....</u>	<u>7</u>
<u>4</u>	<u>"SOMETHING IS WRONG WITH ECP"</u>	<u>13</u>
<u>5</u>	<u>"SOMETHING IS WRONG WITH EDX"</u>	<u>19</u>
<u>6</u>	<u>FIX CERTIFICATES</u>	<u>22</u>
<u>7</u>	<u>FIX QUEUES</u>	<u>25</u>
<u>8</u>	<u>FIX DATABASE.....</u>	<u>28</u>
<u>9</u>	<u>RESET ECP/EDX FROM SCRATCH</u>	<u>29</u>
<u>10</u>	<u>TOOLS</u>	<u>30</u>
<u>11</u>	<u>MONITORING</u>	<u>32</u>

1 Overview

This document aims to

- explain core functionality of ECP in order to understand troubleshooting
- explain stepwise troubleshooting, starting from high level and going to low level
- offer some tools to help manage and troubleshoot
- explain monitoring

There are some important difference in the operation of ECP 4.12/EDX 1.13 and the newer version of ECP 4.14/EDX 1.14. This document will whenever necessary point out what to do for the v4.12 and below, and for v4.14 and above. Until all users have upgraded to 4.14 or above (will happen in 2025) this document will be a little less readable because of this added complexity.

1.1 Some terminology

Short form	Explanation
BA	Business Application, usually connected to EDX-Toolbox
CD	Component Directory – stores vital information about 'components', which is both Endpoints and Brokers.
BR	ECP Central Broker, necessary to transmit messages from one EP to another EP
ECP	Most often this will refer to ECP-endpoint, but it could also be used to refer to the platform or other parts of the platform.
EDX	EDX-Toolbox, an application that sits between BA and EP.
EP	ECP-Endpoint, we consider EDX (EDX-Toolbox) as an add-on on top of EP.
IBR	ECP Internal Broker, an ActiveMQ-broker which runs within the EP
MP	Message Path, used to route message to an EP
MT	Message Type, import/mandatory header-information for ECP-messages
SC	EDX Service Catalogue, an important concept for addressing (think of DNS vs IP)
TSO	Transmission System Operator (Statnett, SvK, Energinet, Fingrid)

2 Core Concept: Certificates

When something goes wrong, it can be related to certificates. If you suspect that to be the case it might be worthwhile to get an overview of how certificates are used, and therefore how they might fail/interfere.

2.1 Which certificates are created and their usage

When an ECP-endpoint (EP), also known as a Component, is registered in Component Directory (CD) it makes 3 certificates. They have all a private part kept in the database of the EP and a public part shared with the CD. These certificates are:

- Authentication (AUTH): Used to authenticate the EP when it creates HTTPS-connection to the CD and when it creates AMQPS-connection to the ECP Central Broker (BR).
- Encryption (ENC): Used to encrypt the message content
- Signing (SIGN): Used to make a signature of the message headers

When an BR, also a Component, is registered in CD it makes 1 certificate:

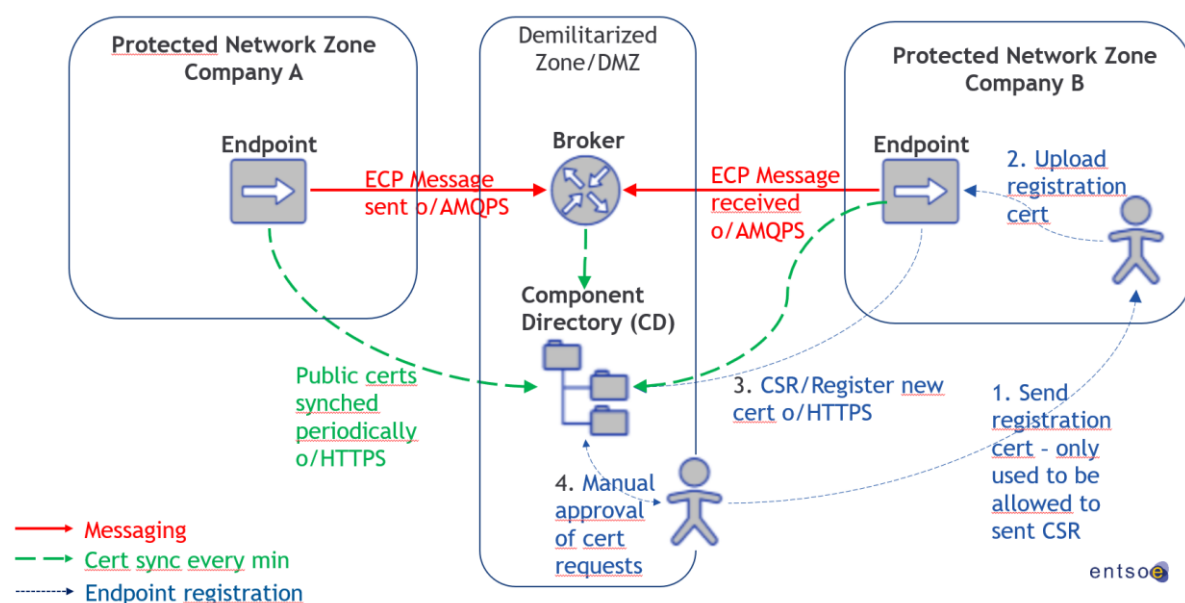
- Authentication (AUTH): Used in the same way as the EP to connect to the CD. It is also used in the AMQPS-connection, but only for EPs to trust it – not for EPs to authenticate the broker.

When a CD, also a Component, is created, it too makes an AUTH-certificate. It is used in the HTTPS-connection to all other Components, but only for trust.

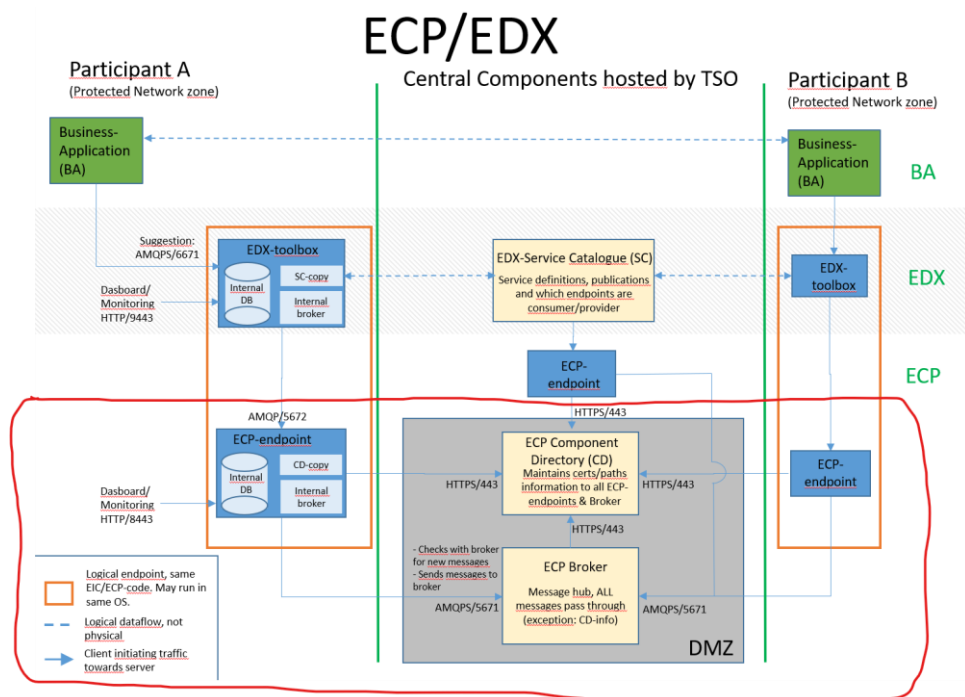
The CD keeps track of all the public parts of these certificates and let every Component in the ECP-network know about them. If a certificate is renewed, all Components will know within minutes.

2.2 Certificate creation, distribution and usage in messaging

Below you see 3 processes: The blue process is "certificate creation"; how to register a new EP with 3 certs. The green process is showing how certificates are distributed (called 'synchronization' in ECP) over HTTPS (using AUTH cert). The red process is how messages are sent from A to B (using AUTH-cert for AMQPS, and ENC+SIGN for A-to-B messaging).



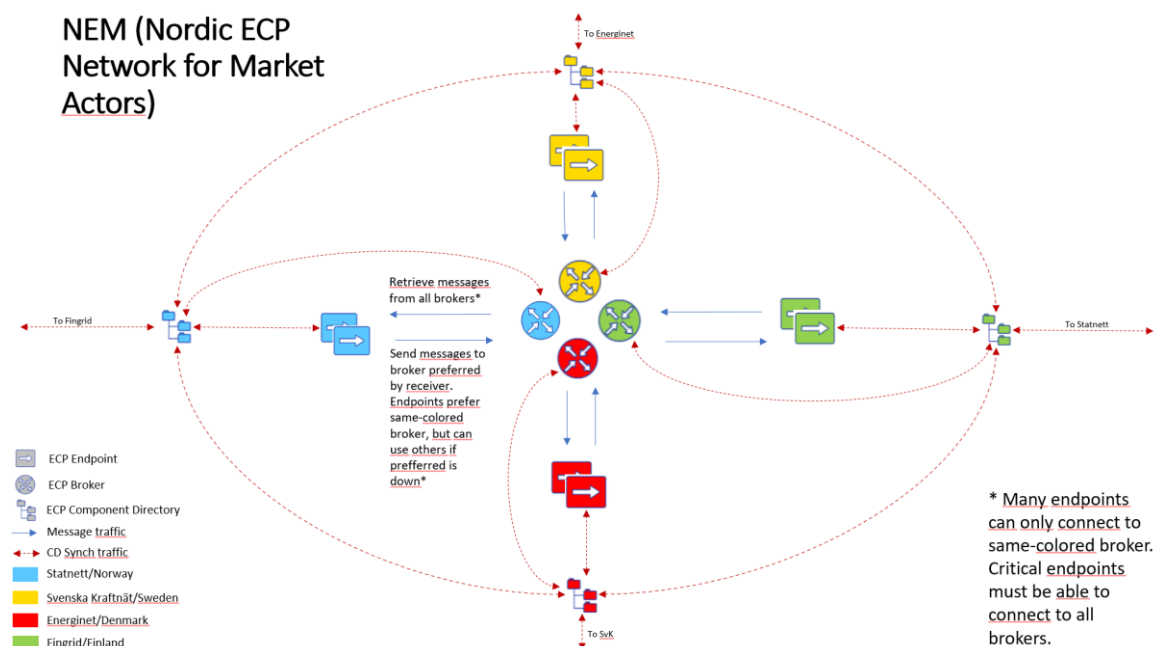
The drawing above is similar to the drawing in the NEX Installation Guide (shown below) but focus only on part of it (shown with red ring).



Most problems related to certificates come down to AUTH-certificates not being renewed (expired) or not distributed.

