EMDIS 4.0

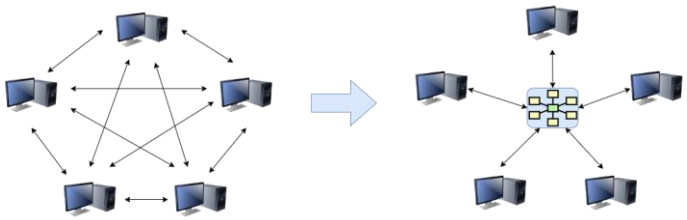
Version 1.0

2020-07-23

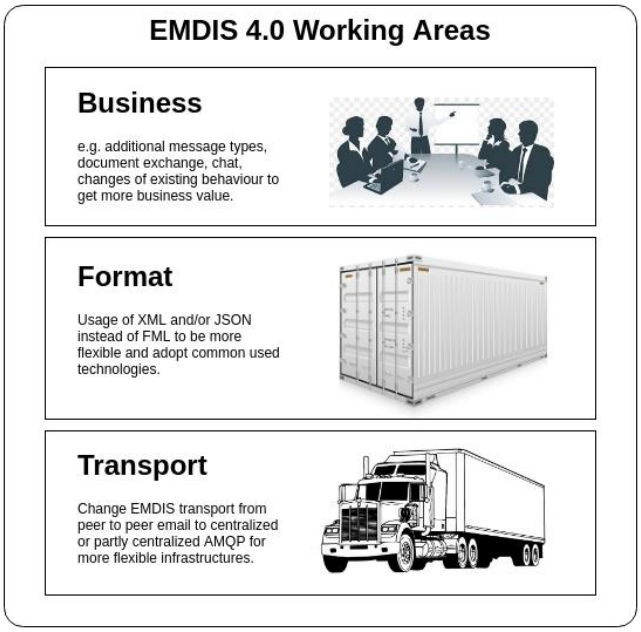
Table of Contents

1. Introduction

This document describes the infrastructure and the technical details of EMDIS 4.0. This is a major upgrade of EMDIS 3 in that it updates the technology stack and changes some of the existing techniques and workflows. Starting with EMDIS 4.0 EMDIS will transition from a peer to peer network to a centralized infrastructure as shown in the figure below.



Older Versions of EMDIS use e-mail as transport layer for messages. EMDIS 4.0 changes the transport to AMQP 1.0 based messaging, introduces new messages and creates the foundation for further changes, e.g. the transition from FML to XML based messages. The working areas of EMDIS 4.0 can be seen in three areas: transport, format and business.



While EMDIS 4.0 introduces a whole new technology stack, it also tries to make the transition from older versions as easy as possible.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", “SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (https://tools.ietf.org/html/rfc2119).

2. AMQP Primer

The Advanced Message Queuing Protocol (AMQP, www.amqp.org) is a standardized, open message protocol for communication between applications. The origins of the protocol are in the financial sector, but nowadays it is also used for general machine-to-machine communication (M2M) and the Internet of Things (IoT).

AMQP 1.0 is a wire level binary protocol and has support for direct and broker based communication. Being defined on the wire level (instead of API, like JMS) guarantees platform independence.

The protocol has support for different degrees of reliability for messages ("at most once", "at least once" and "exactly once"), grouping of messages that logically belong together and even transactions. Brokers are used to queue and route messages, and keep messages in store if clients are temporarily not available.

AMQP uses a dedicated TCP connection on port 5671 (direct TLS) or 5672 (plain TCP with upgrade to TLS, which is used by most implementations).

There are different versions of AMQP existing. This document always refers to AMQP 1.0. This is important to notice as AMQP 1.0 is not compatible to older, still popular versions of the protocol, e.g. AMQP 0-9-1.

Brokers implement different concepts and use the following nomenclature:

- **Broker**: Server that is responsible for receiving and temporarily store messages on behalf of receivers. A broker decouples sender and receiver.
- **Queue**: Store and send messages to consumers (subscribed readers, see below). Brokers guarantee strict ordering of messages within queues.
- **Producer**: publishes messages to queues or exchanges.
- **Consumer**: Consumers subscribe and "listen" to queues and receive messages from them.

3. Basic Concepts

Main discussion points about older versions of EMDIS are:

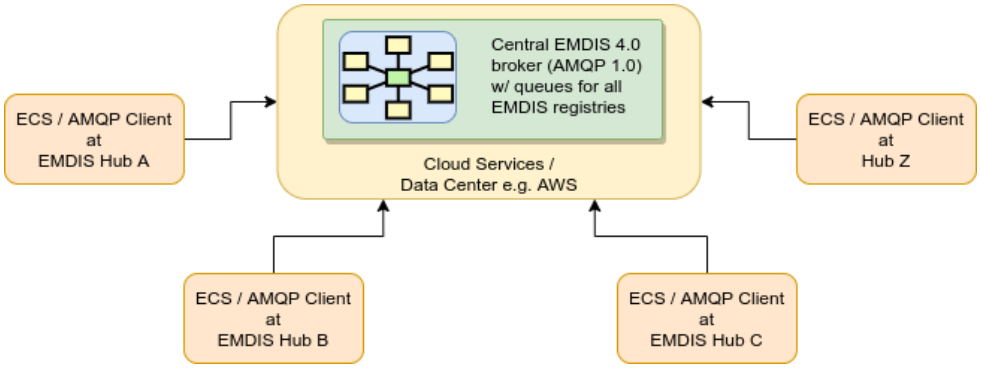
- E-mail transport is hard to configure and seems to be outdated for B2B messaging.
- FML is not an industry standard format and may be tricky to parse.
- Peer-to-peer connections between registries are complex and it is hard to join that network - especially for emerging registries.

