UNICORN Systems

entsoe

ENTSO-E

Energy Communication Platform

Administration Guide v4.4.0

www.unicornsystems.eu

Jankovcova 1037/49, CZ – 170 00 Prague 7

Project:	ENTSO-E
Project – Subject:	Energy Communication Platform
Document Title:	Administration Guide v4.4.0
Author:	Jiří Dudek, Jiří Neuman
Contact:	E-mail: info@unicornsystems.eu Tel.: (+420) 221 400 111

1 Executive Summary

This document describes administrative tasks for all ECP components.

2 Content

3 Revision History

Version	Date	Author	Description
4.1.4	02.02.2018	Laura Betcherová	Documentation review
4.2.0	21.03.2018	Josef Brož	Version update
4.3.0	26.06.2018	Josef Brož	Version update
4.3.0	05.10.2018	Barbora Klabačková	Java update procedure
4.3.0	18.10.2018	Barbora Klabačková	Certificate renewal update
4.4.0	04.02.2019	Jiří Kakrda	Broker NAT
4.4.0	05.06.2019	Barbora Kánská	Document revision

4 Common Topics

4.1 How to change the configuration

ECP has two sources of configuration:

> System configuration

> Runtime configuration

4.1.1 System configuration

System configuration is in the properties file on the disk. After changing the properties, the component must be restarted to update the configuration. System configuration contains only parameters that are required for start-up (such as database connection parameters). System configuration must be manually propagated to all nodes of the cluster.

Location of the system configuration files for all ECP components:

Component	Linux	Windows	Description
ECP endpoint	/etc/ecp-endpoint/ecp.properties	<installation directory>/config/ecp.properties	System configuration of ECP endpoint.
ECP component directory	/etc/ecp-directory/ecp-directory.properties	<installation directory>/config/ecp-directory.properties	System configuration of ECP component directory.
ECP broker	broker.properties	broker.properties	System configuration of ECP broker. The file location depends on individual installation.

4.1.2 Runtime configuration

Runtime configuration can be changed during the operation of the system without the need to restart it. When using HA deployment, configuration changes will affect all nodes in the cluster.

The runtime configuration is available only on the endpoint and component directory.

4.1.2.1 Export current configuration

Open the component directory or endpoint web interface and navigate to the settings page. Scroll down to the configuration section and click the export configuration button. A properties file with the current configuration will be downloaded and will have the following file name pattern: <Component Code>-configuration.properties.

4.1.2.2 Editing the configuration

Open the properties file with a text editor. Any text editor will work although an editor with syntax highlighting (Atom, Notepad++, etc.) is recommended for better readability and navigation.

Each property entry has its purpose comments (example values, possible enum values, further description etc.). Lines starting with # are ignored by the interpreter while importing configuration back into system.

Note: the single backslash character is considered as an escape sequence, therefore, if backslash is required it should be presented twice e.g. dataDirectory=C:\\ECPEndpoint.

4.1.2.3 Import the updated configuration

Change desired configuration properties and upload the configuration back into the system. The import configuration button is beside the export configuration button.

The result of configuration import will be shown at the top of the page.

There are two possible results:

> Import was successful.
  - This indicates successful processing and your component directory/endpoint will reload its configuration shortly.
> Import was not successful.
  - This message will indicate what went wrong e.g. unknown configuration property, invalid value, etc. All invalid properties will be present in the error message to assist the administrator in resolving the issues.

4.2 How to recover from an incorrect configuration

If the application fails to start due to an incorrect configuration, it is possible to use recovery mode. Starting in recovery mode means that the application will load only minimal configuration to display visual use cases to allow the import and export of the current configuration and resolve the issue.

Recovery mode is activated in the system configuration file, to enable recovery mode use the following property: recoveryMode=true. This property needs to be set in the system configuration.

A typical recovery procedure consists of the following steps:

> Stop the failing application

> Locate the system configuration file and paste “recoveryMode=true” to the end of the file (without quotes)

> Start the application and change the configuration using the runtime configuration method

> Stop the application and either remove or comment out the line with the recovery property

> Start the application

4.3 How to set a different application theme

ECP allows the theme of the application to be changed to suit distinct environments (production, test, etc.). This property can be changed in the runtime configuration.

Property	Description	Default value
ecp.appTheme	This property allows customization of the look and feel of the ECP application. Permitted values are DEFAULT, BLUE, WHITE.	DEFAULT

4.4 Recommended JMX remote settings

All ECP components supports monitoring by using JMX. To activate this function, see chapter How to enable JMX for a given component. This chapter describes how to configure secured remote JMX access.

When enabled, Remote JMX interface must be secured by authentication (login/password) and SSL. The following JMX remote properties must be set:

- com.sun.management.jmxremote=true
- com.sun.management.jmxremote.port=<PORT_NUMBER>
- com.sun.management.jmxremote.rmi.port=<PORT_NUMBER>
- com.sun.management.jmxremote.password.file=<PATH_TO>/jmxremote.password
- com.sun.management.jmxremote.access.file=<PATH_TO>/jmxremote.access
- com.sun.management.jmxremote.ssl=true
- com.sun.management.jmxremote.registry.ssl=true
- com.sun.management.jmxremote.ssl.config.file=<PATH_TO>/jmxremote.ssl

4.4.1 JMX authentication settings

JMX authentication is based on predefined users and their permissions. Users and their passwords are specified in ***jmxremote.password*** file. User permissions are specified in ***jmxremote.access*** file. Those files must be created and referenced by com.sun.management.jmxremote.password.file and com.sun.management.jmxremote.access.file jmx property.

Example of *jmxremote.password* file content

User user_password

Administrator admin_password

Example of *jmxremote.access* file content

User readonly

Admininstrator readwrite

Security policy of JMX requires restricted access to the **jmxremote.password** file. The file must be readable by the owner only. In addition to this, file must be readable only by the user who starts the ECP server.

To setup required permission on Linux machine, use the following commands:

- chmod 600 jmxremote.password
- chown <user>:<group> jmxremote.password
    - replace <user> and <group> with ecp user and group

To setup required permission on Windows machine, use the following commands:

- icacls jmxremote.password /grant:r SYSTEM:(r)
- icacls jmxremote.password /inheritance:r
- Use Windows Explorer to navigate to the jmxremote.properties file - show file properties – tab security – click advance button and change owner to SYSTEM and confirm dialog.

4.4.2 JMX SSL configuration settings

Server keypair/certificate must be created to enable remote JMX SSL. OpenSSL tool can be used to achieve this with self-signed certificates.

JMX SSL settings file **jmxremote.ssl** must be created and referenced by com.sun.management.jmxremote.ssl.config.file property.

Example of jmxremote.ssl file

- javax.net.ssl.keyStore=/path/to/keystore.jks
- javax.net.ssl.keyStorePassword=keystore_password
- javax.net.ssl.trustStore=/path/to/truststore.jks
- javax.net.ssl.trustStorePassword=truststore_password

Keystore path is specified by the javax.net.ssl.keyStore property. Java Keystore with private key matching the server name along with certificate chain is expected.

Truststore is specified by the javax.net.ssl.trustStore property. Java Keystore with trusted certificates is expected.

4.5 Multiple Root Certificate Authorities

Every ECP Component trusts the Root Certificate Authority (CA) for the authentication purposes. Root CA issues the certificates for the Integrated CA of every Component Directory. Every authentication certificate in the ECP network is issued by an Integrated CA. ECP components, which authentication certificates come from different Root CA, cannot communicate to each other by default.

Component Directory administrator have possibility to import Root CA and Integrated CA from another Component Directory to its own Component Directory in order to enable communication between the components with different Root CA in one ECP network. Certificate Authorities from another Component Directory are distributed to Endpoints and Brokers by synchronization jobs. To enable communication between directories, Administrator of the second CD will have to import CA from the first CD.

Table “Certificates” on Settings page on Component Directory and Endpoint is divided into two tabs – “Local” and “Network”. “Local” tab shows own certificates or certificates belonging to the home Component Directory. “Network” tab show certificates belonging to another Component Directories.

Component Directory Administrator can import Root CA (Global CA) and Integrated CA of another Component Directory by clicking on “Import Network Certificate” button on Settings page. It is possible to import multiple different certificates of the same type belonging to one Component Directory. After Certificate Authorities update, Component Directory and Broker **must be manually restarted** in order to reload SSL context. Endpoint will restart its external broker automatically.

4.6 Encrypted passwords

For security reasons, passwords in properties files can be written in an encrypted form instead in a plain text. Encrypted password in the properties file must be wrapped in brackets with the keyword **ENC**.

Example of an encrypted password in ecp-users.properties file:

ecp.endpoint.users[0].password = ENC(XcBjfjDDjxeyFBoaEPhG14wEzc6Ja+Xx+hNPrJyQT88=)

Following properties must be set to enable passwords decryption:

Property	Description	Default value
ecp.password.location	Path to the file with stored password, which was used for encryption. Password in the file must be in the first line of the file.	N/A
jasypt.encryptor.algorithm	Algorithm used for encryption. All passwords must be encrypted with this algorithm.	PBEWithMD5AndDES

You can use Jasypt tool for passwords encryption. For more information see: http://www.jasypt.org/cli.html.

4.7 How to enable separate Audit Log file

By default, all audit information is logged at the INFO level into the ECP application log file. Audit log records can be also written to separate file only. Rolling over policy based on file size is then applied for audit log file. Separate audit log file can be enabled by the following settings:

- Set Java property ECP_LOG_PROFILES=ecp-audit
- Set the path to the log directory via the logging.path system configuration property (only when upgrading from ECP version 4.3.)