2.3 NEM – Nordic ECP Network for Market – multiple CD/BR

In NEM we have multiple CDs and BRs, which makes things a little bit more complicated. One CD is responsible for the renewal of one set of EPs (Statnett is responsible for Norwegian EP, SvK for Swedish EP, etc). These CDs synchronize all Component-information with each other every minute. And each endpoint can connect to all BRs if they want. This drawing gives an overview:

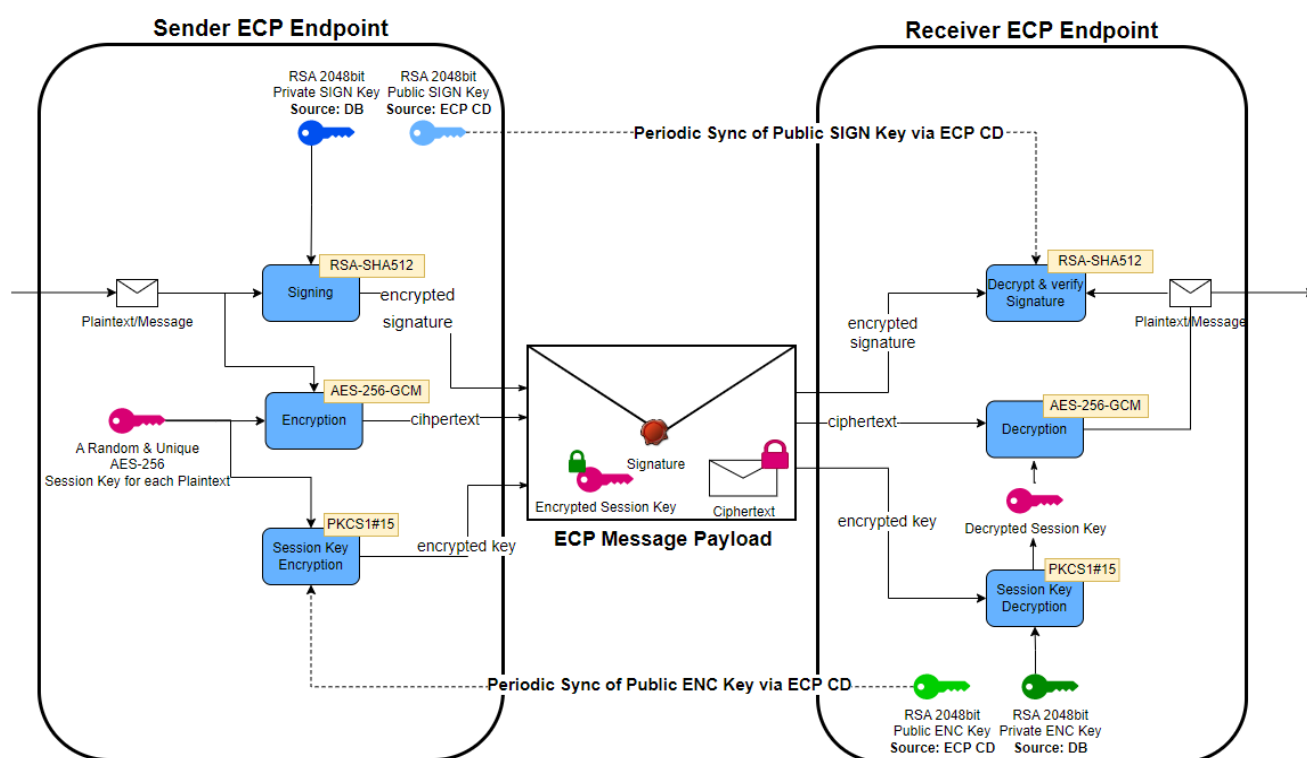


Pay close attention to the arrows. The red arrows show the CD-synchronization traffic, which is only about exchanging certificate information and other type of meta data. The blue arrows show the actual message exchange, but to avoid clutter not all possible combinations from endpoint to brokers are shown. Also note that the CDs will synch information about all endpoints to all other CDs.

2.4 End-to-end Message encryption

Having covered authentication certificate usage, we now focus on message encryption and signing. With the above explanation it's possible to see how your endpoint could be fully connected to a CD and BR, but still have the wrong certificates for a remote EP. This will cause a problem when trying to decrypt a message from that remote EP.

It's not necessary to understand the drawing below fully, but it is a useful reference to show how the certificates are being used. In some special cases it can be necessary to understand this to determine which certificate is missing. The whole scheme is based upon RSA + AES.



- You can disregard the pink key for the time being – it is created for every message (=session) and does not concern us with regards to the SIGN/ENC-certificates.
- The blue keys are the private and public SIGN certificates and shows where they are used. They are used to convince the receiver of the sender's identity.
- The green keys are the private and public ENC certificates. They are used to protect the content from being visible from any other than the receiver. Even the sender cannot decrypt the message.
- Note lock-symbol on the pink session key in the envelope in the middle. It shows how the ENC key is being used to encrypt the pink key, and the pink key is in turn used to encrypt the actual content. The reason for this apparent unnecessary complication is performance.

3 Core Concept: Message Path (MP)

3.1 Definition

A Message Path (MP) is used for an EP to tell how it can be reached by other EPs; a routing mechanism in other words. A formal definition is something like this:

An MP is applied to a certain EP to tell:

- Which Sender/Remote EP is allowed to use the MP to reach the EP
- Which Message Types (MT) is allowed to be sent to the EP
- Which BR to use in order to reach the EP
- For which timespan this path is valid

The screenshot shows a typical MP where every EP is allowed to use all kinds of MT to reach this particular EP (code is not shown here) using the BR 50V000000000119G.

Path Detail

Status	Active		
Senders	All		
Message Type *	*		
Path *	<input type="radio"/> Direct <input checked="" type="radio"/> Indirect	50V000000000119G	
Valid from *	23.11.2018 10:00	to	

[← Back](#)
[Edit](#)
[Delete path](#)

What you must understand about MP is this:

- The MPs ONLY tells you how to reach a specific EP
- In your own EP:
 - You specify NO instruction on how to send to others
 - You specify ONLY how your own EP will be reached
- The MPs you create will be synchronized with the CD every minute
- Consequently, your EP will receive all MPs for all other EP every minute
- Your EP will lookup other EP's MP to determine how to send to them

3.2 Multiple MP

You can create more than one MP. The best reason for doing so is failover. See this example which is considered a standard setup for an endpoint connected to Statnett in NEM-TEST.

Paths

+ New Path

Import Paths

Business

Acknowledgement

Path

☐ Direct
 ☐ Indirect

Status

☐ Active
 ☐ Invalid

Broker

Valid on

Message Type

Search

Senders	Message Type	Path	Valid From	Valid To	Status
All	*	50V000000000112U	08.06.2023 16:00		Active
All	*	46V000000000015S	04.12.2023 11:00		Active

10

25

50

100

In this situation, if a remote EP sends to this EP, it will use the first MP in the list. But if BR 50V000000000112U is down/not connected, then the remote EP will try the next MP which uses another BR (46V...15S).

One could create quite complex and quite many MPs if one wanted to. That is not recommended. All EP should follow the guidelines from NEX (stated in the NEX Installation Guide).

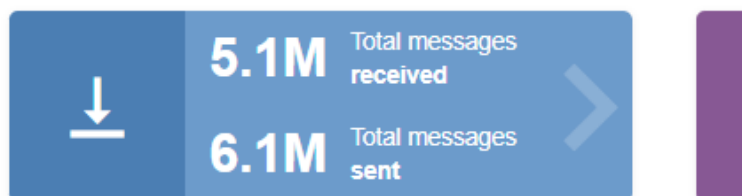
3.3 Broker connectivity

The consequence of MP system described earlier, and the fact that we have multiple BR (and CD) in NEM, is that EP-A **might** send a message to EP-B through BR-B, but when EP-B responds a sends a message back to EP-A it **might** go through BR-A! The traffic thus can pass through two different brokers on it's way back and forth.

This will happen when

- Messaging between EPs listed in two different CD (= different TSO/BR). Study the drawing of NEM in chapter 2.3 and imagine how to EPs registered in two different TSO will communicate.
- Failover occurs

Your EP will keep track of all the BRs it has sent/received messages through at some point. You will find this overview by clicking in the ECP GUI Dashboard:



Message statistics

Component code	Connected	Sum message in	Last operation in	Sum messages out	Last operation out
50V000000000119G	true	6.9M	16.03.2024 17:04	6.3M	16.03.2024 17:04
46V000000000020Z	true	118	30.08.2023 06:20	2.8M	14.03.2024 10:47
45V000000000054X	true	0		99.0k	16.03.2024 17:00
44V000000000009G	true	0		112.0k	16.03.2024 17:00

If the Connected-status is true, that means that the EP will be able to **download (receive/in)** messages to that broker. **However!** The status can be true, but still not possible to **upload (send/out)** message from that same broker because sometimes the EP loses 1 out of the necessary 2 connections! You can check this by issuing a standard command "netstat -an" (Linux) or "netstat -a -n" (Windows) and then search for connections to the BR which almost always are using the port 5671 (AMQPS):

```
[redacted]$ sudo netstat -anp | grep 5671 | sort -n -k5
tcp6      0      0 [redacted] 31.240:44120 52.233.244.188:5671 ESTABLISHED 7620/java
tcp6      0      0 [redacted] 31.240:49088 52.233.244.188:5671 ESTABLISHED 7620/java
tcp6      0      0 [redacted] 31.240:50754 192.121.1.148:5671 ESTABLISHED 7620/java
tcp6      0      0 [redacted] 31.240:35834 195.204.145.170:5671 ESTABLISHED 7620/java
tcp6      0  90    [redacted] 31.240:59708 195.204.145.170:5671 ESTABLISHED 7620/java
tcp6      0      0 [redacted] 31.240:35998 195.234.135.44:5671 ESTABLISHED 7620/java
tcp6      0      0 [redacted] 31.240:36006 195.234.135.44:5671 ESTABLISHED 7620/java
```

By studying the two screenshots we see that we have one missing connection to 192.121.148. To answer the question, which BR is behind IP, one must look in the Components List:

Dashboard

Messages

Components

Settings

Components

Paths

Component Code

Organization

Network

Component Directory

Search

Component Code	Type	Organization	Networks	Version	Component Directory
46V000000000020Z	Broker	Svenska Kraftnät AB	internet, DefaultNetwork	4.10.1.1632	46V000000000019K
45V000000000054X	Broker	Energinet	internet, DefaultNetwork	4.11.0.1775	45V000000000053Z
50V000000000119G	Broker	Statnett SF	DefaultNetwork, internet	4.12.0.1868	50V000000000118I
44V000000000009G	Broker	Fingrid Oyj	internet, DefaultNetwork	4.10.1.1632	44V000000000006M
46V000000000019K	Component Directory	Svenska Kraftnät	DefaultNetwork	4.10.1.1632	46V000000000019K
50V000000000118I	Component Directory	Statnett SF	DefaultNetwork	4.12.0.1868	50V000000000118I

Then open for example the component details for BR 46V....20Z and review the information there:

Dashboard Messages Components Settings								
Component Detail								
Component Type	Broker	Organization	Svenska Kraftnät AB					
Broker code	46V000000000020Z	Contact Person	ECP Management					
Comp. Directory	46V000000000019K	Contact E-mail	ecp@svk.se					
URL and Network	amqp://ecp4.svk.se:5671	Contact Phone	0046103509101					
Created	18.08.2020 09:01:11							
Last Modification	14.03.2024 13:35:05							
Implementation Version	4.10.1.1632							
Broker Restrictions								
Allowed Message Types	*	Allowed Components	*					
Certificates								
Type	Status	Active Since	Valid To	Preferred				
Authentication	Active	14.03.2024 12:34	14.03.2025 13:34	Yes	<input type="button" value=">"/>			
10 25 50 100								

Then check if that matches with IP:

```

tcp6 0 0 [REDACTED]:31.240:36006 195.234.135.44:5671 ESTABLISHED 7620/java
[REDACTED]$ ping ecp4.svk.se
PING ecp4.svk.se (192.121.1.148) 56(84) bytes of data:
^C
--- ecp4.svk.se ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1054ms
[REDACTED]$
  
```

At this point we know that our EP may not be able to upload to that 46V...20Z. But this will quite often be fixed as soon as we sent a message. If not, then restart the ECP-endpoint.