Among others, EMDIS 4.0 tries to address these issues or at least set the starting points for further changes in the future. While introducing changes, another goal of the introduction of EMDIS 4.0 is to make a smooth transition possible for already participating registries.

Therefore, EMDIS 4.0 follows the following ideas and concepts:

1. Use standardized, open and freely available technologies and tools. This enables all registries to implement EMDIS 4.0 without additional costs. Open technologies most often also have a lot of freely available documentation ready.

2. Usage of AMQP 1.0. AMQP 1.0 is a standardized, open wire protocol, which makes the protocol independent from programming languages and platforms.
3. Have a simple infrastructure (central server), but also be open for extension and additional systems.
4. Introduce a central infrastructure instead of running a peer-to-peer network. Connecting to a central system reduces the complexity of the whole network and makes it easier for registries to join.



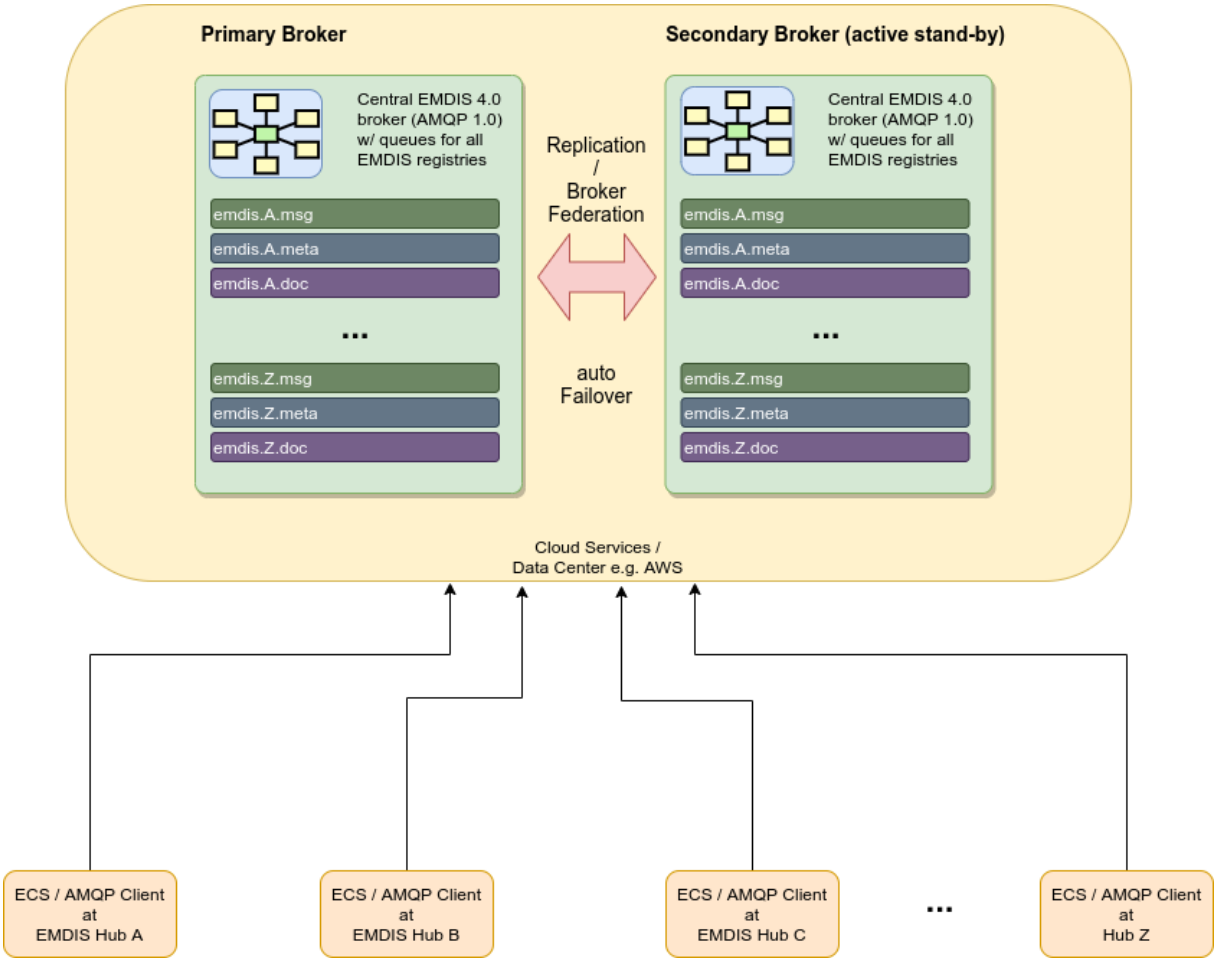
5. Create a basic infrastructure, which is open to change and is able to introduce more changes without changing the whole infrastructure again and again.
6. Keep infrastructure and logic on the central server as simple as possible to avoid complex and hard to handle configurations.
7. Renew transport, functionality and message format carefully, so that existing implementations can smoothly transition to EMDIS 4.0.
8. Use tools that are open and that can easily be replaced by other products. Especially EMDIS 4.0 tries not to use or rely on vendor specific additions or features to avoid vendor lock-ins.

4. Infrastructure

EMDIS relies on a central broker that SHOULD be hosted in the cloud. To reduce the possibility of failure it is RECOMMENDED to run the central broker in a clustered environment either by running additional and independent servers or by having a hot stand-by server. It is also RECOMMENDED to run the failover server on a different location or data center.