For detailed instructions how to set Audit Log Java property see Audit Log chapters of ECP components.

List of Audit events and for which components they are relevant is documented in the ECP4 AuditLog v4.4.0.xlsx.

5 ECP Endpoint Administration

5.1 User authentication

To prevent unauthorized access, ECP endpoint can be configured to require users to log in and to include username and password in the header of the web service request. It is strongly recommended to use HTTPS connection when user authentication is enabled. See chapter 5.12 for details how to enable HTTPS.

To enable user authentication in Endpoint GUI, find the property *spring.profiles.active* in the ecp.properties file and delete profile *disable-user-auth* from the list of profiles. To enable user authentication in web services, find the property *spring.profiles.active* in the ecp.properties file and delete profile *disable-ws-auth* from the list of profiles. With enabled web service authentication, every web service request must include the header following the WS-Security protocol. Example:

```
<soapenv:Header>
        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" wsse:mustUnderstand="1">
            <wsse:UsernameToken>
                <wsse:Username>admin</wsse:Username>
                <wsse:Password>password</wsse:Password>
            </wsse:UsernameToken>
        </wsse:Security>
    </soapenv:Header>
```

A list of allowed users is defined in ecp-users.properties file in the configuration directory. There are two user roles: admin and user. Only admin role has access to the Settings page and all its functionality and can use web services related to message paths.

A new installation comes with two predefined users: admin/password and user/password. It is strongly recommended to replace these users with your own or at least change the passwords to a more secure one.

The pattern is:

ecp.endpoint.users[index].login=user

ecp.endpoint.users[index].password=password

ecp.endpoint.users[index].role=role

The index must be a whole number from a sequence starting with zero. Example:

ecp.endpoint.users[0].login=janedoe

ecp.endpoint.users[0].password=securepassword

ecp.endpoint.users[0].role=admin

ecp.endpoint.users[1].login=johndoe

ecp.endpoint.users[1].password=secret

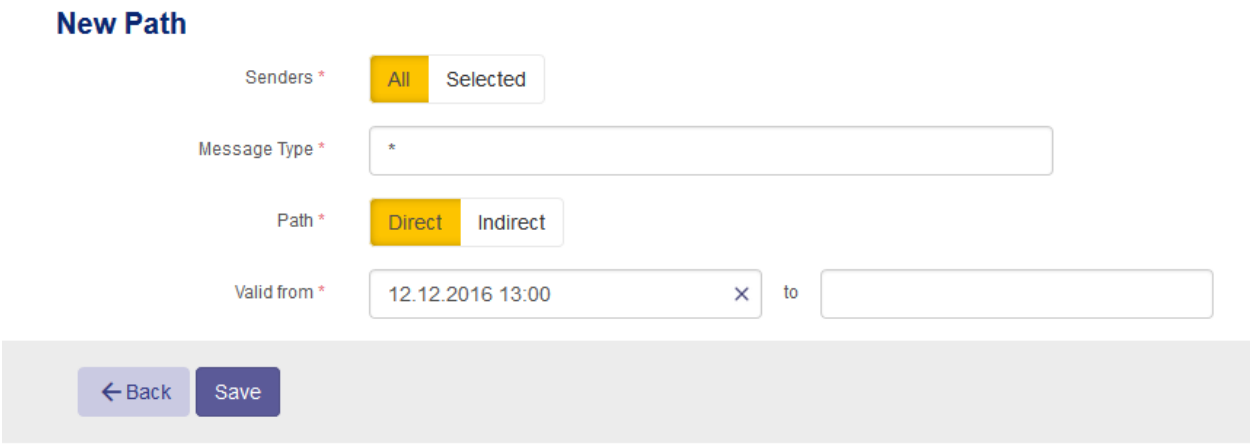
ecp.endpoint.users[1].role=user

5.2 How to configure message paths

After installation of ECP endpoint there will be no message paths present and it will not be possible to send or receive messages to or from the current endpoint. The following instructions describe how to add a new message path.

5.2.1 How to add a new message path

Open the endpoint web interface and navigate to page settings. Locate the paths section and click the new path button. A new path dialog box will open.



Indirect message paths can be created for existing intermediate components (typically ECP broker), which must allow communication with endpoint. See also configuration of broker filter: How to configure component filter.

Direct message paths can be created only when direct messaging is enabled on endpoint. For instructions on how to enable direct messaging on endpoint see How to enable direct communication.

Message path message type support's the definition of message type with a wild card character. For message path matching rules description see also chapter Message type matching rules.

When a new message path is successfully created, it will be distributed to other ECP components through the component directory synchronization. Please note that this process is asynchronous and requires time to complete.

When you want to remove a message path, execute these two steps:

1. Instead of deleting the message path, change its validity to the near future so that all components in the network will know that a given message path will no longer be valid and won't allow any message to be sent using this path.
2. Create a new message path and set its validity start to the date and time when the removed message path validity ends.

5.2.2 Import message paths in a file

Endpoint support batch import of message paths defined in a JSON file. To import message paths file, click on the "Import Paths" button in Settings page of Endpoint GUI. File structure reflect message path structure, single record represents one message path.

Attribute	Type	Description	Required
messageType	string	A text, possibly with a wildcard (*) at the end (e.g. ABC*), which indicates the message-type of the messages that can use the path.	True
path	string	Format: {DIRECT | INDIRECT }:<broker code> <broker code> is mandatory when an INDIRECT path.	True
senderComponent	string[]	The collection of the codes of the sender endpoints that can use the path to send a message. A wildcard (*) means any endpoint.	True
validFrom	dateTime	The date time (inclusive) from which the path is usable.	True
validUntil	dateTime	The date time (exclusive) until the path is usable, forever when not set.	False

Example of a message paths file:

```
[{
  "messageType": "TYPE1",
  "path": "DIRECT",
  "senderComponent": ["*"],
  "validFrom": "2019-05-20T12:00:00+01:00",
  "validUntil": "2020-05-20T12:00:00+01:00"
},
{
  "messageType": "TYPE2",
  "path": "INDIRECT:B1",
  "senderComponent": ["EDC1",”EDC2"],
  "validFrom": "2019-05-20T12:00:00+01:00"
}]
```

Import procedure starts with message paths validation. If single validation error occurs, import procedure is interrupted and validation result is returned to the import caller. Successfully imported message paths are pushed by the Endpoint to the Component Directory.

5.2.3 How to configure processing of centrally managed message paths

Following properties configures processing of centrally managed message paths:

Property	Description	Default value
ecp.endpoint.messagePath.synchronization.enabled	Enable / disable synchronization of centrally managed message paths with Component Directory	true
ecp.directory.client.synchronization.messagePathSynchronizationInterval	Defines period how often the synchronization routine will be run. Value is defined as CRON expression.	0 */5 * * * *

5.3 How to configure FSSF

FSSF can be used for both sending and receiving of messages. It is also possible to configure endpoint FSSF only for message receiving/sending. Note that all directories used by FSSF should exist and have appropriate permissions for file read/write operations (the user under which ECP is running must have read/write permission).

To enable FSSF set the configuration property “ecp.endpoint.fssf.enabled” value to true.

Property	Description	Default value
ecp.endpoint.fssf.enabled	Enable / disable FSSF interface on endpoint.	false

5.3.1 Sending messages through FSSF

To specify the outgoing directory for sending messages through FSSF set the configuration property “ecp.endpoint.fssf.outgoingDirectory” value to the desired directory path (e.g. /opt/fssf/out).

Together with the outgoing directory, the directories for message processing log and errors must also be specified.

To specify the message processing log directory set the configuration property “ecp.endpoint.fssf.logDirectory” value to the desired directory path.

Single log file per day for message processing can be enabled by setting the property “ecp.endpoint.fssf.useSingleLogPerDay” value to true. The default FSSF logging setting is one log file per sent message.

To specify the FSSF error directory where invalid files are sent, set the property "ecp.endpoint.fssf.outgoingErrorDirectory" value to the desired directory path.

All properties required for sending messages over FSSF are listed in the table below:

Property	Description	Default value
ecp.endpoint.fssf.outgoingDirectory	Outgoing directory path.	N/A
ecp.endpoint.fssf.logDirectory	Message processing logs directory path.	N/A
ecp.endpoint.fssf.outgoingErrorDirectory	Unprocessed messages directory path.	N/A
ecp.endpoint.fssf.useSingleLogPerDay	Enable single log file per day for all processed messages.	false

5.3.2 Receiving messages through FSSF

The following receive strategies can be configured:

Incoming messages are received to a single directory regardless of their type.

Incoming messages are routed to different directories based on their message type or group of messages types.

5.3.2.1 Single incoming directory

To specify a single incoming directory for receiving messages regardless of their type, set the configuration property "ecp.endpoint.fssf.incomingDirectory[0].incomingDirectory" value to the desired directory path and set the message type wild character * as a value of the property ecp.endpoint.fssf.incomingDirectory[0].messageTypes

5.3.2.2 Multiple incoming directories

To specify incoming directory per message type(s), the following tuples of properties are required:

ecp.endpoint.fssf.incomingDirectory[n].incomingDirectory = <directory>

ecp.endpoint.fssf.incomingDirectory[n].messageTypes = <message type>

where n represents a number from a continuous series of integers starting from 0.

Multiple message types must be separated by a coma. For message type matching rules description see Message type matching rules.

Property	Description	Default value
ecp.endpoint.fssf.incomingDirectory[n].incomingDirectory	Incoming directory path for messages specified by message types defined by next property.	N/A
ecp.endpoint.fssf.incomingDirectory[n].messageTypes	Incoming directory message type(s). Multiple types must be separated by a coma. See also section 5.3.3 Message type matching rules.	N/A

5.3.3 Message type matching rules

Message type configuration properties support the matching of message type strings against a wildcard character * or a combination of wildcard character and string.