3.4 Sending to a remote EP

To understand which BR is used when sending to a remote EP open the Components List and find the EP you are sending to (ex 50V...120V):

DashboardMessagesComponentsSettings

Components

ComponentsPaths

Component CodeOrganizationNetworkComponent Directory

SearchReset Filter

Component Code	Type	Organization	Networks	Version	Component Directory
50V000000000118I	Component Directory	Statnett SF	DefaultNetwork	4.12.0.1868	50V000000000118I
50V000000000119G	Broker	Statnett SF	DefaultNetwork, Internet	4.12.0.1868	50V000000000118I
50V000000000120V	Endpoint	Statnett SF	DefaultNetwork	4.12.0.1870	50V000000000118I
50V000000000121T	Endpoint	Statnett SF	DefaultNetwork	4.12.0.1870	50V000000000118I
50V000000000118Y	Endpoint	Statnett SF	DefaultNetwork	4.12.0.1870	50V000000000118I
50V000000000227D	Endpoint	Statnett SF (Kons...	DefaultNetwork	4.12.0.1870	50V000000000118I
50V0000000002477	Endpoint	Statnett (NMMS)	DefaultNetwork	4.12.0.1870	50V000000000118I
50V0000000002493	Endpoint	Statnett SF (HT)	DefaultNetwork	4.12.0.1870	50V000000000118I
50V000000000251G	Endpoint	Statnett SF	DefaultNetwork	4.12.0.1870	50V000000000118I

102550100

Then check the details and the Message Path listed:

Component Detail

Component Type	Endpoint	Organization	Statnett SF
Endpoint code	50V000000000120V	Contact Person	IDTD SerCat
Comp. Directory	50V000000000118I	Contact E-mail	ecp@statnett.no
Created	12.04.2018 13:35:13	Contact Phone	004723903000
Last Modification	30.01.2024 09:52:17		
Implementation Version	4.12.0.1870		

Paths

PathDirectIndirectBrokerValid onSearch

Senders	Message Type	Path	Valid From	Valid To
All	*	50V000000000119G	23.11.2018 10:00	
44V000000000024K, 44V000000000029A, 44V000000000036D, 44V0000000000672, 45V0000000000055V, 45V0000000000064U, 45V00000000001136, 45V00000000001144, 45V0000000000129S, 45V0000000000134Z, 45V00000000001403, 45V0000000000143Y, 46V0000000000021X, 46V0000000000022V, 46V0000000000023T, 50V0000000000120V, 50V0000000000121T, 50V0000000000188Y, 50V00000000002388, 50V0000000000243F, 50V00000000002477, 50V00000000002493, 50V0000000000251G, 50V0000000000252E	*	46V000000000020Z	01.02.2022 12:00	

102550100

Certificates

Type	Status	Active Since	Valid To	Preferred
Authentication	Expired	28.02.2023 09:33	28.02.2024 10:33	No
Encryption	Expired	28.02.2023 09:33	28.02.2024 10:33	No
Signing	Expired	28.02.2023 09:33	28.02.2024 10:33	No
Authentication	Active	30.01.2024 08:50	29.01.2025 09:50	Yes
Encryption	Active	30.01.2024 08:50	29.01.2025 09:50	Yes
Signing	Active	30.01.2024 08:50	29.01.2025 09:50	Yes

102550100

The first MP will be used, if the EP lists this BR as "Connected = true" (see screenshot in previous sub chapter). If not, the next MP will be used. Always look out for MPs that are not yet valid.

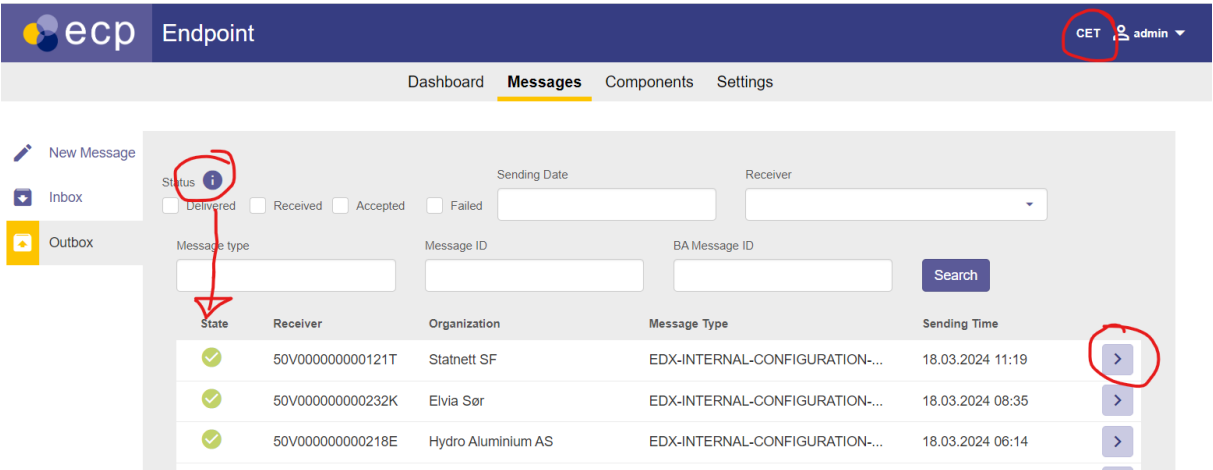
Armed with this knowledge it will now be easier to troubleshoot ECP-issues.

4 "Something is wrong with ECP"

Quite often one can hear the statement mentioned or variations thereof. The first step is to clarify if that statement is correct, because often it is not the source of the problem.

4.1 Investigation of Message Status

Go to ECP-GUI and check Message Status in the Outbox:



Click the (i) button to get information about the various states. The main take-away there is that if you have RECEIVED (green check-mark shown above) or DELIVERED state (yellow-brown), then you know that the remote EP did receive your message. If status is ACCEPTED (also green) then you **cannot** know whether the problem is in your EP or somewhere else. Thus you must continue investigation in the next sub-chapters. Look closely at the timestamps and the timezone-setting (upper right corner) which should be CET. You could also study the details of the particular message:

Message ba9edb4a-b471-442b-92f4-0af93682a0f8

Receiver	50V000000000121T
Message id	ba9edb4a-b471-442b-92f4-0af93682a0f8
BA Message id	
Sender application	
Message type	EDX-INTERNAL-CONFIGURATION-MESSAGE
Sending time	18.03.2024 11:19:33
Delivery time	18.03.2024 11:19:33
Sent file extension	
State	Received

[Refresh](#)

Time	Event	Component	Comp. description	Message
18.03.2024 11:19:33	Accepted	50V000000000120V		Message has been accepted into ECP.
18.03.2024 11:19:33	Delivered	50V000000000121T		Delivery acknowledgement.
18.03.2024 11:19:33	Received	50V000000000121T		Receive acknowledgement.

This shows the timing of the various states, so it can shown if there has been a delay from the time the message was sent (Accepted – first row) and when the remote EP responded with Delivered/Received ACK (second/third row).

Investigating the Inbox part is not so useful, since you cannot see messages that you haven't received. However, by looking at the timestamps or searching for certain Message Type, you can perhaps deduce that there is problems with receiving/downloading message. In that case read the next sub-chapters.

4.2 Investigation of CD connectivity

Go to ECP-GUI-Settings and click on "Check Connectivity" under "Component Directory". If status is OK, then you know that your endpoint is connected and synchronized with the CD (it synchronizes every minute). That means that your endpoint will know all the **public certificates** and the **message paths** of all the other endpoints in the network (see previous Core Concept chapters for more explanation of these terms).

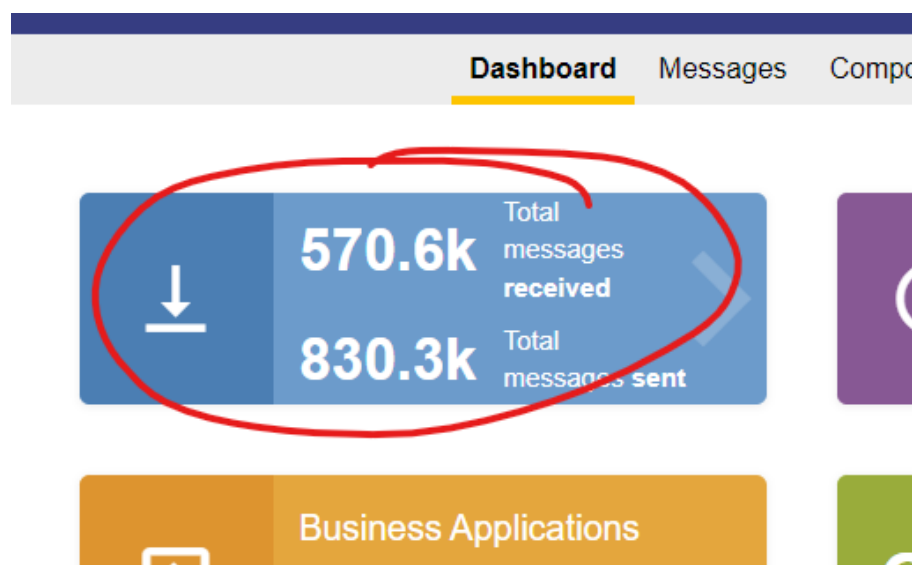
If status is not OK, then you should check the file `ecp.log` to see error messages related to traffic towards the CD. Even if you **do not** have connection to the CD, the message traffic may still flow well, because you have a local copy of the CD. If connection to the CD has been away for many hours/days, then your CD-copy may be outdated and hold expired certificates. Up until v4.12 of ECP new certificates (in the whole ECP-network) is being preferred as soon as they are created, thus a problem could occur when certificates are not synchronized (copied) throughout the network through the CD. Later versions will improve this, to make CD-downtime an almost non-existent problem.

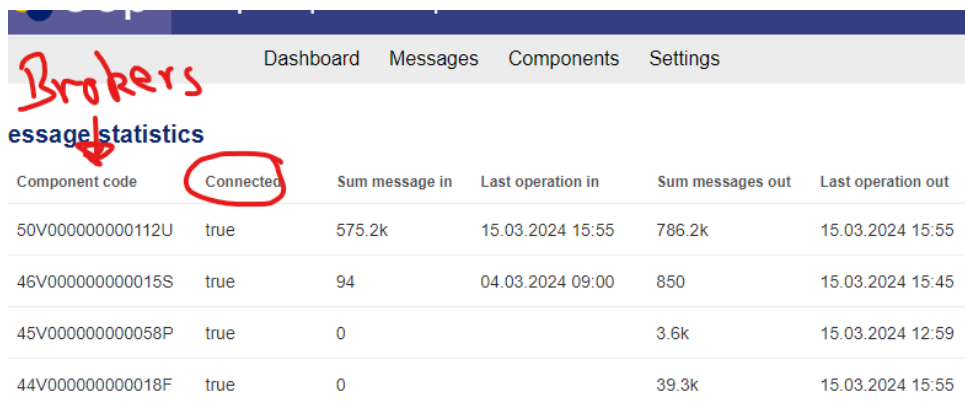
If you do not have connection to the CD it falls into two categories:

- Network problems (firewall? - see NEX Installation Guide)
- Certificate problems (see chapter 6)

4.3 Investigation of Message Connectivity

Go to ECP-GUI-Dashboard->Click on the Blue tile:





Dashboard Messages Components Settings					
Component code	Connected	Sum message in	Last operation in	Sum messages out	Last operation out
50V000000000112U	true	575.2k	15.03.2024 15:55	786.2k	15.03.2024 15:55
46V000000000015S	true	94	04.03.2024 09:00	850	15.03.2024 15:45
45V0000000000058P	true	0		3.6k	15.03.2024 12:59
44V0000000000018F	true	0		39.3k	15.03.2024 15:55

The list shows (one or more) ECP Central Brokers your endpoint is connected to. If all have Connected = true, your endpoint is most likely connected. To examine this thoroughly and avoid any doubt, please read chapter 3.3.

Go to ECP-GUI-Setting and click on "Check Connectivity" under "Messaging Connectivity". Choose the endpoint you want to send to and specify TEST as Message Type. Then send. If you get "OK", then you know that ECP to ECP traffic is working well.

If you get "NOT CONNECTED" you can check the file ecp.log to search for reasons, but you can also try to send to other ECP-endpoints. This type of testing will not show up in ECP GUI of the receiving endpoint, so it is totally fine to use for testing. If some endpoints respond, then you know that your ECP-endpoint works fine, but some other endpoints fail.

You can also check in ECP-GUI-Components -> Choose details on the receiving component. Here you can see if that endpoints has defined Message Paths. The message path tells which broker your endpoint must connect to, to send to it an other endpoint. Then go to ECP-GUI-Dashboard -> Click on blue tile. Check status on the various brokers you're connected to and make sure you can connect to the broker mentioned in the Message Path.