The central broker MUST be available 24x7 with a maximum downtime of 12 hours. The server MUST have the ability to store messages to avoid message loss on reboots or restarts of the service.

The basic infrastructure MAY consist of one server including a fail over solution. It also SHOULD be possible for registries to run their own brokers and connect them to the central server. Scenarios for such infrastructures are described in Appendix A.



Participating registries MUST have an account on the central broker to use the service. To connect to AMQP registries clients MUST be able to access the AMQP port 5671 on the central broker.

5. Administration

Using a central server needs a central administration for broker resources, users, permissions and queues. Running the broker in the cloud also needs a legal representation from EMDIS to rent, pay and administer the common infrastructure. It is RECOMMENDED that the cloud contracts are owned by the EMDIS community. The ownership requires to have access to the account, pay bills invoiced by the cloud provider and collect fees from the community members to run the service.

Additionally it is RECOMMENDED that software administration is delegated to a service provider with good EMDIS understanding. EMDIS technical chairs SHOULD also have access to the technical environment.

Typical tasks for administration are:

- Oversee the usage of resources and recommend resizing the servers if needed.
- Help registries to join the EMDIS 4.0 network.
- Administer user accounts used by the registries to connect the broker.
- Support registries with broker specific problems and connection issues.
- Configure and watch queues on behalf of the registries.
- Approach and help registries that seem to have issues and full queues.
- Maintain administration scripts and helpers.
- React and solve broker issues especially in the case of failover.
- Manage broker federation / inter-broker connections.

The need for a central administration is new to EMDIS 4.0 and can be seen as a major paradigm shift. EMDIS 3 is working with independent peer to peer connections that can be individually handled, whereas EMDIS 4.0 requires an administrated central communication node.

6. Broker

The broker MUST support the AMQP 1.0 protocol. The service also MUST have a static IP or at least a static DNS entry.

The broker MUST allow each participating registry to access the service with username and password. The credentials will be supplied to all participating registries. There SHOULD be exactly one account per hub. The broker also MAY supply additional authentication methods, such as certificate based authentication.

The broker MUST accept encrypted connections (TLS) on TCP port 5671. Brokers MAY also accept connections on TCP port 5672 with optional TLS encryption, although it is strongly RECOMMENDED to only accept secure connections.

The broker MAY allow for other protocols than AMQP 1.0 as long as messages are converted to regular AMQP 1.0 messages for all other participants. Especially the broker provider MAY additionally offer a WebSockets or web service API e.g. as a fallback for registries that can't connect to AMQP 1.0 services due to local firewall restrictions. Fallback scenarios using WebSockets or web services have not been tested in the Pilot phase and may also imply further developments.

It is RECOMMENDED to directly use the AMQP 1.0 protocol whenever possible to benefit from all AMQP 1.0 features – performance in particular.

The broker MUST NOT allow for creation of non-EMDIS related or temporary queue creation or usage.

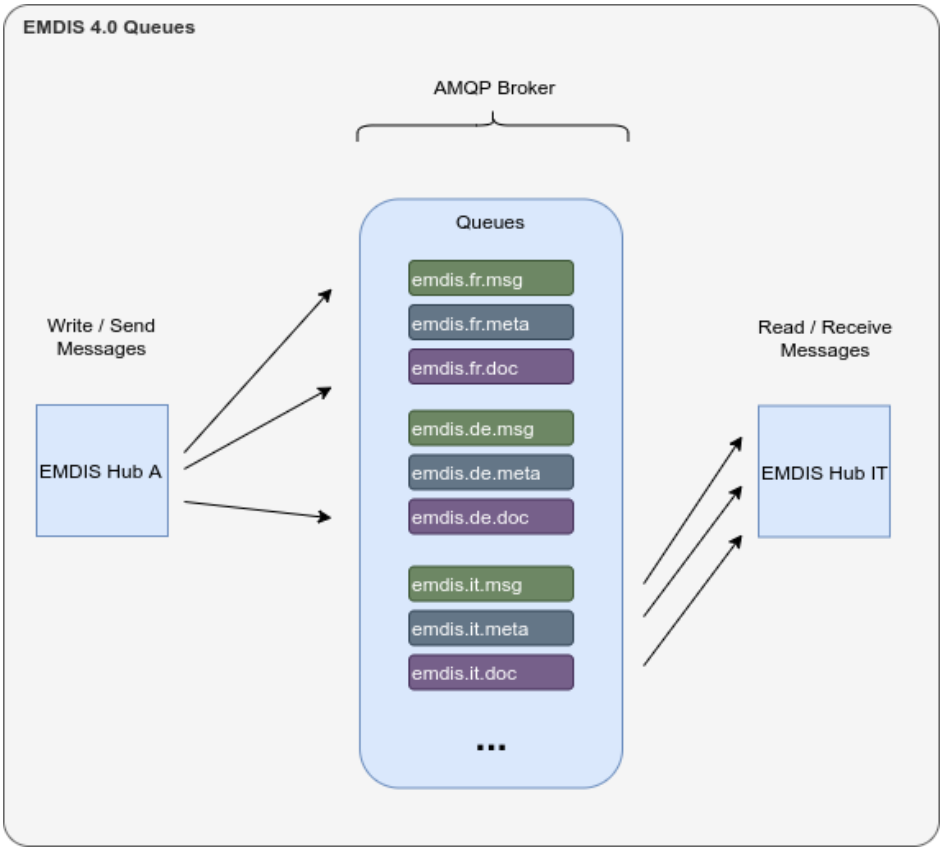
The broker MUST reject undeliverable messages that e.g. are sent to an unknown queue.