Possible message type specifications are:

> wildcard character *

> message type plus wildcard

> message type substring plus wildcard.

> plain message type

The order of properties list specified by configuration is important. The first property matched is selected e.g.

Directory[0].messageType = type*

Direcotry[1].messageType = typeA

Directory[2].messageType = *

Given message type: typeA first property is chosen by system as matching.

Given message type: A the last property is chosen by system as matching.

Obviously, when multiple message types separated by a coma are configured within a single property, message types order within the single property is not important. See the following example:

Directory[0].messageType = typeA, typeB, typeC

However, it is illogical to configure:

Directory[0].messageType = type, *

because the wildcard character matches any message type

5.4 How to configure AMQP API

To enable the AMQP interface set the property “ecp.endpoint.amqpApiEnabled” value to true. Enabled AMQP interface with default settings provides three AMQP queues available for external systems:

> ecp.endpoint.outbox - queue for sending messages

> ecp.endpoint.inbox – queue for receiving messages

> ecp.endpoint.outbox.reply – queue for reply messages

The default inbox queue receives all messages regardless of type.

Property	Description	Default value
ecp.endpoint.amqpApiEnabled	Enable AMQP API by setting this property value to true	false

5.4.1 Setup of multiple receive queues

Several receive queues can be configured for message types or groups of types. The received message is then routed by message type to a queue with a matching message type following the rules described in section Message type matching rules.

To specify the receive queue per message type(s), the following tuples of properties are required to be set: (one tuple per queue):

> ecp.endpoint.routes.incomingQueue[n].incomingQueue = <queue name>

> ecp.endpoint.routes.incomingQueue[n].messageTypes = <message type>

where n represent a number from a continuous series of integers starting from 0.

Multiple message types must be separated by a coma. For message type matching rules description see Message type matching rules.

Property	Description
ecp.endpoint.routes.incomingQueue[n].incomingQueue	Incoming queue for messages specified by message types defined by next property.
ecp.endpoint.routes.incomingQueue[n].messageTypes	Incoming queue message type(s). Multiple types must be separated by a coma. See also section 5.3.3 Message type matching rules.

5.4.2 How to enable AMQP send handler

ECP Endpoint comes with default implementation of send handler for AMQP API. The AMQP send handler is pushing incoming SendEvents into queue named:

- ecp.endpoint.send.event

To enable the AMQP send handler set handler properties as described at table below:

Property	Description
ecp.endpoint.sendHandler [n].beanName	Specify AMQP send handler bean name. This value must be set to: **amqpApiSendHandler** to enable the AMQP send handler.
ecp.endpoint.sendHandler [n].typeName	This property specifies send handler message types. For message type matching rules description see section 5.3.3 Message type matching rules. Use the wildcard character * to enable handler invocation for all messages regardless of their message types.

> where n represents a number from a continuous series of integers starting from 0.

5.4.3 How to secure AMQP API

By default, the ECP Endpoint AMQP API authentication and transport level security is not enabled. To enable login - based authentication and transport level security (AMQPS) the following steps must be done.

System configuration settings change

Property	Description
internalBroker.useAuthentication	Turns on login-based authentication and internal broker SSL.
internalBroker.auth.user	Sets endpoint username for internal broker. Login names and passwords are configured in *user.properties* file.
internalBroker.auth.password	Endpoint user password for internal broker. Login names and passwords are configured in *user.properties* file.
internalBroker.keystore.location	Path to the KeyStore with SSL certificate. AuthKeystore.jks file path can be used.
internalBroker.keystore.password	SSL keystore password.
internalBroker.keystore.authAlias	SSL certificate alias. When AuthKeystore.jks is used, use alias ecp_module_auth.

Creation of the authentication settings files

Authentication settings files **groups.properties** and **users.properties** must be created in the following locations:

> For Windows OS: <install_path>

> For Linux OS: /etc/ecp-endpoint

Example of the groups.properties file	***Example of the users.properties file***
admins=endpoint,toolbox tempDestinationAdmins=endpoint,toolbox users=endpoint,ba,toolbox	endpoint=<password> toolbox=<password> ba=<password>

5.5 How to configure archiving

ECP endpoint provides file system archiving capabilities, however an additional custom archiving plugin can be implemented and used instead of the default file system archiving component.

5.5.1 Setup of file system archiving

To enable the file system archive handler component set the property “ecp.endpoint.archiveEnabled” value to true. The file system archiving component saves message content and message metadata to separate files created within configured directories.

Additional file system handler properties must be set to enable archiving functionality. See table below:

Property	Description	Default value
ecp.endpoint.archiveEnabled	To enable archiving set value of this property to true.	false
ecp.endpoint.archiveHandler.beanName	Use this property to specify selected archive handler. To enable file system archive handler set value to **fileSystemArchiveHandler.**	N/A
ecp.endpoint.archiveHandler.typeName	This property specifies message types to be archived. For message type matching rules description see section 5.3.3 Message type matching rules. Use the wildcard character * to enable archiving of all messages regardless of their message types.	N/A
ecp.endpoint.archive.binaryPathExpression	Define directory path for message content archive files. Path can be constructed dynamically from message properties. See available path expression described in section 5.5.1.1 List of available path expressions.	N/A
ecp.endpoint.archive.metadataPathExpression	Define directory path for message metadata archive files. See available path expression described in section 5.5.1.1 List of available path expressions	N/A

5.5.1.1 List of available path expressions

The following table shows available message properties expressions that can be used for file system archiving paths construction:

Expression	Description
#{#sender}	Message sender code.
#{#receiver}	Message receiver code.
#{#messageType}	Message type.
#{#messageID}	Message ID.
#{#extension}	Message extension (file extension for messages sent over FSSF).
#{#timeSent.Year}	Year part of message sent date.
#{#timeSent.MonthValue}	Mont part of message sent date.
#{#timeSent.DayOfMonth}	Day part of message sent date.

The recommended path expression pattern for content archiving is:

> /#{#timeSent.Year}/#{#timeSent.MonthValue}/#{#timeSent.DayOfMonth}/#{#sender}_#{#receiver}_#{#messageType}_#{#messageID}.#{#extension}

The recommended path expression pattern for metadata archiving is:

> /#{#timeSent.Year}/#{#timeSent.MonthValue}/#{#timeSent.DayOfMonth}/#{#sender}_#{#receiver}_#{#messageType}_#{#messageID}-metadata.xml

Note that paths patterns above are not complete and parent archive directory should be added before the beginning of the patterns.

5.5.2 Registration of custom archive handler

Custom implementations of the archiving handler must be configured properly to be used by ECP endpoint. The implementation class of archive handler can be registered by class name (e.g. eu.custom.archive.handler.HandlerClassName).

For how to deploy custom archive plugin instructions see section 5.14.

Property	Description	Default value
ecp.endpoint.archiveHandler.className	Specify custom handler implementation by Java class name (e.g. eu.custom.archive.handler.HandlerClassName)	N/A
ecp.endpoint.archiveHandler.typeName	This property specifies message types to be archived. For message type matching rules description see section 5.3.3 Message type matching rules. Use the wildcard character * to enable archiving of all messages regardless of their message types.	N/A

5.6 How to configure receive handlers

Multiple handlers can be registered for different message types. The receive handler is invoked when a standard message with matching message type is received by ECP endpoint. Handler implementations are configured as listed in the following properties:

Property	Description	Default value
ecp.endpoint.receiveHandler[n].className	Specify custom handler implementation by Java class name (e.g. eu.custom.receive.handler.HandlerClassName)	N/A
ecp.endpoint.receiveHandler[n].typeName	This property specifies receive handler message types. For message type matching rules description see section 5.3.3 Message type matching rules. Use the wildcard character * to enable handler invocation for all messages regardless of their message types.	N/A

> where n represents a number from a continuous series of integers starting from 0.

For how to deploy receive handler instructions see section 5.14

5.7 How to configure send handlers

Multiple handlers can be registered for different message types. The send handler is invoked when an acknowledgement message with matching message type is received by ECP endpoint. Handler implementations are configured as listed in the following properties:

Property	Description	Default value

ecp.endpoint.sendHandler [n].className	Specify custom handler implementation by Java class name (e.g. eu.custom.send.handler.HandlerClassName)	N/A
ecp.endpoint.sendHandler [n].typeName	This property specifies send handler message types. For message type matching rules description see section 5.3.3 Message type matching rules. Use the wildcard character * to enable handler invocation for all messages regardless of their message types.	N/A

> where n represents a number from a continuous series of integers starting from 0.

For how to deploy send handler instructions see section 5.14

5.8 How to configure message expiration

Messages expire after a defined time to live (TTL) value. TTL value is defined in seconds.

Property	Description	Default value
ecp.endpoint.messageTtl	This property specifies TTL in seconds.	14x24x60x60 (14 days)

Expired messages are processed by the message expiration job. Its purpose is to change the state of expired messages to final state – FAILED. The interval for executing this job can be set by the property described below:

Property	Description	Default value
ecp.messagebox.messageExpirationRunPeriod	Define period how often the message expiration job [ms] will be run.	60x60x1000 (one hour)

5.9 How to configure message priority

Priority is assigned to messages based on message type. Supported priority values are from 0 (lowest) to 9 (highest). Message priorities are configured as listed in the following properties:

Property	Description	Default value
ecp.endpoint.priorityConfiguration[0].priority	Message priority in range 0-9 inclusive is supported.	N/A
ecp.endpoint.priorityConfiguration[0].messageType	This property specifies which priority has message type. For message type matching rules description see section 5.3.3 Message type matching rules.	N/A

> where n represents a number from a continuous series of integers starting from 0.