If you have identified that your EP is not connected to the necessary BR (where you expect to send/receive a message), then it's basically the same as for CD connectivity problems, but it could also be problems related to

- Network problems (firewall? - see NEX Installation Guide).
- Certificate problems (see chapter 6)

4.4 Investigate Internal Broker (applicable for ECP 4.12 or lower)

Each EP has its own "Internal Broker" (IBR). This where the EP stores the messages which are in transit, either outgoing (sending) or incoming (receiving). The number one problem with the IBR is that it can run full. And this can be caused by just a few messages. Check ecp.log to see if there are error messages related to something being 100% full or "flooded". Another way to check is to examine the activemq-*.xml governing the IBR in your EP. These files are found among your configuration files for the EP. There are many of those, most likely all the same setup – and you can search for "<storeUsage>" tag. This tag will show a limit to how much space the IBR can take. Usually in a percent of the available space on the disk. The IBR itself can be found in a similar place like this:

```
[redacted@redacted data]# pwd
/var/lib/ecp-endpoint/internalBroker/data
[redacted@redacted data]# ls -l
total 4424
-rw-r-----. 1 ecp-endpoint ecp-endpoint 33554432 Mar 16 19:30 db-69101.log
-rw-r-----. 1 ecp-endpoint ecp-endpoint 2220032 Mar 16 19:30 db.data
-rw-r-----. 1 ecp-endpoint ecp-endpoint 279064 Mar 16 19:30 db.redo
-rw-r-----. 1 ecp-endpoint ecp-endpoint 8 Mar 14 10:47 lock
[redacted@redacted data]#
```

If you think some queue is full, then go to chapter 7.

4.5 Investigate Artemis Broker (applicable for ECP 4.14 and higher)

With ECP 4.14 and EDX 1.15, the ActiveMQ Artemis message broker is introduced, and it runs in a separate process. We will still refer to it as an "internal broker", but in the sense that it is internal for the ECP/EDX-endpoint. This broker is the next generation, and it seems to be more stable and robust. It will not run full as quickly as the old broker, due to how messages are stored. Currently we have no specific advice on this broker, except what has been given in the NEX Installation Guide for the setup.

4.6 Investigate OS Resources

4.6.1 High CPU/IO-wait

EP is very CPU & I/O intensive. 10 msg/sec is rule of thumb for maximum limit given the hardware recommendation given in NEX Installation Guide. Approaching this limit you may experience problems with throughput, delays and queues filling up. No endpoint in NEM has this level of traffic on average, but if something goes wrong or is halted for a time, then you can get a huge "burst" (ketchup bottle effect).

You can inspect the traffic pattern in several ways:

- Check ECP GUI and count messages pr minute in/out
- Check ecp.log, identify patterns in the log – count such patterns (with some grep-tool)
- Use ekit.jar to analyze ECP/EDX logs (see chapter 11.4.1)
- Setup a Grafana dashboard (see chapter 11.2)

4.6.2 High CPU

Another reason why CPU is high, while I/O might not be high, could be a background job in EP. You can list the background jobs if you go to ECP GUI Dashboard and click on the "Jobs" tile or go to Settings-page and locate the Jobs-section. It looks like this:

Background Jobs

Job	Status	Start	End	Duration	Failure
Synchronization	Completed	18.03.2024 09:58:15.001	18.03.2024 09:58:18.315	3314 ms	
Message Deleting	Completed	18.03.2024 09:14:45.493	18.03.2024 09:14:45.670	177 ms	
Registration Status Checker	Completed	18.03.2024 09:59:00.003	18.03.2024 09:59:00.131	128 ms	
Statistics Synchronization	Completed	18.03.2024 09:58:30.001	18.03.2024 09:58:30.105	104 ms	
Message Path Synchronization	Completed	18.03.2024 09:58:45.001	18.03.2024 09:58:45.048	46 ms	
Cluster Status Manager	Completed	18.03.2024 09:59:13.001	18.03.2024 09:59:13.003	1 ms	
Certificate Renewal	Completed	18.03.2024 09:50:00.001	18.03.2024 09:50:00.002	1 ms	
Expired Certificates Cleaning	Completed	18.03.2024 09:10:00.002	18.03.2024 09:10:00.004	1 ms	
Routes Management	Completed	18.03.2024 09:59:00.001	18.03.2024 09:59:00.002	1 ms	
Configuration Reload	Completed	18.03.2024 09:59:10.000	18.03.2024 09:59:10.001	0 ms	
Message Expiration	Completed	18.03.2024 09:13:49.232	18.03.2024 09:13:49.232	0 ms	
Database Compression	Disabled				
Directory Cache Reload	Waiting for run				
Database Message Compression	Disabled				

Here we show all jobs sorted by "Duration". The jobs that is most likely to take lot of resources are those related to the database, marked with red: "Message Deleting" and "Message Expiration". Database Compression should be disabled if you've followed the NEX Installation Guide. The database keeps track of all messages sent/received and is what you see in Outbox/Inbox in the GUI. If the database has grown very large, or if you have changed some setting regarding message expiration or message retention (see Unicorn ECP Administration Guide), then these jobs have been known to run "forever". But, the list above shows only the completed jobs, not the currently running jobs. If a job has been running for a long time but not completed yet, thus consuming a lot of CPU, it will not show in the table above!

The default behaviour of "Message Deleting" and "Message Expiration" is to run once every hour. If the timestamp show for last start/end is older than 1 hour, then you should think about reducing the database. In that case go to chapter 8.

4.6.3 Disk utilization

If disk is filling up you should check the size of the database by checking the size of the db-folder "seg0":

```

[redacted] pwd
/var/lib/ecp-endpoint/db/seg0
[redacted] ls -lh | head
total 2.8G
-rw-r-----. 1 ecp-endpoint 1.8G Mar 18 10:58 c1570.dat
-rw-r-----. 1 ecp-endpoint 221M Mar 18 10:58 c15d1.dat
-rw-r-----. 1 ecp-endpoint 115M Mar 18 10:58 c1601.dat
-rw-r-----. 1 ecp-endpoint 102M Mar 18 10:58 c1581.dat
-rw-r-----. 1 ecp-endpoint 90M Mar 18 10:58 c640.dat
-rw-r-----. 1 ecp-endpoint 87M Mar 18 10:58 c1591.dat
-rw-r-----. 1 ecp-endpoint 68M Mar 18 10:58 c15f1.dat
-rw-r-----. 1 ecp-endpoint 55M Mar 18 10:58 c15b1.dat
-rw-r-----. 1 ecp-endpoint 47M Mar 18 10:58 c15e1.dat
[redacted] du -h
2.8G
[redacted]

```

This database contains 224K message-entries (not the message payload, only metadata), which gives approximately 12KB pr msg which is good rule of thumb. If you have reason to believe that the disk usage is much higher than 12KB pr msg, database compression could be performed – see chapter 8.1.

5 "Something is wrong with EDX"

This is not often heard, since EDX is not so widely known. But, if everything checks out fine with ECP, but still message flow is interrupted, then it is time to look at EDX. The problems you may encounter here are usually one of these:

1. EDX does not connect to the ECP Internal Broker (see last drawing in chapter 2.2 where EDX-Toolbox is connected to ECP Endpoint over AMQP).
2. EDX does not have a EDX Service Catalogue (SC) copy
3. EDX does not have the correct EDX Service Catalogue (SC) copy (there could be multiple)
4. EDX has received an SC with errors in it (the TSO is responsible)
5. EDX Internal Broker (IBR) is full (this is another IBR than for ECP)
6. EDX does not route message to the correct BA
7. EDX or BA has terminated connection, perhaps because some expired cert

Of these, only 5 is something we would expect to happen without any reconfiguring/restarting, although all of them can happen "out of the blue" given the right circumstances.

5.1 EDX to ECP IBR connectivity

Go back to NEX Installation Guide and check both `ecp.properties` and `edx.properties` carefully to see that `edx.properties` for `ecpBroker` matches the settings you have in `ecp.properties` for `internalBroker`. The EDX is a client to the ECP IBR. Be especially careful with `ecp.broker.url` in `edx.properties` – it alone determine whether EDX will try to use AMQPS or AMQP.

5.2 EDX SC

An EDX might be registered in multiple SC. In the following we assume that BA are addressing correctly (see <https://ediel.org/wp-content/uploads/2023/10/NEX-Addressing-Guidelines.pdf>), so the problem is reduced to having the correct EDX SC. This shows an EDX registered in 4 SC (EDX GUI – Settings):

Settings

Toolbox Code	50V-SN-DK---ATT
Toolbox Name	Statnett (DK)
Party Name	PARTY
Publish Subscribe Version	0

+ Register Toolbox to SC

Service Catalogue	Network Configuration Version	Last Synchronization time
50V000000000113S	940	14.03.2024 05:20:28
46V000000000016Q	306	11.03.2024 15:01:55
45V0000000000059N	397	11.03.2024 14:29:44
44V0000000000023M	271	15.03.2024 19:48:43

10 25 50 100

If no SC is listed, then it must be fixed. If the EDX is new, is very likely that the responsible TSO has not added the EDX to the EDX SC – you could try to notify the TSO. Another option is that the code you've specified for the property `edx.serviceCatalogue.code` in `edx.property` is incorrect. Check NEX Installation Guide again.

Once you made a change or believe the TSO has done a change, restart EDX. This will force it to request a SC-copy again upon startup. You can see the message being sent and possibly also the response in ECP-GUI Outbox/Inbox.

If you do receive a response in ECP-Inbox, you can further track the parsing of this response in the file `edx.log`. If something goes wrong with the parsing of the SC, you can see it there.

At this point you should have at least one default SC listed in Settings.

You may need more than the default SC, because your EDX access EDX-services and toolboxes outside of the default SC. If you know/suspect that you are missing some SC in your list, then you must contact that TSO which is responsible – there is no way to add it manually as of now, but the TSO can trigger their SC to be sent to your EDX.

The TSO can make errors in the SC and they will be propagated to every EDX within minutes. One mistake could be to remove your EP/EDX-code from the SC. Other mistakes could be to change some addressing information within the SC. Notify the TSO in this case.

5.3 Investigate Internal Broker (applicable for ECP 4.12 or below)

Note: For ECP 4.14 and above, a new setup of a separate ActiveMQ Artemis broker has been introduced. This instance is shared between ECP and EDX. See comment about this in the previous chapter (chapter 4.5).

Each EDX has its own "Internal Broker" (IBR). This is where the EDX stores the messages which are in transit, either outgoing (sending) or incoming (receiving). The number one problem with the IBR is that it can run full. And this can be caused by just a few messages. Check `edx.log` to see if there are error messages related to something being 100% full or "flooded". Another way to check is to examine the **jms-context-nonha.xml** governing the IBR in your EDX. This file is found in `<EDX-config-folder/jms/` or `<EDX-config-folder/jms/secured`, depending on the `edx.properties` setting of `internalBroker.useAuthentication=false/true`. In this file search for "`<storeUsage>`" tag, it will determine how much disk IBR is allowed to take, usually as a percent of the available space on the disk. You can change it to a specific limit to have more control over the space set aside for the IBR. Here is an example from Statnett-configuration:

```
<systemUsage>
  <systemUsage>
    <memoryUsage>
      <!-- in-memory buffer for messages -->
      <memoryUsage limit="250 mb" />
    </memoryUsage>
    <storeUsage>
      <!-- storage buffer for messages -->
      <storeUsage limit="1000 mb"/>
    </storeUsage>
    <tempUsage>
      <tempUsage limit="200 mb" />
    </tempUsage>
  </systemUsage>
</systemUsage>
```

The IBR itself can be found in a similar place like this:

```
[redacted]# pwd
/var/lib/edx-toolbox/edx-activemq-data/data
[redacted]# ls -lh
total 4.5M
-rw-r-----. 1 edx-toolbox edx-toolbox 32M Mar 16 20:37 db-8587.log
-rw-r-----. 1 edx-toolbox edx-toolbox 1.1M Mar 16 20:37 db.data
-rw-r-----. 1 edx-toolbox edx-toolbox 301K Mar 16 20:37 db.redo
-rw-r-----. 1 edx-toolbox edx-toolbox 8 Mar 14 10:47 lock
[redacted]#
```

If you think some queue is full, then go to chapter 7.

5.4 Investigate OS Resources

The problems for EDX is almost identical to that of EP (ECP) – see chapter 4.5, but apply that chapter to EDX logs/jobs/database.

5.5 EDX Routing

The `edx.yml` contains information on how to route to correct BA (or rather to correct "integration channel"). This configuration is described in NEX Installation Guide in some detail. However, if the SC changes **and** you have routing based on "Service", then it is possible that the routing fails and the message is sent to "receiveProcessDefaultRoute" (usually never to be seen again!). This is very likely scenario for most EDX setups. If you route based on Message Type (which is not so much recommended) then it is more likely that the routing fails because the BA that sends the message has changed the Message Type.

To understand how the routing is being executed you must study the `edx.log` carefully. Check also for "RoutesConfig" log lines during startup of EDX, because they tell which routes are actually active. Sometime an error in `edx.yml` can lead to that no routes are read/used/applied.