7. Queues

EMDIS 4.0 makes use of three queues per hub. The queues MUST be named

```
emdis.<hub code>.msg
emdis.<hub code>.doc
emdis.<hub code>.meta
```

In order to make it easier to reject undeliverable messages it is RECOMMENDED that the broker instance only host the EMDIS relevant queues. If the broker hosts other queues as well these MUST use different credentials and permissions to avoid hard to track errors.



It is RECOMMENDED to pre-create these queues for all participating hubs in advance and to make them persistent.

Queues MUST be writable for all EMDIS partners of a hub. It is RECOMMENDED to make all queues writable for all EMDIS 4.0 participants. The broker MAY implement a more advanced permission set that only allows sending messages to valid partners according to the EMDIS partnership matrix.

The broker MUST reject undeliverable messages (e.g. messages sent to a non-existing queue). Messages sent to wrong queues (e.g. if there is no relationship between the sending and receiving registry) MAY be rejected or dropped by the

receiving side. It is RECOMMENDED to approach the administrator of the sending side in such cases.

On the other hand, queues MUST only be readable for the hub the queues belong to. All other hubs MUST NOT be able to consume messages from queues that are not their own.

8. EMDIS Messages

The payload of regular messages SHOULD be PGP/GPG encrypted with the existing EMDIS keys. Meta messages are not encrypted, but they MUST be signed instead. Encryption details are handled between two EMDIS partners.

The content MUST stick to the EMDIS semantics and format. It is RECOMMENDED to send single AMQP messages per one EMDIS message.

Messages MUST stick to the defined message headers for EMDIS 4.0 messages. Messages that do not support headers correctly MAY be rejected by the recipient.

Regular EMDIS messages MUST be sent to the `emdis.<recipient hub>.msg` queue.

Documents that are sent with EMDIS 4.0 document exchange MUST be sent to the `emdis.<recipient hub>.doc` queue.

The `emdis.<hub>.meta` queues are only for EMDIS meta and admin messages.

Messages in wrong queues MAY be rejected or dropped by the recipient. It is RECOMMENDED to send messages as soon as they are available on the source system. There is no need to collect messages and send them in batch.

Messages MUST be flagged “durable” (see section “Message Definition and Headers”). This is a per message setting and is needed to have messages stored on disk on the broker until the client consumes messages. Messages that are not marked durable will not be available after a reboot of the broker.

On the consuming side, it is RECOMMENDED to process messages as soon as they are available. Messages SHOULD NOT be left on the broker for more than six hours to save disk space on the broker. Messages MUST be acknowledged on the receiving side to remove them from the queue.

It is RECOMMENDED to acknowledge messages only after they are safely stored on the receiving system.

Messages SHOULD be read from only one system, even if there are more brokers available (e.g. for redundancy reasons). The order of delivery CAN only be guaranteed, when reading from one system. It SHOULD be safe to send messages to one broker and read them from another in multi-broker environments. To keep things simple it is RECOMMENDED to only use one broker at a time, though.

9. Message Definition and Headers

The following table shows the defined structure of EMDIS messages by making use of the AMQP 1.0 message structure. The sections “header”, “message annotations”, “application properties” and “message body” are given by the AMQP 1.0 specification. The application properties allow for setting custom properties whereas

the properties in the header and message annotation sections are given by the protocol.

The definition below follows the idea to separate message content (payload) from the meta data. The latter MUST NOT be encrypted and MAY be used for routing or interpretation of the actual payload. The payload is encrypted with PGP as it is today. For transition it is RECOMMENDED to reuse the existing EMDIS keys.

Although the AMQP 1.0 transport is typically encrypted by using TLS, the messages in the queues on the broker would be readable. The sender MUST handle encryption before the messages are published because AMQP 1.0 itself does not offer payload encryption.

EMDIS messages are still in FML format for the existing message types. Exception to that rule is document exchange, which is a new feature and already uses XML as message format. It is RECOMMENDED to transition all messages from FML to XML in the upcoming releases.

EMDIS Message Structure		
Property	**Value/Examples**	**Remark**
Header		
Durable	`True`	Fixed value to indicate that messages must not be lost. IMPORTANT: This MUST be set per message in most AMQP 1.0 clients.
Message Annotations		
Subject	`EMDIS:<hub_snd>:<seqnr>` e.g. EMDIS:FR:1234	For regular EMDIS messages
	`EMDIS:<hub_snd>:<seqnr>:<partnr>/<totalparts>` e.g. EMDIS:FR:1234:3/5	For EMDIS messages sent in parts, e.g. in EMDIScord
	`EMDIS:<hub_snd>` e.g. EMDIS:FR	For META messages
	`EMDIS:<hub_snd>:<seqnr>:DOC` e.g. EMDIS:DE:5896:DOC	For document exchange

Application Properties		
x-emdis-hub-rcv	e.g. “DE”	EMDIS receiving hub code
x-emdis-hub-snd	e.g. “FR”	EMDIS sending hub code
x-emdis-message-version	e.g. “E3.16”, “E3.17”	EMDIS version.
Message Body		
		Contains the PGP/GPG payload depending on the above properties. The body contains a single EMDIS message in FML or for documents in XML format. The payload is encrypted except for META messages.

10. Document Exchange

The document exchange via AMQP 1.0 is one new feature introduced with the switch from emails to AMQP 1.0 messaging. Just like EMDIS FML, messages with documents MUST have a sequence number. However, instead of having only one counter it was decided to use two different counters: one for FML messages and one for Document Exchange. In this way it is possible, in case of trouble (e.g. with attachment size), to continue the normal message flow with FML messages.