5.10 How to configure logging

You can change the log file location or configure the logger severity for a particular package through the system configuration.

5.10.1 Configure log file location

Log file location can be set via the configuration property logging.file.

Example:

logging.file=/var/log/ecp.log

5.10.2 Configure logger severity

Logger severity for a particular package can be set via the configuration property in the system configuration file. The pattern is logging.level.<package>=SEVERITY. Severity can be one of the following values: TRACE, DEBUG, INFO, WARN, ERROR. Please bear in mind that setting logger severity to TRACE or DEBUG will produce a huge quantity of log messages resulting in increased storage usage and may also slow down the application. Setting logger severity to TRACE or DEBUG should be for diagnostic purposes only.

Example:

logging.level.eu.entsoe.ecp.endpoint.fssf=DEBUG

5.11 Enable HTTP basic authentication

To prevent unauthorized access to the endpoint, HTTP basic authentication can be set. Please note that this option is available only for standalone application deployment. It is strongly recommended to use HTTPS connection when basic authentication is enabled.

It is recommended to use ECP user authentication instead of HTTP basic authentication. See chapter 5.1 for more details.

5.11.1 Tomcat configuration

Navigate to your installation directory and locate the following files:

Windows:

> <install_path>\tomcat\conf\web.xml

> <install_path>\tomcat\conf\tomcat-users.xml

Centos/RHEL:

> <install_path>/conf/web.xml

> <install_path>/conf/tomcat-users.xml

User and roles are defined in tomcat-users.xml in the following format:

<role rolename="ecp-role"/>

<user username="ecp-user" password="<change-me>" roles="ecp-role"/>

Authentication is enabled by adding the security constraint element the web.xml (see following snippet). The following snippet should be pasted as is. The only rule is that users defined in tomcat-users.xml are assigned the same role as the role name in the "auth-constraint" section.

```
<security-constraint>
        <web-resource-collection>
                <web-resource-name>Wildcard means whole app requires authentication</web-resource-name>
                <url-pattern>/*</url-pattern>
                <http-method>GET</http-method>
                <http-method>POST</http-method>
                <http-method>PUT</http-method>
                <http-method>DELETE</http-method>
                <http-method>OPTIONS</http-method>
        </web-resource-collection>
        <auth-constraint>
```

```
            <role-name>ecp-role</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
</security-constraint>
<login-config>
        <auth-method>BASIC</auth-method>
</login-config>
```

5.12 Enable HTTPS

It is also possible to setup Transport Layer Security (TLS) between the endpoint and other business applications. This option is available only for standalone deployment.

5.12.1 Tomcat configuration

Navigate to your installation directory and locate the following files:

Windows:

> <install_path>\tomcat\conf\server.xml

Centos/RHEL:

> <install_path>/conf/server.xml

HTTPS is enabled by adding a new connector definition to Tomcat's server.xml. The parameter keystoreFile targets to standard Java keystore file. This file must contain either a public or private key. Any trusted keypair can be used including the authentication one present in ECP. An example is shown using built-in authentication keystore:

```
<Connector
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443" maxThreads="200"
        scheme="https" secure="true" SSLEnabled="true"
        keystoreFile="/var/lib/authKeystore.jks " keystorePass="<change-me>"
        clientAuth="false" sslProtocol="TLS"
        sslEnabledProtocols="TLSv1.2,TLSv1.3"
        ciphers="ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256"
/>
```

A non-secured connector (port 8080) must be removed or commented out should you wish to use secured communication only.

The blue marked parameter sslEnabledProtocols is optional. Values are represented as comma separated list of SSL protocols to support for HTTPS connections. If specified, only the protocols that are listed and supported by the SSL implementation will be enabled. If not specified, the JVM default (excluding SSLv2 and SSLv3 if the JVM enables either or both of them by default) is used.

Parameter ciphers specifies which ciphers can be used for HTTPS connection. If not specified, the JVM default ciphers from protocols specified in sslEnabledProtocols are used.

5.13 How to enable SOCKS Proxy

Endpoint can be configured to communicate with other ECP components through the SOCKS5 proxy server. The following system configuration properties (see System configuration) are used to setup ECP Endpoint communication through proxy.

Property	Description	Default value
ecp.security.proxy.enabled	turns the use of proxy on / off	False
ecp.security.proxy.host	the hostname or IP address of the proxy server	N/A
ecp.security.proxy.port	the port of the proxy server	1080
ecp.security.proxy.nonProxyHosts	list of hostnames / IP addresses for which the proxy is not used. This can be used, for example, to exclude the communication with Endpoint's external AMQP broker.	N/A
ecp.security.proxy.username	optional username for authentication on the proxy server	N/A
ecp.security.proxy.password	optional password for authentication on the proxy server	N/A

ECP was tested with a Dante proxy server (https://www.inet.no/dante/) with the default configuration.

5.14 How to add plugins to ECP

ECP endpoint plugins (e.g. send, receive or archive handlers) which implement ECP endpoint Java public API can be deployed to ECP endpoint as Java jar packages. To deploy the plugin on Windows, copy the plugin jar file to <endpoint install path>/tomcat/webapps/ECP_MODULE/WEB-INF/lib/.

On Linux, if the ECP endpoint was just installed or upgraded and was not running yet, it is necessary to unzip ECP_MODULE.war located in <endpoint install path>/webapps/. Then copy the plugin jar file to <endpoint install path>/webapps/ECP_MODULE/WEB-INF/lib/.

In case of ECP endpoint installed in HA configuration, jar packages must be present at all cluster nodes.

5.15 How to enable direct communication

Before using the direct messaging feature of ECP endpoint, the following properties must be set:

Property	Description	Default value
ecp.endpoint.directMessagingEnabled	To enable direct messaging set the value of this property to true.	false
ecp.urls	This property specifies the URL of the external endpoint broker. The typical URL pattern looks like: **amqps://<host name>:<port>** where default port is set to **5671**. Multiple URL's separated by a coma can be configured .	N/A

5.16 How to configure message box

There are several configuration properties that influence message box behaviour. See descriptions below:

Property	Description	Default value
ecp.messagebox.retentionPeriod	Defines the retention period of the message box in milliseconds. When the retention period of the archived messages expires, they will be deleted from the message box.	1000x60x60x24x14 (14 days)
ecp.messagebox.messageDeletingRunPeriod	Define period of how often (in milliseconds) the message deleting routine will be run.	1000*60*60 (one hour)

5.17 How to configure component directory synchronization

Synchronization of ECP endpoint with component directory is done by a periodically executed routine that must be properly configured with the following properties:

Property	Description	Default value
ecp.directory.client.synchronization.homeComponentDirectoryPrimaryUrl	Defines URL of primary component directory. This URL will be used: - for the very first synchronization of the component directory. - when the URL of primary CD differs from the URL specified at component directory content which is not reachable.	N/A
ecp.directory.client.synchronization.homeComponentDirectoryBackupCode	Defines code of backup component directory that will be used if primary component directory is not reachable. Can be empty. Backup URL is obtained from component directory content.	N/A
ecp.directory.client.synchronization.directorySynchronizationInterval	Defines period how often the synchronization routine will be run. Value is defined as CRON expression.	0/30 * * * * *
ecp.directory.client.statistics.directorySynchronizationInterval	Defines period how often the routine synchronizing endpoint messaging statistics with component directory will be run. Value is defined as CRON expression.	0/30 * * * * *

5.18 How to configure message signing

ECP endpoint can be configured to sign outgoing messages and verify signatures of incoming messages (and is configured to do so by default).

Property	Description	Default value
ecp.security.messageSigner.default.enabled	This property enables / disables message signing and message signature verification for all messages processed by endpoint regardless their message types.	true

5.18.1 How to configure multiple message signing

Messages can be provided with an additional signature (to default one) created from custom certificates stored at filesystem key store. Configuration of additional (also called special) message signer components is described below:

Property	Description	Default value

ecp.security.messageSigner.special.enabled	This property enables / disables special message signing and message signature verification for all messages processed by endpoint regardless their message types.	false
ecp.security.keyStore.keyAlias	Special message signer key store alias (e.g. ecp_module_sign).	N/A
ecp.security.keyStore.keyStorePass	Special message signer key store password.	N/A
ecp.security.keyStore.keystorePath	Special message signer key stores directory location. The configured directory will be searched for key store files with the following name pattern: - SenderKeystore-<endpoint_code>.jks - ReceiverKeystore-<endpoint_code>.jks Sender key stores are used for signature verification in incoming messages. Receiver key stores are used for message signing of outgoing messages.	N/A

5.19 How to configure compatibility with ECP 3

Endpoint can be configured to communicate with ECP 3 network for backwards compatibility reasons. This can be used in existing ECP 3 networks for a smooth rollout of ECP 4. This compatibility is disabled by default and can be enabled by configuration described in this chapter.

Component code and certificates of the old ECP 3 Endpoint are necessary to setup the backwards compatibility.

Property	Description	Default value
ecp.endpoint.compatibility.compatibilityEnabled	Turns the compatibility with ECP 3 on (true) / off (false).	false
ecp.endpoint.compatibility.ecp3nodeUrl	URL to the home ECP 3 node where the old Endpoint is registered.	No default value
ecp.endpoint.compatibility.componentCode	Component code of the old ECP 3 Endpoint.	No default value
ecp.endpoint.compatibility.keystore.location	Path to the JKS file containing the certificates of the old ECP 3 Endpoint.	No default value
ecp.endpoint.compatibility.keystore.password	Password to the JKS file containing the ECP 3 certificates.	No default value
ecp.endpoint.compatibility.messageDownloadInterval	Interval for downloading new messages from the ECP 3 Node in milliseconds.	30 000 ms
ecp.endpoint.compatibility.cacheEvictionInterval	Interval for cleaning the ECP 3 Component Directory cache. The case is completely cleaned in this interval given by a CRON expression.	30 minutes

An important point to know is that Endpoint uses two separate sets of component code and certificates for ECP 3 network and for ECP 4 network. These networks are separate and ECP4 Endpoint can communicate with both of them using appropriate set of component code and certificates.