5.6 EDX BA Connectivity

It is recommended to have AMQPS or otherwise TLS-communication between BA and EDX. This can lead to break in communication because of expired cert. There is of course a whole range of possible issues here, but that is outside the scope of this guide.

6 Fix certificates

6.1 Comparison

What you can do first is to compare certificate information in your EP with the CD-copy you have. The Core concept chapter about certificates (chapter 2) tries to explain which certificates are created, what they are for and how they are distributed. One basic task is to verify as best as we can that this process has succeeded. We can do that by looking at the ECP GUI Settings page and find the "Certificates"-section:

Certificates

Automatic Renewal

Enabled Disabled Renew Manually Import Certificate

Local Network

Certificate Type	Active Since	Valid To	Preferred	
Global CA	13.06.2019 16:26	17.02.2026 09:49	Yes	>
Integrated CA	22.02.2023 09:54	17.02.2026 09:49	Yes	>
Authentication !	25.01.2024 09:51	24.01.2025 09:50	Yes	>
Encryption	25.01.2024 09:51	24.01.2025 09:50	Yes	>
Signing	25.01.2024 09:51	24.01.2025 09:50	Yes	>
Integrated CA	13.06.2019 16:26	13.06.2024 10:05	No	>
Authentication !	23.02.2023 12:57	23.02.2024 12:56	No	>
Encryption	23.02.2023 12:57	23.02.2024 12:56	No	>
Signing	23.02.2023 12:57	23.02.2024 12:56	No	>
Authentication !	19.02.2023 09:51	19.02.2024 09:50	No	>

10 25 50 100 < Previous 1 2 3 Next >

The thing we focus on first is to find the valid certificates. No expired certificates are useful. Then we identify the Authentication certificates, because this is the certificate used to establish network-connection (HTTPS or AMQPS). One of them is preferred, but there could be some that are not preferred, but still valid – so they can be used.

Next, we check our copy of the CD. Go to ECP GUI Components and find **your own** endpoint code there and click the details-button:

ecp Endpoint CET admin

Dashboard Messages **Components** Settings

Components

Components Paths

Component Code Organization Network Component Directory


50V-SN-DK----ATT

Search Reset Filter

Component Code	Type	Organization	Networks	Version	Component Directory	
50V-SN-DK----ATT	Endpoint	Statnett (DK)	DefaultNetwork	4.12.0.1868	50V000000000111W	>

10 25 50 100

Certificates

Type	Status	Active Since	Valid To	Preferred	
Authentication 	Active	25.01.2024 08:50	24.01.2025 09:50	Yes	>
Encryption	Active	25.01.2024 08:50	24.01.2025 09:50	Yes	>
Signing	Active	25.01.2024 08:50	24.01.2025 09:50	Yes	>

10 25 50 100

We can verify that the CD has the same Authentication certificate as our own EP, by matching the timestamp. We can now reasonably expect that all other components in the ECP-network will also know our EP's public certificates. **At the very least you can be absolutely convinced that the CD has this information and that the HTTPS-connection to the CD will not be blocked because of certificate issues!** If you don't find a match, then it's still possible that your CD has not been able to synchronize the information back to your ECP. But it's also possible that the CD do not have the certificate, and then communication will never work. You are then forced to do option 5 in next sub-chapter.

If you experience problems with the AMQPS-connection to the BR, and you know the CD has the correct information about your EP and the CD is synchronizing every minute, then two possibilities are left:

- The BR has not received cert-info about your EP from the CD
- The BR has not renewed/published its own certificates to the CD

Check the certificates you have for the BR you're communicating with (you may have to understand MP to know which BR to investigate – see chapter 3) and make sure they are valid. If the certificate is not about to expire (renewal happens 30 days before expiry in NEM), then most likely the connection problem is not due to certificate mismatch.

6.2 Options

If you find in `ecp.log` problems indicating a "certificate problem" (SSL-handshake, "expired"-messages, failed-message due to signature etc) you can try to go through this list. The goal is to make sure the CD has received the certificates from our EP.

Each option can potentially solve the issue. BUT! Option 4 and 5 will renew your own certificate. Option 5 also requires assistance from TSO. Options 4 and 5 should not be executed unless you are pretty sure that it is your own ECP-endpoint that is to blame. It could very well be the other ECP-endpoint or the central ECP-broker that has outdated information from the CD. In those cases, no option will help, and 4 and 5 should be avoided.

1. Restart ECP-endpoint. Test message flow.
2. Go to ECP-GUI-Settings.
 - a. Push configuration. If it fails, it's useful to divide into two categories:
 - No matching certificates between your EP and the CD. Read chapter 6.1.
 - Something else: Firewall/DPI, Routing/DNS/Hardware, CD-downtime
 - b. Restart ECP-endpoint
 - c. Wait 3 minutes and test message flow
3. Control that your firewall is not interfering (deep packet inspection) with the SSL-traffic between ECP-endpoint and/or CD/Broker. Check by running something similar to this command (`openssl s_client -connect ecp4.statnett.no:5671 -showcerts`). The firewall must

not touch/change anything in SSL-traffic. You can get hold of openssl for Windows by downloading Git for Windows (<https://git-scm.com/download/win>), openssl is part of the "Git Bash" shell.

4. Go to ECP-GUI-Settings. Renew Manually. Wait 3 minutes and test message flow. This option should usually not be helpful, because either you're are connected to the CD and have valid certificates or not. If it's the latter, then it will not be possible to renew manually – you can only renew if you have existing valid certificates.
5. Re-register: Go to ECP-GUI-Settings and press button for "Initiate Registration". You **may** need to ask the TSO (see NEX Installation Guide for email address) for a new registration keystore and you **must** ask the TSO (again see NEX Installation Guide for email address) to approve a new registration after you've completed your part.

7 Fix queues

7.1 Purge/Remove messages

To purge/remove messages seems a little drastic, but most serious business procedures must expect an ACK from the BA itself before it considers a message delivered. Therefore it can be an option to purge messages from time to time, to get things running. There are several ways to do it. The first 3 out of 5 options are manual, the 2 last are for automated use.

7.1.1 The Hawtio option (valid for ECP 4.12 and below)

This can be done manually following these steps:

1. Install Hawtio (see chapter 10.1). Wait 30 sec.
2. Connect to your ECP or EDX GUI but change the URL path to `"/hawtio/"`.
3. The top-menu should show "ActiveMQ" – click on that choice.
4. All queues will be listed, find queues where queue size > 0
5. Consider purging the queue (click on queue – choose "Delete"-submenu – Purge) or delete specific messages.

7.1.2 Delete IBR option (valid for ECP 4.12 and below)

You can delete the folder named "internalBroker" (EP) or "edx-activemq-data" (EDX) and restart EP/EDX. In this case you will lose all messages in transit, but the IBR will be built anew upon restart. This is the "nuke" option, simple and quick.

7.1.3 The Artemis option (valid for ECP 4.14 and above)

This can be done manually following these steps:

1. Open <https://<artemis-host>:8161/> - and log in. Check firewall settings if you cannot connect. The user/pass-details are specified in the artemis-users.properties.
2. Choose Queues menu and choose "operations" button on the queue that you want to purge.
3. Choose "Remove All Message" operation and execute.

7.1.4 Ekit option I (valid for all ECP versions)

If you want to purge a specific queue, you can consider using Ekit:

1. Install Ekit (see chapter 10.2)
2. Read documentation on QC tool, by running this command:
 - `java -jar ekit.jar QC`
3. Suggested command for purge of queue QUEUENAME is
 - `java -jar ekit.jar QC -c -q QUEUENAME /etc/ecp-endpoint`

The command could be run scheduled (Linux crontab or Windows scheduling). The example shows how to connect with EP (ECP) where Ekit runs on the same host. But you could also connect to EDX IBR or connect remotely, or via AMQPS.

7.1.5 Ekit option II (valid for ECP 4.14 and above)

Ekit option I using QC has two downsides:

- It cannot be run from a remote location, but must run on the same host as ECP
- It will not always detect all queues that is being used (if `-q` option is omitted)

With introduction of Artemis and the HTTP-API available, it was possible to build a BrokerShell (BRS) to query, list and manage the queues. The BRS can be run in "scriptmode" to trigger a series of commands to perform for example removal of messages. Here is one example:

1. Install Ekit (see chapter 10.2)
2. Read documentation on BRS tool, by running this command:
 - `java -jar ekit.jar BRS`
3. Suggested command for purge of queue QUEUENAME is
 - `java -jar ekit.jar BRS -c -c 'eoff;z1;sDLQ|ExpiryQueue;v;r-2;sreply;c0;v;age10;r-2;s;c-1;v;age1440;r-2' https://endpoint:password@localhost:8161`

The command above will perform 3 separate query removal jobs:

- Find all queues with more than 1 message and queue-name contains 'DLQ' or 'ExpiryQueue', then remove all those messages.
- Find all queues where queue-name contains 'reply' and has 0 consumers and are 10 minutes or older, then remove all those messages.
- Find all messages that are 1440 minutes or older, then remove all those messages

To understand the script-commands, use the in-built help function of the BRS. If run without -c option it will provide a prompt for user interaction.

7.2 Advanced queue protection

The Hawtio option above can be used to remove specific messages, but it's a completely manual process. It's also possible to move a message from one queue to the other and back again. But these procedures does not scale well. Ekit offers more features when it comes to so-called "Queue protection". Install Ekit (see chapter 10.2). See next sub-chapters for use cases:

7.2.1 Remove old messages

If a queue is being consumed, but the client consuming is unstable, you can set Ekit to consume any message that is older than 1 hour (3600 seconds). This operation will run until aborted, because of the -f option.

```
>java -jar ekit.jar QC-t3600 -q QUEUENAME amqp://endpoint:password@localhost:5672
```

7.2.2 Burst protection

This case has only been tested on ecp.endpoint.download queue, where all messages are coming into EP. If the queue is being flooded with messages from one or more senders, it will delay other messages. You can decide to remove messages from such troublesome senders. Example: if a sender (ECP-endpoint) has produced more than 50 msg in the download-queue and those messages are more than 10 seconds old:

```
>java -jar ekit.jar QP -f -q0 -s50 -t10 ecp.endpoint.download amqp://endpoint:password@localhost:5672
```

Think of this as DoS-protection.

7.2.3 Burst protection and restore

This case has only been tested on ecp.endpoint.download queue. Instead of purging the messages, it's possible to store them to disk and then restore them back to the queue later on. This is of course described in the documentation of the QP tool if you run

```
>java -jar ekit.jar QP
```

One example of such a configuration is where we start to remove all message older than 20 sec when there is more than 10 msg on the queue, but restore them later – that is, as soon as the burst is over (fewer messages than 10 on the queue or no messages older than 20 sec).

```
>java -jar ekit.jar QP -e -f -m msg-buffer -o qp.json -q10 -s0 -t20 ecp.endpoint.download  
amqp://endpoint:password@localhost:5672
```

The main idea here is to let through new messages, prioritizing "real-time" traffic over old/queued messages.

8 Fix database

The main fix for the database is to reduce its size. There could of course be more exotic problems which could be resolved with the right SQL, but that is outside the scope of this document for now. You need to log in to the database following the steps in chapter 10.3. Then choose the chapter below you want to execute:

8.1 Compress (no loss, but could be slow)

A compression would remove empty space in the database, but all records will be retained. A compression should usually not be necessary, but if you have reduced the message retention time to reduce the database, compression is needed. The database will never reduce in size by itself!

Compression of ECP-tables:

- `call SYSCS_UTIL.SYSCS_COMPRESS_TABLE('ECP', 'COMPONENT_DIRECTORY', 1);`
- `call SYSCS_UTIL.SYSCS_COMPRESS_TABLE('ECP', 'MESSAGE', 1);`

Compression of EDX-table:

- `call SYSCS_UTIL.SYSCS_COMPRESS_TABLE('EDX', 'TOOLBOX_MESSAGE', 1);`

Warning: This operation can take many minutes if the database has gigabyte size. Be patient.