For the document exchange, every EMDIS registry has a separate queue on the Broker. These queues are named:

`emdis.<hub_rcv>.doc`

E.g.: `emdis.de.doc`

If the document queue is cleared by an ECS implementation, ECS is responsible for the decryption of the payload. The receiving side is responsible for the base64 decoding and the processing of the transmitted document.

Although a dedicated sequence number for Document Exchange is in use, a separate queue on the broker for each registry to handle Meta messages related to Document Exchange is not needed.

For this reason two new Meta messages were defined (DOC_MSG_ACK, DOC_RE_SEND) and the meaning of existing one (READY) was improved.

Documents SHOULD NOT exceed the limit of 20 MB in size. Message parts MUST NOT be used for document exchange.

10.1. Using XML instead of FML

The following XML structure was defined for the Document Exchange:

Element	Value/examples	Remark
<MESSAGE>		Root element
<REG_SND>	ION e.g. <reg_snd>1804</reg_snd>	Required.
<REG_RCV>	ION e.g. <reg_rcv>6939</reg_rcv>	Required.
<P_ID	EMDIS patient ID	Optional.
<D_GRID>	GRID for donors	Optional.
<CB_ID>	CB_ID for CBUs	Optional.
<CREATION_DATE>	e.g. 2019-04-27T17:23:45.117Z	Required. Date when the message was created on the sending side.
<TEXT>	e.g. <TEXT> "Hi, here is the form..."</TEXT>	Optional. Additional text for the attachment (255 characters)
<THREAD_ID>	e.g. a reference code, a patient ID, ... e.g. <THREAD_ID>123475</THREAD_ID>	Optional. ID to group several attachments together (same ID shows that several attachments belong together).
<ATTACHMENT>		Required. Single attachment

Element	Value/examples	Remark
<BODY>		Base64 encoded document body
<CATEGORY>	e.g. <CATEGORY> WOR </CATEGORY>	Optional. ACC = Accounting ACT = Activation IDM = IDM Request/Result related OTH = Other SMP = Sample Request/Result related TYP = Typing Request/result related WOR = Workup
<FILENAME>	e.g. <FILENAME> attachmet01.txt </FILENAME>	Required. Filename of the attachment. (Only one attachment per single AMQP message allowed).

<FILETYPE>	e.g.
	<FILETYPE>
	application/pdf
	</FILETYPE>
		Required.
		
		Mime type of attachment[1].
		Valid are PDF, office documents and images.
		
		e.g.
		`application/msword`
		`application/vnd.openxmlformats-officedocument.wordprocessingml.document`
		`application/pdf`
		`application/vnd.ms-powerpoint`
		`application/vnd.openxmlformats-officedocument.presentationml.presentation`
		`application/vnd.ms-excel`
		`application/vnd.openxmlformats-officedocument.spreadsheetml.sheet`
		
		1 List of valid mime types: https://www.iana.org/assignments/media-types/media-types.xhtml

10.2. Example document exchange message

Below an example is given of a message body for a document exchange message:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MESSAGE>
    <REG_SND>7450</REG_SND>
    <REG_RCV>6939</REG_RCV>
    <P_ID>IT01ABCD</P_ID>
    <D_GRID>74507860941580905l9</D_GRID>
    <CREATION_DATE>2019-04-27T17:23:45.117Z</CREATION_DATE>
    <TEXT>Hello message</TEXT>
    <THREAD_ID>123456</THREAD_ID>
    <ATTACHMENT>
        <BODY>QXR0YWNobWVudCAx</BODY>
        <CATEGORY>ACC</CATEGORY>
        <FILENAME>attachmet01.pdf</FILENAME>
        <FILETYPE>application/pdf</FILETYPE>
    </ATTACHMENT>
</MESSAGE>
```

10.3. Best Practices for Document Exchange

Large documents may slow down the communication for all hubs. Messages – including documents - have to be temporarily stored on the broker and therefore require a considerable amount of storage space. Therefore it is recommended to consider the following guidelines for exchanging documents:

- Avoid scanning when possible.
- Scanned documents are not searchable and are considerably larger in size than documents electronically converted (e.g. in PDF format).
- Convert electronic documents directly to PDF instead of printing then scanning
- If you need to scan, use correct scanner settings.
- With the proper settings a 10 MB PDF document can contain over 100 scanned pages. Use resolution of max 300 dpi and a profile for Black and White (do not use not greyscale or color profile).
- If the attachment size exceed 20 MB the receiving hub could accept (raising a Warning message) or refuse (raising a message denial) the document exchange message.
- This limit (very high) is necessary to avoid a slow down for all the other hubs connected to the broker where the message is stored.

10.4. META messages

For the Document Exchange two new Meta messages were introduced: DOC_MSG_ACK and DOC_RE_SEND.

These Meta messages for Document Exchange work in the same way as those for regular FML messages. (MSG_ACK and RE_SEND).

The READY Meta message was improved to also report last received and last sent document sequence number e.g.:

```
READY
msg_type=READY
last_recv_num=XXX
last_sent_num=XXX
last_recv_doc=XXX
last_sent_doc=XXX
# Hello partner, I am alive.
```

For further details see EMDIS Communication System – ECS > v. 2.07 (2015-07-10) (coming soon).

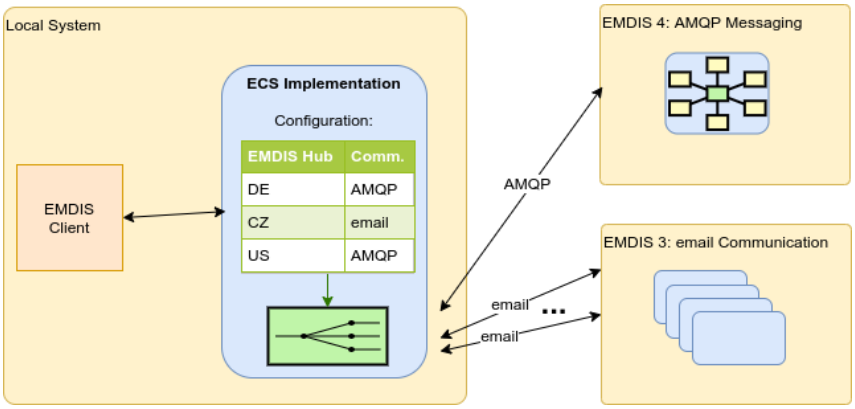
11. Transition to EMDIS 4.0

One of the design goals of EMDIS 4.0 was to allow for a smooth transition with upgraded, but already existing ECS implementation. That is also the reason why the EMDIS message flow including the meta messages has not been changed. Therefore the transition to EMDIS 4.0 requires an updated ECS system (ESTER, perlECS or other local ECS implementations). As a prerequisite all current ECS implementations must be upgraded to support EMDIS 4.0 connectivity via AMQP 1.0.

Future versions of ESTER and perlECS will be able to deal with EMDIS 4.0 hubs by configuring the connection type for each partner individually. Options are e-mail for EMDIS 3 connections and AMQP 1.0 for EMDIS 4.0 connections.

After a switch to AMQP ECS will handle messages and the communication without further changes.

New features like the document exchange still have to be implemented on the processing side, as ECS only takes care of the transport of the documents.



The transition to EMDIS 4.0 will take a while until all EMDIS partners have switched to the new AMQP infrastructure. This means that there will be a transition phase where both versions of EMDIS will be active. It is RECOMMENDED that the duration of the transition phase will be organized with an official transition plan by the EMDIS community.

Appendix A: Extending the Network of Brokers

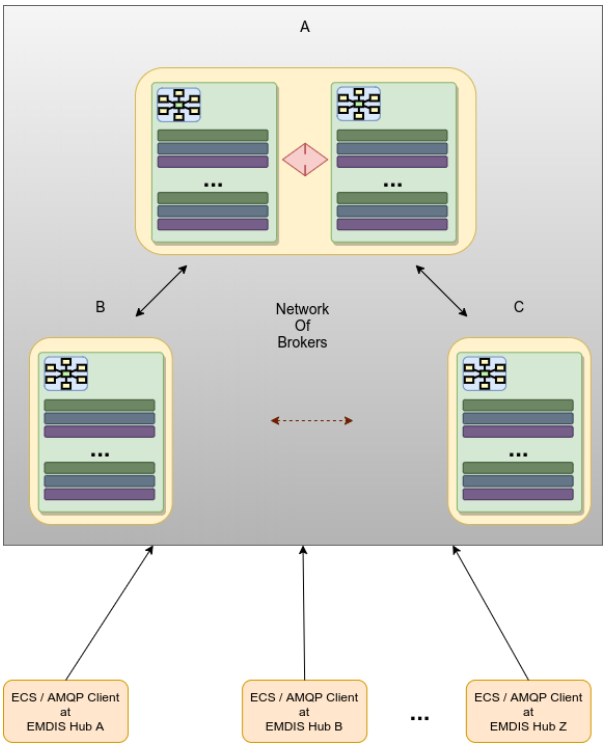
EMDIS 4.0 will use a central infrastructure consisting of a single broker entity which all EMDIS 4.0 registries will connect to. It is technically possible to extend that single broker with additional brokers if needed. The infrastructure and the broker software MUST be able to allow for extension. Still, it is also RECOMMENDED to keep the infrastructure as simple as possible to avoid unwanted complexity or make it hard to track down errors.

Additional scenarios for extension might also be as described in the following sections:

- Extending the network of brokers for more redundancy by using broker federation with the central broker.
- Running a local broker to hide AMQP clients behind the corporate firewall for local connection aggregation or security reasons.
- Connection aggregation for service providers that run multiple hubs as some kind of proxy on transport level.

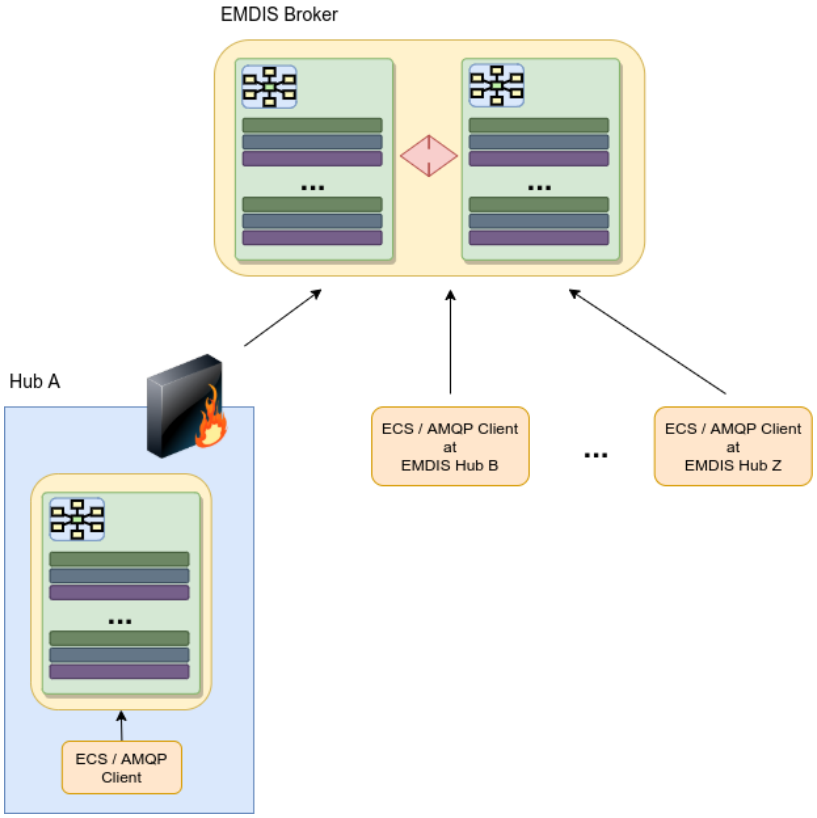
Extending the network of brokers for more redundancy