5.20 How to configure NAT

Before using network address translation (NAT), at least one record of the NAT table must be defined. Each network address is compared to the NAT table to search a matching record with the same network code, IP address and port. If a match is found, the IP address and the port is translated.

Property	Description	Default value
ecp.natEnabled	This property enables (true) or disables (false) NAT.	false
ecp.natTable[i].network	Network code of the network address. If not provided, default code is used.	N/A
ecp.natTable[i].sourceIp	Source IP address to match. This property is mandatory.	N/A
ecp.natTable[i].destinationIp	Source IP will be translated to this IP address. This property is mandatory.	N/A
ecp.natTable[i].sourcePort	Source port to match. If not provided, matching will be based only on IP address.	N/A
ecp.natTable[i].destinationPort	Source port will be translated to this port. Must be defined when sourcePort is provided.	N/A

> where *i* represents an index of a record in the NAT table starting from 0.

5.21 How to configure antivirus

To use antivirus scan on incoming message's payload, the external antivirus service must be running and accessible from the ECP Endpoint. ECP currently supports two antiviruses: ClamAV 0.100.0 and Symantec Protection Engine for Cloud Services 7.9.1.12.

Endpoint with enabled antivirus scan requires antivirus integration libraries *endpoint-antivirus-scan.jar, clamav-client-1.0.1.jar* and *SymJavaAPI-7.9.1.2.jar*. These libraries must be placed in the folder */usr/share/ecp-endpoint/webapps/ECP_MODULE/WEB-INF/lib.* Antivirus settings is configured in the endpoint runtime configuration using properties from the following table. Finally, endpoint must be restarted in order to start using the antivirus scanning.

Property	Description	Default value
ecp.endpoint.antivirus.antivirusEnabled	This property enables (true) or disables (false) antivirus scan.	false
ecp.endpoint.antivirus.antivirusClient	This property sets the antivirus service name used for scanning. Value must be either "clamav" (ClamAV server) or "symantec" (Symantec Protection Engine for Cloud Services server). This property is mandatory.	N/A
ecp.endpoint.antivirus.antivirusIp	IP address of the server where the external antivirus service is running. This property is mandatory.	N/A
ecp.endpoint.antivirus.antivirusPort	Network port of the external antivirus service.	3310 (ClamAV), 1344 (Symantec)
ecp.endpoint.antivirus.antivirusTimeout	Maximum time (in milliseconds) of an antivirus scan before timeout.	10000
ecp.endpoint.antivirus.quarantinePath	Path to the quarantine folder. This property is mandatory.	N/A

Endpoint uses IP address of the server defined in the configuration file to connect to the external antivirus service. If an infection is found, an infected payload is moved to the quarantine folder and the message will not be delivered.

5.22 How to enable Hawtio console

Hawtio has been removed from endpoint installer for the security reason. However, it is still possible to install Hawtio manually and secure it with the following instructions.

1. Download Hawtio web war from MAVEN Central Repository (https://mvnrepository.com/artifact/io.hawt/hawtio-web/1.5.11)
2. Rename downloaded hawtio-web-1.5.11.war to hawtio.war
3. Copy hawtio.war to the webapps directory (see path location below)
4. Follow the instruction below to secure hawtio

To enable an authentication on Hawtio web console, open *web.xml* file in the following path:

Windows:

> <install_path>\tomcat\webapps\hawtio\WEB-INF\web.xml

Centos/RHEL:

> <install_path>/webapps/hawtio/WEB-INF/web.xml

>

In the *web.xml* file, find *<env-entry>* section with a line *"<env-entry-name>hawtio/authenticationEnabled</env-entry-name>"*. In this section, change line *"<env-entry-value>false</env-entry-value>" to "<env-entry-value>true</env-entry-value>"* and restart ECP Endpoint.

When authentication is enabled, a login screen will be shown when Hawtio web console is opened. Users and passwords are stored in *tomcat-users.xml* file in the following path:

Windows:

> <install_path>\tomcat\conf\tomcat-users.xml

Centos/RHEL:

> <install_path>/conf/tomcat-users.xml

User and roles are defined in the following format:

<role rolename="ecp-role"/>

<user username="ecp-user" password="<change-me>" roles="ecp-role"/>

5.23 How to restrict Jolokia access

Both ECP Endpoint and Hawtio use Jolokia API. By default, Jolokia on ECP Endpoint allows connections only from localhost, while Jolokia on Hawtio is not restricted. Security policy of Jolokia is defined in *jolokia-access.xml* file. For ECP Endpoint Jolokia, it is in the following path:

Windows:

> <install_path>\tomcat\webapps\ECP_MODULE\WEB-INF\classes\jolokia-access.xml

Centos/RHEL:

> <install_path>/webapps/ECP_MODULE/WEB-INF/classes/jolokia-access.xml

For Hawtio Jolokia, it is in the following path:

Windows:

> <install_path>\tomcat\webapps\hawtio\WEB-INF\classes\jolokia-access.xml

Centos/RHEL:

> <install_path>/webapps/hawtio /WEB-INF/classes/jolokia-access.xml

In this file, <*remote*> section defines list of IP addresses, which can connect to the Jolokia. If the section is missing, all connections from any IP address are allowed. It is necessary to restart ECP Endpoint if the file is changed.

5.24 How to enable JMX for remote monitoring

ECP4 Endpoint supports monitoring by using JMX. To activate this function on ECP4 Endpoint, it is necessary to enable it.

5.24.1 Enabling JMX

For enabling JMX on Unix or Windows system, follow the steps bellow:

1. Edit file located in the folder
   - Linux system: /etc/ecp-endpoint/jmxremote.properties
   - Windows System: <install_path>\jmxremote.properties
2. Add JMX parameters described at chapter Recommended JMX remote settings.
3. Save the file.
4. Edit system configuration file (see chapter **Chyba! Nenalezen zdroj odkazů.**)
   - set property spring.jmx.enabled=true
5. Save the file.
6. Restart ECP4 Endpoint.
7. After the restart you can connect to ECP4 Endpoint using JMX.

5.25 Other configuration settings

To enable batch receiving of delivered messages (GUI use case) set the value of the following property to true:

Property	Description	Default value
ecp.messagebox.showReceiveAll	This property enables receipt of all delivered messages at once by downloading a compressed file.	false

5.25.1 How to configure connectivity check

Connectivity check use case timeout can be adjusted by the following properties:

Property	Description	Default value
ecp.endpoint.connectivityCheckAttemptTimeout	Connectivity check attempt timeout interval in milliseconds. Specifies time to wait for response from remote ECP component.	1000ms
ecp.endpoint.connectivityCheckAttemptsCount	Number of connectivity check attempts to receive tracing acknowledgements	5

5.25.2 How to configure message compression

ECP endpoint can be configured to compress messages payload to save network bandwidth. Payload compression can be enabled by message types:

Property	Description	Default value

ecp.endpoint.compressionEnabled	To enable compression of messages set value of this property to true.	true
ecp.endpoint.messageTypesToCompress	Defines which message types are marked to be compressed (multiple values separated by coma). For type matching rules description see section 5.3.3 Message type matching rules.	*
ecp.endpoint.messageTypesSkipCompress	Defines which message types are marked to skipped during compression (multiple values separated by coma). For message type matching rules description see section 5.3.3 Message type matching rules.	N/A

ECP4 is using Java deflate (level 9) compression algorithm for compression of the data before sending them over the ECP4 network. This compression algorithm best fits for the text files, e.g. text file that has 10MB can be compressed to 150kB.

The differentiation between some BusinessTypes to be compressed and others not, is because of that some messages can have the binary content that does not make sense to compress (e.g. content is already compressed).

5.25.3 How to configure JMS headers validation

ECP endpoint can be configured to validate jms headers. This configuration is enabled in default and it can be changes in parameter:

Property	Description	Default value
ecp.endpoint.jmsHeadersValidationEnabled	To enable / disable JMS headers validation (e.g. replyTo).	True

5.26 Certificates renewal

ECP Endpoint supports automatic or manual certificates renewal. Both procedures are available from the web interface of the Endpoint.

Certificates which can be renewed are: Authentication, Signing, Encryption. Registration certificate will not be renewed anymore.

- Manual renewal – navigate to the Settings tab, scroll down to the Certificates section and click on “Renew Manually”
    - Certificates can be renewed manually at any time
    - When performing manual renewal, certificates must be updated before the expiration of the old ones
- Automatic renewal – navigate to the Settings tab, scroll down to the Certificates section and click on “Enabled” under the Automatic Renewal label
    - Automatic renewal job is triggered every day at 00:00, but the certificates are renewed only when there is less than 30 days remaining until their expiration.

The certificate renewal request is approved automatically. Certificates will be shown in the Settings tab under the Certificates section.

5.27 How to update Java

If you need to upgrade Java to the current version, the following steps need to be performed:

1. Change the JRE_PATH variable to point to the new Java directory (C:\Program Files\Java\jre<current-version>)
2. Change the ECP service to point to the new Java directory

a. In the command line (run as administrator), change the directory to the folder where tomcat9w.exe is located: cd <install_path>/tomcat/bin

b. Open the configuration of Tomcat: `tomcat9w.exe //ES//ecp-endpoint`

c. Go to the Java tab and change the value of the parameter Java Virtual Machine Path to the new path of JRE.