8.2 Truncate (loss, but very quick)

For EDX there is a set of SQL statements that can be run to quickly truncate the large tables. This will delete all the message records (only metadata/logdata, no payload is lost). Run in order:

Remove foreign keys:

- `alter table EDX.TOOLBOX_MESSAGE drop constraint FK_TOOLBOX_MESSAGE_TOOLBOX_MSG;`
- `alter table EDX.PULL_MESSAGE drop constraint FK_PULL_MESSAGE_TOOLBOX_MSG;`
- `alter table EDX.TOOLBOX_MESSAGE_LOG drop constraint FK_TOOLBOX_MSG_LOG_TOOLBOX_MSG;`
- `alter table EDX.INFLIGHT_EXTERNAL_PROCESSING drop constraint FK_INFL_EXT_PROC_TOOLBOX_MSG;`

Truncate tables:

- `truncate table EDX.TOOLBOX_MESSAGE_LOG`
- `truncate table EDX.PULL_MESSAGE;`
- `truncate table EDX.INFLIGHT_EXTERNAL_PROCESSING;`
- `truncate table EDX.TOOLBOX_MESSAGE;`

Add foreign keys back:

- `alter table EDX.TOOLBOX_MESSAGE add constraint FK_TOOLBOX_MESSAGE_TOOLBOX_MSG foreign key (PARENT_MESSAGE_ID) references EDX.TOOLBOX_MESSAGE(ID);`
- `alter table EDX.PULL_MESSAGE add constraint FK_PULL_MESSAGE_TOOLBOX_MSG foreign key (TOOLBOX_MESSAGE_FK) references EDX.TOOLBOX_MESSAGE(ID);`
- `alter table EDX.TOOLBOX_MESSAGE_LOG add constraint FK_TOOLBOX_MSG_LOG_TOOLBOX_MSG foreign key (MESSAGE_ID) references EDX.TOOLBOX_MESSAGE(ID);`
- `alter table EDX.INFLIGHT_EXTERNAL_PROCESSING add constraint FK_INFL_EXT_PROC_TOOLBOX_MSG foreign key (MESSAGE_ID) references EDX.TOOLBOX_MESSAGE(ID);`

These commands are very quick, but it is advised to test it first on a test-endpoint. These commands have been tested on EDX 1.13, but will most likely work on older versions of EDX as well.

For EP (ECP) no truncate commands has been explored so far.

9 Reset ECP/EDX from scratch

Apart from re-installing, there is a way to reset the ECP/EDX more or less from scratch. Assuming configuration is otherwise correct, but the system has run into a state which is hard to figure out, then a reset can be useful.

9.1 Hard reset

It consists of two things:

- Delete IBR of both ECP/EDX (see chapter 7.1.2)
- Delete database of both ECP/EDX

Upon restart all these resources will be built anew, but content is of course lost. Deleting IBR will delete messages in transit, that is, messages in the queues. So it can cause a loss of messages. Usually business process can tolerate that.

To delete the databases then delete "db"-folders under ECP/EDX-installation. However, to recover from this can take time:

For EP:

- No longer be connected to the CD, all private certificate information is lost
- Re-registration must be performed (see NEX Installation Guide) and the **TSO must approve manually**.

For EDX:

- No copy of the necessary Service Catalogues, without them EDX will not work at all.
- When EDX starts it will request the default SC (see edx.properties 'edx.serviceCatalogue.code'). Until the SC-response has been received by EDX, all messages that is processed will Fail.
- If your EDX depend on other SCs than the default SC then you must ask the TSOs to trigger the SC to be sent to you. **This is a manual process**.

As you can see, this is not really a good option, only a last resort.

9.2 Soft reset

This may not remove all problems in the database, but otherwise it may often solve problems.

- Delete IBR of both ECP/EDX (see chapter 7.1.2)
- Truncate EDX (see chapter 8.2)
- Compress ECP (see chapter 8.1)

10 Tools

10.1 How to install Hawtio

1. Add/change the property "spring.jmx.enabled=true" to ecp.properties and edx.properties. This will make it possible for Hawtio to see the queues. Restart ECP or EDX if change was necessary.
2. Download hawtio: <https://repo1.maven.org/maven2/io/hawt/hawtio-web/1.5.11/hawtio-web-1.5.11.war>
3. Rename the file to "hawtio.war" and place the file in the webapps-folder of ECP and/or EDX – it will automatically install.
4. After you've done using Hawtio you should remove the war-file – Hawtio is a liability security-wise. You should also consider reversing the jmx-settings introduced in 1.

10.2 How to install Ekit

EKit is short for "ECCoSP Toolkit" and is a software developed by Statnett (Morten Simonsen) to solve issues related to ECP and EDX. The software has of course no guarantees, but it is running in Statnett. The software can be downloaded from ediel.org: <https://ediell.org/nordic-ecp-edx-group-nex/nex-statnett/>

It requires at JRE 17 to run (same as for ECP 4.12/EDX 1.13). It runs like this:

```
>java -jar ekit.jar
```

which will return this – which explains briefly the 9 tools it contains:

```
ekit v1.10.5 is short for ECCoSP Kit, built by Statnett (Morten Simonsen) for ECP 4.8-4.14 (EDX v1.9-1.14)
The kit offers the following. Main interface/API/source used is specified in brackets []
```

QC (QueueCleaner)	v3.0.3	[AMQP]	- clean queues based on filters
QP (QueueProtector)	v1.1.9	[AMQP]	- protects download-queue from bursts
QM (QueueMonitor)	v1.0.1	[HTTP]	- monitor queues in Artemis-broker (4.14+)
BRS (BrokerShell)	v1.4.15	[HTTP]	- manage/monitor Artemis-brokers (4.14+)
CC (ConnectivityCheck)	v1.0.3	[HTTP]	- check connectivity to ECP-endpoints
EMD (ECPMsgDelay)	v1.9.2	[LOGS]	- calculate delays for messages transmitted over the ECP-network
EPA (ECPPerfAnalyzer)	v2.0.1	[LOGS]	- calculate throughput of the ECP/EDX-endpoint (4.14+)
EML (ECPMsgLogger)	v1.5.5	[LOGS]	- messages to/from ECP pretty-printed to log
GM (GapMonitor)	v1.2.3	[EML]	- monitor gaps in the ECP-message traffic - very specific usecase

```
To get more help on usage, run 'java -jar ekit.jar <QC|QP|QM|BRS|CC|EMD|EPA|EML|GM>'
```

To maintain a stable ECP/EDX-endpoint there are two fundamental aspects which every endpoint should handle:

- 1) Queues must not grow for a long time or too large
 - a) For ECP 4.12 and below, use QC for queue cleaning and some monitoring
 - b) For ECP 4.14 and above, use BRS for queue cleaning, and QM for queue monitoring
- 2) Connectivity to the remote endpoints you communicate with, must always be monitored. Use CC for this. The other tools are useful in their own right, but not strictly necessary.

The tool usage is further explained in chapter 7 and 11.4 (EMD + EPA Tool).

10.3 How to connect to Database (Derby)

If you have followed the NEX Installation Guide you are running the default installation database which is a Java-based database named Derby. You can connect to this database following these steps:

1. Download Derby client: https://db.apache.org/derby/derby_downloads.html (download 10.15.2.0 if you're running ECP 4.10 or 10.16.1.1 if you're running ECP 4.12+)
2. Unzip into a folder, ex /opt/derby
3. Change folder to your ECP og EDX, ex /var/lib/ecp-endpoint (your database you should now be found on "db"-folder in the folder you are located)
4. Stop ECP/EDX - you cannot edit the DB from more than one user at the time (or make a copy of the DB-folder and work on that)
5. Run '/opt/derby/db-derby-10.15.2.0-bin/bin/ij' or '/opt/derby/db-derby-10.16.1.1-bin/bin/ij'
 - a. ij> connect 'jdbc:derby:db';
 - b. The "db" marked in red above is the name of the folder, so this works if you want to copy the database to another folder and work on it while the ECP endpoint is running. While inside the ij-tool you can do all SQL and some other commands:
 - c. ij> help;
 - d. ij> show tables;
 - e. ij>DO-SOMETHING-USEFUL
 - f. ij> exit;
6. After exit and if you've edited the database (by INSERT/UPDATE/DELETE) you should check that file permission/ownership is the same as before – and if not, change back so that the ECP/EDX process can read/change the database.

11 Monitoring

There are many possibilities to monitor your endpoint. NEX Installation Guide (Appendix) mentions the Connectivity Check which is a useful and basic monitoring of your endpoint's connectivity. If you skipped that part of the installation guide, please go back and re-read. With v4.12 of ECP it's now also possible to get information from various URLs which can be fed into monitoring software. These URLs are:

EP (ECP):

- /ECP_MODULE/actuator/prometheus **(NB! Can be very slow before ECP v4.15)**
- /ECP_MODULE/actuator/info
- /ECP_MODULE/actuator/health
- /ECP_MODULE/actuator/readiness

EDX:

- /actuator/prometheus
- /actuator/info
- /actuator/health
- /actuator/readiness

ARTEMIS (ECP 4.14+)

- /metrics/ **(returns prometheus metrics)**

Of these, we will focus on the prometheus option, since the other contain much less information and are meant for container-environment. **NB!** If your ECP-database contains hundreds of thousands of entries, the prometheus-URL on ECP will be very slow, because it scans the largest table 8 times. In this situation it is not recommended to run the prometheus-URL or at the very least, do scraping rarely. We have measured up to 60s response time on this URL on large DB.

11.1 Prometheus

Prometheus-interface returns several hundred metrics. These metrics can be categorized into three groups:

- OS resources
- Application Server (Java/Tomcat/Spring) resources
- ECP/EDX resources

To further reduce the task at hand we focus mostly on the ECP/EDX metrics. The reason for this is that OS-resources is a generic problem that is most likely (or should be) solved/monitored already. The Application Server is a relatively stable part of the stack and should not have much trouble unless in high traffic situation – and that will be detected in the ECP/EDX metrics.

Focusing on the ECP/EDX resources we suggest that it is almost exclusively the queue metrics that matter. As an over-simplification we can say that if you have no problems with the queues, then ECP/EDX will work fine in everyday operation. Many of the issues in this document is more about special/initial problems. If you follow the advice from NEX Installation Guide to have Connectivity Checks and then add Prometheus metrics to monitor the queues, you have come a long way to know the state of ECP/EDX. ECP/EDX Administration Guides has a chapter explaining these metrics.

11.1.1 The most useful metrics (for ECP 4.12 and below)

If queue-size on any queue is above 0 for a long time it could be a problem


```
ecp_internal_broker_queue_queue_size  
edx_internal_broker_queue_queue_size
```

With dequeue count you can know the rate of consumption from the queue, effectively knowing the processing rate.

```
ecp_internal_broker_queue_dequeue_count  
edx_internal_broker_queue_dequeue_count
```

Capture if InternalBroker (IBR) is running full – if 100 the IBR will block, nothing will work.

```
ecp_internal_broker_store_percent_usage  
edx_internal_broker_store_percent_usage
```

11.1.2 The most useful metrics (for ECP 4.14 and above)

Access /metrics/ URL of the 8161 port of the Artemis broker to retrieve the following metrics

```
artemis_message_count  
artemis_messages_acknowledged  
artemis_disk_store_usage
```