A central node also comes with the risk of also being a single point of failure, although it is RECOMMENDED to at least run one active stand by server for redundancy reasons. If the single-broker-solution still needs additional redundancy, further brokers CAN be added. This increases resilience on the one hand but also increases complexity on the other hand.



The figure shows a network of brokers. Brokers B and C are extending the central EMDIS 4.0 server A. They are connected to A and do message replication. It is also possible to connect B and C to have a meshed network.

If full replication is in place, clients can connect to either broker. It is strictly RECOMMENDED to stick to one connection within one session to guarantee the order of messages.



Running a local broker to hide AMQP clients behind a corporate firewall

Registries may want to run their own brokers in their networks to aggregate connections or to have one single node connecting to the internet and the EMDIS broker respectively.

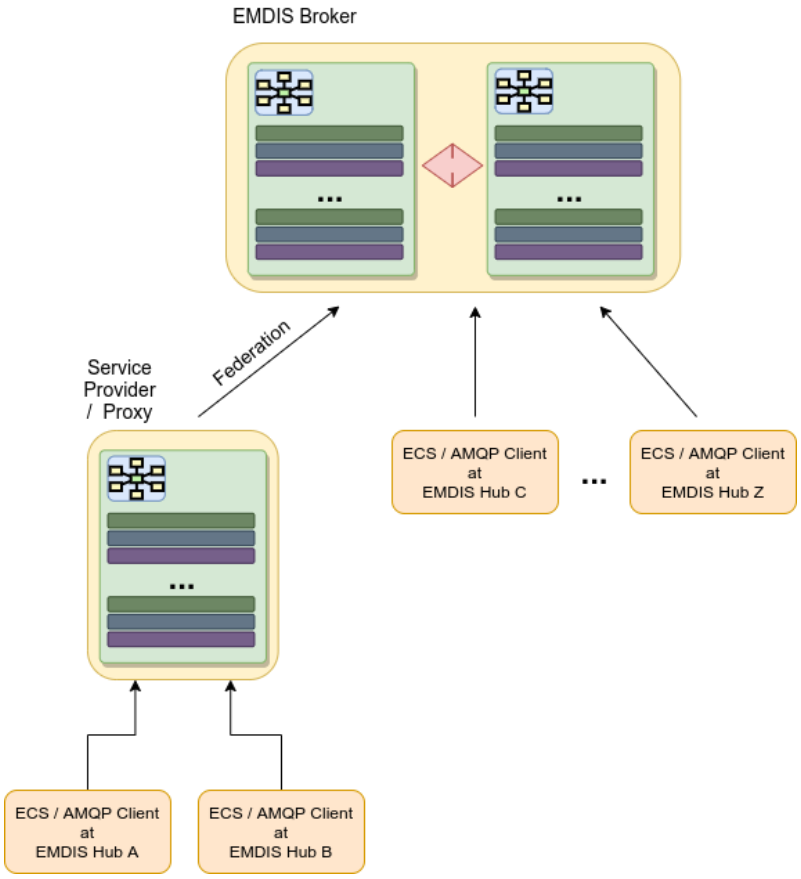
The figure shows Hub A having its own local broker that connects on behalf of the internal EMDIS clients.

Connection aggregation for service providers that run multiple hubs

Service providers that offer proxies for other registries may also want to aggregate the connections of their users and run only a single connection to the central EMDIS 4.0 broker. Instead of running a local broker it might be easier to connect clients to the central infrastructure instead.

As a full connection including replication from a local server to the central infrastructure also includes risks for the central EMDIS server because of the

increased complexity, the administrators of the central infrastructure CAN reject such requests and recommend direct connections instead.



The figure above shows an additional broker that aggregates connections from Hub A and B and connects to the central EMDIS broker.

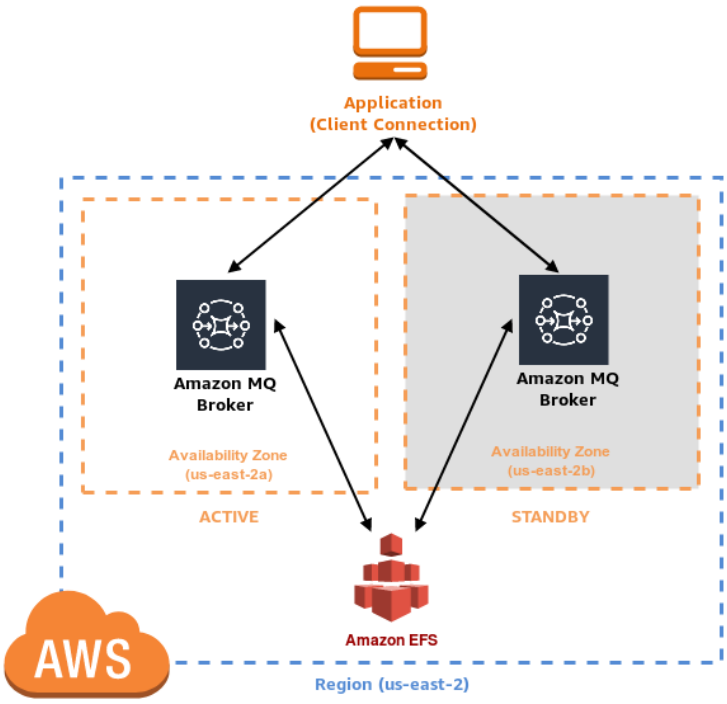
Appendix B: Implementations: AmazonMQ and ActiveMQ

After testing several brokers and providers it is RECOMMENDED to implement EMDIS 4.0 by using Amazon AWS. The managed service on AWS is called AmazonMQ and is based on the Open Source software Apache ActiveMQ. The features described in this document can be achieved by using AmazonMQ and ActiveMQ respectively.

It is RECOMMENDED to implement the active stand-by version of AmazonMQ to have good reliability and a secure environment.

Local brokers MUST run on Apache ActiveMQ “Classic” if replication is needed. It is also RECOMMENDED to use ActiveMQ as most parts of the configurations can be shared between ActiveMQ and AmazonMQ brokers. ActiveMQ is available for free on http://activemq.apache.org/components/classic/download/.

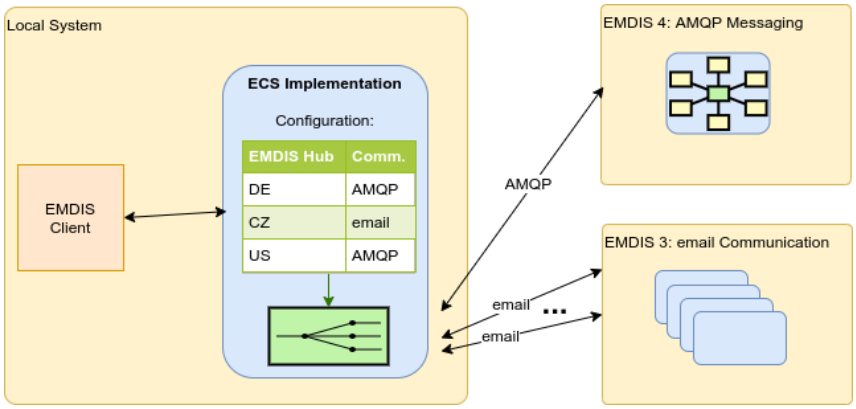