- Example:
  `C:\Program Files\Java\jre1.8.0_121*\bin\server\jvm.dll` to the
  `C:\Program Files\Java\jre1.8.0_181*\bin\server\jvm.dll`

5.28 Backup and restore data and configuration

This chapter describes how to backup Endpoint data and configuration. If Endpoint uses external database, refer to instructions from the vendor of the external database system for database backup and restoration.

5.28.1 Windows installation

Backup these files and folders containing Endpoint data and configuration. Copy backed up files to these paths for installation restore.

- <install_path>\authKeystore.jks
- <install_path>\ecp.properties
- <install_path>\ecp-users.properties
- <install_path>\jmxremote.properties
- <install_path>\keystore.jks
- <install_path>\content
- <install_path>\db
  - if embedded database is used
- <install_path>\internalBroker
  - if embedded database is used

5.28.2 Linux installation

Backup these folders containing Endpoint data and configuration. Copy backed up files to these paths for installation restore.

- /etc/default/ecp-endpoint
- /etc/ecp-endpoint
- /var/lib/ecp-endpoint*

* This corresponds to the value of dataDirectory in ecp.properties. If the data directory is set differently, the backup path would be the one set in the ecp.properties file.

5.29 How to enable separate Audit Log file

To enable separate audit log file the ECP_LOG_PROFILES Java property must be set as described below. See also chapter You can use Jasypt tool for passwords encryption. For more information see: http://www.jasypt.org/cli.html.

How to enable separate Audit Log file.

5.29.1 Windows installation

If installed as a service, open command prompt,

- navigate to the <install_path>\tomcat\bin
- execute command tomcat9w.exe //ES//ecp-endpoint
- navigate to the Java tab

- add -DECP_LOG_PROFILES=ecp-audit to the Java Options
- restart service

Otherwise

- navigate to the <install_path>\tomcat\bin
- edit file setenv.bat
- add -DECP_LOG_PROFILES=ecp-audit to the CATALINA_OPTS
- start/restart endpoint

5.29.2 Linux installation

- navigate to the /etc/default/ecp-endpoint/
- edit file ecp-endpoint.defaults
- add -DECP_LOG_PROFILES=ecp-audit to the MANDATORY_OPTS
- start/restart endpoint

6 ECP Component Directory Administration

This section describes configurations available in the component directory.

6.1 User authentication

To prevent unauthorized access, Component Directory requires users to log in by default. To disable user authentication, find the property *spring.profiles.active* in the ecp.properties file and add profile *disable-user-auth* to the list of profiles. It is strongly recommended to use HTTPS connection when user authentication is enabled.

A list of allowed users is defined in ecp-users.properties file in the configuration directory. A new installation comes with one predefined user: admin/password. It is strongly recommended to replace this user with your own or at least change the password to a more secure one.

The pattern is:

ecp.directory.users[index].login=user

ecp.directory.users[index].password=password

The index must be a whole number from a sequence starting with zero. Example:

ecp.directory.users[0].login=janedoe

ecp.directory.users[0].password=securepassword

ecp.directory.users[1].login=johndoe

ecp.directory.users[1].password=secret

6.2 HTTPS connector configuration

Navigate to your installation directory and locate the following files:

Windows:

> <install_path>\tomcat\conf\server.xml

Centos/RHEL:

> <install_path>/conf/server.xml

In Component Directory there is https connector enabled and configured by default, if you want to change the parameters of https, you should edit one of the parameters listed in file of server.xml, where you can find similar code to:

```
<Connector
    port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150" SSLEnabled="true" scheme="https"
    secure="true" clientAuth="want" sslProtocol="TLS"
    keystoreFile="<PATH_TO_JKS>authKeystore.jks" keystorePass="<PASSWORD>" keyAlias="ecp_module_auth"
    truststoreFile="<PATH_TO_JKS>authKeystore.jks" truststorePass="<PASSWORD>"
    sslEnabledProtocols="TLSv1.2,TLSv1.3"
    ciphers="ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256"
/>
```

The blue marked parameters are paths where you store .jks files, it depends on OS and as mentioned in Endpoint configuration as a keystore file can be used any trusted keystore pair including the authentication one present in ECP (that is the default filled up parameter).

Parameter ciphers specifies which ciphers can be used for HTTPS connection. If not specified, the JVM default ciphers from protocols specified in sslEnabledProtocols are used.

6.3 Certificates renewal

ECP Component Directory supports automatic and manual certificates renewal.

Automatic certificates renewal is not enabled by default in Component Directory. To enable this feature, it is necessary to add the following property to the configuration file:

Property	Description	Default value
ecp.automaticUpdate.enabled	Enable / disable automatic certificate renewal on Component Directory	False

Certificate renewal job starts every day at 00:00. If there is less than 30 days remaining until the certificates' (Registration, Authentication) expiration, they will be automatically renewed.

The Component Directory must be restarted after the old certificates expire because Apache Tomcat needs to update its runtime Authentication certificate.

When performing manual renewal, certificates must be updated before the expiration of the old ones.

6.4 How to enable JMX for remote monitoring

ECP4 Component Directory supports monitoring by using JMX. To activate this function on ECP4 Component Directory, it is necessary to enable it.

6.4.1 Enabling JMX

For enabling JMX on Unix or Windows system, follow the steps bellow:

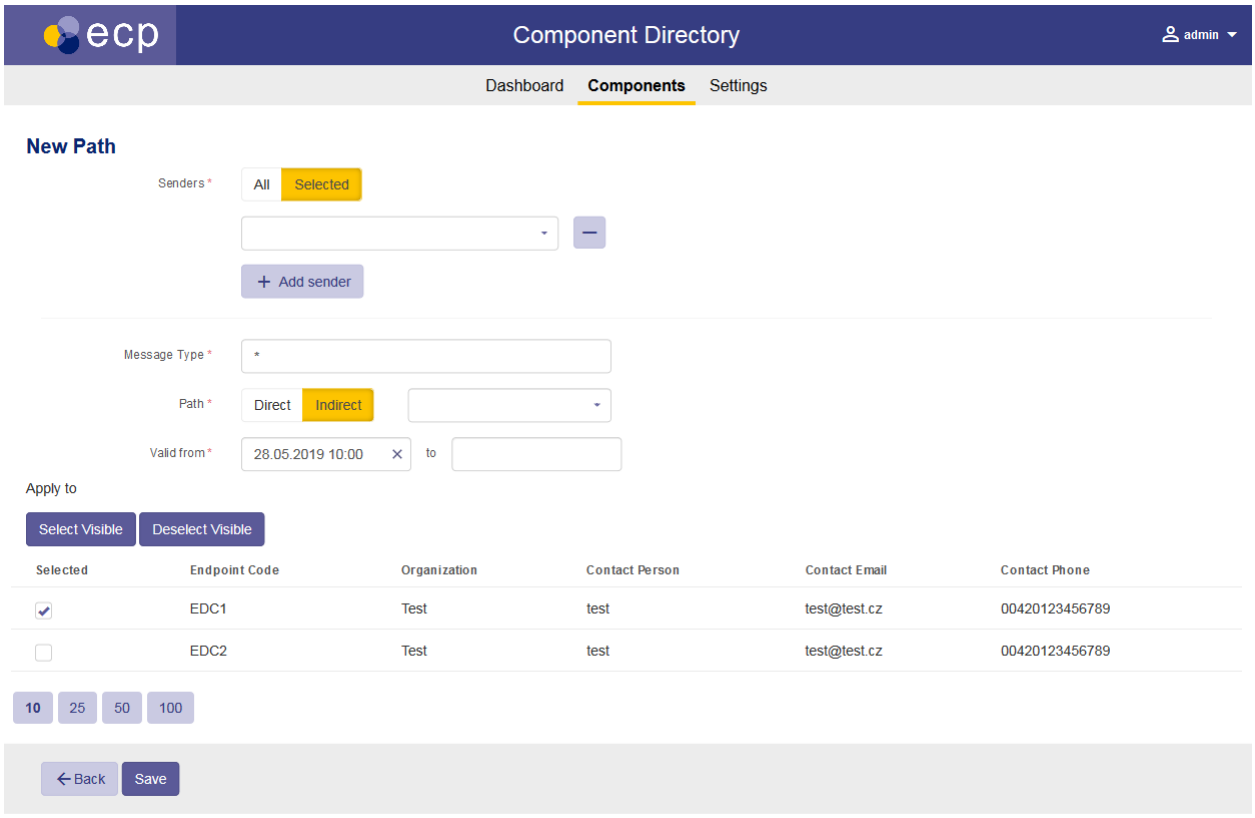
1. Edit file located in the folder
   - Linux system: /etc/ecp-directory/jmxremote.properties
   - Windows System: <install_path>\jmxremote.properties
2. Add JMX parameters described at chapter Recommended JMX remote settings.
3. Save the file.
4. Edit system configuration file (see chapter **Chyba! Nenalezen zdroj odkazů.**)
   - set property spring.jmx.enabled=true
5. Save the file.
6. Restart ECP4 Component Directory.
7. After the restart you can connect to ECP4 Component Directory using JMX.

6.5 Central Message Paths Management

Instead of managing message paths for individual Endpoint, it is possible to add or remove message paths centrally through the Component Directory GUI. Endpoint periodically run message path synchronization job with its home Component Directory, which updates message paths on Endpoint.