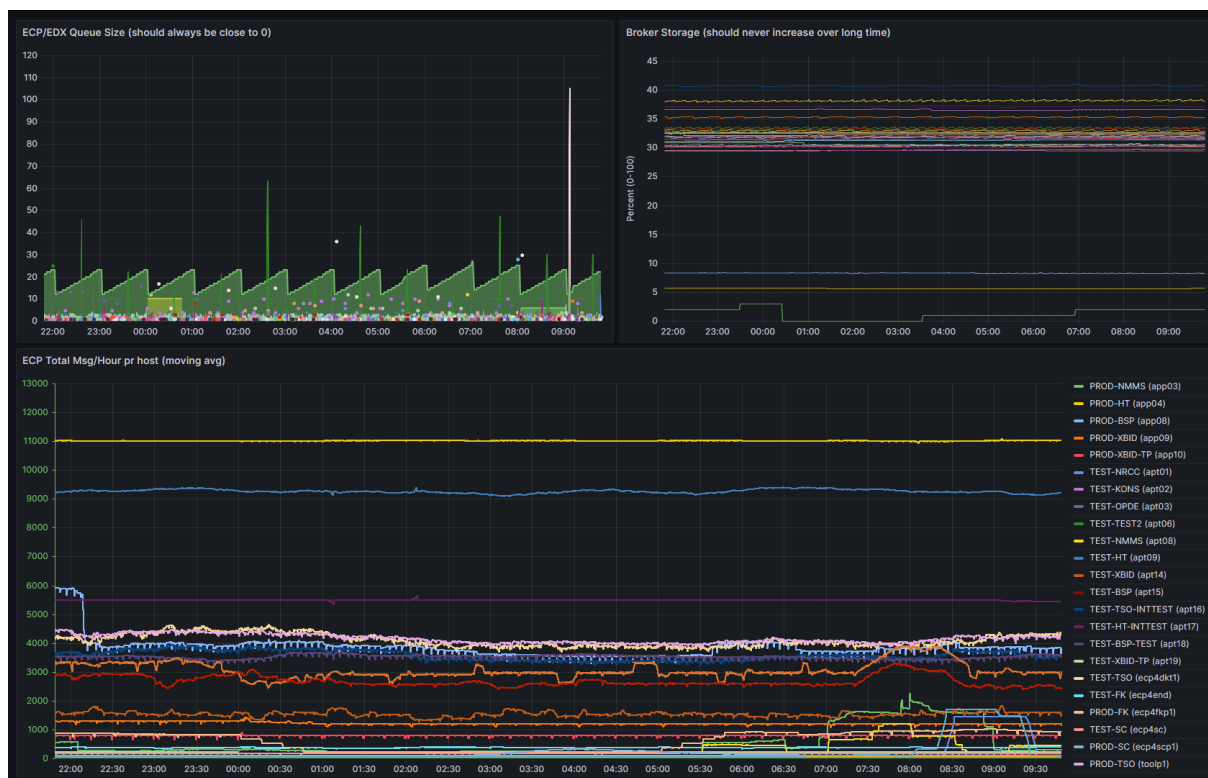
With these three metrics you'll measure

- queue-size (should always stay close to 0)
- dequeue-count (should be steadily increasing whenever new messages arrive, indicates processing rate)
- disk-usage (should not increase over a long time, must never fill the disk)

11.2 Grafana

11.2.1 Queue dashboard

Given the metrics mentioned in previous chapter, it is possible to make a useful dashboard in Grafana. After some years of experience, we've come to a simple solution that is not hard to implement in Grafana. The dashboard can cover all ECP/EDX-endpoints at once, it is not necessary to create separate dashboard for each endpoint:



Panel top-left:

Shows the queues that have at least one message. The "metrics browser" query goes like this:

```
QueueSize > 0 (for ECP 4.12 or below)
artemis_message_count > 0 (for ECP 4.14 or above)
```

Panel top-right:

```
artemis_disk_store_usage*100
BrokerStorePercentUsage
```

Panel bottom (only shown for ECP 4.14+, but similar can be built for ECP 4.12 and below using the metrics explained in previous chapter):

```
sum(delta(artemis_messages_acknowledged{queue=~".*(download|upload).*"}) [60m])
> 0) by (instance)
```

11.2.2 Essential queues

If you want to have more control of which queues are doing what, here is a quick explanation of what the various queues are used for. This information can be used if you want to build more specialized panels for certain queues.

Sending queues, starting from BA, through EDX, to ECP and to BR:

- edx.endpoint.outbox* (From BA)
- edx.internal.sendProcess-delivery-send (To ECP)

- ecp.endpoint.outbox (From EDX)
- ecp.endpoint.upload.* (To BR)

Receiving queues, starting from BR, through ECP to EDX and to BA:

- ecp.endpoint.download (From BR)
- ecp.endpoint.inbox (MSG to EDX)
- ecp.endpoint.send.event (ACK to EDX)
- edx.endpoint.inbox* (To BA)

Communication to/from BA could be other ways than through these queues, it is merely a hint.

11.3 QueueMonitoring (for ECP 4.14 and above)

Ekit (see chapter 10.2) offers QueueMonitoring (QM) as an alternative or supplement to the Grafana dashboards. This tool uses the exact same metrics as described above, but adds some logic to the monitoring, to provide data for precise alarms (not too many, not too few).

```
[redacted] ecp-endpoint]# java -jar ekit.jar QM
QueueMonitor (QM) v1.0.1 will monitor queues in an Artemis broker (ECP 4.14+). The tool will output
an ERROR message if one or more queues are not dequeued for 1 minute or if the queue-size is not decreasing
within the last 10 minutes.

The tool will run in a forever loop, checking the queues every minute and print the status to a log-file every minute.
To allow the tool to be triggered by cron-job, the tool will only run if it detects that no other instance is running
A running instance is detected by checking that the modified timestamp of the log-file is younger than 70 sec.

Usage : java -jar ekit.jar QM <OPTIONS> <LOG-FILE> <ECP-URL>

The arguments:
OPTIONS
-i <IGNORE-Q>      : List of strings, commaseparated. If a queue-name contains any of these strings, the queue will be ignored
-d <DELAY-Q>       : List of strings, commaseparated. If a queue-name contains any of these strings the monitoring will be
                    : delayed by 60 min. Useful for queues that are purged by BRs or similar tools, to prevent false alarms
-r day|week|month  : Rotate log-file every day, week or month. Requires -o option
-m<noLogFiles>     : Allow maximum number of log-files. Default is 10. Requires -r option
-n<noDays>         : Do not rotate data younger than noDays. Default is 0. If set higher, the output log will
                    : always contain the last noDays of data, even if the log-file is rotated. This is useful
                    : for analysis of the last days of data (GapMonitor). Requires -r option
-z                : Compress rotated logs. Requires -r option
-x                : Will print the internal state of QM to stdout, useful for debugging
MANDATORY ARGUMENTS
<LOG-FILE>        : File to log output to - new data will be appended to the file every minute
<ECP-URL>         : Specify url on this form: http[s]://user:pass@host:port

Example 1: Run QueueMonitor with default options
java -jar ekit.jar QM https://localhost:8161

Example 2: Run QueueMonitor with ignore and delay options. Ignore DLQ/Expiry queues and delay monitoring of reply-queues
java -jar ekit.jar QM -i DLQ,Expiry -c reply https://localhost:8161

Example 3: Run QueueMonitor with log-rotation every day and max 10 log-files. Compress rotated logs
java -jar ekit.jar QM -r day -m10 -z https://localhost:8161
```

The output of this could be like this:

```
{
  "timestamp": "2024-11-23 12:55:28",
  "timestamp": 1732362928,
  "status": "ERROR",
  "queues": [
    {
      "name": "edx.endpoint.inbox",
      "size": 1,
      "cons": 0,
      "reason": "stopped"
    }
  ]
},
{
  "timestamp": "2024-11-23 12:56:28",
  "timestamp": 1732362988,
  "status": "ERROR",
  "queues": [
    {
      "name": "edx.endpoint.inbox",
      "size": 1,
      "cons": 0,
      "reason": "stopped"
    }
  ]
},
{
  "timestamp": "2024-11-23 12:57:28",
  "timestamp": 1732363048,
  "status": "ERROR",
  "queues": [
    {
      "name": "edx.endpoint.inbox",
      "size": 1,
      "cons": 0,
      "reason": "stopped"
    }
  ]
},
{
  "timestamp": "2024-11-23 12:58:28",
  "timestamp": 1732363108,
  "status": "ERROR",
  "queues": [
    {
      "name": "edx.endpoint.inbox",
      "size": 1,
      "cons": 0,
      "reason": "stopped"
    }
  ]
},
{
  "timestamp": "2024-11-23 12:59:28",
  "timestamp": 1732363168,
  "status": "ERROR",
  "queues": [
    {
      "name": "edx.endpoint.inbox",
      "size": 1,
      "cons": 0,
      "reason": "stopped"
    }
  ]
},
{
  "timestamp": "2024-11-23 13:00:28",
  "timestamp": 1732363228,
  "status": "ERROR",
  "queues": [
    {
      "name": "edx.endpoint.inbox",
      "size": 1,
      "cons": 0,
      "reason": "stopped"
    }
  ]
},
{
  "timestamp": "2024-11-23 13:01:28",
  "timestamp": 1732363288,
  "status": "OK"
},
{
  "timestamp": "2024-11-23 13:02:28",
  "timestamp": 1732363348,
  "status": "OK"
},
{
  "timestamp": "2024-11-23 13:03:28",
  "timestamp": 1732363408,
  "status": "OK"
},
{
  "timestamp": "2024-11-23 13:04:28",
  "timestamp": 1732363468,
  "status": "OK"
}
```

11.4 Logs

The logs contain a lot of information and some of it we can extract using Ekit (see chapter 10.2 for installation). There are three log tools:

- ECPerfAnalyzer – analyze the flow of traffic through the ECP/EDX-endpoint
- ECPMessageDelay – analyze the flow of traffic from this EP to remote EPs
- ECPMsgLogger – pretty-prints traffic to/from your ECP-endpoint (like the ECP-GUI)

11.4.1 ECPerfAnalyzer (EPA) (for ECP 4.14 and above)

It can be hard to get an overview on how your ECP/EDX-endpoint process messages, especially to understand if messages are blocked in a certain direction, in either ECP or EDX, at a certain time. EPA



is the answer to this information need, and quickly process the logs to show you the "flow" of the messages through your endpoint.

Run Ekit like this to see options on EPA:

```
>java -jar ekit.jar EPA

ECPPerfAnalyzer (EPA) v2.0.1
Usage : java -jar ekit.jar EPA [-af] <ecp-log-directory> <edx-log-directory>

Example: java -jar ekit.jar EPA /var/log/ecp-endpoint /var/log/edx-toolbox
Example: java -jar ekit.jar EPA -af /var/log/ecp-endpoint /var/log/edx-toolbox

The analyzer is compatible with ECP 4.14+ - it depends upon certain log events. Older
versions of ECP/EDX is not tested as of now. By default it only counts PLAIN/STANDARD msg.
Numbers in parenthesis denotes an incomplete count (not a full minute/hour or log-rotation).

Option: a      Also count the ACKs, not just PLAIN/STANDARD msg
Option: f      Tailing the logs
Option: l      Only process the newest logfile (the gz-files will be skipped)
```

Running EPA you will with the -f option will also provide CPU/Load information (see last part of screenshot – showing 2 minutes run):

MINUTE: 20241128-09:55	24	24	0	36	36	0		
MINUTE: 20241128-09:56	24	24	0	36	36	0		
MINUTE: 20241128-09:57	24	24	0	37	37	0		
MINUTE: 20241128-09:58	24	24	0	36	36	0		
MINUTE: 20241128-09:59	24	24	0	36	36	0		
HOURL: 20241128-09	1456	1456	0	2178	2178	0		

Time	EDX	SEND ECP	Diff	ECP	RECEIVE EDX	Diff	CPU PERCENT	LOAD
MINUTE: 20241128-10:00	24	24	0	38	38	0		
MINUTE: 20241128-10:01	24	24	0	36	36	0		
MINUTE: 20241128-10:02	24	24	0	36	36	0		
MINUTE: 20241128-10:03	24	24	0	37	37	0		
MINUTE: 20241128-10:04	24	24	0	36	36	0		
MINUTE: 20241128-10:05	24	24	0	36	36	0		
MINUTE: 20241128-10:06	24	24	0	37	37	0		
MINUTE: 20241128-10:07	24	24	0	36	36	0		
MINUTE: 20241128-10:08	24	24	0	36	36	0		
MINUTE: 20241128-10:09	20	20	0	37	32	5		
MINUTE: 20241128-10:10	28	28	0	36	41	-5		
MINUTE: 20241128-10:11	24	24	0	36	36	0		
MINUTE: 20241128-10:12	24	24	0	37	37	0		
MINUTE: 20241128-10:13	24	24	0	36	36	0		
MINUTE: 20241128-10:14	26	26	0	36	36	0		
MINUTE: 20241128-10:15	24	24	0	37	37	0		
MINUTE: 20241128-10:16	24	24	0	36	36	0		
MINUTE: 20241128-10:17	24	24	0	36	36	0		
MINUTE: 20241128-10:18	24	24	0	37	37	0		
MINUTE: 20241128-10:19	24	24	0	36	36	0		
54 MINUTE: 20241128-10:20	24	24	0	36	36	0	0	0.37
MINUTE: 20241128-10:21	24	24	0	37	37	0	8	0.51

Here you see the traffic going in each direction, through ECP and EDX

- 2 major columns SEND and RECEIVE
- 3 minor column EDX, ECP and Diff
- The Diff-column will show if EDX and ECP columns differ

You can easily see that traffic is flowing in both direction, that all traffic is going through both components. The only exception here is 5 messages that we receive in ECP (10:09) is received in EDX in the next minute (10:10), which is to be expected from time to time.