As an alternative registries can also run their own AmazonMQ instances instead of ActiveMQ and connect them to the central servers.



Extension of the broker by building a network of brokers can be achieved by using ActiveMQ broker federation. More details on AmazonMQ may be found on the website https://docs.aws.amazon.com/amazon-mq/latest/developer-guide/welcome.html

Hints on how to build local installation based on ActiveMQ may be found on the ActiveMQ website http://activemq.apache.org/components/classic/

Broker federation / replication is broker dependent and work only between ActiveMQ/ AmazonMQ brokers within the same major version. To connect to those servers any AMQP 1.0 compliant client or library will work.



Appendix C: Proxied Connections / EMDIS Proxy

Proxied connections will still be possible. In a proxied connection the three queues `emdis.<proxy id>.msg`, `emdis.<proxy id>.meta` and `emdis.<proxy id>.doc` will be used for more than one hub. It is the responsibility of the proxy provider to manage the messages correctly on behalf of the participating hubs.

That means that the proxy provider sends the messages to the destination queue and distributes incoming messages from the proxied queues to the final destination.

The broker does not have any specific logic or processing for proxies.

Revision History

Date	Version	Author	Remark
2020-06-03	0.1	Andrea Timm	• Initial draft with document structure, headings, … • Copied some first information from different places. • Included documentation about Document exchange • Removed obsolete message properties
2020-06-08	0.2	Andrea Timm	• Changes in Structure • Incorporated texts of Daniel Freund. • Added approval mark (to be removed after approval of the members of the EMDIS 4.0 pilot group)
2020-06-17	0.3	Andrea Timm	• Changes after Pilot Group Call: corrected typos, removed some more obsolete message annotations.
2020-06-26	0.4	Daniel Freund	• Added remarks from Marco Vitale • Added sections „Administration“ and Appendix B, C and D • Added more details and figures
2020-07-10	0.9	Daniel Freund, Andrea Timm	• Minor corrections and addtions • Preparation for approval and final version
2020-07-17	0.9.1	Daniel Freund	• Added feedback from Marco Vitale and Mario Gran
2020-07-17	0.9.2	Marco Vitale	• Added section about best practices for document exchange

Date	Version	Author	Remark
2020-07-23	1.0	Andrea Timm, Daniel Freund	• Changes after Pilot Group Call • Moved version to 1.0 after final approval from the members of the EMDIS 4.0 pilot group: Amar Baouz (FR), Steve Finch (US), Daniel Freund (DE), Mario Gran (ES), Joel Schneider (US), David Steiner (Steiner Company), Andrea Timm (DE), Marco Vitale (IT), Dan Vaněček (Steiner Company)