6.5.1 How to add a new centrally managed message path

Open the Component Directory web interface and navigate to Components page. Open Central Management Paths tab in Registered components section and click the New Path button. A new path dialog box will open.



Indirect message paths can be created for existing intermediate components (typically ECP broker), which must allow communication with endpoints. See also configuration of broker filter: How to configure component filter.

Direct message paths can be created only when direct messaging is enabled on endpoints. For instructions on how to enable direct messaging on endpoint see How to enable direct communication.

Message path message type support's the definition of message type with a wild card character. For message path matching rules description see also chapter Message type matching rules.

New centrally managed message path is saved to the Component Directory Message Path Store. Centrally managed message paths from the Component Directory Message Path Store are accessible through authenticated REST API.

Endpoints periodically run message paths synchronization job with their home Component Directory. The new centrally managed message paths are validated for conflicts with the local existing message. If at least one conflict occurs the new centrally managed message path cannot be stored. For more information please see chapter Message Paths Conflict Resolution. Valid centrally managed message paths are applied to the Endpoint message paths and becomes active Endpoint message paths distributed over ECP network through Component Directory Data (using pushConfiguration REST API).

Centrally managed message path can be deleted in its detail page on Component Directory. Centrally managed message path is deleted for all Endpoints. Change is distributed to all local Endpoints via Component Directory Message Path Store REST API – using same communication patter as for creation of the new Centrally managed message path described above.

6.5.2 Message Paths Conflict Resolution

To prevent message paths conflicts, only Endpoint can push its message paths to the Component Directory. This means that Centrally Managed Message Paths received from Component Directory must be validated by the Endpoint.

- Two message paths (centrally managed / local) are equals when their message type, senders' restrictions, path definition and validity are equals. Such message paths are not considered to be conflicting.
- Two message paths are in conflict when they have same message type and path definition, but senders' restrictions or validity are different.
    - When centrally managed and local message paths are in conflict, local message path takes precedence over centrally managed path.
- Receival of the Centrally managed message paths can be disabled by the Endpoint configuration settings.
- Endpoint uses only its primary (home) Component Directory for centrally managed message paths management.

6.6 Registration keystores with specific component code

By default, any component can send registration request with generic registration keystore downloaded from Component Directory. Component Directory can be configured to create and accept registration keystores with specific component code. In that case, table "Registration Certificates" is shown on the Settings page. Component Directory Administrator can generate new registration certificate with specified component code and date of expiration. New registration certificate is stored in Component Directory repository and added into the table. From the table, registration keystore with the certificate can be downloaded or certificate detail page can be opened. It is possible to delete a certificate from Component Directory repository on certificate detail page, which prevents all components with this certificate to register to Component Directory.

Component code is not mandatory, so it is possible to create a certificate, which can be used by all components for registration. If certificate does not have a component code, then value "NOT_SPECIFIED" is shown in the table. It is not possible to add a certificate with component code already existing in one of the certificates. It is necessary to delete the certificate and then create a new one.

If the generation of the registration keystore with specific component code is enabled, it is not possible to download a generic registration keystore of Component Directory. Generic registration keystore which have been issued before is no longer accepted by Component Directory and cannot be used for registration process.

When Component Directory receives a registration request, a registration certificate from the request is compared with registration certificates stored in the repository of the Component Directory. Registration of the component will proceed only if a valid matching certificate is found and a component code from the request matches the component code from the certificate (only if certificate contains a component code).

Registration using keystores with specific component code can be enabled with the following property in the configuration file:

Property	Description	Default value
ecp.directory.registration.newCertificate.specificCodeEnabled	Enable / disable registration using keystores with specific component code	False

6.7 How to configure synchronization with remote Component Directories

Configuration of synchronization with remote Component Directories is done in GUI of Component Directory – page Components, section Other component directories. In this section, there is a list of remote Component Directories. Local Component Directory will synchronize with these CDs by periodically executed synchronization job. Synchronization interval is configured by the following property:

Property	Description	Default value
ecp.directory.component.directorySynchronizationInterval	Synchronization interval with other component directories. Standard CRON expression.	0 */5 * * * * (every fifth minute)

Component Directory Administrator can also initiate synchronization with all CDs or specified CD manually from GUI. GUI shows if last synchronization attempt was successful.

List of remote Component Directories can be changed by importing text file with records of CDs and synchronization setting (button Import List). Import of a new file will replace current configuration of remote CDs. To make change of a current configuration, export current configuration by clicking on Export List button, make changes in exported file and import it back.

Configuration file is a regular text file with space separated values as defined below (note that first row is header shown only for the sake of clarity - it is not supported by the configuration import). Each line in the file represent one remote Component Directory record.

Unique CD identifier	CD type	Sync mode	CD URLs
CODE	CENTRAL / COMMON	ONE-WAY / TWO-WAY / PASSIVE	URLs

> CODE – unique identification code of the remote CD

> CD type

- CENTRAL federated component directory type, provides content of multiple component directories
- COMMON type of component directory providing its own content

> Sync mode

- TWO-WAY synchronization mode where the configured directory pushes its own content to the remote directory specified by CODE actively together with pulling content of the remote directory. Can be enabled only on one synchronization side.
- ONE-WAY synchronization mode where the configured directory asks remote directory specified the CODE for its content. Can be enabled on both synchronization sides.
- PASSIVE synchronization mode where the configured directory allows the remote directory specified by the CODE to access / modify directory data. This mode has authentication meaning and will be used typically as a counterpart configuration for TWO-WAY or ONE-WAY sync.

Note that component directory synchronized with central CD usually with TWO-WAY sync mode.

Example of the TWO-WAY synchronization configuration is below.

This configuration will be set on component directory **CD1**:

- *CD2 CENTRAL TWO-WAY https://192.168.82.1 https://192.168.82.2*

CD2 setting will be than set as follows:

- *CD1 CENTRAL PASSIVE https://192.168.82.3 https://192.168.82.4*

6.8 Other configuration

Property	Description	Default value
ecp.directory.registration.newCertificate.validity	Validity length of issued certificates in days. Must be non-negative whole number.	365
ecp.directory.client.synchronization.directorySynchronizationInterval	Synchronization interval with other component directories. Standard CRON expression.	0 */5 * * * * (every fifth minute)
ecp.urls	URLs of this component directory. This value is set during the registration process.	No default value

ecp.directory.ttl	Synchronized components lifetime in milliseconds. Must be non-negative whole number.	21600000

6.9 Backup and restore data and configuration

This chapter describes how to backup Component Directory data and configuration. If Component Directory uses external database, refer to instructions from the vendor of the external database system for database backup and restoration.

6.9.1 Windows installation

Backup these files and folders containing Component Directory data and configuration. Copy backed up files to these paths for installation restore.

- <install_path>\authKeystore.jks
- <install_path>\ecp-directory.properties
- <install_path>\ecp-users.properties
- <install_path>\jmxremote.properties
- <install_path>\keystore.jks
- <install_path>\db
    - if embedded database is used

6.9.2 Linux installation

Backup these folders containing Component Directory data and configuration. Copy backed up files to these paths for installation restore.

- /etc/default/ecp-directory
- /etc/ecp-directory
- /var/lib/ecp-directory*

* This corresponds to the value of dataDirectory in ecp-directory.properties. If the data directory is set differently, the backup path would be the one set in the ecp-directory.properties file.

7 ECP Broker Administration

7.1 How to change broker configuration

Broker configuration can be changed manually by editing the broker properties file. See Editing the configuration for instructions how to properly edit the properties file.

For broker configuration update, the following steps should be performed:

1. Locate <broker install path>/broker.properties file
2. Edit broker.properties file content
3. Push broker configuration to ECP component directory
4. Restart broker

For instructions on how to push broker configuration and restart broker see the ECP installation guide.

Note that for broker deployed in HA setup configuration, files located at each broker node must be consistent.

Further, it is worth noting that pushed broker configuration is synchronized through the ECP component directory and thus it takes some time for changes to come into effect.

7.2 How to configure component filter

Components filtering provides broker with the ability to reject messages coming from unauthorized components or reject messages with unsupported message types.

Property	Description	Default value
ecp.broker.filter.components	List of ECP component codes that can communicate through broker, separated by comas. The wild card symbol is supported. See section 5.3.3 Message type matching rules.	N/A
ecp.broker.filter.types	List of message types that can be exchanged through broker, separated by comas. The wild card symbol is supported. See section 5.3.3 Message type matching rules.	N/A

7.3 How to configure NAT

Network address translation (NAT) is used for synchronization with the Component Directory and sending statistics to the Component Directory. Before using NAT, at least one record of the NAT table must be defined. Each network address is compared to the NAT table to search a matching record with the same network code, IP address and port. If a match is found, the IP address and the port is translated.

Property	Description	Default value
ecp.natEnabled	This property enables (true) or disables (false) NAT.	false
ecp.natTable[i].network	Network code of the network address. If not provided, default code is used.	N/A
ecp.natTable[i].sourceIp	Source IP address to match. This property is mandatory.	N/A
ecp.natTable[i].destinationIp	Source IP will be translated to this IP address. This property is mandatory.	N/A
ecp.natTable[i].sourcePort	Source port to match. If not provided, matching will be based only on IP address.	N/A
ecp.natTable[i].destinationPort	Source port will be translated to this port. Must be defined when sourcePort is provided.	N/A

> where *i* represents an index of a record in the NAT table starting from 0.

7.4 How to enable JMX for remote monitoring

ECP4 Broker supports monitoring by using JMX. To activate this function, on ECP4 Broker it is necessary to enable it.