You can also see that processing speed of the entire endpoint this way – and you can go as far back in history as you have logs.

11.4.2 ECPMessageDelay (EMD)

Run Ekit like this to see options for EMD:

```
>java -jar ekit.jar EPA
```

ECPMsgDelay (EMD) v1.10.0, a tool for analyzing message delays when sending/receiving from other ECP-endpoints

Usage : java -jar ekit.jar EMD [OPTION] <ecp-log-directory>

OPTION:

```
-s          : process only 'sent messages' - this is default behaviour, these timestamps are most trustworthy
-r          : process only 'received messages', -s will override -r
-d<sec>    : The delay-limit columns will show the number based on this limit in seconds. Default is 10
-e<sec>    : The expire-limit columns will show the number based on this limit in seconds. Default is 60. Disabled if lower or equals to -d
-o <filename> : The entries that exceed the delay limit will be appended to this file
-c <filename> : Received entries that has been received before sent, indicating clock skew (only with -r)
-p <filename> : Append summary pr opposite endpoint pr hour to this file
-x <filename> : Produces monthly summary of the file specified. The file must contain the std-out or the opposite file (-p).
-y<noday>   : Only valid with -x. Specifies the number of days to include in the summary. Default is last 7 days.
-t<hours>   : Default i 1, which means previous hour will be processed. Any other numbers is meant to be used for testing purposes.
-z <filename> : Debug-file will contain information about the ECPMsgDelay processing, in case something fails.
```

Example 1: Summary of all sent messages, delay/expire-limit is 10/60 sec, storing output to stdout.json (which is used in Ex 5)

```
java -jar ekit.jar EMD /var/log/ecp-endpoint > stdout.json
```

Example 2: Summary of last hour received messages, delay/expire-limit is 10/60 sec:

```
java -jar ekit.jar EMD -r /var/log/ecp-endpoint
```

Example 3: Summary of last hour received messages + 3 more files for different purposes, delay/expire-limit is 20/60 sec:

```
java -jar ekit.jar EMD -r -d20 -c neg.json -o lim.json -p opp.json /var/log/ecp-endpoint
```

Example 4: Summary of last hour sent messages + 2 files for different purposes, delay/expire-limit is 10/60 sec:

```
java -jar ekit.jar EMD -h -o lim.json -p opp.json /var/log/ecp-endpoint
```

Example 5: Summary of summary, gives daily + monthly summary based on the standard output (see Ex 1):

```
java -jar ekit.jar EMD -x stdout.json -y7
```

Example 6: Summary of summary, gives daily + monthly summary based on the opposite output (see Ex 3 and 4):

```
java -jar ekit.jar EMD -x opp.json -y7
```

The delay analyzer is compatible with ECP 4.9-4.15 - it depends upon certain log events. Other

versions of ECP might not give any useful results. It calculates the delays of the message

transmissions in both directions, SEND and RECEIVE:

- * SEND delay is recorded when first DELIVERED or RECEIVED ACK is received and then compared with SEND tms.
- * RECEIVE delay is recorded when STANDARD message is received and compared with 'Generated' tms in the msg

The main output is a series of json-objects, one pr hour with a summary of the delays/sizes.

The output is meant to be easy to read for humans and easy to parse for Splunk, etc. The OPTIONS allow you to investigate the delays in more detail.

Since generated tms inside the message (RECEIVE case) can come from an endpoint with a clock skewed compared to this endpoint, we might get negative RECEIVE delays. Those are not counted, but can be listed using the -c option. The -o option will list all messages that exceed the limits in seconds. The -p option will give a summary of the traffic for each opposite endpoint pr hour.

Explanations of the various terms used in the output:

timestamp	Timestamp for the start of the time period of measurements
direction	Direction, snd or rcv
endpoint	Sending or receiving endpoint
measureEP	The local endpoint, the endpoint we're measuring from
oppositeEP	The opposite/remote endpoint, the endpoint we're measuring to/from
broker	The ECP broker that transferred the message
msgId	The ECP message id
cnt	Total number of messages
cntDel	Number of messages that exceed the delay limit and below expire limit
cntExp	Number of messages that exceed the expire limit
cntLim	Number of messages that exceed the delay limit
ms	Delay for the message in milliseconds

```

avgMs      Average delay for all messages
avgMsDel   Average delay for messages that exceed the delay limit and below expire limit
avgMsExp   Average delay for messages that exceed the expire limit
avgMsLim   Average delay for messages that exceed the delay limit
KB         Size of the message in kilobytes
avgKB      Average size for all messages
avgKBDel   Average size for messages that exceed the delay limit and below the expire limit
avgKBExp   Average size for messages that exceed the expire limit
avgKBLim   Average size for messages that exceed the delay limit
pct        Percent of messages compared to all messages
pctLim     Percent of delayed messages compared to all delayed messages
pctLimSelf Percent of delayed messages compared to all messages from this endpoint
troubleIndex The higher the index, the worse the problem is. It is calculated as:
              overrepresentation * pctLimSelf * cntLim * avgMsLimIndex / 100
              overrepresentation is 100 if limits are not overrepresented nor underrepresented
              pctLimSelf is 100 if all traffic to the endpoint exceed the limit
              limitCount is simply the number of messages that exceed the limit
              avgMsLimIndex is 0 if avgMsLim < delayLimit. 1 if avgMsLim == delayLimit. rising to 100 if avgMsLim >= expireLimit
    
```

This is tool measures the delay in the ECP-network. You can specify what you think is classified as "delay" and "expired". It's most useful for TSOs or those EP that communicate with many different EPs. There are too many options to explain in any detail here – test the various examples to see the output.

11.4.3 ECPMsgLogger (EML)

This tool offers a way to make a nice and clean log of the traffic going in and out of your ECP-endpoint. You can think of it like the ECP-GUI, showing messages sent and received in a list. The tool is made to run continuously and to print 1 minute of logs every minute, as soon as that minute has passed. This way you can tail on that log and see the traffic in almost real-time.

Another interesting feature is that you can keep log-history way longer than you do in your ECP-database (14 days is the default there). And the log is made to be easy to search for all the useful information about each message and works perfectly with tools like grep.

Here you'll see the output of 10 messages sent or received, divided into two screenshots because the lines of the out is very long.

```

2024-12-04T08:59:15.567Z wk=2024W49, qh=172739, dir=SEND, ack=Y, stat=OK, ms=490, rem=46V00000000000052M, bro=46V00000000000015S, loc=50V000000000001150,
2024-12-04T08:59:35.301Z wk=2024W49, qh=172739, dir=SEND, ack=Y, stat=OK, ms=313, rem=46V000000000000052M, bro=46V00000000000015S, loc=50V000000000001150,
2024-12-04T09:13:47.299Z wk=2024W49, qh=172740, dir=RECEIVE, ack=-, stat=OK, ms=131, rem=46V000000000000052M, bro=50V00000000000112U, loc=50V000000000001150,
2024-12-04T09:14:05.356Z wk=2024W49, qh=172740, dir=SEND, ack=Y, stat=OK, ms=411, rem=46V000000000000052M, bro=46V00000000000015S, loc=50V000000000001150,
2024-12-04T09:14:08.405Z wk=2024W49, qh=172740, dir=RECEIVE, ack=-, stat=OK, ms=105, rem=46V000000000000052M, bro=50V00000000000112U, loc=50V000000000001150,
2024-12-04T09:14:16.293Z wk=2024W49, qh=172740, dir=SEND, ack=Y, stat=OK, ms=371, rem=46V000000000000052M, bro=46V00000000000015S, loc=50V000000000001150,
2024-12-04T09:23:06.996Z wk=2024W49, qh=172741, dir=SEND, ack=Y, stat=OK, ms=297, rem=44V00000000000028C, bro=44V00000000000018F, loc=50V000000000001150,
2024-12-04T09:23:10.524Z wk=2024W49, qh=172741, dir=RECEIVE, ack=-, stat=OK, ms=123, rem=44V00000000000028C, bro=50V00000000000112U, loc=50V000000000001150,
2024-12-04T09:43:57.887Z wk=2024W49, qh=172742, dir=RECEIVE, ack=-, stat=OK, ms=182, rem=46V000000000000052M, bro=50V00000000000112U, loc=50V000000000001150,
2024-12-04T09:44:10.941Z wk=2024W49, qh=172742, dir=SEND, ack=Y, stat=OK, ms=410, rem=46V000000000000052M, bro=46V00000000000015S, loc=50V000000000001150,
    
```

```

mId=7993dbb2-7640-464f-9702-40e3bf2b6aba, sz=2, type=EDO-ODM-A49-ACK, bId=3576f026b3444c6c97e162404c062d33--3576f026-b344-4c6c-97e1-62404c062d33, bSn=Nemo
mId=020db6c2-32ec-4f9e-874a-d74bc1e25b2, sz=2, type=EDO-ODM-A49-ACK, bId=cc27a7fb-f8564bb68f1b941e200d3938--cc27a7fb-f856-4bb6-8f1b-941e200d3938, bSn=Nemo
mId=383828d7-3ad0-44c1-8084-6f54d5f9e9ea, sz=15, type=EDO-ODM-A45, bId=cnull, bSn=cnull,
mId=8ee87e2d-d6ee-4d10-b025-a7075a829033, sz=2, type=EDO-ODM-A45-ACK, bId=14422825719a4186b9b23a6f21de21a6--14422825-719a-4186-b9b2-3a6f21de21a6, bSn=Nemo
mId=64b00b52-4fal-4e35-b92a-ed3c5e954633, sz=24, type=EDO-ODM-A49, bId=cnull, bSn=cnull,
mId=9f0aa920-c7bf-4319-8551-c35aec980965, sz=2, type=EDO-ODM-A49-ACK, bId=b478fce9ab78402b84e97c5b5db3e264--b478fce9-ab78-402b-84e9-7c5b5db3e264, bSn=Nemo
mId=e948e217-54a7-4331-a28f-553c792663b4, sz=3, type=NBS-REPI, bId=TIP-ESETT-20241203080430e4f666, bSn=STATNETT-SETR
mId=1b986440-5881-49f2-9a11-49727355f555, sz=1, type=NBS-ACK-REPI, bId=3b74e0fb19d74e63ab3b9ad491d6956e, bSn=STATNETT-SETR
mId=4ef0af40-07f6-40ef-8f1f-1b9ab2d3397, sz=24, type=EDO-ODM-A49, bId=cnull, bSn=cnull,
mId=efbf52b4-8790-4ce2-ace0-7d11d2c7f492, sz=2, type=EDO-ODM-A49-ACK, bId=d7ec951dd692409a9c489a81bfa02836--d7ec951d-d692-409a-9c48-9a81bfa02836, bSn=Nemo
    
```

Some of the output is self-explanatory, so here we'll try to explain what may not be evident:

- Qh is short for quarter-hour. Number of quarter-hours since 1. Jan 2020. Useful to count events pr quarter, since quarters are an important measure in the energy balancing markets.
- Ack is Y/N when you SEND a message, to indicate if we've received a technical ACK from the remote ECP-endpoint. Ack is – when we RECEIVE a message.
- Stat can be OK (normal case), DL (Delayed ACK), EX (Expired ACK) or NG (the remote endpoint sent a message, but clock on the remote endpoint is in the future, so it appears that the message was received before it was sent, hence 'negative' transmission time)

- Rem, Bro and Loc is short for Remote-endpoint, Central-Broker and Local-endpoint (your endpoint)
- Sz is the Size of the message in KB
- BId is the BAMessageId
- BSn is the BASender

You get the same information plus a little more if you run EML like this:

```
java -jar ekit.jar EML
```

A reasonable configuration to run would be like this, where the black argument below indicates where you store your ecp-logs.

```
java -jar ekit.jar EML -r week -m12 -z -n8 /var/log/ecp-endpoint eml.log
```

If EML stops for some reason, and is started again – it will pick up the slack so to speak, and add the missing rows in the eml-log-output, without duplicating any rows. The point is that you should be able to trust EML that it represents the real traffic.