7.4.1 Enabling JMX on Unix system

For enabling JMX on Unix system, follow the steps bellow:

1. Edit file env located in the folder /<broker_install_path>/activemq/bin/
2. Find a row, where parameter ACTIVEMQ_OPTS is defined and add following parameters:
    - -Dcom.sun.management.config.file=/<broker_install_path>/jmxremote.properties
3. Save the file.
4. Create file /<broker_install_path>/jmxremote.properties and add parameters defined at chapter Recommended JMX remote settings.
5. Save the file.
6. Restart ECP4 Broker.
7. After the restart you can connect to ECP4 Broker using JMX.

7.4.2 Enabling JMX on the Windows

For enabling the JMX on Windows, follow the steps below:

1. Edit file wrapper.conf located in the folder C:/<broker_install_path>/activemq/bin/win<32/64>/
2. Find, uncomment and edit the rows mentioned below:
    a. wrapper.java.additional.<PARAMETER_NUMBER>=

    -Dcom.sun.management.config.file=\\<broker_install_path>\\jmxremote.properties
3. Please note, for the PARAMETER_NUMBER it is necessary to use the next number from the previous configuration row for parameter wrapper.java.additional. (Number from the row above, incremented by 1.)
4. Save the file.
5. Create file /<broker_install_path>/jmxremote.properties and add parameters defined at chapter Recommended JMX remote settings.
6. Save the file
7. Restart of ECP4 Broker.
8. After the restart you can connect to ECP4 Broker using JMX.

7.5 Certificates renewal

ECP Broker supports only manual certificates renewal. Certificates must be updated before the expiration of the old ones.

1. Go to the installation folder of the Broker
    a. For example: /opt/ecp-broker/
2. Find the folder with broker-registration-tool.jar, and from this folder execute the commands mentioned below
3. Replace the brackets with proper values and execute the commands mentioned below:
    a. java -D"spring.config.location=<broker_configuraiton_file_path>" -jar broker-registration-tool.jar -m UPDATE_AUTH_CERTIFICATE
        i. Downloads the new Authentication certificate from Component Directory

b. java -D"spring.config.location=<broker_configuraiton_file_path>" -jar broker-registration-tool.jar -m PUSH_CONFIGURATION

i. After a successful renewal of the Authentication certificate, it is necessary to push the changes to the Component Directory

7.6 Web Console configuration

7.6.1 Users management

Users that can access ActiveMQ web console are defined in <broker_install_path>/activemq/conf/jetty-realm.properties file. A new installation comes with two predefined users: admin and user. It is strongly recommended to replace these users with your own or at least change the passwords. Users are stored in the following format:

username: password [, rolename]

7.6.2 Enable HTTPS

It is possible to enable HTTPS on ActiveMQ web console. HTTPS is enabled by uncommenting bean section with id *SecureConnector* in <broker_install_path>/activemq/conf/jetty.xml file. The parameter *keyStorePath* targets to a keystore file. This file must contain either a public or private key. Any trusted keypair can be used including the authentication one present in ECP. An example is shown using built-in authentication keystore:

```
<bean id="SecureConnector" class="org.eclipse.jetty.server.ServerConnector">
        <constructor-arg ref="Server" />
        <constructor-arg>
                <bean id="handlers" class="org.eclipse.jetty.util.ssl.SslContextFactory">
                        <property name="keyStorePath" value="<broker_install_path>/authKeystore.jks" />
                        <property name="keyStorePassword" value="password" />
                </bean>
        </constructor-arg>
        <property name="port" value="8162" />
</bean>
```

A non-secured connector (bean with id *Connector*) must be removed or commented out should you wish to use secured access only.

7.7 Backup and restore data and configuration

This chapter describes how to backup Broker data and configuration. If Broker uses external database, refer to instructions from the vendor of the external database system for database backup and restoration.

7.7.1 Windows installation

Backup these files and folders containing Endpoint data and configuration. Copy backed up files to these paths for installation restore.

- <install_path>\broker.properties
- <install_path>\authKeystore.jks
- <install_path>\registrationKeystore.jks
- <install_path>\cd\
- <install_path>\activemq\conf\ecp-config.xml
- <install_path>\activemq\data\

7.7.2 Linux installation

Backup these files and folders containing Broker data and configuration. Copy backed up files to these paths for installation restore.

- <install_path>/broker.properties
- <install_path>/authKeystore.jks
- <install_path>/registrationKeystore.jks
- <install_path>/cd/
- <install_path>/activemq/conf/ecp-config.xml
- <install_path>/activemq/data/
- <install_path>/activemq/bin/env

7.8 How to enable separate Audit Log file

To enable separate audit log file the ECP_LOG_PROFILES Java property must be set as described below. See also chapter You can use Jasypt tool for passwords encryption. For more information see: http://www.jasypt.org/cli.html.

How to enable separate Audit Log file.

7.8.1 Windows installation

If installed as a service, open command prompt,

- navigate to the <install_path>\tomcat\bin
- execute command tomcat9w.exe //ES//ecp-directory
- navigate to the Java tab
- add -DECP_LOG_PROFILES=ecp-audit to the Java Options
- restart service

Otherwise

- navigate to the <install_path>\tomcat\bin
- edit file setenv.bat
- add -DECP_LOG_PROFILES=ecp-audit to the CATALINA_OPTS
- start/restart component directory

7.8.2 Linux installation

- navigate to the /etc/default/ecp-directory/
- edit file ecp-directory.defaults
- add -DECP_LOG_PROFILES=ecp-audit to the MANDATORY_OPTS
- start/restart component directory

8 Monitoring

ECP provides three ways to check or monitor the status of the component.

> A GUI interface suitable for the human users, who can check the basic health indicators on the dashboard and the messaging statistics pages.

> A JMX (Java Management Extension) interface suitable for monitoring systems like Nagios that can use this Java-specific technology to monitor the health of the component including low-level details such as the memory usage or the number of active threads.

> A Jolokia REST-style API suitable for machine users and monitoring systems that can use HTTP protocol to obtain any information available through the JMX. Jolokia functions as an HTTP bridge for JMX. Please refer to https://jolokia.org/ for more information.

Since each ECP component (endpoint, broker, and component directory) provides different functions, the metrics and information provided in monitoring are described separately for each component.

8.1 Endpoint

The endpoint exposes its monitoring metrics through a dashboard screen in the GUI and through the JMX and Jolokia REST interfaces. The endpoint provides information about its overall status and about the messaging processes.

The table below lists the metrics available through the monitoring.

Group	Name	Description
Connectivity	# connected brokers	Number of brokers to which the endpoint is connected.
	# brokers with broken connection	Number of brokers that have connectivity issues.
Messaging	# sent messages	Number of messages sent through this endpoint by business applications since it was installed.
	# received messages	Number of messages received through this endpoint by BAs since it was installed.
	# waiting to deliver messages	Number of messages waiting to be delivered to the receiver endpoint.
	# waiting to receive messages	Number of messages waiting on the endpoint to be received by a BA.
	# messages in store	Number of messages in the message store.
	# live messages in store	Number of messages in the message store that are not in the final state.
	# messages to be archived	Number of messages in the message store that are in the final state and waiting to be archived.
Component Directory	Endpoint status	Status of the endpoint's registration at home directory.
	Certificates status	Information if the certificate set is valid and complete.
	Closest certificate expiration date	Closest date when a certificate expires.
	Message paths status	Information if all message paths are valid.
	# invalid message paths	Number of invalid message paths.
	Synchronization status	Status of the synchronization with a component directory.
	Version check	Information if there is a newer version installed on the home component directory.

In addition to the information provided by ECP itself, it is also important to monitor the embedded ActiveMQ broker and the external and internal queues used by the endpoint. The way to monitor the embedded broker is the same as for the Message Broker described in section 8.2 Message broker.

If the ECP is configured to use an external database, which is required only for an HA deployment, the monitoring should include the database health as well. This topic is out of ECP scope.

8.2 Message broker

The monitoring of the message broker is limited to the API provided by the AMQP broker, i.e. ActiveMQ. It provides an administration web console with a GUI to the broker administrator and a JMX API as an interface to external monitoring tools. There is no custom interface implemented within ECP. Any tool with JMX support can be used to monitor the broker.

The table below provides a list of some interesting metrics available through the monitoring. The cumulative values are reset on broker restart or failover to passive instance.

Group	Name	Description
Broker	Total # of consumers	Total number of consumers connected to the broker.
	Total # of producers	Total number of producers connected to the broker.
	Total # of connections	Total number of connections opened to the broker.
	Total memory usage (%)	Total percentage of memory consumption.
	Total storage usage (%)	Total percentage of persistent store consumption.
Queues	# of consumers	Number of consumer connected to the queue.
	# of producers	Number of producers connected to the queue.
	# of expired messages	Number of messages that expired waiting on the queue.
	# of written messages	Number of messages written to the queue.
	# of read messages	Number of messages read from the queue.

If the message broker is deployed in a cluster, additional monitoring should be applied to the selected cluster deployment solution (network shared folder, database). This monitoring is out of ECP scope.

8.3 Component directory

The component directory has the same way of providing the monitoring features as the endpoint. Only the set of provided information and metrics is different, since the component directory does not participate in the messaging and its main purpose is to manage all security information.

All the monitoring information available is listed in the table below.

Group	Name	Description
Components	# of waiting approvals	Number of requests waiting for registration approval.
	# of active components	Number of components in active state registered at the component directory.
	# of expired components	Number of components with expired certificates.
	Synchronization status	Status of the synchronization with other component directories.
Certificates	Certificates status	Information if the certificate set is valid and complete.
	Closest certificate expiration date	Closest date when any certificate expires.

If the ECP is configured to use an external database, which is required only for the HA deployment, the monitoring should include the database health as well. This topic is out of ECP scope.